КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОЗПІЗНАННЯ ТЕКСТУ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ»

Виконав:
студент 4-го курсу бакалаврату
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Андрій Кузьмич _____

Науковий керівник:
кандидат технічних наук, асистент
Олександр Самощенко _____

Рецензент:
_____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

Реферат

Випускна кваліфікаційна робота бакалавра містить 66 сторінок, 25 рисунків, 1 додаток, використано 19 інформаційне джерело.

Об‘єкт дослідження – технологія оптичного розпізнавання символів на статичних зображеннях та зображеннях у реальному часі.

Мета роботи – розробити мобільний додаток на iOS платформі для розпізнавання тексту на статичних зображеннях та зображеннях у реальному часі.

В першому розділі роботи розглянуто процес розпізнавання символів, наведено основні методи розпізнавання та вказано головні переваги та недоліки кожного із них. Розглянуто основні нейронні мережі, які використовуються у процесі розпізнавання символів.

В другому розділі розглянуто популярні мови програмування, які дозволяються створювати мобільні додатки для iOS платформи, вказано на переваги та недоліки кожної з них. Проаналізовано методи реалізації розпізнавання символів як на комп’ютері, так і на мобільному телефоні, наведено переваги на недоліки кожної із них. Оглянуто існуючі системи розпізнавання тексту для комп’ютерів та для мобільних телефоні на iOS платформі та наведено переваги та недоліки.

В третьому розділі вибрано головні елементи для створення додатку. Обрано фреймворк за допомогою якого реалізовано функціонал розпізнавання тексту. Розроблено додаток на iOS платформі для розпізнавання тексту із статичних зображень та в режимі реального часу.

IOS-ЗАСТОСУНОК, ORC, SWIFT, CRAFT, РОЗПІЗНАВАННЯ ТЕКСТУ, SWIFTUI, MVVM, VISION, VISIONKIT.

ABSTRACT

The bachelor's thesis contains 68 pages, 25 figures, 1 appendix, and 19 information sources.

The object of research is the technology of optical character recognition in static and real-time images.

The purpose is to develop a mobile application on the iOS platform for recognizing text on static and real-time images.

The first section of the paper describes the process of character recognition, presents the main recognition methods, and indicates the main advantages and disadvantages of each of them. The main neural networks and architectures used in the process of character recognition are discussed.

The second section discusses the popular programming languages that allow the creation of mobile applications for the iOS platform and points out the advantages and disadvantages of each of them. The methods for implementing character recognition both on a computer and on a mobile phone are analyzed, and the advantages and disadvantages of each are presented. The existing text recognition systems for computers and for mobile phones on the iOS platform are reviewed and their advantages and disadvantages are presented.

In the third section, the main elements for creating an application are selected. A framework is chosen to implement the text recognition functionality. An application on the iOS platform for text recognition from static images and in real-time is developed.

IOS APPLICATION, ORC, SWIFT, CRAFT, TEXT RECOGNITION, SWIFTUI, MVVM, VISION, VISIONKIT.

Зміст

ВСТУП

У сучасному світі технологія оптичного розпізнавання символів (OCR) активно використовується для ефективного швидкого розпізнавання тексту та документообігу. Складність задач поставлених перед даною технологією різняться в залежності від типу пристроїв на яких її використовують.

На персональних комп’ютерах технологія оптичного розпізнавання символів застосується у сферах із великим об’ємом документів, таких як: архівація документів для сканування цифрових копій документів, автоматизація бізнес-процесів для розпізнавання інформації з рахунків, договорів та у фінансовій справі для розпізнавання банківських чеків, квитанцій, податкових документів. Від мобільних телефонів вимагається менш ресурсно затратна робота.

Мобільні телефони, а особливо смартфони стали невід’ємним об’єктом користування людства. Однією із актуальних задач для телефону є саме розпізнавання тексту. На відміну від персональних комп’ютерів смартфони не мають значного запасу ресурсів, отже і технологія OCR використовується для не ресурсно затратних завдань. Головним завданням розпізнавання тексту на телефоні є швидко розпізнати необ’ємний шматок тексту із статичного зображення, або в режимі реального часу. Найчастіше це розпізнати невеликий текст у побутових справах: номер банківської карти, номер телефону, імейл адресу, адресу місцерозташування та інші.

У дипломній роботі розроблено мобільний застосунок для швидкого сканування невеликих об’ємів текстів на платформі iOS із статичних зображень і в режиму реального часу.

РОЗДІЛ 1

Огляд основних методів оптичного розпізнавання символів

1.1 Оптичне розпізнавання символів

Оптичне розпізнавання символів — це механічний або електронний процес перетворення зображень рукописного, машинописного або друкованого тексту в послідовність кодів для наступного відображення в текстовому редакторі. Оптичне розпізнавання має широкий спектр застосувань: перетворення книг і документів в електронну форму, автоматизації систем бухгалтерського обліку або публікації тексту на веб-сторінках [2]. З текстом, перетвореним в електронну форму за допомогою оптичного розпізнавання, можна виконати редагування, пошук слова чи фрази, зберегти, відобразити чи надрукувати матеріал у більш компактній формі без втрати якості, аналізувати інформацію та застосовувати електронний переклад у текст, форматувати або конвертувати в мову. Оптичне розпізнавання тексту - це прогресивна, активно досліджуваний технічний напрям, який нерозривно пов'язаний з розпізнаванням зображень, мови та комп'ютерного зору [1].

Перші системи оптичного розпізнавання тексту вимагали постійного калібрування для роботи з кожним конкретним шрифтом; у попередніх версіях потрібно було розпізнавати зображення кожного символу. Програма може використовувати лише один шрифт одночасно. У 1970 роки створені «розумні» системи, здатні з високою точністю розпізнавати більшість відомих зображень і шрифтів [4]. Наступним етапом стала поява систем, здатних не тільки розпізнавати текст із значною якістю, а й відновлювати вихідний формат тексту, включаючи стовпці, таблиці та інші найпростіші графічні компоненти.

Сучасні технології дозволяють автоматично перетворювати великі обсяги паперових документів в цифрову форму за допомогою сканерів і подальшої цифрової обробки.

1.2 Процес розпізнавання символів

OCR працює за схожим принципом як здатність людини читати текст і розпізнавати візерунки та символи. Зазвичай люди читають текст, а потім витягують необхідну інформацію, вручну вводячи дані в систему, файл даних або базу даних. В свою чергу, технологія OCR покращує якість відсканованого тексту чи зображення та виконує кілька етапів для вилучення отриманих даних.
Увесь процес розпізнавання символів передбачає 6 етапів [3]:

- Отримання зображення
- Попередня обробка
- Сегментація
- Витяг особливостей
- Розпізнавання символів
- Постобробка

На рисунку 1.1 – схематично показано процес оптичного розпізнавання символів.

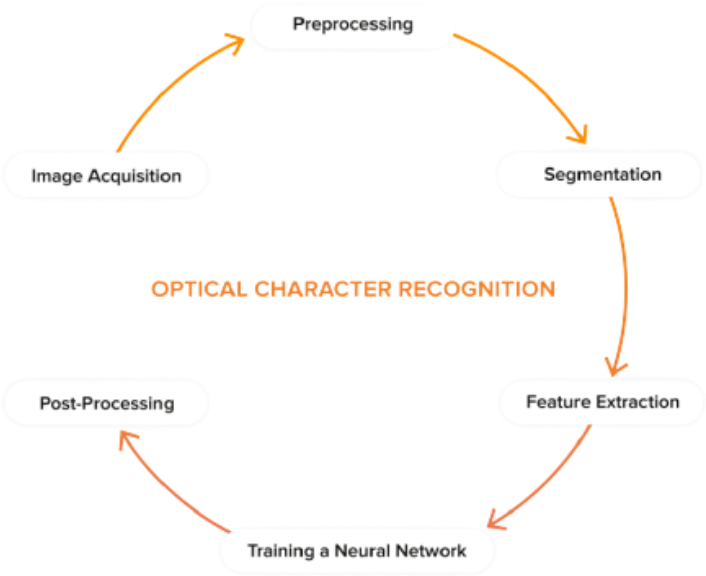

Рисунок 1.1 – Процес оптичного розпізнавання символів

Отримання зображення та попередня обробка

Зображення може бути зроблене за допомогою камери смартфону, або вибрана із галереї користувача. Для обробки зображення системою OCR потрібно провести попередню обробку.

Попередня обробка зображення включає:

- Виправлення перекосу - зображення, отримане на попередньому етапі, може бути неправильно орієнтоване, воно може бути вирівняне під будь-яким кутом. Тому необхідно виконати корекцію перекосу, щоб переконатися, що зображення, яке пересилається на наступні етапи, правильно орієнтоване.
- Бінаризація - перетворення кольорового зображення на двійкове (що містить лише чорно-білі кольори). Зазвичай на практиці це перетворення кольорового зображення у двійкове виконується за допомогою проміжного зображення в градаціях сірого .
- Видалення шуму - шум (маленькі крапки або компоненти переднього плану) може з’явитись в зображення під час його сканування під час отримання зображення через різні причини, як-от низька чіткість камери, тінь на зображенні тощо. Цей шум слід видалити, щоб зображення буде чистим і однорідним.

- Розрідження та скелетонізація- різні зображення мають текст із різною шириною штрихів. Ця мінливість дуже висока у випадку рукописних слів. Скелетонізація - це техніка, за допомогою якої можемо зробити всі штрихи однаковими за шириною (можливо, 1 піксель завширшки або кілька пікселів завширшки)

Сегментація

Після отримання чистого зображення після етапу попередньої обробки наступним етапом є сегментація . Це важливий етап розпізнавання, який передбачає розбиття текстових областей на зображенні на окремі символи або слова. Мета сегментації символів - визначити межі між символами та розділити їх на окремі області, які можуть бути розпізнані та інтерпретовані системою OCR.

Існує кілька методів сегментації символів, залежно від складності тексту та характеристик вхідного зображення. Деякі з найпоширеніших методів включають:

- Сегментація пробілів: Цей метод передбачає використання пробілів між символами як підказки для їх розділення. Це може бути ефективним підходом для тексту, надрукованого однаковим шрифтом і макетом, з чіткими пробілами між символами.
- Аналіз зв'язаних компонентів: Цей метод передбачає аналіз зв'язку між сусідніми пікселями зображення для виявлення кластерів пікселів, які відповідають окремим символам. Це може бути ефективним підходом для розпізнавання рукописного тексту або тексту, надрукованого складним шрифтом з символами, що перекриваються.
- Виявлення контурів: Цей метод передбачає виявлення контурів окремих символів за допомогою виявлення країв і морфологічних операцій. Це може бути ефективним підходом для тексту, надрукованого однорідним шрифтом з чіткими краями та межами.
- Сегментація на основі машинного навчання: Цей метод передбачає навчання моделі машинного навчання розпізнавати межі між символами на

основі таких характеристик, як текстура, форма та колір. Це може бути ефективним підходом для тексту, надрукованого різними шрифтами та стилями.

Витяг особливостей

На цьому етапі витягуються деякі унікальні особливості сегментованих під компонентів, отриманих на попередньому етапі. Є багато методів, за допомогою яких можна виділити такі особливості, як його форма, штрихи тощо. Існує кілька методів, які використовуються для вилучення ознак, залежно від складності тексту та характеристик вхідного зображення. Деякі з найпоширеніших методів включають:

- Гістограма орієнтованих градієнтів (HOG): Цей метод передбачає аналіз орієнтації та градієнта пікселів на зображенні для ідентифікації форми та структури символів. Це може бути ефективним підходом для тексту, який надруковано однорідним шрифтом з чіткими краями та межами.
- Масштабно-інваріантне перетворення ознак (SIFT): Ця техніка передбачає визначення характерних точок або областей на зображенні, які є інваріантними до масштабу, повороту та перекладу. Це може бути ефективним підходом для тексту, надрукованого різними шрифтами та стилями.
- Локальні бінарні патерни (LBP): Ця техніка передбачає аналіз текстури символів шляхом порівняння значень інтенсивності сусідніх пікселів. Це може бути ефективним підходом для тексту, який надруковано однорідним шрифтом з чіткими візерунками та текстурами.
- Згорткові нейронні мережі (CNN): Цей метод передбачає навчання моделі глибокого навчання для автоматичного вивчення та вилучення найбільш релевантних особливостей символів тексту. Це може бути ефективним підходом для тексту, надрукованого різними шрифтами і стилями, а також для розпізнавання рукописного тексту.

Розпізнавання символів

Розпізнавання символів - це процес ідентифікації та розпізнавання окремих символів у зображенні тексту. Це важливий етап розпізнавання, оскільки він дозволяє системі перетворити зображення тексту в цифровий формат, який можна обробляти та аналізувати.

Існує кілька методів розпізнавання символів, залежно від складності тексту та характеристик вхідного зображення. Ось деякі з найпоширеніших методів:

- Зіставлення з шаблоном: цей метод передбачає порівняння вхідного зображення символу з базою даних відомих символів для пошуку найкращого збігу. Це може бути ефективним підходом для тексту, надрукованого однаковим шрифтом з чіткими краями та межами.
- Нейронні мережі: Цей метод передбачає навчання моделі глибокого навчання для розпізнавання та класифікації окремих символів на основі їхніх особливостей, таких як форма, розмір і текстура. Це може бути ефективним підходом для тексту, надрукованого різними шрифтами та стилями, а також для розпізнавання рукописного тексту.
- Приховані марковські моделі (HMM): Цей метод передбачає моделювання розподілу ймовірностей послідовності символів у тексті та використання цієї моделі для виведення найімовірнішої послідовності символів на вхідному зображенні. Це може бути ефективним підходом для тексту, надрукованого однаковим шрифтом з чіткими шаблонами та структурами.
- Машини опорних векторів (SVM): Цей метод передбачає навчання моделі машинного навчання для класифікації вхідного зображення в одну з декількох категорій на основі його особливостей. Це може бути ефективним підходом для тексту, надрукованого різними шрифтами та стилями.

Постобробка

Найімовірніші помилки, які можуть виникнути в системі OCR, пов’язані з неправильним передбаченням на етапі класифікації (це може бути наслідком поганого виділення ознак, великої кількості шумів на зображенні тощо). У більшості випадків ці помилки передбачення призведуть до невеликих орфографічних помилок через неправильне передбачення однієї чи двох літер у слові. Таким чином, ці типи орфографічних помилок можна виправити за допомогою мовних моделей, моделей Word2Vec (наприклад, CBOW і skip-gram) тощо.

1.3 Методи розпізнавання символів

Шаблонні методи

Алгоритм шаблонного методу заснований на зіставленні вхідного графічного зображення з ідеальним шаблоном в базі даних. Першим етапом методу шаблону є перетворення отриманого зображення одного символу в растр. У процесі розпізнавання шаблони сортуються та обчислюється відстань від зображення до шаблону. Результатом розпізнавання є клас, шаблон у якому найменша кількість точок, відмінних від вхідного зображення.

Такі методи поділяються на дві категорії: залежні від шрифту та незалежні від шрифту. Підхід, незалежний від шрифтів, використовує попередньо визначені шаблони та є загальним для всіх типів шрифтів. Однак при такому підході ймовірність правильної ідентифікації зменшується.

Шрифтозалежні алгоритми розроблені тільки для одного шрифту, що покращує якість їх роботи, але вони абсолютно неефективні при використанні інших шрифтів. Запропоновано метод розпізнавання тексту великого об’єму, за умови, що деякі символи розпізнаються незалежним від шрифту методом, на основі розпізнаних символів створюється шаблон алгоритму, пов’язаного зі шрифтом[5] .

Перевагами такого методу є:

- Простота реалізації;
- Надійна робота за відсутності перешкод;
- Висока точність визначення дефектних символів;
- Швидкість розпізнавання маленьких літер.
- До недоліків можна віднести:
- Залежність від шаблонів і труднощі у виборі найкращого шаблону;
- Не розпізнає шрифти, відмінні від вбудованих у систему;
- Повільна робота та багато перешкод;

- Чутливість до обертання, шуму та спотворень.

Структурні методи

Алгоритми цих методів полягають у розбиття вихідного зображення на складові, які можна описати як стійкі та ідеальні елементи.

Деякі з методів даного напрямку використовують для розпізнавання топологічних описів зображень. Іншими словами, стандарт містить інформацію про взаємне розташування окремих компонентів . Водночас розмір символів, що розпізнаються, і шрифт при друку вже не важливі. Зображення, які можуть становити той чи інший клас, можна уявити як результат гомеоморфного перетворення еталонного образу, відповідного цього класу .

Завдання розпізнавання полягає у встановленні ізоморфізму між зображенням, що пред'являється, і одним з еталонних зображень. За допомогою фазового інваріанту можна знайти властивості зображення, яке не змінюється при синфазному перетворенні. Інваріантом, що дозволяє дати числовий опис зображення, є, наприклад, кількість ліній, що сходяться до точки. Відповідний опис виходить шляхом обходу порядку контуру зображення з одночасним фіксуванням точок.

Встановлення гомеоморфізму (самосвідомості) зводиться до зіставлення опису образу з еталонним чином класу. Важливою перевагою топологічного опису є несприйнятливе до сильних деформацій зображення, включаючи всі перетворення подібності, за умови, що кожному зображенню зіставлено певну характерну точку, з якої починається обхід. Вивчення коду топології полягає у підтримці набору еталонних описів.

Інші структурні методи реалізують алгоритми розпізнавання подій. Подієві методи засновані на топологічній структурі об'єкта. Він складається з ліній і змінюється при невеликих деформаціях зображення. Лінії є частиною зображення, і кожна секція має лише один інтервал. Жирні лінії у деяких сітках визначають події. Уявлення події — це формальний набір функцій, а й

правильне опис топології. Вивчення цього зводиться до складання списку бенчмарків для досить великого набору зображень.

Перевагами цього методу є:

- Стійкість до різних стилістичних варіацій шрифтів
- Стійкість до зсуву символу до невеликих кутів
- Розпізнавання спотворених символів
- Швидка обробка символів у нижньому регістрі

Основними проблемами таких методів є розпізнавання знаків що мають дефекти (наприклад, розрив лінії або злиття сусідніх символів) та ідентифікація символу при повороті понад десяти градусів.

Ознакові методи

Ознакові методи засновані на тому, що зображення порівнюються з N-вимірними векторами ознак. Розпізнавання полягає у порівнянні з набором еталонних векторів тієї ж розмірності. У рамках імовірнісних підходів існує низка математичних рішень для визначення приналежності зображення до одного із класів на основі аналізу обчислених ознак. Тип та кількість ознак суттєво впливає на якість розпізнавання. Формування вектора відбувається під час аналізу зображення. Така процедура називається вилученням ознак. Еталони кожного класу виходять аналогічною обробкою символів навчальних вибірок.

Переваги цього методу:

- Простота реалізації
- Висока здатність до узагальнення
- Стійкість до морфінгу символів
- Швидкість.

Недоліками цього є:

- Нестійкість до різних дефектів зображення.
- Втрата інформації про символи на етапі отримання символів.

Методи засновані на використанні нейронних мереж

Ідея даних методів полягає у моделюванні функціонування людського мозку. На вхід попередньо навченої нейронної мережі надходять вектори, що становлять вхідне зображення (пікселі, частотні характеристики , вейвлети). На виході нейрон, що відповідає класу символів, що розпізнаються, видає максимальне значення функції активації. Крім того, багато важливих характеристик зображення виводяться та обробляються іншими системами. Нейронна мережа навчається на ряді навчальних прикладів. Крім того, можливе навчання вчителем (Персептрон) чи самоорганізація (мережа Кохонена) [6]. На рисунку 1.2 як приклад схематично показана двошарова нейронна мережа, що містить 35 входів, 26 виходів (кількість літер) і 10 нейронів у прихованому шарі. Як функцію активації для нейронної мережі використовується сигмоїдальна функція, а її вихідні значення знаходяться в діапазоні від 0 до 1 і зручно переводяться в булеву алгебру.

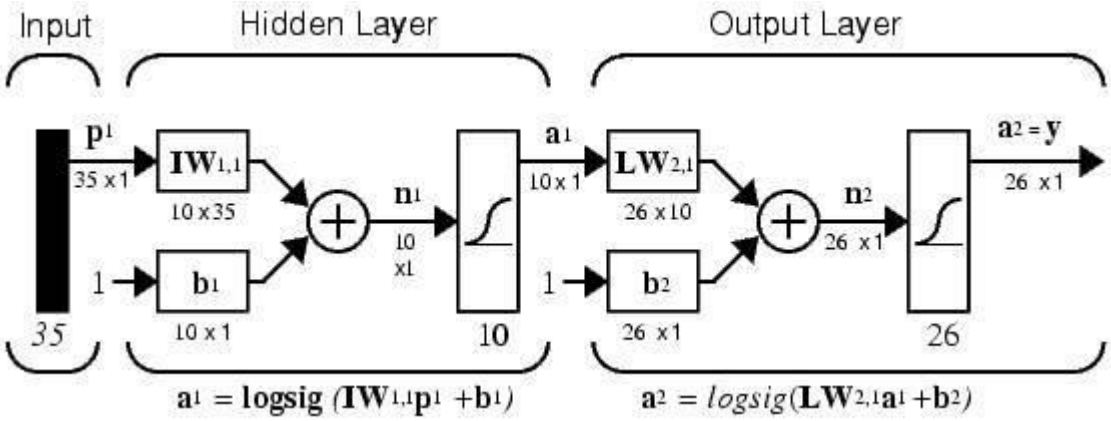

Рисунок 1.2 – Приклад двошарової нейронної мережі

Існує кілька алгоритмів порівняння тексту із стандартами.

Найпростіший варіант – попіксельне порівняння, але для порівняння вам потрібний рівень розміру зображення. Іншими варіантами є «Накладання» та «Накладання зі зміщенням», коли зображення вирівнюються один з одним. Методи еталонного порівняння рукописних та друкованих текстів схожі, але дуже різняться. Рукописні стандарти, на відміну від шрифтових знаків, є лише грубими зразками, що значно збільшує ймовірність помилки.

Перевагами даного методу можна назвати:

- Здастність до узагальнення
- Висока шкидкість роботи

Недоліками методу є:

- Чутливість до обертання та спотворення символів
- Складання підбору навчальної вибірки та алгоритму навчання

1.4 Нейронні мережі для розпізнавання тексту

Згортково рекурентна нейронна мережа

Згортково рекурентна нейронна мережа (CRNN) - це архітектура глибокого навчання, яка поєднує CNN (Згорткова нейронна мережа) з RNN(Рекурентна нейронна мережа) для вилучення ознак із вхідного зображення та декодування послідовності ознак для отримання вихідного тексту [9].

Згорткові нейронні мережі - це тип нейронних мереж, які використовують операцію згортки (ковзання фільтра по зображенню) для того, щоб виділити релевантні ознаки. Вони краще працюють з даними (ніж звичайні щільні нейронні мережі), в яких існує сильна кореляція між, наприклад, пікселями, оскільки не втрачається просторовий контекст. Цей тип нейронних мереж використовують фільтри для того, щоб виділити ознаки. Фільтри - це матриці, які "ковзають" по зображенню. Вони модифікуються під час навчання для того, щоб виділити найбільш релевантні ознаки [8].

У той час як згорткові нейронні мережі допомагають виокремлювати релевантні ознаки на зображенні, рекурентні нейронні мережі допомагають NNet враховувати інформацію з минулого для того, щоб робити прогнози або аналізувати.

Переваги згортково рекурентних нейронних мереж:

- Використання згорткових шарів для ефективного вилучення корисних ознак із зображень, які можна використовувати для різноманітних завдань, таких як класифікація зображень, виявлення об'єктів і розпізнавання тексту.
- Оскільки CRNN використовує згорткові шари, вона є стійкою до шуму і спотворень зображень, що робить її корисним інструментом для таких завдань, як оптичне розпізнавання символів (OCR) і розпізнавання мови.
- Використання рекурентних шарів для обробки послідовностей різної довжини, що робить її корисним для задач, які передбачають обробку вхідних даних змінної довжини, наприклад, розпізнавання мови.
- Наскрізне навчання, тобто вся мережа навчається спільно, що може призвести до кращої продуктивності порівняно з навчанням різних компонентів мережі окремо.

Недоліки:

- Велика кількість параметрів, що може зробити її обчислювально затратною і вимагати великого обєму даних для навчання.
- Для досягнення оптимальної продуктивності потребує ретельного налаштування пареметрів, таких як кількість згорткових та рекурентних шарів, розмір прихованих станів та швидкість навчання.

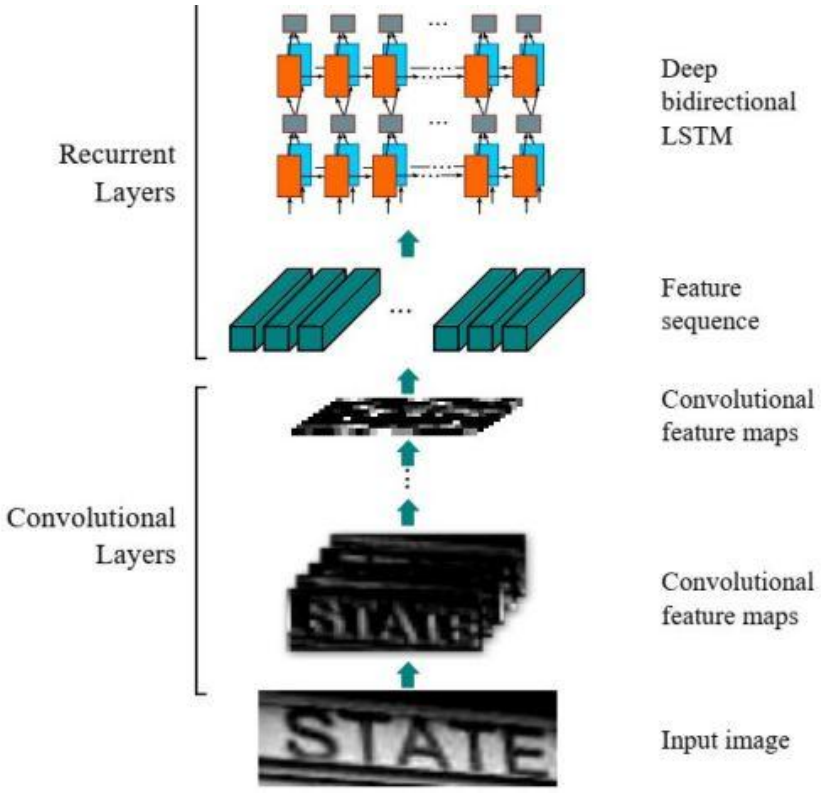

Рисунок 1.3 - Згортково рекурентна нейронна мережа

Повністю згорткова нейронна мережа

Повністю згорткова нейронна мережа (FCN) - це архітектура, яка використовує лише згорткові шари для вилучення ознак із вхідного зображення. Вона широко використовується для таких задач, як сегментація зображень, виявлення об'єктів і розпізнавання тексту.

FCN працює шляхом обробки вхідного зображення через серію згорткових шарів, де кожен шар застосовує набір фільтрів до вхідної карти ознак для вилучення все більш складних і абстрактних ознак. Результатом кожного шару згортки є нова карта ознак, яка кодує наявність і розташування певних візуальних патернів на вхідному зображенні.

У повністю згорткової нейронної мережі кінцевий згортковий шар створює карту ознак з просторовою роздільною здатністю, нижчою, ніж у вхідного зображення, завдяки застосуванню шарів об'єднання, які зменшують дискретизацію карти ознак. Для отримання результату з такою ж роздільною здатністю, як і вхідне зображення, FCN використовує процес, який називається підвищенням дискретизації. Підвищення дискретизації досягається шляхом

застосування шарів згортки до карти об'єктів, які підвищують її просторову роздільну здатність.

Переваги повністю згорткової нейронної мережі:

- Наскрізне навчання, тобто вся мережа навчається спільно, що може призвести до кращої продуктивності порівняно з навчанням різних компонентів мережі окремо.
- FCN використовує згорткові шари для ефективного вилучення корисних ознак із зображень, які можуть бути використані для різних завдань, таких як виявлення об'єктів і семантична сегментація.
- Стійкість до масштабу зображення так, як FCN використовує згорткові шари, що робить його корисним інструментом для таких задач, як виявлення об'єктів.
- Швидке отримання результату, оскільки повністю складається зі згорткових шарів, які можна ефективно розпаралелити.

Недоліки:

- Потребує великих обсягів даних, щоб навчитися добре представляти об'єкти та їхні межі на зображеннях.
- Може страждати від перенавчання, особливо при навчанні на невеликих наборах даних, що може призвести до поганої продуктивності узагальнення на нових даних.
- Має обмежену просторову роздільну здатність через зменшення вибірки, яка відбувається під час вилучення ознак. Це може ускладнити сегментацію невеликих об'єктів або об'єктів з дрібними деталями.

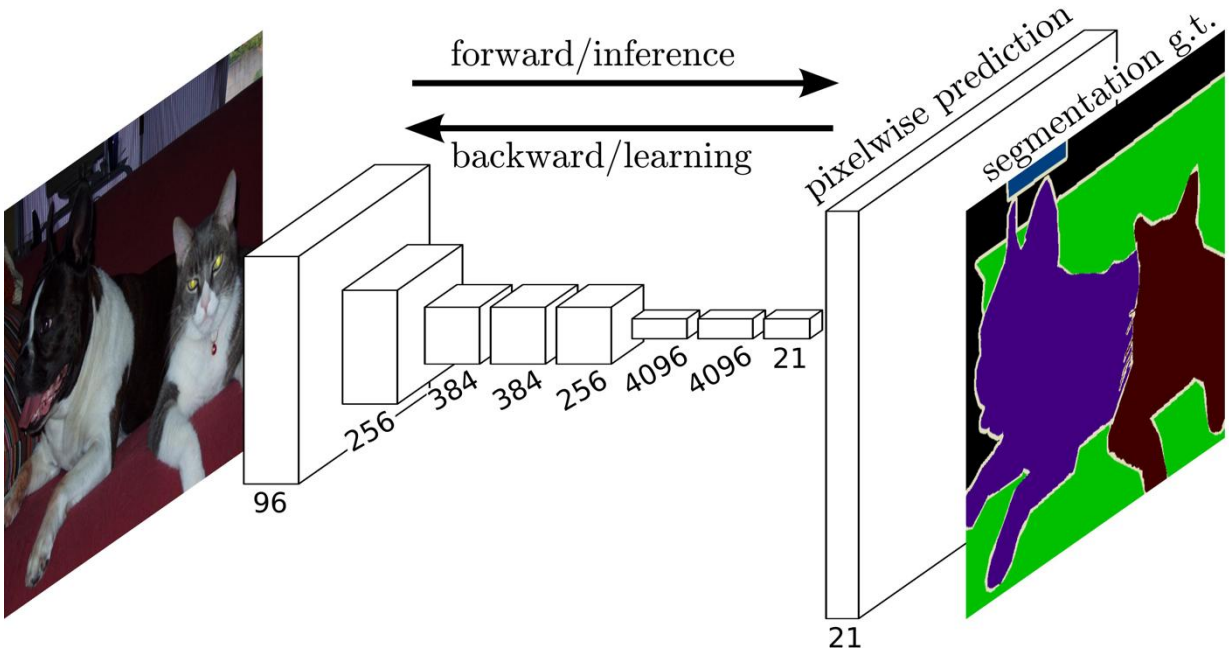

Рисунок 1.4 - Повністю згорткова нейронна мережа

CRAFT

CRAFT - це глибока нейронна мережа, яка прогнозує дві оцінки (Region score і Affinity score) для кожного символу тексту. Спочатку генерується карта ознак із вхідного зображення за допомогою згорткової нейронної мережі (CNN). Потім ця карта обробляється повністю згортковою мережею (FCN), яка складається з декількох шарів розширених згорткових фільтрів. Розширені згорткові фільтри дозволяють моделі фіксувати ознаки в різних масштабах і роздільній здатності, що важливо для виявлення текстових областей, які можуть відрізнятися за розміром і формою [8].

Результатом роботи FCN є теплова карта, яка показує ймовірність належності кожного пікселя до текстової області. Потім теплова карта обробляється для отримання бінарного зображення, де білі пікселі відповідають текстовим областям, а чорні пікселі - фону.

Щоб виділити окремі символи з текстових областей, CRAFT використовує мережу пропозицій символів, яка передбачає набір обмежувальних рамок, що щільно оточують кожен символ. Мережа пропозицій навчається за допомогою комбінації позитивних і негативних зразків, щоб навчитися розрізняти регіони символів і несимвольні регіони[7].

Переваги CRAFT:

- CRAFT досягає високої якості розпізнавання за рахунок різноманітних еталонних наборів даних для розпізнавання тексту.
- Стійка до змін якості зображення, таких як розмиття та нерівномірне освітлення.
- Ефективно виявляє текст на зображеннях, що робить його корисним для додатків, які працюють у режимі реального часу.
- Точна локалізація текстової області на зображеннях, що важливо для таких завдань, як розпізнавання тексту.

Недоліки:

- Вимагає великих обсягів даних, щоб навчитися добре розпізнавати текстові області на зображеннях.
- Так як це архітектура глибокого навчання, тому вона вимагає значних обчислювальних витрат, що може ускладнити її роботу на пристроях з обмеженими ресурсами.

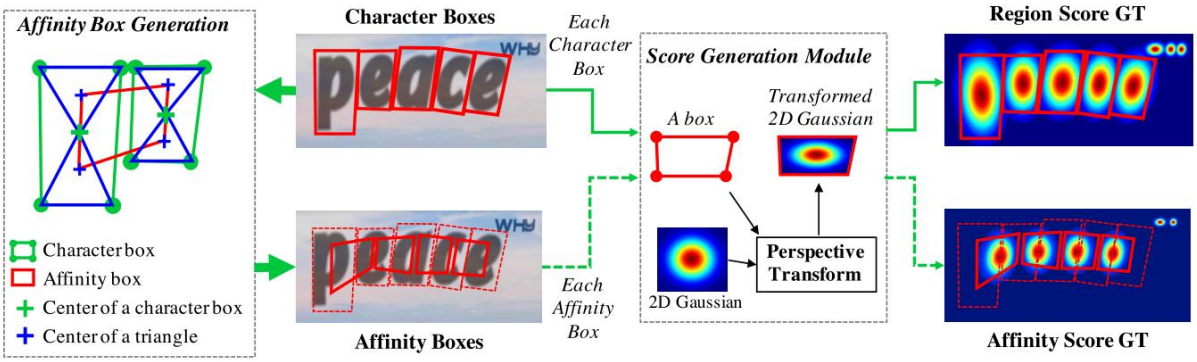

Рисунок 1.5 – Робота нейронної мережі CRAFT

BERT

BERT (Bidirectional Encoder Representations from Transformers) - це попередньо навчена нейромережева архітектура на основі Transformer для задач обробки природної мови (NLP). Це потужна модель, яка може бути налаштована для широкого спектру завдань NLP, таких як питання-відповідь,

аналіз настрою та розпізнавання іменованих об'єктів. BERT є двонаправленою моделлю, яка може обробляти всю вхідну послідовність одночасно, а не покладатися на RNN або CNN для послідовної обробки послідовності.

BERT використовує архітектуру Transformer, яка складається з декількох шарів самоуваги та нейронних мереж зі зворотним зв'язком. На відміну від спрямованих моделей, які зчитують введений текст послідовно (зліва направо або справа наліво), кодер Transformer зчитує всю послідовність слів одразу. Тому він вважається двонаправленим, хоча точніше було б сказати, що він ненаправлений. Ця характеристика дозволяє моделі вивчати контекст слова на основі всього його оточення (ліворуч і праворуч від слова)[10].

Переваги BERT:

- BERT досягає найвижчої результативності в широкому діапазоні завдань обробки мови.
- Двонаправлене мовне моделювання, що дозволяє враховувати контекст при обробці тексту, що призводить до кращої продуктивності при виконанні багатьох завдань.
- BERT пройшов попереднє навчання на декількох мовах, що дозволяє йому добре розпізнавати текст багатьма мовами.

Недоліки:

- Вимагає великих обсягів навчальних даних,щоб навчитися добре розпізнавати мову.
- Ресурсно затратна так, як є великою і складною моделлю, і її навчання і розгортання може вимагати значних ресурсів, що робить її складною для використання на пристроях з обмеженими ресурсами.
- Низька ефектривність розпізнавання рідкісних слів. оскільки вони можуть з'являтися в навчальних даних недостатньо часто, щоб їх можна було ефективно вивчити.

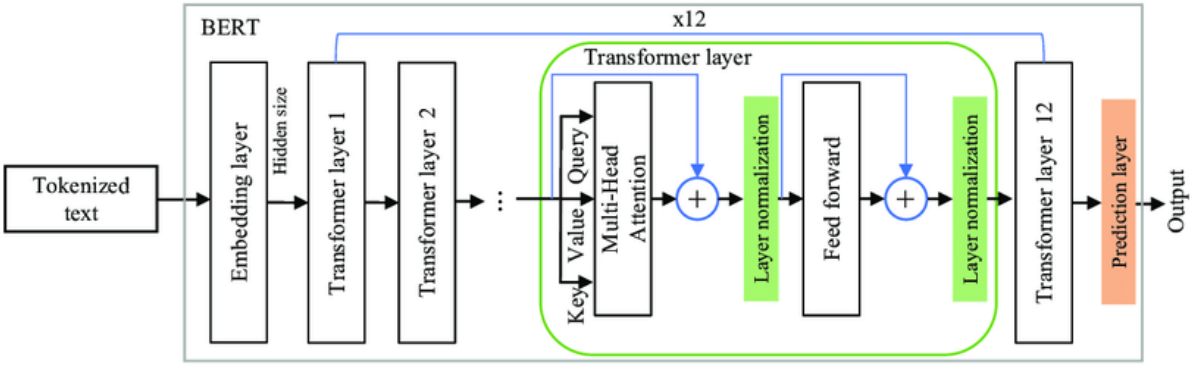

Рисунок 1.6 – Робота нейронної мережі BERT

1.5 Висновки до розділу

За результатами аналітичного огляду встановлено, що процес розпізнавання символів передбачає застосування дій на наступних основних етапах: отримання зображення, попередня обробка, сегментація, витяг особливостей, розпізнавання символів та постобробка.

Наведено основні методи розпізнавання символів та вказано головні переваги та недоліки кожного із них. Розглянуто основні нейронні мережі та архітектури, які використовуються у процесі розпізнавання символів. В ході аналізу вирішено використовувати метод розпізнавання символів на основі нейронної мережі CRAFT. Дана нейронна мережа досягає високої якості розпізнавання тексту та є стійкою до зображень з низькою якістю, що є актуальною проблемою при розпізнаванні тексту із зображень створених камерою смартфону.

Розділ 2

ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ МОБІЛЬНОГО ЗАСТОСУНКУ ТА ОГЛЯД АНАЛОГІВ

2.1 Огляд мов програмування

Swift

Swift - це мова програмування, розроблена компанією Apple у 2014 році для розробки програмного забезпечення для iOS, macOS, watchOS та tvOS [11]. Створена для заміни Objective-C, мови, яка використовувалася для розробки програмного забезпечення для продуктів Apple з 1983 року. Swift має сучасний синтаксис та вбудовану підтримку для багатьох функцій, які допомагають писати безпечний та ефективний код. Swift також дозволяє легко інтегрувати об'єктно-орієнтований та функціональний підходи до програмування [13].

Swift розроблена для запобігання поширеним помилкам програмування, таким як нульові вказівники та неініціалізовані змінні. Мова надає такі функції, як «optional», що дозволяють розробникам безпечно працювати з нульовими значеннями, та оператори захисту, які дозволяють розробникам достроково вийти з функції, якщо певна умова не виконується.

Swift використовує автоматичний підрахунок посилань (ARC) для відстеження та керування використанням пам'яті вашого додатку. У більшості випадків це означає, що керування пам'яттю у Swift "просто працює", і вам не потрібно думати про керування пам'яттю самостійно. ARC автоматично звільняє пам'ять, яку використовують екземпляри класів, коли ці екземпляри більше не потрібні.

Переваги Swift:

- Рідна мова iOS
- Автоматичний підрахунок пам'яті
- Простий у вивченні та використанні

- Стабільна та надійна
- Регулярні оновлення
- Хороша масштабованість

Недоліки:

- Не так багато Swift розробників
- Обмежена підтримка спільноти у порівнянні з іншими мовами, такими як Python, Java та C++

C#

C# - популярна, сучасна та високорівнева мова програмування, розроблена компанією Microsoft у 2000 році. C# - це об'єктно-орієнтована мова, створена для того, щоб бути простою та ефективною у вивченні, що робить її популярним вибором для розробників, які працюють над широким спектром додатків.

Однією з ключових переваг C# є її здатність безперешкодно працювати з .NET Framework, всеосяжною та надійною платформою для розробки програмного забезпечення, яка надає широкий спектр бібліотек, інструментів та сервісів для спрощення розробки додатків. .NET Framework доступний для Windows, macOS та Linux, що робить його універсальним інструментом для створення крос-платформних додатків. Однак C# не розглядалася як мова програмування для додатків для iOS, поки не з'явився Xamarin.

Xamarin - це кросплатформенне середовище розробки, яке дозволяє розробникам використовувати C# та .NET Framework для створення нативних мобільних додатків для iOS, Android та Windows. Компанія Xamarin була заснована у 2011 році, а у 2016 році була придбана корпорацією Майкрософт.

Однією з ключових переваг Xamarin є можливість створювати додатки, які мають нативний вигляд на кожній платформі. Це досягається завдяки використанню Xamarin.iOS та Xamarin.Android, які забезпечують прив'язку до нативних API та елементів керування інтерфейсом користувача на кожній платформі. Це означає, що розробники можуть використовувати C# для

створення нативних користувацьких інтерфейсів, доступу до специфічних для платформи функцій та використання переваг продуктивності нативної розробки [14].

Переваги C#:

- Велика спільнота, оскільки це популярна і зріла мова.
- Відкритий вихідний код
- Висока продуктивність
- Має величезну бібліотеку
- Крос-платформна підтримка
- Масштабована та оновлювана

Недоліки:

- Є відносно важкою у навчанні для початківців
- Обмежена гнучкість, оскільки він базується на структурі Microsoft.NET

Objective-C

Objective-C - це високорівнева об'єктно-орієнтована мова програмування, яка в основному використовується для розробки додатків для iOS та macOS. Створена на початку 1980-х років Бредом Коксом і Томом Лавом і пізніше прийнята компанією Apple як основна мова для розробки додатків для операційних систем macOS та iOS.

Дана мова є розширенням мови C, яка включає в собі набір об'єктно-орієнтованих функцій, таких як класи, успадкування та динамічне зв'язування. Це динамічно типізована мова, що означає, що типи змінних визначаються під час виконання, а не під час компіляції. Це забезпечує більшу гнучкість та динамічну поведінку програми.

Однією з ключових переваг Objective-C є тісна інтеграція з фреймворками Cocoa та Cocoa Touch, які надають набір готових бібліотек та інструментів для розробки додатків для iOS та macOS. Ці фреймворки дозволяють розробникам легко створювати користувацькі інтерфейси,

обробляти користувацьке введення, отримувати доступ до функцій пристрою та виконувати широкий спектр інших завдань.

Головний недоліком є те, що це відносно застаріла мова програмування (востаннє її оновлювали у 2016 році). В iOS розробці її можна розглядати, лише для створення програми для старіших пристроїв iOS, які не підтримують Swift. Для створюворення програми для пристроїв iOS поточного та наступного поколінь, краще використовувати Swift.

Objective-C також як і Swift підтримує автоматичне керування пам'яттю за допомогою підрахунку посилань, що допомагає спростити керування пам'яттю та зменшити ймовірність витоку пам'яті та інших проблем, пов'язаних з пам'яттю.

Перевеги Objective-C:

- Це стабільна і зріла мова
- Підтримує динамічний набір тексту

Недоліки:

- Є важкою у навчанні для початківців
- Це застаріла мова і не достатньо безпечна
- Не має відкритого вихідного коду
- Обмежена функціональність
- Відсутність нових оновлень
- Низька ефективність розробки

Dart

Dart - це об'єктно-орієнтована мова програмування загального призначення з відкритим вихідним кодом і синтаксисом у стилі C, розроблена компанією Google у 2011 році. Розроблена як масштабована, високопродуктивна мова, яку можна використовувати для широкого спектру

додатків, включаючи веб- та мобільну розробку. Dart - це об'єктно-орієнтована мова, яка має строгу типізацію[12].

Однією з ключових переваг Dart є можливість компіляції як у власний машинний код, так і в JavaScript. Це робить її універсальною мовою, яку можна використовувати як для серверної, так і для клієнтської розробки. Для серверної розробки Dart можна використовувати з такими фреймворками, як Aqueduct та Angel, які надають інструменти для створення веб-додатків та API. Для клієнтської розробки Dart можна використовувати з такими фреймворками, як Flutter, який є фреймворком мобільної розробки для створення високопродуктивних, крос-платформних додатків.

Dart також підтримує асинхронне програмування завдяки використанню синтаксису async/await, що дозволяє розробникам писати код, який виконується паралельно, не блокуючи основний потік. Це полегшує написання коду, який може обробляти кілька завдань одночасно, наприклад, мережеві запити або введення даних користувачем.

На відміну від інших мов програмування, Dart підтримує більшість загальних концепцій мов програмування, таких як класи, інтерфейси, функції. Мова Dart не підтримує масиви безпосередньо, проте підтримує колекцію, яка використовується для реплікації структури даних, таких як масиви, узагальнення та необов'язкова типізація.

Dart також включає в себе ряд сучасних мовних функцій, таких як виведення типів, узагальнення та необов'язкові параметри. Ці функції допомагають спростити код і зменшити кількість шаблонів, що полегшує написання та підтримку складних додатків

Переваги Dart:

- Мультиплатформенна підтримка
- Відкритий вихідний код
- Легко створювати привабливі візуальні ефекти для вашого додатку
- Надає широку колекцію віджетів для кастомізації вашого додатку

Недоліки:

- Невелика спільнота розробників
- Невелика кількість бібліотек для полегшення розробки

2.2 Огляд засобів реалізації розпізнавання обличчя

Tesseract

Tesseract - це безкоштовний розпізнавач з відкритим вихідним кодом, розроблений компанією Google. Розроблений у 1980-х роках у лабораторіях Hewlett-Packard Laboratories, а згодом випущений як відкритий код у 2006 році. Відтоді Tesseract постійно вдосконалюється та підтримується спільнотою. Остання версія Tesseract, версія 4, випущена в 2018 році і включає значні покращення точності та продуктивності [15].

Переваги Tesseract:

- Відкритий вихідний код, який можна вільно використовувати та модифікувати.
- Здатний розпізнавати понад 100 мов, включаючи складні шрифти, такі як арабська, китайська та японська.
- Здатний розпізнавати різні типи шрифтів, включаючи машинописний і рукописний текст.
- За замовчуванням підтримує Unicode (UTF-8), багато форматів зображень, таких як PNG, JPEG і TIFF. Він також підтримує багато вихідних форматів, таких як PDF, TEXT файли, TSV і текст тільки для читання.
- Має режим навчання, який дозволяє користувачам навчати систему розпізнавати нові мови або типи шрифтів.
- Широко використовується і має велику та активну спільноту розробників, а це означає, що він постійно оновлюється та вдосконалюється завдяки новим функціям та виправленням помилок.

Недоліки Tesseract:

- Точність розпізнавання є нижчою ніж у комерційних систем розпізнавання, особливо для складних сценаріїв і розпізнавання рукописного тексту.
- На точність розпізнавання впливають такі фактори, як якість зображення, тип шрифту та макет.
- Повільно працює при обробці великих обсягів тексту або складних документів.
- Режим навчання вимагає значної кількості часу і зусиль для навчання рушія новим мовам або типам шрифтів.
- Документація може бути неповною або застарілою, що може ускладнити пошук інформації про конкретні можливості або функції.
- Вимагає додаткових кроків попередньої обробки, таких як покращення зображення або зменшення шуму, щоб підвищити точність розпізнавання.

OpenCV

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення для комп'ютерного зору та машинного навчання з відкритим вихідним кодом. Вперше розроблена компанією Intel у 1999 році і з тих пір підтримується спільнотою розробників. OpenCV написана на C++ і підтримує безліч мов програмування, включаючи Python, Java і MATLAB. Переваги OpenCV:

- Бібліотека з відкритим вихідним кодом, що означає, що її можна вільно використовувати та модифікувати.
- Надає широкий спектр функціональних можливостей для задач комп'ютерного зору, включаючи обробку зображень і відео, виявлення і розпізнавання об'єктів, вилучення ознак і машинне навчання.
- Написана на C++, але підтримує багато мов програмування, включаючи Python, Java і MATLAB, що робить його доступним для розробників з різними мовними уподобаннями.
- Крос-платформенна і може використовуватися на різних операційних системах, включаючи Windows, Linux, macOS, Android та iOS.

- Має велику та активну спільноту розробників, а це означає, що він постійно оновлюється та вдосконалюється завдяки новим можливостям та виправленням помилок.
- Має графічний інтерфейс користувача (GUI) для тестування і розробки програм комп'ютерного зору.

Недоліки OpenCV:

- Досить складна бібліотека, і її може бути важко вивчити розробникам без попереднього досвіду роботи з комп'ютерним зором.
- C++ API OpenCV може бути об’ємним і складним у використанні, що може забрати багато часу на розробку.
- Python API OpenCV, хоча і простіший у використанні, ніж C++ API, може працювати повільніше, ніж чиста реалізація на C++.
- Функціональність машинного навчання OpenCV не настільки розвинена, як у інших популярних бібліотек машинного навчання, таких як TensorFlow або PyTorch.
- Попередньо навчені моделі OpenCV для виявлення та розпізнавання об'єктів можуть бути не такими точними, як сучасні моделі, такі як YOLO або Mask R-CNN.
- Документація OpenCV може бути неповною або застарілою, що може ускладнити пошук інформації про конкретні можливості або функції.

Kraken

Kraken - це OCR-двигун, розроблений для розпізнавання історичних та пошкоджених текстів. Це розпізнавач з відкритим вихідним кодом, написаний на мові Python і заснований на системі розпізнавання OCRopus [15].

Kraken використовує нейронні мережі для сегментації та розпізнавання символів, що дозволяє працювати з різними типами і стилями шрифтів, а також з погіршеним або пошкодженим текстом. Kraken також може розпізнавати

текст, написаний різними шрифтами, що робить його корисним для розпізнавання тексту в багатомовних документах.

Переваги Kraken:

- Відкритий вихідний код, який можна вільно використовувати та модифікувати.
- Спеціально розроблений для розпізнавання історичних і пошкоджених текстів, що робить його корисним інструментом для оцифрування старих документів і книг.
- Використовує нейронні мережі для сегментації та розпізнавання символів, що дозволяє йому працювати з різними типами і стилями шрифтів, а також з деградованим або пошкодженим текстом.
- Підтримує розпізнавання різних шрифтів, зокрема латиницю, кирилицю, грецьку, арабську та індійську абетки, а також може розпізнавати текст, написаний кількома шрифтами.
- Має модульну архітектуру, що дозволяє розробникам налаштовувати і розширювати його функціональність.
- Містить навчальний модуль, який дозволяє користувачам тренувати сервіс для певних мов або типів шрифтів, що робить його корисним інструментом для розпізнавання тексту нестандартними шрифтами або мовами.

Недоліки Kraken:

- На якість розпізнавання фреймворку можуть впливати такі фактори, як якість зображення, тип шрифту та макет.
- Вимагає додаткових кроків попередньої обробки для підвищення точності розпізнавання, що може зробити робочий процес розпізнавання більш трудомістким.
- Націлений для використання у середовищі командного рядка, що може зробити його менш доступним для розробників без досвіду програмування.
- Інтерфейс користувача є менш інтуїтивно зрозумілим, ніж у аналогів, що може ускладнити його використання деякими користувачами.

- Вимагає більше навчальних даних і зусиль для навчання OCR-двигуну для певних мов або типів шрифтів порівняно з іншими реалізаціями OCR.
- Документація Kraken може бути не такою вичерпною, як в інших розпізнавальних системах, що може ускладнити її вивчення деякими користувачами.

ML Kit Text Recognition

ML Kit Text Recognition - це хмарний механізм розпізнавання тексту, розроблений компанією Google. Є частиною набору інструментів машинного навчання ML Kit і призначений для того, щоб допомогти розробникам додати функції розпізнавання тексту до своїх мобільних додатків.

Фреймворк використовує алгоритми машинного навчання для розпізнавання тексту в режимі реального часу на мобільних пристроях. Підтримує розпізнавання тексту різними мовами і типами шрифтів, включаючи рукописний текст.[16]

Переваги ML Kit Text Recognition:

- Фреймворк надає розробникам простий інтерфейс для додавання функції розпізнавання тексту до своїх програм за допомогою лише кількох рядків коду.
- Використовує алгоритми глибокого навчання для розпізнавання тексту в режимі реального часу на мобільних пристроях, що дозволяє підвищити якість розпізнавання порівняно з традиційними механізмами OCR.
- Підтримує розпізнавання текст різними мовами і типами шрифтів, включаючи рукописний текст, що робить його корисним для широкого спектру застосувань.
- Дозволяє використовувати як хмарний сервіс, що означає, що розпізнавання тексту може виконуватися на стороні сервера, зменшуючи навантаження на мобільний пристрій.
- Інтеграція з іншими інструментами ML Kit, такі як маркування зображень і розпізнавання облич, що дозволяє легко інтегрувати розпізнавання тексту з іншими функціями машинного навчання.

- Недоліки:

- Потрібне підключення до Інтернету для доступу до хмарного сервісу ML Kit Text Recognition, що може бути обмеженням у районах з поганим підключенням до Інтернету або там, де важливою є конфіденційність даних.

- Обмежені можливості кастомізації: фреймворк надає простий інтерфейс для доступу до розпізнаного тексту, але не може забезпечувати такого ж рівня кастомізації, як інші механізми розпізнавання або інструменти розпізнавання тексту.

- Обмежений вихідний формат: ML Kit Text Recognition надає розпізнаний текст у простому рядковому форматі, який може не підходити для програм, що вимагають складніших форматів виводу.

VisionKit

VisionKit - це фреймворк, розроблений компанією Apple для iOS та iPadOS, який надає інструменти для реалізації можливостей комп'ютерного зору та аналізу зображень у мобільних додатках. Фреймворк включає ряд функцій, таких як виявлення об'єктів, розпізнавання облич і розпізнавання тексту, що робить його корисним для широкого спектру додатків[17]. Переваги VisionKit:

- Інтегрується з Core ML, платформою машинного навчання Apple, що дозволяє додавати власні моделі машинного навчання для виконання таких завдань, як виявлення об'єктів або класифікація зображень.

- Підтримує ряд функцій, таких як виявлення об'єктів, розпізнавання облич і розпізнавання тексту.

- Обробка в реальному часі, що робить його придатним для додатків, які вимагають швидкого відгуку.

- Інтегрується зі новим медотодом написання інтерфейсу користувача від Apple - SwiftUI, що дозволяє легко створювати користувацькі інтерфейси, які включають можливості комп'ютерного зору та аналізу зображень. Недоліки VisionKit:

- Надає простий інтерфейс для доступу до розпізнаних об'єктів, проте кастомізації є недоступною так, як у інших фреймворків комп'ютерного зору або інструменти для аналізу зображень.
- Складність роботи з складними зображеннями, таких як зображення з низьким контрастом або складним фоном, порівняно з іншими системами комп'ютерного зору.
- Обмежена крос-платформенна підтримка,що обмежує його корисність для додатків, які потребують крос-платформної підтримки.

2.3 Огляд існуючих систем розпізнавання тексту

Abbyy FineReader

Abbyy FineReader – це комерційний продукт для оптичного розпізнавання символів, розроблений компанією ABBYY, безперечним лідером у галузі оптичного розпізнавання символів. FineReader заснований на запатентованій технології оптичного розпізнавання символів ABBYY OCR, яку використовують такі провідні світові компанії, як Fujitsu, Panasonic, Xerox та Samsung.

Переваги Abbyy FineReader:

- Висока точність в розпізнаванні тексту різними мовами та шрифтами, включаючи складні тексти та мови з декількома наборами символів.
- Можливість конвертувати широкий спектр типів документів у формати з можливістю пошуку та редагування, такі як Microsoft Word, Excel і PowerPoint. Ця функція особливо корисна для підприємств, організацій та приватних осіб, які мають справу з великими обсягами документів.
- Розширений аналіз документів такий як, аналіз макета сторінки, попередня обробка зображень і розпізнавання штрих-кодів. Ці функції полегшують обробку та аналіз документів, що може призвести до підвищення ефективності та продуктивності.
- Підтримує понад 200 мов

- Інтеграція з іншим програмним забезпеченням, таким як Microsoft SharePoint, щоб оптимізувати обробку документів і робочий процес.
- Зручний інтерфейс, що дозволяє користувачам легко орієнтуватися і використовувати його функції.

Недоліки:

- Не є безкоштовним програмним забезпеченням, і його ціна варіюється в залежності від варіанту ліцензування та необхідних функцій.
- Обмежена функціональність у версії для Mac порівняно з версією для Windows.
- Використовує великі об’єми ресурсів, особливо при обробці великих обсягів документів.
- Підтримка рукописного тексту обмежена порівняно з друкованим текстом.
- Відсутність мобільного додатку, що може бути незручним для користувачів, яким потрібно обробляти документи на ходу.

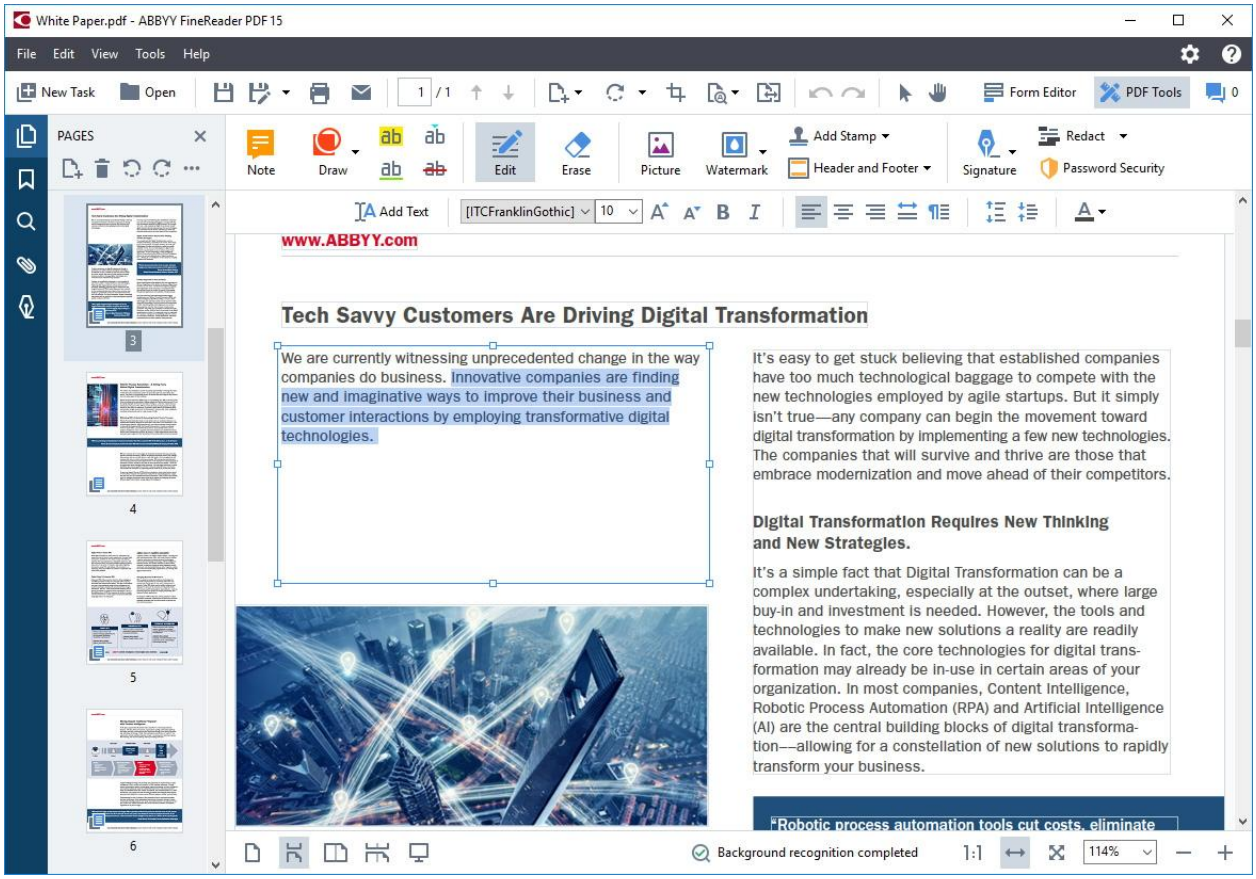

Рисунок 2.1 - Інтерфейс додатку Abbyy Fine Reader

OCRopus

OCRopus – система оптичного розпізнавання символів, спочатку спрямовану перетворення в електронний вигляд великого обсягу документів з урахуванням власного розпізнавального ядра. Це пакет програмного забезпечення для розпізнавання тексту з відкритим кодом, що розповсюджується під ліцензією Apache License 2.0.

Переваги OCRopus:

- OCRopus відомий своєю високою точністю розпізнавання тексту різними мовами та шрифтами, включно зі складними текстами та мовами з декількома наборами символів.
- Розширена обробка зображень, яка може покращити якість відсканованих документів і зображень, полегшуючи точне розпізнавання тексту.
- OCRopus дуже добре налаштовується, що дозволяє користувачам змінювати програму відповідно до своїх потреб, наприклад, додавати підтримку нових мов або шрифтів.
- Безкоштовний і з відкритим вихідним кодом, що робить його доступним для ширшого кола користувачів і дозволяє модифікувати вихідний код.
- Це інструмент командного рядка, що означає, що його можна використовувати в скриптах або інтегрувати в інші програми, допомагаючи оптимізувати обробку документів і робочий процес.

Недоліки:

- Відсутність графічного інтерфейсу користувача, що може бути недоліком для користувачів, які надають перевагу візуальному підходу до програмного забезпечення.
- Вимагає великого обєму ресурсів, особливо при обробці великих обсягів документів, що вимагає значних обсягів пам'яті та обчислювальної потужності.

LiveScan

LiveScan для iPhone, iPad та Mac це одна з популярних для вилучення тексту з зображень на iPhone та iPad. Програма розроблена Gentlemen Coders,

які створюють популярний додаток для редагування фотографій RAW Power для iOS/macOS. Після недовгого налаштування можна вибрати мову, текст якої потрібно витягти. У додатку простий та зручний інтерфейс. Доступна функція витягування тексту із готових зображень з фотопотоку або за допомогою камери. Фотографії не зберігаються на пристрої. Після того, як текст буде розпізнаний, можна скопіювати його, поділитись ним, виконати пошук або перекласти текст. Можна навіть редагувати його вручну. При цьому всі дані не зберігаються і нікуди не відправляються.

Переваги LiveScan:

- Зручна програма з простою навігацією, що дозволяє користувачам легко сканувати та зберігати свої документи.
- Висока точність розпізнавання тексту.
- Підтримує безліч форматів, зокрема PDF, Word і Excel, що полегшує обмін документами та спільну роботу з іншими користувачами.
- Доступний безкоштовно на платформах iOS та Android.

Недоліки:

- Обмежена кількість підтримуваних мов.
- Програма не може створювати високоякісні зображення в умовах недостатнього освітлення або розмитості, що може вплинути на точність розпізнавання тексту.

Рисунок 2.2 – Інтерфейс додатку LiveScan

Microsoft Office Lens

Microsoft Office Lens – одна з передових програм для розпізнавання тексту від IT гіганта Microsoft. Дозволяє сканувати друкований текст або рукописні нотатки та перетворенням їх на текст, який можна редагувати пізніше. Оскільки працює на платформі Microsoft, ви також можете редагувати ці відсканові зображення в документах і PowerPoint. Є підтримка популярних сайтів хмарних сховищ, і один із них – OneNote.[18]

Плюси Microsoft Office Lens:

- Microsoft Office Lens - це зручна програма, яка проста у використанні та навігації.
- Забезпечує високу точність розпізнавання тексту, гарантуючи, що текст буде правильно вилучений із зображень.
- Додаток підтримує різні формати, зокрема PDF, Word і PowerPoint, що полегшує обмін даними та спільну роботу з іншими.
- Програма інтегрована зі службою Microsoft OneDrive та іншими хмарними сервісами, що дає змогу користувачам легко зберігати документи та ділитися ними.
- Доступний безкоштовно на платформах iOS та Android.

Переваги:

- Обмежений функціонал, додаток призначений в першу чергу для сканування документів і розпізнавання тексту і не пропонує розширених функцій, таких як інструменти для редагування або оптичне розпізнавання символів рукописного тексту.
- Обмежена кількість підтримуваних мов.
- Програма не може створювати високоякісні зображення в умовах недостатнього освітлення або розмитості, що може вплинути на точність розпізнавання тексту.

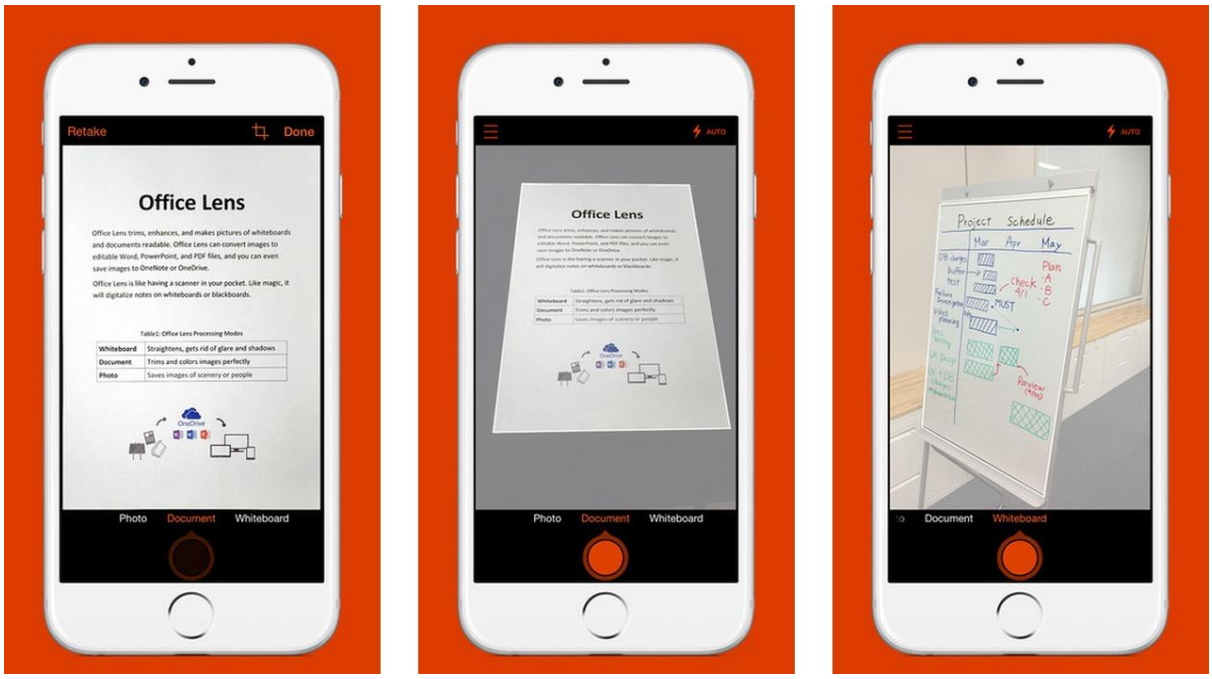

Рисунок 2.3 - Інтерфейс додатку Microsoft Office Lens

Adobe Scan

Додаток сканера Adobe Scan перетворює пристрій у потужний портативний сканер, який автоматично розпізнає текст (OCR) і дозволяє зберігати у кількох форматах файлів, включаючи PDF та JPEG. Дозволяє сканувати кватанції, нотатки, документи, фотографії, візитні картки, дошки з текстом, який можна використовувати повторно з кожного PDF -файлу та сканування фотографій. За допомогою розширеної технології зображення автоматично визначає кордони, загострює відсканований вміст і розпізнає текст. Його сильною стороною є можливість експорту документів до Adobe Acrobat, де можна зручно редагувати PDF-файли та зберігання в хмарному середовищі Adobe [18].

Рисунок 2.4 – Інтерфейс додатку Adobe Scan

Переваги:

- Зручна програма з простою навігацією, що дозволяє користувачам легко сканувати та зберігати свої документи.
- Висока точність розпізнавання тексту.
- Підтримує безліч форматів, зокрема PDF, Word і Excel, що полегшує обмін документами та спільну роботу з іншими користувачами.
- Програма інтегрована з Adobe Document Cloud та іншими хмарними сервісами, що дозволяє користувачам легко зберігати та ділитися своїми документами.
- Доступний безкоштовно на платформах iOS та Android.

Недоліки:

- Обмежена кількість підтримуваних мов.
- Програма не може створювати високоякісні зображення в умовах недостатнього освітлення або розмитості, що може вплинути на точність розпізнавання тексту.

2.4 Висновки до розділу

Розглянуто популярні мови програмування, які дозволяються створювати мобільні додатки для iOS платформи, вказано на переваги та недоліки кожної з них. З наведених варіантів обрано – Swift, що дозволяє створювати ефективні та безпечні мобільні додатки. Swift є офіційною мовою програмування для створення додатків для платформ iOS та розроблена компанією Apple .

Проаналізовано методи реалізації розпізнавання символів як на комп’ютері, так і на мобільному телефоні, наведено переваги на недоліки кожної із них та обрано фреймворк для реалізації функціоналу додатку - VisionKit. Даний фреймворк є розробкою компанії Apple та вбудований в iOS , це означає, що не потрібно встановлювати додаткові бібліотеки або сторонні інструменти, щоб використовувати його. Також VisionKit використовує передові техніки комп'ютерного зору, такі як нейронні мережі, для розпізнавання тексту на зображеннях з високою якістю.

Визначено характеристики існуючих систем розпізнавання тексту для комп’ютерів та для мобільних телефоні на iOS платформі. Наведено переваги на недоліки кожної із існуючих систем.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Вибір технологій для реалізації додатку

Для створення інтерфейсу додатку використано технологію від компанії Apple – SwiftUI. SwiftUI використовує декларативний метод що, дозволяє зручно та швидко створювати інтерфейс користувача. Від застарілого методу написання інтерфейсу користувача з використанням фреймворку UIKit вирішено відмовитись тому що:

- Використовується імперативний синтаксис, що ускладнює читання і написання коду.
- Вимагає більше шаблонного коду, ніж SwiftUI, що уповільнює - розробку.
- Не підтримує попередній перегляд у реальному часі, а це означає, що при кожній перевірці коректності інтерфейсу, доводиться запускати додаток [19].

Головні критерії для вибору фреймворку: актуальність, швидкість, якість розпізнавання тексту та легка інтеграція в систему iOS. Фаворитом серед ознайомлених фреймворків є Vision з його доповненням VisionKit [17].

Vision – це фреймворк від компанії Apple, який дозволяє використовувати високо рівневі API для запуску алгоритмів комп’ютерного зору для зображень та відео. Фреймворк Vision виконує розпізнавання обличь, виявлення тексту, розпізнавання штрих-кодів, реєстрацію зображень. Vision також дозволяє використовувати моделі CoreML для таких завдань, як класифікація або виявлення об'єктів.

VisionKit – бібліотека від компанії Apple, що надає функції для розпізнавання тексту на зображеннях і у відео в реальному часі з камери iOS. В реальному часу аналізує зображення для виявлення тексту, даних у тексті та машинних кодів. В iOS 13 фреймворк Apple Vision також додає підтримку OCR

(оптичне розпізнавання символів), яке дозволяє виявляти та розпізнавати текст у відсканованих документах.

Для повторного перегляду попередньо розпізнаного тексту, обрано зберігати отримані дані до хмарного сховища. Обрано використовувати хмарну базу даних - Firebase. Firebase – це хмарне середовище з широким спектром можливостей покращення та спрощення створення мобільних додатків. Текстові дані збергіються в JSON форматі, які можна зчитувати, записувати та вилучати.

В пакеті функціоналу Firebase є доступні: автентифікація, аналітика додатку, надсилання повідомлень, хостинг, та база даних в режимі реаольного часу.

3.2 Реалізація розпізнавання тексту у додатку

Функціональність розпізнавання тексту у додатку розроблено на основі фреймворку Vision від компанії Apple. Vision надає високорівневі API для обробки зображень і відео в режимі реального часу. Фреймворк використовує алгоритми машинного навчання та методи комп'ютерного зору для ідентифікації об'єктів і характеристик на зображеннях і відеопотоках. Фреймворк містить попередньо навчені моделі для таких завдань, як виявлення об'єктів, сегментація зображень і розпізнавання облич, а також інструменти для навчання користувацьких моделей. Фреймворк VisionKit є підмножиною фреймворку Vision і спеціально розроблений для розпізнавання тексту. Він надає прості у використанні API для розпізнавання тексту на зображеннях та інтеграції цього тексту в додаток. VisionKit побудований на основі фреймворку Vision і використовує ту ж саму базову технологію для аналізу зображень і розпізнавання тексту.

У ході аналізу існуючих реалізації мобільних додатків для розпізнавання тексту та не спеціалізованих додатках вирішено розробити три типи функціоналу розпізнавання:

- Розпізнавання великого тексту, такого як: сторінки книг, документи, статті.
- Розпізнавання типізованого тексту в режимі реального часу: номеру телефону, адреси, імейлу, URL – адреси та будь якого тексту в одному рядку.
- Сканування банківських карток, яке працює і з горизонтальним форматом карток, і з вертикальним форматом.

Додатковим фунціоналом для зручності використання додатку та роботи із розпізнаваним текстом, додано деякі особливості. Редагування розпізнаваного тексту на окремому екрані. Це зручна особливість у випадках коли текст розпізнаний з помилкою, через природні завади, або через помилку самого ядра OCR у фреймворку VisionKit.

Розпізнавання великого тексту

Розпізнавання великих об’ємів тексту у фреймворку VisionKit виконується за допомогою класу VNDocumentCameraScan. Особливістю цього рішення є можливість розпізнавання тексту із декількох фотографій. Зрозуміло, що через обмеженні ресурси смартфонів розпізнавання тексту із великої кількості фотографій займе тривалий час та багато ресурсів. Фотографії для розпізнавання можливо вибрати із галереї смартфона і зробити їх за допомогою задньої камери на телефоні. Також особливістю VNDocumentCameraScan є можливість розпізнавання тексту не із усієї фотографії, а вибір площини на фотографії із якої потрібно виконати розпізнавання.

```
final class DocumentViewModel{
    let cameraForDocumentScan: VNDocumentCameraScan
    init(cameraForDocumentScan: VNDocumentCameraScan) {
        self.cameraForDocumentScan = cameraForDocumentScan
    }

    func recognizeeText(withCompetionHandler comletionHabdler: @escaping([String])-> Void) {
        DispatchQueue.main.async {
            let images = (0..<self.cameraForDocumentScan.pageCount).compactMap({
                self.cameraForDocumentScan.imageOfPage(at:$0).cgImage

            })

            let imagesAndRequest = images.map({(image: $0, request:VNRecognizeTextRequest())})
            let resultText = imagesAndRequest.map{image,request->String in
                let handler = VNImageRequestHandler(cgImage: image, options: [:])
                do{
                    try handler.perform([request] as! [VNRequest])
                    guard let observations = request.results as? [VNRecognizedTextObservation] else {return ""}
                    return observations.compactMap({$0.topCandidates(1).first?.string}).joined(separator: "\n")
                }catch{
                    print(error)
                    return ""
                }
            }
            DispatchQueue.main.async {
                comletionHabdler(resultText)
            }

        }
    }
}
```

Рисунок 3.1 – Підключення VNDocumentCameraScan та обробка результату розпізнавання

Розпізнавання типізованого тексту

Не кожного разу користувачу потрібно розпізнавати великі об’єми тексту, в більшості випадків у сьогоденні потрібно швидко розпізнати невеликий шматок тексту в режимі реального часу та використати його у своєму смартфоні. Для даних потреб у фреймворку VisionKit можна використати DataScannerViewController. З використанням даного класу створено функціонал для розпізнавання тексту певного типу у режимі реального часу з можливість швидкого копіювання та використання у інших додатках користувача.

Підтримується 5 типів тексту:

- Звичайний текст
- URL (Uniform Resource Locator) адреси
- Номери мобільних телефонів
- Email адреси
- Домашні адреси

```
@Binding var recognizedItems: [RecognizedItem]
let recognizedDataType: DataScannerViewController.RecognizedDataType
let recognizesMultipleItems: Bool

func makeUIViewController(context: Context) -> DataScannerViewController {
    let vc = DataScannerViewController(recognizedDataTypes: [recognizedDataType],
                                       qualityLevel: .accurate, recognizesMultipleItems: recognizesMultipleItems,
                                       isHighFrameRateTrackingEnabled: false,
                                       isPinchToZoomEnabled: true,
                                       isGuidanceEnabled: true,
                                       isHighlightingEnabled: true)
    return vc
}
class Coordinator: NSObject, DataScannerViewControllerDelegate {

    @Binding var recognizedItems: [RecognizedItem]
    func dataScanner(_ dataScanner: DataScannerViewController, didAdd addedItems: [RecognizedItem], allItems: [RecognizedItem]) {
        UINotificationFeedbackGenerator().notificationOccurred(.success)
        recognizedItems.append(contentsOf: addedItems)
        print("didAddItems \(addedItems)")
    }

    func dataScanner(_ dataScanner: DataScannerViewController, didRemove removedItems: [RecognizedItem], allItems: [RecognizedItem]) {
        self.recognizedItems = recognizedItems.filter { item in
            !removedItems.contains(where: {$0.id == item.id })
        }
        print("didRemovedItems \(removedItems)")
    }

    init(recognizedItems: Binding<[RecognizedItem]>) {
        self._recognizedItems = recognizedItems
    }
```

Рисунок 3.2 – Під’єднання DataScannerViewController та обробка отриманого результату

Сканування банківських карток вертикального та горизонтального формату

Головною особливістю даного додатку є саме реалізація сканування банківських карток горизонтального та вертикального формату. Сканування відбувається з допомогою вище згадуваного класу VNDocumentCameraScan з подальшою обробкою отриманих даних.

Підтримуються картки більшості відомих банківських систем:

- Master Card
- Visa
- Amex
- Discover
- Diners Club

```
    func parseResults(for recognizedText: [String]) -> String {
        // Credit Card Number
        let creditCardNumber = recognizedText.first(where: { $0.count >= 14 && ["4", "5", "3", "6"].contains($0.first) })

        // Expiry Date
        let expiryDateString = recognizedText.first(where: { $0.count > 4 && $0.contains("/") })
        let expiryDate = expiryDateString?.filter({ $0.isNumber || $0 == "/" })

        // Name
        let ignoreList = ["GOOD THRU", "GOOD", "THRU", "Gold", "GOLD", "Standard", "STANDARD", "Platinum",
                          "PLATINUM", "WORLD ELITE", "WORLD", "ELITE", "World Elite", "World", "Elite"]
        let wordsToAvoid = [creditCardNumber, expiryDateString] +
            ignoreList +
            CardType.allCases.map { $0.rawValue } +
            CardType.allCases.map { $0.rawValue.lowercased() } +
            CardType.allCases.map { $0.rawValue.uppercased() }
        let name = recognizedText.filter({ !wordsToAvoid.contains($0) }).last
        let results =
        """
        Card Number: \(creditCardNumber ?? "")
        Name: \(name ?? "" )
        ExpiryDate: \(expiryDate ?? "")
        """
        return results

    }
}
```

Рисунок 3.3 – Обробка отриманих даних після розпізнавання

```
public enum CardType: String, CaseIterable, Identifiable {
    case masterCard = "MasterCard"
    case visa = "Visa"
    case amex = "Amex"
    case discover = "Discover"
    case dinersClubOrCarteBlanche = "Diner's Club/Carte Blanche"
    case unknown

    public init(number: String?) {
        guard let count = number?.count, count >= 14 else {
            self = .unknown
            return
        }
        switch number?.first {
        case "3":
            if count == 15 {
                self = .amex
            } else if count == 14 {
                self = .dinersClubOrCarteBlanche
            } else {
                self = .unknown
            }
        case "4": self = (count == 13 || count == 16) ? .visa : .unknown
        case "5": self = count == 16 ? .masterCard : .unknown
        case "6": self = count == 16 ? .discover : .unknown
        default: self = .unknown
        }
    }
```

Рисунок 3.4 – Модель типу картки

Збереження тексту у хмарній базі даних

Збереження розпізнаваного тексту до хмарної бази данних Firebase, а саме Firebase Realtime Database. Дане середовище дозволяє завантажувати та миттєво зчитувати данні при будь якій їх зміні та видаляти їх за потреби. Головною перевагою даного рішення є можливість збереження даних без доступу до мережі. Усі дані які були змінені будуть надіслані до серверу одразу після відновлення підключення смартфону до мережі.

```
func loadToDataBase(item: ScanData){
    let textItemRef = self.ref.child(item.id)
    textItemRef.setValue(item.content)
}
func fetchData(){
    ref.observe(.value) { snapshot in
        guard let child = snapshot.children.allObjects as? [DataSnapshot] else{
            return
        }
        self.texts = child.compactMap({ snapshot in
            return ScanData(id: snapshot.key, content: snapshot.value as! String)

        })
        print(self.texts)
    }
}
func remove(id: String){
    print(id)
    ref.child(id).setValue(nil)
}
```

Рисунок 3.5 – Завантаження результату до бази даних

3.3 Реалізація застосунку

Мобільний застосунок реалізовано під iOS платформу у середовищі XCODE. Для створення структури додатку вирішено використовувати архітектуру MVVM – що є однією з популярних архітектурних підходів в розробці додатків для iOS. Вона дозволяє розділити логіку програми на різні компоненти, такі як модель(Model), представлення (відоме також як View) та модель представлення (відома також як ViewModel). Даний підхід розділення логіки програми чудово інтегрується з використним фреймворком написання UI частини додатку SwitUI. SwiftUI має вбудовану підтримку прив'язки даних, що дозволяє автоматично оновлювати відображення на UI при зміні даних в ViewModel. Це дозволяє забезпечити автоматичне оновлення UI без прямої

маніпуляції над UI елементами. MVVM використовує цю функцію SwiftUI для забезпечення розділення відображення даних та бізнес-логіки.

Створений функціонал застосунку налічує 5 екранів:

- Головний екран
- Екран розпізнавання великого тексту
- Екран розпізнавання типізованого тексту
- Екран розпізнавання банківських карток
- Екран перегляду та редагування розпізнаваного тексту

Головний екран

Рисунок 3.6 – Головний екран додатку

На етапі коли користувач вперше увійшов у додаток на головному екрані висвічується надпис “Add your first recognized text”, що означає що користувач

ще не розпізнавав текст, або видалив усі попередньо розпізнанні елементи із хмарної бази данних. У нижній частині екрану знаходяться 3 кнопки:

- Scan Document – реалізує перехід від головного екрану до екрану розпізнавання великого тексту
- Scan typed text – реалізує перехід від головного екрану до екрану розпізнавання типізованого тексту
- Scan Card – реалізує перехід від головного екрану до екрану розпізнавання карток.

Після розпізнавання будь якого із видів тексту та занесення його до хмарної бази даних текст у центральній частині додатку зміниться елементом List, що реалізує таблицю у фреймворкові SwiftUI.

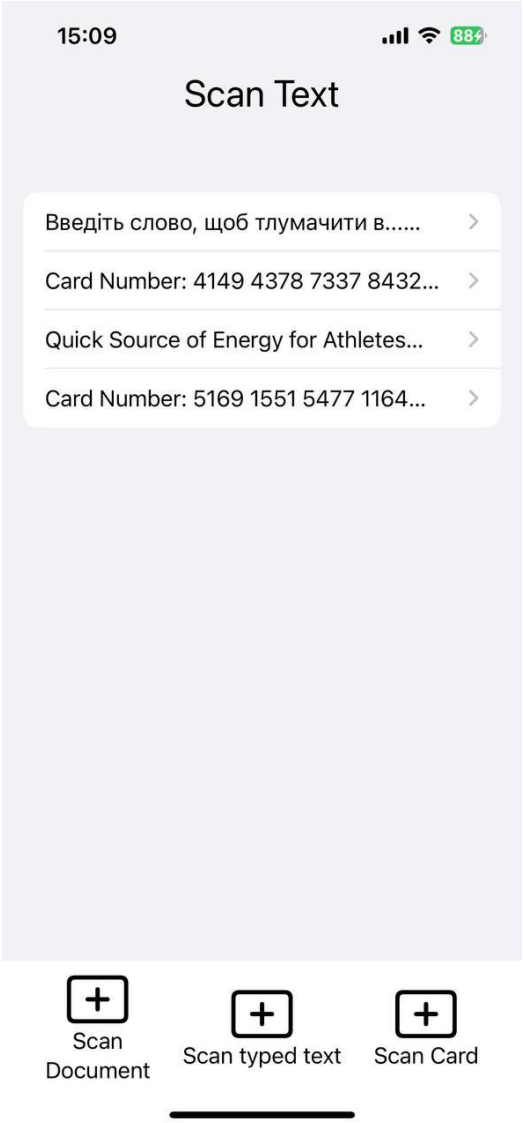

Рисунок 3.7 – Головний екран із таблицею

У даній таблиці знаходяться усі дані попередньо розпізнаваного тексту, який зберігається у хмарній базі даних. Дані оновлюються в режимі реального часу, тобто як тільки інформація буде завантажена, або видалена програма одразу про це дізнається та оновлює таблицю. По натиску по кожному із елементів таблиці реалізовано перехід до екрану перегляду та редагування.

Екран редагування

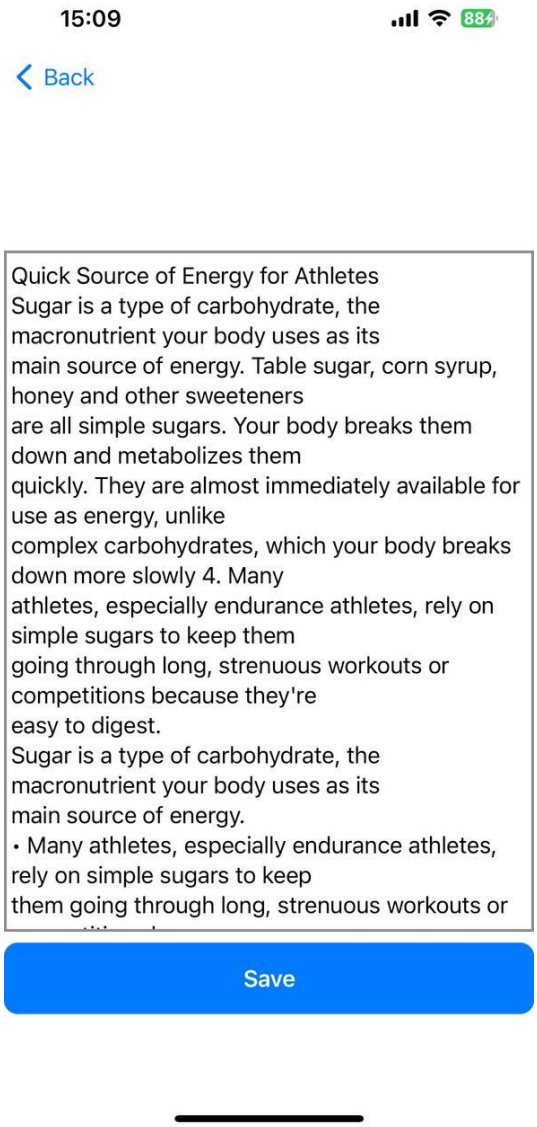

Рисунок 3.8 – Екран редагування тексту

На даному екрані відображається попередньо розпізнаваний текст. Даний текст можна насамперед перечитати, відредагувати під свої потреби та скопіювати для подальшого використання в потребах користувача.

Відредаговані зміни можна зберегти до хмарної бази даних натиснувши кнопку “Save”.

Екран розпізнавання великого тексту

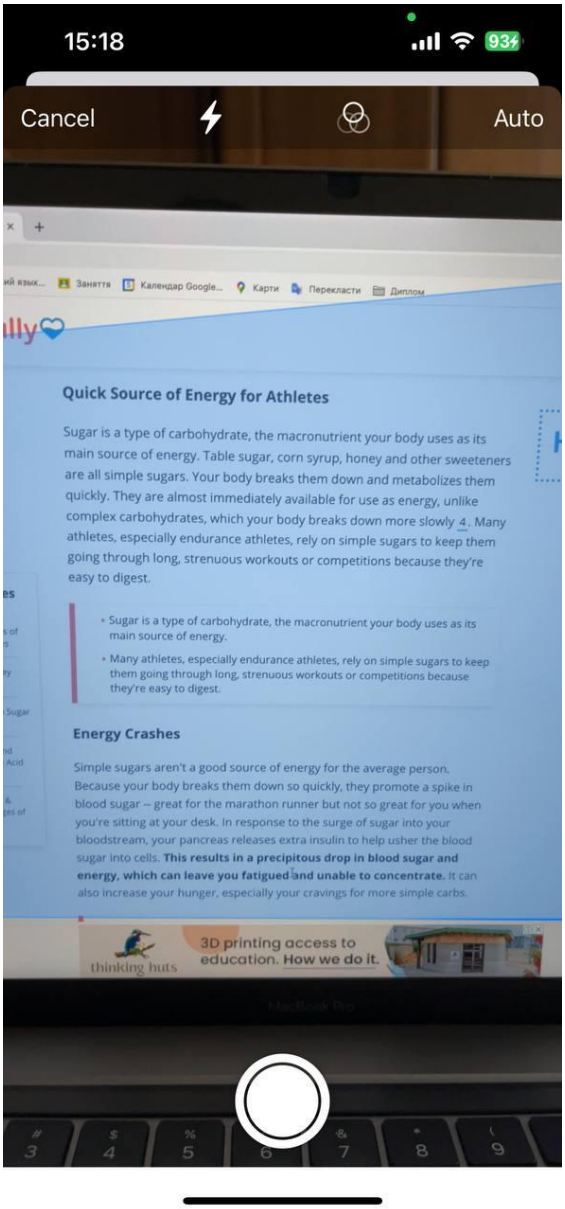

Рисунок 3.9 – Екран розпізнавання великого тексту

На даному екрані користувач має змогу навести камеру смартфону на потрібний текст для розпізнавання. Потрібний об’єм тексту може автоматично в режимі реального часу обрати вбудований функціонал класу VNDocumentCameraScan, так як наведено на рисунку 3.8, або ж користувач може зробити фотографію тексту та самостійно обрати необхідний об’єм тексту (рисунок 3.9).

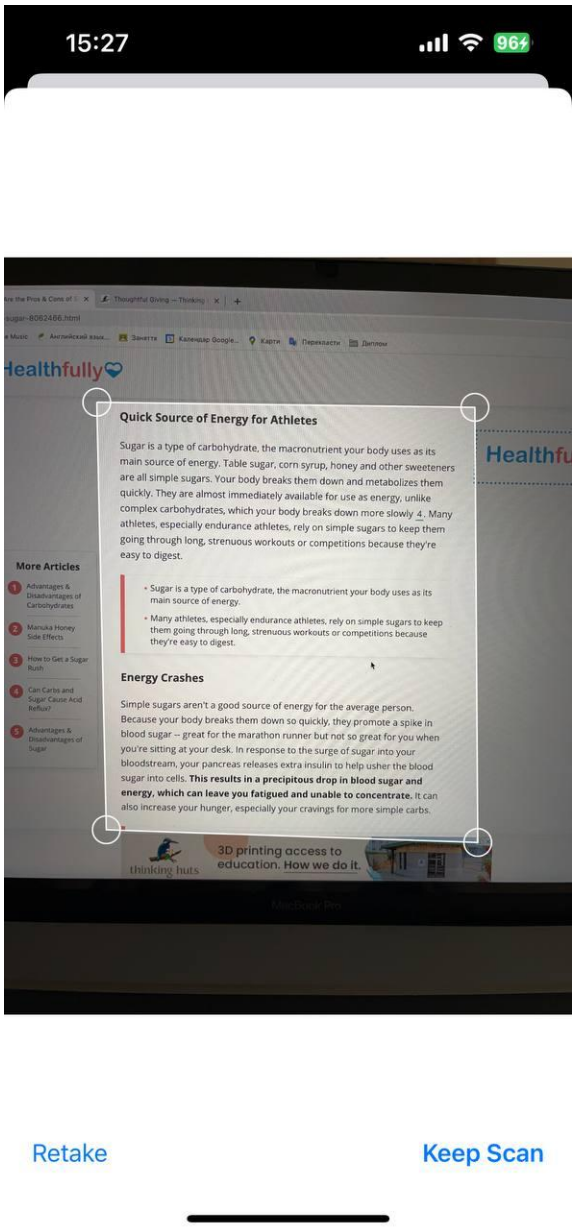

Рисунок 3.10 – Вибір об’єму тексту для сканування

У разі поганої якості зображення користувач може перезняти фотографію натиснувши кнопку “Retake”. У разі успішної фотографії та вибору об’єму тексту користувач натискає кнопку “Keep Scan” для продовження фотографування необхідного тексту, або переходу до перегляду розпізнаваного тексту.

Екран розпізнавання типізованого тексту

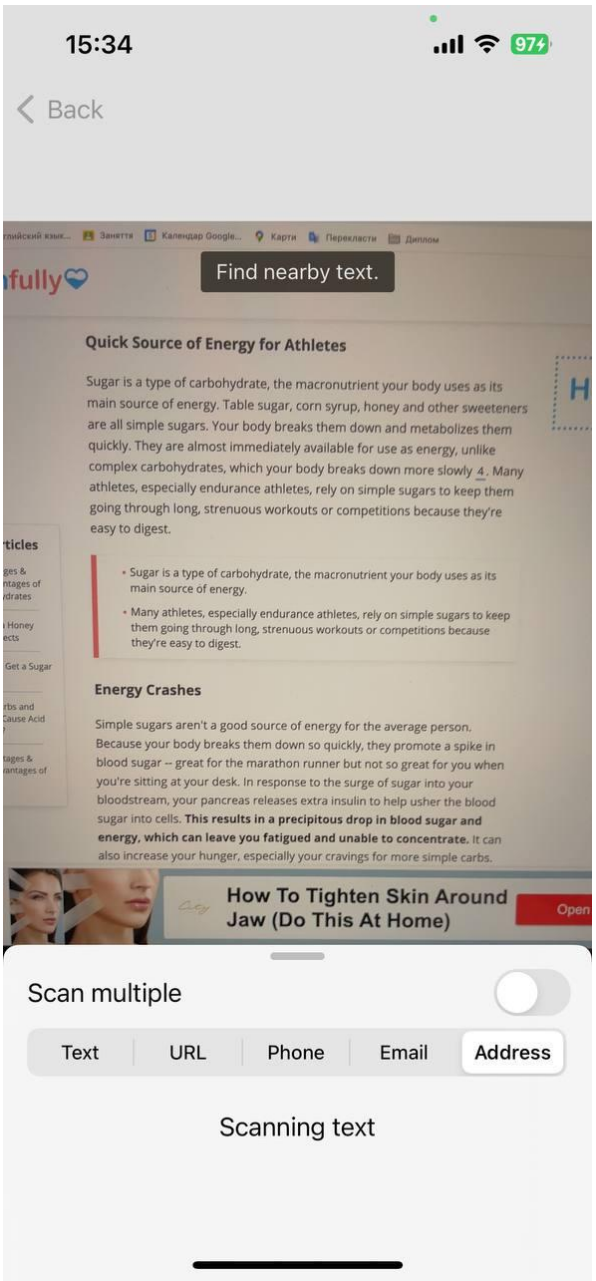

Рисунок 3.11 – Екран типізованого розпізнавання тексту

На даному екрані користувач може швидко розпізнавати в режимі реального життя деякий типізований текст. У нижній частині екрані знаходиться екран налаштувань та вибору типу тексту. Користувач може обрати сканування тільки одного шматка тексту вибраного типу, чи одразу декілька перемкнувши елемент “Toggle” в увімкнуте положення.

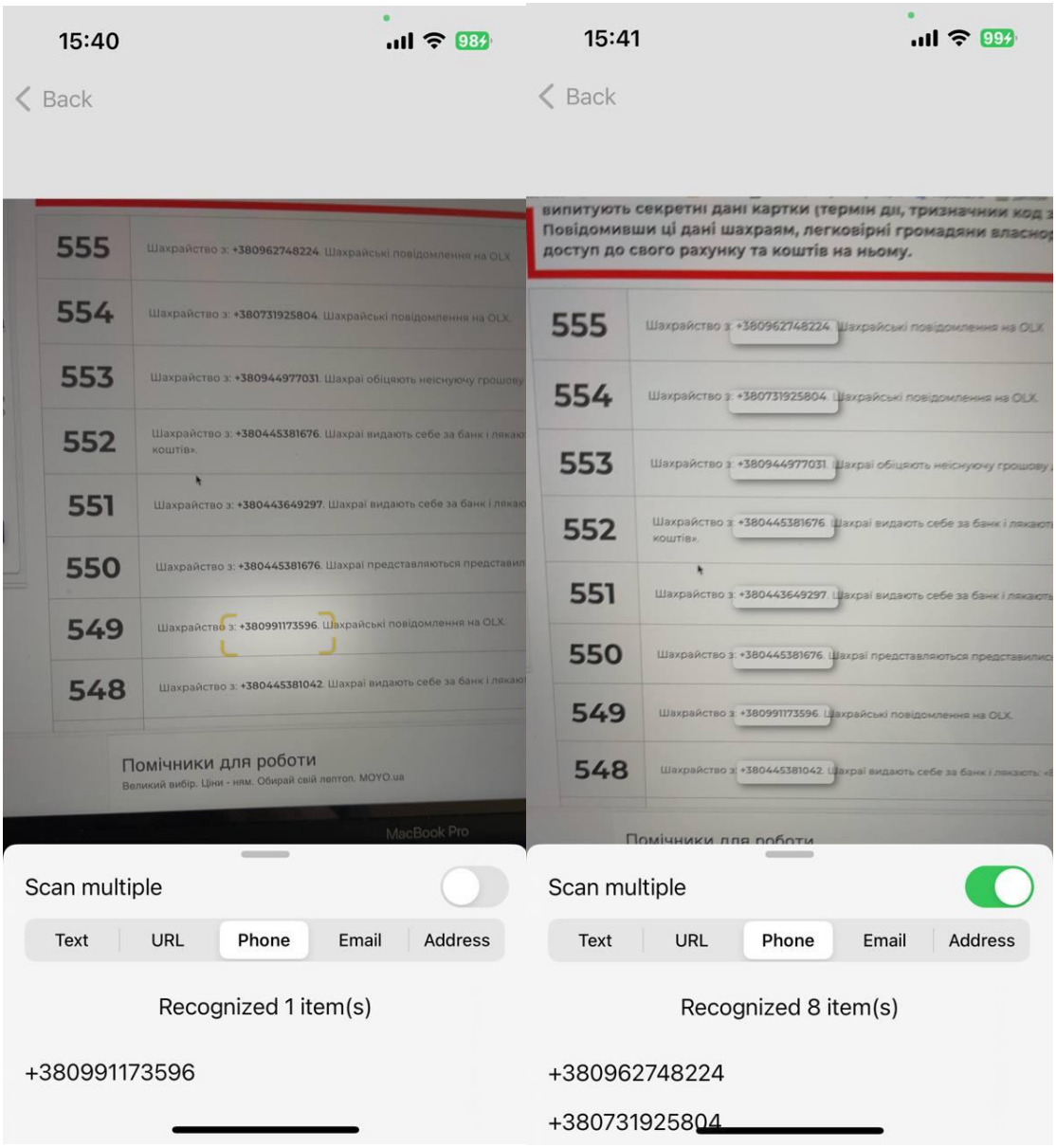

Рисунок 3.12 – Приклад розпізнавання з режимом “Scan multiple” та без

Нижче користувач може обрати тип тексту який потрібно сканувати. На даний момент на цей екран включені тільки ті типи, які підтримуються фреймворком VisionKit, а саме:

- Звичайний текст
- URL-адреси
- Номери мобільних телефонів
- Email-пошт
- Домашні адреси

У самому низу екрану знаходиться місце, куди виводяться усі розпізнавані елементи на екрані. Даний текст користувач може скопіювати та використати у подальших потребах.

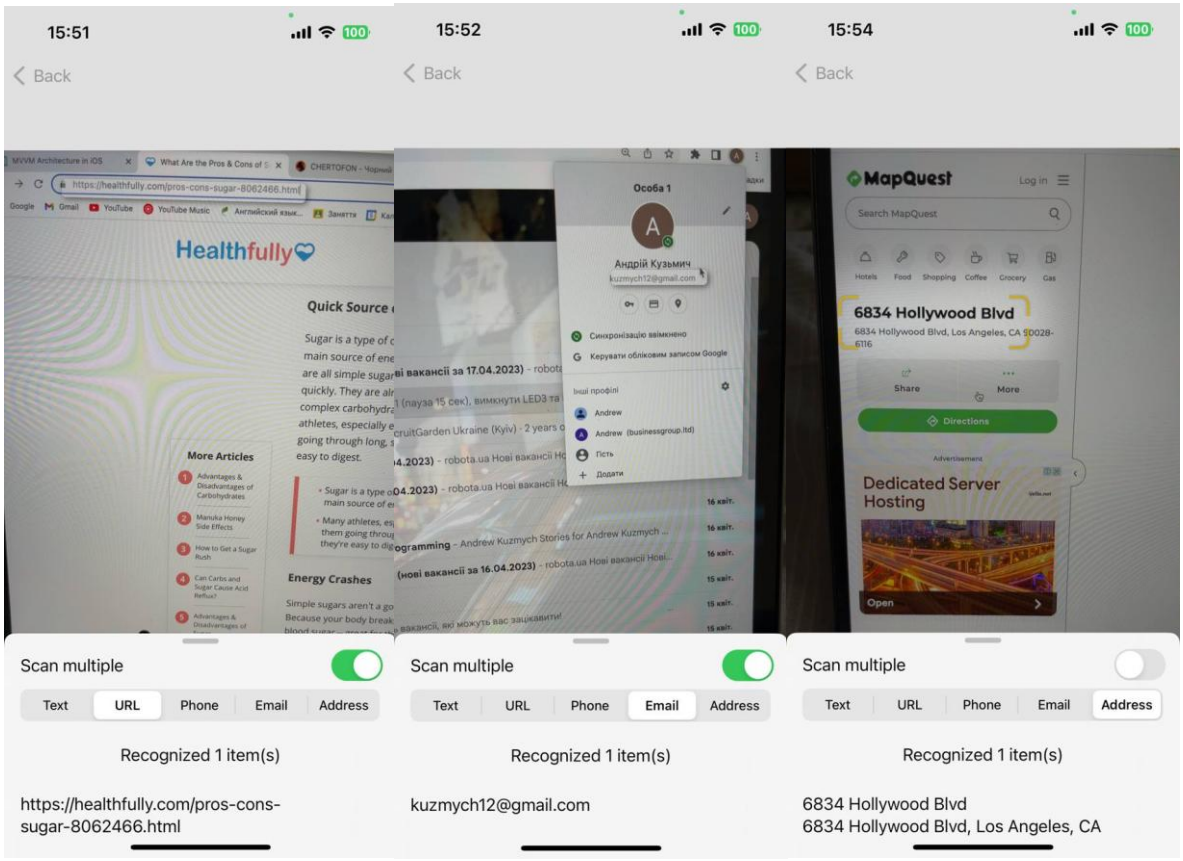

Рисунок 3.13 – Результати розпізнавання типізованого тексту

Розпізнавання банківських карток

Фрейворк VisionKit не підтримує тип банківських карток, хоча це розповсюдженна задача розпізнавання тексту у мобільних застосунках. Також, проаналізувавши деякі з додатків, де присутнє розпізнавання банківських карток, виявлено, що реалізовано тільки розпізнавання стандартного горизонтального формату карток. Натомість, як менш розповсюджений вертикальний формат не підтримується ж взагалі, або розпізнавання працює не належним чином. Тому вирішено реалізувати дану фукнцію у додатку. Процес розпізнавання виконується попереднім скинування тексту із фотографію за допомогою класу VNDocumentCameraScan та подальшою обробкою інформації та відкиданням непотрібної інформації.

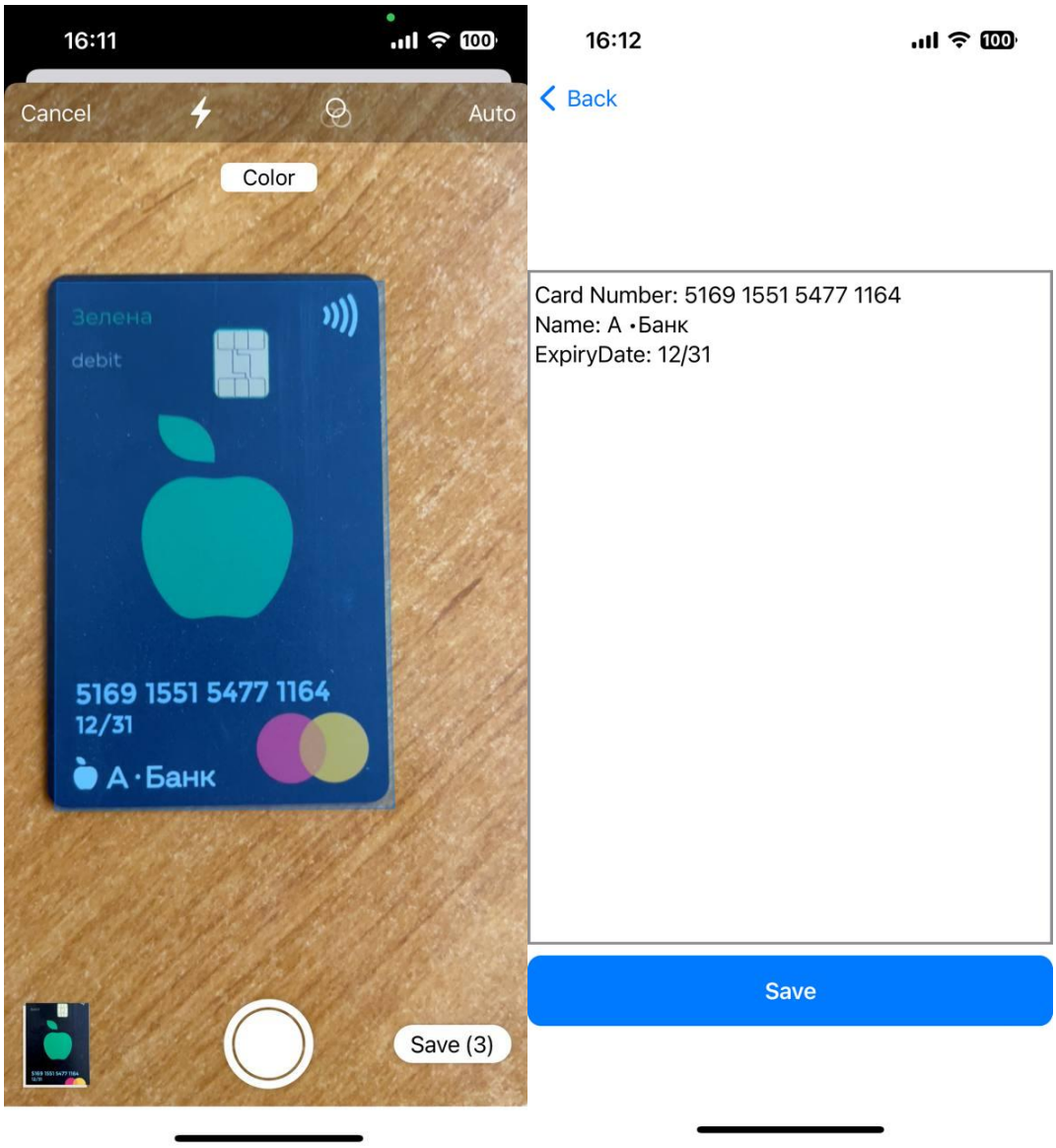

Рисунок 3.14 – Результат розпізнавання вертикального формату карток

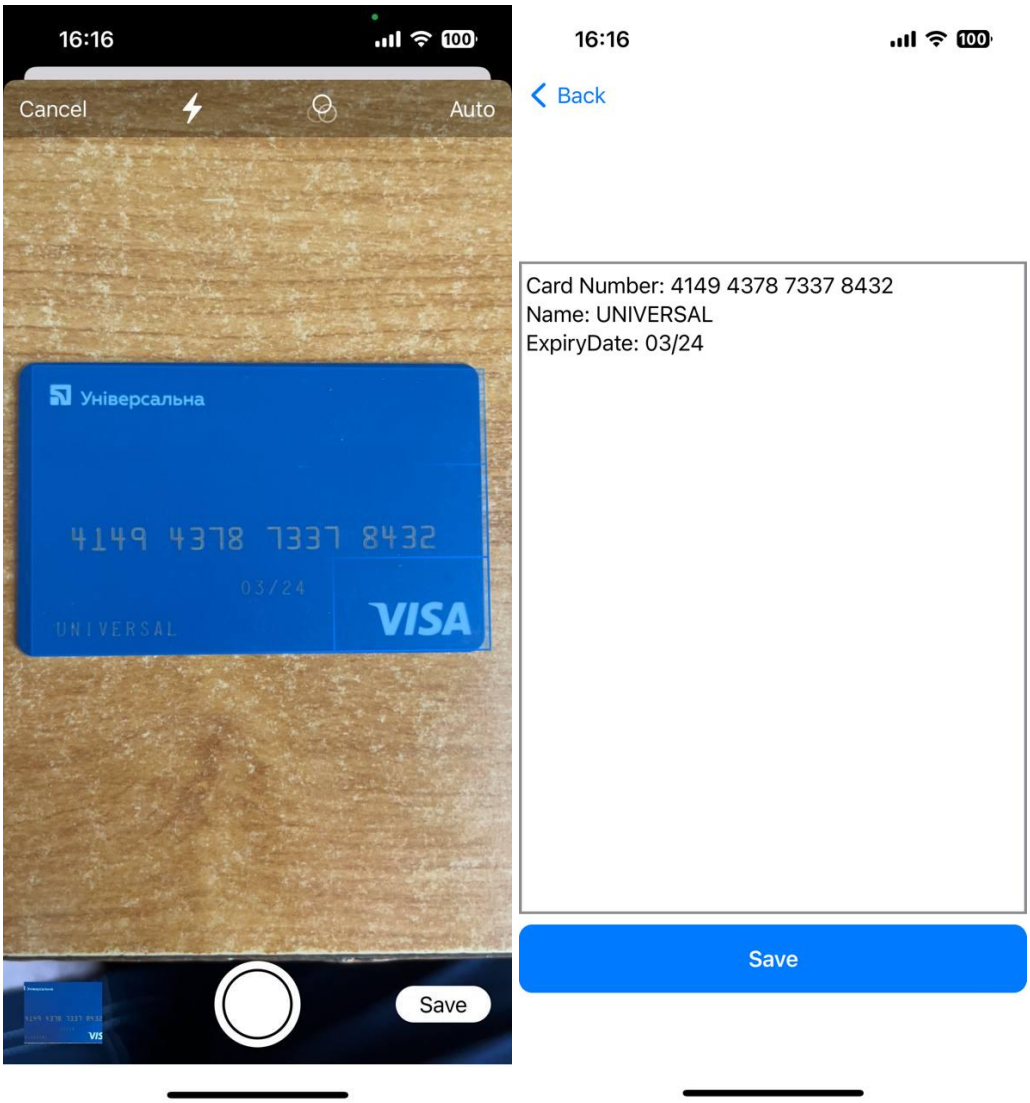

Рисунок 3.15 – Результат розпізнавання горизонтального формату карток

3.4 Висновки до розділу

В даному розділі вибрано головні елементи для створення iOS додатку: мова програмування, середовище розробки, фреймворк створення UI частини та база даних для збереження результату.

Обрано фреймворк за допомогою якого реалізовано функціонал розпізнавання тексту. Наведено основні пункти функціоналу додатку та їх реалізацію:

- Розпізнавання великого тексту
- Розпізнавання типізованого тексту
- Розпізнавання банківських карток

Наведено вигляд кожного з сторінок додатку: головний екран, екрани розпізнавання та екран редагування тексту. Показано функціонал кожного із типів функціоналу та наведено результати їх виконання.

ВИСНОВКИ

В ході виконання дипломної роботи розглянуто основні теоретичні відомості про технологію ORC, методи розпізнавання тексту та основі нейронні мережі на архітектури для вирішення задач розпізнавання символів.

Для реалізацію цільового мобільного додатку, проаналізовано можливі засоби за допомогою яких можливо створити даний застосунок. Розглянуто та наведено переваги та недоліки існуючих систем розпізнавання тексту. Вирішено розробити застосунок на платформу iOS використавши наступні технології:

- Мова програмування Swift
- Фреймворк SwitUI для створення інтерфейсу користувача додатку
- Хмарна базу даних Firebase Realtime Database для збереження розпізнаного тексту
- Фреймворк VisionKit для створення функціоналу розпізнавання тексту у додатку

В результаті проведеної роботи створено додаток для розпізнавання тексту з 5 екранами з наступним функціоналом:

- Головний екран із переходами до екранів сканування та відображення попередньо сканованого тексту
- Екран сканування великого тексту
- Екран сканування типізованого тексту
- Екран сканування банківських карток
- Екран редагування та перегляду розпізнаного тексту

Отримані результати зберігаються у хмарній базі даних Firebase Realtime Database.

Головними перевагами розробленого додатку перед аналогами є розширений функціонал. На екрані сканування типізованого тексту користувач

може обрати потрібний тип тексту, який потрібно розпізнати, що пришвидшує процес отримання необхідного тексту користувачу. На екрані сканування банківських карток розроблено функціонал, який дозволяє сканувати дані із банківських карток як горизонтального формату, так і вертикального, в той час як аналоги дозволяють виконувати сканування лише карток горизонтального формату.

Розробку мобільного додатку виконано в повному обсязі. Застосунок відповідає поставленим вимогам. Вихідний код застосунку розташований в відкритому доступі, що дозволяє використовувати його іншим розробникам.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Оптичне розпізнавання символів [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Оптичне_розпізнавання_символів
2. What is OCR (Optical Character Recognition)?[Електронний ресурс].Режим доступу: https://aws.amazon.com/what-is/ocr/
3. OCR Algorithm: Improve and Automate Business Processes [Електронний ресурс]. Режим доступу: https://indatalabs.com/blog/ocr-automate-business-processes
4. Optical character recognition [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Optical_character_recognition
5. Perret, B., Chierchia, G., Cousty, J., Guimarães, S. J. F., Kenmochi, Y., & Najman, L. (2019). Higra: Hierarchical graph analysis. SoftwareX, 10, 100335.
6. Деркач, О. І. (2016). Аналітична обробка текстової інформації за допомогою засобів кластеризації. Young, 34(7)
7. Deep learning – OCR Text Detection and Text Recognition [Електронний ресурс]. Режим доступу: https://serengetitech.com/tech/deep-learning-ocr-text-detection-and-text-recognition/
8. PyTorch: Scene Text Detection and Recognition by CRAFT and a Four-Stage Network [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/pytorch-scene-text-detection-and-recognition-by-craft-and-a-four-stage-network-ec814d39db05
9. A Short Intuitive Explanation of Convolutional Recurrent Neural Networks [Електронний ресурс]. Режим доступу: https://www.analyticsvidhya.com/blog/2020/11/a-short-intuitive-explanation-of-convolutional-recurrent-neural-networks/
10. BERT language model [Електронний ресурс]. Режим доступу: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model

11. The 6 Best iOS Development Languages [Електронний ресурс]. Режим доступу: https://distantjob.com/blog/best-language-for-ios-app-development/
12. What is Dart Programming [Електронний ресурс]. Режим доступу: https://www.javatpoint.com/flutter-dart-programming
13. Swift. A powerful open language that lets everyone build amazing apps. [Електронний ресурс]. Режим доступу: https://www.apple.com/swift/
14. What is Xamarine [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin
15. OCR Libraries and Projects [Електронний ресурс]. Режим доступу: https://medevel.com/os-ocr-libraris-and-frameworks/
16. Comparing On-device OCR Frameworks Apple Vision and Google MLKit [Електронний ресурс]. Режим доступу: https://www.bitfactory.io/de/dev-blog/comparing-on-device-ocr-frameworks-apple-vision-and-google-mlkit/
17. VisionKit [Електронний ресурс]. Режим доступу: https://developer.apple.com/documentation/visionkit
18. UIKit vs. SwiftUI: How to Choose the Right Framework for Your App [Електронний ресурс]. Режим доступу: https://getstream.io/blog/uikit-vs-swiftui/
19. 9 BEST OCR APPS IN 2022 [Електронний ресурс]. Режим доступу: https://charli.ai/ocr-apps/

ДОДАТКИ

Додаток 1

Посилання на репозиторій

https://github.com/kuzmychAndrew/TextRecognition