**МІНІСТРЕТСВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Сумський державний університет**

Факультет електроніки та інформаційних технологій

Кафедра комп’ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

червня 2024 р.

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття освітнього ступеня бакалавр**

зі спеціальності 122 – Комп’ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Android додаток аудіовізуальної ідентифікації об'єктів у реальному часі для осіб з вадами зору»
здобувача групи ІН-01 Рубана Івана Олеговича

Робота містить опис власних досліджень, використаних технологій. Використані дослідження інших авторів мають посилання на відповідне джерело.

_____ Іван РУБАН
(підпис)

Керівник,
Старший викладач кафедри _____ Артем КОРОБОВ
(підпис)
комп’ютерних наук, к.т.н

**Суми 2024**

**Сумський державний університет**

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

# ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

**на здобуття освітнього ступеня бакалавр**

зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»

здобувача групи ІН-01 Рубана Івана Олеговича

1. Тема роботи: «Android додаток для аудіовізуальної ідентифікації об'єктів у реальному часі для осіб з вадами зору»

затверджую наказом по СумДУ від «22» квітня 2024 року № 0414-VI

2. Термін здачі здобувачем кваліфікаційної роботи до 1 червня 2024 року

_____

3. Вхідні дані до кваліфікаційної роботи

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

*1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд існуючих методів та рішень для вирішення поставленої задачі 3) Розробка додатку для аудіовізульної ідентифікації об'єктів для слабозорих 4) Аналіз результатів.*

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

_____

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

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |

7. Дата видачі завдання «____» _____ 2023 р.

Завдання прийняв до виконання _____ Керівник _____

(підпис) (підпис)

# КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів кваліфікаційної роботи | Термін виконання | Примітка |
|---|---|---|---|
| *1* | *Аналіз проблеми предметної області, постановка й формування завдань дослідження* | | |
| *2* | *Огляд існуючих методів та рішень для створення додатку* | | |
| *3* | *Розробка Android додатку для аудіовізуальної ідентифікації об'єктів* | | |
| *4* | *Аналіз отриманих результатів* | | |
| *5* | *Оформлення пояснювальної записки до кваліфікаційної роботи* | | |

Здобувач вищої освіти _____ (підпис)

Керівник _____ (підпис)

# АНОТАЦІЯ

**Записка:** 50 стр., 19 рис., 2 табл., 10 використаних джерел

**Обґрунтування актуальності теми дослідження –** Тема роботи є актуальною оскільки вади із зором є поширеною проблемою не тільки у дорослих людей, а і в молоді.

**Об’єкт дослідження –** використання новітніх технологій розпізнавання об’єктів для вирішення побутових проблем.

**Мета роботи –** розробка додатку, який зможе допомогти людям із вадами зору у побуті.

**Методи дослідження –** при виконанні роботи було використано новітні методи і бібліотеки машинного навчання, зокрема для створення моделі застосовано відкриту бібліотеку TensorFlow. Для розробки Android додатку використано новітній стек технологій, а також мову програмування Java.

**Результати дослідження –** створено альфа-версію Android додатка здатну до класифікації та розпізнавання об’єктів, як на фото, так і в режимі реального часу. Додаток може використовуватись людьми з вадами зору, завдяки зачитуванню тексту на екрані.

ДЕТЕКТОР, КЛАСИФІКАТОР, МАШИННЕ НАВЧАННЯ, ДЛЯ СЛАБОЗОРИХ, ANDROID.

ЗМІСТ

# ВСТУП

**Актуальність**. Актуальність дослідження зумовлена швидким розвитком нейронних мереж, та їх інтеграцією в повсякденне життя. Нейронні мережі з використанням технологій комп'ютерного зору здатні майже повністю замінити природній зір.

**Об’єкт дослідження**. Процес розробки нейронних мереж та можливість їх використання в умовах повсякденного життя.

**Предмет дослідження**. Рішення та методи для створення Android застосунку з вбудованою нейронною мережею.

**Гіпотеза**. Застосування нейронних мереж у повсякденному житті може значно полегшити життя як людям з певними вадами так і без них.

**Наукова новизна**. Дослідження спрямоване на інтеграцію новітніх технологій у сфері штучного інтелекту в повсякденне життя. Фінальним результатом дослідження стане розробка застосунку метою якого є полегшення процесу інтеграції у суспільство людей з вадами зору. Саме у цьому полягає наукова новизна дослідження.

**Структура**. Структура роботи складається з таких пунктів: вступ, розділ з аналізом предметної області, формулювання постановки задачі, вибір методів для вирішення задачі, розгляд програмного забезпечення, опис процесу розробки, висновки, список використаних джерел та додатки.

# 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Машинне навчання

Машинне навчання (англ. Machine learning) – це підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися» (тобто, поступово покращувати продуктивність у певній задачі) з даних без явного програмування. Еволюціонувавши з досліджень розпізнавання образів та теорії обчислювального навчання в галузі штучного інтелекту, машинне навчання досліджує вивчення та побудову алгоритмів, які можуть навчатися й робити прогнозування, опираючись на наявні дані – такі алгоритми долають слідування строго статичним програмним інструкціям, здійснюючи керовані даними прогнози або ухвалювання рішень шляхом побудови моделі з вибіркових входів.[1]

## 1.2 Вступ до глибокого навчання та нейронних мереж

Глибоке навчання є підгалуззю машинного навчання, яка спрямована на розв’язання складних завдань завдяки застосуванню нейронних мереж. Цей підхід заснований на імітації роботи людського мозку, де нейрони обробляють інформацію та передають її через зв'язки між ними. Нейронні мережі можуть виявляти складні закономірності в даних, що робить їх особливо потужними в задачах розпізнавання зображень, обробки природної мови, аналізу даних та інших областях.

Серед задач, які нейронні мережі здатні вирішити можна виділити такі:

- Класифікація об’єктів за класами, наперед визначеними в моделі;
- Класифікація без учителя. Класи наперед не задано. Модель класифікує надані зображення за певними ознаками на різні групи;

- Апроксимація функцій. Визначити функцію, за заданим набором даних типу «вхід-вихід»;
- Організація пам’яті;
- Оптимізація;
- Прогнозування послідовностей;
- Фільтрація;
- Керування певними системами.

## 1.3 Згорткові нейронні мережі (Convolutional Neural Networks – CNNs)

Згорткові нейронні мережі є одними з найбільш ефективних та популярних типів нейронних мереж для обробки зображень. Вони були розроблені спеціально для розпізнавання образів та класифікації зображень.

Для задач розпізнавання в умовах використання великих об’ємів даних необхідно, щоб модель мала високу здатність до навчання та великий відсоток правильних припущень щодо ознак зображення. У порівнянні з традиційними нейронними мережами прямого поширення зі схожою кількістю шарів, згорткові нейронні мережі мають вищу здатність до навчання, оскільки містять набагато менше параметрів та зв’язків. У традиційних повнозв’язних нейронних мереж для їх правильного навчання в задачах розпізнавання образів необхідна набагато більша кількість даних, оскільки кожне вхідне зображення може мати розмірність у щонайменше кілька сотень тисяч, що вимагатиме відповідну кількість прикладів з різними значеннями для кожного виміру.[2]

Згоркові мережі приймають і обробляють дані у вигляді так званих тензорів – чотирьохвимірних масивів даних. Зазвичай цей масив складається з таких величин як: ширина та довжина зображення, глибина, карта виявлених ознак.

Глибина зображення - це величина що визначається кодуванням зображення. Найбільш поширеним із них є RGB-кодування, в якому колір кожного пікселя на зображенні кодується за допомогою значень для трьох кольорів – червоного, зеленого та синього. Такі категорії значень, що описують зображення, називаються каналами.[3]

Карта виявлених ознак являє собою тривімирний масив, який відображає в собі значення що відповідають певному ступеню присутності тої чи іншої ознаки на зображенні.

Основна ідея згорткових нейронних мереж полягає в використанні згорткових шарів для автоматичного вивчення властивостей та характеристик зображень. Ці шари використовують фільтри, щоб визначити особливості зображення та підкреслити їх важливість для класифікації.

Зазвичай така нейронна мережа складається з декількох видів шарів (Рис. 1.1):

- Входу;
- Прихованих шарів;
- Виходу.

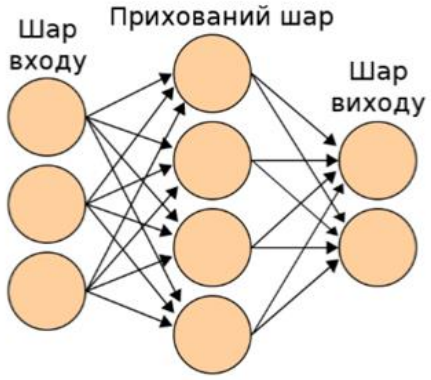


Рисунок 1.1 - Будова нейронної мережі

В той час приховані шари можуть містити в собі:

- Згорткові шари;

- Агрегувальні шари;
- Повноз'єднані шари;
- Шари нормалізації.

Згорткові шари виконують операцію згортки і потім передають зображення до наступного шару. Наприклад, повноз'єднаний шар для (маленького) зображення розміром 100 × 100 має 10 000 ваг. Операція згортки дає змогу розв'язати цю проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів. Наприклад, незалежно від розміру зображення, області замощування розміру 5 × 5, кожна з одними й тими ж спільними вагами, вимагають лише 25 вільних параметрів. Таким чином, це розв'язує проблему зникання або вибуху градієнтів у тренуванні традиційних багатошарових нейронних мереж з багатьма шарами за допомогою зворотного поширення.[1]

Агрегувальні шари призначені для фільтрування вибірки значень з декількох кластерів нейронів. Наприклад, мінімізаційне агрегування виконує вибір найменшого значення з кожного кластеру нейронів попереднього шару.

Повноз'єднані шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару. Це, в принципі, є тим же, що й традиційна нейронна мережа багатошарового перцептрону. ЗНМ використовують спільні ваги в згорткових шарах, що означає, що для кожного рецептивного поля шару використовується один і той же фільтр (банк ваг); це зменшує обсяг необхідної пам'яті та поліпшує продуктивність.[1]

### 1.4 Задача класифікації зображень

Задача класифікації зображень полягає в призначенні міток або класів до зображень на підставі їх вмісту. Наприклад, це може бути класифікація зображень на категорії "кіт", "собака" або "автомобіль". Моделі класифікації зображень засновані на тренуванні нейронних мереж з великими наборами

даних, що містять як зображення, так і відповідні мітки класів.

Завдання класифікації зображень знаходить широке застосування у різних областях, таких як розпізнавання об'єктів, медична діагностика, автономні транспортні засоби, військова справа тощо.

У цьому дослідженні ми зосередимось на створенні користувацької моделі класифікації зображень з використанням згорткових нейронних мереж у TensorFlow.

Для альфа-версії додатку навчимо модель класифікувати фрукти. Класифікація фруктів не потребує величезної бази зображень для навчання моделі, тому для початкової версії додатку є ідеальним рішенням.

### 1.5 Задача розпізнавання об’єктів

Задача розпізнавання зображень полягає у створенні так званого «детектора» для розпізнавання і класифікації об’єктів в режимі реального часу. Сама задача розпізнавання потребує більш широкого розуміння контексту зображення, на відміну від простої класифікації. Одним із ключових аспектів такої задачі є здатність моделі до розпізнавання об’єктів на зображенні не залежно від їх положення, масштабу, орієнтації та інших факторів. Ця вимога означає те що база даних для такої моделі буде значно більшою за таку ж модель лише для класифікації.

Завдання розпізнавання зображень також дуже популярне у таких областях як: розпізнавання об'єктів, медична діагностика, автономні транспортні засоби, військова справа, автоматизація виробництва, робототехніка та інші.

Для альфа-версії додатку взято готову модель-детектор. Вона є публічно доступною в мережі і містить в собі деякі готові класи об’єктів.

### 1.6 Постановка задачі

Ґрунтуючись на інформацію вище можна зробити висновок що нейронні мережі в наш час здатні повністю, або частково, замінити природній зір. Ця властивість може стати надзвичайно корисною для людей з вадами зору.

Зрозуміло, що для повної заміни природного зору модель мережі має містити в собі величезну кількість даних, тому основною метою роботи є створення альфа-версії такого додатку, з можливістю його вдосконалення в майбутньому. Моделі нейронних мереж будуть мати обмежену кількість класів (наприклад фруктів та овочів).

Додаток повинен мати простий інтерфейс, а також вміти зачитувати текст з екрану, щоб люди з вадами зору мали змогу ним користуватися. Повинна бути можливість розпізнавання об’єктів як на фото так і в режимі реального часу.

Етапи виконання задачі включають:

- Вибір методів для вирішення поставленої задачі;
- Створення та навчання моделі нейронної мережі;
- Створення зручного інтерфейсу для доступу і використання моделі;
- Розгортання моделі на операційній системі Android;
- Тестування моделі у різних режимах.

Такий підхід дозволить створити додаток, який буде ефективно використовувати новітні технології сфери нейронних мереж для допомоги людям з обмеженими можливостями.

# 2. ОГЛЯД ТИ ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

## 2.1 Операційна система Android

Android – це операційна система з відкритим кодом, створена компанією-гігантом Google, на базі ядра Linux.

Вперше початковий код Android був опублікований 21 жовтня 2008 року. З тих пір ця операційна система вдосконалювалася і розвивалася величезними кроками.

Зараз Android має зручний інтерфейс, величезну базу підтримуваних додатків та пристроїв, а також різноманітні системи безпеки даних. Завдяки наявності різних віджетів, шпалер, бібліотеки додатків Google Play Store, користувач може налаштовувати свій пристрій на свій лад. Це робить систему адаптивною під потреби окремих користувачів.

Близько 70.43% смартфонів (див. Таблиця 2.1) використовують саме цю операційну систему саме через її зручність, а також відносно низьку вартість підтримуваних пристроїв.

Таблиця 2.1 – Статистика ОС для смартфонів[4]

| Операційна система | Відсоток використання |
|---|---|
| Android | 70.43% |
| iOS | 29.06% |
| Samsung | 0.16% |
| KaiOS | 0.11% |
| Windows | 0.07% |
| Tizen | 0.02% |
| Nokia | 0.02% |

Що цікаво, оскільки в наш час саме мобільні пристрої стали більш

популярні ніж статичні ПК, і мабуть в кожного завжди при собі такий пристрій, то популярність Android системи є навіть більшою ніж популярність такої ОС як Windows (див. Таблиця 2.2).

Таблиця 2.2 – Загальна світова статистика використання ОС[4]

| Операційна система | Відсоток використання |
|---|---|
| Android | 39.77% |
| Windows | 32.31% |
| iOS | 17.66% |
| OS X | 7.98% |
| Linux | 0.69% |
| Chrome OS | 0.47% |
| Xbox | 0.1% |

Виходячи з такої популярності, саме Android розробка є дуже перспективною галуззю розробки програмного забезпечення та додатків.

Для розробників Android є особливо зручним завдяки відкритому коду, а також великої кількості готових рішень та документацій. ОС забезпечує єдиний підхід до розробки додатків що дає можливість створювати додатки під різні платформи використовуючи один і той самий код.

Найбільш популярними інтегрованими середовищами розробки (IDE) є IntelliJ IDEA та Android Studio. Ці середовища створені компанією JetBrains.

IntelliJ IDEA є потужним інструментом призначеним під різні технології розробки. Підтримує велику кількість мов та бібліотек, а також має вбудовану систему контроля версій. Розробник може конфігурувати цю систему під свої потреби шляхом вбудованої бібліотеки розширень. Однак з цієї адаптивності випливає і величезний мінус цієї середи. Останнім часом IntelliJ IDEA стало надзвичайно перевантаженою системою, тому компанія-розробник JetBrains почала створювати IDE спеціалізовані під певні

напрямки.

Android Studio в свою чергу є «наслідником» IntelliJ IDEA. Це модифікація середи розробки призначена для розробки саме під Android. Вона має в собі вбудовані засоби для розгортання та тестування мобільних додатків, функціонал для побудови та компоновки інтерфейсів, живі макети та шаблони для поширених Android додатків. Android Studio постійно оновлюється щоб підтримувати новітні технології та методики у розробці програмного забезпечення.

Android Studio містить інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.[5]

Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше опублікувати версію мобільного додатка в Google Play App Store. Для цього є також підтримка інструментів LINT, Pro-Guard і App Signing. [5]

## 2.2 Мови програмування додатку

Для розробки Android додатків існує дві популярні мови програмування: Java та Kotlin. Це звісно не єдині мови, які підтримують розробку мобільних застосунків, проте більшість додатків розроблено саме з їх використанням.

Java є доволі популярною мовою в наш час. Ця класична мова програмування зіграла важливу роль у розробці Android-додатків. Java була першою офіційною мовою програмування для Android. Це рішення Google ухвалив з метою зробити розробку додатків на Android більш доступною і поширеною. За допомогою Java було створено величезну кількість додатків, які досі успішно працюють на мільйонах пристроїв.[6] Завдяки тому що ця мова програмування існує доволі довго, вона нажила величезну кількість

фанатів і розробників. Завдяки цьому існує безліч готових рішень, фреймворків, бібліотек та ресурсів.

Kotlin в свою чергу є відносно новою мовою програмування, але з початку свого існування вона стала надзвичайно популярною серед мобільних розробників. Ця мова виявилася зручнішою і ефективнішою у використанні ніж Java. Підтвердженням такого успіху мови стало те, що компанія Google у 2017 році оголосила Kotlin офіційною мовою розробки на Android.

Основною перевагою Kotlin стала економія часу написання коду, оскільки для більшості задач код займав значно менше розміру ніж код для аналогічної задачі на Java. Також плюсом стала сумісність з Java. Це дозволяє розробникам впроваджувати Kotlin у вже готові проекти Java.

Але, все ж, однією з ключових переваг Kotlin є його нативна інтеграція з Android Studio – офіційним середовищем розробки для Android від Google. Це означає, що ви можете створювати додатки на Kotlin з максимальним рівнем зручності та підтримки.[6]

Для виконання цієї роботи я обрав Java. Спираючись на досвід роботи з нею, а також велику підтримку зі сторони ком’юніті розробників, я вважаю що використання цієї мови буде доцільним.

### 2.3 Технології для створення нейронної мережі

Нейромережа є ключовим елементом в роботі додатку. Саме від неї залежить точність і успішність розпізнавання об'єктів. Тому вибір технологій написання нейромережі є головною задачею розробника.

Серед найбільш популярних мов програмування, що підтримують машинне навчання, є Python, R, Java, C++ та Julia. Всі ці мови мають в собі велику кількість різних бібліотек та пакетів, що надають можливість розробнику створювати моделі для статистичних розрахунків.

Для виконання даного завдання було обрано мову Python. Вона має кілька переваг над іншими мовами:

- Простота написання коду. Багато розробників виділяють Python, як надзвичайно зручну мову для новачків через простоту її синтаксису.
- Велике ком'юніті розробників. Наслідком великої популярності цієї мови в наш час є величезне співтовариство розробників, готових допомогти один одному у вирішенні складних задач.
- Великий вибір бібліотек написання нейронних мереж. Python може надати розробнику можливість обрати оптимальну бібліотеку для рішення його задачі. Це можуть бути такі бібліотеки як: TensorFlow, Keras, Scikit-Learn, OpenCV, PyTorch та інші.

Для виконання моєї роботи оптимальним рішенням я вважаю використати бібліотеку TensorFlow. TensorFlow є відкритою бібліотекою для машинного навчання, створеною Google. Вона забезпечує високорівневий інтерфейс для побудови та тренування нейронних мереж та інших моделей машинного навчання.

TensorFlow підтримує графову обчислювальну модель, яка дозволяє використовувати автоматичну диференціацію та ефективно виконувати обчислення на графічних процесорах (GPU) та тензорних процесорах (TPU). Вузли графу – це операції різних видів, як наприклад запуск нейрона, або обчислювальні дії з матрицями.

Для виконання операцій у графі використовуються сесії. Це – середовище виконання для графа, яке керує потоком даних між операціями. Коли сесія запускається, вона забезпечує виконання всіх необхідних операцій для отримання необхідного результату.[7]

Хоча Python надає зовнішній API для TensorFlow, фактичні математичні операції не виконуються в Python. Натомість високопродуктивні

двійкові файли C++ виконують ці операції за лаштунками. Python спрямовує трафік між частинами та об’єднує їх разом за допомогою абстракцій програмування високого рівня.[8]

Одним із головних переваг TensorFlow є його широке застосування у галузях комп'ютерного бачення, обробки природньої мови, генеративних моделях та багатьох інших областях машинного навчання.

### 2.4 Розгортання моделі на Android пристроях

У 2017 році Google представила нову версію TensorFlow під назвою TensorFlow Lite. TensorFlow Lite оптимізовано для використання на вбудованих і мобільних пристроях. Це готова кроссплатформенна структура глибокого навчання з відкритим вихідним кодом, яка перетворює попередньо навчену модель TensorFlow у спеціальний формат, який можна оптимізувати для швидкості чи зберігання.[8]

TensorFlow Lite виконує оптимізацію шляхом застосування квантування та скорочення ваг. Також за допомогою апаратних прискорювачів відбувається поліпшення швидкості, точності й оптимізація енергоспоживання.

### 2.5 Середа розробки нейронної мережі

Google Colab (Colaboratory) - це безкоштовна хмарна платформа для виконання коду Python, яка надає доступ до ресурсів Google, включаючи обчислювальну потужність та обсяг пам'яті. Colab дозволяє користувачам створювати та виконувати Jupyter-подібні блокноти, що включають код Python, текстові комірки, графіки та інші візуальні елементи.

Однією з ключових переваг Colab є можливість використання обчислювальних ресурсів Google, включаючи GPU та TPU, для прискорення виконання коду, зокрема для навчання глибоких нейронних мереж. Крім

того, він має попередньо встановлені бібліотеки для машинного навчання, такі як TensorFlow, що робить його зручним інструментом для початківців та досвідчених дослідників.

Також, використання Colab дозволяє легко ділитися кодом та результатами досліджень з колегами або публікувати їх онлайн для спільної роботи та дискусій. Це робить Google Colab популярним інструментом для навчання та досліджень у галузі машинного навчання та глибокого навчання.

Colab широко застосовується у сфері машинного навчання для таких завдань:

- початок роботи з TensorFlow;
- розробка й навчання нейронних мереж;
- експерименти з тензорними процесорами;
- поширення досліджень у сфері ШІ;
- створення навчальних посібників. [9]

# 3. Реалізація системи

## 3.1 Розробка нейронної мережі

Для початку було знайдено зображення, необхідні для навчання моделі та поділено на три папки: тестові, тренувальні та для валідації. В кожній папці наявно три класи зображень: яблуко, апельсин, банан. Всі зображення поміщено в архів.

Переходимо до Google Colab та створюємо там новий блокнот (Рис. 3.1). Далі завантажуємо даний архів до Google Colab, та виконуємо розархівування файлів (Рис. 3.3).

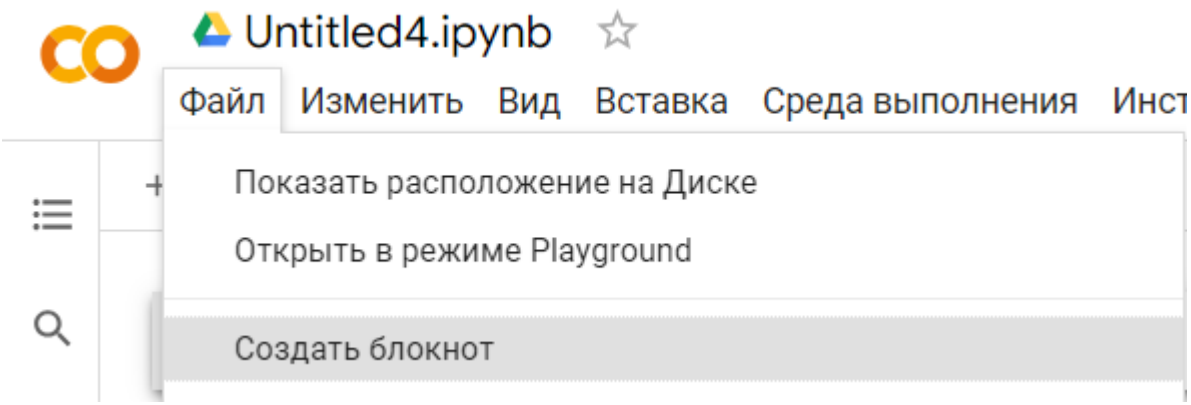


Рисунок 3.1 - Створення блокноту

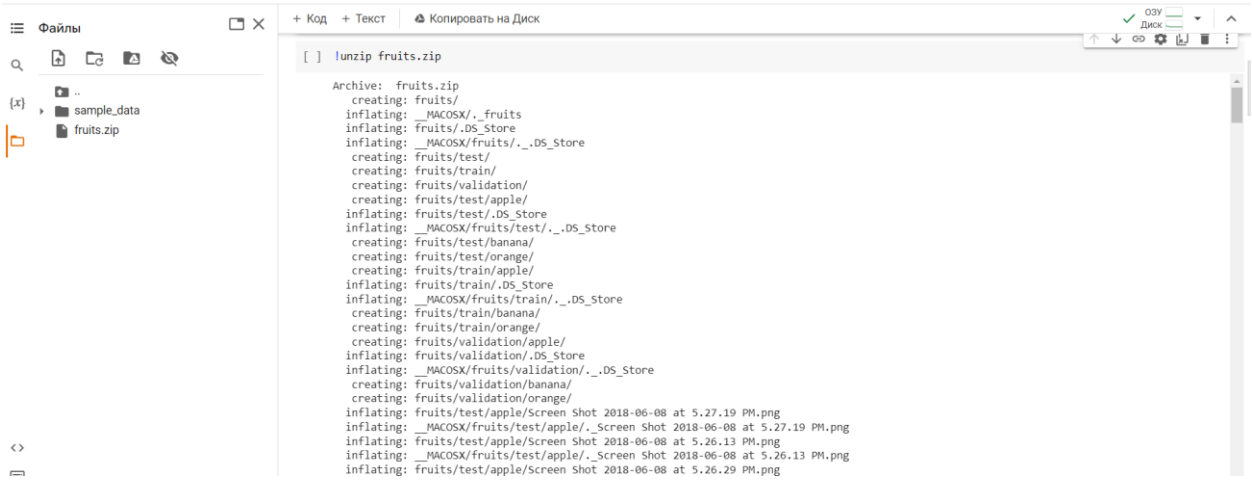


Рисунок 3.2 - Розархівування

Після того як в лівій частині з’явилися відповідні файли, необхідно підключити бібліотеки (Рис 3.3).

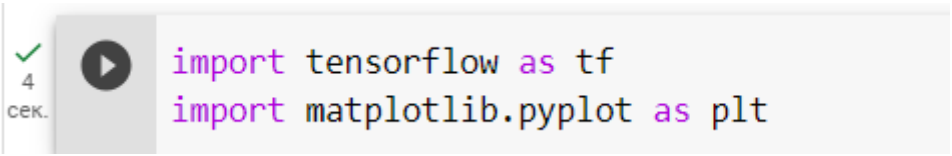


Рисунок 3.3 - Підключення бібліотек

Далі необхідно створити змінні (Рис. 3.4) в яким буде присвоєно зображення фруктів. Розмір зображень змінюємо на 32x32 пікселі.

```
img_height, img_width = 32, 32
batch_size = 20

train_ds = tf.keras.utils.image_dataset_from_directory(
    "fruits/train",
    image_size = (img_height, img_width),
    batch_size = batch_size
)
val_ds = tf.keras.utils.image_dataset_from_directory(
    "fruits/validation",
    image_size = (img_height, img_width),
    batch_size = batch_size
)
test_ds = tf.keras.utils.image_dataset_from_directory(
    "fruits/test",
    image_size = (img_height, img_width),
    batch_size = batch_size
)
```

```
Found 460 files belonging to 3 classes.
Found 66 files belonging to 3 classes.
Found 130 files belonging to 3 classes.
```

Рисунок 3.4 - Створення змінних

Протестуємо як виглядають навчальні зображення і чи співпадають назви їх класів (Рис. 3.5 – 3.6).

```
class_names = ["apple", "banana", "orange"]
plt.figure(figsize=(10,10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Рисунок 3.5 - Перегляд зображень

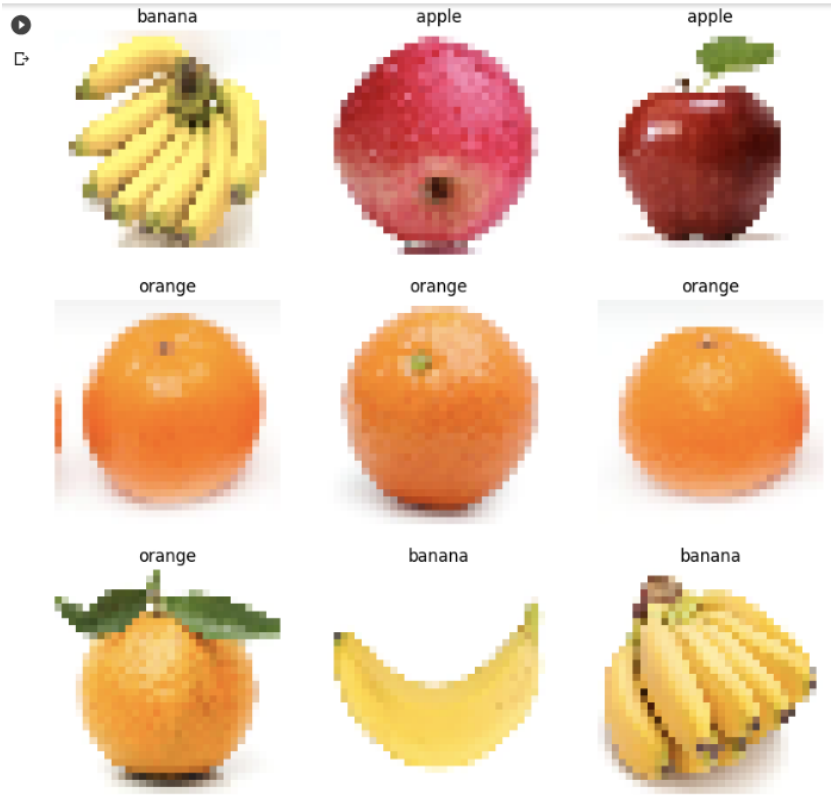


Рисунок 3.6 - Навчальні зображення

Розпочнемо створення моделі. Створимо змінну model і за допомогою функції Sequential напишемо шари нашої моделі (Рис. 3.7):

1) Перший шар змінить значення RGB кожного пікселя з системи 0-255, в систему 0-1.
2) Наступний шар – згортковий. Він застосовує фільтри до зображення щоб дізнатись про його певні особливості. В моєму випадку застосовую 32 фільтри з розміром ядра 3x3.
3) Далі шар для зменшення карти зображення. Він порівнює значення кожного пікселя в області 2x2, та обирає найбільше значення.
4) Повторимо декілька разів другий і третій шар.
5) Наступний шар перетворює вхідні дані в один вимір замість двох.
6) Наступні шари містять нейрони, які будують рішення стосовно

класу об’єкта на зображенні посилаючись на результати попередніх шарів.

```
model = tf.keras.Sequential(
    [
     tf.keras.layers.Rescaling(1./255),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Flatten(),
     tf.keras.layers.Dense(128, activation="relu"),
     tf.keras.layers.Dense(3)
    ]
)
```

Рисунок 3.7 - Шари моделі

Компілюємо створену модель (Рис. 3.8). Вказуємо оптимайзер та функцію втрат, вони визначають наскільки ефективно буде навчатися модель. Виводимо точність для оцінки моделі.

```
model.compile(
    optimizer="adam",
    loss=tf.losses.SparseCategoricalCrossentropy(from_logits = True),
    metrics=['accuracy']
)
```

Рисунок 3.8 - Компіляція моделі

Почнемо навчання нашої моделі. Для цього, покажемо нашій моделі навчальні зображення десять разів (Рис. 3.9). Можна побачити, що з кожним наступним паком зображень, точність (accuracy, val_accuracy) збільшується.

```
model.fit(
    train_ds,
    validation_data = val_ds,
    epochs = 10
)
```

```
Epoch 1/10
23/23 [==============================] - 13s 58ms/step - loss: 1.0553 - accuracy: 0.4783 - val_loss: 0.9212 - val_accuracy: 0.6515
Epoch 2/10
23/23 [==============================] - 2s 49ms/step - loss: 0.7232 - accuracy: 0.7261 - val_loss: 0.6831 - val_accuracy: 0.6667
Epoch 3/10
23/23 [==============================] - 2s 48ms/step - loss: 0.4499 - accuracy: 0.8000 - val_loss: 0.5202 - val_accuracy: 0.7424
Epoch 4/10
23/23 [==============================] - 3s 89ms/step - loss: 0.3012 - accuracy: 0.8696 - val_loss: 0.3060 - val_accuracy: 0.8939
Epoch 5/10
23/23 [==============================] - 2s 48ms/step - loss: 0.2617 - accuracy: 0.9065 - val_loss: 0.4174 - val_accuracy: 0.8182
Epoch 6/10
23/23 [==============================] - 2s 49ms/step - loss: 0.2391 - accuracy: 0.9152 - val_loss: 0.2599 - val_accuracy: 0.8939
Epoch 7/10
23/23 [==============================] - 2s 50ms/step - loss: 0.1644 - accuracy: 0.9370 - val_loss: 0.2612 - val_accuracy: 0.9091
Epoch 8/10
23/23 [==============================] - 2s 50ms/step - loss: 0.1964 - accuracy: 0.9217 - val_loss: 0.1203 - val_accuracy: 0.9697
Epoch 9/10
23/23 [==============================] - 2s 53ms/step - loss: 0.1499 - accuracy: 0.9413 - val_loss: 0.2338 - val_accuracy: 0.8939
Epoch 10/10
23/23 [==============================] - 2s 49ms/step - loss: 0.1436 - accuracy: 0.9500 - val_loss: 0.1391 - val_accuracy: 0.9545
<keras.callbacks.History at 0x7e7d20c83520>
```

Рисунок 3.9 - Навчання моделі

Протестую модель на зображеннях, які вона ще ніколи не бачила (Рис. 3.10-3.11). Точність становить 93.85%.

```
model.evaluate(test_ds)
```

```
7/7 [==============================] - 0s 20ms/step - loss: 0.2248 - accuracy: 0.9385
[0.22483572363853455, 0.9384615421295166]
```

Рисунок 3.10 - Результати тестування

```
import numpy

plt.figure(figsize=(10,10))
for images, labels in test_ds.take(1):
  classifications = model(images)
  # print(classifications)

  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    index = numpy.argmax(classifications[i])
    plt.title("Pred: " + class_names[index] + " | Real: " + class_names[labels[i]])
```

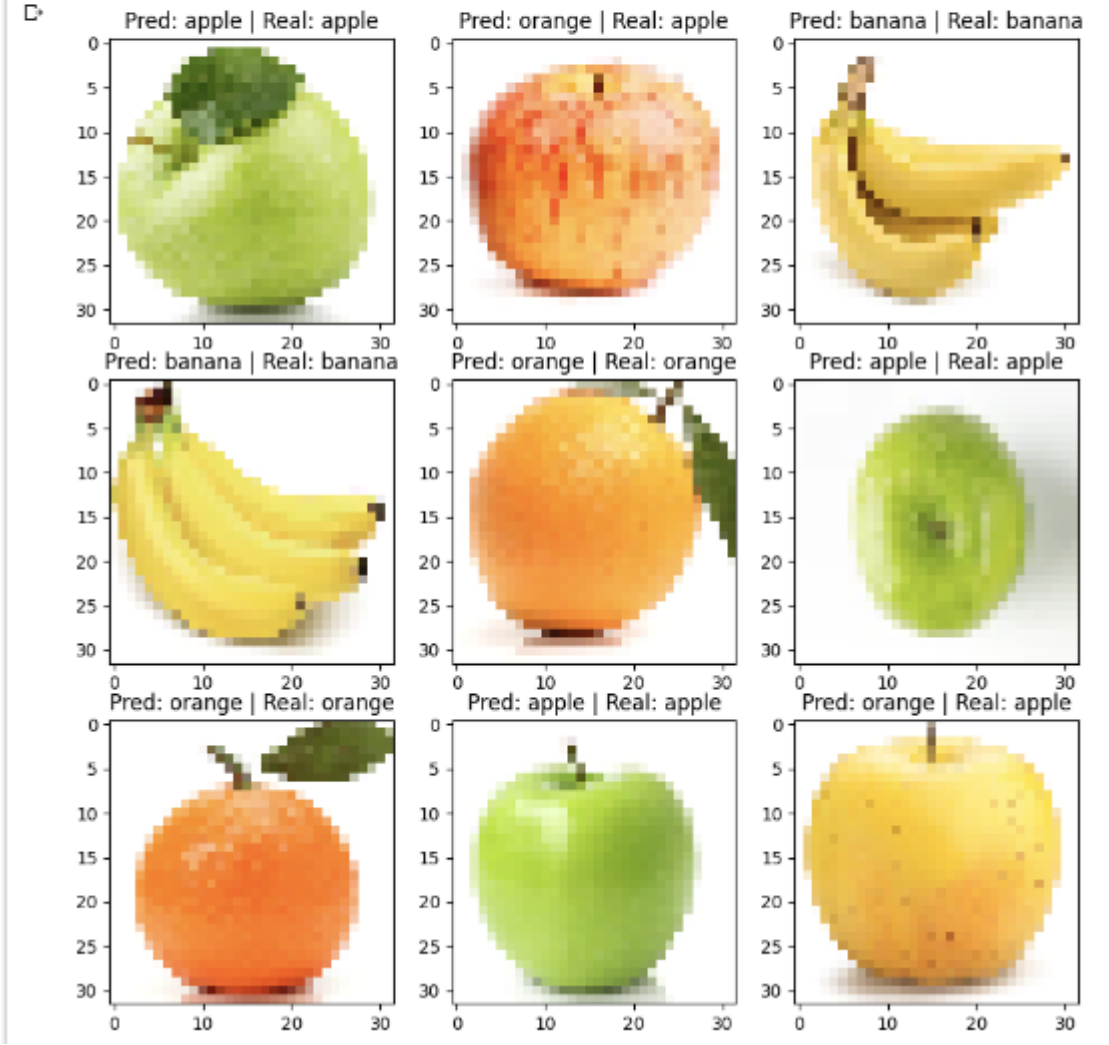


Рисунок 3.11 - Тест моделі

Фінальним кроком є конвертація цієї моделі в модель TensorFlowLite (Рис. 3.12). Готовий файл моделі доступний до завантаження і подальшого використання в Android додатках (Рис. 3.13).

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

with open("model.tflite", 'wb') as f:
  f.write(tflite_model)
```

Рисунок 3.12 - Конвертація моделі

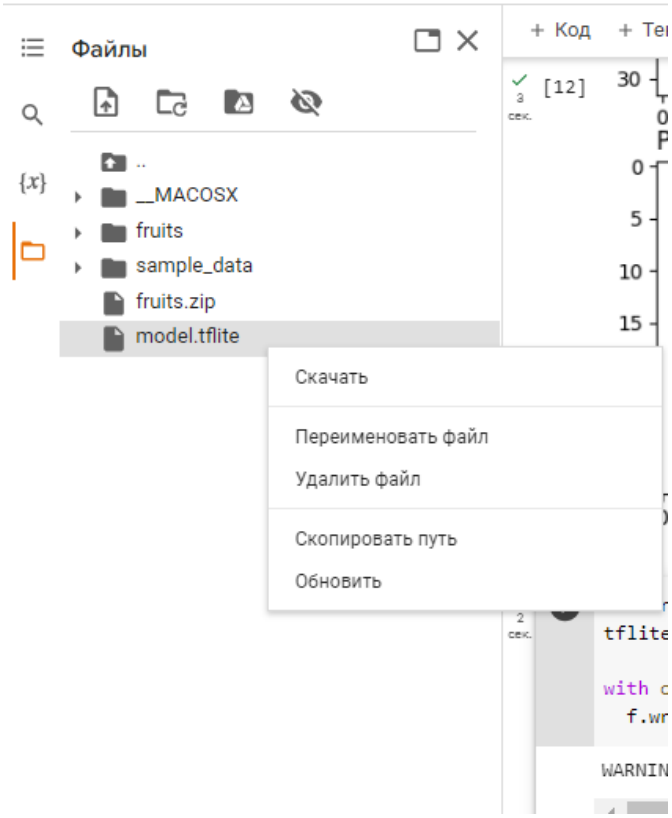


Рисунок 3.13 - Файл моделі

### 3.2 Розробка Android додатку

Основна мета Android додатку – надати зручний інтерфейс для роботи з нейронною мережею. Інтерфейс має бути простим і контрастним для того, щоб люди з вадами зору могли ним користуватися. Також має бути присутній голосовий спікер для озвучування інформації на екрані.

Головне меню містить дві кнопки (Рис. 3.14). Кожна з кнопок відкриває відповідну модель – детектор чи класифікатор. Голосовий спікер озвучує фразу для розуміння стану програми: «Main Page. If you want to classify image press the left button. If you want to recognize – press the right side».

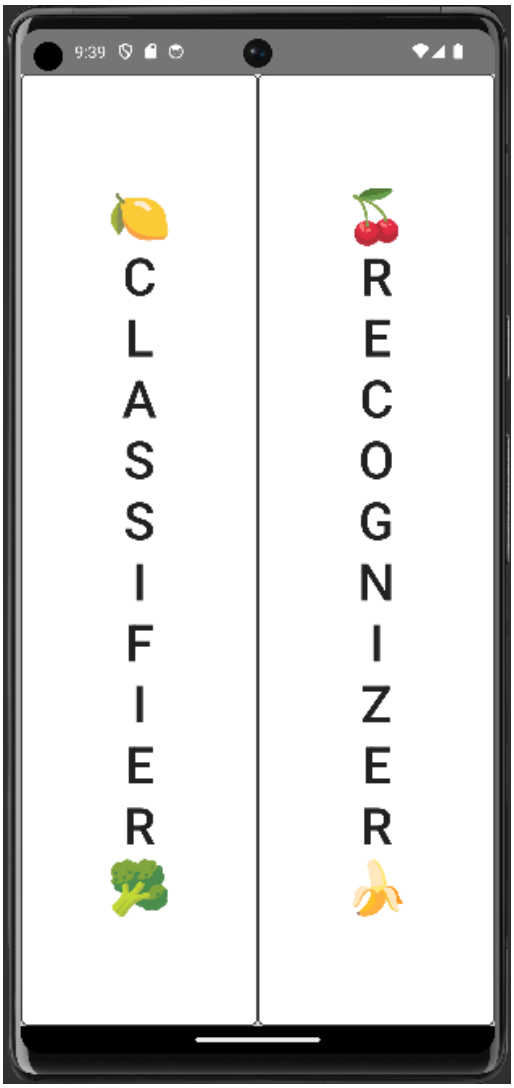


Рисунок 3.14 - Головне меню додатку

Сторінка класифікатора містить 2 кнопки (Рис. 3.15). Вони теж доволі чіткі та контрастні. Перша кнопка відповідає за відкриття камери для виконання фото. Друга кнопка відповідає за відкриття галереї. Таким чином користувач може обирати яким чином він хоче завантажити зображення в додаток. Голосовий спікер озвучує фразу для розуміння стану програми:

«You opened the classifier! If you want to take picture press the bottom-left side of screen. If you want to pick picture from gallery - press the bottom-right side.».

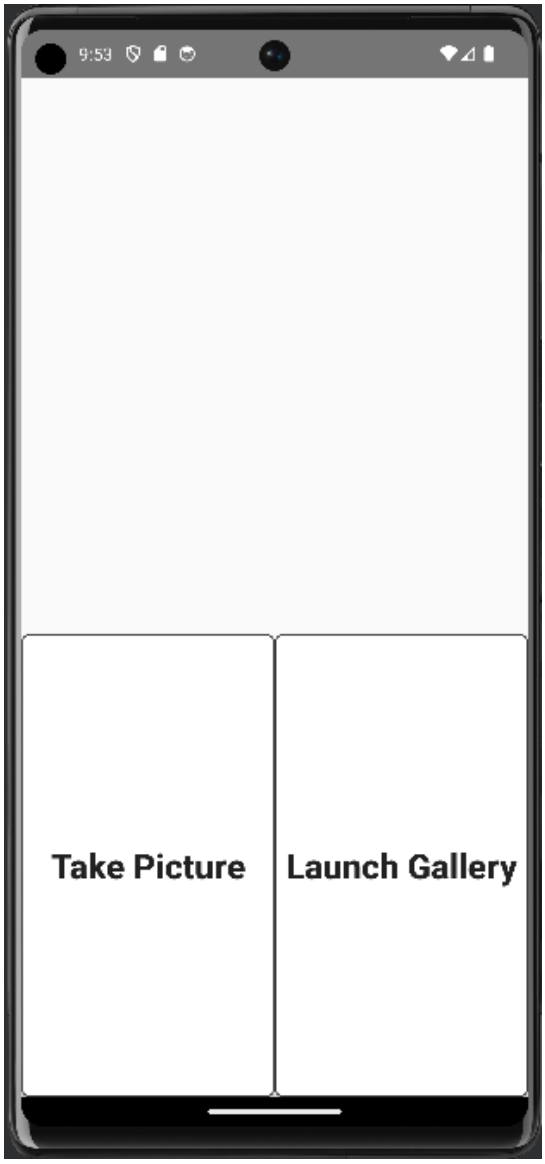


Рисунок 3.15 - Сторінка класифікатора

При першому запуску програми і натисканні на одну з кнопок додаток запросить доступ до камери або галереї (Рис. 3.16).

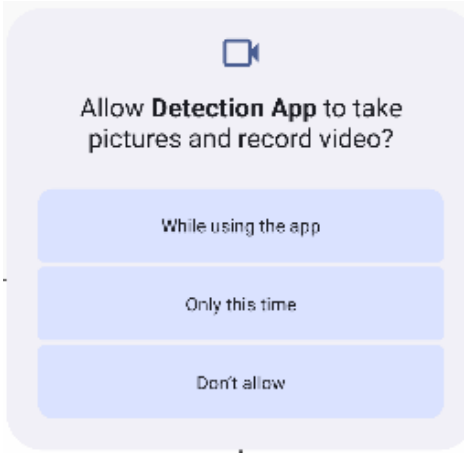


Рисунок 3.16 - Запрос дозволу на доступ

Після завантаження зображення користувач побачить його на екрані. Нижче буде виведено припущення моделі про наявний на ньому об’єкт. Також голосовий спікер озвучить припущення про об’єкт. В подальшому планується створити список імовірностей відношення об’єкта до того чи іншого класу, він буде знаходитись на цій сторінці.


Рисунок 3.17 - Екран з припущенням про клас об’єкту

Якщо на головному меню користувач натисне на кнопку «Recognizer» то він опиниться на сторінці детектора (Рис. 3.18). Користувач почує фразу: «You opened the detector! Now you can point the camera at an object for detection.». На екрані відкриється сторінка яка у реальному часі за допомогою камери буде розпізнавати об’єкти. Голосовий спікер буде озвучувати кожен об’єкт, який вдасться розпізнати.


Рисунок 3.18 - Сторінка детектора

Сторінка детектора використовує бібліотеку OpenCV для роботи з камерою в реальному часі. OpenCV (Open Source Computer Vision Library) — бібліотека комп’ютерного зору та машинного навчання з відкритим кодом. OpenCV було створено, щоб забезпечити загальну інфраструктуру для програм комп’ютерного зору та прискорити використання машинного сприйняття в комерційних продуктах.[10] Це дозволяє використовувати метод

OnCameraFrame. Під час використання камери ми постійно відправляємо кадри з неї для обробки їх моделлю. Якщо модель знаходить об’єкт на кадрі, відбувається озвучування цього результату.

# ВИСНОВКИ

Під час виконання цієї роботи було виконано дослідження предметної області. Проведено аналіз актуальності та перспективності проблеми. Визначено основні поняття що стосуються теми нейронних мереж, а також поставлено чітке завдання для вирішення.

У першому розділі виконано огляд понять машинного навчання, глибокого навчання, нейромереж. Досліджено і чітко описано задачі класифікації та розпізнавання об'єктів на зображеннях.

У другому розділі проведено дослідження методів реалізації поставленої задачі. Описано властивості операційної системи Android. Розглянуто переваги та недоліки різних мов програмування. Проведено детальний розбір бібліотеки Tensor Flow та Tensor Flow Lite, а також середовища для написання нейронної мережі – Google Colab.

На поточний момент реалізовано Android додаток здатний до класифікації та розпізнавання обмеженої кількості об’єктів. Успішно виконано розгортання моделі Tensor Flow на мобільний пристрій. Це лише тестова версія додатку, яка ще має доопрацьовуватись. Проте вже зараз можна впевнено заявити про перспективність такого додатку.

В майбутньому, перш за все необхідно створити велику базу об’єктів що підлягають розпізнаванню. Кожен клас об’єкта має містити велику кількість зображень для навчання моделі. Це дозволить зробити модель більш точною. Модель нейронної мережі теж можна підлаштувати під певні вимоги. Доопрацюванню підлягає і інтерфейс застосунку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Невмержицький Р. ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ : Наукова робота. Житомир. 1 с. URL: https://doi.org/chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf (дата звернення: 17.03.2024).
2. Bengio Y., Lecun, Y. Convolutional Networks for Images, Speech, and Time-Series. 1997. URL: https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_TimeSeries/links/0deec519dfa2325502000000.pdf
3. Женчук О. В. Згорткові нейронні мережі та їх застосування до розпізнавання дорожніх об’єктів в умовах зашумленості : Дипломна робота на здобуття ступеня бакалавра. Київ, 2019. 126 с. URL: https://doi.org/chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://ela.kpi.ua/server/api/core/bitstreams/caf0baab-794e-45ff-90a4-bb99dde9a6f5/content (дата звернення: 17.03.2024).
4. Найпопулярніші операційні системи світу 2020 • Marketer. *Marketer*. URL: https://marketer.ua/ua/stats-operating-system-2020/ (дата звернення: 24.03.2024).
5. Android Studio: переваги та особливості. *QualityAssuranceGroup*. URL: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/ (дата звернення: 24.03.2024).
6. Мови програмування для андроїд: коротке порівняння. *FoxmindEd*. URL: https://foxminded.ua/movy-prohramuvannia-dlia-android/ (дата звернення: 24.03.2024).
7. Tensorflow що це і які основні можливості застосування. *FoxmindEd*. URL: https://foxminded.ua/tensorflow-shcho-tse/ (дата звернення:

06.04.2024).

8. Що таке Tensorflow? - Оксім. *Оксім - IT допомога*. URL: https://www.oksim.ua/2023/08/07/shho-take-tensorflow/#Як_працює_Tensorflow (дата звернення: 06.04.2024).
9. Google Colaboratory. *Google Colab*. URL: https://colab.research.google.com/?hl=uk#scrollTo=ufxBm1yRnruN (дата звернення 06.04.2024).
10. About. *OpenCV*. URL: https://opencv.org/about/ (date of access: 21.04.2024).

# ДОДАТОК А

```
[ ]  !unzip fruits.zip
```

```
[ ]  import tensorflow as tf
     import matplotlib.pyplot as plt
```

```
[ ]  img_height, img_width = 32, 32
     batch_size = 20

     train_ds = tf.keras.utils.image_dataset_from_directory(
         "fruits/train",
         image_size = (img_height, img_width),
         batch_size = batch_size
     )
     val_ds = tf.keras.utils.image_dataset_from_directory(
         "fruits/validation",
         image_size = (img_height, img_width),
         batch_size = batch_size
     )
     test_ds = tf.keras.utils.image_dataset_from_directory(
         "fruits/test",
         image_size = (img_height, img_width),
         batch_size = batch_size
```

```
[ ]  class_names = ["apple", "banana", "orange"]
     plt.figure(figsize=(10,10))
     for images, labels in train_ds.take(1):
       for i in range(9):
         ax = plt.subplot(3, 3, i + 1)
         plt.imshow(images[i].numpy().astype("uint8"))
         plt.title(class_names[labels[i]])
         plt.axis("off")
```

```
[ ]  model = tf.keras.Sequential(
         [
          tf.keras.layers.Rescaling(1./255),
          tf.keras.layers.Conv2D(32, 3, activation="relu"),
          tf.keras.layers.MaxPooling2D(),
          tf.keras.layers.Conv2D(32, 3, activation="relu"),
          tf.keras.layers.MaxPooling2D(),
          tf.keras.layers.Conv2D(32, 3, activation="relu"),
          tf.keras.layers.MaxPooling2D(),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dense(128, activation="relu"),
          tf.keras.layers.Dense(3)
         ]
     )
```

```
[ ]  model.compile(
         optimizer="adam",
         loss=tf.losses.SparseCategoricalCrossentropy(from_logits = True),
         metrics=['accuracy']
     )
```

```
[ ]  model.fit(
         train_ds,
         validation_data = val_ds,
         epochs = 10
     )
```

```
[ ]  model.evaluate(test_ds)
```

```
[ ]  import numpy

     plt.figure(figsize=(10,10))
     for images, labels in test_ds.take(1):
       classifications = model(images)
       # print(classifications)

       for i in range(9):
         ax = plt.subplot(3, 3, i + 1)
         plt.imshow(images[i].numpy().astype("uint8"))
         index = numpy.argmax(classifications[i])
         plt.title("Pred: " + class_names[index] + " | Real: " + class_names[labels[i]])
```

```
[ ]  converter = tf.lite.TFLiteConverter.from_keras_model(model)
     tflite_model = converter.convert()

     with open("model.tflite", 'wb') as f:
       f.write(tflite_model)
```

# ДОДАТОК В

## MainActivity.java

```
package com.example.detectionapp;

import android.content.Intent;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

import com.example.detectionapp.Classifier.ClassifierActivity;
import com.example.detectionapp.Detector.DetectorActivity;

import java.util.Locale;

public class MainActivity extends AppCompatActivity implements
TextToSpeech.OnInitListener{

    private TextToSpeech mTextToSpeech;
    private boolean mIsInit;
    Button classify_btn, recognize_btn;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mTextToSpeech = new TextToSpeech(this, new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int i) {
                if (i == TextToSpeech.SUCCESS) {
                    Locale locale = new Locale("en");
                    int result = mTextToSpeech.setLanguage(locale);
                    if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                        mIsInit = false;
                    } else {
                        mIsInit = true;
                        String textToSpeech = "Main Page. If you want to
classify image press the left side of screen. If you want to recognize -
press the right side.";
                        mTextToSpeech.speak(textToSpeech,
TextToSpeech.QUEUE_FLUSH, null, "id1");
                    }
                } else {
                    Log.e("TextToSpeech", "Initialization failed");
                    mIsInit = false;
                }
            }
        });
```

```
classify_btn = findViewById(R.id.classify_btn);
        recognize_btn = findViewById(R.id.recognize_btn_btn);

    }

    @Override
    protected void onStart() {
        super.onStart();

        recognize_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent recognizerIntent = new Intent(MainActivity.this,
DetectorActivity.class);
                startActivity(recognizerIntent);
            }
        });
        classify_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent classifyIntent = new Intent(MainActivity.this,
ClassifierActivity.class);
                startActivity(classifyIntent);
            }
        });
    }

    @Override
    protected void onResume() {
        super.onResume();
        mTextToSpeech = new TextToSpeech(this, new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int i) {
                if (i == TextToSpeech.SUCCESS) {
                    Locale locale = new Locale("en");
                    int result = mTextToSpeech.setLanguage(locale);
                    if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                        mIsInit = false;
                    } else {
                        mIsInit = true;
                        String textToSpeech = "Main Page. If you want to
classify image press the left side of screen. If you want to recognize -
press the right side.";
                        mTextToSpeech.speak(textToSpeech,
TextToSpeech.QUEUE_FLUSH, null, "id1");
                    }
                } else {
                    Log.e("TextToSpeech", "Initialization failed");
                    mIsInit = false;
                }
            }
        });
    }

    @Override
    protected void onRestart() {
        super.onRestart();

        mTextToSpeech = new TextToSpeech(this, new
TextToSpeech.OnInitListener() {
```

```
            @Override
            public void onInit(int i) {
                if (i == TextToSpeech.SUCCESS) {
                    Locale locale = new Locale("en");
                    int result = mTextToSpeech.setLanguage(locale);
                    if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                        mIsInit = false;
                    } else {
                        mIsInit = true;
                        String textToSpeech = "Main Page. If you want to
classify image press the left side of screen. If you want to recognize -
press the right side.";
                        mTextToSpeech.speak(textToSpeech,
TextToSpeech.QUEUE_FLUSH, null, "id1");
                    }
                } else {
                    Log.e("TextToSpeech", "Initialization failed");
                    mIsInit = false;
                }
            }
        });
    }

    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            Locale locale = new Locale("en");
            int result = mTextToSpeech.setLanguage(locale);
            if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                mIsInit = false;
            } else {
                mIsInit = true;
            }
        } else {
            Log.e("TextToSpeech", "Initialization failed");
            mIsInit = false;
        }
    }
}
```

**Classifier.java**

```
ackage com.example.detectionapp.Classifier;

import android.content.Context;
import android.graphics.Bitmap;
import android.speech.tts.TextToSpeech;

import com.example.detectionapp.ml.ClassificationModel;

import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Classifier {

    private int imageSize = 32;
```

```
private String[] classes = {"Apple", "Avocado" , "Banana", "Eggplant",
"Lettuce", "Onion", "Orange", "Walnut"};
    private TensorBuffer outputFeature0;
    private String confidencesString = "";



    public String classifyImage(Bitmap image, Context context){
        try {
            ClassificationModel model =
ClassificationModel.newInstance(context);

            // Creates inputs for reference.
            TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new
int[]{1, 32, 32, 3}, DataType.FLOAT32);
            ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize *
imageSize * 3);
            byteBuffer.order(ByteOrder.nativeOrder());

            int[] intValues = new int[imageSize * imageSize];
            image.getPixels(intValues, 0, image.getWidth(), 0, 0,
image.getWidth(), image.getHeight());
            int pixel = 0;
            //iterate over each pixel and extract R, G, and B values. Add
those values individually to the byte buffer.
            for(int i = 0; i < imageSize; i ++){
                for(int j = 0; j < imageSize; j++){
                    int val = intValues[pixel++]; // RGB
                    byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 1));
                    byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 1));
                    byteBuffer.putFloat((val & 0xFF) * (1.f / 1));
                }
            }

            inputFeature0.loadBuffer(byteBuffer);

            // Runs model inference and gets result.
            ClassificationModel.Outputs outputs =
model.process(inputFeature0);
            outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

            float[] confidences = outputFeature0.getFloatArray();
            // find the index of the class with the biggest confidence.
            int maxPos = 0;
            float maxConfidence = 0;
            for (int i = 0; i < confidences.length; i++) {
                if (confidences[i] > maxConfidence) {
                    maxConfidence = confidences[i];
                    maxPos = i;
                }
            }

            Arrays.sort(confidences);

            for(int i = 0; i < 3; i++){
                confidencesString += String.format("%s: %.1f%%\n",
classes[i], confidences[i] * 100);
            }
            // Releases model resources if no longer used.
            model.close();
            return classes[maxPos].toString();
```

```
        } catch (IOException e) {
            // TODO Handle the exception
        }
    return null;
    }

//    public String getConfidencesString() {
//        return confidencesString;
//    }
}
```

**ClassifierActivity.java**

```
package com.example.detectionapp.Classifier;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.media.ThumbnailUtils;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;


import com.example.detectionapp.R;

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


public class ClassifierActivity extends AppCompatActivity implements
TextToSpeech.OnInitListener {

    private TextToSpeech mTextToSpeech;
    private boolean mIsInit;
    Button camera, gallery;
    ImageView imageView;
    TextView result, confidencesView;
    int imageSize = 32;
    Classifier classifier = new Classifier();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_classifier);

        camera = findViewById(R.id.button);
        mTextToSpeech = new TextToSpeech(this, this);
        gallery = findViewById(R.id.button2);
```

```
result = findViewById(R.id.result);
        imageView = findViewById(R.id.imageView);
//          confidencesView = findViewById(R.id.confidenceTextView);

        mTextToSpeech = new TextToSpeech(this, new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int i) {
                if (i == TextToSpeech.SUCCESS) {
                    Locale locale = new Locale("en");
                    int result = mTextToSpeech.setLanguage(locale);
                    if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                        mIsInit = false;
                    } else {
                        mIsInit = true;
                        String textToSpeech = "You opened the classifier! If
you want to take picture press the bottom-left side of screen. If you want to
pick picture from gallery - press the bottom-right side.";
                        mTextToSpeech.speak(textToSpeech,
TextToSpeech.QUEUE_FLUSH, null, "id1");
                    }
                } else {
                    Log.e("TextToSpeech", "Initialization failed");
                    mIsInit = false;
                }
            }
        });


        camera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (checkSelfPermission(Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {
                    Intent cameraIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivityForResult(cameraIntent, 3);
                } else {
                    requestPermissions(new
String[]{Manifest.permission.CAMERA}, 100);
                }
            }
        });
        gallery.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent cameraIntent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(cameraIntent, 1);
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        if(resultCode == RESULT_OK){
            if(requestCode == 3){
                Bitmap image = (Bitmap) data.getExtras().get("data");
                int dimension = Math.min(image.getWidth(),
image.getHeight());
```

```
                image = ThumbnailUtils.extractThumbnail(image, dimension,
dimension);
                imageView.setImageBitmap(image);

                image = Bitmap.createScaledBitmap(image, imageSize,
imageSize, false);
                result.setText(classifier.classifyImage(image,
getApplicationContext()));
            }else{
                Uri dat = data.getData();
                Bitmap image = null;
                try {
                    image =
MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                imageView.setImageBitmap(image);

                image = Bitmap.createScaledBitmap(image, imageSize,
imageSize, false);
                result.setText(classifier.classifyImage(image,
getApplicationContext()));
            }

//            String confidenceString = classifier.getConfidencesString();
//            confidencesView.setText(confidenceString);

            if (mIsInit) {
                String textToSpeech = (String) result.getText();
                mTextToSpeech.speak(textToSpeech, TextToSpeech.QUEUE_FLUSH,
null, "id1");
            }
        }
        super.onActivityResult(requestCode, resultCode, data);
    }

    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            Locale locale = new Locale("en");
            int result = mTextToSpeech.setLanguage(locale);
            if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                mIsInit = false;
            } else {
                mIsInit = true;
            }
        } else {
            Log.e("TextToSpeech", "Initialization failed");
            mIsInit = false;
        }
    }

}
```

**ObjectDetectorClass.java**

```
package com.example.detectionapp.Detector;

import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.graphics.Bitmap;

import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Point;
```

```
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;
import org.tensorflow.lite.Interpreter;
import org.tensorflow.lite.gpu.GpuDelegate;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.lang.reflect.Array;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public class ObjectDetectorClass {
    // should start from small letter

    // this is used to load model and predict
    private Interpreter interpreter;
    // store all label in array
    private List<String> labelList;
    private int INPUT_SIZE;
    private int PIXEL_SIZE=3; // for RGB
    private int IMAGE_MEAN=0;
    private  float IMAGE_STD=255.0f;
    // use to initialize gpu in app
    private GpuDelegate gpuDelegate;
    private int height=0;
    private  int width=0;
    private String predictedObject;
    private boolean recognized = true;

    ObjectDetectorClass(AssetManager assetManager,String modelPath,int
inputSize) throws IOException{
        INPUT_SIZE=inputSize;
        // use to define gpu or cpu // no. of threads
        Interpreter.Options options=new Interpreter.Options();
        gpuDelegate=new GpuDelegate();
        options.addDelegate(gpuDelegate);
        options.setNumThreads(4); // set it according to your phone
        // loading model
        interpreter=new
Interpreter(loadModelFile(assetManager,modelPath),options);
        // load labelmap
        labelList = new
ArrayList<>(Arrays.asList("potato","capsicum","garlic","tomato", "onion"));


    }

    private List<String> loadLabelList(AssetManager assetManager, String
labelPath) throws IOException {
        // to store label
        List<String> labelList=new ArrayList<>();
        // create a new reader
        BufferedReader reader=new BufferedReader(new
InputStreamReader(assetManager.open(labelPath)));
```

```
        String line;
        // loop through each line and store it to labelList
        while ((line=reader.readLine())!=null){
            labelList.add(line);
        }
        reader.close();
        return labelList;
    }

    private ByteBuffer loadModelFile(AssetManager assetManager, String
modelPath) throws IOException {
        // use to get description of file
        AssetFileDescriptor fileDescriptor=assetManager.openFd(modelPath);
        FileInputStream inputStream=new
FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel=inputStream.getChannel();
        long startOffset =fileDescriptor.getStartOffset();
        long declaredLength=fileDescriptor.getDeclaredLength();

        return
fileChannel.map(FileChannel.MapMode.READ_ONLY,startOffset,declaredLength);
    }
    // create new Mat function
    public Mat recognizeImage(Mat mat_image){
        // Rotate original image by 90 degree get get portrait frame

        // This change was done in video: Does Your App Keep Crashing? |
Watch This Video For Solution.
        // This will fix crashing problem of the app

        Mat rotated_mat_image=new Mat();

        Mat a=mat_image.t();
        Core.flip(a,rotated_mat_image,1);
        // Release mat
        a.release();

        // if you do not do this process you will get improper prediction,
less no. of object
        // now convert it to bitmap
        Bitmap bitmap=null;

bitmap=Bitmap.createBitmap(rotated_mat_image.cols(),rotated_mat_image.rows(),
Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(rotated_mat_image,bitmap);
        // define height and width
        height=bitmap.getHeight();
        width=bitmap.getWidth();

        // scale the bitmap to input size of model
        Bitmap
scaledBitmap=Bitmap.createScaledBitmap(bitmap,INPUT_SIZE,INPUT_SIZE,false);

        // convert bitmap to bytebuffer as model input should be in it
        ByteBuffer byteBuffer=convertBitmapToByteBuffer(scaledBitmap);

        // defining output
        // 10: top 10 object detected
        // 4: there coordinate in image
        //  float[][][]result=new float[1][10][4];
        Object[] input=new Object[1];
        input[0]=byteBuffer;
```

```
        Map<Integer,Object> output_map=new TreeMap<>();
        // we are not going to use this method of output
        // instead we create treemap of three array (boxes,score,classes)

        float[][][]boxes =new float[1][10][4];
        // 10: top 10 object detected
        // 4: there coordinate in image
        float[][] scores=new float[1][10];
        // stores scores of 10 object
        float[][] classes=new float[1][10];
        // stores class of object

        // add it to object_map;
        output_map.put(0,boxes);
        output_map.put(1,classes);
        output_map.put(2,scores);

        // now predict
        interpreter.runForMultipleInputsOutputs(input,output_map);
        // Before watching this video please watch my previous 2 video of
        //      1. Loading tensorflow lite model
        //      2. Predicting object

        Object value=output_map.get(0);
        Object Object_class=output_map.get(1);
        Object score=output_map.get(2);

        // loop through each object
        // as output has only 10 boxes
        for (int i=0;i<10;i++){
            float class_value=(float) Array.get(Array.get(Object_class,0),i);
            float score_value=(float) Array.get(Array.get(score,0),i);
            // define threshold for score

            recognized = false;

            // Here you can change threshold according to your model
            if(score_value>0.88){
                Object box1=Array.get(Array.get(value,0),i);
                // we are multiplying it with Original height and width of
frame

                float top=(float) Array.get(box1,0)*height;
                float left=(float) Array.get(box1,1)*width;
                float bottom=(float) Array.get(box1,2)*height;
                float right=(float) Array.get(box1,3)*width;
                // draw rectangle in Original frame //  starting point    //
ending point of box  // color of box       thickness
                Imgproc.rectangle(rotated_mat_image,new Point(left,top),new
Point(right,bottom),new Scalar(0, 255, 0, 255),2);
                // write text on frame
                // string of class name of object  // starting point
// color of text           // size of text
                predictedObject = labelList.get((int) class_value);
                Imgproc.putText(rotated_mat_image,predictedObject,new
Point(left,top),3,1,new Scalar(255, 0, 0, 255),2);
                recognized = true;
                break;
            }
            if (!recognized) {
                // Set recognized to false if no object is detected with
```

```
                recognized = false;
            }

        }
        // select device and run

        // before returning rotate back by -90 degree

        // Do same here
        Mat b=rotated_mat_image.t();
        Core.flip(b,mat_image,0);
        b.release();
        // Now for second change go to CameraBridgeViewBase
        return mat_image;
    }

    private ByteBuffer convertBitmapToByteBuffer(Bitmap bitmap) {
        ByteBuffer byteBuffer;
        // some model input should be quant=0  for some quant=1
        // for this quant=0
        // Change quant=1
        // As we are scaling image from 0-255 to 0-1
        int quant=1;
        int size_images=INPUT_SIZE;
        if(quant==0){

byteBuffer=ByteBuffer.allocateDirect(1*size_images*size_images*3);
        }
        else {

byteBuffer=ByteBuffer.allocateDirect(4*1*size_images*size_images*3);
        }
        byteBuffer.order(ByteOrder.nativeOrder());
        int[] intValues=new int[size_images*size_images];

bitmap.getPixels(intValues,0,bitmap.getWidth(),0,0,bitmap.getWidth(),bitmap.g
etHeight());
        int pixel=0;

        // some error
        //now run
        for (int i=0;i<size_images;++i){
            for (int j=0;j<size_images;++j){
                final  int val=intValues[pixel++];
                if(quant==0){
                    byteBuffer.put((byte) ((val>>16)&0xFF));
                    byteBuffer.put((byte) ((val>>8)&0xFF));
                    byteBuffer.put((byte) (val&0xFF));
                }
                else {
                    // paste this
                    byteBuffer.putFloat((((val >> 16) & 0xFF))/255.0f);
                    byteBuffer.putFloat((((val >> 8) & 0xFF))/255.0f);
                    byteBuffer.putFloat((((val) & 0xFF))/255.0f);
                }
            }
        }
        return byteBuffer;
    }

    public String getPredictedObject() {
```

```
        return predictedObject;
    }

    public boolean isRecognized() {
        return recognized;
    }
}
```

**DetectorActivity.java**

```
package com.example.detectionapp.Detector;


import androidx.annotation.Nullable;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.util.Log;

import com.example.detectionapp.R;

import org.opencv.android.CameraActivity;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;

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

public class DetectorActivity extends CameraActivity implements
TextToSpeech.OnInitListener{

    private Mat mRgba;
    private Mat mGray;
    private CameraBridgeViewBase cameraBridgeViewBase;
    private ObjectDetectorClass objectDetectorClass;
    private TextToSpeech mTextToSpeech;
    private boolean mIsInit;
    private int fpsCounter=0;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detector);

        getPermission();

        cameraBridgeViewBase = findViewById(R.id.cameraView);

        mTextToSpeech = new TextToSpeech(this, new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int i) {
                if (i == TextToSpeech.SUCCESS) {
                    Locale locale = new Locale("en");
                    int result = mTextToSpeech.setLanguage(locale);
                    if (result == TextToSpeech.LANG_MISSING_DATA || result ==
```

```
TextToSpeech.LANG_NOT_SUPPORTED) {
                        mIsInit = false;
                    } else {
                        mIsInit = true;
                        String textToSpeech = "You opened the detector! Now
you can point the camera at an object for detection.";
                        mTextToSpeech.speak(textToSpeech,
TextToSpeech.QUEUE_FLUSH, null, "id1");
                    }
                } else {
                    Log.e("TextToSpeech", "Initialization failed");
                    mIsInit = false;
                }
            }
        });

        cameraBridgeViewBase.setCvCameraViewListener(new
CameraBridgeViewBase.CvCameraViewListener2() {
            @Override
            public void onCameraViewStarted(int width, int height) {
                mRgba=new Mat(height,width, CvType.CV_8UC4);
                mGray =new Mat(height,width,CvType.CV_8UC1);
            }

            @Override
            public void onCameraViewStopped() {
                mRgba.release();
            }

            @Override
            public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
                mRgba=inputFrame.rgba();
                mGray=inputFrame.gray();

                // now call that function
                Mat out = new Mat();
                out=objectDetectorClass.recognizeImage(mRgba);
                fpsCounter++;

                if ((!mTextToSpeech.isSpeaking()) &&
(objectDetectorClass.isRecognized())){

mTextToSpeech.speak(objectDetectorClass.getPredictedObject(),
TextToSpeech.QUEUE_FLUSH, null, "id1");
                    fpsCounter=0;
                }

                return out;
            }
        });
        if(OpenCVLoader.initDebug()){
            cameraBridgeViewBase.enableView();
        }

        try {
            objectDetectorClass = new ObjectDetectorClass(getAssets(),
"Recognition_model.tflite", 320);
            Log.d("MainActivity", "Model successfully loaded");
        } catch (IOException e) {
            Log.d("MainActivity", "Model getting some error");
            e.printStackTrace();
        }
```

```
    }

    @Override
    protected List<? extends CameraBridgeViewBase> getCameraViewList() {
        return Collections.singletonList(cameraBridgeViewBase);
    }

    void getPermission(){
        if(checkSelfPermission(Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED){
            requestPermissions(new String[]{Manifest.permission.CAMERA},
101);
        }
    }
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            Locale locale = new Locale("en");
            int result = mTextToSpeech.setLanguage(locale);
            if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                mIsInit = false;
            } else {
                mIsInit = true;
            }
        } else {
            Log.e("TextToSpeech", "Initialization failed");
            mIsInit = false;
        }
    }
}
```