МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна магістерська робота

на тему:

«Інформаційна технологія оптичного розпізнавання тексту з зображення для мобільного додатку»

Завідувач кафедрою **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студент групи ІН м.-02 **Погибелєва Л. В.**

СУМИ 2021

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *122 «Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Погибелєвій Людмилі Віталіївні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія оптичного розпізнавання тексту з зображення для мобільного додатку.*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналітичний огляд та постановка задач 2) Вибір алгоритму 3) Інформаційне та програмне забезпечення системи*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми та постановка задачі*		
2.	*Вибір алгоритму*		
3.	*Інформаційне та програмне забезпечення системи*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 55 стор., 20 рис., 2 додатка, 29 джерел.

Об’єкт дослідження — процес розпізнавання друкованих символів

Мета роботи — розробка інформаційної технології оптичного розпізнавання тексту з зображення для мобільного додатку

Методи дослідження — методи обробки зображення, методи розпізнавання образів.

Результати — розроблено інформаційне, алгоритмічне та програмне забезпечення системи інтелектуального аналізу даних з зображень з метою виявлення розпізнавання тексту. В роботі використовується комплекс алгоритмів розпізнавання символів. Розроблений алгоритм реалізовано мовою програмування JS.

СИСТЕМА РОЗПІЗНАВАННЯ ОБРАЗІВ, РОЗПІЗНАВАННЯ СИМВОЛІВ, РОЗПІЗНАВАННЯ ТЕКСТУ

ЗМІСТ

ВСТУП

Обчислювальні системи існують вже десятки років. Метою їх створення була можливість замінити людину, виконати трудомістку роботу, що вимагає складних розрахунків. Одне з таких завдань — як навчити комп’ютер розпізнавати зображення (зокрема, який алгоритм потрібно створити, щоб комп’ютер міг приймати зображення з текстів).

Розпізнавання тексту є дуже складним завданням з теоретичної та практичної точки зору. Людина, наприклад, використовує весь спектр знань і досвіду. Він визначає текст із набору смислових сигналів, ідентифікує кожен символ, визначає характеристики символів і на основі свого досвіду приходить до висновку про значення символу та тексту загалом.

Комп’ютер набагато частіше робить помилки в процесі розпізнавання. На сьогоднішній день не існує абсолютно точного методу визначення тексту та символу за його зображенням. Багато комерційно розроблених проектів використовують свої власні методи і навіть вони не можуть похвалитися ідеальним рішенням.

У багатьох випадках, продукт пропонує комплексний підхід до проблеми. Завдання розпізнавання тексту поділено на підзадачі: відфільтрувати зображення від шуму, видалити не текстові зображення із текстових, вибрати ознаки символів та порівняти ці ознаки із збереженими зразками. Кожне завдання містить багато рішень, лише деякі з яких є більш-менш оптимальними.

1 АНАЛІТИЧНИЙ ОГЛЯД ТА ПОСТАНОВКА ЗАДАЧІ

Цей розділ спрямований на огляд концепцій, що лежать в основі технологій обробки зображень, а також надання деякої інформації про вибір методів та засобів, які аналізувалися і застосовувалися в роботі. Перші підрозділи присвячено проблемі виявлення та розпізнавання текстових символів.

1.1 Опис алгоритмів розпізнавання

1.1.1 Граф лінійного розподілу

Почнемо з графа лінійного поділу [22]. Нехай у нас є чорно-біле зображення рядка тексту. Ідея полягає в наступному – шукаємо вертикальні білі просвіти та далі кластеризуємо їх за шириною: великі – це прогалини, маленькі – це відстань між літерами. Ідея хороша, але ширина прогалин не є однозначним показником, наприклад, для тексту з нахилом або специфічного поєднання символів.

В такому випадку існує два рішення.

Перше – ввести якусь «видиму» ширину просвітів. Тільки це рішення працює не завжди добре, так як текс може бути пошкоджений при скануванні. Друге рішення – це графи лінійного поділу. Ідея полягає в тому, щоб відзначити всі можливі точки розподілу. Частина зображення між двома крапками і буде буквою. Якщо якість і чіткість зображення хороша, то проблем з визначенням точок не буде.

1.1.1.1 Гіпотеза зображення

Перед тим, як розпочати розпізнавання окремих слів, нам слід розібрати ще одну тему - гіпотези зображення фрагмента.

Нам дано зображення тексту, який ми оброблятимемо. Звичайно, хочеться того, щоб будь-які зображення оброблялися однаковим способом, але, оскільки всі зображення різні, вони можуть мати різну якість і отримані різними способами.

Складається думка, що різноманітність різних можливих дефектів (спотворень) буде дуже великим, але якщо розібратися, то ми знайдемо тільки певний комплект можливих спотворень. Для цього ми використовуємо систему гіпотез тексту[20].

Обробка гіпотез є певним алгоритмом:

- За зображенням генерується найбільш підходяща гіпотеза.
- Виправляються викривлення від обраної гіпотези
- Розпізнання отриманого зображення
- Оцінка якості розпізнавання

Порівняння оброблених даних із вихідними з метою виявлення помилки при застосуванні будь-якої гіпотези.

1.1.1.2 Оцінка якості слова

Залишилися ще два важливі моменти: оцінка якості розпізнавання слова та символів. Спочатку розберемо оцінку якості розпізнавання слова.

На цьому етапі задіяна ідея моделей. Моделлю слова є спільний опис конкретного типу слів у мові. Існують моделі стандартних слів у мові, моделі чисел, дат, скорочень, абревіатур тощо.

Модель повинна вміти швидко визначати, чи підходить для неї варіант слова або ні[24]. Звичайна перевірка, як правило, містить у собі всі перевірки дозволених наборів символів для кожної літери у слові. Також модель вміє оцінювати якість слова, що розпізнається.

При оцінюванні якості моделей не слід забувати про те, що наше завдання полягає у порівнянні моделей, тому їх оцінка має бути узгодженою.

1.1.1.3Розпізнавання символів

Наступним етапом є формування гіпотез (тобто визначення того, що це за символ) і, надавши оцінку ймовірності, передавання на вхід механізму розпізнавання символів. Але насправді ми розпізнаємо не символи, а графеми.

Графеми – це уявлення символу у так званому графічному вигляді. Після складання гіпотез, у тому які це знаки, йде словникова перевірка. ABBYY [25] застосовує такі типи класифікаторів:

- растровий;
- ознаковий;
- диференціальний;
- контурний;
- структурний;
- структурний диференціальний.

Принцип растрового класифікатора полягає у попіксельному порівнянні символа на зображенні зі зразками. Цей класифікатор порівнює товщину, розмір, нахил тощо. Хоч він швидкий і простий, але в нього дуже низька точність.

Принцип ознакового класифікатора полягає у визначенні ознак символу на зображенні. Після визначення ознак складається ряд гіпотез, про те який це символ. Цей класифікатор швидкий і простий, але не стійкий до дефектів зображення. Також він використовується разом із растровим класифікатором.

Контурний класифікатор відрізняється від ознакового лише тим, що аналізує контури символу. Його точність менша, ніж у ознакового класифікатора.

Ознаковий диференціальний класифікатор схожий на ознаковий класифікатор [23], тільки він використовується для розрізнення схожих символів. Спочатку йому передаються гіпотези з ранніх етапів, а потім він уже аналізує лише ті області, у яких знаходяться відмінності. Цей класифікатор не висуває жодних гіпотез, лише уточнює існуючі.

Структурно-диференціальний класифікатор використовується для розпізнавання рукописного тексту [24]. Його принцип роботи набагато складніший ніж у попереднього класифікатора, а швидкість роботи менша, ніж у всіх класифікаторів.

Структурний класифікатор використовується на останніх етапах і вступає в дію досить рідко.

Хоч і на кожному етапі точність стає все вищою і вищою, але все одно алгоритм розпізнавання не ухвалює остаточного рішення. Число гіпотез зростає з геометричною прогресією кожному етапі. І тому перевірка всіх гіпотез навряд чи виявиться ефективною, і тому в OCR-системах ABBYY застосовується метод структурування гіпотез, тобто, відносить їх до тих чи інших моделей. Ось кілька їх типів: словникове слово, арабські цифри, спец символ тощо

Після визначення всіх символів починається зворотне складання: символів:

- слова;
- слова в речення;
- речення в текст.

Хоч і розробки в галузі розпізнавання символів зайшли вже достатньо далеко, точність все одно не ідеальна і навряд чи колись досягне 100% рівня.

1.1.2. Аналіз меж та фігур, контурний аналіз

Найочевидніший спосіб виділення символу– пошук контуру[26]. Цей спосіб працює тільки за наявності чіткого контуру з досить високою роздільною здатністю та з рівним кордоном, за відсутності перешкод між камерою та текстом.

Суть контурного аналізу: проводиться фільтрація зображення знаходження кордонів, потім проводиться виділення всіх знайдених контурів та їхній аналіз.

Цей спосіб непрактичний, оскільки контур символа може бути нечітким (художній текст, шуми).

1.1.2.1Аналіз частини кордонів

Набагато цікавіше, стабільніше та практичніше представляється підхід, де аналізується тільки частина символу [28]. Виділяються контури, після чого

шукаються всі вертикальні прямі. Для будь-яких двох прямих, розташованих неподалік одне від одного, з невеликим зрушенням по осі ординат, з правильним ставленням відстані між ними до них довжині розглядається гіпотеза того, що символ знаходиться між ними.

1.1.2.2 Гістограмний аналіз областей

Одним із найпопулярніших методів підходу є аналіз гістограм зображення. Підхід полягає в припущенні, що частотна характеристика символу відрізняється від частотної характеристики її рис.

На зображенні виділяються високочастотні просторові компоненти зображення. Будується проекція зображення на вісь *y* (іноді і на вісь *x*) [29]. Максимум отриманої проекції може збігтися з номером. Такий підхід має суттєвий мінус – символ за розміром має бути порівнянним з розміром кадру, так як фон може містити написи та інші деталізовані об'єкти.

1.1.2.3 Статистичний аналіз, класифікатори

Недоліки попередніх методів полягають у тому, що на практичному застосуванні, для кількості не стандартизованих вхідних даних немає ані виражених меж, ані вираженої статистики .

Найкращі, але рідко використовувані методи – це методи, що спираються на різні класифікатори. Ці методи дають можливість аналізувати задану область символів щодо наявності в ній характерних для символу рис які дозволяють розпізнати символ за складних та нетипових умов. Наприклад, навчений каскад Хаара [27]. Його робота проводиться за методом Віоли-Джонса, в основі якого лежать примітиви Хаара, які є розбивкою заданої прямокутної області на набори різнотипних прямокутних підобластей (рис 1.1)

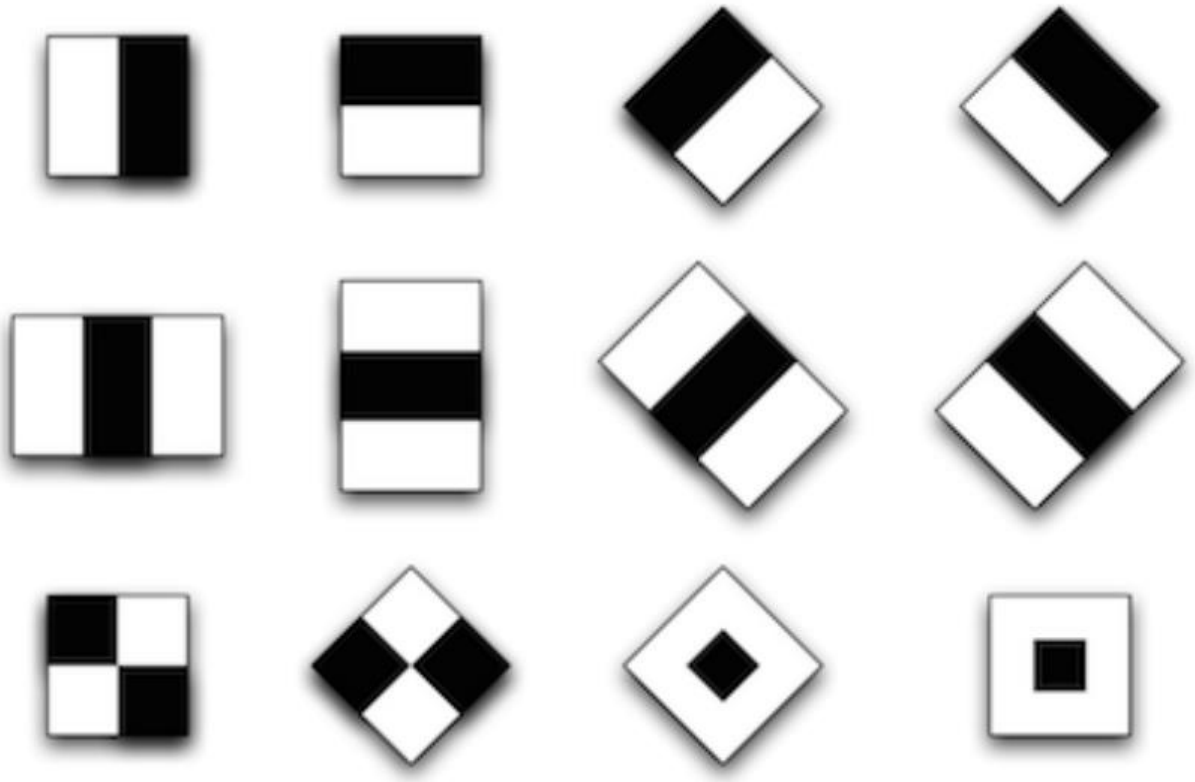

Рисунок 1.1 – Приклади розбивки прямокутної області на набори різнотипних підобластей

У реальних алгоритмах багато методів побічно або безпосередньо базуються на наявності меж символа. Навіть якщо при детектуванні номери кордону не використовуються, вони можуть використовуватись у подальшому аналізі.

1.1.2.4 Застосування Вейвлет-перетворень

У загальному випадку цей метод складається з трьох частин:

- вейвлет-перетворення зображення;
- фільтрація у вейвлет-домені;
- знаходження та виділення області номера.

Вейвлет-перетворення є вкрай необхідним, оскільки воно дозволяє збільшити ставлення корисного сигналу до шуму. Так, наприклад, застосування вейвлет-перетворення та подальшої фільтрації дозволило розпізнати по рутинним алгоритмам малоконтрастне зображення.

Для виконання завдання фіксується масштаб вейвлет-перетворення у значенні, що визначається розмірами області символа, частотою чергування чорного та білого в цій області та порядку обраного вейвлета.

При дослідженні кількох типів як дискретних, так і безперервних вейвлет-перетворень, різних сімейств і порядків, з'ясувалося, що результативнішими, як і передбачалося, виявилися симетричні вейвлети. Найкращий результат серед дискретних дали вейвлети сімейства койфлетів, а серед безперервних – гаусові вейвлети. Фіксуючи порядок кожного рядка вихідного зображення при виконанні вейвлет-перетворення, ми отримуємо таким чином не двомірний, а одномірний спектр.

Щоб описати чергування чорних та білих смуг в області символа, відповідні вейвлет коефіцієнти мають бути досить великими.

Крутизна меж області вейвлет-коефіцієнтів, згадана раніше, відповідна символу, що розпізнається, може бути детектована за великим значенням другої похідної. Обчислення другий похідний реалізується за допомогою методу бінаризації вейвлет-пікселів з подальшим застосуванням спеціального клітинного автомата (КА) для кластеризації. При переведенні в цифровий формат зображення всі пікселі вейвлет-зображення впорядковуються за рівнем сірого. Розташовані у впорядкованій послідовності знизу 70% пікселів стають чорними (=0), а 30% - білими (=1). Виживання кожної білої клітини КА залежить від восьми її сусідів організованих так, щоб «виживала» злита група одиничних пікселів, відповідна область символа, а оточуючі її окремі одиниці вимирали. Після застосування КА на периферії підсумкового вейвлет-зображення можуть бути присутні малі ділянки неправильної форми. Ці елементи легко можна обчислити за площею (числу пікселів у них) та відношенню довжини до ширини.

1.1.3 Алгоритм Бравермана

Алгоритм Бравермана, розроблений у 70-80-х роках, використовує виділення характерних фрагментів, і заснований на геометрії символів. Локальними геометричними особливостями вважаються як особливі точки на

контурних лініях - злами, перетину, закінчення, так і особливі крапки на межах чорних чи білих плям. Процес аналізу пред'явленого безлічі зображень починається з виділення на них так званих характерних фрагментів - ділянок зображень, що містять локальні геометричні особливості.

Після того, як на всіх зображеннях всі характерні фрагменти виділені, алгоритм приступає до формування словника форм характерних фрагментів Для цього кожному фрагменту ставиться у відповідність вектор, компоненти якого визначають його форму, потім безліч цих векторів піддається обробці за допомогою алгоритмів автоматичної класифікації У результаті накопичена безліч характерних фрагментів виявляється розділеним на класи «схожих» фрагментів. Клас приймається за окреме слово Словник форм є сукупність таких слів Для довільного нового фрагмента завжди можна вказати, до якого з отриманих класів він найближчий за формою.

Для кожного з виділених фрагментів будується набір характеристик його місця на зображенні. Ці характеристики необхідні для опису взаємного розташування геометричних особливостей.

Наступний етап алгоритму Бравермана полягає у побудові описів зображень та формуванні правила порівняння (розпізнавання) отриманих описів. Основна вимога до цього правила полягає в тому, щоб з його допомогою можна було б об'єднувати в класи (образи) такі описи, які відповідають зображенням, візуально схожим з людської точки зору.

Розроблений у 90-ті роки алгоритм адаптивного розпізнавання також використовує накопичення характеристик символів, що розпізнаються Алгоритм є двопрохідним процесом з навчанням на результатах першого проходу. Схема адаптивного розпізнавання поділяється на кілька етапів первинне розпізнавання, збір статистики, кластеризація зібраної статистики, формування еталонів та дорозпізнавання.

Визначимо кожен із названих етапів

- Первинне розпізнавання - це розпізнавання всієї сторінки за допомогою шрифтонезалежного алгоритму;

- Збір статистики – це процес відбору надійно розпізнаних символів, які згодом складуть навчальну вибірку для шрифтозалежного алгоритму;
- Кластеризація – це розбиття навчальної вибірки на кластери. З
- допомогою такого розбиття уточнюються результати розпізнавання, отримані на етапі первинного розпізнавання;
- Формування еталонів - це створення остаточних наборів даних, за якими проводитиметься дорозпізнавання;
- Дорозпізнавання – це другий прохід розпізнавання по всій сторінці з метою уточнити результати первинного розпізнавання, виставити адекватні оцінки точності, дорозпізнати те, що було не розпізнане раніше, відзначити ненадійно розпізнані символи.

1.2 Методи розпізнавання символів

1.2.1 Tesseract OCR

Це метод, який використовує відкрите програмне забезпечення, виконує автоматичне розпізнавання символів (літер), а також розпізнавання тексту. Tesseract існує для будь-яких ОС, стабільно працює та легко навчаємо – у цьому його гідність. Але він дуже погано працює з битим, брудним та деформованим текстом. При розпізнаванні символівза допомогою ПЗ правильно розпізнаються близько 20-30% з існуючої бази: найчіткіші та монохромні.

1.2.2 K-nearest

Дуже простий для розуміння метод розпізнавання символів, який, незважаючи на свою примітивність, часто може перемагати найвдаліші реалізації SVM або нейромережевих методів.

Принцип роботи:

1) попередньо записується певна кількість зображень реальних символів, коректно розбитих на класи;

2) вводиться міра відстані між символами (якщо зображення бінаризовано, то операція XOR буде оптимальна);

3) потім, при спробі розпізнавання символу, по черзі розраховується дистанція між символом, що розпізнається, і усіма символами в базі.

Серед найближчих сусідів, можливо, будуть представники різних класів. Вибір класу для символу проходить за класністю сусідніх символів. Символ отримує такий самий клас, як і у більшості сусідніх символів

Теоретично, за наявності великої бази з прикладами символів, зафіксованих у різних умовах, K-nearest – це все, що потрібно. Недоліком може бути лише те, що потрібно дуже швидко розраховувати дистанцію між зображеннями, тобто, бінаризувати його та використовувати XOR, а бінаризація абсолютно непередбачувано змінює символ. Тому у випадку із шумами або художніми символами будуть проблеми.

У багатьох випадках дуже важливо розуміти, як працює ваш алгоритм. Цей метод має одну, дуже важливу, у цьому плані перевагу: він простий і прозорий, а отже, легко налагоджується та налаштовується на оптимальний результат.

1.2.3 Кореляційний метод розпізнавання

Метод розпізнавання образів, за якого для кожного класу об'єктів, що розпізнаються, в декартовому просторі ознак задається еталонна область, і будь-який об'єкт що розпізнається відноситься до класу, що відповідає найближчій еталонної області. Область формується шляхом допустимих перетворень одного чи кількох еталонних векторів класу.

Якщо зображення не бінаризоване, і амплітуда сигналу невідома, відповідно, невідома яскравість символу.

Розпізнавання зводиться до операції розрахунку коваріації вхідного сигналу з гіпотетичним рахуванням заданих зсувів і поворотів: $cov(X,Y) = E[(X-EX)(Y-EY)]$, де X – вхідний сигнал, Y – гіпотеза, E – математичне очікування.

При необхідності вибору з різних символів, гіпотези щодо повороту та зміщення будуються окремо кожного символу. Якщо відомо, що вхідне зображення містить символ, то максимум коваріації з усіх гіпотез визначить

символ, його зміщення та нахил. Тут можна зіткнутися з проблемою близькості зображень різних символів («p» та «b», «o» та «c» та ін.).

Найпростіший спосіб вирішення цієї проблеми. введення кожного символу вагової матриці коефіцієнтів.

Іноді такі методи називаються «templatematching» (зпівставлення шаблонів), що повністю відображає їх суть: вибір зразків — потім порівняння вхідного зображення із зразками. У разі виникнення невизначеності за параметрами, проводиться перебір всіх можливих варіантів, або застосовуються адаптивні підходи.

Переваги методу:

- Передбачуваний і добре вивчений результат, якщо шум, навіть малою мірою, відповідає обраній моделі;
- при строго заданому шрифті, даний метод дозволяє розпізнати дуже неякісний символ.

Недоліки:

- Чималі витрати на процедури обчислення.

1.2.4 Розпізнавання скелетних образів

Насамперед символ, що розпізнається, піддається процедурі скелетизації (утончення). Існують різноманітні методи одержання скелета символу, відмінні один від одного.

Методи отримання скелету символа:

- Метод Щепіна;
- Скелетизація з застосуванням шаблонів;
- Хвильовий метод.

1.2.4.1 Метод Щепіна

Складається у визначенні вихідних верхніх лівих точок для кожного зовнішнього та внутрішнього контуру зображення. Далі розглядається конфігурація восьми сусідів кожної чергової точки контуру. Якщо точка не виявляється кінцевою, і після її видалення сусідні точки все ще залишаються

зв'язним безліччю, то крапка видаляється. Таким чином аналізується кожна крапка і її сусіди, наступна точка аналізу вибирається таким чином, щоб залишатися на межі зображення.

Після аналізу точки та її сусідів, і можливого видалення точки здійснюється перехід до наступної точки контуру таким чином, щоб залишитися на межі зображення. Послідовно аналізуючи кожну точку на межі зображення шарам, видаляємо шари до тих пір, поки не залишиться безліч точок, що не видаляються (рис 1.2). Вони й складуть скелет зображення.

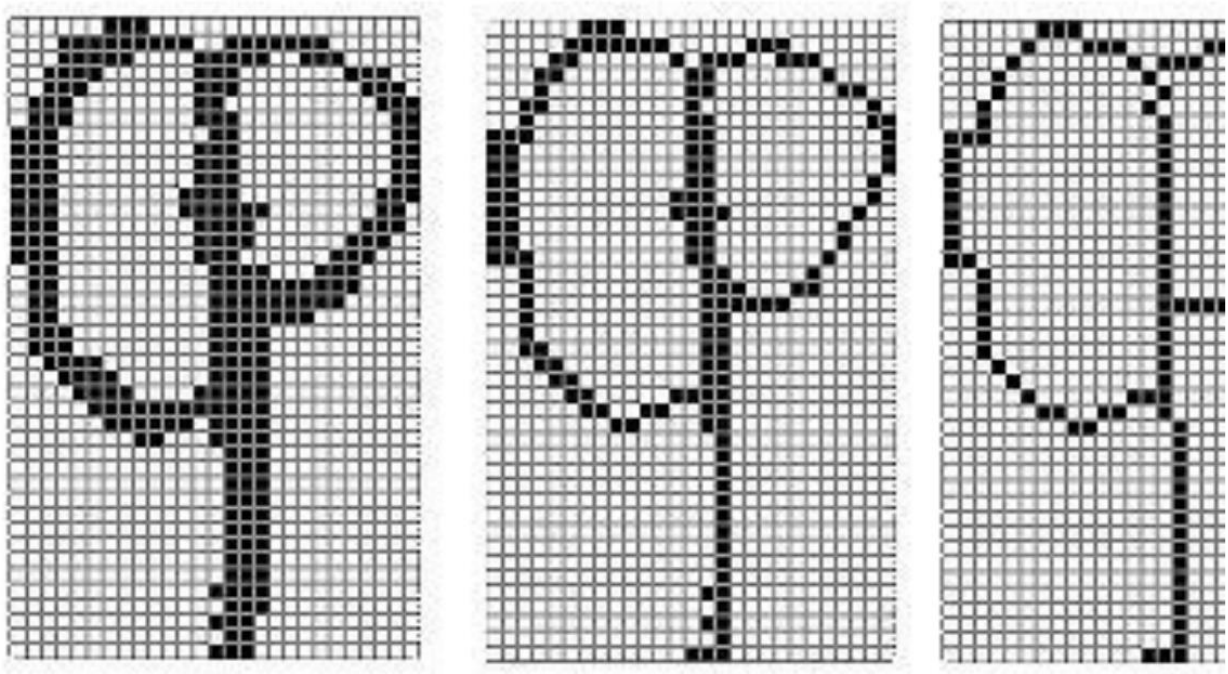

Рисунок 1.2 – Скелетизація літери «Ф», яка складається з одного внутрішнього та двох зовнішніх контурів

1.2.4.2 Скелетизація із застосуванням шаблонів

Існують шаблони, які можна використовувати для видалення зайвих пікселів та отримання скелетного зображення. Знаком «X» у них позначаються будь-які кольорові пікселі . Чорний піксель, що знаходиться в центрі, видаляється в будь-якій області, що відповідає хоча б одному шаблону. Таким

чином робиться кілька проходів по зображенню – до тих пір, поки не залишиться пікселів, які необхідно видалити.

Підготовчий етап перед розпізнаванням - виділення в отриманому скелеті зображення ключових точок (крапок, в яких з'єднуються три або чотири ребра скелета, а також кінцевих точок).

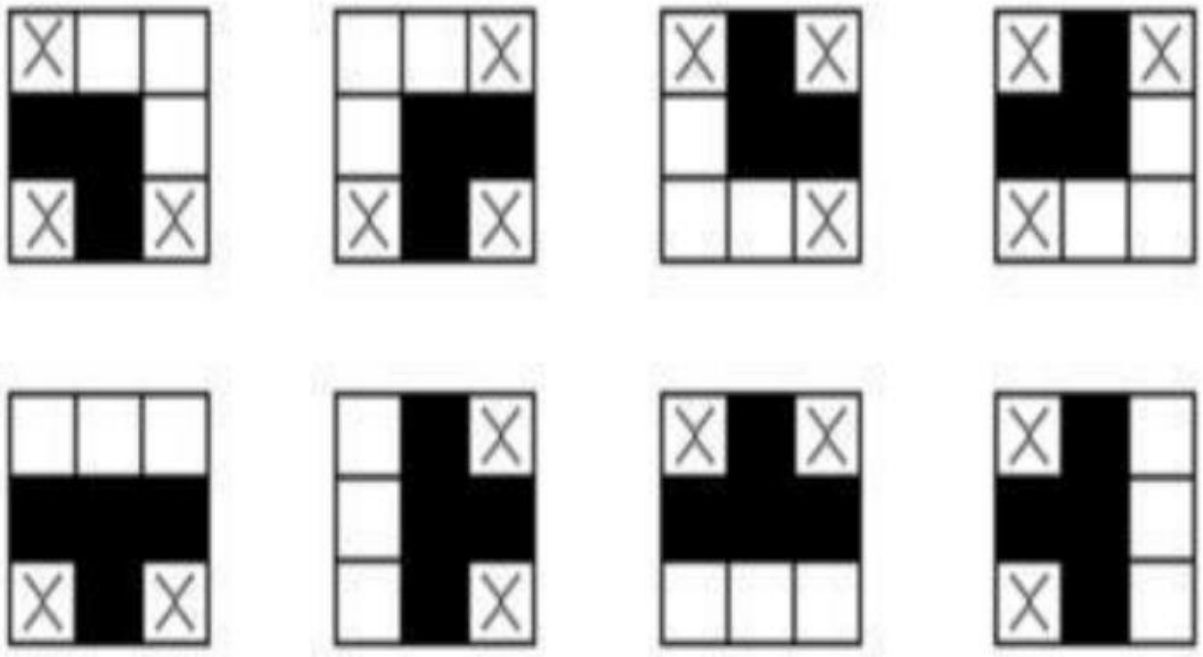

Рисунок 1.3 – Шаблони скелетизації

1.2.4.2.1 Алгоритм роботи

1. Створюється порожній стек, в якому будуть збережені координати ребер (початку та кінця) та точок розгалуження скелета.
2. У створений стек заносяться зазначені вище координати будь-якої точки кістяка.
3. Наступні дії (кроки 4-7) повторюються доти, доки стек не порожній.
4. Вибирається одна з точок, що у стеку.
5. З вибраної точки будується послідовність ребер, доки не відбудеться розгалуження скелета, або не буде досягнуто кінцевої точки.

У разі досягнення кінцевої точки, або досягнення зазначеного раніше ребра, масив даних вноситься пройдений шлях.

У разі досягнення точки розгалуження кістяка, знайдено місце з'єднання ребер.

6. У масив даних заноситься послідовність ребер.
7. У стек заносяться координати точки розгалуження скелета.

Переходимо до наступного пункту.

Отриманий опис скелета проходить попередню обробку, при якій видаляються короткі лінії скелета та об'єднання близьких тріодів.

1.2.4.3 Хвильовий метод

У цьому вся методі розглядається шлях проходження зображенню сферичної хвилі (рис 1.4), тобто, покроково аналізується зміщення центру мас точок, яке утворює нову генерацію хвилі щодо його попередніх положень.

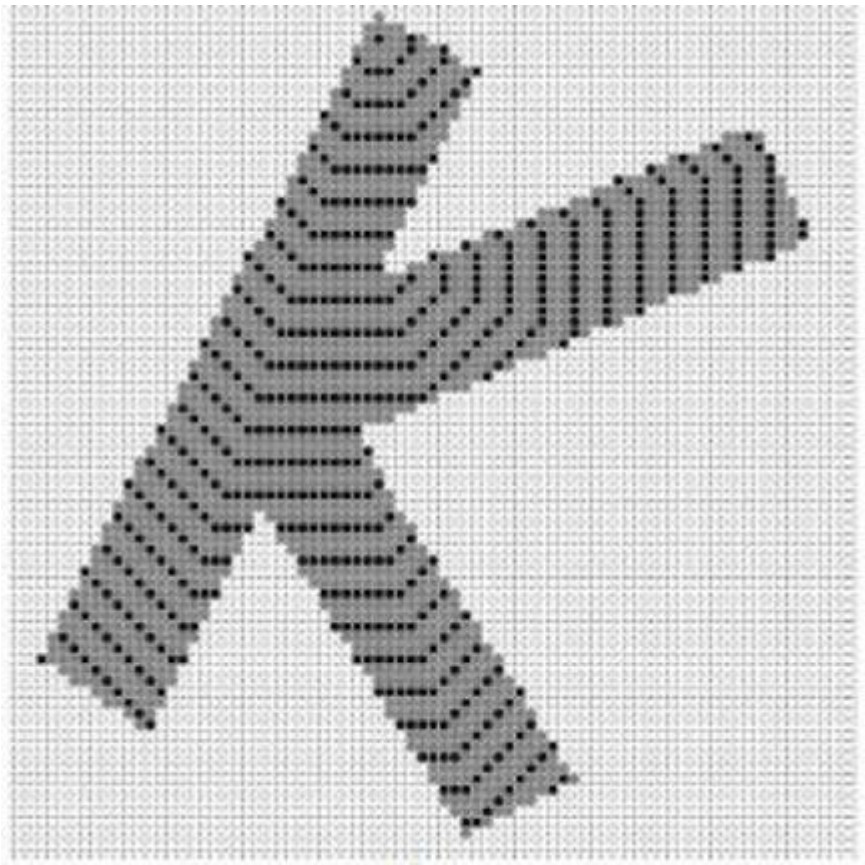

Рисунок 1.4 – Проходження сферичної хвилі по зображенню

Етапи, з яких складається метод:

- з допомогою сферичної хвилі виробляється побудова скелета зображення;
- робиться оптимізація отриманого скелета.

Лінії зображення спостерігаються шляхом відстеження переміщення центру відрізка, утвореного крайніми точками генерації хвилі (рис 1.5). Після того, як здійснено відстеження ліній, можливе згладжування отриманих відрізків.

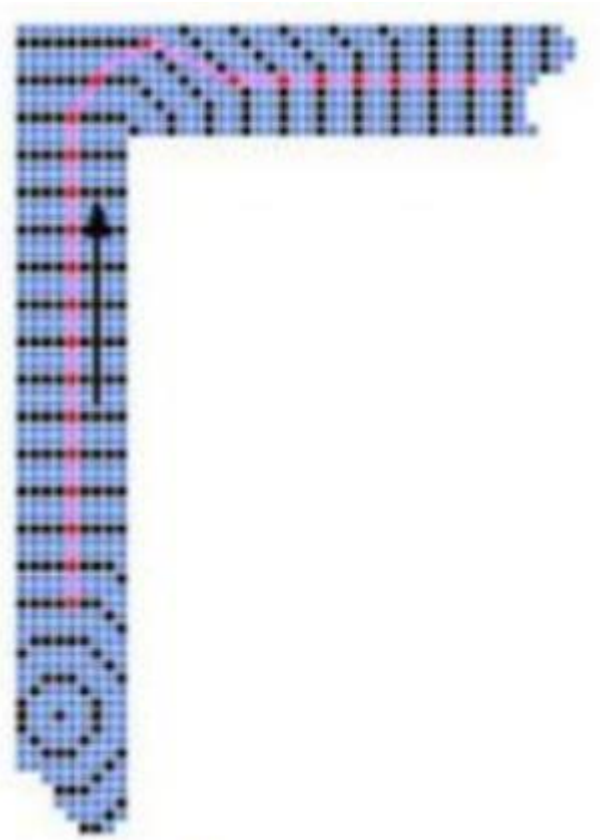

Рисунок 1.5 – Відстеження ліній зображення

Внаслідок роботи методу виходить не оптимальний скелет. Основна причина цього в тому, що робота ведеться з растровим зображенням, тобто зі спотворенням, які зростають у міру зменшення розміру зображення в символах. Тому для оптимізації скелета зображення, отриманого таким методом, потрібна оптимізація. Відрізок зображення у скелеті може бути представлений деякою послідовністю ребер. Оцінюючи відхилення лінії, що

виходить від прямої в послідовності ребер, можна цього позбутися. Якщо відхилення знаходиться в допустимих межах, послідовність ребер може бути замінена одним ребром. У процесі оптимізації скелета зображення проглядаються точки, де відбувається поділ хвилі на напівхвилі (тобто. околиці виділених точок з'єднання відрізків).

1.2.4.3.1 Адаптивне розпізнавання

Існує два підходи:

- Шрифтовий (шрифтозалежний)
- Безшрифтовий (шрифтонезалежний)

Перший підхід.

Програма вимірює і аналізує різні параметри шрифту і заносить в свою основу стандартних параметрів. Після закінчення процесу шрифтова програма оптичного розпізнавання символів (ОРС) готова до розпізнавання даного конкретного шрифту. Цей процес можна назвати навчанням програми.

Далі навчання повторюється для деякої множини шрифтів, яка залежить від області застосування програми. До недоліків даного підходу можна віднести такі фактори:

- алгоритм повинен заздалегідь знати шрифт;
- для роботи програми розпізнавання необхідний блок налаштування на конкретний шрифт.

З іншого боку, шрифтовий підхід має істотну перевагу: маючи детальну апріорну інформацію про символи, можна побудувати досить точні та надійні алгоритми розпізнавання.

Другий підхід.

Ці алгоритми вимірюють та аналізують різні характеристики (ознаки), властиві буквам як таким, безвідносно шрифту та абсолютного розміру (кегля), яким вони надруковані. До недоліків даного підходу можна віднести такі фактори:

- якість розпізнавання істотно нижча, ніж у шрифтових алгоритмів;

- низьке значення коефіцієнта надійності розпізнавання.

Основні переваги:

- Універсальність: алгоритм однаково працює з різними шрифтами
- технологічність: простота навчання;
- можливість реалізувати автоматизоване навчання.[19]

1.2.5 Ознаковий метод

Ознаковий метод базується на тому, що зображення відповідає N-вимірному вектору ознаки. Розпізнавання — це порівняння з набором опорних векторів однієї розмірності. Завдання розпізнавання та визначення атрибуції зображень у класі на основі аналізу обчислених ознак має декілька строгих математичних рішень у рамках детермінованого та стохастичного підходу.

Системи розпізнавання символів найчастіше використовують класифікацію, засновану на обчисленні евклідової відстані між вектором ознак розпізнаного символу та вектором ознак еталонного опису. Вид і кількість ознак, насамперед, визначають якість розпізнавання. Формування вектора здійснюється під час аналізу попередньо підготовленого зображення. Цей процес називається екстрагуванням. Стандарти класу отримують шляхом подібної обробки символів.

Основні переваги функціональних методів - простота реалізації, хороша узагальнення, хороша стійкість до зміни форм символу, низька кількість відхилень, швидкість. Найсерйознішим недоліком цих методів є їх нестійкість до різних дефектів зображення. Крім того, ознаковий метод має ще один серйозний недолік: на етапі видалення ознак необоротно втрачаються деталі інформації про символ. Видалення ознак відбувається незалежно, тому у будь-якому випадку інформація про взаємне розташування елементів знаку втрачається.

1.3 Постановка задачі

Метою роботи є розробка інформаційної технології розпізнавання тексту з зображення для мобільного додатку. Для досягнення цієї мети нам потрібно виконати такі завдання:

1. Формування вхідного математичного опису системи інтелектуального аналізу зображень.
2. Вибір типу та структури класифікатору зображень символів, що використовується інтелектуальною системою.
3. Розробка алгоритмів оптимізації функціональних параметрів інтелектуальної системи
4. Програмна реалізація функції інтелектуального аналізу зображень і її інтеграція в мобільний додаток.
5. Перевірка працездатності розробленої інтелектуальної системи.

2 АНАЛІЗ НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗПІЗНАВАННЯ СИМВОЛІВ

2.1 Оптичне розпізнавання символів

2.1.1 Визначення

Оптичне розпізнавання символів — це механічний або електронний переклад зображень рукописного, набраного або друкованого тексту в текстові дані, які використовуються для представлення символів на комп’ютері (наприклад, у текстовому редакторі). Розпізнавання широко використовується для перенесення книг і документів в електронну форму, для автоматизації систем обліку в компаніях або для публікації тексту на веб-сторінці. Оптичне розпізнавання символів дозволяє редагувати текст, шукати слова чи фрази, зберігати його в більш компактному вигляді, відображати чи друкувати матеріал без втрати якості, аналізувати інформацію та застосовувати електронний переклад, форматування чи редагувати текст. Оптичне розпізнавання тексту — предмет дослідження в області розпізнавання зображень, штучного інтелекту та комп’ютерного зору.

Оптичне розпізнавання символів стало дуже популярним у сучасному світі. Широке використання новітніх технологій призвело до перекладу всіх паперових документів і книг в електронний формат. Розпізнавання символів також використовується для редагування тексту, пошуку слів або фраз у тексті та застосування електронного перекладу до тексту.

Розвиток оптичного розпізнавання символів почався в 1929 році, коли Густав Таушек отримав патент на свій метод в Німеччині [1]. За ним послідував Гендель, який отримав патент у США в 1933 р. У 1950 р. Девід Х. Шепард розробив машину для перетворення друкованих повідомлень на машинну мову для комп’ютерної обробки. Пізніше він отримав патент, а згодом, заснував компанію з розробки інтелектуальних машин, яка запустила перші в світі комерційні системи оптичного розпізнавання символів. З цього моменту почався перехід на нові технології в різних сферах, як військових, так і державних.

Наступною за важливістю була перша система оптичного розпізнавання символів, яка розпізнавала друкований текст будь-яким шрифтом. У 1974 році Рей Курзвіль заснував Kurzweil Computer Products для створення цієї системи. Виріб було представлено 13 січня 1976 року під час прес-конференції. Вже існує багато програм для розпізнавання тексту. Серед них такі програми, як OCR SCAN, ABBYY FineReader, CuneiForm та багато інших.

2.2 Нейронні мережі для розпізнавання тексту

2.2.1 Визначення

Штучні нейронні мережі (ШНМ), або конекціоністські системи, є комп'ютерними системами, натхненними біологічними нейронними мережами, які утворюють мозок тварин. Такі системи вивчають завдання (поступово покращують їх виконання), розглядаючи приклади, як правило, без спеціального програмування завдання. Наприклад, у розпізнаванні зображень вони можуть навчитися ідентифікувати зображення, які містять собак, аналізуючи приклади зображень із позначками «собака» і «не собака» і використовуючи результати, щоб ідентифікувати собак на інших зображеннях. Вони роблять це без будь-яких апріорних знань про собак, наприклад, які мають собачу шерсть, хвости, вухи та морди . Натомість, вони розробляють власний набір відповідних характеристик з навчального матеріалу, який обробляють.

2.2.2 Метод навчання персептрона розпізнаванню текстових символів

Штучні нейронні мережі широко використовуються в розпізнаванні образів. Нейронну мережа зворотного зв'язку та конкуруючих нейронів можна використовувати для розпізнавання текстових символів, але коли розпізнаються шумні шаблони (текстові символи), використовуються нейронні мережі, такі як Хеммінга або Гроссберга. Найбільший ефект завдяки пластичності забезпечують мережі Гроссберга. Процес навчання збігається як функція значення параметра подібності[3].

Найпростіше навчання розпізнаванню текстових символів здійснюється за допомогою нейронних мереж прямого поширення. Оскільки мережа прямого розповсюдження є найпростішою, проблема розпізнавання цією мережею зашумлених текстових символів є актуальною.

Для навчання персептрона, кожному з відрізків поділу площини існування зображення призначається значення нуль або одиниця залежно від наявності на ньому сліду зображення. У цьому випадку пропонується ідентифікувати трасу зображення з однією з основних функцій, заданих положенням лінії в відрізку. Для цього кожен сегмент розбивається на підсегменти таким чином, щоб при кодуванні підсегменту двійковим вектором спотворення компонентів не призвело б до зменшення значень компонентів, які позначають сегмент. В якості такої міри вибирається поріг суми компонентів, позначених підсегментами відповідних сегментів[2].

Для визначення двійкових значень підсегментів необхідно визначити всі можливі функції (символи) для відображення символу в сегменті за допомогою бінарних компонентів, що позначають підсегменти.

При розпізнаванні текстових символів на площині такими функціями є: пряма, паралельна осі абсцис, пряма, паралельна осі ординат, пряма з додатним нахилом відносно осі абсцис, пряма з від’ємним нахилом. нахили осі відносно осі абсцис.

Щоб навчити персептрон розпізнавати текстовий символ, область призначення символу була поділена на пронумеровані квадрати (рис. 2.1) з цілими натуральними числами.

1	2	3
4	5	6
7	8	9

Рисунок 2.1 - Область призначення символу

Ці поля для текстових символів можуть містити одну з цих функцій, або бути пустими. Сортування за межами бінарних векторів до позначки області символу визначається за номером і топологією символу і для текстових символів, які ми визначаємо як 5x5[4].

Той самий розмір може бути пристосований для полів всередині квадрату. Поля для представлення раніше визначених функцій наведено на рис.2.2, рис 2.3, рис 2.4, рис 2.5, рис 2.6. Як і в розбитті на сегменти, 0 – округляє відсутність сліду зображення, 1 – округлює наявність сліду зображень під час бінарного кодування.

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Рисунок 2.2 - Пусте поле

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0

Рисунок 2.3 -Поле прямої паралельної осі абсцис

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0

Рисунок 2.4 - Поле прямої паралельної осі ординат

0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0

Рисунок 2.5 - Поле прямої з позитивним кутом нахилу

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Рисунок 2.6 - Поле прямої з негативним кутом нахилу

Тоді, нейронна мережа з п'яти нейронів (рис. 2.8) з бінарною функцією активації може одним із відомих методів навчання кластеризації цих функцій у кожному із сегментів. При виборі порогу кожного нейрона A_i вихідний сигнал набуває одиничного значення. При досягненні порога $S_n = 3$ при вагових коефіцієнтах $W_k = 1$, що відповідає $\sum_{i=l}^{5} \quad x_i = 3$.

Позначивши цю мережу з 5 нейронів символом R, приймемо $Z_1 = 0$, якщо $X_i = 0$, i =1;

$Z_2 = 1$, якщо двійковий вектор визначається лінійним кодом, відповідним масиву на рис. 2.3;

$Z_3 = 1$, якщо двійковий вектор, відповідає масиву на рис. 2.4;

$Z_4 = 1$якщо двійковий вектор, відповідає масиву на рис. 2.5;

$Z_5 = 1$якщо двійковий вектор, відповідає масиву на рис. 2.6;

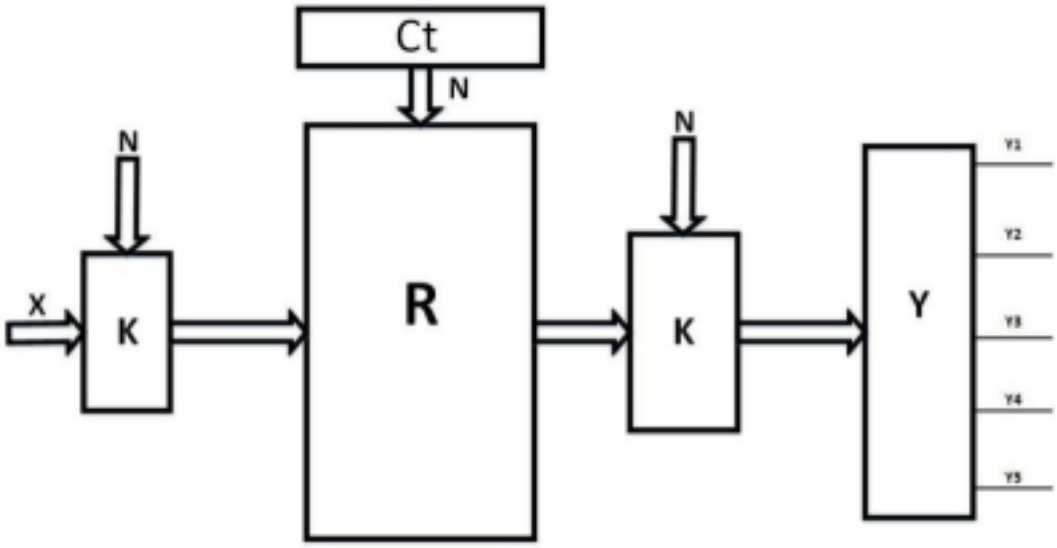

Рисунок 2.7 - Нейронна мережа

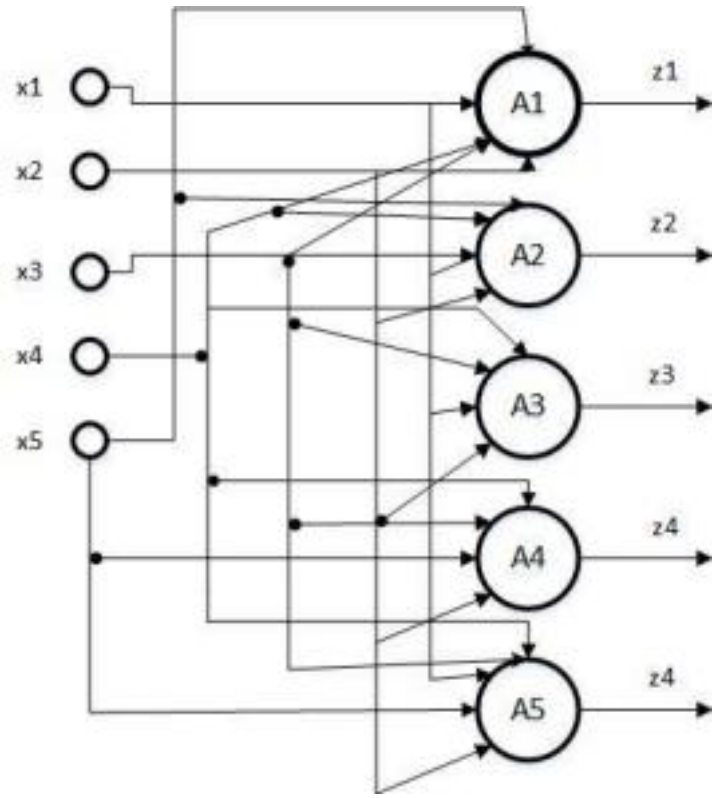

Рисунок 2.8 - Нейронна мережа з п’яти нейронів

Мережа включає два шари нейронів *R*, *Y* і два комутатора *K*, на керуючі входи яких надходять коди, що задають послідовність сегментів розбиття області завдання зображення. Шар нейронів *Y* включає фіксатори виходів шару *R* та *S* нейронів за кількістю текстових символів, які розпізнаються.

Шар фіксаторів представляє групи, кожна з яких містить 4 запам'ятовуючі елемента, в які заносяться одиниці, що відрізняють відповідний з 4 функцій, зображення якої є у вибраному сегменті, що задається кодом *N*, сформованим лічильником *Ct*.

Шар вихідних нейронів формує одиничний сигнал на виході, що визначається істинністю логічного виразу$y_i = f(M_i)$, де M_i - поодинокі виходи відповідних фіксаторів виходів шару *R*.

Шар нейронів *R* є мережею *A*, наведена на рис. 2.7. Вхід *X* представляє бітову послідовність п'ятикомпонентного вектора. Довжина послідовності визначається числом сегментів розбиття області завдання зображення.

Мережа може не містити комутаторів, тоді шари R та Y будуть містити групи нейронів за кількістю характерних функцій для шару R у кількості, що визначається числом сегментів розбиття, а в шарі Y кількість нейронів визначається числом символів, що розпізнаються.

Допустимі спотворення вектора X округлюються числом бітів, спотворення яких не призводять до переходу характеристичної функції f_i в $f_k, i = \underline{1,4}, k = \underline{1,4}, i \neq k$.

Алгоритм навчання мережі полягає у виділенні відповідно до двійкового вектору, у сегменті заданих характерних функцій, наприклад, за правилом Хебба. Виділені характеристики функції є входами шару Y, виходи якого задаються булевими функціями вхідних змінних як таблиці істинності. Навчання шару Y може бути проведено або за правилом Хебба, або за допомогою дельта правила. За будь-якого способа навчання, мережа детермінована, і кожному вхідному зображенню відповідає єдиний вихід із сигналом рівним одиниці (Рис 2.7).

2.2.3 Розпізнавання символів за допомогою згорткової нейронної мережі

Аналіз методів розпізнавання образів поки що зал, що для вирішення цього завдання ефективно використовувати штучні нейронні мережі. Здебільшого використання нейронних мереж для обробки зображень – навчання системи для виділення ключових характеристик символів із навчальних наборів[6].

Найчастіше у завданнях розпізнавання та ідентифікації зображень використовуються классичні нейромережеві архітектури (багатошарний персептрон, мережі з радіально базисною функцією та ін.), але з аналізу даних методів та експериментальних досліджень випливає, що застосуванняня класичних нейромережевих архітектурній задачі неефективно з наступних причин:

- велика кількість параметрів збільшує місткість системи та відповідно вимагає більшої тренувальної вибірки, збільшує час та обчислювальну складність процесу навчання;
- для підвищення ефективності роботи системи бажано застосовувати кілька нейронних мереж (навчені з різними початковими значеннями синаптичних коефіцієнтів і порядком пред'явлення образів), але це збільшує обчислювальну складність і час виконання завдання;
- відсутня інваріантність змін мас штабу зображення, ракурсів зйомки камери та інших геометричних спотворень вхідного сигналу [8].

Тому для розпізнавання символів на номірних знаках були обрані згорткові нейронні мережі, тому що вони забезпечують часткову стійкість до змін масштабу, зміщень, по воротах, зміні ракурсу та інших спотворень. Архітектура згорткової нейронної мережі складається з багатьох шарів. Шари бувають двох типів: свер точні та підвиборні, вони чергуються один з одним. Нейрони в межах шару організовані в площині. У кожному шарі є набір з кількох площин, причому нейрони однієї площини мають однакові синаптичні коефіцієнти, що ведуть до всіх локальних ділянок попереднього шару. Кожен нейрон шару отримує входи від деякої області попереднього шару (локальне рецеп тивне поле), тобто вхідне зображення попереднього шару хіба що сканується невеликим вікном і пропускається крізь набір синаптичних коефіцієнтів, а результат відображається на відповідний нейрон. Набір площин є картами характеристик, і кожна площина знаходить «свої» ділянки зображення в будь-якому місці попереднього шару. Розмір локального рецептивного поля вибирається самостійно в процесі розробки нейронної мережі . Підвибірковий шар [5] зменшує масштаб площин шляхом локального усереднення значень виходів нейронів. Таким чином досягається її рархічна організація. Наступні шари спричиняють більш загальні характеристики, менше висять від спотворень зображення.

Поступово нейронна мережа навчається виділяти ключові характеристики зображень, що надходять на вхід[8] .

Розглянемо нейронну мережу, що складається з 3х прихованих шарів, представлена на рис. 2.8[15]. Вхідними даними нейронної мережі є відмасштабовані зображення розміром 29×29 пікселів. Слідом за вхідним шаром знаходиться перший прихований шар, який є згортковим. Він складається із шести карт ознак розміром 13×13. Кожен елемент карти ознак з'єднаний з рецептивним полем розміром 5×5 на вхідному зображенні та поодиноким зміщенням. Отже, кожен елемент карти має 26 вагових коефіцієнтів, що навчаються. Значення всіх елементів карти ознак обчислюються шляхом послідовного сканування рецептивним полем вхідного шару. Таким чином, у першому прихованому шарі міститься 26364 синаптичних зв'язків і 156 об'єднаних параметрів.

Розмір згорткової площини визначається відповідно до наступних виразів:

$$w_c = \frac{(w_u - 3)}{2},$$

де w_c – ширина згорткової площини; w_u – шири на площині попереднього шару.

$$h_c = \frac{(h_u - 3)}{2}$$

де h_c - Висота згорткової площини; h_u – висота поверхні попереднього шару. Другий прихований шар також є шаром згортки. Він складається з 50 карт ознак розміром 5×5. У цьому шарі 5×5×50=1250 нейронів. Кожний елемент у карті ознак пов'язаний із шістьма областями розміром 5×5 шести карт попереднього шару. Таким чином, у другому прихованому шарі міститься 7550 ваг і 188750 зв'язків. Два перших згорткових шари можна розглядати як шари для вилучення ознак із зображення

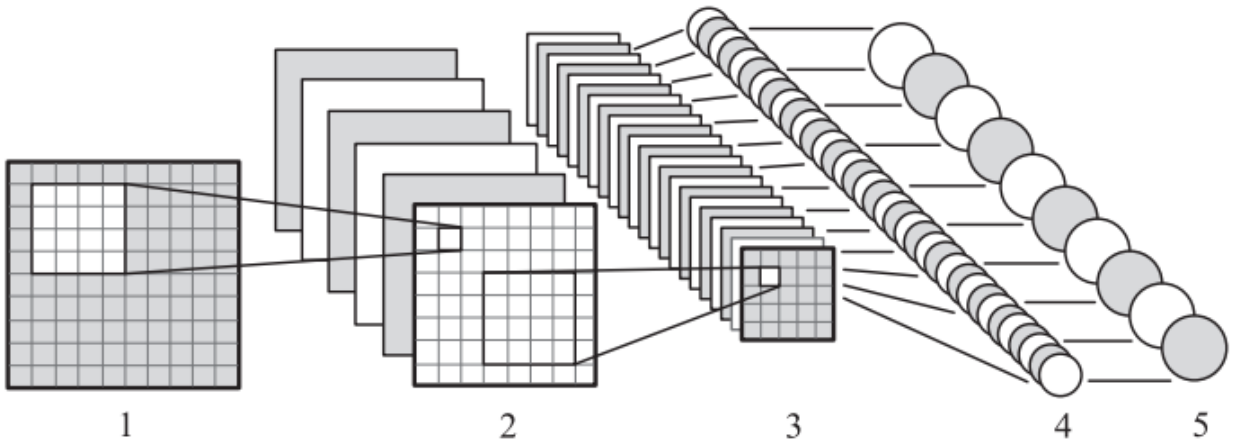

Рисунок 2.9 - Архітектура згорткової нейронної мережі: 1) вхід; 3) приховані згорткові шари; 4) шар із звичайних нейронів; 5) виходи мережі

Наступні два шари є шарами класифікації. У цих шарах кожен нейрон з'єднаний із усіма нейронами попереднього шару.

Третій прихований шар складається із 100 нейронів. У цьому шарі є 125100 зв'язків і 125100 параметрів, що навчаються. Для десяти класів зображень, відповідних десяти цифр, 100 нейронів достатньо хорошої узагальнюючої можливості мережі[9].

Дана нейронна мережа працює як класифікатор, тому четвертий вихідний шар складається з 21 нейрона, оскільки може розпізнавати 21 символ. Відповідно до ГОСТ Р 50577 93 на реєстраційні знаки транспортних засобів можуть містити такі символи: А, В, Е, К, М, Н, О, Р, С, Т, Х, У і всі цифри від 0 до 9. Отже система повинна розпізнавати 21 символ.

Як активаційну функцію було обрано гіперболічний тангенс:

$$f(a) = Atanh(Sa),$$

де A – амплітуда цієї функції, S – визначає її становище щодо початку відліку. Функція f – непарна, її горизонтальні асимптоти дорівнюють A і -A.

Ця функція має ряд переваг для вирішення задачі:

- симетричні активаційні функції типу гіперболічного тангенсу, що забезпечують більш швидку збіжність, ніж стандартна логістична функція;

- має безперервну першу похідну;
- має просту похідну, яка може бути обчислена через її значення, що дає економію обчислень.

$$y_k^{(i,j)} = b_k + \sum_{s=1}^{K}\sum_{t=1}^{K} w_{k,s,t}\, x^{((i-1)+s,(j+t))} \tag{2.1}$$

Формула функціонування нейрона згорткового шару (2.1), де $y_k^{(i,j)}$- нейрон k-ї площини згорткового шару;

b_k – нейронне зміщення k-ї площини;

K – розмір рецептивної області нейрона;

$w_{k,s,t}$ – матриця синаптичних коефіцієнтів;

x - виходи нейронів попереднього шару[10].

Використовується стандартний для нейронних мереж алгоритм зворотного розповсюдження помилки. Для вимірювання якості розпізнавання використовувалася функція середньоквадратичної помилки (2.2)

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{2.2}$$

де E_p Ep – величина функції помилки образу p;

t_{pj} - бажаний вихід нейрона j для образу p;

o_{pj} -дійсний вихід нейрона j для образу p[13].

Остаточна корекція синаптичних коефіцієнтів відбувається за (2.3)

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_{pj} o_{pj} \tag{2.3}$$

де η - Коефіцієнт пропорційності, що впливає на швидкість навчання[12].

На кожній ітерації алгоритм зворотного розповсюдження помилки розраховується для всього навчального набору даних, щоб обчислити середній чи справжній градієнт.

Коли не налаштованій мережі пропонують вхідний образ, вона видає деякий випадковий вихід. Функція помилки є різницею між поточним виходом мережі та ідеальним виходом, який потрібно отримати. Для успішного навчання мережі потрібно наблизити вихід мережі до бажаного виходу, тобто послідовно зменшувати величину функції помилки[11]. Це досягається налаштуванням міжнейронних зв'язків. Кожен нейрон у мережі має свої ваги, які налаштовуються, щоб зменшити величину функції помилки.

Значення вагових коефіцієнтів були обрані випадковим чином із нормального розподілу з нульовим середнім та стандартним відхиленням $\delta_w = \sqrt{m}$, де m - число зв'язків що входять в нейрон[18].

Експериментально доведено, що локальне рецептивне поле, що дорівнює 5×5 пікселів, дозволяє уникнути обчислювальної надмірності та забезпечити надійне розпізнавання символів.

Для навчання мережі використовувалася база зображень, що складається із 5000 символів[17]. Значення вихідних синаптичних ваг для всіх шарів мережі з гіперболічним тангенсом вибираються на основі рівномірного розподілу з нульовим математичним очікуванням та дисперсією, зворотною квадратного кореня, з кількості синаптичних зв'язків нейрона

Нейронна мережа забезпечує можливість розпізнавання символів на рівні 95 % і швидкість роботи 0,5 секунд. У ході експериментів було встановлено, що мережа помилково розпізнає символи у разі відхилення пластини номерного знака по горизонталі та вертикалі більше 45° та повороті пластини на поверхні до 15°[16]. У цих випадках атрибути символів нашаровуються один на одного і не піддаються розпізнаванню навіть людським оком.

2.3 Вибір методу

Процес розпізнавання тексту з зображення складається з багатьох кроків:

1) сканування тексту;
2) передача отриманих данних до програми розпізнавання ;
3) обробка зображення;
4) розмітка зображення;
5) розпізнавання розміченого зображення;
6) перетворення результатів у формат Word

Для реалізації функції розпізнавання тексту в моєму додаткові, я вирішила обрати згорткові нейронні мережі.

Перевагаги згорткових мереж перед персептронами - це використання спільної ваги в згорткових шарах, тобто, для кожного пікселя шару використовується один фільтр.

Принцип роботи нейронів штучної мережі дозволяє виявити їхню активність, яка визначається їх параметрами: вагами та функцією активації.

Згорткова нейронна мережа:

- має вхід фіксованого розміру та генерує вихідні дані фіксованого розміру;
- відноситься до типу штучної нейронної мережі зі зворотним зв’язком: варіації багатошарових персептронів, які розраховані на використання мінімальної кількості попередньої обробки;
- використовує схему взаємодії нейронів, подібну до організації зорової кори тварин.

2.4 Актуальність використання штучних нейронних мереж

На початку свого існування,застосування нейронних мереж в інтелектуальному аналізі даних викликало скептичне відношення, зважаючи на недоліки, властиві нейронним мережам: складна структура, погана

інтерпретованість і довгий час навчання . Однак їх переваги, такі як висока допустимість до зашумлених даних і низький коефіцієнт помилок, безперервне вдосконалення та оптимізація різних алгоритмів навчання мереж, алгоритму вилучення правил, алгоритму спрощення мереж, роблять нейронні мережі все більш і більш перспективним напрямком у data mining[21] . Області застосування нейронних мереж великі: автоматизація процесів розпізнавання образів, прогнозування, адаптивне управління, створення експертних систем, організація асоціативної пам'яті, обробка аналогових та цифрових сигналів, синтез та ідентифікація електронних ланцюгів та систем. Таким чином, можна сказати, що використання нейронних мереж у технології інтелектуального аналізу даних є актуальним напрямком, який безперервно розвивається, шляхом усунення недоліків.

Використання нейронних мереж у цій роботі спирається на можливість реалізації цієї техніки для використання в доступному, для будь якого пристрою, мобільному додатку. Розробку здійснено мовою програмування JS, яка задовольняє умови створення кросплатформного мобільного додатку та підтримку різнопланових функцій застосунку. Мобільність та доступність користування всіма перевагами прогресу - пріоритет в розвитку сучасних технологій, направлених на максимальну комерціалізацію будь якого продукту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ФУНКЦІЇ

3.1 Засоби розробки нейронної мережі

Для програмної реалізації нейронної мережі котру я могла б використовувати в мобільному додатку, що працює лише на стороні клієнту (через відсутність сервера), я обрала мову програмування JavaScript, так як вона, на мій погляд, задовільнює всі мої технічні потреби.

JavaScript мова програмування, яка дозволяє створити динамічно оновлюваний контент.

Ядро мови JavaScript складається з деякої кількості звичайних можливостей, які дозволяють:

- Зберігати дані всередині змінних.
- Проводити операції над фрагментами текстів (відомими у програмуванні як "рядки").
- Запускати код відповідно до певних подій, що відбуваються в додатку.

Ще більш цікавою є функціональність, створена поверх основної мови JavaScript. Так звані інтерфейси прикладного програмування, які надають додаткові надздібності для використання у коді JavaScript.

API – це готові набори блоків коду, які дозволяють розробнику реалізовувати програми, які інакше було б важко чи неможливо реалізувати. Вони роблять те саме для програмування, що готові комплекти меблів роблять для домашнього будівництва - набагато простіше брати готові панелі і скручувати їх разом ніж проходити всі етапи з нуля.

Вони зазвичай поділяються на дві категорії.

1. API-інтерфейси браузера вбудовані у веб-браузер і можуть відображати дані з навколишнього комп'ютерного оточення або робити корисні складні речі. Наприклад:
2. API-інтерфейс DOM (Document Object Model) дозволяє маніпулювати HTML і CSS, створювати, видаляти та змінювати HTML, динамічно застосовувати нові стилі до сторінки тощо.

3. API геолокації отримує географічну інформацію.
4. API Canvas та WebGL дозволяють створювати анімовані 2D та 3D-графіки.
5. Аудіо та відео API, такі як HTMLMediaElement та WebRTC, дозволяють робити дійсно цікаві речі з мультимедіа, такі як програвання аудіо та відео прямо на веб-сторінці, або захоплювати відео з веб-камери та відображати його на Чужий комп'ютер

Всупереч поширеній думці, JavaScript не є "інтерпретованим Java". У двох словах, JavaScript - це динамічна скриптова мова, що підтримує прототипне створення об'єктів. Базовий синтаксис навмисно схожий на Java та C++, щоб зменшити кількість нових концепцій, необхідних вивчення мови. Такі мовні конструкції, як if, for, while, switch, try...catch схожі на конструкції цих мов.

JavaScript може функціонувати і як процедурна, і як об'єктно-орієнтована мова. Об'єкти можна створювати програмно під час виконання, шляхом приєднання методів та властивостей або порожніх об'єктів під час виконання, на відміну від синтаксичних визначень класів у мовах, що компілюються, таких як C++ або Java. Після того, як об'єкт був створений, він може бути використаний як план (або прототип) для створення схожих об'єктів.

Динамічні можливості JavaScript включають: створення об'єктів під час виконання, змінну кількість параметрів, динамічне створення скриптів (за допомогою eval), перебір об'єктів (за допомогою for ... in), відновлення вихідного коду (програми на JavaScript можуть декомпілювати тіла функцій назад у вихідний код).

Якщо спрощувати, то JavaScript має наступні переваги:

1) Швидкість. JavaScript має тенденцію бути дуже швидким, тому що він часто запускається відразу в браузері клієнта. Поки він не потребує зовнішніх ресурсів, JavaScript не уповільнюється через виклики внутрішнього сервера. Крім того, всі основні браузери підтримують JIT-

компіляцію (точно вчасно) для JavaScript, а це означає, що немає потреби компілювати код перед його запуском.

2) Простота - синтаксис JavaScript був натхненний Java і його відносно легко вивчити в порівнянні з іншими популярними мовами, такими як C++.
3) Популярність - JavaScript всюди в мережі, а з появою Node.js він все частіше використовується в серверній частині. Є безліч ресурсів для вивчення JavaScript. І StackOverflow, і GitHub демонструють зростання кількості проектів, що використовують JavaScript, і очікується, що популярність, здобута останніми роками, тільки зросте.
4) Interoperability - На відміну від PHP або інших мов сценаріїв, JavaScript може бути вставлений у будь-якій веб-сторінці. JavaScript може використовуватись у багатьох різних програмах завдяки підтримці інших мов, таких як Pearl та PHP.
5) Навантаження на сервер - JavaScript є клієнтським, тому він знижує потребу в серверах в цілому, а простим програмам сервер може взагалі не знадобитися (саме як в моєму випадку).
6) Багаті інтерфейси.
7) Розширена функціональність - розробники можуть розширити функціональність веб-сторінок, написавши фрагменти коду JavaScript для сторонніх надбудов, таких як Greasemonkey.
8) Універсальність - є багато способів використовувати JavaScript через сервери Node.js. Якби ви завантажували Node.js за допомогою Express, використовували базу даних документів, таку як MongoDB, і використовували JavaScript у зовнішньому інтерфейсі для клієнтів, можна було б розробити цілу програму JavaScript від початку до кінця, використовуючи тільки JavaScript.
9) Оновлення - з моменту появи ECMAScript 5 (специфікації сценаріїв, на якій базується JavaScript), ECMA International щорічно оновлює JavaScript.

Не зважаючи на всі переваги, JavaScript все ж має і недоліки:

1) Безпека на стороні клієнта - оскільки код JavaScript виконується на стороні клієнта, помилки та упущення іноді можуть використовуватися в зловмисних цілях. Через це деякі люди вважають за краще повністю відключати JavaScript.
2) Підтримка браузерів - У той час як серверні скрипти завжди дають той же результат, різні браузери іноді інтерпретують код JavaScript різному. У наші дні відмінності є мінімальними, і вам не варто турбуватися про це, якщо ви тестуєте свій скрипт у всіх основних браузерах.

Стосовно нашого випадку, недоліки не є суттевими. Додаток, для якого буде реалізована нейронна мережа не використовує підключення до інтернету і приватність даних в цілому залежить від надійності захисту даних користувачем – мисливці на інформацію не матимуть змоги дізнатись збережені дані не отримавши фізичний носій – пристрій. Враховуючи вище вказане, я вважаю JavaScript ідеальним засобом реалізації нейронної мережі.

Прийшов час розробити план реалізації.

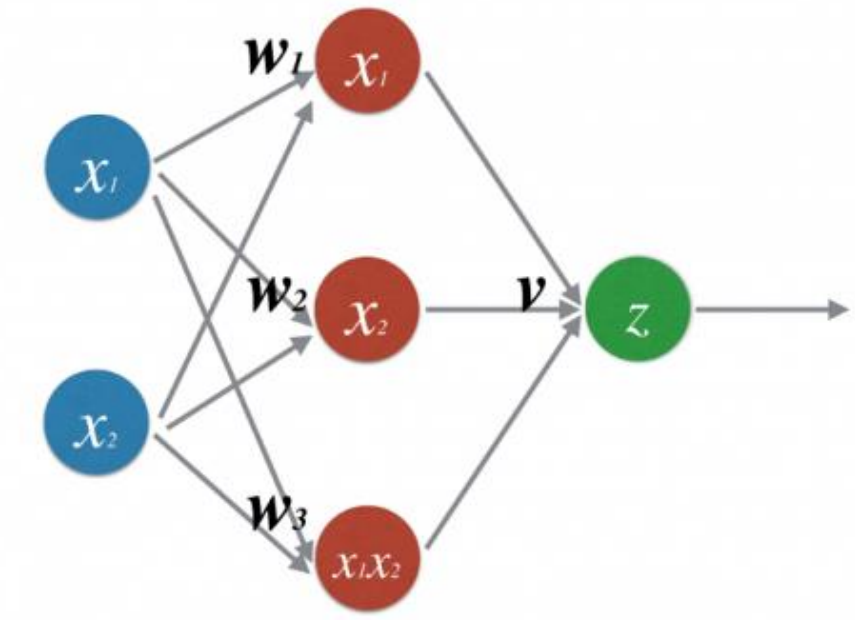

Рисунок 3.1 - Мережа з 2-3-1 нейронами у вхідному, прихованому та вихідному шарі.

1. Перше, що потрібно зробити, це створити шари. Це робиться за допомогою функції new Layer у synaptic. Число в дужках визначає, скільки кожен шар повинен утримувати нейронів;
2. З'єднати ці шари разом і створити інстанс нової мережі;
3. На цьому етапі має вийти мережа з 2-3-1 нейронами у вхідному, прихованому та вихідному шарі, відповідно. Архітектура цієї мережі зображена на рис. 3.1 .
4. Запустити мережу 20000 разів. Щоразу поширюємося у прямому та зворотному напрямку чотири рази, проходячи через чотири можливі входи для цієї мережі: [0,0], [0,1], [1,0], [1,1].
5. Почати треба з команди myNetwork.activate ([0,0]), де [0,0] — дані, які надсилаються до мережі. Це пряме поширення, яке також називається активізацією мережі. Після кожного прямого розповсюдження потрібно зробити і зворотне, тоді мережа оновить свої ваги і зсув.
6. Здійснити зворотне розповсюдження командою myNetwork.propagate(learningRate, [0]). Параметр learningRate - константа, яка говорить мережі, наскільки щоразу потрібно змінювати ваги. Другий параметр - 0 представляє коректну відповідь для заданого входу [0,0].
7. Далі мережа порівнює свій прогноз з правильною міткою. На цьому етапі визначається, чи було передбачення зроблено правильно.
8. Мережа використовує порівняння як базис для коригування значень ваг та зсуву. Тому наступного разу прогноз буде трохи точнішим.
9. Після виконання 20 000 ітерацій у циклі for можна подивитися, наскільки добре навчилася мережа за допомогою активізації мережі з усіма чотирма можливими входами
10. Якщо округлити ці значення до найближчого цілого числа, то отримаємо коректні відповіді для XOR операцій.

    Програмна реалізація даного плану буде представлена в додаткові.

Щодо навчання моеї нейронної мережі: аби навчити програму, я під’єднаю базу даних з допомогою NoSQL.

Загальних характеристик для всіх NoSQL небагато, тому що під лейблом NoSQL зараз ховається безліч різнорідних систем, але я виділю основні

1. Не використовується SQL

Мається на увазі ANSI SQL DML, тому що багато баз намагаються використовувати query languages схожі на загальновідомий улюблений синтаксис, але повністю його реалізувати не вдалося нікому і навряд чи вдасться. Хоча за чутками є стартапи, які намагаються реалізувати SQL.

2. Неструктурованість (schemaless)

Сенс такий, що в NoSQL базах на відміну від реляційних структура даних не регламентована (або слабо типізована, якщо проводити аналогії з мовами прогамування) — в окремому рядку або документі можна додати довільне поле без попередньої декларативної зміни структури всієї таблиці. Таким чином, якщо виникає необхідність змінити модель даних, то єдина достатня дія - відобразити зміну в коді програми.

В моєму випадку, я підключала базу даних окремим документом

У коді програми дані часто представлені як об'єкт (або документ) у форматі, JSON, оскільки це ефективна та інтуїтивна модель даних. Документні бази даних зберігати та вимагати дані в БД за допомогою тієї ж документної моделі, яка використовуються у коді програми. Гнучкий, напівструктурований, ієрархічний характер документів та документних баз даних дозволяє відповідати потребам додатку. Документна модель добре працює в каталогах, профілях і системах управління контентом, де кожен документ унікальний і змінюється з часом.

3.2 Тестування функції розпізнавання тексту

Ідеальний детектор повинен розпізнавати всі текстові символи на зображені. Програма обмежена кількістю підключених до неї бібліотек що

відповідають за мовні обмеження (латиниця, кирилиця, спецсимволи-умлаути, та цифри). В випадку що ми оглядаємо в даній роботі, програма містить бібліотеки для розпізнавання англомовних текстів (латиниці), цифр та деяких спецсимволів: дужок, фігурних дужок, слешів тощо. Виходячи з вищесказаного, для програми що оглядається, ідеальним показником буде вважатися розпізнавання всіх символів англійської мови, цифр та спецсимволів. Для того щоб перевірити якість роботи проведемо аналіз декількох прикладів результатів роботи програми.

Перед початком аналізу сформуємо очікуваний результат. В якості предмету експерименту використаємо текст статті з інтернету. Еталонним результатом буде перенесення інформації з зображення в текстовий формат, зі збереженням всіх великих та малих літер, послідовності символів, розділових знаків, та інших допустимих символів передбачених підключеними бібліотеками.

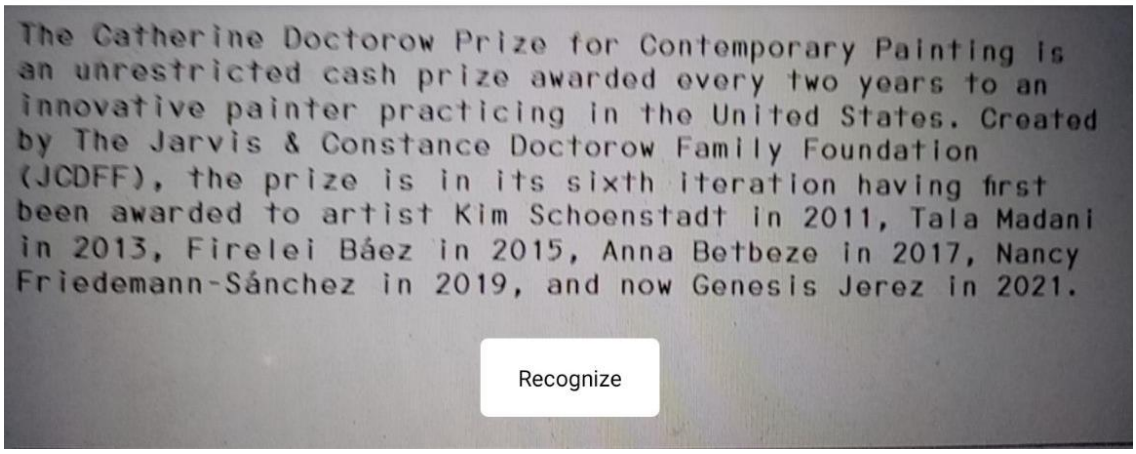

Рисунок 3.2 – Тестовий текст для перевірки коректності роботи програми.

На рис. 3.2 приведений скріншот використання функції розпізнавання на практиці. Як видно, на зображенні читабельний англомовний текст що містить цифри та розділові знаки. Також, текст стилістично поділений на “абзаци”. На рис 3.3 та рис. 3.4 ознайомимося з результатом зчитування символів.

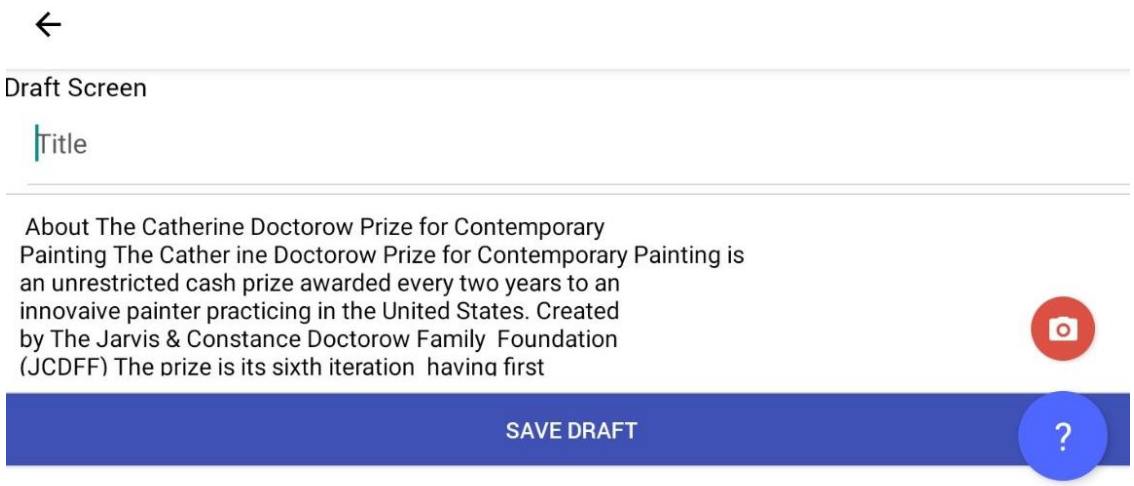

Рисунок 3.3 - Результат зчитування символів та їх конвертування в текстовий формат в середині додатку (частина 1)

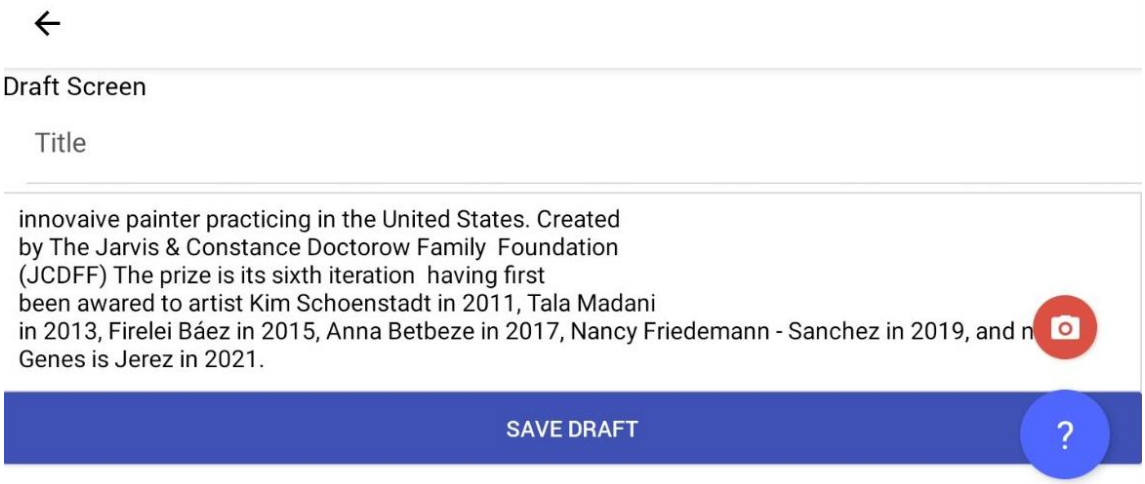

Рисунок 3.4 - Результат зчитування символів та їх конвертування в текстовий формат в середині додатку (частина 2)

На практиці функція переносить текст зі збереженням послідовності символів, та їх розміру. Окрім того програма розпізнає абзаци та зберігає текст поділеним як на вхідному матеріалі. Після аналізу результату, робимо висновок що результат майже еталонний, так як в результаті є два відхилення від вхідного матеріалу:

1. Не зберігся один з наголосів в середині тексту;

2. Програма знехтувала останнім абзацом, записавши два останні рядки прикладу як один.

Враховуючи складний стиль шрифту, відхилення не є критичними, тож результат можна вважати задовільним, а похибку допустимою. Отже, при позитивних умовах програма працює коректно, але як вона відреагує на текс який складається з символів котрих немає в бібліотеці?

Влаштуємо додатку ще один тест. На рис. 3.4 можна зображено попередній екран додатку та новий текст з яким ми будемо мати справу. Текст написаний українською і включає в себе символи кирилиці, розпізнавання яких не передбачено програмою.

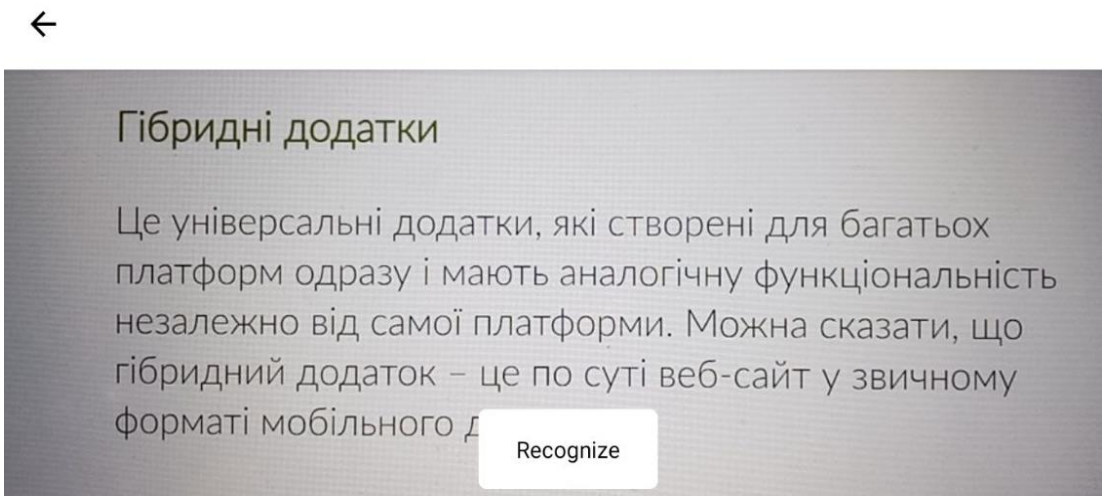

Рисунок3.5 - Тестовий текст для перевірки роботи програми з некоректними символами.

- Спрогнозуймо декілька сценаріїв ймовірних результатів:
- Програма не впізнає жодного символу і видасть пустий результат;
- Програма розпізнає деякі символи, які співпадають з символами в її бібліотеці і в кінцевому результаті видасть лише їх;
- Програма буде намагатися розпізнати всі символи, підбираючи найбільш подібні до вхідних даних .

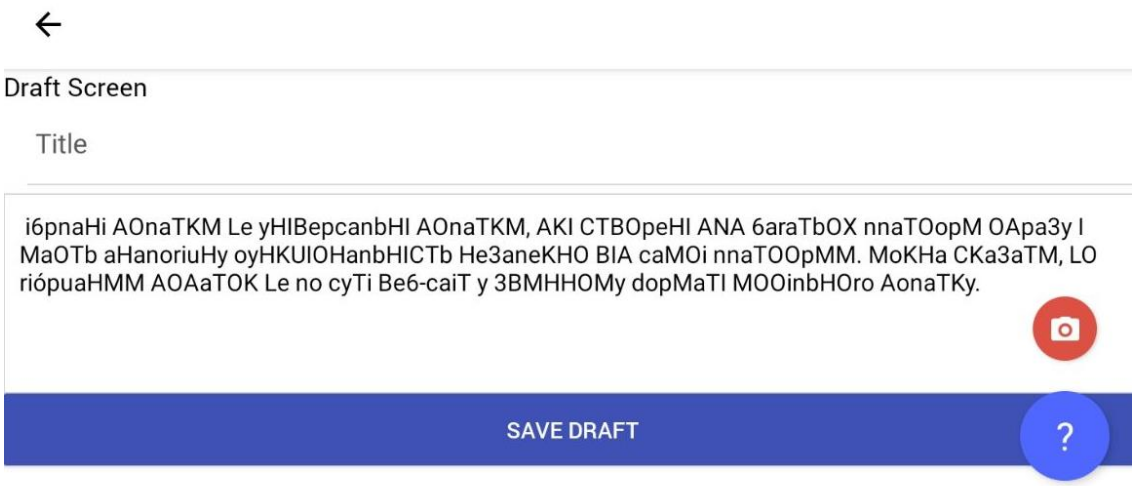

Рисунок3.6 – Результат що видала програма після перевірки тексту з некоректними символами.

Як результат, на виході ми маємо нечитабельний набір символів, що відповідає третьому прогнозованому сценарію. Замінивши специфічні елементи кирилиці на доступні їй аналоги, мережа в результаті зберегла кількість символів та їх розташування відносно сусідів.

Так як результат відповідає прогнозованому, на мою думку, тест можна вважати вдалим.

Вже на досягнутому етапі, мережа не втрачає данні, тож в потенціалі, при достатній кількості ресурсів (підключених бібліотек та баз даних), дана мережа зможе ефективно розпізнавати будь які символи.

ВИСНОВКИ

В кваліфікаційній магістерській роботі було проведено розробку інформаційної технології розпізнавання тексту з зображення для мобільного додатку. При цьому було виконано такі завдання виконати такі завдання:

1. Було детально розглянуто та порівняно різні методи та алгоритми розпізнавання символів.
2. Сформовано вхідний математичний опис системи інтелектуального аналізу зображень.
3. Проведено вибір типу та структури класифікатору зображень символів, що використовується інтелектуальною системою.
4. Розроблено алгоритми оптимізації функціональних параметрів інтелектуальної системи
5. Виконано програмну реалізацію функції інтелектуального аналізу зображень і її інтеграція в мобільний додаток. Для програмної реалізації обрано мову програмування JS. Навчання мережі реалізувалось з допомогою підключення бази даних NoSQL.
6. Проведено Перевірку працездатності розробленої інтелектуальної системи.

СПИСОК ЛІТЕРАТУРИ

1. Розенблат Ф. Принципи нейродинаміки: Персептрон та теорія механізмів мозку. Пер. с англ. – М.: Мир, 2017. - 175 с.

2. Головко В.А. Нейроінтеллект: Теорія і застосування. Книга 2. Самоорганізація, відмовостійкість та застосування нейронних мереж – Брест:БПИ, 2019, - 228с.

3. Хайкін, Саймон. Нейронні мережі: повний курс, 2-е видавнитство. : Пер.з англ. – М.: Видавництво дім «Вільямс», 2018. – 1104 с

4. Корсунов, Н.І. Метод зворотних перетворень в виявленні погрішностей при непрямих змінах / Н.І.Корсунов, А.А. Начетов // Научні Відомості БілДУ. Серія Історія. Політологія. Економіка. Інформатика. – 2019 № 8(151). – Випуск 26/1. – с. 104-107

5. Фаворська М.Н., Зотін А.С., Горошкін А.Н. Морфологічна обробка контурних зображень в системах разпізнавания текстових символів // Вісник СибГАУ. – 2017. – № 1 (14). – С. 70–75.

6. Бредихін Р.Н. Про один підхід до розпізнавання оптичних образів текстів // Вісник МЕИ. – 2018. – № 2. – С. 134–141.

7. Садихов Р.Х., Ваткін М.Є. Модифікований алгоритм навчання РБФмережі для розпізнавання рукописних символів // Ідентифікація образів.– 2022. – Т. 1. – № 3. – Р. 7–16.

8. Нгуен Тоан Тханг, Спіцин В.Г. Алгоритмічне та програмне забезпечення для розпізнавання форми руки в реальному часі з використанням SURFдескрипторів та нейронної мережі // 2019. – Т. 320. – № 5. – С. 48–54.

9. Буй Тхі Тху Чанг, Фан Нгок Хоанг, Спіцин В.Г. Розпізнавання осіб на основі застосування методу Віоли Джонса, вей в перетворення і методу головних компонент //2020. – Т. 320. – № 5. – С. 54–59.

10. Болотова Ю.А., Спіцин В.Г., Фомін А.Е. Застосування моделі ієрархічної тимчасової пам'яті в розпізнаванні зображень //2019.–Т. 318. – № 5. –С. 60–63.

11. Кермані Коланкех А., Спіцин В.Г., Хамкер Ф. Знаходження параметрів та видалення постійної складової фільтра Габора для обробки зображень // – 2019. – Т. 318. – № 5. – С. 57–59.

12. Plamondon R., Srinari S. Online and offline handwriting recognit ion: A comprehensive survey // IEEE Transactions on pattern analy sis and machine intelligence. – 2020. – V. 22. – P. 105–109.

13. Ronse C., Najman L., Decenciere E. Mathematical morphology: 40 years on // Proceedings of the VII International symposium of mathematical morphology. – Netherlands, 2017. – № 30. – P. 350–370.

14. Le Cun Y., Bengio Y. Convolutional networks for images, speech and time series// The handbook of brain theory and neural networks. – 2018. – V. 7. – № 1. – P. 255–258.

15. Rowley H.A., Baluja S., Kanade T. Neural networkbased face de tection // Pattern anal. mach. intell. – 2000. – V. 5. – P. 23–38.

16. Feraud R., Bernier O., Viallet J., Collobert M. A fast and accurate face detector based on neural networks // Transactions on pattern analysis and machine intelligence. – 2020. – V. 3. – № 23. – P. 42–53.

17. Yu N., Notkin B.S., Sedov V.A. Neuroiterative algorithm of tomo graphic reconstruction of the distributed physical fields in the fibre optic measuring systems // Computer optics. – 2019. – V. 33. – № 4. – P. 446–455.

18. Wilson D.R., Martinez T.R. The general inefficiency of batch trai ning for gradient descent learning

19. Іван Гудфелов, Йоша Бенгіо, Арон Коурвілле. «Машинне навчання» MIT Press, 2017. Режим доступу: http://www.deeplearningbook.org.

20. https://habrahabr.ru/company/recognitor/blog/221891/

21. https://habrahabr.ru/company/recognitor/blog/228195/

22. https://www.ibm.com/developerworks/ru/library/os-avto/

23. Лукошенко Г. Н. Розпізнавання скелетних образів – h]ttp://www.ocrai.narod.ru/skeletrecognize.html

24. Арлазаров Ст Л., Троянкер В.В., Котович Н.В. Адаптивне розпізнавання символів. – http://www.ocrai.narod.ru/adaptive.html.

25. Глушков В.М., Амосов Н.М., Артеменко И.А. Енциклопедія кібернетики. Том 1. Киев, 2019 г.

26. Візільтер Ю. В., Жовтов С. Ю., Князь В. А., Ходарєв А. Н., Моржин А. В. Обробка та аналіз цифрових зображень з прикладами на LabVIEW IMAQ Vision. - М.: ДМК Прес, 2017. – 464 с.

27. Стадник О.В. Дисертація. «Використання штучних нейронних мереж та вейвлет-аналізу для підвищення ефективності у завданнях розпізнавання та класифікації». Іванівський державний університет. Іванове, 2017р.

28. Саймон Хайкін. Нейронні сіті. Повний курс - М.: Вільямс, 2018. – 1104 стр.

29. Волков С.В., Колдов А.С., Захарова О.В., Чапаєв В.С. Система автоматичного контролю та керування параметрами об'єкта.

ДОДАТОК А

```
const { Layer, Network } = window.synaptic;

var inputLayer = new Layer(2);

var hiddenLayer = new Layer(3);

var outputLayer = new Layer(1);

inputLayer.project(hiddenLayer);

hiddenLayer.project(outputLayer);

var myNetwork = new Network({

 input: inputLayer,

 hidden: [hiddenLayer],

 output: outputLayer

});

// train the network - learn XOR

var learningRate = .3;

for (var i = 0; i < 20000; i++) {

  // 0,0 => 0

  myNetwork.activate([0,0]);

  myNetwork.propagate(learningRate, [0]);

  // 0,1 => 1

  myNetwork.activate([0,1]);

  myNetwork.propagate(learningRate, [1]);

  // 1,0 => 1

  myNetwork.activate([1,0]);

  myNetwork.propagate(learningRate, [1]);
```

```
  // 1,1 => 0

  myNetwork.activate([1,1]);

  myNetwork.propagate(learningRate, [0]);

}

console.log(myNetwork.activate([0,0]));

-> [0.015020775950893527]

console.log(myNetwork.activate([0,1]));

->[0.9815816381088985]

console.log(myNetwork.activate([1,0]));

-> [0.9871822457132193]

console.log(myNetwork.activate([1,1]));

-> [0.012950087641929467]
```

ДОДАТОК Б

```
 {
          "name": "rn_trApp",
          "version": "0.0.1",
          "private": true,
          "scripts": {
            "start": "react-native start",
            "test": "jest",
            "lint": "eslint ."
          },
          "dependencies": {
            "react": "16.8.6",
            "react-native": "0.60.5"
          },
          "devDependencies": {
            "@babel/core": "7.5.5",
            "@babel/runtime": "7.5.5",
            "@react-native-community/eslint-config": "0.0.3",
            "babel-jest": "24.9.0",
```

```
    "eslint": "6.2.1",
    "jest": "24.9.0",
    "metro-react-native-babel-preset": "0.54.1",
    "react-test-renderer": "16.8.6"
  },
  "jest": {
    "preset": "react-native"
  }
}
```