МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о завідувача кафедри
_____ Сафонова С.О.
«____»_____ 2020 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ РОЗПІЗНАВАННЯ ОБРАЗІВ НА МОБІЛЬНИХ ПЛАТФОРМАХ

Освітньо-кваліфікаційний рівень «Магістр»
Спеціальність 122 – «Комп'ютерні науки»

Науковий керівник роботи:	_____ (підпис)	Л. О. Шумова (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я. О. Критська (ініціали, прізвище)
Студент:	_____ (підпис)	І.В. Рудий (ініціали, прізвище)
Група:		КН-18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки
(шифр і назва)
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри
С.О. Сафонова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Рудому Іллі Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та програмна реалізація засобів розпізнавання образів на мобільних платформах

керівник проекту (роботи) Шумова Лариса Олександрівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "11" 10 2019 р. № 135/15.15

2. Строк подання студентом роботи 10.01.2020

3. Вихідні дані до роботи Матеріали науково-дослідної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області та постановка задачі
Аналіз підходу на базі машинного навчання
Реалізація програмного додатка
Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу

Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Критська Я. О. ст. викладач		

7. Дата видачі завдання 02.09.2019

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Аналіз завдання	02.09.19 – 30.09.19	
	Літературний пошук. Визначення вимог до роботи.	01.10.19 – 13.10.19	
	Огляд відомих засобів і методів рішення	14.10.19. – 27.10.19	
	Дослідження і побудова рішення	28.10.19 – 10.11.19	
	Практична реалізація	11.11.19 – 01.11.19	
	Розгляд питань охорони праці та основних напрямків їх дотримання	02.12.18 – 22.12.19	
	Оформлення пояснювальної записки	23.12.19 – 02.01.20	
	Оформлення презентації роботи	03.01.20 – 10.01.20	

Студент _____ Рудий І.В.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Шумова Л.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Рудий І.В. Дослідження та програмна реалізація засобів розпізнавання образів на мобільних платформах.

Розглянуті актуальні завдання розпізнавання образів на мобільних платформах. Проведено аналіз відкритих бібліотек для розпізнавання друкованого тексту та засобів машиного навчання. Розроблено додаток для мобільних пристроїв на базі операційної системи Android, що у реальному часі знаходить текст, який знаходиться у фокусі камери пристрою та озвучує знайдений текст.

Ключові слова: машинне навчання, розпізнавання тексту, Android, Java, нейронні мережі.

АНОТАЦИЯ

Рудой И.В. Исследование и программная реализация средств распознавания образов на мобильных платформах.

Рассмотрены актуальные задачи распознавания образов на мобильных платформах. Проведен анализ открытых библиотек для распознавания печатного текста и средств машинного обучения. Разработан приложение для мобильных устройств на базе операционной системы Android, в реальном времени находит текст, который находится в фокусе камеры устройства и озвучивает найденный текст.

Ключевые слова: машинное обучение, распознавание текста, Android, Java, нейронные сети.

ABSTRACT

Rudiy I.V. Research and software implementation of pattern recognition tools on mobile platforms.

Topical tasks of pattern recognition on mobile platforms are considered. Open libraries were analyzed for the recognition of printed text and machine learning tools. An Android-based mobile application has been developed to detect text in real time that in the focus of the device's camera and voice the found text.

Keywords: machine learning, text recognition, Android, Java, Google ML Kit, neural network.

ЗМІСТ

ВСТУП

Актуальність теми. Обчислювальні системи існують вже не один десяток років. Метою їх створення була можливість замінити людину, зробити за нього роботу з важкими обчисленнями.

Одна з таких задач це – навчити комп’ютер розпізнавати образи, тобто ідентифікувати конкретний об’єкт (текст, зображення), чи відносити об’єкт до конкретного класу.

Розпізнавання текстів - дуже складне завдання з теоретичної та практичної точок зору. Людина, наприклад, задіє для цього весь комплекс знань і досвіду. Він визначає текст з сукупності сигналів органів почуттів, виділяє кожен символ, виділяє характерні ознаки символів і на підставі свого досвіду приходить до висновку про значення символу і всього тексту в цілому. Комп'ютер помиляється в процесі розпізнавання набагато частіше людини.

Сьогодні не існує абсолютно точного методу визначення тексту і символу по їх зображенню. Багато розроблені комерційні проекти використовують свої запатентовані методи і не можуть похвалитися ідеальним рішенням завдання. У багатьох випадках гарне рішення дає комплексний підхід до задачі.

Сама задача розпізнавання тексту поділяється на підзадачі: фільтрація зображення від шуму, виділення зображень символів з зображення тексту, виділення ознак символів і порівняння цих ознак з збереженими зразками. Кожне завдання містить багато варіантів рішень, з яких лише деякі є більш-менш оптимальними.

Все вищенаведене свідчить про те, що проблема дослідження засобів розпізнавання образів є актуальною на даний момент.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне завдання** розробки програмних засобів розпізнавання образів на мобільних платформах.

Об'єкт дослідження – процеси розпізнавання образів.

Предмет дослідження – програмні засоби розпізнавання образів на мобільних платформах з застосуванням відкритих бібліотек.

Мета і завдання дослідження. Метою дослідження є аналіз процесів розпізнавання образів та роззробка програмного засобу розпізнавання образів на мобільних платформах.

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

- провести аналіз існуючих засобів ~~реалізаціію~~ розпізнавання образів;
- розглянути існуючі рішення додатки для розпізнавання тексту;
- розробити програмний додаток для розпізнавання друкованого тексту на базі операційної системи Android.

Методи рішення поставлених задач базуються на комплексному використанні глибокого навчання. У якості методів математичного моделювання у роботі використовувалися методи нейро-мережевого моделювання.

Наукова новизна одержаних результатів.

Удосконалено процеси розпізнавання образів на мобільних платформах. Набули подальшого розвитку програмні засоби для розпізнавання тексту на мобільних платформах з застосуванням відкритих бібліотек.

Апробація результатів роботи. Основні результати магістерської атестаційної роботи докладалися на V науково-практичному молодіжному форумі «ІТ-ідея 2019» та у всеукраїнській науково-практичній конференції «Майбутній науковець – 2019» (м. Сєвєродонецьк)**.**

Практичне значення отриманих результатів. В результаті досліджень розроблено додаток для мобільних пристроїв на базі операційної системи Android, що у реальному часі знаходить текст, який знаходиться у фокусі камери пристрою та озвучує знайдений текст.

Публікації. Основні результати магістерської атестаційної роботи опубліковано в 2 наукових працях: серед яких тези у збірнику науково-практичних праць V молодіжного форуму «ІТ-ідея 2019» [1], та тези доповіді на всеукраїнської науково-практичної конференції «Майбутній науковець – 2019» [2]**.**

Структура і обсяг роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. Загальний обсяг роботи складає сторінок, з яких анотація на 1 сторінці, зміст на 2 сторінках, вступ на 2 сторінках, основний текст на 72 сторінках, висновки на 1 сторінці, список використаних джерел із 35 найменувань на 3 сторінках, додатки на 15 сторінках. Робота містить 5 таблиць та 28 рисунків.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Розпізнавання образів

Теорія розпізнавання образів - це один із розділів кібернетики, який вивчає методи класифікації, основи розпізнавання предметів, явищ, сигналів, які можна охарактеризувати набором властивостей і ознак.

Розпізнавання образів можна розділити на два напрямки:

- моделювання;
- дослідження здібностей живих істот.

Розвиток теорій та методів, що вивчають по будову пристроїв, які призначені для розв'язання окремих задач у прикладних цілях. Таких як розпізнавання тексту, штрих-кодів, автомобільних знаків, обличь, мови, зображень та ділянок землі з родовищами корисних копалин.

Розпізнавання тексту, а саме текстових знаків(літер) має велике значення для інформаційних технологій. Системи, що розпізнають літери використовуються у сканерах, ал такі пристрої лише створюю зображення тексту, для того щоб розпізнати окремі символи потрібно використовувати оптичне розпізнавання тексту. Вже отриманий текст можна використовувати у текстових редакторах, змінювати його, виконувати пошук то що. Розпізнавання літер необхідне для підтримки пристроїв, що розпізнають рукописні символи.

1.2 Оптичне розпізнавання тексту

Оптичне розпізнавання тексту (OCR) – переведення зображення у послідовність кодів, які потім будуть представлені у текстовому редакторі. Таке розпізнавання популярне для конвертації книг та документів в електронну форму, потім такі електронні документи використовуються для автоматизації документообігу у державних установах та у бізнесі.

Документи, які були розпізнано можливо редагувати, зберігати у потрібному форматі, шукати по тексту. Розпізнаний текст не втрачає якості, можна аналізувати інформації.

Ця сфера досліджується науковцями зі штучного інтерфейсу, розпізнавання образів та компю'терного зору.

OCR системи мають бути відкалібровані для роботи за певним шрифтом, програма створена на ранніх версіях, могли розпізнавати тільки один шрифт. На даний момент можна використовувати інтелектуальні системи для розпізнавання тексту, такі системи можуть розпізнавати велику кількість шрифтів та з високим ступенем точності. Інколи системи можуть розпізнавати і зображення, колонки та форматування тексту.

Точність розпізнавання друкованих символів можливе лише при чіткому зображенні, наприклад надрукований документ. Точність такого розпізнавання перевищує 99%, а щоб досягти сто відсоткової точності можна лише передавши розпізнаний текст на редагування людині.

Системи для розпізнавання рукописного тексту, які працюють онлайн та розпізнають у реальному часі стали великими комерційними продуктами. Такі алгоритми задають спеціальні вимоги до тексту, а саме спеціальні форми письма. Такі алгоритми не можуть використовуватися для розпізнавання друкованого тексту. Якщо рукописний текст без артефактів, то можна досягти точності приблизно 80 % - 90%, але навіть з такою точність у відтвореному тексті буде велика кількість помилок. Тому така технологія має обмежену кількість застосувань[3].

На даний момент існують декілька підходів для розпізнавання тексту шаблонний, структурний, ознаковий і підхід з використанням нейронних мереж та глибокого навчання.

1.3 Шаблонний підхід

Шаблонний підхід потребує попередню обробку для точності розпізнавання символів. Попередня обробка це представлення зображення, що буде зчитуватися у вигляді растрового зображення, нормалізація розміру, нахил та товщина штриха символу. Цей підхід є прямим порівнянням зображення символу, що розпізнається із шаблонними символами, які були заздалегідь сформовані. Системи, які використовують цей підхід, зазвичай після розпізнавання декількох символів визначають шрифт і порівнюють символи саме з шаблоном символів цього шрифту. До переваг цього методу можна віднести простоту реалізації, високу швидкість розпізнавання і високу стійкість до дефектів символів, що зчитуються. До недоліків можна віднести те, що алгоритм заздалегідь має знати який шрифт буде зчитуватися, що б підібрати потрібний шаблон. Ці алгоритми не являються універсальними.

1.4 Структурний підхід

Структурний підхід розпізнає символи на основі топології представлених зображень символів, у яких міститься інформація розташування частин символів. Алгоритми такого підходу шукають особливі характеристики символів, такі як точки і дуги. Такі особливі характеристики можуть бути представлені у вигляді графа. Перевагою даного підходу є те, що він забезпечує інваріантність щодо типів і розмірів шрифтів символів. Таким чином, точність розпізнавання символів алгоритмами даного підходу не залежить від шрифтів символів. А до недоліків можна віднести низьку швидкість розпізнавання і погане розпізнавання символів, що мають дефект.

1.5 Ознаковий підхід

Ознаковий підхід працює на основі представлення зображень символів у вигляді вектора ознак. Процес розпізнавання символів полягає у порівнянні векторів ознак зображень символів з набором векторів зображень символів вибірки, що навчається, тієї ж розмірності. Процес формування вектора, який представляє собою зображення символу, називається процесом виділення ознак. До переваг можна віднести простоту реалізації, хорошу узагальнюючу здатність, хорошу стійкість до змін форми, розміру і шрифту, низьку чисельність відмов від розпізнавання і високу швидкість розпізнавання. Недоліками даного підходу є нестійкість до різних дефектів зображень і можливість втрати деякої інформації про символ на етапі виділення ознак. Це відбувається через те, що виділення ознак ведеться незалежно, тому інформація про взаємне розташування елементів символу втрачається.

1.6 Метод з використанням штучних нейронних мереж

Метод з використанням штучних нейронних мереж на даний момент один із найбільш популярних підходів до розпізнавання символів. Від класичних методів відрізняється тим, що нейронні мережі навчаються, не програмуються. Використання нейронних мереж є ефективним та продуктивним методом, але для навчання нейронних мереж потрібно мати велику кількість даних, що будуть використовуватися для навчання. Раніше, щоб використовувати нейронні мережі потрібні були великкі обчислювальні можливості.

Нейронні мережі, які розглядаються в даній роботі, є однією з різновидів алгоритмів машинного навчання, або machine learning. Це один з підрозділів штучного інтелекту. Основною властивістю алгоритмів machine learning є їх здатність навчатися в процесі роботи. Наприклад, алгоритм побудови дерева рішень, не маючи ніякої попередньої інформації про те, що являють собою дані і які в них існують закономірності, а тільки деякий вхідний набір об'єктів і значення деяких ознак для кожного з них разом з міткою класу, в процесі побудови дерева сам виявляє приховані закономірності, тобто навчається, і після навчання здатний передбачати клас вже для нових об'єктів, які він не бачив раніше.

Виділяють два основних типи машинного навчання: вчителем і без вчителя. Навчання з вчителем передбачає, що алгоритму крім самих вихідних даних надається деяка додаткова інформація про них, яку він може в подальшому використовувати для навчання. До числа найбільш популярних задач для навчання з вчителем ставляться завдання класифікації і регресії. Наприклад, завдання класифікації можна характезувати так: маючи певний набір об'єктів, кожен з

яких відноситься до одного з декількох класів, необхідно установити, до якого з цих класів відноситься новий об'єкт.

Навчання без вчителя відрізняється від навчання з вчителем тим, що алгоритму не надається ніякої додаткової інформації крім самого набору вихідних даних. Найбільш популярним прикладом завдання навчання без вчителя є завдання кластеризація. Суть завдання кластеризації полягає в наступному: на вхід алгоритму подається деяка кількість об'єктів, які належать різним класам (але якого класу належить який об'єкт невідомо, може бути невідомо також сама кількість класів), і мета роботи алгоритму - розбити це безліч об'єктів на підмножини «схожих» об'єктів, тобто належать одному класу.

Серед всіх алгоритмів машинного навчання виділяють кілька основних сімейств [4]. Якщо говорити про завдання класифікації, до найбільш популярним таким родинам відносяться, наприклад:

- класифікатори, засновані на правилах (Rule – based Classifiers) - основна ідея таких класифікаторів полягає в пошуку правил визначення об'єктів до онго чи іншого класу в формі " IF - THEN ". Для пошуку таких правил зазвичай використовуються деякі статистичні метрики, також часто зустрічається побудова правил на основі дерева рішень[5].
- логістична регресія – пошуку лінійної площині, максимально точно розділяє простір на два півпростору так, що об'єкти різних класів належать різним півпростором. При цьому рівняння цільової площині, що шукається, предсталвлено, як лінійна комбінація вхідних параметрів. Для навчання подібного класифікатора може бути застосовано метод градієнтного спуску.
- байєсівський класифікатор - як випливає з назви, класифікатор заснований на теоремі Байеса, яка записується в формі:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_n)} \quad (1.6.1)$$

Ідея класифікатора в даному потрібна для знаходження класу з максимальною апостеріорної ймовірністю за умови, що всі параметри мають ті значення, які вони мають для примірника, що класифікується. У загальному випадку ця задача передбачає попереднього знання дуже великої кількості умовних ймовірностей і, відповідно, величезного розміру навчальної вибірки і високої складності обчислень, тому на практиці частіше за все застосовується різновид байєсівського класифікатора, який має назву наївний баєсів класифікатор, в якому передбачається, що всі параметри незв’язані між собою, відповідно, формула приймає набагато простіший вигляд і для її використання потрібно знати лише невелику кількість умовних ймовірностей.

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)} \quad (1.6.2)$$

Хоча дане припущення зазвичай є далеким від реальності, наївний байєсівський класифікатор часто показує непогані результати [6].

- дерева рішень - в спрощеному вигляді, даний алгоритм полягає в побудові дерева, в якому кожен вузол відповідає деякому тесту, який виконується над параметрами об'єкту, а листям є підсумкові класи. Існує безліч різновидів дерев рішень і алгоритмів їх побудови. Наприклад – C4.5.
- нейронні мережі - модель, яка надається у вигляді сукупностей елементів (нейронів) пов’язаних між собою, які можуть бути спрямованими або не направлення і мають деякі ваги. В ході роботи нейронної мережі на частину її нейронів, які називаються вхідними, надходить сигнал (вхідні дані), який певним чином поширюється і перетворюється, і на виході мережі (вихідних нейронах) можна бачити результат роботи мережі, наприклад, ймовірності окремих класів.
- машини опорних векторів - концепція алгоритму полягає також, як у випадку логістичної регресії, в пошуку розділяє площині (або декількох площин), проте спосіб пошуку даної площини в цьому випадку відрізняється - шукається площину така, що відстань від неї до найближчих точок - представників обох класів максимально, для чого зазвичай використовуються методи квадратичної оптимізації.
- Lazy learners - особливий різновид алгоритмів класифікації, які, замість того щоб попередньо будувати модель і в подальшому приймати рішення про визначання належності об'єкту до одного чи іншого класу на її основі, ґрунтуються на ідеї, що схожі об'єкти найчастіше мають один і той ж клас. Коли такий алгоритм виходить на вхід об'єкт для класифікації, він шукає серед переглянутих раніше об'єктів схожі на нього і, користуючись інформацією про їх класах, формує свою пророкування щодо класу цільового об'єкта.

Алгоритми класифікації можуть мати найрізноманітніші ідеї та в своїй основі і, зрозуміло, для різних типів завдань показують різну ефективність. Так, для задач з невеликою кількістю вхідних ознак, можуть виявитися корисними системи, засновані на правилах, якщо для вхідних об'єктів можна швидко і зручно обчислити деяку метрику схожості - ледачі класифікатори, якщо йдеться про завдання з великою кількістю параметрів, які до того само важко ідентифікувати або інтерпретувати, таких як розпізнавання об’єктів на зображенні або мови, найбільш підходящим методом класифікації стає нейромережевий метод.

Штучні нейронні мережі є одним за найвикористовуючих типом моделей машинного навчання. Вони базуються на імітації нервової системи тварин і людей.

Спрощена модель нервової системи тварин при цьому представляється у вигляді системи клітин, кожна з них має тіло і відгалуження двох типів: дендрити і аксони. У певний момент клітина отримує сигнали від інших клітин через дендрити і, якщо ці сигнали будуть достатньої сили, збуджується і передає це збудження іншим клітинам, з якими вона пов'язана, через аксони. Таким чином сигнал (збудження) поширюється по всій нервової системи. Модель нейронних мереж влаштована аналогічним чином. Нейронна мережа складається з нейронів і спрямованих зв'язків між ними, при цьому кожна зв'язок має деяку вагу. При цьому частина нейронів є вхідними - на них надходять дані з зовнішнього середовища. Потім на кожному кроці нейрон отримує сигнал від всіх вхідних нейронів, обчислює зважену суму сигналів, застосовує до неї деяку функцію і передає результат на кожен зі своїх виходів. Також мережа має кілька вихідних нейронів, які формують результат роботи мережі. Так, для завдання класифікації вихідні значення цих нейронів можуть означати прогнозовані ймовірності кожного з класів для вхідного об'єкта. Відповідно, навчання нейронної мережі – це підбор спеціальних ваг для зв'язків між нейронами, щоб вихідні значення для всіх вхідних даних виявлялися максимально близькими до дійсних.

Виділяється кілька основних типів нейронних мереж за їх архітектурою:

- мережа прямого поширення (feed – forward network) - має на увазі, ацикільчний граф, що утворенно нейронами і зв'язками між ними, де сигнали поширюються тільки в одному напрямку. Саме такі мережі є найпопулярнішіми і широко вивченими, і їх навчання представляє найменші труднощі;
- рекурентні нейронні мережі (recurrent neural networks) - в таких мережах, на відміну від мереж прямого поширення, сигнали може передаватися в обох напрямках, і можуть надходити на один і той же нейрон кілька разів в процесі обробки одного вхідного значення. Приватної різновидом рекурентних нейронних мереж є, наприклад, машина Больцмана. Основною трудністю в роботі з такими мережами є їх навчання, так як створити ефективний алгоритм є складним завданням і до сих пір не має універсального рішення.
- самоорганізовані карти Кохонена - нейронна мережа, призначена в першу чергу для кластеризації і візуалізації даних.

Розвиток нейронних мереж поділяють на 3 основні періоди підйому. Перші в досліджені штучних нейронних мереж відносяться до 40-х років 20 століття. У 1954 році Дж. Маккалок і У. Піттс опублікували роботу «Логічне числення ідей, що відносяться до нервової діяльності» [4], в якій були викладені основні принципи побудови штучних нейронних мереж. У 1949 році була опублікована книга Д. Хебба «Організація поведінки», де автор розглянув теоретичні основи навчання нейронних мереж і вперше сформулював концепцію навчання нейронних мереж як настройку ваг між нейронами. У 1954 році В.Кларк вперше здійснив спробу реалізувати аналог мережі Хебба за допомогою комп'ютера.

У 1958 році Ф. Росенблатт запропонував модель персептрону, який представляв собою по суті нейронну мережу з одним прихованим шаром [5]. Принциповий вигляд персептрону Росенблатта представлений на рисунку 1.1.

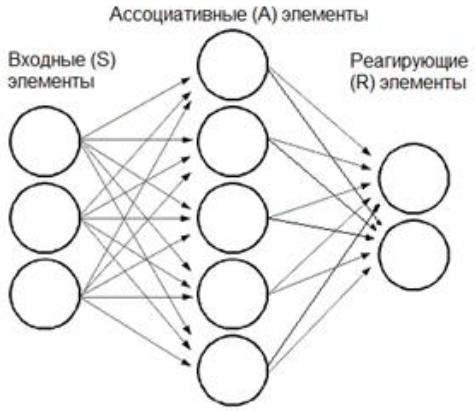

Рисунок 1.1 - Персептрон Росенблатта

Дана модель навчалася за допомогою методу корекції помилок, який полягав в тому, що ваги залишаються незмінними до того моменту, коли вихідне значення персептрону коректне, в разі ж помилки вага зв'язку переходить на 1 в бік, зворотний знаку сталася помилки. Даний алгоритм, як було доведено Росенблаттом, завжди сходиться. Використовуючи таку модель, вдалося створити комп'ютер, що розпізнає деякі букви латинського алфавіту, що, безсумнівно, було великим успіхом в той час.

Однак інтерес до нейронних мереж істотно знизився після публікації М. Мінським і С. Паперт книги «Персептрони» ("Perceptrons") в 1969 році [6], де вони описали істотні обмеження, які має модель персептрона, зокрема, неможливість подання функції виключає або, а також вказали на занадто високі вимоги до необхідніх потужностей комп'ютерів на яких відбувається тренування нейронних мереж. Через те, що ці вчені мали дуже високий авторитет в науковому співтоваристві, нейронні мережі були на деякий час визнані малоперспективною технологією. Ситуація змінилася тільки після створення в 1974 році алгоритму зворотного поширення помилки.

Він був запропонований в 1974 році одночасно і незалежно двома вченими, П.Вербосом і А. Галушкіно. Даний алгоритм заснований на методі градієнтного спуску. Основна ідея алгоритму полягає в поширенні інформації про помилку від виходів мережі до її входів тобто в зворотному напрямку по відношенню до стандартного підходу. При цьому ваги зв'язків коригуються на основі дійшла до них інформації про помилку. Основна вимога, яку накладає даний алгоритм - те, що

функція активації нейронів повинна бути диференційована, так як метод градієнтного спуску, що не дивно, обчислюється на основі градієнта.

Алгоритм зворотного поширення помилки дозволяє легко навчати мережу, що має декілька прихованих шарів, що дозволяє обійти обмеження персептрона, що блокують розвиток даної галузі раніше[11]. Якщо подивитися на даний алгоритм з математичного зору, то ідея зводиться до послідовного перемножування матриць - що є досить добре вивченою і оптимізується завданням. Крім того, даний алгоритм добре паралелізується, що дозволяє істотно прискорити час навчання мережі. Все це разом призвело до нового розквіту нейронних мереж і безлічі активних досліджень в даному напрямку.

Алгоритм backpropagation, в той же час, має ряд проблем. Так, використання градієнтного спуску передбачає ризик сходження до локального мінімуму. Іншою важливою проблемою є тривалий час навчання алгоритму при наявності великої кількості шарів, так як помилка в процесі зворотного поширення має властивість все сильніше зменшуватися при наближенні до початку мережі, відповідно, навчання початкових шарів мережі відбуватиметься вкрай повільно. Ще одним недоліком, властивим нейронних мереж в цілому, є складність в інтерпретації результатів їх роботи. Навчена модель – це чорна скринька, на вхід якої подається об'єкт і на виході виходить прогноз, однак визначити, які ознаки вхідного об'єкта при цьому враховувалися і який з нейронів за що відповідає, зазвичай досить проблематично. Це робить нейронні мережі багато в чому менш привабливими в порівнянні, наприклад, з деревами рішень, в яких навчена модель сама по собі представляє деяку квінтесенцію знань про розглянутої предметної області і досліднику легко зрозуміти, чому цей об'єкт був віднесений до того чи іншого класу.

Дані недоліки, в поєднанні з тим, що, отримувані результати можна було порівняти з результатами інших класифікаторів, наприклад, що набирають популярність машин опорних векторів, при цьому результати останніх були набагато простіше в інтерпретації, а навчання вимагало меншого часу, привело до чергового спаду розвитку нейронних мереж.

Цей спад завершився тільки в 2000-х роках 21 століття, коли з'явилося і стало поширюватися поняття deep learning, або глибокого навчання. Відродженню нейронних мереж сприяла поява нових архітектур, таких, наприклад, як згорткові мережі, restricted bolzman machines, стекові автоенкодери і т.і., які дозволили домогтися істотно більш високих результатів в таких сфера машинного навчання, як розпізнавання образів. Істотним фактором для їх розвитку стало також поява і поширення потужних відеокарт і їх застосування для обчислювальних задач. Відкрите, відрізняючись значно більшою кількістю ядер по порівняння з процесором, нехай і меншої потужності кожне, ідеально підходять для завдань навчання нейронних мереж. В поєднанні з істотно збільшеною останнім часом продуктивністю комп'ютерів в цілому і

поширенням обчислювальних кластерів дозволило навчати істотно більш складні і глибокі архітектури нейронних мереж, ніж раніше.

Однією з найваждивіших проблем, з якими доводиться стикатися при використанні алгоритмів машинного навчання, це проблема вибору правильних ознак, на основі яких проводиться навчання. Особливо значимою стає дана проблема при розгляді таких завдань, як розпізнавання тексту, розпізнавання мови, і тому подібних, тобто тих, де відсутня очевидний набір ознак, які можуть використовуватися для навчання. Зазвичай вибір набору ознак для навчання здійснюється самим дослідником шляхом деякої аналітичної роботи, і саме обраний набір ознак багато в чому визначає успішність роботи алгоритму. Так, для задачі розпізнавання об'єктів на зображені в якості таких ознак може служити переважаючий колір в зображенні, ступінь його зміни, наявність на зображенні чітких меж або щось ще.

Однак такий підхід має істотні недоліки. По-перше, даний підхід має на увазі істотний обсяг роботи щодо виявлення ознак, причому ця робота здійснюється вручну дослідником і може вимагати великих часових витрат. По-друге, виявлення ознак, за допомогою яких можна отримати якісний алгоритм, в даному випадку стає багато в чому випадковим, до того ж, таким чином малоймовірно, що будуть взяті до уваги ознаки, які можуть надавати важливе вплив на внутрішню структуру зображення, але при це неочевидні для людини. Таким чином, особливо привабливою виглядає ідея автоматичного визначення ознак, які в подальшому можна використовувати для роботи з алгоритмами машинного навчання. І саме таку можливість надає використання підходу deep learning.

У теорії машинного навчання, глибоке навчання є підмножиною так званого representation learning.

Основна концепція representation learning - якраз таки автоматичний пошук ознак, на підставі яких в подальшому буде працювати деякий алгоритм, наприклад, класифікації.

З іншого боку, ще одна важлива проблема, з якою доводиться стикатися при використанні машинного навчання - це наявність факторів варіації, які можуть зробити істотний вплив на зовнішній вигляд вихідних даних, однак при цьому не мають відношення до самої їх суті, яку дослідник і намагається аналізувати [8]. Так, в задачі розпізнавання образів такими факторами можуть бути кут, під яким предмет на зображенні повернуть до спостерігача, час доби, освітлення і т.і. Так, в залежності від точки зору і погоди червона машина може мати на фотографії різний відтінок і форму. Тому для подібних завдань, наприклад, ідентифікації предмета, зображеного на фотографії, виглядає розумним враховувати не конкретні низькорівневі факти, такі як колір певного пікселя, а характеристики вищого рівня абстракції, наприклад, наявність коліс.

Однак очевидно, що визначити на основі вихідного зображення, чи присутні у нього колеса - завдання нетривіальне, і її рішення безпосередньо може бути досить складним. Крім того,

наявність коліс - тільки одне з безлічі можливих ознак, і визначення їх всіх і складання алгоритмів для перевірки зображення на наявність їх виглядає не дуже реалістичним. Саме тут дослідники можуть використовувати всі переваги підходу deep learning. Глибоке навчання заснований на наданні вихідного об'єкта у вигляді ієрархічної структури ознак, таким чином, що кожен наступний рівень ознак будується на основі елементів попереднього рівня. Якщо йдеться про зображення, як самого нижчого рівня будуть виступати вихідні пікселі зображення, наступним рівнем будуть відрізки, які можна виділити серед цих пікселів, потім - кути і інші геометричні фігури, в які складаються відрізки. На наступному рівні їхніх постатей утворюються вже впізнавані для людини об'єкти, наприклад, колеса, і нарешті, останній рівень ієрархії відповідає за конкретні предмети на зображенні, наприклад, автомобіль.

Для реалізації підходу deep learning у сучасній науці використовуються нейронні мережі з великою кількістю шарів різних архітектур. Нейронні мережі ідеально підходять для рішеннях завдання виявлення з даних і побудови ієрархічної безлічі ознак, так як, по суті, У той же час, нейронні мережі в своєму найбільш поширеному вигляді, самі по собі представляють ієрархічну структуру, де кожен наступний шар нейронів використовує в якості свого входу виходи нейронів попереднього шару - або, іншими словами, ознаки більш високого рівня формуються на основі ознак нижчого рівня.

Поширенню такого підходу і, в зв'язку з цим, чергового розквіту нейронних мереж, послужило три взаємопов'язаних причини:

- поява нових архітектур нейронних мереж, заточених для вирішення певних завдань (згорткові мережі, машини Больцмана і т.і.);
- розвиток і доступність обчислень з використанням gpu і паралельних обчислень в цілому;
- поява і поширення підходу пошарового навчання нейронних мереж, при якому кожен шар навчається окремо за допомогою стандартного алгоритму backpropagation (зазвичай на нерозмічених даних, тобто по суті відбувається навчання автоенкодера), що дозволяє виявити суттєві ознаки на даному рівні, а потім всі верстви об'єднуються в єдину мережу і відбувається донавчання мережі вже із застосуванням розмічених даних для вирішення конкретного завдання (fine - tuning).

Даний підхід має дві істотні переваги. По-перше, таким чином істотно підвищується ефективність навчання мережі, так як в кожен момент часу навчається неглибока структура, а мережа з одним прихованим шаром - в результаті зникають проблеми зі зменшенням значень помилки у міру підвищення глибини мережі і відповідним зниженням швидкості навчання. І по-друге, даний підхід до навчання мережі дозволяє використовувати при навчанні нерозмічені дані, яких зазвичай набагато більше ніж розмічених - що робить навчання мережі простішим і

доступнішим для дослідників. Розмічені дані в такому підході потрібні тільки в самому кінці для доналаштування мережи на вирішення певної задачі класифікації, і при цьому, оскільки загальна структура ознак, що описують дані, вже створена в процесі попереднього навчання, для доналаштування мережі потрібно значно менше даних, ніж для початкового навчання з метою виявлення ознак. Крім скорочення необхідної кількості розмічених даних, використання подібного підходу дозволяє навчити один раз мережу з використанням великої кількості нерозмічену даних і потім використовувати отриману структуру ознак для вирішення різних завдань класифікації, налаштовувати мережу за допомогою різних наборів даних - за набагато менший час, ніж треба було б в випадку повного навчання мережі кожен раз.

Розглянемо трохи детальніше основні архітектури нейронних мереж, як правило, використовується в контексті глибокого навчання:

- багатошаровий персептрон - являє собою звичайну повнозв'язну нейронну мережу з великим кількість шарів. Питання про те, яка кількість шарів вважається досить великим, немає однозначної відповіді, але зазвичай мережі, що мають 5-7 шарів, вже вважаються «глибокими». Дана архітектура нейронних мереж, хоча і не має принципових відмінностей від мереж, які використовувати раніше до поширення поняття глибокого навчання, може виявитися вельми ефективною в разі успішного вирішення завдання її навчання, що було головною проблемою роботи з такими мережами раніше. В даний час дана проблема вирішується шляхом використання для навчання мережі графічних карт, що дозволяє прискорити навчання і, відповідно, провести більшу кількість ітерацій навчання, або пошарового навчання мережі, згаданого раніше. Так, в 2012 році Ciresan з колегами опублікували статтю «Deep big multilayer perceptrons for digit recognition» [9], в якій зробили припущення, що багатошаровий персептрон з великою кількістю шарів, в разі достатньої тривалості навчання (яка досягається за розумний час з використанням паралельних обчислень на gpu) і достатній кількості даних для навчання (яке досягається шляхом застосування різних випадкових трансформацій до вихідного безлічі даних) може показати результативність не гірших ніж інші, більш складні моделі. Їх модель, що представляє собою нейронну мережу з 5 прихованими шарами, при класифікації цифр з датасета MNIST, показала відсоток помилки 0.35, що краще, ніж опубліковані раніше результати більш складних моделей. Також, шляхом об'єднання кількох навчених таким чином мереж в єдину модель, їм вдалося знизити показник помилки до 0.31%. Таким чином, незважаючи на уявну простоту, багатошаровий персептрон є цілком успішним представником алгоритмів глибокого навчання.

- Stacked autoencoder (стековий автоенкодер) - дана модель тісно пов'язана з багатошаровим персептроном і в цілому із завданням навчання глибоких нейронних мереж. Саме з використанням стекового автоенкодеру реалізується пошарове навчання глибоких мереж. Однак

дана модель використовується не тільки для цілей навчання інших моделей, а часто має велике практичне значення сама по собі. Щоб описати суть стекового автоенкодера, розглянемо спочатку поняття звичайного автоенкодера. Автоенкодер це алгоритм навчання без учителя, в якому в якості очікуваних вихідних значень нейронної мережі виступають її ж вхідні значення. Схематично модель автоенкодера представлена на рисунку 1.2:

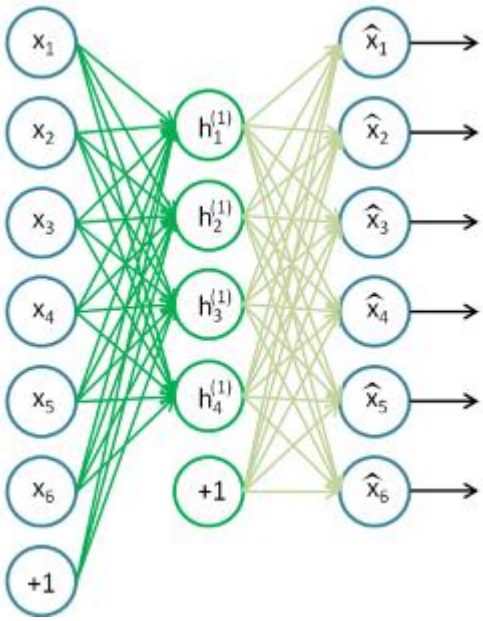

Рисунок 1.2 – Класичний автоенкодер

Очевидно, що завдання навчання подібної моделі має тривіальне рішення, якщо кількість нейронів в прихованому шарі дорівнює кількості вхідних нейронів - тоді прихованого шару досить просто транслювати свої вхідні значення на вихід. Тому при навчанні автоенкодерів вводяться додаткові обмеження, наприклад, кількість нейронів в прихованому шарі встановлюється значному меншим ніж у вхідному шарі, або застосовуються особливі методики регуляризації, спрямовані на те, щоб забезпечити високу ступінь розрідженості нейронів прихованого шару [11]. Одне з найбільш поширених застосувань автоенкодерів в чистому вигляді - це завдання отримання стислого представлення вихідних даних. Так, наприклад, автоенкодер з 30 нейронами в прихованому шарі, навчений на датасета MNIST, дозволяє відновити на вихідному шарі вихідні зображення цифр практично без змін, що означає, що насправді кожна з вихідних зображень можна досить точно описати тільки 30 числами. В даному своєму застосуванні автоенкодери часто розглядаються в якості альтернативи методу головних компонент. Стекові ж автоенкодер є по суті комбінацію декількох звичайних автоенкодерів, яких навчають шар за шаром. При цьому вихідні значення навчених нейронів прихованого шару першого з автоенкодеров виступають в якості вхідних значень для другого з них і т.і.

Згорткові мережі - одна з найбільш популярних останнім часом моделей глибокого навчання, що застосовується у першу чергу для розпізнавання зображень [12]. Концепція згортальних мереж побудована на трьох основних ідеях:

- локальна чутливість (local receptive fields) - якщо говорити про завдання розпізнавання зображень, це означає, що на розпізнавання того чи іншого елемента на зображенні має в першу чергу впливати його безпосереднє оточення, в той час як пікселі, що знаходяться в іншій частині зображення, швидше за всього з даним елементом ніяк не пов'язані і не містять інформації, яка допомогла б правильно його ідентифікувати;
- розділені ваги (shared weights) - наявність в моделі розділених ваг фактично уособлює припущення, що один і той же об'єкт може бути знайдений в будь-якій частині зображення, при цьому для його пошуку у всіх частинах зображення застосовується один і той же патерн (набір ваг);
- сабсемплінг (subsampling) - концепція, що дозволяє зробити модель більш стійкою до незначних відхилень від шуканого патерну - в тому числі, пов'язаних з дрібними деформаціями, зміною освітлення і т.і. Ідея сабсемплінга полягає в тому, що при зіставленні з патерном враховується не точне значення для даного пікселя або області пікселів, а його агрегація в деякому околі, наприклад, середнє або максимальне значення.

З математичної точки зору, основою згорткових нейронних мереж є операція матричної згортки, яка полягає в поелементному перемножуванні матриці, що представляє собою невелику ділянку вихідного зображення (наприклад, 7×7 пікселів) з матрицею того ж розміру, званої ядром згортки, і подальше підсумовування отриманих значень. При цьому ядро звірки є по суті деякий шаблон, а отримане в результаті підсумовування число характеризує ступінь схожості даної області зображення на цей шаблон. Відповідно, кожен шар згорткової мережі складається з певної кількості шаблонів, і завдання навчання мережі полягає в підборі правильних значень в цих шаблонах - так, щоб вони відображали найважливіші характеристики вихідних зображень. При цьому кожен шаблон порівнюється послідовно з усіма частинами зображення - саме в цьому знаходить вираз ідея поділу ваг.

Шари такого типу в згортковій мережі називаються шарами згортки. Крім шарів згортки, в згорткових мережах присутні шари сабсемплінга, які замінюють невеликі області зображення одним числом, тим самим одночасно зменшуючи розмір зразка для роботи наступного шару і роблячи мережу більш стійкою до невеликих змін в даних. В останніх же шарах згортальної мережі зазвичай використовується один або кілька полносвязанних шарів, яких навчають для виконання безпосередньо класифікації об'єктів. В останні роки використання згорткових мереж стало фактично стандартом при класифікації зображень і дозволяє добити найкращих результатів в цій області.

Обмежені машини Больцмана (Restricted Boltzmann Machines) - ще один різновид моделей глибокого навчання, на відміну від згортальних мереж, що застосовується в першу чергу для задачі розпізнавання мови. Машина Больцмана в своєму класичному розумінні являє собою неорієнтований граф, ребра в якому відображають залежності між вузлами (нейронами). При цьому частина нейронів є видимими, а частина прихованими. З точки зору нейронних мереж машина Больцмана являє собою по суті рекурентну нейронну мережу, з точки зору статистики - випадкове Марківське поле. Важливими поняттями для машин Больцмана є поняття енергії мережі і стану рівноваги. Енергія мережі залежить від того, яка кількість сильно пов'язаних між собою нейронів одночасно знаходяться в активованому стані, завдання ж навчання такої мережі полягає в її сходженні до стану рівноваги, при якому її енергія мінімальна.

Основним недоліком подібних мереж є великі проблеми з навчанням їх у загальному вигляді. Для вирішення даної проблеми Дж. Хінтон з колегами була запропонована модель обмежено машини Больцмана (Restricted Boltzmann Machines), яка накладає обмеження на структуру мережі, представляючи її у вигляді двудольного графа, в одній частині якого знаходяться тільки видимі нейрони, а в іншій - тільки приховані , відповідно, зв'язку присутні тільки між видимими і прихованими нейронами. Дане обмеження дозволило розробити ефективні алгоритми для навчання мереж такого виду, завдяки чому було здійснено суттєвий прогрес у вирішенні завдань розпізнавання мови, де дана модель практично витіснила популярну раніше модель прихованих мереж Маркова.

1.7 Аналіз існуючих додатків розпізнавання текстів

1.7.1 ABBYY FineReader

FineReader[13] - система для розпізнавання текстів. Цей програмний продукт може розпізнавати, майже, будь-який шрифт, але можна, якщо потрібно, навчити системи для розпізнавання конкретного шрифту. FineReader має високу точність, та майже не зважає на дефекти друку.

OCR-технології від компанії ABBYY також підтримують зональне розпізнавання (на рівні полів), необхідне в багатьох ключових бізнес-процесах, таких як класифікація за ключовими словами, індексування за ключовими словами і введення даних з форм PDF / A, searchable PDF, CSV і текстові (plain text) файли.

Інтерфейс

Інтерфейс користувача може бути налаштований на свій розсуд:

- змінити масштаб вікон і розташування;
- кастомізувати панель швидкого доступу, до команд, що використовуються найчастіше;
- налаштувати гарячі клавіші – вже є заготовлені гарячі клавіші, але можна змінити стандартні та додати свої;
- вибрати потрібну мову інтерфейсу та ін.

Робочий простір:

- головне вікно програми;
- панелі інструментів;
- налаштування робочого простору програми під кожного користувача;

Можливості:

- дозволяє витягувати текстові дані з цифрових зображень;
- отримане в результаті розпізнавання може бути збережено в різних форматах.

Додаткові можливості:

- використання шаблонів;
- розпізнавання з навчанням;
- колективна робота в мережі.

1.7.2 CuneiForm

CuneiForm - це програма для оптичного розпізнавання тексту документів в редагований вигляд. Результати розпізнавання можна редагувати у офісних програмах, текстових редакторах, зберігати у популярних форматах, та проводити пошук по розпізнаному тексту.

CuneiForm – це одна найперших систем, яка з’явилась ще до промислових систем розпізнавання. Велика кількість технологічних ноу-хау, результати досліджень, що були покладені в основу CuneiForm, зараз успішно застосовуються і удосконалюються донині в комерційних продуктах Cognitive Technologies.

Можливості:

- при розпізнаванні за допомогою CuneiForm зберігається структура документа і його форматування;
- програма розпізнає таблиці різних структур і складностей, в тому числі і без відображення ліній табличній сітки;
- розпізнає майже зі всіх джерел;
- системи розпізнавання тектсу, які вбудовані у програму дозволяють розпізнавати текст з джерел, що можуть мати дефекти, наприклад матричний принтер, ксерокопія чи факс;
- У програмі вбудована словникова перевірка, яка підвищує якість розпізнавання. Стандартний словник можна розширити за допомогою функції імпорту з текстових файлів.

Переваги CuneiForm:

- практично єдина безкоштовна OCR-програма професійного рівня;
- велика кількість мов розпізнавання;
- простий і зрозумілий інтерфейс;
- кроссплатформенность(Windows, Mac OS, Linux).

1.7.3 OCRopus

OCRopus - система оптичного розпізнавання символів, спочатку спрямована на перетворення в електронний вигляд великого обсягу документів на базі власного розпізнає ядра. Це програми для розпізнавання, який розповсюджується за типом OpenSource. Програма дозволяє підключати додаткові модулі для аналізу макета вмісту, попереднього поліпшення зображення. За задумом розробників, OCRopus має використовуватися для визначення символів на графічних файлках за рядками та конвертувати його у звичай текст, який можна редагувати.

Програма розпізнає і рукописні символи, але OCRopus немає графічного інтерфейсу, а тільки має підтримку командного рядка. Додаток працює лише на операційній системі Linux.

1.7.4 Tesseract

Tesseract - однойменний зі своїм ядром розпізнавання, додаток, що поширюється за вільною системою, призначнений для розпізнавання текстів, розроблений Hewlett-Packard у 1980 році, але через декілька років був закритим. У серпні 2006 року компанія Google викупила її та відкрила вихідні тексти під ліцензією Apache 2.0, щоб продовжити роботу над перспективним проектом. Використовує командний рядок. Підтримує більш 100 мов.

1.8 Постановка задачі

В результаті проведеного аналізу були виявлені недоліки у вже існуючих додатках:

- багато з додатків мають не цілком зрозумілий інтерфейс;
- розглянуті додатки не мають можливості працювати у фоновому режимі;
- не мають графічного інтерфейсу користувача у деяких рішень;
- жоден з додатків не має змоги розпізнавати текст у реальному часі, а робить це тільки після зйомки фото;

Після проведеного аналізу прийнято рішення використовувати підхід з використанням машинного навчання. А саме використовувати Google ML Kit для розробки додатка на базу операційної системи Android.

Висновок до розділу 1

У розділі було показано існуючі методи розпізнавання образів та існуючі додатки для розпізнавання тексту. Після огляду існуючих методів та додатків було виділено машинне навчання, як найбільш підходящий варіант для вирішення поставленого завдання.

РОЗДІЛ 2 АНАЛІЗ ПІДХОДУ НА БАЗІ МАШИННОГО НАВЧАННЯ

У попередньому розділі були проаналізовані підходів розпізнавання тексту та обрано підхід з використанням машинного навчання. Тому було вирішено використовувати Google ML Kit.

2.1 Google ML Kit

Не секрет, що інтеграція алгоритмів на основі доповненої реальності в мобільні додатки непросте завдання, і це вимагає хороших навичок і багато часу. Для початку розробникам потрібно вивчити деякі відповідні бібліотеки машинного навчання, такі як TensorFlow, Torch, PyBrain, Azure тощо. Потім нейронному мережу слід навчити виконувати всі необхідні завдання. Потім побудовати модель машинного навчання, яка буде не дуже важкою, оскільки вона повинна працювати безперебійно на мобільних пристроях. І тільки після того, як модель побудована, розробники можуть трохи перевести подих. Як бачите, це завдання є досить складним навіть для досвідчених розробників[14].

Але ML Kit SDK дозволяє розробникам робити все швидше і простіше. Розробникам потрібно передати дані до потрібного API та чекати відповіді від SDK. Представники Google запевняють, що розробникам не потрібно бути висококваліфікованими в нейронних мережах, щоб реалізувати свої API. Все, що потрібно розробникам - це додати кілька рядків кодів, і новий додаток отримає класні функції на основі AI.

Google ML Kit гарний варіант для кваліфікованих розробників. Якщо існуючі ML Kit API не надають розробникам необхідні функції, розробник може використати власну модель ML. На веб-сайті Firebase є пояснення того, як розробники можуть це зробити і отримати вигоду від своїх моделей.

ML Kit працює з TensorFlow[15], бібліотекою машинного навчання для операційних систем iOS та Android. Ця бібліотека дозволяє розробникам завантажувати свою модель на консоль Firebase і зв'язувати її зі своїм продуктом. Якщо вбудована модель занадто важка, розробники можуть залишити її в хмарі, а потім організувати динамічне завантаження в додаток. Це дає змогу зменшити розмір програми, і користувачі завантажуватимуть його швидше. Крім цього, оновлення моделей також динамічні, і це важливо. Це означає, що модель буде оновлена, тоді як вся програма не буде оновлена.

Розробники можуть вибирати API, що буде використовувати хмарові потужності чи пристрою, залежно від їх потреб. Якщо ви не знаєте, який варіант краще, вам слід врахувати всі відмінності, щоб зробити правильний вибір. Хмарні API обробляють дані на платформі Google Cloud, тому розпізнавання об'єктів виконується більш точно.

Однак хмарних моделей більше, ніж моделей на пристроях. Моделі на пристроях вимагають менше місця, вони можуть функціонувати в режимі офлайн, а обробка даних відбувається швидше, але ці моделі не такі точні.

ML Kit SDK - це багатоплатформна структура, тобто API може бути використано для додатків iOS та Android. Таким чином, новий ML Kit може конкурувати з CoreML від Apple. Однак сьогодні CoreML має більше переваг, ніж ML Kit. Наприклад, CoreML використовує TensorFlow, а також приймає ONYX, інструменти Python, Apache MXNet.

ML Kit включає в себе (рис. 2.1):

- Google Cloud Vision API;
- TensorFlow lite;
- Neural Network API.

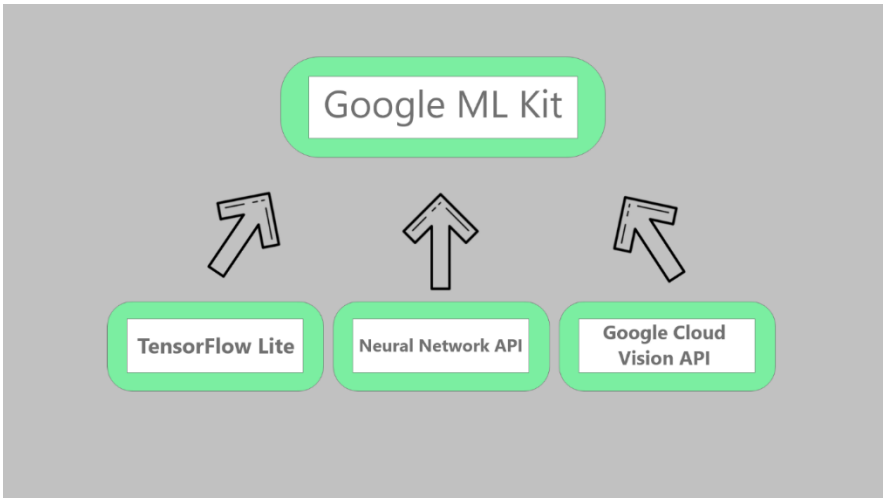

Рисунок 2.1 – Архітектура Google ML Kit

Google Cloud Vision API - хмарова система яка обробляє данні які передає користувач, як раз за рахунок Google Cloud Vision API ML Kit може оброблювати деякі данні у хмарі не використовуючи потужності локального пристрою.

TensorFlow lite – це бібліотека, спеціально розроблена для навчання на мобільних пристроях.

Бібліотека швидко ініціалізується та запускає модель машинного навчання. Моделі такого типу можна навчати, на всіх мобільних операційних системах. Вбудована підтримка апаратного прискорення на мобільних пристроях.

На даний момент більшість мобільних пристроїв мають вбудовані апаратні засоби для ефективної обробки навантажень під час машинного навчання. Нейромережевий API, який використовується з TensorFlow Lite підтримує прискорення машинного навчання.

2.2 Архітектура TensorFlow Lite

TensorFlow Lite моделі швидко працюють на мобільних пристроях, через те що залучають центральний процесор, якщо апаратні засоби прискорення недоступні (рис.2.2).

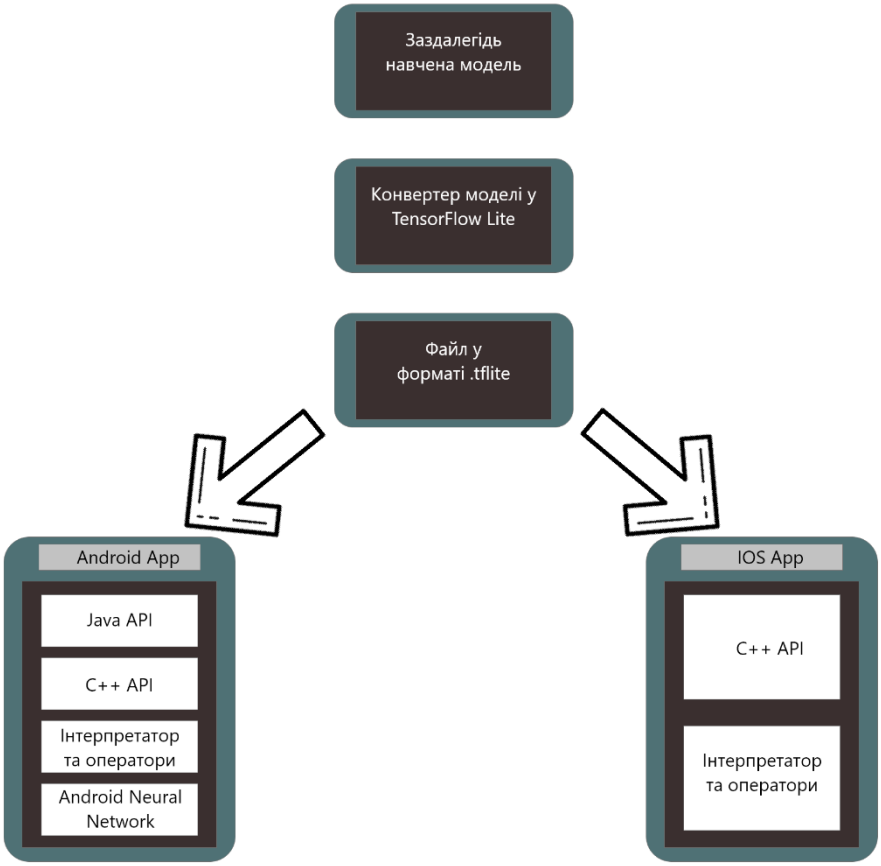

Рисунок 2.2 – Архітектура роботи бібліотеки TensorFlow Lite

На першому етапі маємо TensorFlow Model, яка була заздалегідь навчена, далі ця модель перетворюється у TensorFlow Lite модель, за допомогою вбудованого конвертора TensorFlow Lite Converter, потім отриманий файл з розширенням .tflite, який був оптимізовано для максимальної швидкості та мінімізовано розмір.

Після конвертації моделі у формат .tflite, розробник обирає на якому пристрої буде використовуватися розроблений додаток, вибір між Android та IOS, далі вибір між двома API, одне на мові Java, яке є оболонкою C++ API і використовується тільки на базі ОС Android. Друге це C++ API, яке викликає інтерпретатор. Цей варіант підходить, і для Android, і для IOS.

На наступному етапі за допомогою набору спеціальних операторів інтерпретатор за пускає модель. Інтерпретатор може мати динамічну кількість операторів, завдяки цьому вага його вага

може бути від 70 кб до 300 кб, якщо завантажені усі оператори. Розробник може сам обрати потрібні оператори.

Для апаратного прискорення на пристроях, що підтримують Android Neural Network API інтерпретатор буде використовувати його для апаратного прискорення.

TensorFlow Lite містить у собів декілька моделей, заздалегіть навчених і оптимізованих для мобільних пристроїв:

Спеціальні моделі класу MobileNet для комп'ютерного зору, які розроблені для ефективної роботи на мобільних та вбудованих пристроях.

Inception v3 аналог MobileNet, високоточна модель розпізнавання образів, але має більший розмір ніж попередня модель.

Розумна система відповідей на вхідні повідомлення з назвою Smart Reply, що використовується надає список відповідей проаналізувавши прийняте повідомлення.

2.3 Нейромережевий API (Android Neural Networks API)

API нейронних мереж Android (NNAPI) - це Android API, призначений для виконання обчислювально інтенсивних операцій для машинного навчання на пристроях Android. NNAPI призначений для забезпечення базового рівня функціональності для систем машинного навчання вищого рівня, таких як TensorFlow Lite та Caffe2, які будують і тренують нейронні мережі. API доступний на всіх пристроях Android з ОС Android 8.1 (рівень API 27) або вище[16].

NNAPI підтримує inferencing(виведення), застосовуючи дані з пристроїв Android до їх попередньо підготовлених розробниками моделей. Приклади виведення це включення класифікації зображень, прогнозування поведінки користувачів та вибір відповідних відповідей на пошуковий запит.

Розробнику не потрібно надсилати запит через мережеве з'єднання та чекати відповіді. Наприклад, це може бути критично важливим для відеопрограм, які обробляють послідовні кадри, що надходять з камери.

Доступність і швидкість: програма запускається навіть поза межами мережі інтернет, чим потужніший ваш пристрій тим швидше буде працювати додаток, тому що NNAPI забезпечує значно швидше обчислення, ніж єдиний процесор загального призначення.

Конфіденційність: дані не залишають пристрою Android.

Вартість: всі обчислення виконуються на пристрої Android, не потрібна ферма сервера.

Є також компроміси, про які розробник повинен пам'ятати.

Використання системи: Оцінка нейронних мереж потребує значних обчислень, що може збільшити споживання енергії акумулятора. Ви можете розглянути можливість контролю стану акумулятора, якщо це турбує вашу програму, особливо для довготривалих обчислень.

Розмір програми: зверніть увагу на розмір ваших моделей. Моделі можуть займати багато мегабайт місця. Якщо поєднання великих моделей у вашій APK неправомірно вплине на ваших користувачів, можливо, якщо вам буде потрібно завантажити модулі після встановлення додатка, використання менших моделей або проведення обчислень у хмарі. NNAPI не забезпечує функціональність для запуску моделей у хмарі.

NNAPI можна називати бібліотеками, рамками та інструментами машинного навчання, що дозволяють розробникам навчати свої моделі поза пристроєм та розгортати їх на пристроях Android. Зазвичай програми не використовуватимуть NNAPI безпосередньо, а замість цього використовуватимуть структури машинного навчання вищого рівня. Ці рамки, в свою чергу, можуть використовувати NNAPI для виконання апаратних прискорень операцій висновку на підтримуваних пристроях.

Виходячи з вимог програми та апаратних можливостей на пристрої Android, час роботи нейронної мережі Android може ефективно розподіляти обчислювальну навантаження між доступними процесорами на пристроях, включаючи спеціальне обладнання нейронної мережі, одиниці графічної обробки (графічні процесори) та цифрові процесори сигналу (DSP).

Схема роботи додатку, приблизно, має наступний вигляд. Додаток звертається до фрейворку чи бібліотеки машинного навчання, фреймворк чи бібліотека звертаються до NNAPI (рис.2.3), якщо цей API підтримується пристроєм. Потім NNAPI надсилає запити до NN драйверів, які у свою чергу звертаються до процесорів приладу.

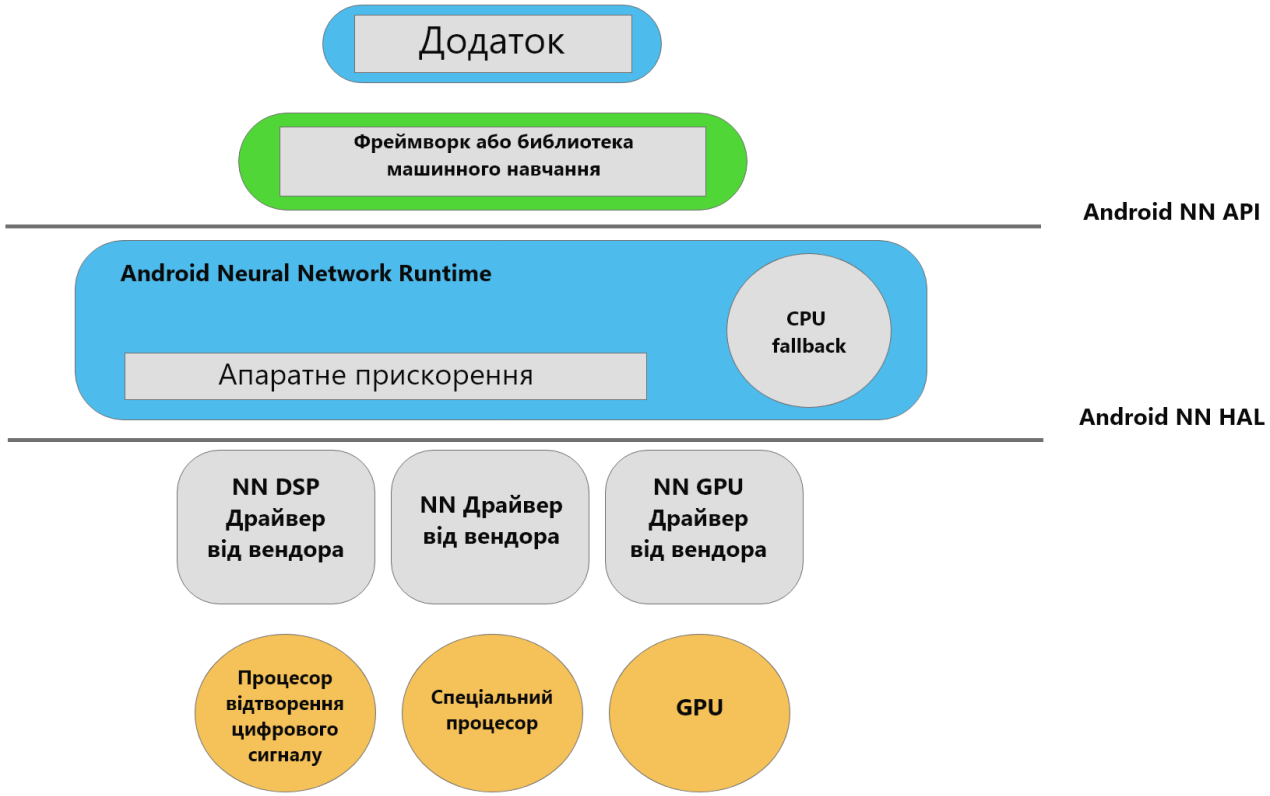

Рисунок 2.3 – Архітектура Android Neural Network API

Для пристроїв Android, у яких відсутній спеціалізований драйвер постачальника, NNAPI виконує запити прямо до процесора[17].

Для прикладу можна навести декілька запитів фреймворка чи бібліотеки до драйвера постачальника (рис.2.4).

Перший – це запит можливостей драйвера, у відповідь драйвер надсилає список своїх можливостей. Цей етап називається ініціалізацією.

Другий – це запит на список операторів, які підтримує драйвер для цієї моделі. Відповідь це список операторів, що підтримуються. Це етап перевірки підтримки.

Третій - запит на виконання моделі. На цей запит, зазвичай, не надається відповідь. Це етап запуску.

Четвертий – фреймворк надсилає запит із списком вхідних і вихідних операндів для виконная. Відповідь – це звіт про виконання запиту. Етап виконання.

І наступний, п’ятий етап – це запит на закриття обміну запитами.

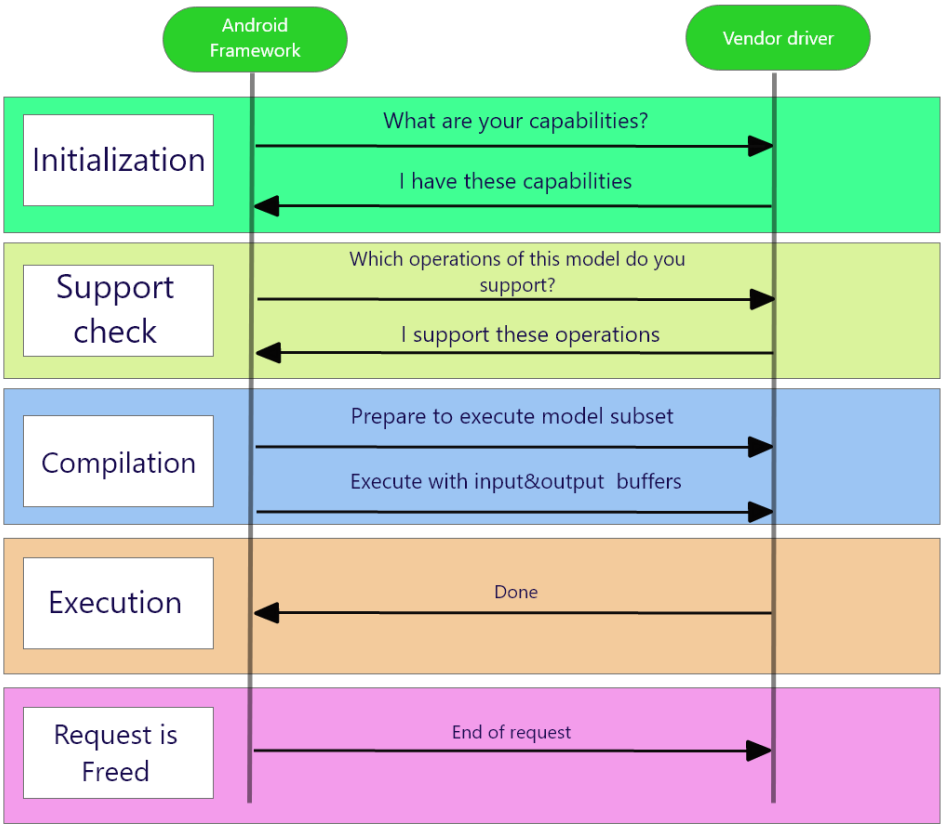

Рисунок 2.4 – «Спілкування» фрейворку та дрейверів постачальника пристрою

ML Kit SDK може працювати бездоганно з іншими службами Firebase. Позначення зображень можна зберігати у Cloud Firestore. Або Google Analytics можна використовувати для вимірювання залучень користувачів. Крім того, A / B тестування користувацьких моделей ML можна легко виконати за допомогою віддаленої настройки та ML Kit.

2.4 Можливості Google ML Kit

Основні можливості наведені у таблиці, які Google ML Kit надає розробнику наведені у табиці 2.1

Таблиця 2.1 – Можливості Google ML Kit

Особливість	На пристрої	У хмарі
Розпізнавання тексту	+	+
Розпізнавання об'єктів на зображенні	+	+
Визнання орієнтирів(наприклад точок на обличчі)	-	+
Сканування штрих-кодів	+	-
Розпізнавання обличь	+	-
Розпізнавання об'єктів та стеження за ними	+	-
Визначення мови	+	-
Переклад	+	-
«Розумна відповідь»	+	-
Тренування моделі	+	-
Власна модель	+	-

2.4.1 Розпізнавання тексту

Ця функція доступна як у хмарі, так і на пристрої. API розпізнавання тексту використовується для розпізнавання будь-якої мови на основі латині у текстовій формі. Це бути використовуватися для автоматизації дратівливого введення даних для кредитних карток, квитанцій та візитних карток. API на основі хмар дозволяє розробникам витягувати текст із зображень документів. Цей текст можна використовувати тоді для перекладу документів або збільшення доступності. Крім того, програми з функцією розпізнавання тексту можуть читати номери на реальних об'єктах. Наприклад, клієнтам потрібно дістати дані з картинки в магазині.

2.4.2 Маркування зображень

ML Kit API дозволяє розпізнавати об'єкти на зображеннях. Немає необхідності в наданні додаткових метаданих, і розпізнавання реалізується окремо. Ця функція доступна також офлайн і в хмарі. За допомогою функції маркування зображень у мобільному додатку користувачі зможуть знати все, що зображено на малюнку. Користувачі отримали список усіх об’єктів, розпізнаних на цьому зображенні: людей, тварин, будівель тощо. Кожна мітка поставляється з оцінкою, яка вказує

на рівень довіри моделі ML в розпізнаванні того чи іншого об'єкта. Використовуючи цю функцію, користувачі можуть виконувати автоматичну генерацію метаданих та модерацію вмісту.

2.4.3 Визнання орієнтирів(наприклад точок на обличчі)

Цей API доступний лише у хмарі. Це дає можливість розпізнати відомі орієнтири на будь-якому зображенні. Щоб перевірити це, розробники можуть розмістити зображення в цьому API, і тоді будуть показані всі розпізнані орієнтири. Крім цього, вказуються також геокоординати орієнтиру. Тому можна побачити, звідки зроблено цю картину. Ця інформація може використовуватися для створення метаданих, надаючи користувачам персональний досвід.

2.4.4 Виявлення обличчя

API для виявлення обличчя на зображеннях, ключових рис обличчя та побудови контурів виявлених обличь. Ця функція використовується лише офлайн. За допомогою цього API користувачі можуть витягувати обличчя із зображень та редагувати їх за допомогою різних фільтрів. Також можна створювати аватари з фотографій користувачів. Оскільки ML Kit надає користувачам можливість застосовувати функцію розпізнавання облич у режимі реального часу, розробники можуть інтегрувати її у відеочаті чи ігри. Отже, якщо вам потрібно розробити додаток для розпізнавання обличь, ви можете скористатися цим API з ML Kit.

2.4.5 Сканування штрих-коду

API сканування штрих-коду дозволяє користувачам читати дані зі штрих-кодів, використовуючи більшість стандартних форматів штрих-коду. Сканування виконується на пристрої, і підключення до Інтернету не потрібно. За допомогою штрих-кодів користувачі можуть з’ясувати, що приховано в кодованих даних. Наприклад, це може бути контактна інформація або будь-які платіжні дані. Ця функція дуже зручна для виявлення закодованих даних.

2.4.6 Виявлення та відстеження об'єктів

Цей дивовижний API, який працює на пристрої, допоможе користувачам локалізувати та відстежувати об’єкти на зображенні чи в прямому ефірі камери. Виявлений об'єкт можна класифікувати до вибраної загальної категорії. Якщо користувачам потрібно створити досвід

візуального пошуку, ця функція може бути вигідною. Коли всі об'єкти виявлені, користувачі можуть надсилати їх у хмарний бек-енд або власну модель, якщо розробники будують цю модель.

2.4.7 Ідентифікація мови

З цією функцією все зрозуміло. API ідентифікації мови дозволяє ідентифікувати мову з одного текстового рядка. Ця особливість може бути особливо корисною для перекладачів чи істориків, яким потрібно з’ясувати, якою мовою написано зображення чи документ.

2.4.8 Переклад

Цей API такий же простий, як і попередній. Він використовується для перекладу тексту до 59 мов, і користувачі можуть перемикатися між мовами. Тобто, можна вибрати різні комбінації перекладів, використовуючи доступні 59 мов. API для перекладу використовує ті самі моделі, які використовує автономна програма Google Translate.

2.4.9 «Розумна відповідь»(Smart Replay)

Інтелектуальний API відповіді генерує пропозиції відповідей на основі всієї бесіди. І ця функція може створити повну пропозицію, а не лише короткі відповіді, такі як "так" чи "ні". Таким чином, користувачі відповідають на повідомлення швидко, оскільки всі відповіді формуються автоматично. Але наразі підтримується лише англійська мова, і цю функцію можна використовувати лише у випадкових розмовах, а не для конкретних дискусій.

2.4.10 Використання власної моделі

Функція виводу моделі AutoML дозволяє розробникам тренувати свої моделі маркування зображень. Модель API маркування зображень на пристрої навчається для визначення до 400 різних категорій. Але може знадобитися звузити кількість об’єктів, щоб розпізнати більш конкретні. Наприклад, користувачі повинні будуть відрізняти породу собак від іншої. І тут інструмент AutoML Vision Edge використовується для тренування потрібної моделі зі своїми зображеннями, завантаженими розробниками.

Висококваліфіковані розробники, які не знайшли відповідної готової моделі, можуть створити свою власну модель за допомогою ML Kit. Для цього розробники можуть використовувати модель TensorFlow Lite з ML Kit.

2.5 Принципи комп’ютерного моделювання

Спочатку комп’ютерне моделювання використовувалося для проведення аналізу, дослідження з імітаційними моделями. Але з розвитком графічних інтерфейсів та графічних пакетів набуло популярності структурно-функціональне моделювання для створення систем штучного інтелекту.

Комп’ютерна модель включає в себе образ об’єкта, системи, чи процесів, які описуються за допомогою таблиць, діаграм, графіків, рисунків, анімацій, схем, що зв’язані між собою. Ці елементи відображають структуру та зв’язки всередині об’єкта дослідження[18]. А також програмну частину, яка може включати в себе декілька програмних продуктів, які послідовно виконують обчислення та графічно відображають результати системи імітації моделі, що моделюється.

Загалом, комп’ютерне моделювання – це метод для розв’язання задачі аналізу чи синтезу складної системи, що базується на комп’ютерній моделі.

Модель, що досліджується має відповідати декільком вимогам[19] Модель має бути повною, тобто надавати можливість обчислення усіх характеристик системи з високою точністю та достовірністю. Гнучкість, модель дозволяє відтворювати і програвати різні ситуації і процеси, змінювати структуру та параметри досліджуваної системи. Швидкість розробки і реалізації – характеристика часових витрат на створення моделі. Блоковість, модель має надавати змогу додавати, змінювати чи видаляти деякі блоки з моделі. А також зручну роботу користувачу, зв’язок з базою даних.

Комп’ютерне моделювання складається за наступних етапів (рис. 2.5).

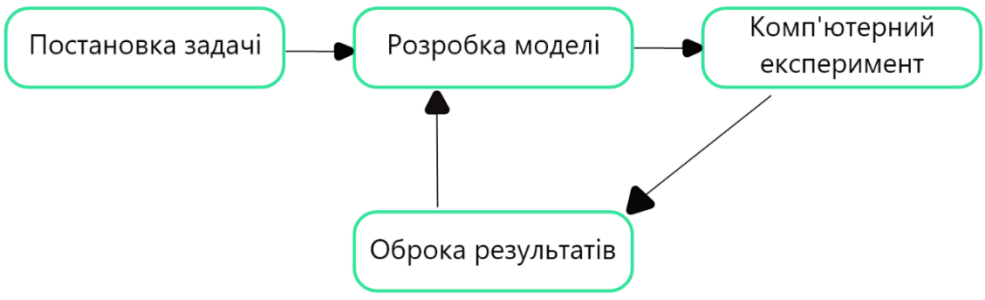

Рисунок 2.5 – Етапи комп’ютерного моделювання

Постановка задачі включає в себе опис моделі, а саме вхідні дані, описання форми отримання результатів моделювання.Проведення аналізу сфери моделювання, визначення мети, та як дані, що отримані будуть використовуватися. Визначення умов при яких можливе отримання необхідних результатів. І останнє визначення, які результати вважати вірними.

Розробка моделі – це написання розгорнутого опису задачі, заміна її на математичну модель за допомогою математичних залежностей. Обрати та обґрунтувати метод розв'язання задачі.

Комп'ютерний експеримент – це цьому етапі визначається або створюється алгоритм розв'язання задачі. Найчастіше метод, який було обрано для розв'язання задачі визначає алгоритм, але і існують методи, які можна розв'язати різними алгоритмами.

На останнюму етапі обробки результатів обробляються результати, перевіряється чи можливо отримані результати вважати вірними. Якщо отримані результати не влаштовують, то приймається рішення корегувати модель.

Комп'ютерна модель має відповідати деяким принципам [20] а саме:

- адекватність, модель має враховувати усі сторони об'єкту, що досліджується, відображувати його властивості.
- простота і економічність, модель має бути простою, щоб використовувати її було ефективно та економічно вигідно. чим складніше модель тим вона дорожча.
- здійсненність. модель має виконувати усі поставленні цілі за для виконання дослідження за поставлений термін.
- множинність та єдність. модель відображує лише одну частину реальної системи. щоб повністю проаналізувати великі процеси, потрібно побудувати декілька моделей, що будуть доповнювати одна одну.
- системність. система, що досліджується може бути представлена, як сукупність взаємодіючих підсистем, які моделюються математичними методами. хоча властивості системи не являються сумою властивостей системи.
- параметризація. підсистеми, модельованої системи можуть бути охарактеризовані векторами, матрицями, графіками чи формулами.

2.6 Схема розпізнавання тексту

Для правильного проектування роботи додатка потрібно побудувати схему роботи додатка (рис. 2.6). На першому етапі додаток отримує вхідний потік даних за камери - це текстові, друковані дані. На наступному отримані дані розпізнаються на блоки тексту, потім на рядки, а потім на окремі слова. Після цього користувачу виділяється текст, який розпізнано. Наступний етап очікування натискання користувача на текст і останній етап аудіо відтворення тексту.

Рисунок 2.6 – Схема моделі роботи додатку

Схема розпізнавання тексту має наступний вигляд (рис. 2.7). Дані з камери отримуваються CamerSource, далі переходять RecProccessor, який аналізує чи це текстову данні та структурує їх, потом переходять у 4 блок, яким самє розпізнає текст, наступний етап обведення текст на екрані і відображення розпізнаного тексту, далі очікується натискання користувача на текст, відбувається перевірка куди саме користувач натиснув, якщо користувач натиснув на місце де немає тексту, то видається помилка, якщо на текст – аудіо відтворюється текст.

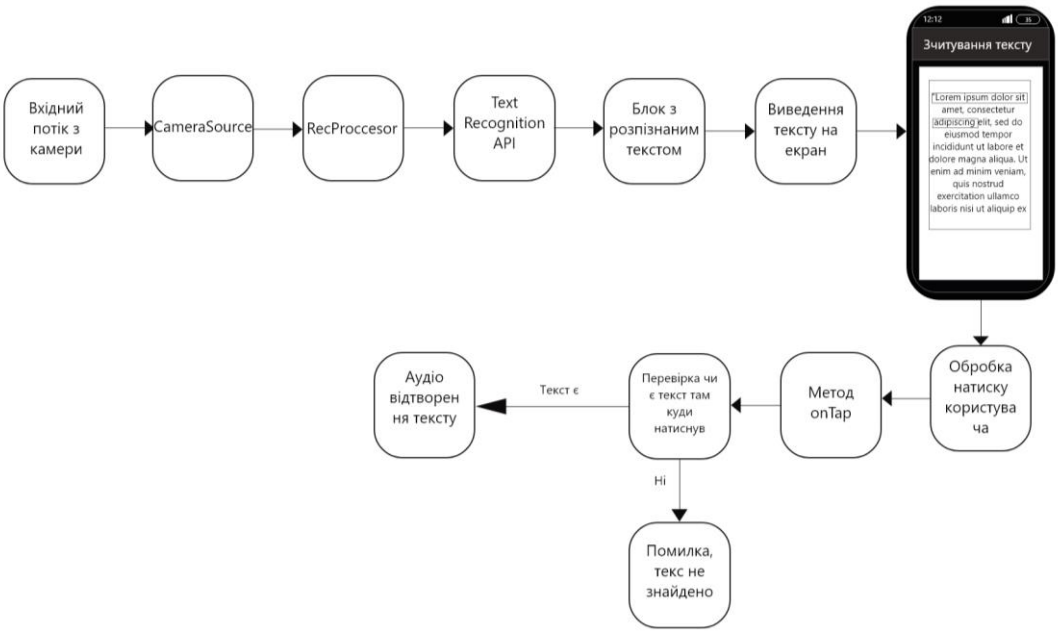

Рисунок 2.7 – Схема розпізнавання тексту

2. 7 Висновок до розділу 2

Було проаналізовано спеціальну систему для інтеграція глибокого навчання на мобільні додатки Google ML Kit, яка підтримується Android Studio. А також розроблено схему роботи додатка, для реалізації поставленого завдання.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ДОДАТКА

Для реалізації додатку потрібно обрати середовище розробки та мову програмування. Було вирішено обирати серед мов Kotlin, Java та C#, а серед середовищ розробки вибір пав на Xamarin Studio, Android Studio, Microsoft Visual Studio.

3.1 Вибір мови програмування

3.1.1 Мова програмування Java

Спочатку мова Java була названа Oak та створена була Джеймсом Гослінгом для управління побутовими пристроями. Через деякий час мова отримала назву Java і почала використовуватися задля забезпечення управління серверними пристроями, або для створення клієнтських додатків. На логотипі мови можна побачити чашку гарячої кави, за однією з версій саме на честь кави Java мова отримала свою назву. А за іншою версією чашка кави на логотипі – це кава – машина, побутовий пристрій, для яких спочатку мова і була створена.

Java – це об'єктно-орієнтована мова програмування, була створена компанією Sun Microsystems, яка в подальшому стала частиною Oracle. Датою офіційного запуску для використання називається - 23 травня 1995 року. Java не залежить від архітектури комп'ютера, тому що запускається у віртуальній Java-машині.

Взявши за основу мову C++, Java відкинула з неї речі, які є потенційно небезпечні, такі як вказівники, що не контролюються виходом за межі. Віртуальна машина була створена для роззсередеженних обчислень і є посередником між текстом програми та інструкціями комп'ютера чи іншого пристрою.

Мова Java одна з найпопулярніших для створення мобільних додатків на базі операційної системи Android. Додатки можна розробляти, як у середовищі Android Studio, так і у таких середовищах як Xamarin Studio, Microsoft Visual Studio з використанням спеціальних плагінів Android Development Tool чи IntelliJ Idea.

3.1.2 Мова програмування Kotlin

Kotlin – мова програмування, яка розробляється компанією JetBrains, що працює поверх віртуальної машини Java і також може бути скомпільована у JavaScript. Мова отримала свою назву на честь острова, що знаходиться у Фінській затоці,

Авторами була поставлена мета створити більш лаконічну та безпечнішу мову, ніж Java, та простішу ніж мова Scala. У порівнянні зі Scala, Kotlin більш швидше компліюється та має кращу підтримку IDE.

Розробка Kotlin почалась у 2010 році, перша версія була представлена через рік у липні 2011. А ще через рік було відкрито код, та випущена перша версія, яка містила плагін IDEA. Через декілька місяців з'явилась підтримка Android. Зимою 2012 року вийшла 4 версія, яка ввела підтримку Java 7, а взимку 2015 з'явилась перед-релізна версія 1.0. На початку 2016 року була випущена перша стабільна версія з номером 1.0.

Весною 2017 року на своїй конференції для розробників компанія Google, додала мову Kotlin до списку офіційно підтримуваних для створення додатків для операційної системи Android.

3.1.3 Мова програмування C#

C# - це об'єктно-орієнтована мова, розроблена за типізацією платформ .NET. Мова була розроблена під наглядом Microsoft Research. як заміна мови Java.

Синтаксис мови C# схожий на C++ чи Java. Підтримує строгу статичну типізацію, поліформізми, оператори можуть бути перезавантаженні, підтримується перезавантаження функцій-членів класів, атрибутів, властивостей, подій, винятків, коментарі можна писати у форматі XML.

C# спирається на досвід попередників та на практику їх використання, тому вимикає деякі модулі, які задавали проблем під час розробки додатків, таких як проблема множинного спадкування.

C# є послідовником мови Java. яка стала дуже популярною і отримала ліцензування компанією Microsoft. Через деякий час компанія Sun звинуватила компанію Microsoft, у тому що та порушила норми ліцензійної угоди і суперечить самій концепції машино-незалежного середовища виконання і виключає Java з платформи Windows. Компанія Microsoft відмовилася прияти вимоги компанії Sun та справа перейшла до судового процесу. Суд прийняв позицію Sun і приняв рішення наказати Microsoft зупинити використовування мови Java ліцензією.

У такій ситуації компанія Microsoft прийняла рішення створити заміну мови Java, яку назвала C# і тепер має повні права на використання її. Із Java була взята концепція віртуальної машину, високу безпеку вихідного коду програм.

Одним з нововведень C# - це легша взаємодія, різних мов програмування у одному проекті, що є важливим нововведенням. Віртуальна машина .NET при використанні різних мов програмування автоматично займається сумісністю типів даних, значень змінних, інакше кажучи бере на себе заходи щодо сумісності програм.

Зараз C# визнана головною мовою корпорації Microsoft, тому що інші мови програмування не повністю підтримують віртуальну машину .NET, а C# повністю підтримує усі можливості .NET.

Символ решітки # можна інтерпретувати, як дві пари ++, ніби натякаючи на новий крок розвитку мови C++, або як музичний символ дієз, який разом з символом C, означає ноту до-дієз. Яке і стало назвою мови. Незважаючи на те, що символ # де-факто – це символ позначення номера на більшості клавіатур.

C# створювалась, як мова прикладного рівня для CLR і тому має велику залежність від можливостей CLR. Це вплинуло, наприклад, на системи типів у C#. З кожним оновленням C# отримувала все більше можливостей.

Специфікація C# передбачає невеликий набір типізованих бібліотек та бібліотеки класів. Зазвичай C# використовується разом з Common Language Infrastructure.

Стандартним компілятор для C# є Microsoft Visual Studio. Також існують і інші компілятори, вони зазвичай включають в себе CLI та бібліотеки класів .NET.

Такими компіляторами, наприклад є проект Microsoft Rotor, який ліцензовано тільки для навчання і дослідницької діяльності.

Ще існує SharpDevelop, створений компанією icsharpcode, альтернатива Visual Studio, підтримує повну реалізацію CLI. Цей компілятор має схожий інтерфейс до Visual Studio і тому робить перехід розробника більш простим.

Ще одним схожим проектом є Mono, який був розпочато компанією Xamarin. Це компілятор з відкритим код, який підтримує CLI, включає в себе усі фреймворки бібліотеки які потрібні для отримання специфікації ECMA.

3.2 Вибір середовища розробки

3.2.1 Android Studio

Android Studio стала заміною платформи Eclipse. Для розробки використовувався відкритий код IntelliJ IDEA Community Edition, розроблений компанією JetBrains. Середовище розробки ліцензовано Apache 2.0 та розповсюджується у рамках моделі з відкритим кодом.

Android Studio кросс-платформенне середовще розробки, існують версії для усіх популярних операційних систем, таких, як Windows, MacOS та ОС на базі Linux. Середовище розробки надає змогу створювати додатки, як мобільних пристроїв(планшетів, смартфонів) так і для носимих пристроїв, які використовують Android Wear, автомобільних систем, що використовують Android Auto, телевізорів з системою Android TV. Існує спеціальний інструмент для імпортування додатків, що були розроблені у Eclipse.

Середовище розробки має пристосоване для виконання для вирішення усіх завдань, які виникають у процесі розробки додатків на базі Android. Існують інтегровані засоби для тестування програм на сумісність з різними версіями Android, засіб для проектування додатків, який

допомагає адаптувати дизайн додатків для різних розмірів екранів пристроїв, таких як телефони, ноутбуки, планшети, автомобільні системи. Унікальними можливостями Android Studio є уніфікована підсистема складання, розгортання застосунків, тестування, які засновані на інструменті Grandle та підтримка засобів безперервної інтеграції.

Android Studio включає в себе базу типізованих елементів інтерфейсу, та спеціальний візуальний редактор, що допомагає прискоренню розробки додатка, завдяки тому що розробник може заздалегідь мати уяву, як елементи будуть розміщені на різних версіях операційної системи Android та на екранах із різною роздільною здатністю. Якщо розробнику потрібно створити власний інтерфейс, не схожий на шаблони, які надає середовище, вбудовано майстер створення власного дизайну, який включає в себе усі потрібні компоненти. Також розробник може завантажити проект з типових прикладів з GitHub через вбудовану функцію імпорту.

Середовище розробки включає в себе спеціальний інструмент для рефакторінгу коду, перевірки коду на сумісність коду додатку з попередніми випусками, перевірки продуктивності додатка, виявлення споживання ресурсів та оцінювання інтерфейсу користувача. Редактор коду підсвічує помилки у написанні коду, виявлення помилок підтримки Android API. Вбудовано генерація цифрових підписів. Інтегровано інтерфейс для перекладанням на інші мови.

До особливостей Android Studio можна віднести

Живі макети (layout) які можна змінювати у реальному часі.

Редактор коду у реальному часі підсвічує синтаксис мови програмування.

Вбудована консоль, яка надає підказки для оптимізації, допомагає з інтеграцію гугл сервісі та метрик гугл аналітики.

Зберігання бета версій та допомога з порівнянням релізів.

Android-орієнтований рефакторинг та швидкі виправлення.

Засоби для стеження за сумісністю версій, продуктивністю та юзабіліті.

Дизайн шаблони з популярними і оптимізованими рішеннями.

Редактор макетів, який має зручний drag&drop інтерфейс, і можливість попереднього перегляду інтерфейсу при різному розмірі екрану.

3.2.2 Xamarin Studio

Xamarin Studio - кроссплатформенная IDE, яка працює як на Mac OS X, так і на Windows. На вигляд ця вона виглядає дуже простий і доброзичливою, проте за зовнішньою простий ховається, але це середовище є потмужним інструментом, який включає в себе багато функціональних можливостей своїх конкурентів:

- підсвічує синтаксис мови програмування;
- автоматичне доповнення коду, а саме назв методів, класів, атрибутів і т.д.;
- пошук у коді, між файлами, назвами методів, атрибутів;
- розвинена навігація, переход до описання класів, від підкласів до базових і т.і.;
- зручні механізми для рефекторінгу і підказки;
- спеціальні розвинені функції для дебагу та стеження;
- вбудовані можливості для зв'язку з системи для контролю версій з Github, Bitbucket та іншими;

3.2.3 Microsoft Visual Studio

Інтегроване середовище розробки Visual Studio - це оригінальна середу запуску, яка дозволяє редагувати, налагоджувати і створювати код, а потім публікувати додатки. Інтегроване середовище розробки (IDE) - це багатофункціональна програма, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE, Visual Studio включає в себе компілятори, засоби виконання коду, графічні конструктори і багато інших функцій для спрощення процесу розробки програмного забезпечення.

Такі функції, як допомога редактора у написанні коду, підсвічування синтаксису, підказки та «розумне» доповнення коду. Термінал разом з повідомленням про помилку підказує можливе рішення.

Це середовище найбільш підходить для розробників .Net, але за допомогою плагінів можна встановити підтримку, майже всіх мов програмування.

3.3 Обґрунтування вибору середовища програмування та мови програмування

Як середовище розробки Android додатка був обраний програмний продукт Android Studio. Тому що має нативну підтримку Java, у цьому середовищі є вбудований емулятор, завдяки цьому можна не створювати кожен раз .apk файл, а перевіряти роботу додатку прямо у Android Studio.

Мовою програмування обрано Java, ця мова має велику кількість бібліотек і добре підтримується обраним середовищем розробки.

3.4 Реалізація

Для розпізнавання тексту буде використано Text recognition API, що входить до Google ML Kit. Цей API розпізнає латинські символи у реальному часі. Сегментує текст на окремі блоки, рядки та групи слів (рис. 3.1)

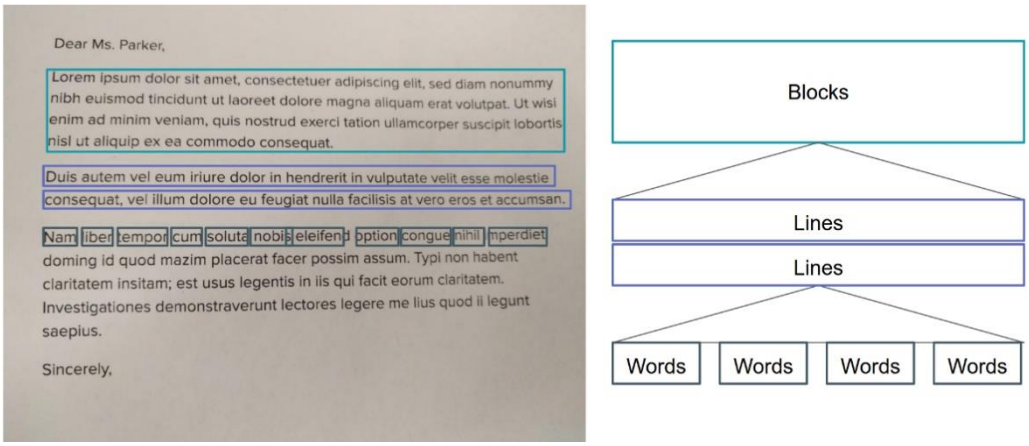

Рисунок 3.1 – Текст із сегментованими блоками

Далі у Android Studio додаємо елементи інтерфейсу користувача. Xml файл з розміткою має наступний вигляд:

```
      <?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/topLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">
<com.google.android.gms.vision.ocrreader.ui.camera.CameraSourcePreview
        android:id="@+id/preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
<com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay
            android:id="@+id/graphicOverlay"
            android:layout_width="match_parent"
            android:layout_height="match_parent"/>
</com.google.android.gms.vision.ocrreader.ui.camera.CameraSourcePreview>

</LinearLayout>
```

Спочатку потрібно додати залежності до файлу build.gradle :

```
      dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.google.android.gms:play-services-vision:17.0.2'
    implementation 'com.android.support:support-v4:27.1.1'
    implementation 'com.android.support:design:27.1.1'



}
```

Змінюємо файл string.xml, який відповідає за текстові змінні у додатку:

```
      <resources>
    <string name="ok">OK</string>
    <string name="permission_camera_rationale">Для виявлення потрібен доступ
до камери</string>
    <string name="no_camera_permission">Ця програма не може запускатися,
оскільки не має дозволу камери. Тепер програма закриється.</string>
    <string name="low_storage_error">Недостатньо місця у сховиші</string>
    <string name="title_activity_main">Розпізнавач</string>
    <string name="ocr_header">Натисність "Розпізнавати"</string>
    <string name="read_text">Розпізнати текст</string>
    <string name="auto_focus">Авто фокус</string>
    <string name="use_flash">Використовуйте Ліхтарик</string>
    <string name="ocr_success">Текст успішно прочитано</string>
    <string name="ocr_failure">Тектс не знайдено</string>
    <string name="ocr_error">"Помилка зчитування: %1$s"</string>
</resources>
```

Android Studio надає змогу не збираючи .apk файл дізнатися, як виглядає додаток на даному етапі (рис 3.2).

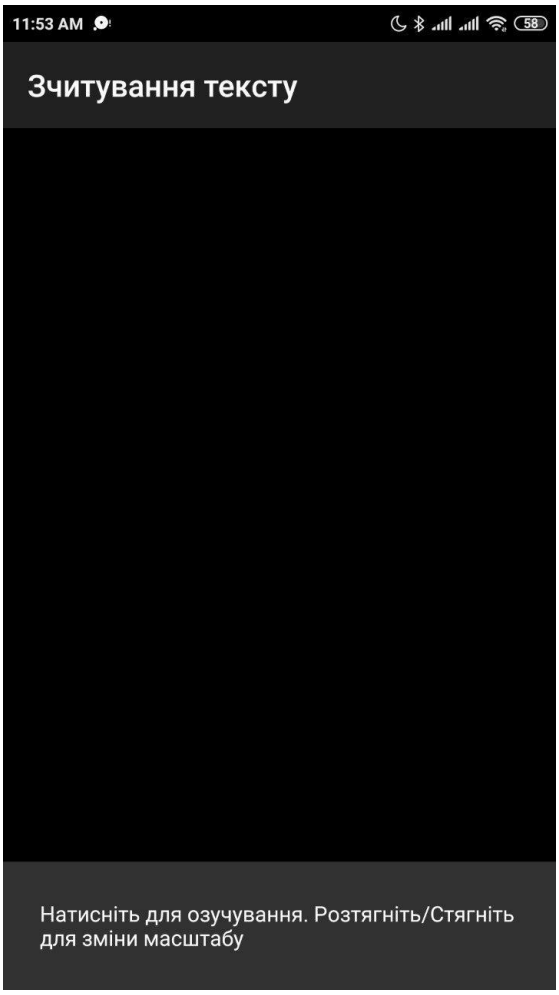

Рисунок 3.2 - вигляд інтерфейсу користувача після додавання графічних елементів.

Створюємо ReaderActivity.java у якому буде описано весь функціонал додатку, у який спочатку імпортуємо

```
      import android.Manifest;
import android.annotation.SuppressLint;
```

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.vision.ocrreader.ui.camera.CameraSource;
import com.google.android.gms.vision.ocrreader.ui.camera.CameraSourcePreview;
import com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.text.TextBlock;
import com.google.android.gms.vision.text.TextRecognizer;

import java.io.IOException;
import java.util.Locale;
```

Далі створюємо файл RecGraphic.java у нього імпортуємо:

```
      import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;

import com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.text.Text;
import com.google.android.gms.vision.text.TextBlock;

import java.util.List;
```

Далі створюємо файл RecognationProcessor.java у цьому файлі описується процесор у якому описується знаходження елементів:

```
import android.util.Log;
import android.util.SparseArray;

import com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;
```

Наступним етапом у файлі ReaderActivity.java створюємо метод Camera, який має параметри autoFocus, useFlash:

```
      private void CameraForRec (boolean CameraFocus, boolean CameraFlash) {
      Context info = getApplicationContext();}
```

Далі потрібно створити об’єкт-детектор Recognizer до методу Camera:

```
private void CameraForRec (Boolean CameraFocus, boolean useFlash) {
Context info = getApplicationContext();
Recognizer textRecognizer = new TextRecognizer.Builder(info).build();}
```

Додаємо перевірку Recognizer, чи достатньо пам’яті та чи доступні залежності Ocr:

```
private void CameraForRec (Boolean CameraFocus, boolean CameraFlash) {
Context info = getApplicationContext();
Recognizer textRecognizer = new TextRecognizer.Builder(info).build();
if (!Recognizer.isOperational()) {
Log.w(TAG “Залежності не задані”);
#перевіка чи достатньо пам’яті
IntentFilter Filterlowstorage = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
                boolean hasLowStorage = registerReceiver(null,
Filterlowstorage) != null;

                if (hasLowStorage)
                {
                    Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
                    Log.w(TAG, getString(R.string.low_storage_error));

}}
```

Наступним пунктом додає можливість розпізнавання у реальному часі. CameraSource додаємо та всередині нього задаємо, яка камера буде активна, роздільну здатність камери, кількість кадрів у секунду, використання «спалаху» та автофокусу:

```
ReccameraSource =
                    new CameraS.Builder(getApplicationContext(),
textRecognizer)
                    .setFacing(CameraS.CAMERA_FACING_BACK)
                    .setRequestedPreviewSize(1920, 1080)
                    .setRequestedFps(2.0f)
                    .setFlashMode(CameraFlash ?
Camera.Parameters.FLASH_MODE_TORCH : null)
                    .setFocusMode(CameraFocus ?
Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO : null)
                    .build();
```

Створюємо Processor, що буде обробляти текст з камери у реальному часі. Створюємо файл RecognationProcessor.java, у ньому створюємо клас RecognationProcessor:

```
public class RecognationProcessor implements
Detector.Processor<TextBlock> {
private GraphicOverlay<RecGraphic> graphicOverlay;
   RecDetectorProcessor(GraphicOverlay<OcrGraphic> recGraphicOverlay) {
       graphicOverlay = recGraphicOverlay;
   }
```

Далі потрібно визначити два методи receiveDetections, release. Перший отримує TextBlock від Recognizer, другий використовується для очищення ресурсів Recognizer:

```
@Override

public void receiveDetections(Detector.Detections<TextBlock>
detections) {
```

```
            graphicOverlay.clear();
            SparseArray<TextBlock> items = detections.getDetectedItems();
            for (int i = 0; i < items.size(); ++i) {
                TextBlock item = items.valueAt(i);
                if (item != null && item.getValue() != null) {
                    Log.d("Процесор", "Текст розпізнанно! " +
item.getValue());
                    RecGraphic graphic = new RecGraphic(graphicOverlay,
item);
                    graphicOverlay.add(graphic);
                }
            }
        }

      @Override
         public void release() {
            graphicOverlay.clear();
         }}
```

Після налаштування процесора додаємо наступний код до файлу ReaderActivity.java *:*

```
      /**
 * Створюємо і запускаємо камеру
 */
@SuppressLint("InlinedApi")
private void createCameraSourceRec(boolean CameraFocus, boolean CameraFlash)
{
    Context info = getApplicationContext();

            TextRecognizer textRecognizer = new
TextRecognizer.Builder(info).build();
    textRecognizer.setProcessor(new RecDetectorProcessor (graphicOverlay));

    if (!textRecognizer.isOperational()) {

        Log.w(TAG, "Detector dependencies are not yet available.");


        IntentFilter Filterlowstorage = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
        boolean hasLowStorage = registerReceiver(null, Filterlowstorage) !=
null;

        if (hasLowStorage) {
            Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
            Log.w(TAG, getString(R.string.low_storage_error));
        }
    }
```

Переходимо у початок ReaderActivity.java та ініціалізуємо запуск камери одразу підчас запуску додатка:

```
       @Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.ocr_capture);

    preview = (CameraSourcePreview) findViewById(R.id.preview);
    graphicOverlay = (GraphicOverlay<OcrGraphic>)
findViewById(R.id.graphicOverlay);
```

```
    boolean CameraFocus = true;
    boolean CameraFlash = false;


    int rc = ActivityCompat.SelfPermissioncheck(this,
Manifest.permission.CAMERA);
    if (rc == PackageManager.PERMISSION_GRANTED) {
        createCameraSourceRec(CameraFocus, CameraFlash);
    } else {
        CameraPermission();
    }

    gestureDetector = new GestureD (this, new CaptureGestureListener());
    scaleGestureDetector = new ScaleGestureD (this, new ListenerOfScale ());

    Snackbar.make(graphicOverlay, "Натисніть для озучування.
Розтягніть/Стягніть для зміни масштабу",
            Snackbar.LENGTH_LONG)
            .show();

    TextToSpeech.OnInitListener listener =
            new TextToSpeech.OnInitListener()
      {
                @Override

                 public void onInit(final int status) {
                    if (status == TextToSpeech.SUCCESS) {
                        Log.d("OnInitListener", "Механізм з текстовим
мовленням запустився успішно.");
                        tts.setLanguage(Locale.US);
                    } else {
                        Log.d("OnInitListener", "Помилка запуску тексту до
мовного мовлення.");
                    }
                }
            };
    tts = new TextToSpeech(this.getApplicationContext(), listener);
}
```

Додаємо метод, що надає запит на дозвіл доступу до камери(рис.3.4.3):

```
      private void CameraPermission() {
             Log.w(TAG, "Дозвіл на камеру не надається. Запит дозволу");

             final String[] permissions = new
String[]{Manifest.permission.CAMERA};

             if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
                  Manifest.permission.CAMERA)) {
                 ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
                 return;
             }

             final Activity thisActivity = this;

             View.OnClickListener listener = new View.OnClickListener()
      {
                 @Override

                 public void onClick(View view) {
```

```
                    ActivityCompat.requestPermissions(thisActivity,
permissions,
                            RC_HANDLE_CAMERA_PERM);
                }
            };
            Snackbar.make(graphicOverlay,
R.string.permission_camera_rationale,
                    Snackbar.LENGTH_INDEFINITE)
                    .setAction(R.string.ok, listener)
                    .show();
        }
```

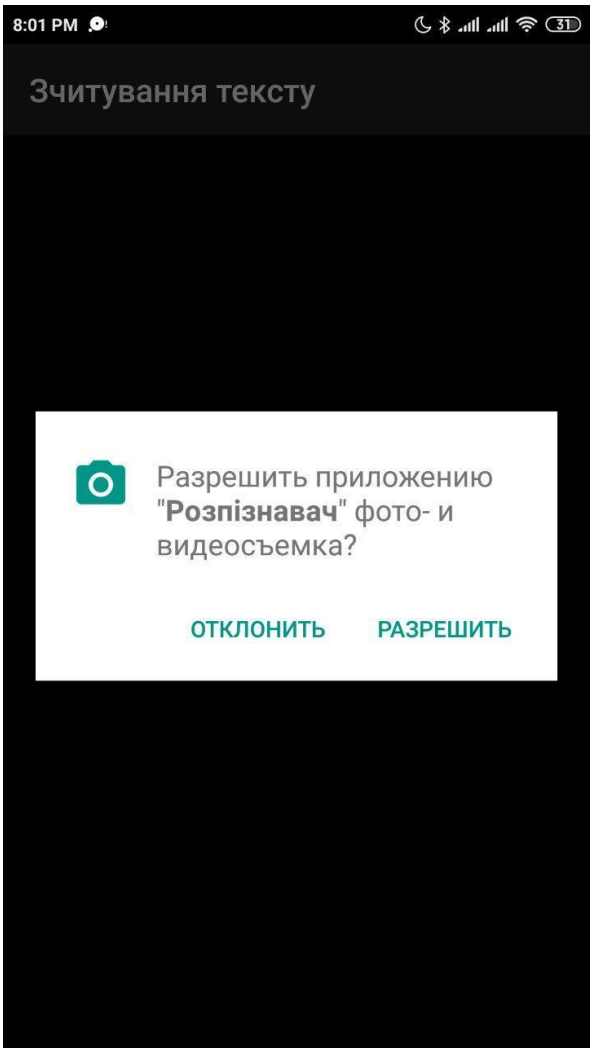

Рисунок 3.3 – Запит додатка на дозвіл доступу до камери

Додаємо перевірку чи отримані усі дозволи для роботи додатка, для цього створюємо метод onRequestPermissionsResult:

```
    @Override

    public void RequestForPermissionsResult(int requestingCode,
                                    @NonNull String[] permissions,
                                    @NonNull int[]Resultsgrant) {
    if (requestingCode != RC_HANDLE_CAMERA_PERM) {
        Log.d(TAG, "Неочікуваний результат на запит дозволу" +
requestingCode);
        super.RequestForPermissionsResult(requestingCode, permissions,
```

```
grantResults);
        return;
    }

    if (Resultsgrant.length != 0 && Resultsgrant [0] ==
PackageManager.PERMISSION_GRANTED) {
        Log.d(TAG, "Дозвіл на використання камери отримано - ініціалізувати
джерело камери");

        boolean CameraFocus = getIntent().getBooleanExtra(CameraFocus,true);
        boolean CameraFlash = getIntent().getBooleanExtra(CameraFlash,
false);
        createCameraSourceRec(CameraFocus, CameraFlash);
        return;
    }

    Log.e(TAG, "Дозівл не отримано, код помилки:" + grantResults.length +
            " Код помилки = " + (grantResults.length > 0 ? grantResults[0] :
"(empty)"));

    DialogInterface.OnClickListener listener = new
DialogInterface.OnClickListener()
      {
        public void Click(DialogInterface dialog, int id)
       {
            finish();
        }
    };

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Multitracker sample")
            .setMessage(R.string.no_camera_permission)
            .setPositiveButton(R.string.ok, listener)
            .show();
}
```

Додаємо метод, який буде викликати запуск камери та одночасно перевіряти чи доступні гугл сервіси, якщо не можливо запусти камеру, метод записує повідомлення про помилку з її кодом до лог файлу:

```
      private void startCameraSource() throws SecurityException {
    // перевірка роботоспособності сервісів гугл
    int code =
GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
            getApplicationContext());
    if (code != ConnectionResult.SUCCESS) {
        Dialog dlg =
                GoogleApiAvailability.getInstance().getErrorDialog(this,
code, RC_HANDLE_GMS);
        dlg.show();
    }

    if (cameraSource != null) {
        try {
            preview.start(cameraSource, graphicOverlay);
        } catch (IOException e) {
            Log.e(TAG, "Не можливо запустити камеру", e);
            cameraSource.release();
            cameraSource = null;
        }
```

```
    }
}
```

Описуємо клас RecGraphic, який наслідує OcrGraphicOverlay.Graphic, у ньому описується методи які визначають виділення розпізнаного тексту яким буду виділено розпізнаний текст:

```
public class RecGraphic extends GraphicOverlay.Graphic {

private int id;

private static final int TEXT_COLOR = Color.Blue;

private static Paint rectPaint;
private static Paint textPaint;
private final TextBlock textBlock;

RecGraphic(GraphicOverlay overlay, TextBlock text) {
    super(overlay);

    textBlock = text;

    if (rectPaint == null) {
        rectPaint = new Paint();
        rectPaint.setColor(TEXT_COLOR);
        rectPaint.setStyle(Paint.Style.STROKE);
        rectPaint.setStrokeWidth(4.0f);
    }

    if (textPaint == null) {
        textPaint = new Paint();
        textPaint.setColor(TEXT_COLOR);
        textPaint.setTextSize(54.0f);
    }

    postInvalidate();
}
```

Тепер додаємо виділення тексту на екрані. Додаємо у файл RecGraphic.java метод draw:

```
@Override

public void drawing(Canvas canvas) {
if (textBlock == null) {
    return;
}


RectF rect = new RectF(textBlock.getBoundingBox());
rect = translateRect(rect);
canvas.drawingRect(rect, rectPaint);
```

На даний момент у весь текст розпізнається, як суцільний блок, додаємо розпізнавання окремих рядків:

```
@Override

public void drawing(Canvas canvas) {
    if (textBlock == null) {
   return;
}
```

```
        RectF rect = new RectF(textBlock.getBoundingBox());
        rect = translateRect(rect);
        canvas.drawingRect(rect, rectPaint);

    //Розпізнавання окремих рядків
        List<? extends Text> textComponents = textBlock.getComponents();
        for(Text currentText : textComponents) {
            float left = translateX(currentText.getBoundingBox().left);
            float bottom = translateY(currentText.getBoundingBox().bottom);
            canvas.drawingText(currentText.getValue(), left, bottom, textPaint);
        }
    }
```

Запускаємо емулятор та бачимо, що камера працює, текст виділяється та показується текст який розпізнаному, у даному випадку це назва банку «monobank Universal Bank» (рис. 3.4).

Рисунок 3.4 – Приклад роботи програми

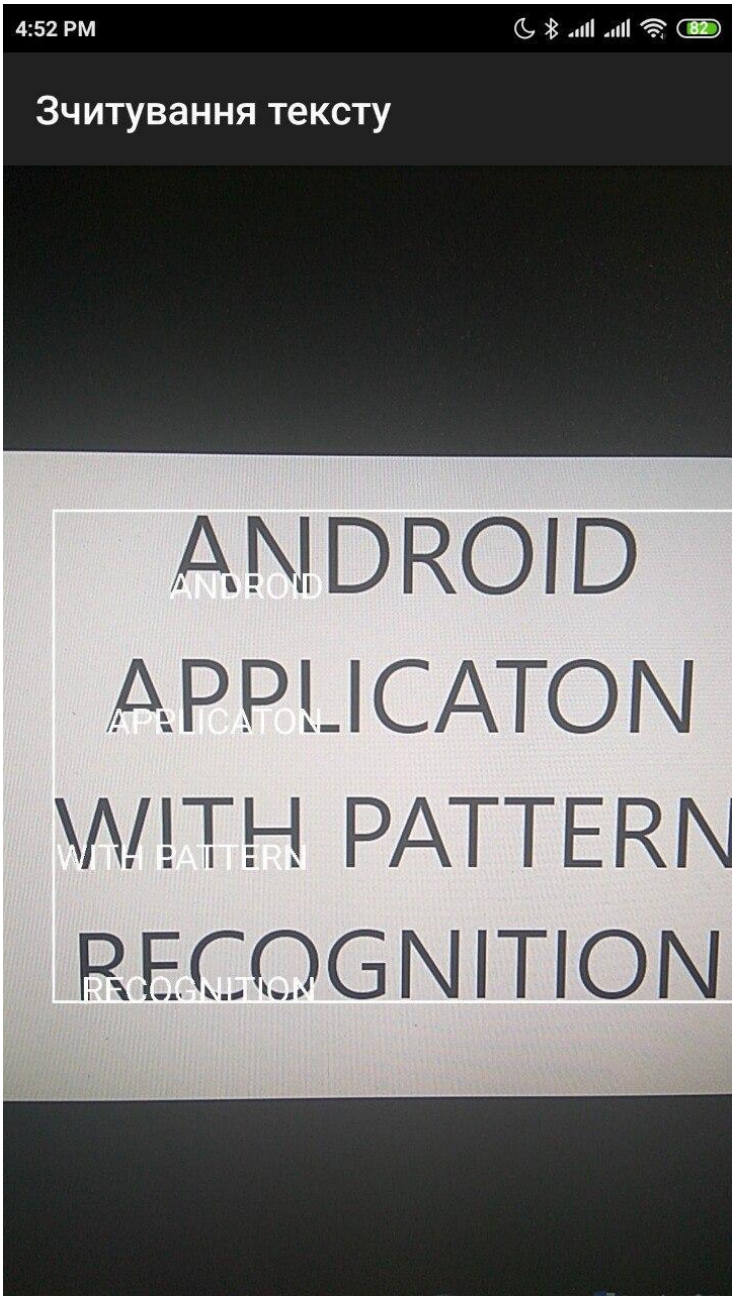

Рисунок 3.5 – Приклад роботи програми

Додаємо можливість аудіо-відтворення текст, який розпізнано, для цього у файл RecGraphic.java додаємо метод contains:

```
  public boolean contains(float x, float y) {
    if (textBlock == null) {
        return false;
    }
    RectF rect = new RectF(textBlock.getBoundingBox());
    rect = translateRect(rect);
    return rect.contains(x, y);
}
```

Додаємо метод onTap у ReaderActivity.java для опрацювання натискання на виділений текст:

```
      private boolean TapDetector(float rawX, float rawY) {
             RecGraphic graphic = graphicOverlay.getGraphicAtLocation(rawX,
rawY);
             BlockText text = null;
             if (graphic != null) {
                 text = graphic.getBlockText ();
                 if (text != null && text.getValue() != null) {
                     Log.d(TAG, "Текст було проговоренно! " +
text.getValue());

                     tts.speak(text.getValue(), TextToSpeech.QUEUE_ADD, null,
"DEFAULT");
                 }
                 else {
                     Log.d(TAG, "text data is null");
                 }
             }
             else {
                 Log.d(TAG,"Текст не знайдено");
             }
             return text != null;
         }
```

Текст який буде зчитуватися може знаходитися на різній відстані, то додаємо можливість приближення та віддалення об’єктів, для цього створюємо класс ListenerOfScale, який наслідує за ScaleGestureDetector.OnScaleGestureListener, який ми імпортували на самому початку:

```
      private class ListenerOfScale implements
ScaleGestureDetector.OnScaleGestureListener {


        @Override

         public boolean Scale(ScaleGestureDetector detector)
         {
            return false;
        }


        @Override

           public boolean BeginonScale (ScaleGestureDetector detector)
          {
            return true;
        }


        @Override

         public void EndonScale (ScaleGestureDetector detector)
         {
            if (cameraSource != null) {
                cameraSource.doZoom(detector.getScaleFactor());
            }
        }
    }
}
```

Наприкінці нам потрібно додати методи, що будуть призупиняти, відновлювати та вимикати камеру, це потрібно, щоб у коли ви виходите, або згортаєте додаток переставав використовувати камеру, та відновлював її роботу коли повертаєтесь до додатка

```
      @Override
protected void onResume() {
    super.onResume();
    startCameraSource();
}


@Override
protected void onPause() {
    super.onPause();
    if (preview != null) {
        preview.stop();
    }
}


@Override
protected void onDestroy() {
    super.onDestroy();
    if (preview != null) {
        preview.release();
    }
}
```

3.5 Висновок до розділу 3

Як середовище розробки Android додатка було обрано Android Studio,тому що вона має вбудовану підтримку Java, у цьому середовищі є вбудований емулятор, завдяки якому можна не створювати кожен раз .apk файл, а перевіряти роботу додатку прямо у Android Studio. Мовою програмування обрано Java, ця мова має велику кількість бібліотек і добре підтримується обраним середовищем розробки.

Розроблено Android додаток, що розпізнає текст у реальному часі та має функціональну можливість аудио-відтворювати текст, який було розпізнано.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» [26] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

4.2 Аналіз стану умов праці

Робота над створенням інформаційного ресурсу проходитиме в приміщенні відповідної установи. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

Геометричні розміри приміщення зазначені в табл. 4.1. та відповідають нормам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [27].

Таблиця 4.1 – Розміри приміщення

Найменування	Значення
Довжина, м	5
Ширина, м	3,5
Висота, м	2,5
Площа, $м^2$	17,5
Об’єм, $м^3$	43,75

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [28] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷ 800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷ 500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷ 800

4.2.1 Навантаження та напруженість процесу праці

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви (потрібно вибрати):

- для розробників програм тривалість 15 хв через кожну годину роботи;
- для операторів персональних комп’ютерів тривалістю 15 хв через дві години роботи;
- для операторів комп’ютерного набору тривалістю 10 хв через кожну годину роботи.

4.3 Виробнича санітарія

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00.-7.15-18 «Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями» [34].

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ	2	[27]
- підвищена або знижена вологість повітря	-//-	2	[27]
- підвищений рівень електромагнітного випромінення	-//-	2	[33]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[33] [34]
- підвищений рівень статичної електрики	-//-	2	[32]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[30]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[34] [28]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці-сидіння користувача,) та організації робочого часу - безперервна робота)	2	[34] [28]

4.3.1 Пожежна безпека

Для гасіння пожеж в квартирі пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними. Горючими матеріалами в приміщенні, де розташовані ЕОМ, є: поліаміди, полівінілхлорид, ізоляційний матеріал, пластикат кабельний, деревина.

Згідно НАПБ А. 01.001-2014 «Правила пожежної безпеки в Україні» [29] таке приміщення, площею 17,5 м^2, відноситься до категорії "В" (пожежонебезпечної). Відповідно до норм первинних засобів пожежогасінні пропонується використовувати: повсть 1х1 м^2 , кошму 2х1,5м^2 в кількості 1 шт.

4.3.2 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Параметри мікроклімату

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [27] і наведені в табл. 4.4:

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

4.4.2 Освітлення

У проекті, що розробляється, передбачається використовувати суміщене освітлення, ДБН В.2.5-28:2018 «Природне і штучне освітлення» [30]. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 3{,}5 = 17{,}5$ м^2,

$S = 1/8 \cdot 17{,}5 = 2{,}1875$ м^2.

Приймаємо 2 вікна площею S=1,1 м^2 кожне.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \tag{4.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 17,5 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \cdot 17,5 \cdot 1,1 \cdot 1,5}{5400 \cdot 0,575 \cdot 2} \approx 1$$

Приймаємо освітлювальну установку, яка складається з 1-го світильника, який складається з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4.3 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при V приміщення > 40 м^3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом НПАОП 40.1-1.01-97 «Правила безпечної експлуатації електроустановок» [31], НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» [35] приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 220 В. Опір контура заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлювача η – це відношення діючої провідності цього заземлювача до найбільш можливої його провідності за нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача η_c.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \tag{4.3}$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (рис.4.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \tag{4.4}$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м}$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (4.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (4.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2– 3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03– 0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (4.6):

$$t = h_в + \frac{l_в}{2}, \quad (4.6)$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0,8 + \frac{3}{2} = 2,3$ м

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.7)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.8)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_с = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{ м}$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_с} \cdot \ln \frac{2 \cdot l_с^2}{d_{см} \cdot h_г}, \quad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизонтальних заземлювачів (0,5 м);

$l_с$ - довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом}$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги η_c=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_c + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

4.6 Охорона навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: робота за компьютером в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на відалення (знешкодження), утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності комп'ютера виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

Відпрацьовані люмінесцентні лампи - I клас небезпеки

Батарейки та акумулятори (малі) -III клас небезпеки

Змінні носії інформації - IV клас небезпеки

Відпрацьований ізолюючий матеріал, дроти та кабелі - IV клас небезпеки

Макулатура - IV клас небезпеки

Побутові відходи - IV клас небезпеки

Висновок до розділу 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника. А також визначені основні екологічні аспекти впливу на навколишнє природне середовище та зазначені поводження з ними.

ВИСНОВОК

Метою магістерської роботи було проведення аналізу засобів що допомагають у створенні додатків для розпізнавання образів, а також створення додатка, що розпізнає текст і аудіо відтворює его користувачу.

У першому розділі було проведено аналіз предметної області розпізнавання образів, проаналізовано підходу до розпізнавання тексту. Виділені такі підходи, як шаблонний, структурний, ознаковий та метод з використанням штучного інтелекту. А також були проаналізовані існуючі додатки. В результаті стало зрозуміло, що розпізнавання образів дуже важливе завдання, тому що може бути використано у будьякій галузі. Підходом до розпізнавання було обрано машинне навчання, як найкращий варіант для вирішення поставленого завдання. У існуючих додатків були знайдені недоліки, які розробленний додаток має виправити.

У другому розділі проводиться аналіз методу з використанням машинного навчання. Вибір пав на Google ML Kit, який має великий спектр можливостей у сфері розпізнавання образів. Завдяки Google ML Kit можна розпізнавати текст, об’єкти на зображеннях, штрих-коди, створювати системи для розпізнавання обличь та багато іншого, Google ML Kit надає змогу використовувати як уже навчені моделі компанією Google, так і власні. Також підчас проведеня було створено схему роботи додатка та схему яким чином буде проводитися розпізнавання тексту.

У третьому розділі проаналізовані середовища для розробки Android додатків та зроблено вибір мови програмування додатку. Мовою програмування обрано Java, ця мова має велику кількість бібліотек. Як середовище розробки було обрано Android Studio. Тому що вона підтримує моу Java, у цьому середовищі є вбудований емулятор, завдяки якому можна не створювати кожен раз .apk файл, а перевіряти роботу додатку прямо у Android Studio. Розроблено Android додаток, що розпізнає текст у реальному часі та має функціональну можливість аудио-відтворювати текст, який було розпізнано.

У четвертому розділі зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Мета і завдання були повністю виконані. Додаток швидко друкований розпізнає текст та відтворює его користувачу. Надалі, можна збільшувати функціонал додатка, зберігати текст який було розпізано, додати переклад розпізнаного тексту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рудий І.В. Аналіз методів розпізнавання образів на базі операційної системи Android // Збірник науково-практичних праць V молодіжного форуму «ІТ-ідея 2019 С. 99-100. Режим доступу: http://idea.turion.info/asset/proceedings/IT_IDEA_2019_proceedings.pdf -
2. Рудий І.В. Програмні засоби розпізнавання образів на мобільних платформах // Майбутній науковець – 2019 : матеріали всеукр. наук.-практ. конф. з міжнар. участю Ч. І. С- 175-177. Режим доступ: http://dspace.snu.edu.ua:8080/jspui/handle/123456789/2998
3. Optical character recognition, 2019 Режим доступу: https://en.wikipedia.org/wiki/Optical_character_recognition
4. Han J., Kamber M., Pei J. Data mining Concepts and techniques. – Morgan kaufmann, 2006.
5. Mitchell T. Generative and discriminative classifiers: naive Bayes and logistic regression, 2005 Режим доступу: https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf
6. McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity //The bulletin of mathematical biophysics. – 1943. – Т. 5. – №. 4. – С. 115-133.
7. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain //Psychological review. – 1958. – Т. 65. – №. 6. – С. 386.
8. Minsky M., Seymour P. Perceptrons. – 1969.
9. Bengio Y., Goodfellow I., Courville A. Deep Learning. – MIT Press, book in preparation
10. Cireşan D. C. et al. Deep big multilayer perceptrons for digit recognition /Neural Networks: Tricks of the Trade. – Springer Berlin Heidelberg, 2012. – С. 581-598.
11. Baldi P. Autoencoders, Unsupervised Learning, and Deep Architectures // ICML Unsupervised and Transfer Learning. – 2012. – Т. 27. – С. 37-50.
12. LeCun Y., Bengio Y. Convolutional networks for images, speech, and time series // The handbook of brain theory and neural networks. – 1995. – Т. 3361. – С. 310.
13. Abby FineReader Offical website Режим доступу: https://finereaderonline.com
14. ML Kit for Firebase Режим доступу: https://firebase.google.com/docs/ml-kit/
15. TensorFlow Documentation Режим доступу: https://www.tensorflow.org/api_docs
16. Neural Network Documentation Режим доступу: https://developer.android.com/ndk/guides/neuralnetworks
17. Neural Network API Drivers Режим доступу: https://source.android.com/devices/neural-networks
18. Майер Р.В. Компьютерное моделирование: Учебник для педвузов – С. 23-27. Режим доступу: https://sites.google.com/site/mayerrv/comp_mod

19. Советов Б.Я., Яковлєв С.А. Моделювання систем: Підруч, для вузів – М.: Виш.Шк., 2001. – 343 с.

20. Кунин С. Вычислительная физика. — М.: Мир, 1992. — 518 с.

21. Chawki Djedddi, Akhtar Jamil, Imran Siddiqi, Pattern Recognition and Artificial Intelligence, Режим доступу: https://www.researchgate.net/publication/337874546_Pattern_Recognition_and_Artificial_Intelligence

22. Shafait F. Perfomance Comparison of Six Algorithms for Page Segmentation / F. Shafit, D. Keysers, T. Breuel / Image Understanding and Pattern Recognition (IUPR) research group. – 2006. С. 12.

23. В. В. Жихаревич, С. Е. Остапов, І. В. Миронів Аналіз методів розпізнавання символів тексту / Радіоелектронні і комп'ютерні системи. - 2016. - № 5. - С. 137–142. - Режим доступу: http://nbuv.gov.ua/UJRN/recs_2016_5_23.

24. Кирсанов, М.Н. Модификация и анализ фильтров выделения контуров изображений [Текст] / Вестник государственногоуниверситета морского и речного флота им. адмирала С.О. Макарова. – 2015г – том 5. – № 33. – С. 201-206.

25. Касьян К.Н., Братчиков В.В., Шкарупило В.В. Розробка модифікованого алгортиму розпізнавання тексту на стандартизованому зображені Режим доступу: http://ena.lp.edu.ua/bitstream/ntb/38377/1/93_196-197.pdf

26. Закон України «Про охорону праці». Вводиться в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669. – Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/2694-12

27. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Вводиться в дію Постановою ВР № 42 від 01.12.1999. – Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

28. ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Вводиться в дію Постановою ВР № 7 від 10.12.1998. – Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

29. НАПБ А. 01.001-2014 «Правила пожежної безпеки в Україні». Затверджено Наказом Міністерства внутрішніх справ України № 1417 від 30.12.2014. – Режим доступу: www. URL: https://zakon4.rada.gov.ua/laws/show/z0252-15

30. ДБН В.2.5-28:2018 «Природне і штучне освітлення». – Режим доступу: www. URL: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf

31. НПАОП 40.1-1.01-97 «Правила безпечної експлуатації електроустановок». Затверджено наказом Державного комітету України по нагляду за охороною праці № 257 від 6 жовтня 1997 р. – Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0011-98

32. ДСТУ 7237:2011 «Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту». Затверджено Держспоживстандартом України № 37 від 02.02.2011. – Режим доступу: www. URL: http://online.budstandart.com/ru/catalog/doc-page.html?id_doc=30045

33. ГОСТ 13109-97 «Електрична енергія. Сумістність технічних засобів електромагнітних». Дата введення 01.01.1999. – Режим доступу: www. URL: https://dnaop.com/html/42313/doc-ГОСТ_13109-97

34. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями». Зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за № 508/31960. – Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0508-18

35. НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів». Затверджено Наказом Держнаглядохоронприці України № 4 від 09.01.98 – Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0093-98

ДОДАТОК А. Лістинг програми

```
package com.google.android.gms.vision.ocrreader;

import android.util.Log;
import android.util.SparseArray;

import com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;


public class RecDetectorProcessor implements Detector.Processor<TextBlock> {

    private GraphicOverlay<OcrGraphic> graphicOverlay;

    OcrDetectorProcessor(GraphicOverlay<OcrGraphic> ocrGraphicOverlay) {
        graphicOverlay = ocrGraphicOverlay;
    }

    @Override
    public void receiveDetections(Detector.Detections<TextBlock> detections)
{
        graphicOverlay.clear();
        SparseArray<TextBlock> items = detections.getDetectedItems();
        for (int i = 0; i < items.size(); ++i) {
            TextBlock item = items.valueAt(i);
            if (item != null && item.getValue() != null) {
                Log.d("OcrDetectorProcessor", "Text detected! " +
item.getValue());
                OcrGraphic graphic = new OcrGraphic(graphicOverlay, item);
                graphicOverlay.add(graphic);
            }
        }
    }

    @Override
    public void release() {
        graphicOverlay.clear();
    }
}


package com.google.android.gms.samples.vision.ocrreader;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
```

```
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.vision.ocrreader.ui.camera.CameraSource;
import com.google.android.gms.vision.ocrreader.ui.camera.CameraSourcePreview;
import com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.text.TextBlock;
import com.google.android.gms.vision.text.TextRecognizer;

import java.io.IOException;
import java.util.Locale;


public final class RecCaptureActivity extends AppCompatActivity {
    private static final String TAG = "OcrCaptureActivity";


    private static final int RC_HANDLE_GMS = 9001;


    private static final int RC_HANDLE_CAMERA_PERM = 2;


    public static final String AutoFocus = "AutoFocus";
    public static final String UseFlash = "UseFlash";
    public static final String TextBlockObject = "String";

    private CameraSource cameraSource;
    private CameraSourcePreview preview;
    private GraphicOverlay<OcrGraphic> graphicOverlay;


    private ScaleGestureDetector scaleGestureDetector;
    private GestureDetector gestureDetector;


    private TextToSpeech tts;


    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.ocr_capture);

        preview = (CameraSourcePreview) findViewById(R.id.preview);
        graphicOverlay = (GraphicOverlay<OcrGraphic>)
findViewById(R.id.graphicOverlay);


        boolean autoFocus = true;
        boolean useFlash = false;


        int rc = ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA);
        if (rc == PackageManager.PERMISSION_GRANTED) {
            createCameraSource(autoFocus, useFlash);
```

```
124             } else {
125                 requestCameraPermission();
126             }
127 
128             gestureDetector = new GestureDetector(this, new
129         CaptureGestureListener());
130             scaleGestureDetector = new ScaleGestureDetector(this, new
131         ScaleListener());
132 
133             Snackbar.make(graphicOverlay, "Натисніть для озучування.
134         Розтягніть/Стягніть для зміни масштабу",
135                     Snackbar.LENGTH_LONG)
136                     .show();
137 
138             // Set up the Text To Speech engine.
139             TextToSpeech.OnInitListener listener =
140                     new TextToSpeech.OnInitListener() {
141                         @Override
142                         public void onInit(final int status) {
143                             if (status == TextToSpeech.SUCCESS) {
144                                 Log.d("OnInitListener", "Механізм з текстовим
145         мовленням запустився успішно.");
146                                 tts.setLanguage(Locale.US);
147                             } else {
148                                 Log.d("OnInitListener", "Помилка запуску тексту
149         до мовного мовлення.");
150                             }
151                         }
152                     };
153             tts = new TextToSpeech(this.getApplicationContext(), listener);
154         }
155 
156 
157         private void requestCameraPermission() {
158             Log.w(TAG, "Дозвіл на камеру не надається. Запит дозволу");
159 
160             final String[] permissions = new
161         String[]{Manifest.permission.CAMERA};
162 
163             if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
164                     Manifest.permission.CAMERA)) {
165                 ActivityCompat.requestPermissions(this, permissions,
166         RC_HANDLE_CAMERA_PERM);
167                 return;
168             }
169 
170             final Activity thisActivity = this;
171 
172             View.OnClickListener listener = new View.OnClickListener() {
173                 @Override
174                 public void onClick(View view) {
175                     ActivityCompat.requestPermissions(thisActivity, permissions,
176                             RC_HANDLE_CAMERA_PERM);
177                 }
178             };
179 
180             Snackbar.make(graphicOverlay, R.string.permission_camera_rationale,
181                     Snackbar.LENGTH_INDEFINITE)
182                     .setAction(R.string.ok, listener)
183                     .show();
184         }
185 
186         @Override
```

```
187             public boolean onTouchEvent(MotionEvent e) {
188                 boolean b = scaleGestureDetector.onTouchEvent(e);
189 
190                 boolean c = gestureDetector.onTouchEvent(e);
191 
192                 return b || c || super.onTouchEvent(e);
193             }
194 
195             /**
196              * Створюємо і запускаємо камеру
197              */
198             @SuppressLint("InlinedApi")
199             private void createCameraSource(boolean autoFocus, boolean useFlash) {
200                 Context context = getApplicationContext();
201 
202                         TextRecognizer textRecognizer = new
203         TextRecognizer.Builder(context).build();
204                 textRecognizer.setProcessor(new
205         RecDetectorProcessor(graphicOverlay));
206 
207                 if (!textRecognizer.isOperational()) {
208 
209                     Log.w(TAG, "Detector dependencies are not yet available.");
210 
211 
212                     IntentFilter lowstorageFilter = new
213         IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
214                     boolean hasLowStorage = registerReceiver(null, lowstorageFilter)
215         != null;
216 
217                     if (hasLowStorage) {
218                         Toast.makeText(this, R.string.low_storage_error,
219         Toast.LENGTH_LONG).show();
220                         Log.w(TAG, getString(R.string.low_storage_error));
221                     }
222                 }
223 
224 
225                 cameraSource =
226                         new CameraSource.Builder(getApplicationContext(),
227         textRecognizer)
228                         .setFacing(CameraSource.CAMERA_FACING_BACK)
229                         .setRequestedPreviewSize(1280, 1024)
230                         .setRequestedFps(2.0f)
231                         .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH :
232         null)
233                         .setFocusMode(autoFocus ?
234         Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO : null)
235                         .build();
236             }
237 
238 
239             @Override
240             protected void onResume() {
241                 super.onResume();
242                 startCameraSource();
243             }
244 
245 
246             @Override
247             protected void onPause() {
248                 super.onPause();
249                 if (preview != null) {
```

```
250                 preview.stop();
251             }
252         }
253
254
255         @Override
256         protected void onDestroy() {
257             super.onDestroy();
258             if (preview != null) {
259                 preview.release();
260             }
261         }
262
263
264         @Override
265         public void onRequestPermissionsResult(int requestCode,
266                                                @NonNull String[] permissions,
267                                                @NonNull int[] grantResults) {
268             if (requestCode != RC_HANDLE_CAMERA_PERM) {
269                 Log.d(TAG, "Неочікуваний результат на запит дозволу" +
270     requestCode);
271                 super.onRequestPermissionsResult(requestCode, permissions,
272     grantResults);
273                 return;
274             }
275
276             if (grantResults.length != 0 && grantResults[0] ==
277     PackageManager.PERMISSION_GRANTED) {
278                 Log.d(TAG, "Дозвіл на використання камери отримано -
279     ініціалізувати джерело камери");
280
281                 boolean autoFocus = getIntent().getBooleanExtra(AutoFocus,true);
282                 boolean useFlash = getIntent().getBooleanExtra(UseFlash, false);
283                 createCameraSource(autoFocus, useFlash);
284                 return;
285             }
286
287             Log.e(TAG, "Дозівл не отримано, код помилки:" + grantResults.length +
288                     " Код помилки = " + (grantResults.length > 0 ?
289     grantResults[0] : "(empty)"));
290
291             DialogInterface.OnClickListener listener = new
292     DialogInterface.OnClickListener() {
293                 public void onClick(DialogInterface dialog, int id) {
294                     finish();
295                 }
296             };
297
298             AlertDialog.Builder builder = new AlertDialog.Builder(this);
299             builder.setTitle("Multitracker sample")
300                     .setMessage(R.string.no_camera_permission)
301                     .setPositiveButton(R.string.ok, listener)
302                     .show();
303         }
304
305
306         private void startCameraSource() throws SecurityException {
307             // перевірка роботоспособності сервісів гугл
308             int code =
309     GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
310                     getApplicationContext());
311             if (code != ConnectionResult.SUCCESS) {
312                 Dialog dlg =
```

```
313                     GoogleApiAvailability.getInstance().getErrorDialog(this,
314         code, RC_HANDLE_GMS);
315                 dlg.show();
316             }
317
318             if (cameraSource != null) {
319                 try {
320                     preview.start(cameraSource, graphicOverlay);
321                 } catch (IOException e) {
322                     Log.e(TAG, "Не можливо запустити камеру", e);
323                     cameraSource.release();
324                     cameraSource = null;
325                 }
326             }
327         }
328
329
330         private boolean onTap(float rawX, float rawY) {
331             OcrGraphic graphic = graphicOverlay.getGraphicAtLocation(rawX, rawY);
332             TextBlock text = null;
333             if (graphic != null) {
334                 text = graphic.getTextBlock();
335                 if (text != null && text.getValue() != null) {
336                     Log.d(TAG, "Текст було проговоренно! " + text.getValue());
337                     // Speak the string.
338                     tts.speak(text.getValue(), TextToSpeech.QUEUE_ADD, null,
339         "DEFAULT");
340                 }
341                 else {
342                     Log.d(TAG, "text data is null");
343                 }
344             }
345             else {
346                 Log.d(TAG,"Текст не знайдено");
347             }
348             return text != null;
349         }
350
351         private class CaptureGestureListener extends
352         GestureDetector.SimpleOnGestureListener {
353
354             @Override
355             public boolean onSingleTapConfirmed(MotionEvent e) {
356                 return onTap(e.getRawX(), e.getRawY()) ||
357         super.onSingleTapConfirmed(e);
358             }
359         }
360
361         private class ScaleListener implements
362         ScaleGestureDetector.OnScaleGestureListener {
363
364
365             @Override
366             public boolean onScale(ScaleGestureDetector detector) {
367                 return false;
368             }
369
370
371             @Override
372             public boolean onScaleBegin(ScaleGestureDetector detector) {
373                 return true;
374             }
375
```

```
            @Override
            public void onScaleEnd(ScaleGestureDetector detector) {
                if (cameraSource != null) {
                    cameraSource.doZoom(detector.getScaleFactor());
                }
            }
        }
    }
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.gms.vision.ocrreader"
    android:installLocation="auto" >

    <uses-feature android:name="android.hardware.camera" />

    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:fullBackupContent="false"
        android:hardwareAccelerated="true"
        android:icon="@drawable/icon"
        android:label="Розпізнавач"
        android:supportsRtl = "true"
        android:theme="@style/Theme.AppCompat" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.gms.vision.DEPENDENCIES"
            android:value="ocr" />

        <activity
android:name="com.google.android.gms.samples.vision.ocrreader.RecCaptureActi
vity"
            android:label="Зчитування тексту">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

```
<resources>
    <string name="ok">OK</string>
    <string name="permission_camera_rationale">Для виявлення потрібен доступ
до камери</string>
    <string name="no_camera_permission">Ця програма не може запускатися,
оскільки не має дозволу камери. Тепер програма закриється.</string>
    <string name="low_storage_error">Недостатньо місця у сховиші</string>
    <string name="title_activity_main">Розпізнавач</string>
    <string name="ocr_header">Натисність "Розпізнавати"</string>
    <string name="read_text">Розпізнати текст</string>
    <string name="auto_focus">Авто фокус</string>
    <string name="use_flash">Використовуйте Ліхтарик</string>
    <string name="ocr_success">Текст успішно прочитано</string>
```

```
437             <string name="ocr_failure">Тектс не знайдено</string>
438             <string name="ocr_error">"Помилка зчитування: %1$s"</string>
439         </resources>
440
441         <?xml version="1.0" encoding="utf-8"?>
442
443         <LinearLayout
444             xmlns:android="http://schemas.android.com/apk/res/android"
445             android:id="@+id/topLayout"
446             android:layout_width="match_parent"
447             android:layout_height="match_parent"
448             android:keepScreenOn="true">
449
450             <com.google.android.gms.vision.ocrreader.ui.camera.CameraSourcePreview
451                 android:id="@+id/preview"
452                 android:layout_width="match_parent"
453                 android:layout_height="match_parent">
454
455                 <com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay
456                     android:id="@+id/graphicOverlay"
457                     android:layout_width="match_parent"
458                     android:layout_height="match_parent"/>
459
460             </com.google.android.gms.vision.ocrreader.ui.camera.CameraSourcePreview>
461
462         </LinearLayout>
463
464         package com.google.android.gms.vision.ocrreader;
465
466         import android.graphics.Canvas;
467         import android.graphics.Color;
468         import android.graphics.Paint;
469         import android.graphics.RectF;
470
471         import com.google.android.gms.vision.ocrreader.ui.camera.GraphicOverlay;
472         import com.google.android.gms.vision.text.Text;
473         import com.google.android.gms.vision.text.TextBlock;
474
475         import java.util.List;
476
477
478         public class OcrGraphic extends GraphicOverlay.Graphic {
479
480             private int id;
481
482             private static final int TEXT_COLOR = Color.WHITE;
483
484             private static Paint rectPaint;
485             private static Paint textPaint;
486             private final TextBlock textBlock;
487
488             RecGraphic(GraphicOverlay overlay, TextBlock text) {
489                 super(overlay);
490
491                 textBlock = text;
492
493                 if (rectPaint == null) {
494                     rectPaint = new Paint();
495                     rectPaint.setColor(TEXT_COLOR);
496                     rectPaint.setStyle(Paint.Style.STROKE);
497                     rectPaint.setStrokeWidth(4.0f);
498                 }
499
```

```
500                 if (textPaint == null) {
501                     textPaint = new Paint();
502                     textPaint.setColor(TEXT_COLOR);
503                     textPaint.setTextSize(54.0f);
504                 }
505 
506                 postInvalidate();
507             }
508 
509             public int getId() {
510                 return id;
511             }
512 
513             public void setId(int id) {
514                 this.id = id;
515             }
516 
517             public TextBlock getTextBlock() {
518                 return textBlock;
519             }
520 
521 
522             public boolean contains(float x, float y) {
523                 if (textBlock == null) {
524                     return false;
525                 }
526                 RectF rect = new RectF(textBlock.getBoundingBox());
527                 rect = translateRect(rect);
528                 return rect.contains(x, y);
529             }
530 
531 
532            @Override
533             public void draw(Canvas canvas) {
534                 if (textBlock == null) {
535                     return;
536                 }
537 
538 
539                 RectF rect = new RectF(textBlock.getBoundingBox());
540                 rect = translateRect(rect);
541                 canvas.drawRect(rect, rectPaint);
542 
543               //  canvas.drawText(text.getValue(), rect.left, rect.bottom,
544         textPaint);
545                 List<? extends Text> textComponents = textBlock.getComponents();
546                     for(Text currentText : textComponents) {
547                         float left = translateX(currentText.getBoundingBox().left);
548                         float bottom =
549         translateY(currentText.getBoundingBox().bottom);
550                     canvas.drawText(currentText.getValue(), left, bottom,
551         textPaint);
552                 }
553             }
554 
555         }
```

ДОДАТОК Б. Електронна презентація

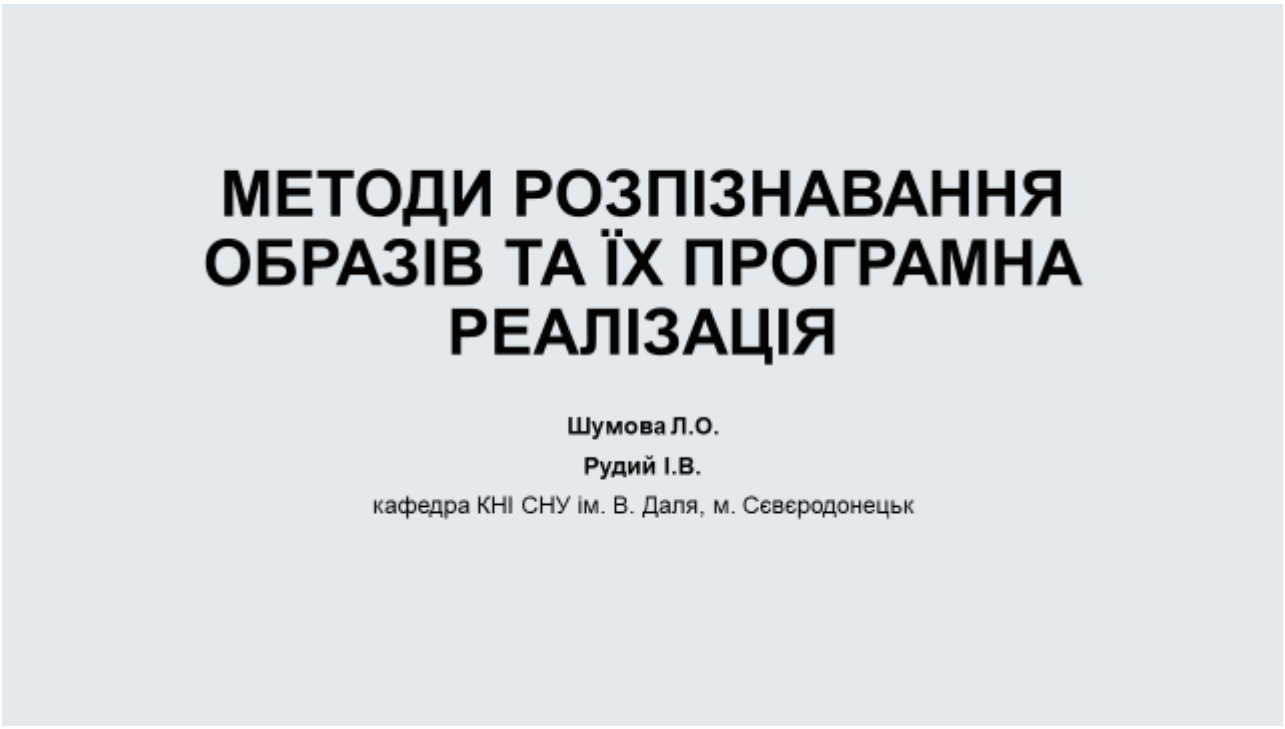

Рисунок Б.1 – Слайд1

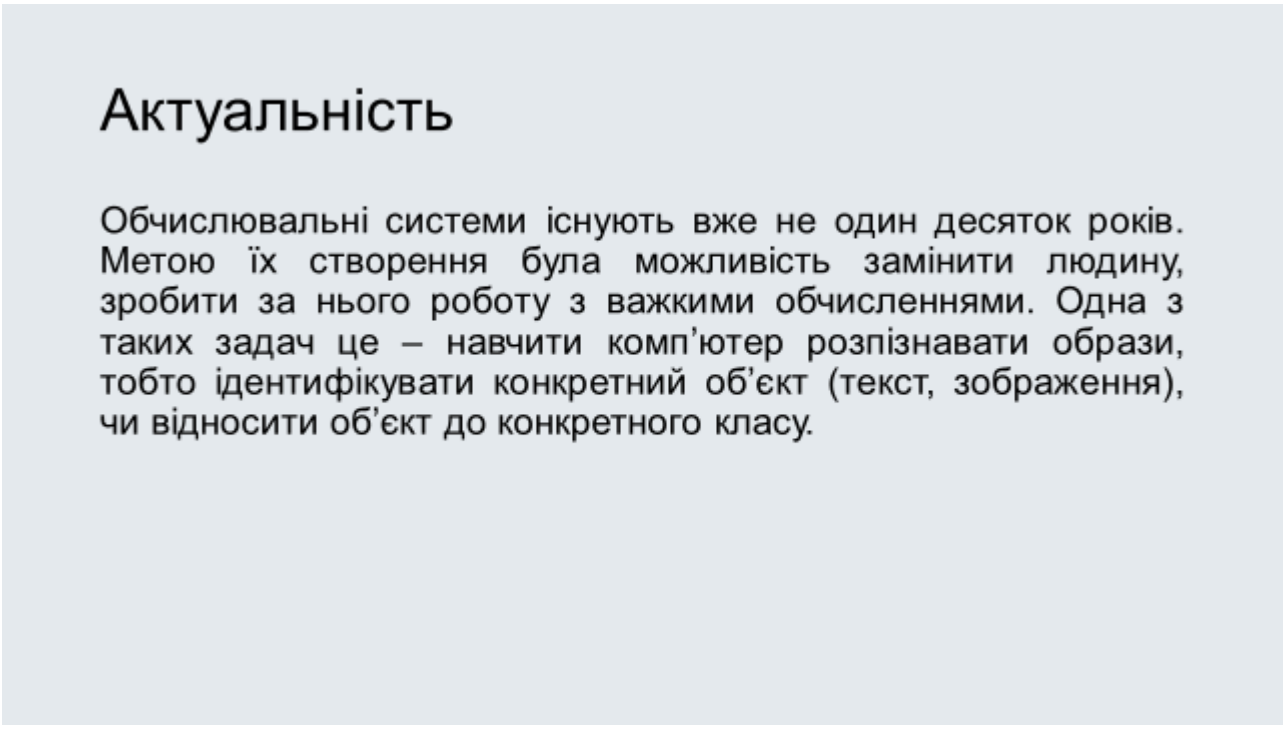

Рисунок Б.2 – Слайд2

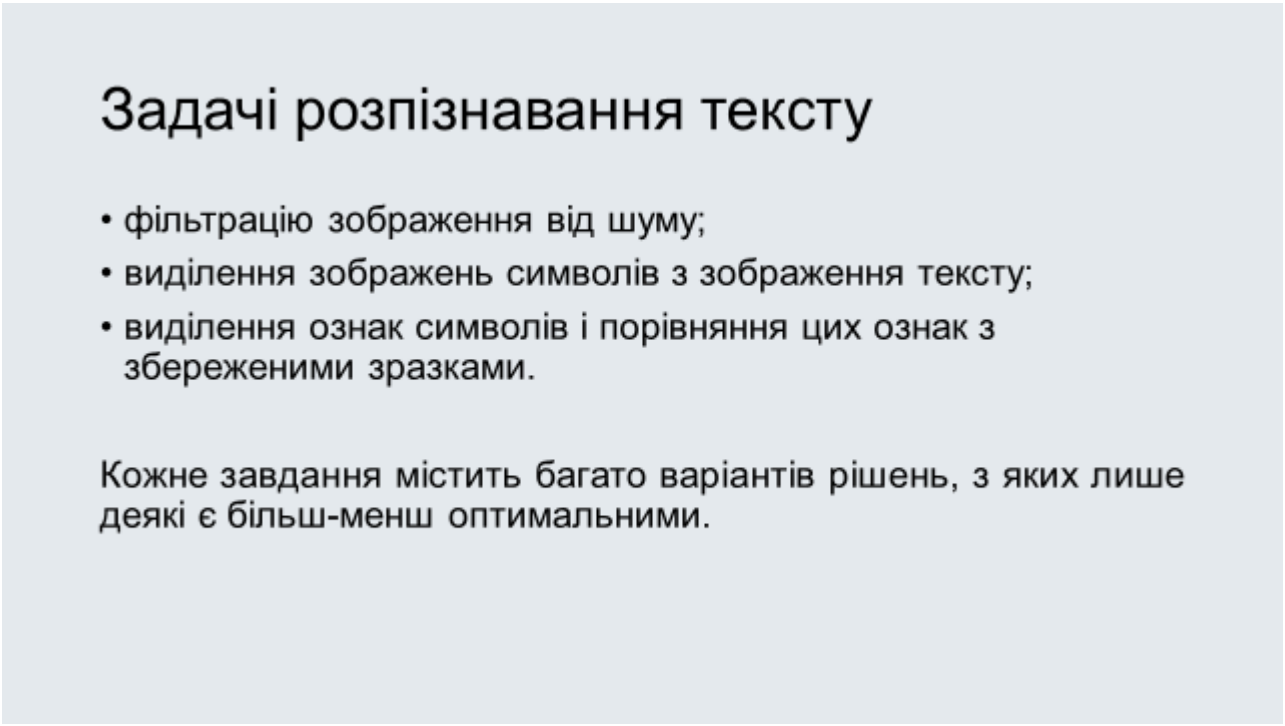

Рисунок Б.2 – Слайд3

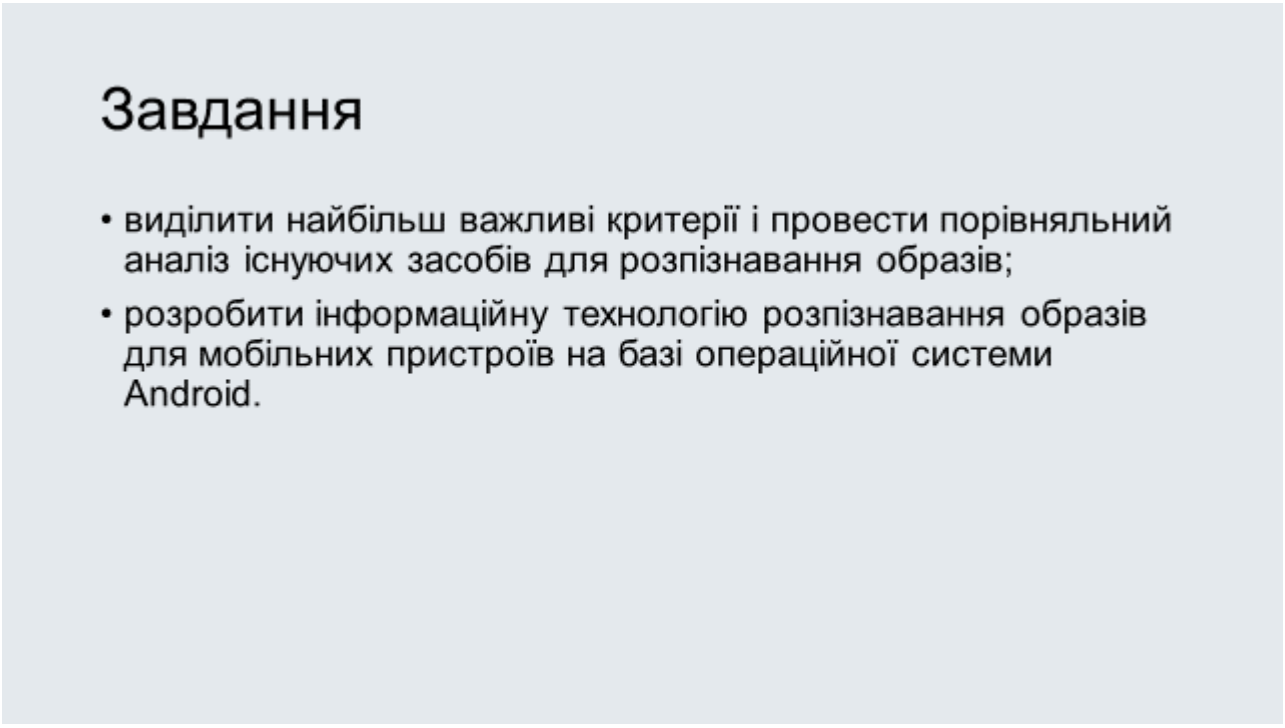

Рисунок Б.2 – Слайд4

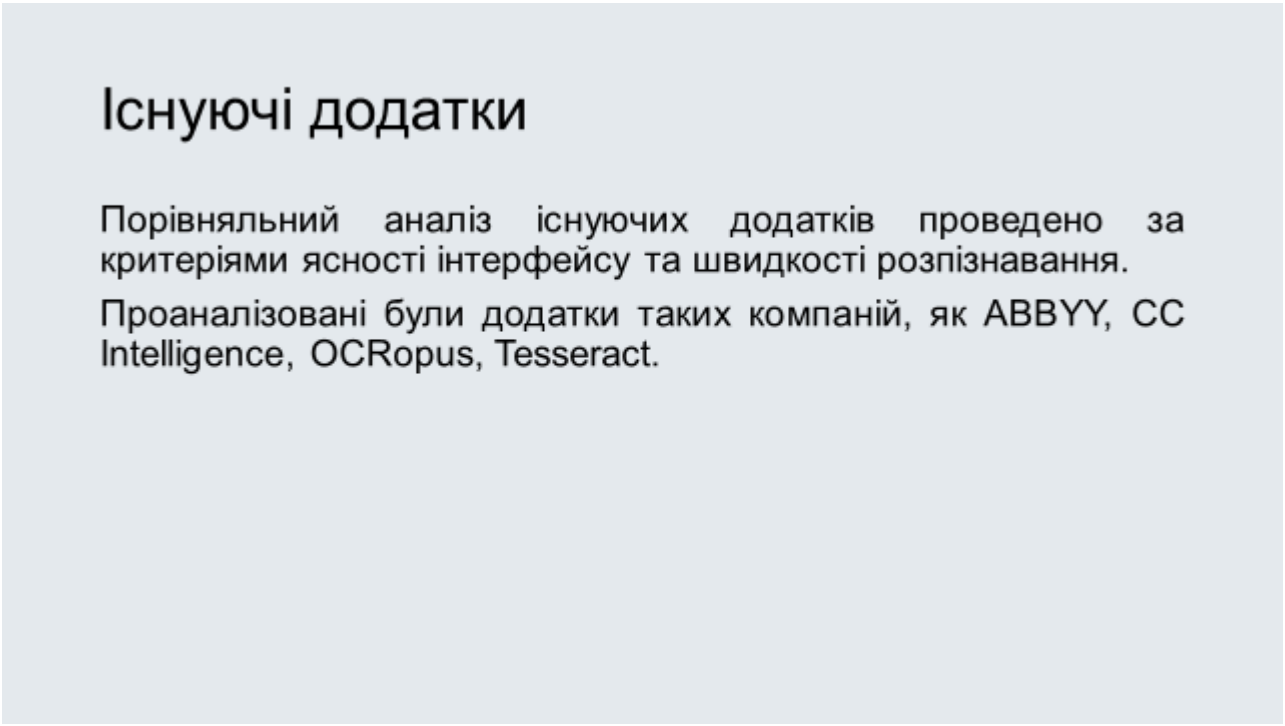

Рисунок Б.2 – Слайд5

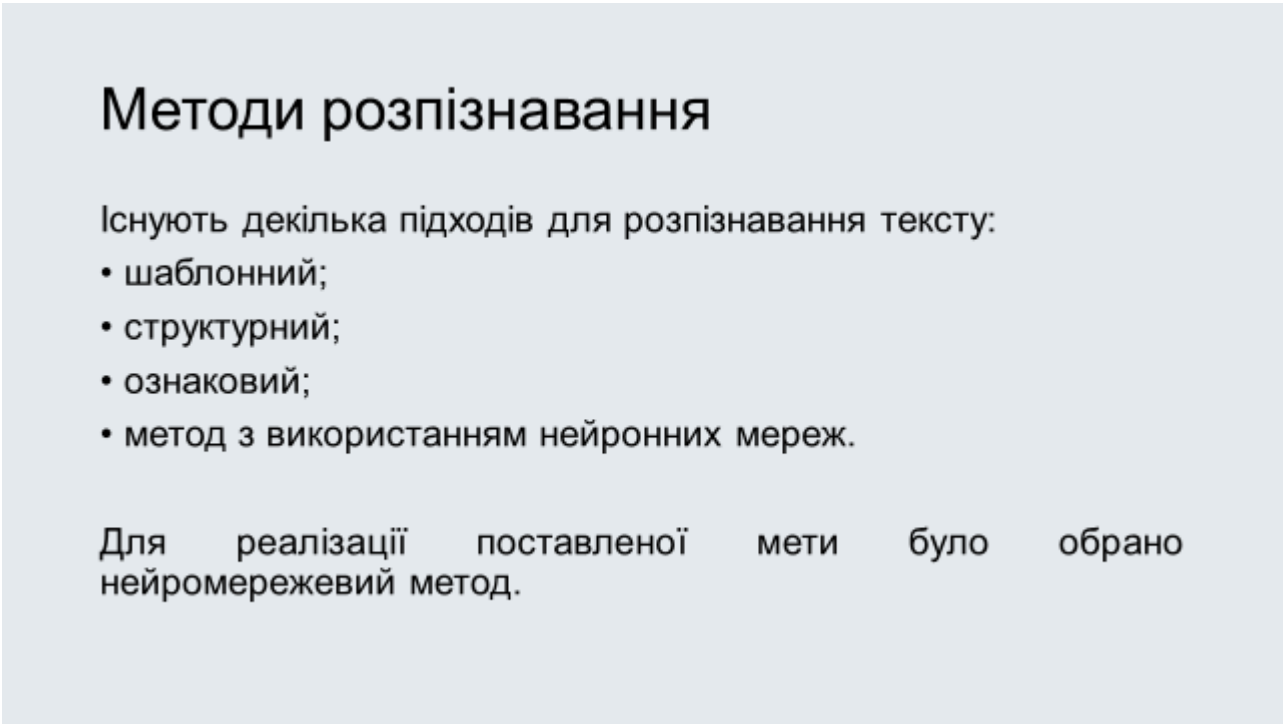

Рисунок Б.2 – Слайд6

Вимоги до додатку

Додаток має бути на мобільній платформі на даний момент можливо тільки на базі ОС Android, додаток повинен розпізнавати текст у реальному часі.

Рисунок Б.2 – Слайд7

Реалізація

Було вирішено використовувати Google ML Kit, к базову платформу для розгортання потрібної функціональності додатка, яка включає в себе Google Cloud Vision API, TensorFlow lite, Neural Network API.

Android Studio як середовище розробки, мову програмування Java.

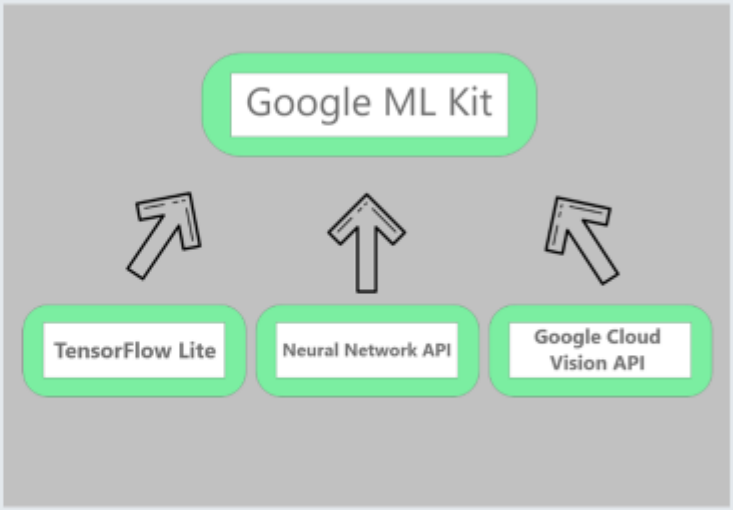

Рисунок Б.2 – Слайд8

Рисунок Б.2 – Слайд9

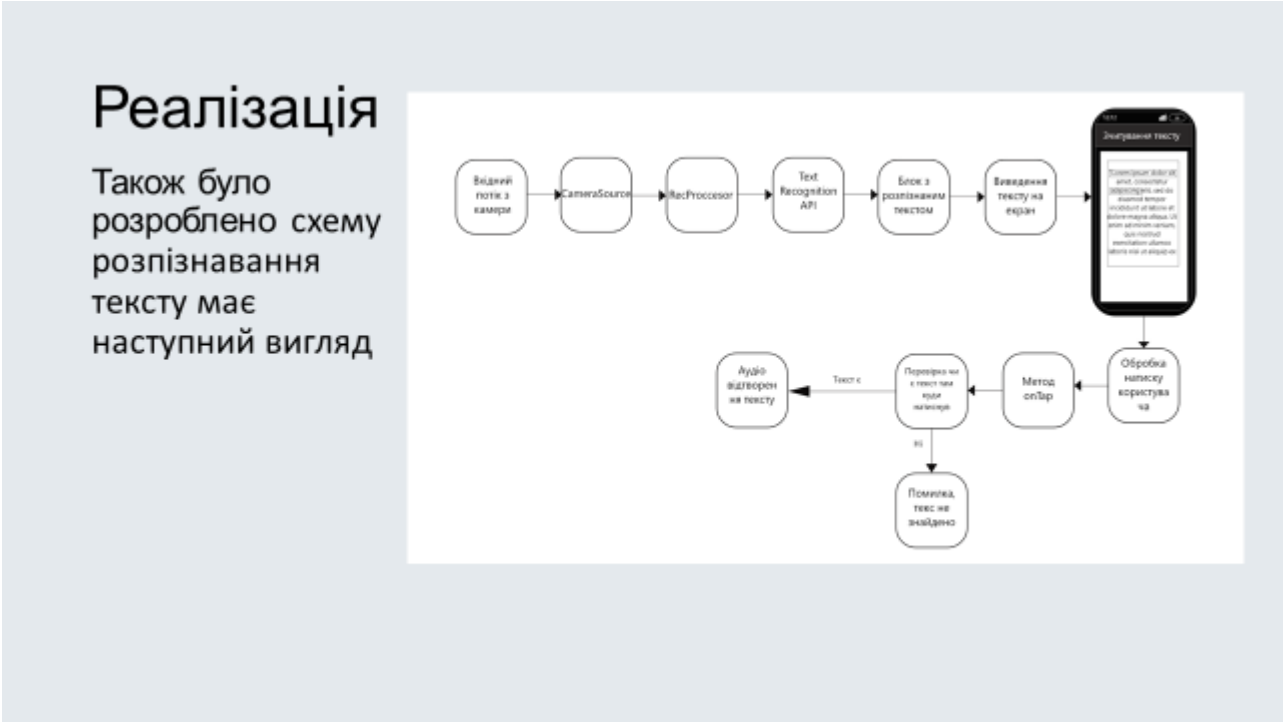

Рисунок Б.2 – Слайд10

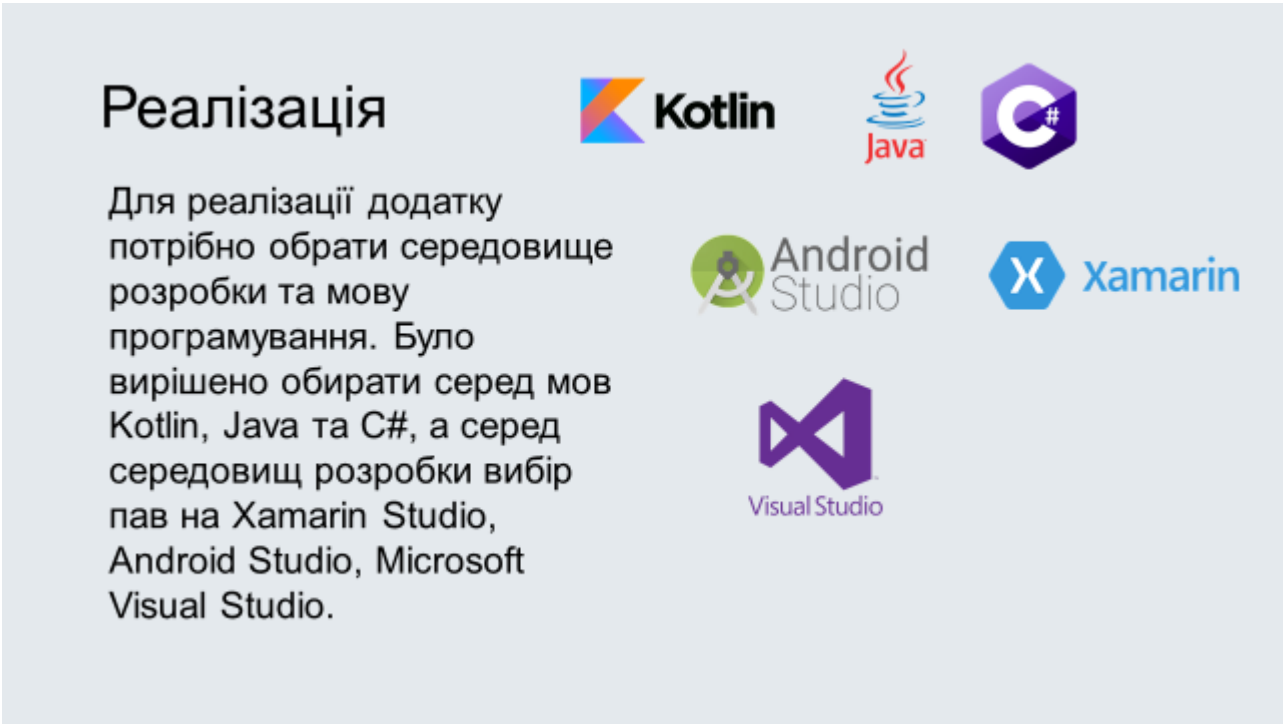

Рисунок Б.2 – Слайд11

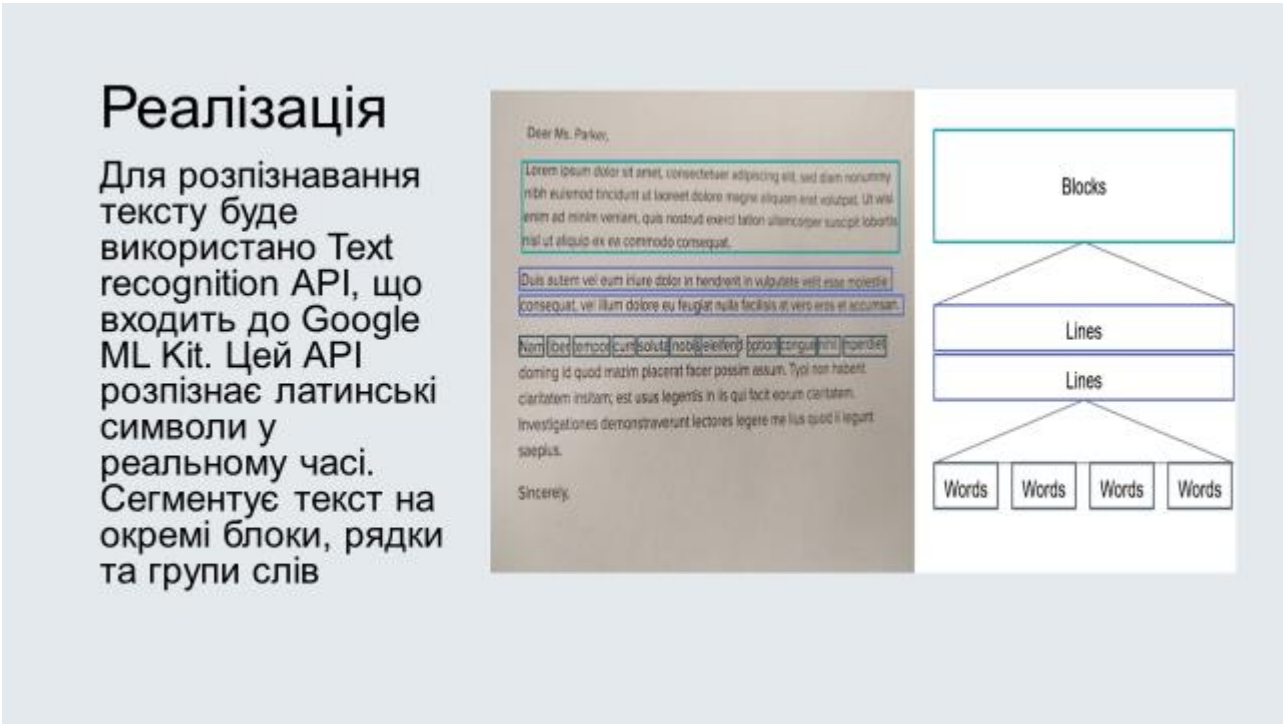

Рисунок Б.2 – Слайд12

Реалізація

Додаток розпізнає текст, виділяє розпізнаний текст та обробляє натиски, якщо користувач натиснув, то текст відтворюється у аудіо-форматі

Рисунок Б.2 – Слайд13

Висновок

Уході роботи було проведення аналізу засобів що допомагають у створенні додатків для розпізнавання образів, а також створення додатка, що розпізнає текст і аудіо відтворює его користувачу.

Рисунок Б.2 – Слайд14