Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

РОЗРОБКА МОБІЛЬНОГО ANDROID ЗАСТОСУНКУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ МАШИННОГО НАВЧАННЯ (ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ)

Текстова частина до кваліфікаційної роботи

за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи

с.в. Борозенний С.О.

(прізвище та ініціали)

_____

(підпис)

“____” _____ 2023 р.

Виконала студентка

Бойко Х.А.

(прізвище та ініціали)

“____” _____ 2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

доцент, к.ф-м.н.

_____О. П. Жежерун (підпис)

„____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Бойко Христині Анатоліївні факультету інформатики 4-го курсу

ТЕМА Розробка мобільного Android застосунку з використанням технології машинного навчання (згорткових нейронних мереж)

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Вступ

1 Огляд предметної області

2 Огляд наявних Android застосунків

3 Розробка власного Android застосунку

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі „___” _____ 2023 р. Борозенний О.С. _____ (підпис)

Завдання отримав _____ (підпис)

Тема: Розробка мобільного Android застосунку з використанням технології машинного навчання (згорткових нейронних мереж)

Календарний план виконання роботи:

№ п/п	Назва етапу кваліфікаційної роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на кваліфікаційну роботу	жовтень 2022	
2.	Аналіз матеріалів за темою	24.01.2023	
3.	Розробка Android застосунку, створення моделей згорткових нейронних мереж	15.04.2023	
4.	Написання текстової частини до кваліфікаційної роботи	30.04.2023	
5.	Створення презентації для доповіді	09.05.2023	
6.	Коригування виконаної роботи	17.05.2023	
7.	Остаточне оформлення роботи	24.05.2023	
8.	Захист кваліфікаційної роботи	31.05.2023	

Бойко Х. А. _____

Борозенний О. С. _____

“_____”_____

Зміст

Анотація

Робота присвячена розробці Android застосунку з використанням згорткових нейронних мереж для розпізнавання й класифікації видів квітів.

Для реалізації задачі було використано набір даних «flowers»[1] на платформі Kaggle. Моделі зі структурою згорткових нейронних мереж були створені та навчені на мові Python на платформі Google Colab. Моделі з метаданими були збережені у форматі TensorFlow Lite для використання в Android застосунку. Мобільний застосунок був розроблений на мові Java в інтегрованому середовищі розробки Android Studio. Результати моделей були проаналізовані через такі метрики: точність, влучність, повнота, F-міра. Усі моделі були додані до Android застосунку, тому кожну з них можна випробувати на власних зображеннях.

Ключові слова: машинне навчання, згорткова нейронна мережа, Android застосунок, класифікація зображень.

Вступ

Актуальність теми

У 2023 році 3,3 мільярда людей користуються мобільними пристроями з операційною системою Android, яка займає 71,8% мобільних операційних систем на світовому ринку[2]. Розвиток мобільних пристроїв дав можливість використовувати в застосунках технології машинного навчання, що вирішують різноманітні непрості задачі, такі як: класифікація зображень, розпізнавання мови, передбачення цін, виявлення спаму, прогнозування продажів, надання персоналізованих рекомендацій тощо. Через аналіз великої кількості даних машинне навчання знаходить закономірності та передбачає результати для нових даних. Нині цю технологію можна використовувати в застосунках безпосередньо на пристроях, а не віддалено на хмарних сервісах завдяки вдосконаленню їхнього апаратного забезпечення, обчислювальної потужності, об'єму пам'яті. Обробка вхідних даних на пристрої, без пересилань на сервер, забезпечує конфіденційність персональних даних, зменшує час виконання операцій, адже немає затримок як при передачі даних через мережу. До того ж такий застосунок є дешевшим в розробці, бо не вимагає використання хмарних обчислень, та може використовуватися в офлайн режимі. Через обмеження обсягу пам'яті, обчислювальної потужності мобільних пристроїв моделі машинного навчання повинні бути невеликі за розміром. Якщо для вирішення складніших задач необхідні більші та потужніші моделі, то їх слід переносити на сервер і використовувати віддалено [3].

Поставлена задача полягає у використанні згорткових нейронних мереж для класифікації зображень. Завдяки багатошаровій структурі взаємопов'язаних нейронів згорткові нейронні мережі здатні розпізнавати різні елементи на зображеннях: лінії, краї, форми[4]. Оскільки на кожному шарі застосовуються спільні вагові коефіцієнти для всіх областей зображення, а згорткові шари не є повнозв'язаними, тому модель з меншою кількістю параметрів залишається за структурою глибокою та з меншою кількістю зв'язків уникає перенавчання.

Згортковим нейронним мережам властива трансляційна інваріантність (translation invariance), тому вони розпізнають об’єкти незалежно від їхнього положення на зображенні. Для створення згорткових нейронних мереж можна використовувати передавальне навчання (transfer learning), яке полягає в застосуванні для розв’язання задач попередньо навчених моделей, які частково змінюються під нові завдання[5].

У сучасних мобільних застосунках використовуються згорткові нейронні мережі. Наприклад, Google Lens розпізнає зображення, ідентифікує об’єкти та надає інформацію про них за допомогою згорткових нейронних мереж [6]. Застосунок FaceApp, використовуючи згорткові нейронні мережі, аналізує риси обличчя і за допомогою фільтрів надає можливість змінювати їх [7].

Тому розробка Android застосунку, що використовує згорткові нейронні мережі для класифікації зображень, є вельми актуальною задачею.

Мета та завдання кваліфікаційної роботи

Мета: Розробити мобільний Android застосунок для розпізнавання й класифікації квітів з використанням згорткових нейронних мереж. З’ясувати, як створювати моделі зі структурою згорткових нейронних мереж, як їх покращувати, і також як створювати мобільний Android застосунок з використанням власних моделей.

Завдання: Розробити Android застосунок, який допоможе розпізнавати й визначати вид квітів на зображенні. Реалізувати класифікацію зображень за допомогою навчених моделей зі структурою згорткової нейронної мережі. Проаналізувати результати моделей за метриками. З’ясувати, як використовувати власні розроблені моделі машинного навчання в мобільному Android застосунку.

Об’єкт дослідження

Створення Android застосунку, використання згорткових нейронних мереж для класифікації зображень, використання натренованих моделей в Android застосунку.

Предмет дослідження

Розробка мобільного Android застосунку для розпізнавання та класифікації зображень квітів з використанням згорткових нейронних мереж. Дослідження різних моделей і аналіз їхніх результатів класифікації.

Методи дослідження

Використання технології машинного навчання – згорткових нейронних мереж.

Джерела дослідження

Було досліджено електронні ресурси з інформацією про згорткові нейронні мережі, створення Android застосунків, використання навчених моделей в мобільних застосунках. Було розглянуто роботу подібних мобільних застосунків, в яких використовується класифікація зображень.

Наукова новизна одержаних результатів

У результаті роботи було розроблено мобільний Android застосунок для класифікації видів квітів. Для вирішення задачі було створено власну згорткову нейронну мережу та дві моделі на основі попередньо навчених MobileNetV2 і EfficientNet-Lite0. Отримані результати показали, що модель, побудована на MobileNetV2, є найкращою серед інших через найменший розмір – 2,7 Мб та високу точність класифікації – 94,3%.

Практичне значення одержаних результатів

Застосунок буде корисним для тих, хто хоче визначити вид квітів на зображенні за допомогою смартфона (фотографія з галереї або з камери смартфона), використовуючи різні моделі.

Постановка задачі

1. Створити моделі зі структурою згорткових нейронних мереж для класифікації зображень квітів.
2. Навчити моделі на тренувальних даних і протестувати на тестових даних.

3. Проаналізувати результати моделей за допомогою метрик.
4. Розробити мобільний Android застосунок з можливістю використовувати власні моделі для визначення й класифікації зображення з галереї та камери смартфона
5. Додати власні моделі в Android застосунок
6. Протестувати роботу застосунку в емуляторі та на мобільному пристрої

Мобільний Android застосунок має надавати можливість розпізнати й визначати вид квітів на зображенні з використанням різних навчених моделей. Користувач повинен мати можливість обирати зображення з галереї або робити фото за допомогою камери смартфона. Для створення моделей слід використовувати згорткові нейронні мережі, деякі з них будувати на основі попередньо навчених моделей. Тренувальні і тестові дані для всіх моделей мають бути однаковими.

Структура роботи

Кваліфікаційна робота містить анотацію, вступ, три розділи, висновки, список використаних джерел.

У першому розділі детально розглянуто технологію машинного навчання – згорткові нейронні мережі, Android платформу, особливості написання Android застосунків з використанням власних навчених моделей. Пояснено, яким чином ця технологія допоможе вирішити проблему проєкту.

Другий розділ присвячений вивченню готових рішень: Android застосунків, які використовують згорткові нейронні мережі для класифікації зображень.

У третьому розділі описано практичну частину роботи. Пояснено вибір технології, структури моделей та використаних інструментів. Розглянуто процес розробки проєкту зі вставками коду. Представлено принцип роботи застосунку.

Основна частина

Розділ 1. Огляд предметної області

1.1 Огляд технологій машинного навчання: згорткові нейронні мережі

Машинне навчання – це підклас штучного інтелекту, який навчається на наборі даних, покращуючи продуктивність, і передбачає результати без явного програмування[8]. Залежно від поставленої задачі необхідно знайти або самостійно зібрати достатньо великий набір даних, щоб машині було легше знайти закономірності й дати точніші відповіді[9]. Усі дані поділяються на тренувальні та тестові. Математичним представленням підсумку навчання на тренувальних даних є модель, яка здатна прогнозувати результати для нових тестових даних[10].

Машинне навчання поділяється на такі типи: навчання з вчителем, навчання без вчителя, напівконтрольоване навчання та навчання з підкріпленням. Навчання з вчителем особливе тим, що модель тренується на промаркованих даних, для яких відомі очікувані результати. У навчанні без вчителя підказок немає, і модель самостійно визначає зв’язки між даними на основі характеристик. Напівконтрольоване навчання поєднує в собі попередні два типи: спочатку вивчається певна кількість промаркованих даних, а далі – нові дані без міток. У навчанні з підкріпленням встановлюються правила для досягнення мети та як підказки надаються винагороди й покарання[8].

Нині машинне навчання використовується для розв’язання різних задач: прогнозування поведінки користувачів, підбору рекомендацій, розпізнавання об’єктів, медичної діагностики, виявлення шахрайства тощо. Застосування цієї технології дозволяє автоматизувати й оптимізувати вирішення завдань[11].

Класифікація належить до навчання з вчителем і полягає в поділі даних на категорії. Щоб визначити, до якого класу належать вхідні дані, навчена модель

обчислює оцінку ймовірності (probability score). Категорія, яка отримала найвищу ймовірність, є прогнозованим результатом. За допомогою цього підходу можна класифікувати документи, електронні листи, виявляти спам, діагностувати хвороби, розпізнавати голос тощо[12]. Визначення, до якої категорії з багатьох належить зображення, є багатокласовою класифікацією.

Штучні нейронні мережі – це одна з технологій машинного навчання, створення якої було натхненне структурою й роботою мозку людини. Вони складаються з штучних нейронів – вузлів, що пов'язані між собою[13]. На рисунку 1 зображено структуру штучного нейрона.

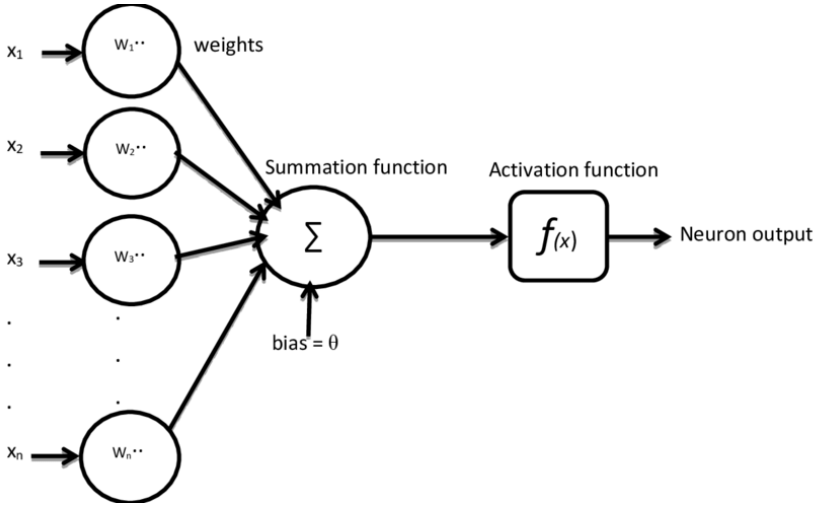

Рисунок 1. Структура штучного нейрона [31]

Вхідні сигнали $x_1,\ldots x_n$ є виходами нейронів з попереднього шару. Вагові коефіцієнти $w_1,\ldots w_n$ визначають силу зв'язку між нейронами. Спершу вони підбираються випадковим чином, проте в процесі навчання, який має певну кількість епох (ітерацій), підлаштовуються для зменшення похибок і передбачення кращих результатів. Для розрахунку вихідного сигналу застосовується функція активації, яка приймає на вхід зважену суму вхідних сигналів, помножених на відповідні вагові коефіцієнти, з додаванням зміщення b за формулою:

$$\sum_{i=1}^{n} w_i * x_i + b$$

Функція активації визначає, чи штучний нейрон буде активованим, і додає нелінійності в нейронну мережу. Для багатокласової класифікації зазвичай на останньому шарі нейронів застосовується сигмоїдна функція softmax, яка обчислює ймовірності для кожної категорії[14].

Дані з навчального набору мають відповідати формату, що приймає модель, тобто бути нормалізованими. Наприклад, якщо вхідними даними є зображення, то їх потрібно звести до однакового розміру, перетворити значення пікселів у числа в межах від 0 до 1.

Нейронні мережі можуть складатися з багатьох шарів: вхідного, вихідного та прихованих (див. нижче на рисунку 2), у яких виявляються закономірності в даних[13]. Кожен шар отримує дані від попереднього шару, обробляє їх і передає далі. Для коригування вагових коефіцієнтів застосовується зворотне поширення помилки, яке полягає зменшенні функції втрат – різниці між вихідним результатом та очікуваним[15]. Під час навчання штучної нейронної мережі ваги оновлюється, а точність моделі з кожною епохою підвищується.

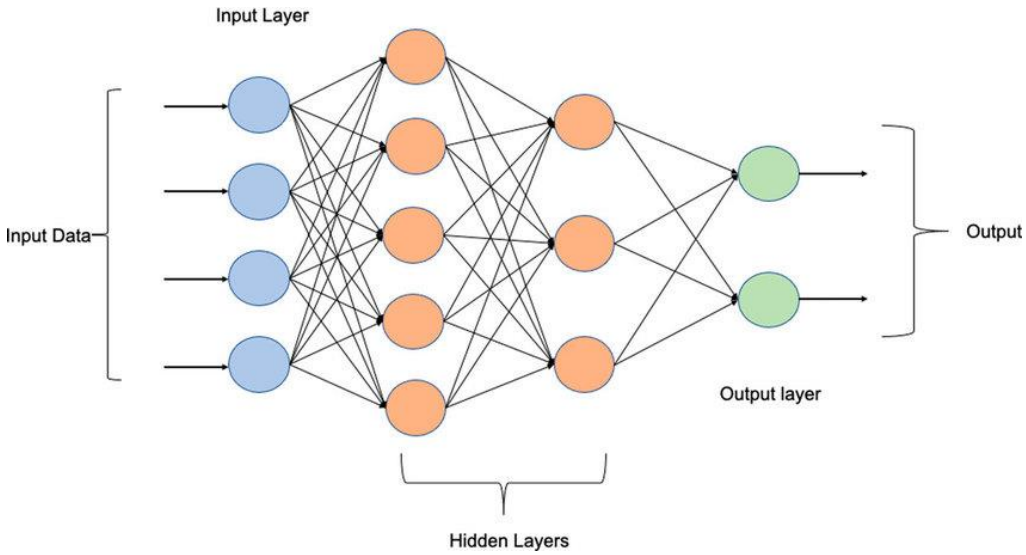

Рисунок 2. Структура багатошарової нейронної мережі [32]

Згорткові нейронні мережі є видом глибинних штучних нейронних мереж, створення яких було натхненне будовою зорової кори людини. Тому нейрони обробляють сигнали в рецептивних полях, які перекриваються частково, покриваючи усю область вхідних даних. Ці мережі використовуються для

розпізнавання й класифікації зображень, обробки природньої мови, в рекомендаційних системах тощо.

Згорткові нейронні мережі складаються із згорткових (convolutional), агрегувальних (pooling) та повнозв'язних (fully connected) шарів (див. нижче на рисунку 3). Мережа з більшою кількістю шарів є складнішою, проте більш здатною розпізнавати об'єкти[16].

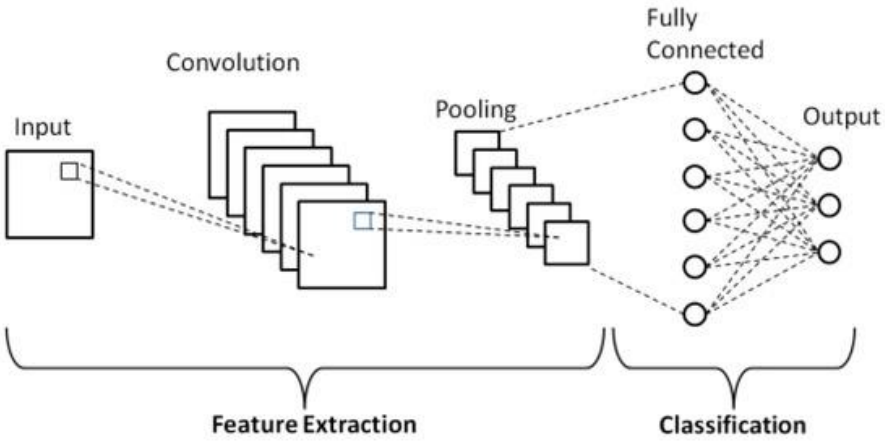

Рисунок 3. Структура згорткової нейронної мережі [33]

Для розпізнавання елементів зображення в згортковому шарі застосовуються фільтри (ядра), що є двовимірними масивами вагових коефіцієнтів. Переміщуючись по зображенню, фільтр сканує кожну область й обчислює скалярний добуток між пікселями. Результатом операції згортки є карта ознак (feature map), до якої зазвичай застосовується функція активації ReLu. За згортковими йдуть агрегувальні шари, які разом з першими утворюють згортковий блок (convolutional block), що дозволяє вивчати ознаки та отримувати високу точність в класифікації зображень. Агрегувальний шар зменшує розмір даних, використовуючи функцію агрегації до значень рецептивного поля. Наприклад, обчислюється середнє значення поля (усереднювальне агрегування) або обирається піксель з максимальним значенням з області (максимізаційне агрегування). Завдяки цьому знижується складність мережі й ризик перенавчання (overfitting) і також підвищується точність моделі, незважаючи на втрату частини даних. Останні повнозв'язні шари, використовуючи розпізнані ознаки на попередніх шарах, класифікують зображення [16]. Вихідний шар має таку саму

кількість нейронів, скільки є категорій для класифікації. Тому ознаки певного класу пов’язані з відповідною міткою. У результаті аналізу зображення на кожну категорію модель повертає оцінку ймовірності, значення якої знаходиться в діапазоні між 0 та 1.

Згорткові нейронні мережі не потребують багато попередньої обробки даних в порівнянні з іншими моделями класифікації. Вони складаються з багатьох шарів взаємозв’язаних нейронів, що дозволяє їм знаходити закономірності в зображеннях. Згорткові шари не зв’язані повністю та мають спільні параметри для локальних рецептивних полів, що знижує складність моделі та запобігає її перенавчанню. Важливою характеристикою є трансляційна інваріантність – мережі здатні розпізнавати об’єкт незалежно від його розташування. Завдяки цим особливостям згорткові мережі здатні ефективно класифікувати зображення.

Для класифікації зображень можна створювати власні моделі або використовувати моделі, що були попередньо натреновані на наборі даних. Зображення мають такі універсальні характеристики, як лінії, краї, форми. Передавальне навчання (transfer learning) полягає в тому, щоб використати згорткові шари готових моделей, які навчені виявляти ці прості елементи зображення, і змінити їхні кілька останніх шарів, які відповідають за класифікацію[17].

З 2010 по 2017 рік на наборі даних ImageNet, який містить більше 14 мільйонів зображень[18], проводилися змагання ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Завдяки цьому конкурсу відбувся значний прогрес в глибокому навчанні, були розроблені моделі з високою точністю для виявлення об’єктів та класифікації зображень: AlexNet, VGGNet, ResNet, GoogleNet, MobileNet, EfficientNet та інші[19].

Розгляньмо попередньо навчені моделі MobileNet та EfficientNet-Lite, які були розроблені спеціально для використання в мобільних застосунках. Це глибокі згорткові нейронні мережі, що є невеликими за розміром (до 18 Мб)[20] і мають досить високу точність (від 70%)[21] на наборі даних ImageNet.

Нині існує три основні версії MobileNet. MobileNetV1 має точність 70,9% на наборі даних ImageNet[21]. Модель MobileNetV2 є меншою за розміром, швидшою й точнішою за MobileNetV1. Її розмір до 14 Мб, а точність складає 74,7%[22]. MobileNetV3 є вдосконаленою версією MobileNetV2, більшою за розміром (21,11 Мб)[23], але й більш точною (79,0%)[22].

Враховуючи те, що в проєкті модель повинна використовуватися в мобільному застосунку, було обрано модель MobileNetV2, яка має вищу точність, ніж MobileNetV1 і є меншою за розміром, ніж MobileNetV3. Особливістю її структури є згорткові шари з фокусом на глибину (depthwise) й з фокусом на точки (pointwise). Перші згортки використовують 1 фільтр незалежно до кожного каналу на вході. Далі розташовані згортки з фокусом на точки, що зі згортками розміром 1x1 поєднують вихідні канали згорток з фокусом на глибину. Така структура (наведена на рисунку 4) дозволяє виявляти важливі елементи вхідних даних за нижчого обсягу параметрів й обчислень[16].

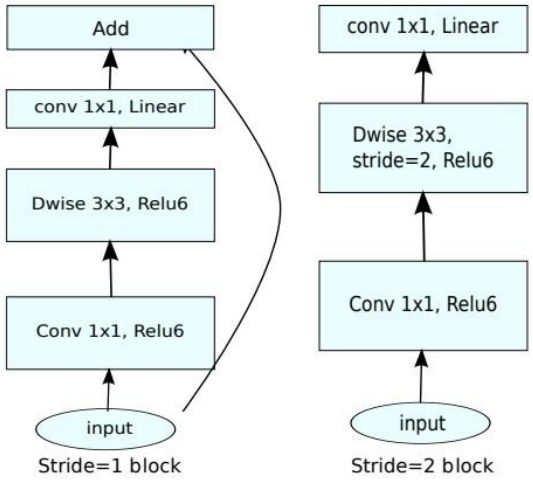

Рисунок 4. Структура моделі MobileNetV2 [34]

EfficientNet-Lite – це група моделей, які були створені для роботи на мобільному CPU, GPU й EdgeTPU. Існує 5 версій цих моделей, кожна наступна з яких є складнішою, важчою, проте дає вищу точність на ImageNet (від 75.1% до 80%)[20]. EfficientNet моделі особливі тим, що використовують складений коефіцієнт, щоб рівномірно масштабувати глибину, ширину, роздільну здатність мережі за

допомогою фіксованих коефіцієнтів. Масштабування пояснюється тим, що для обробки більшого зображення необхідно більше шарів і каналів, щоб розпізнати різні елементи. EfficientNet-Lite моделі застосовують функцію активації ReLU6 і не містять блоків SAE (squeeze-and-excitation blocks)[24].

Оскільки в пріоритеті є моделі з меншим розміром для використання в мобільному застосунку, тому було обрано модель EfficientNet-Lite0, що є найменшою за розміром серед EfficientNet моделей з точністю 75.1%[25].

На графіках нижче показано точність, затримку, розмір різних моделей EfficientNet-Lite та MobileNetV2[20].

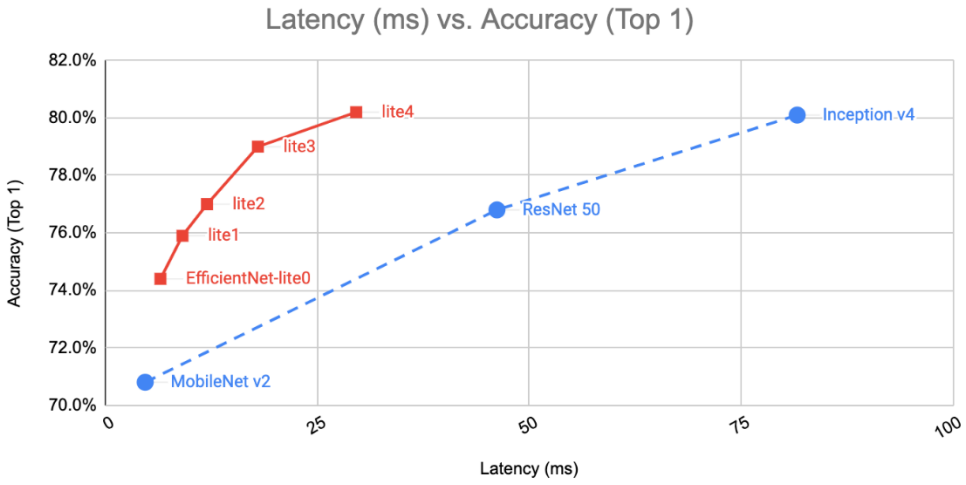

Рисунок 5. Графік затримки й точності моделей [35]

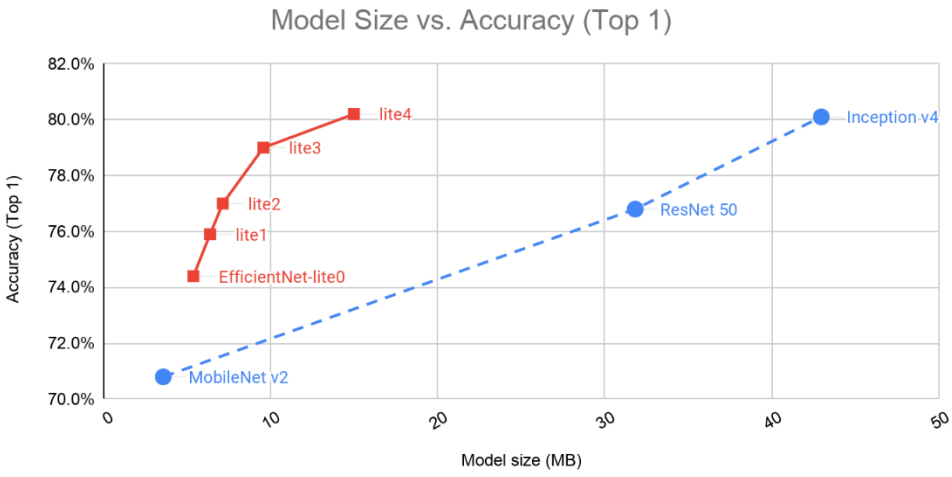

Рисунок 6. Графік розміру й точності моделей [36]

Особливості моделей MobileNet та EfficientNet-Lite надають їм перевагу у використанні в мобільних застосунках з обмеженою пам’яттю, обчислювальною потужністю. Отже, як було пояснено раніше, для проєкту були обрані дві моделі EfficientNet-Lite0 та MobileNetV2.

Щоб оцінити ефективність моделі для класифікації зображень, необхідно проаналізувати її за такими метриками, як точність, влучність, повнота, F-міра, і розглянути матриці невідповідностей (confusion matrix) на тестових даних. Усі результати моделі можна розділити на чотири категорії: істинно позитивні (True Positive), істинно негативні (True Negative), хибно позитивні (False Positive), хибно негативні (False Negative). Істинно позитивні та істинно негативні відповіді є правильними передбаченнями моделі. Інші результати – некоректні.

Точність визначається як відношення кількості правильних передбачень до загальної кількості передбачень.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Влучність є пропорцією істинно позитивних передбачень до всіх визначених як позитивні (істинно позитивні та хибно позитивні).

$$Precision = \frac{TP}{TP + FP}$$

Повнота визначає, скільки було знайдено істинно позитивних відповідей серед всіх справді релевантних відповідей (істинно позитивних та хибно негативних).

$$Recall = \frac{TP}{TP + FN}$$

F-міра є середнім гармонійним попередніх двох метрик - влучності й повноти, обчислюється за формулою:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Матриця невідповідностей показує, наскільки результати моделі на тестових даних збігаються з очікуваними (див. приклад на рисунку 8)[26]. Кожен стовпчик представляє прогнозований клас, кожен рядок – правильний клас. За допомогою матриці можна проаналізувати результати моделі для кожної класу.

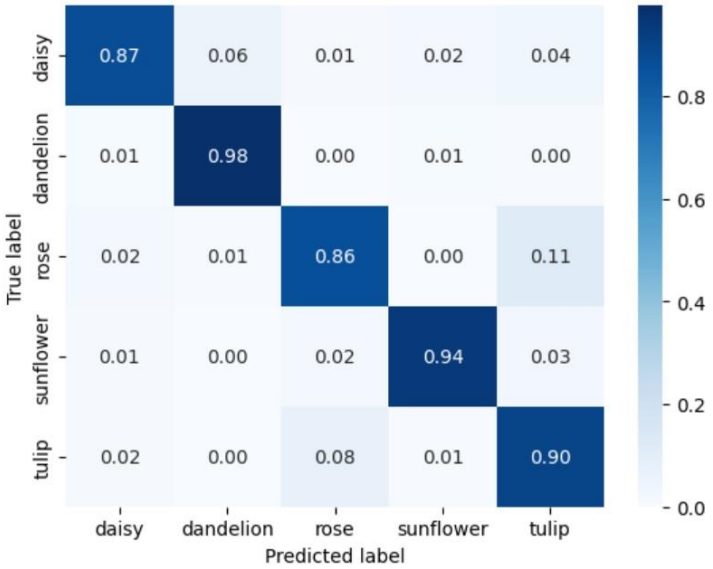

Рисунок 7. Матриця невідповідностей

Для обчислення метричних даних зручно користуватися бібліотекою sklearn.metrics, яка має методи accuracy_score(), precision_score(), recall_score(), f1_score(), confusion_matrix(). Матрицю невідповідностей, графіки точності моделі можна побудувати за допомогою бібліотеки matplotlib.pyplot.

Існують різні способи для покращення створених моделей. Наприклад, можна налаштовувати параметри: кількість епох, фільтрів, вузлів у шарах, розмір фільтрів тощо. Також можна підбирати оптимізатор для компіляції моделі, коефіцієнт швидкості навчання, функцію активації. Для кращого розпізнавання об’єктів варто зробити модель глибшою, додавши більше шарів до її структури. Якщо даних недостатньо, то можна штучно розширити їх (image augmentation), згенерувавши нові зображення за допомогою обертання, зсуву, масштабування. Адже чим більше даних для навчання, тим точнішою є модель.

Під час розробки моделі може виникнути проблема з перенавчанням (overfitting) або недонавчанням (underfitting). «Перенавчена» модель не здатна

узагальнювати, занадто прив’язана до навчальних даних, тому на них показує високу точність і дає невідповідні результати на тестових даних. Щоб уникнути перенавчання, можна зменшити кількість ітерацій навчання, зменшити кількість зв’язків з додаванням шару Dropout, збільшити кількість даних тощо. Модель, яка недостатньо навчалася, має низьку точність на тренувальних та тестових даних, може бути занадто простою за структурою. У такому випадку варто ускладнити модель, збільшивши глибину та налаштувавши параметри[27].

1.2 Огляд платформи Android

Для створення мобільних Android застосунків використовують інтегроване середовище розробки Android Studio, яке надає багато інструментів: система збирання проєкту Gradle, емулятор з широким функціоналом, інтеграція з GitHub тощо[28]. В Android Studio можна створювати програми на мові Java, Kotlin або C++. Враховуючи власний тривалий досвід програмування, я зупинилася на Java для створення цього застосунку.

В Android Studio проєкт складається з одного або декількох модулів і містить файли ресурсів та вихідного коду. На рисунку показано структуру проєкту, яка створюється за замовчуванням.

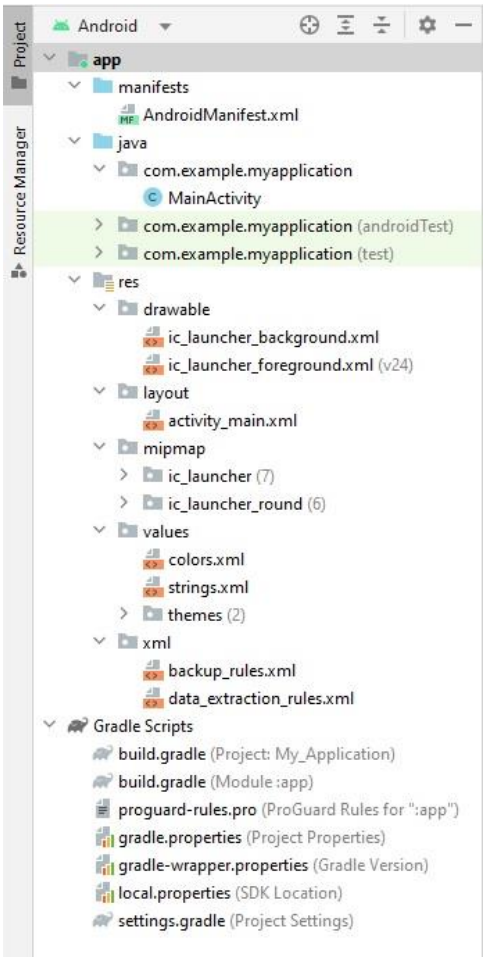

Рисунок 8. Структура Android проєкту

Структура проєкту поділяється на папки: manifests, java і res. У manifests розташований файл AndroidManifest.xml, у якому вказуються компоненти

застосунку. Також у файлі можуть бути прописані дозволи («uses-permission») на доступ до даних мобільного пристрою (наприклад, камери чи галереї). У папці java розміщені вихідні коди на Java (наприклад, Activities) та тестові коди JUnit. Папка res містить ресурси, такі як растрові або векторні зображення, значення рядків, кольорів інтерфейсу, шаблони вікон (layout). До структури проєкту можна додавати власні папки, наприклад, папку ml для розміщення файлів з навченими моделями машинного навчання. Також можна додавати модулі, наприклад, OpenCV для обробки зображень в Android застосунку.

У Gradle Scripts розміщені файли збірки проєкту. У файлі build.gradle можна зазначати залежності від бібліотек, щоб потім їх імпортувати й використовувати в класах. Залежності прописуються в dependency наступним чином: implementation і далі в лапках назва бібліотеки (див. рисунок нижче).

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Рисунок 9. Залежності, прописані у файлі build.gradle

Код із файлами даних і ресурсів за допомогою Android SDK компілюється у файл .apk або Android App Bundle. Android застосунок складається з компонентів, які можуть бути пов’язаними між собою. Існують такі компоненти: activities, services, broadcast receivers та content providers. Activity є точкою входу й представляє собою екран з користувацьким інтерфейсом. Service є компонентом, що підтримує у фоновому режимі роботу застосунку для виконання тривалих або віддалених процесів. Інший компонент - broadcast receiver передає події в програму поза звичайним потоком, аби застосунок міг реагувати на загальносистемні сповіщення (наприклад, будильник, нагадування). Content providers керує спільними даними, до яких застосунок має доступ. Через цей компонент застосунок може отримувати, змінювати дані, якщо це дозволено[29].

1.3 Огляд засобів для поєднання технологій машинного навчання з Android застосунком

Власні навчені моделі можна використовувати в мобільних Android застосунках такими способами: додавати безпосередньо в проєкт у вигляді файлу або зберігати віддалено на зовнішньому ресурсі й користуватися API. Якщо модель буде змінюватися з часом, то краще викласти її на зовнішній ресурс, щоб не оновлювати постійно застосунок.

Одним зі зручних форматів для використання моделей в Android застосунку є TensorFlow Lite, особливостями якого є зменшений розмір, знижене енергоспоживання, відсутність затримок на пристрої, відсутність потреби в підключенні до мережі, забезпечення конфіденційності даних[30]. Усі ці фактори важливі для коректної роботи мобільного застосунку. Навчену модель необхідно експортувати у формат tflite і далі додати файл в проєкт Android застосунку в папку ml. Також можна додати метадані до файлу з описом моделі, інформацією про тип даних, який модель приймає на вхід, і дані, які модель повертає як результат.

Інший спосіб, зі зберіганням моделі на зовнішньому ресурсі, потребує зв’язку Android застосунку для отримання результатів моделі з сервера. Наприклад, можна розгорнути модель в Firebase[31] і використовувати API. Перевагою такого підходу є те, що модель не займає місце в мобільному застосунку і може бути змінена віддалено, проте через потребу в підключенні до мережі можуть виникати проблеми з затримкою, безпекою даних тощо.

Оскільки в проєкті створені моделі з часом не будуть змінюватися, і застосунок повинен працювати без підключення до мережі, то було обрано варіант додавання в проєкт моделі у форматі TensorFlow Lite.

Висновки до розділу 1

З метою вирішення поставленої задачі у кваліфікаційній роботі оптимальним способом мною було розглянуто згорткові нейронні мережі й проаналізовано наявні моделі, навчені на достовірних наборах даних. Враховувалася структура, процес навчання моделей, оцінювалася ефективність використання їх для класифікації зображень. Увага була зосереджена на рішеннях, які найбільше підходять для використання на мобільних пристроях. Таким чином було обрано EfficientNet-Lite0 і MobileNetV2 моделі для власного застосунку. Також було проведено оцінку вибору технології розміщення моделі на пристрої та на зовнішньому ресурсі. Обґрунтовано вибір TensorFlow Lite для використання власних моделей у застосунку, пояснено переваги цієї технології. Серед наявних середовищ розробки було обрано Android Studio за широкий функціонал та зручність розробки.

Розділ 2. Огляд наявних Android застосунків

У цьому розділі розгляньмо мобільні Android застосунки, які використовують технологію машинного навчання – згорткові нейронні мережі. Оскільки поставленою задачею є розпізнавання та класифікація квітів, то до уваги беруться відповідні найпопулярніші застосунки в цій сфері: PlantNet, iNaturalist, PictureThis, Flora Incognita.

PlantNet – це мобільний застосунок, у якому користувачі можуть завантажити свої фотографії для визначення виду рослин. Для розробки цього застосунку використовувалася дрібнозерниста мережа на основі білінійної згорткової нейронної мережі та трансферного навчання[32]. PlantNet навчався на величезному наборі даних, який налічує 1081 вид рослин та 306146 зображень[33]. Користувач завантажує зображення з галереї або робить фотографію за допомогою камери, і далі обирає відповідний елемент: листок, квітка, плід, кора, природнє середовище або інше. Застосунок повертає результати розпізнавання рослини у вигляді списку передбачених видів рослин з відповідними відсотками ймовірності та зображеннями рослин. Першою в списку є рослина з найвищим значенням відсотка – найбільш підхожий вид рослини на зображенні користувача. Також в застосунку можна переглянути фотографії різних рослин, посортованими за родиною, родом та видом. PlantNet надає можливість обирати флору, наприклад: світова флора, корисні рослини певного регіону, бур’яни, інвазивні рослини тощо. У застосунку зберігаються фотографії користувача, які використовувалися для розпізнавання рослин. PlantNet також дозволяє користувачам додавати свої фотографії у спільну базу.

PictureThis – це мобільний застосунок, який використовує технології машинного навчання, зокрема згорткові нейронні мережі для визначення виду рослини на зображенні, наданому користувачем. Програма здатна ідентифікувати понад 17000 видів із точністю 98%, як зазначають творці[34]. У застосунку можна переглянути варіанти передбачення рослини, дізнатися детальну інформацію про рослину (стан здоров’я, опис, характеристики, цікаві факти, наукова

класифікація, місця поширення), отримати корисні поради й рекомендації щодо догляду та лікування рослини. Користувач може зберігати рослини, які ідентифікує, в застосунку до свого «саду» та встановлювати нагадування про час поливу, удобрення, пересаджування рослин тощо.

Flora Incognita також використовує згорткові нейронні мережі для розпізнавання й класифікації видів рослин на зображеннях[35]. За допомогою цього застосунку можна розпізнати рослину з фотографії зі смартфона, дізнатися детальну інформацію про рослину (назва, опис, фотографії, наукова класифікація, середовище поширення). Також користувач має можливість зберігати свої результати спостережень у застосунку, таким чином розширюючи колекцію власних рослин. Flora Incognita надає інформацію про різні види рослин, розділених за категоріями: дерево, польові квітка або кущ, трава, мох тощо. Користувач може дозволити в застосунку відслідковувати своє місцезнаходження за допомогою GPS, щоб записувати локацію створення фотографії рослини. Також в Flora Incognita є можливість поширювати результати своїх спостережень з іншими користувачами.

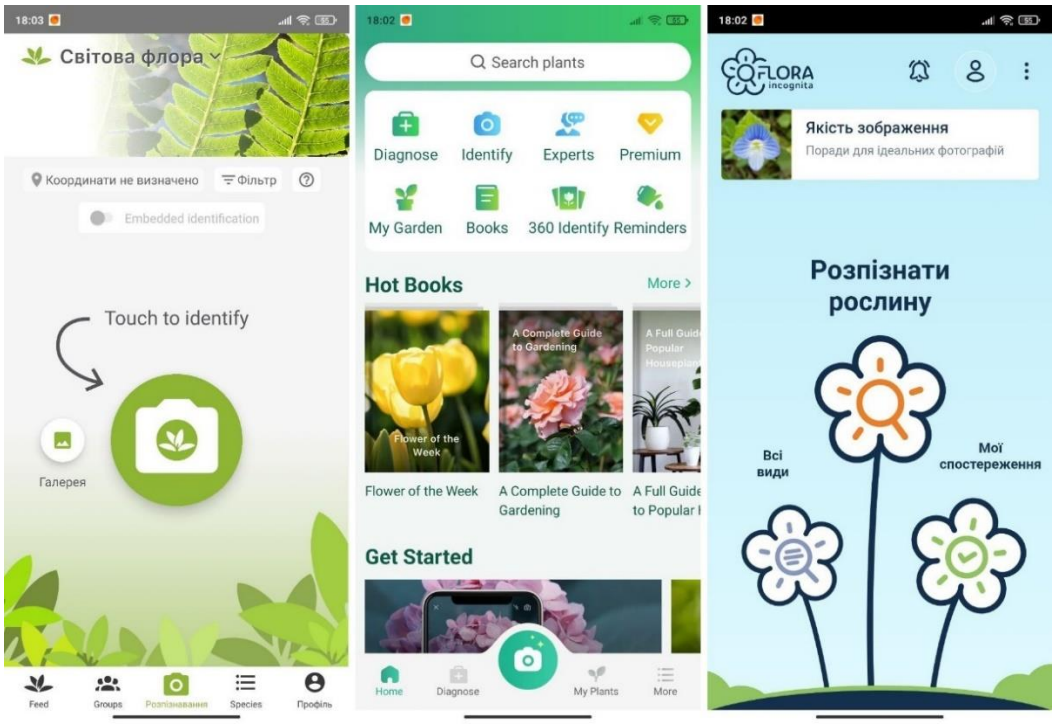

Рисунок 10. Скриншоти застосунків PlantNet, PictureThis, Flora Incognita (розміщені зліва направо)

Висновки до розділу 2

Існує безліч мобільних застосунків, які використовують згорткові нейронні мережі для класифікації зображень. У цьому розділі були розглянуті відомі застосунки для визначення виду рослин: PlantNet, PictureThis, Flora Incognita. Вони дозволяють користувачам ідентифікувати рослини за фотографіями, отримувати про них детальну інформацію і зберігати результати спостережень. Незважаючи на наявність готових рішень для класифікації зображень квітів, було корисно самостійно реалізувати подібне рішення, проекспериментувати над налаштуваннями параметрів моделей та донавчанням попередньо навчених моделей на власному наборі даних. Застосування згорткових нейронних мереж в застосунках сприяють доступності знань, надають можливості для вивчення навколишнього світу, що раніше було складно й затратно.

Розділ 3. Розробка власного Android застосунку

3.1 Аналіз вимог до застосунку

Поставлена задача полягала в розробці Android застосунку з використанням згорткових нейронних мереж для класифікації зображень квітів. У застосунку має бути можливість використовувати різні навчені моделі, вибирати зображення з галереї або робити фотографію камерою смартфона. Необхідно, щоб тренувальні і тестові дані були однаковими для навчання кожної моделі.

Робота складалася з таких завдань: створення і навчання моделей для класифікації зображень, аналіз результатів моделей за допомогою метрик, розробка Android застосунку з можливістю використання камери і галереї для вибору зображення, додавання моделей до застосунку й створення функціоналу для їхнього використання, тестування роботи застосунку.

3.2 Обґрунтування вибору засобів розробки

Набір даних «flowers»[1] був обраний на платформі Kaggle. Він містить 4317 зображень та 5 видів квітів: троянди (784 зображень), соняшники (733 зображень), кульбаби (1052 зображень), ромашки (764 зображень) і тюльпани (984 зображень). Дані записані по папках і рівномірно поділені за категоріями, тому модель, навчаючись на таких даних, уникає упередженості щодо найбільшого класу й має вищу точність. Набір даних було розділено на тренувальні та тестові дані в пропорції 4 до 1 (80% тренувальні і 20% тестові).

Для створення згорткових нейронних мереж була обрана платформа Google Colab, що надає доступ до обчислювальних ресурсів (GPU, TPU), дозволяє ефективно розробляти моделі, їх оптимізовувати та зберігати на Google Drive або комп'ютер. Моделі були створені на мові Python, яка підтримує такі бібліотеки для машинного навчання, як tensorflow, keras, scikit-learn та інші.

У процесі навчання й оцінення моделей були використані різні бібліотеки:

- keras – для побудови власної згорткової нейронної мережі
- tflite-model-maker – для побудови моделей на основі попередньо навчених
- tensorflow – для збереження моделей у форматі TensorFlow Lite для зручного використання в Android застосунку
- numpy – для роботи з масивами даних
- cv2 – для попередньої обробки вхідних зображень для власної моделі
- matplotlib – для побудови матриць невідповідностей на тестових даних
- sklearn.metrics – для обчислення точності, влучності, повноти, F-міри

Для розробки мобільного Android застосунку було обрано середовище Android Studio та мову Java.

3.3 Реалізація застосунку

Попередню обробку зображень для навчання власної моделі було проведено з бібліотекою OpenCV. Розміри всіх зображень були зведені до однакового розміру IMG_SIZE (224x224). На рисунку 16 наведено відповідний метод для обробки вхідних даних.

```
# метод для обробки даних: маркування зображення, перетворення в масив, зміна розміру зображення
def make_train_data(flower_type,DIR):
    for img in tqdm(os.listdir(DIR)):
        label=assign_label(img,flower_type)
        path = os.path.join(DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))

        x_train.append(np.array(img))
        y_train.append(str(label))
```

Рисунок 11. Обробка зображень за допомогою OpenCV

Зображення зчитувалися з файлів, змінювався їхній розмір за допомогою методу cv2.resize(), і далі вони були записані в numpy масив. Також було створено масив з мітками (назви виду квітів) та промарковано зображення відповідно їхньому розміщенню в папках.

Було створено 3 моделі: 1 власна згорткова нейронна мережа та дві моделі, побудовані на основі MobileNetV2 та EfficientNet-Lite0. У розділі 1 було пояснено вибір попередньо навчених моделей: вони мають розмір до 16 Мб та точність від 72% на наборі даних ImageNet, що дозволяє їх використовувати на мобільних пристроях з обмеженими ресурсами. Усі моделі навчалися на тренувальних даних і були перевірені - на тестових.

Власна згорткова нейронна мережа містить 4 згорткові шари, 4 агрегувальні шари, шар Flatten, який перетворює вхідні дані в одновимірний масив, і повнозв’язний шар Dense. На згорткових шарах було використано функцію активації ReLu, а на вихідному шарі – функцію softmax для обчислення ймовірності кожної категорії. За кожним згортковим шаром іде агрегувальний шар MaxPooling2D, у якому з рецептивного поля обирається піксель з

найбільшим значенням. Повнозв'язний шар Dense містить 512 нейронів. Останній вихідний шар моделі має 5 нейронів, що відповідають за кожну категорію квітів. Таким чином на виходах отримуємо ймовірність для кожного виду квітів, і той вид, що має найвищу ймовірність, є прогнозованим моделлю. На рисунку 17 можна переглянути код створення моделі.

```
# CNN модель
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters =96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters = 96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(5, activation = "softmax"))
```

Рисунок 12. Створення власної згорткової нейронної мережі

Для покращення результатів власної моделі використовувалося штучне розширення кількості даних за рахунок обертання, масштабування, зсувів зображення (див. код на рисунку 18).

```
datagen = ImageDataGenerator(featurewise_center=False, samplewise_center=False,
 featurewise_std_normalization=False, samplewise_std_normalization=False,
 zca_whitening=False, rotation_range=10, zoom_range = 0.1, width_shift_range=0.2,
 height_shift_range=0.2, horizontal_flip=True, vertical_flip=False)
datagen.fit(x_train)
```

Рисунок 13. Використання ImageDataGenerator для розширення кількості даних

Моделі на основі MobileNetV2, EfficientNetLite0 були створені за допомогою бібліотеки tflite-model-maker. Їхні вихідні шари були змінені відповідно до кількості видів квітів. Нижче наведено код створення моделей.

```
model_efficientnet_lite0 = image_classifier.create(train_data, model_spec=model_spec.get('efficientnet_lite0'))
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hub_keras_layer_v1v2_1 (Hub  (None, 1280)             3413024
 KerasLayerV1V2)

 dropout_1 (Dropout)         (None, 1280)              0

 dense_1 (Dense)             (None, 5)                 6405

=================================================================
```

```
Total params: 3,419,429
Trainable params: 6,405
Non-trainable params: 3,413,024
_________________________________________________________________
None
Epoch 1/5
107/107 [==============================] - 231s 2s/step - loss: 0.8741 - accuracy: 0.7631
Epoch 2/5
107/107 [==============================] - 234s 2s/step - loss: 0.6719 - accuracy: 0.8817
Epoch 3/5
107/107 [==============================] - 228s 2s/step - loss: 0.6361 - accuracy: 0.9121
Epoch 4/5
107/107 [==============================] - 241s 2s/step - loss: 0.6181 - accuracy: 0.9159
Epoch 5/5
107/107 [==============================] - 229s 2s/step - loss: 0.6048 - accuracy: 0.9246
```

Рисунок 14. Створення і навчання моделі на основі EfficientNet-Lite0

```
model_mobilenet_v2 = image_classifier.create(train_data, model_spec=model_spec.get('mobilenet_v2'))

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hub_keras_layer_v1v2 (HubKe  (None, 1280)             2257984
 rasLayerV1V2)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 5)                 6405

=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
None
Epoch 1/5
107/107 [==============================] - 517s 5s/step - loss: 0.9275 - accuracy: 0.7561
Epoch 2/5
107/107 [==============================] - 197s 2s/step - loss: 0.7127 - accuracy: 0.8741
Epoch 3/5
107/107 [==============================] - 203s 2s/step - loss: 0.6696 - accuracy: 0.8858
Epoch 4/5
107/107 [==============================] - 201s 2s/step - loss: 0.6469 - accuracy: 0.9010
Epoch 5/5
107/107 [==============================] - 196s 2s/step - loss: 0.6257 - accuracy: 0.9109
```

Рисунок 15. Створення і навчання моделі на основі MobileNetV2

Результати моделей були проаналізовані за допомогою таких метричних даних, як: точність, влучність, повнота, F-міра. У таблиці наведено результати для кожної з моделей (CNN – власна згорткова нейронна мережа, MobileNetV2 – модель, побудована на основі MobileNetV2, EfficientNet-Lite0 – модель, побудована на основі EfficientNet-Lite0).

№	Модель	Точність на тренувальних даних	Точність на тестових даних	Влучність	Повнота	F-міра
1	CNN	0,889	0,809	0,811	0,809	0,808
2	MobileNetV2	0,943	0,890	0,892	0,890	0,889
3	EfficientNet-Lite0	0,949	0,913	0,914	0,913	0,913

Найкращою за показниками є модель EfficientNet-Lite0, а найгіршою - модель CNN, яка має найнижчу точність – 88,9%.

Також було переглянуто матриці невідповідностей для кожної моделі.

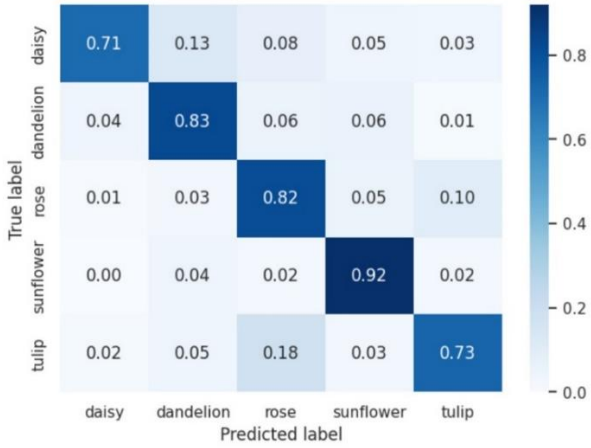

Рисунок 16. Матриця невідповідностей моделі CNN

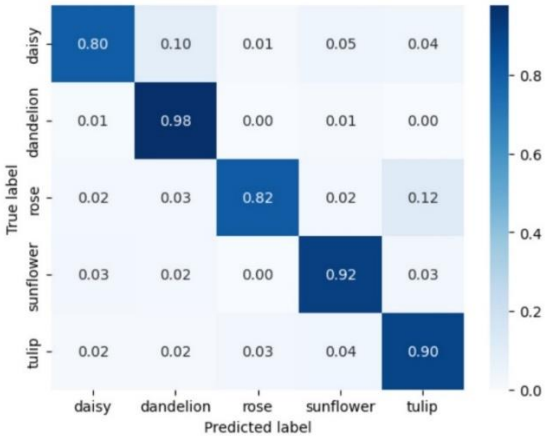

Рисунок 17. Матриця невідповідностей моделі MobileNetV2

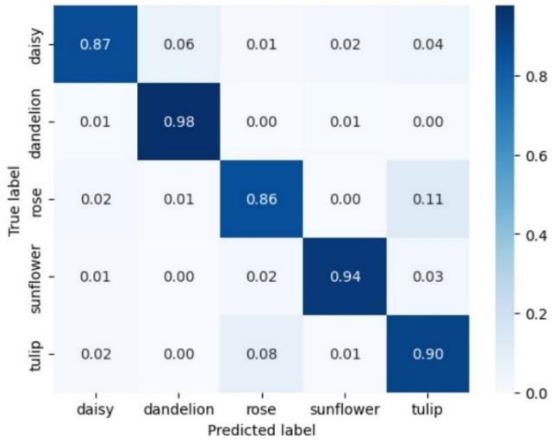

Рисунок 18. Матриця невідповідностей моделі EfficientNet-Lite0

З матриць невідповідностей можна зробити висновок, що більшість передбачень моделей є коректними, проте найбільше невідповідностей є між класами tulip та rose (тюльпани і троянди). Модель CNN класифікує 10% зображень троянд як тюльпани, модель MobileNetV2 – 12% зображень, модель EfficientNet-Lite0 – 11%. Також можна зазначити, що модель CNN в порівнянні з іншими моделями має більше невідповідностей, тобто гірше класифікує зображення з тестових даних.

Далі моделі було збережено з метаданими в TensorFlow Lite формат для зручного використання в Android застосунку (див. код на рисунку).

```
import tensorflow
tensorflow.keras.models.save_model(model, 'model.pbtxt')
converter = tensorflow.lite.TFLiteConverter.from_keras_model(model = model)
model_tflite = converter.convert()
open("model.tflite", "wb").write(model_tflite)
```

Рисунок 19. Збереження моделі у форматі TensorFlow Lite (tflite)

Важливо зазначити, що після збереження моделей в файли у форматі TensorFlow Lite їхні розміри вийшли такими: CNN – 16,1 Мб, MobileNetV2 – 2,7 Мб, EfficientNet-Lite0 – 3,9 Мб. Модель, побудована на основі MobileNetV2, є найменшою за розміром і має високу точність – 94,3%, що не поступається точності EfficientNet-Lite0 (94,9%) і робить її найкращою серед інших моделей для використання в Android застосунку.

Також до моделі були додані відповідні метадані, які можна переглянути в Android Studio (див. рисунок 24).

Model

Name	Proprietary convolutional neural network
Description	The model with the structure of a convolutional neural network for recognition and classification images by flower species
Version	v1
Author	Khrystyna Boiko
License	License. Version 1.0

Tensors

Inputs

Name	Type	Description	Shape	Min / Max
input image	Image <float32>	Input image of flower to be classified. The expected image is 150 x 150, with three channels (red, blue, and green) per pixel. Each element in the tensor is a value between min and max, where (per-channel) min is [-1] and max is [1].	[1, 150, 150, 3]	[-1] / [1]

Outputs

Name	Type	Description	Shape	Min / Max
probability of class	Feature <float32>	Probabilities for each flower species (5 outputs).	[1, 5]	[0] / [1]

Рисунок 20. Інформація про модель, взята з метаданих

В Android Studio був створений застосунок з відповідними вікнами: головне вікно з вибором моделі для визначення квітів, вікно інструкції користувача, вікно для вибору зображення (з галереї чи зробити фото камерою), вікно відповіді (визначення моделлю квітів на фото).

Навчені моделі були додані до проєкту застосунку в папку ml (див. рисунок).

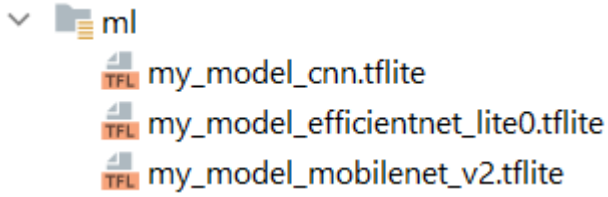

Рисунок 21. Файли моделей в папці ml проєкту

В Android застосунку модель завантажується з файлу tflite. Для цього створюється об’єкт Interpreter, що отримує з моделі форму вхідного тензора. Створюється, заповнюється масив із даними зображення і передається в метод run() об’єкта Interpreter. Таким чином результати моделі (оцінки ймовірності для кожної категорії) будуть отримані в масиві output (див. код на рисунку).

```
String modelPath = "my_model_cnn.tflite";
Interpreter.Options options = new Interpreter.Options();
Interpreter tflite = null;
try {
    tflite = new Interpreter(loadModelFile(context, modelPath), options);
} catch (IOException e) {
    throw new RuntimeException(e);
}
Tensor inputTensor = tflite.getInputTensor( inputIndex: 0);
int[] inputShape = inputTensor.shape();
ByteBuffer imgData = ByteBuffer.allocateDirect( capacity: inputShape[0] * inputShape[1] * inputShape[2] * inputShape[3] * 4);
imgData.order(ByteOrder.nativeOrder());
for (float value : imgFloats) {
    imgData.putFloat(value);
}
imgData.rewind();
int[] outputShape = tflite.getOutputTensor( outputIndex: 0).shape();
float[][] output = new float[outputShape[0]][outputShape[1]];
tflite.run(imgData, output);
```

Рисунок 22. Використання моделі в Android застосунку

Результати в масиві output сортуються і виводяться в порядку спадання – від найбільш імовірного класу до найменш (див. рисунок). Далі результати у вигляді рядків додаються до TextView для представлення інформації у вікні застосунку.

```
List<ClassProbability> classOutputs = new ArrayList<>();
for (int i = 0; i < output[0].length; i++) {
    classOutputs.add(new ClassProbability(classNamesUA[i], output[0][i]));
}
Collections.sort(classOutputs);
```

Рисунок 23. Сортування результатів моделі для кожного класу

Проєкт в Android Studio містить такі основні компоненти: класи Activity, файл Manifest, файли моделей, в папці res шаблони для вікон, зображення.

У MainActivity іконки моделей та підписи під ними додаються до RecyclerView у вигляді списку, тому за необхідності можна додавати нові іконки моделей.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main1);
    ArrayList<ModelItem> arrayList = new ArrayList<>();
    arrayList.add(new ModelItem(R.mipmap.icon_model_efficientnet, getString(R.string.efficientnet_lite0_model), FlowerClassificationEfficientNetLite.class));
    arrayList.add(new ModelItem(R.mipmap.icon_model_cnn, getString(R.string.cnn_model), FlowerClassificationCNN.class));
    arrayList.add(new ModelItem(R.mipmap.icon_model_mobilenet, getString(R.string.mobilenetv2_model), FlowerClassificationMobileNet.class));
    ModelAdapter modelAdapter = new ModelAdapter(arrayList, listener: this);
    RecyclerView recyclerView = findViewById(R.id.main_recycler_view);
    recyclerView.setAdapter(modelAdapter);
    recyclerView.setLayoutManager(new GridLayoutManager( context: this, spanCount: 1));
    if(OpenCVLoader.initDebug()) Log.d( tag: "LOADED", msg: "success");
    else Log.d( tag: "LOADED", msg: "err");
}
```

Рисунок 24. Додавання іконок моделей на головному вікні

Усі моделі для будь-якого зображення обчислюють оцінку ймовірності для кожного виду квітів. У деяких випадках ймовірність для передбачуваного класу є низькою, наприклад, ромашка – 34%, кульбаба – 26%, троянда – 19%, соняшник – 17%, тюльпан – 4%. У такому разі можна вважати, що вид квітів на зображенні не було розпізнано. Тому було прийнято рішення дати можливість користувачеві обирати допустиму мінімальну оцінку ймовірності для передбачуваної категорії квітів. Якщо користувач підвищить допустиму оцінку ймовірності до певного значення (наприклад, 90%), то в застосунку результати моделі будуть показані лише тоді, коли передбачуваний клас має ймовірність більшу за встановлене значення. Інакше виведеться текст про неможливість ідентифікувати вид квітів на зображенні.

У застосунку було додано можливість для користувача змінювати допустиму мінімальну оцінку ймовірності. Вона зберігається в SharedPreferences та використовується для визначення результату для моделей.

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        Context context = getApplicationContext();
        SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putInt( s: "minTolerance", progress);
        editor.apply();
        TextView textViewProgress = findViewById(R.id.Tolerance);
        textViewProgress.setText("Допустима ймовірність: " + seekBar.getProgress());
    }
```

Рисунок 25. Зміна значення допустимої оцінки ймовірності при зміні повзунка

У застосунку отримується дозвіл на доступ до камери мобільного пристрою.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if(requestCode == CAMERA_PERMISSION_REQUEST_CODE){
        if(grantResults[0] == PackageManager.PERMISSION_GRANTED){
            invokeCamera();
        } else {
            Toast.makeText( context: this,  text: "Cannot take photo without permission.", Toast.LENGTH_LONG).show();
        }
    }
}
```

Рисунок 26. Метод для доступу до камери

В залежності від вибору користувача, обробляється зображення галереї або камери. Зображення перетворюється у формат Bitmap і встановлюється на ImageView для показу на вікні. Зображення передається у метод runDetection() для розпізнавання виду квітів моделлю.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            Bitmap bitmap = getCapturedImage();
            rotateIfRequired(bitmap);
            inputImageView.setImageBitmap(bitmap);
            runDetection(bitmap);
        } else { // Result was a failure
            Toast.makeText( context: this,  text: "Picture wasn't taken!", Toast.LENGTH_SHORT).show();
        }
    } else if (requestCode == PICK_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            Bitmap takenImage = loadFromUri(data.getData());
            inputImageView.setImageBitmap(takenImage);
            runDetection(takenImage);
        } else { // Result was a failure
            Toast.makeText( context: this,  text: "Picture wasn't selected!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Рисунок 27. Вибір камери або галереї

3.4 Принцип роботи і тестування застосунку

На головному екрані користувач обирає модель, за допомогою якої можна визначити вид квітів на зображенні, натиснувши на відповідну кнопку. Далі вибирає зображення: це може бути фото з галереї або можна зробити фото камерою смартфона. Обране зображення відобразиться у застосунку разом з визначенням виду квітів.

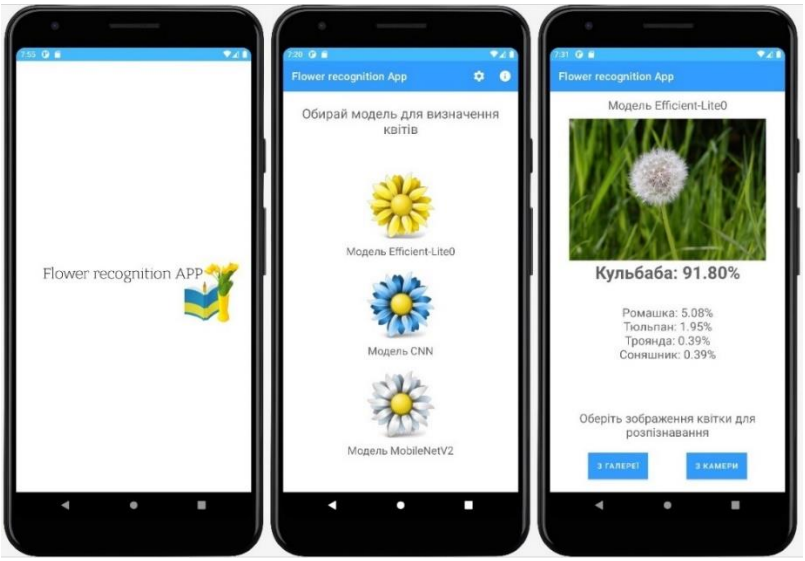

Рисунок 28. Інтерфейс застосунку

Моделі мають різну будову, тому дають різні результати для одного й того самого вхідного зображення:

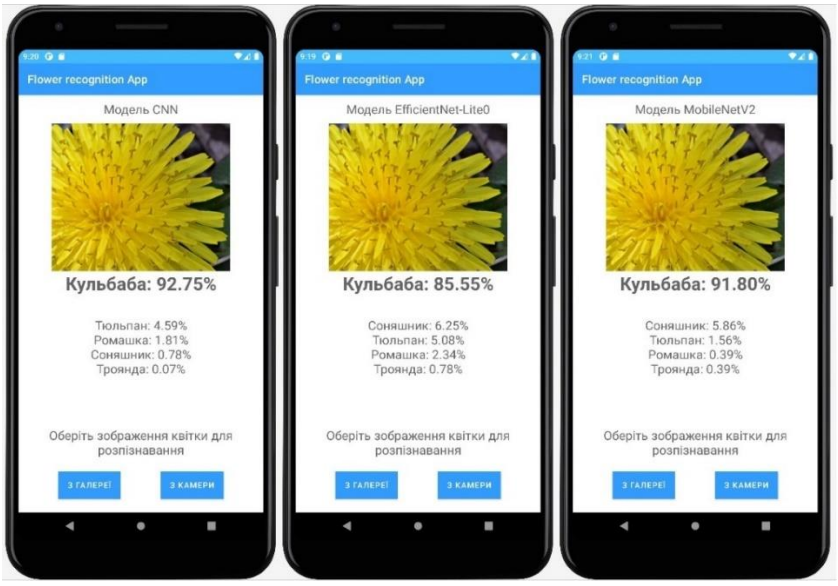

Рисунок 29 . Результати моделей для того самого зображення

Користувач може з головного вікна перейти на вікно інструкції, як користуватися застосунком, або на вікно налаштування допустимої мінімальної оцінки ймовірності, яку можна змінювати.

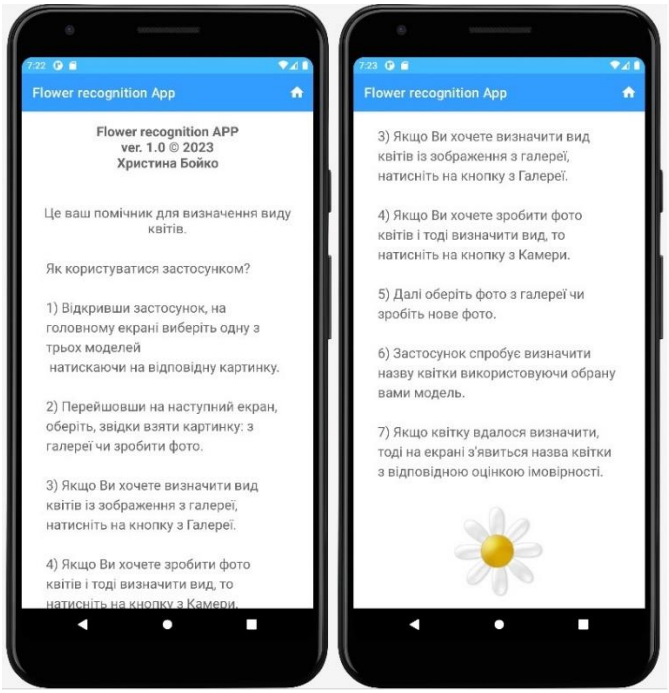

Рисунок 30. Інструкція для користувача

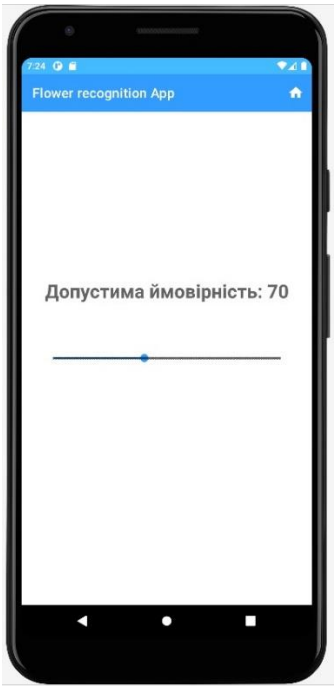

Рисунок 31. Налаштування допустимої оцінки ймовірності для моделей

Після інсталяції застосунку на мобільний пристрій користувач для розпізнавання квітів повинен дозволити доступ до фото й медіа на пристрої (якщо зображення з галереї) або до камери (якщо потрібно зробити нове фото).

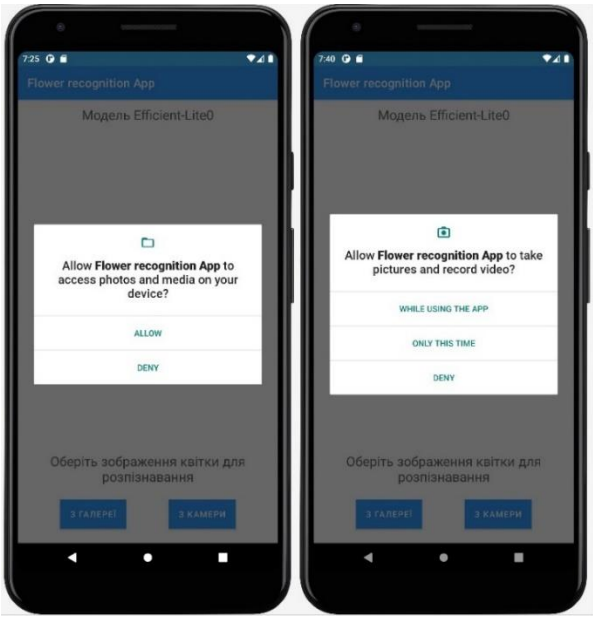

Рисунок 32. Запит дозволів на доступ до галереї, камери пристрою

Якщо зображення отримало оцінку ймовірності для передбачуваного класу меншу, за встановлену користувачем, тоді на екран виводиться текст, що модель не здатна розпізнати квіти на зображенні.

Рисунок 33. Не розпізнавання виду квітів на зображенні

Застосунок тестувався в емуляторі в Android Studio та на фізичному пристрої Android (Xiaomi MI 8, Android 10.0, Motorola g20, Android 11.0,).

Застосунок має певні недоліки. Оскільки набір даних обмежений 5-ма видами квітів, то на зображеннях, що не містять цих квітів, модель може «розпізнавати» види квітів. Для подальшого збільшення точності класифікації зображень доцільно побудувати більш ефективні згорткові нейронні мережі, шляхом зміни параметрів, їхньої структури (кількості шарів, фільтрів тощо), і також навчанням на наборі даних з більшою кількістю зображень і категорій.

Висновки до розділу 3

На базі проаналізованого теоретичного матеріалу та обраних підходів й інструментів було самостійно створено Android застосунок «Flower Recognition App», що вміє з заданою точністю визначати вид квітів на зображеннях. Для цього були створені й натреновані моделі, що є згортковими нейронними мережами. Найкращими за показниками вийшла модель, побудована на основі EfficientNet-Lite0, проте за розміром найменшою є модель, побудована на основі MobileNetV2. Моделі було додано до Android застосунку. Робота застосунку була перевірена в емуляторі в Android Studio та на мобільних пристроях. Під час роботи виникали труднощі з підключенням правильної комбінації версій бібліотек, зі створенням метаданих, з перетворенням зображень в застосунку до відповідного розміру й формату, який приймає модель. Попри складнощі вдалося реалізувати мобільний застосунок з функціоналом, який повністю відповідає вимогам поставленого індивідуального завдання на кваліфікаційну роботу.

Висновки

Результатом кваліфікаційної роботи є створений мобільний Android застосунок, який дозволяє розпізнавати й класифікувати види квітів на зображеннях. Для вирішення цієї задачі були створені й навчені згорткові нейронні мережі: одна власна модель та дві моделі на основі MobileNetV2 та EfficientNet-Lite0. У застосунку можна обирати модель для визначення виду квітів, налаштовувати допустиму мінімальну оцінку ймовірності для передбачуваного класу й обирати зображення з галереї або робити фото за допомогою камери.

У роботі досліджено згорткові нейронні мережі: вивчено їхню структуру й особливості навчання. Висвітлено способи для покращення результатів моделей та можливості використання їх в Android застосунках. Також розглянуто роботу сучасних мобільних застосунків, які використовують згорткові нейронні мережі для класифікації зображень. Обґрунтовано, чому згорткові нейронні мережі є потужним підходом для класифікації зображень, а також які попередньо навчені моделі найкраще підходять для використання в мобільному застосунку. Результати моделей проаналізовані через метрики. Створено компоненти Android застосунку, реалізовано можливість використовувати в ньому різні моделі, обирати зображення. Застосунок протестовано в емуляторі та на мобільних пристроях.

У майбутньому можна покращити моделі через підбір параметрів, дослідження результатів навчання, розширення набору даних з більшою кількістю категорій зображень для точнішої класифікації.

Список використаних джерел

1. Flowers Recognition [Електронний ресурс] – Режим доступу: https://www.kaggle.com/datasets/alxmamaev/flowers-recognition (дата звернення: 17.04.2023)
2. 20 Android Statistics In 2023 (Market Share & Users) [Електронний ресурс] / Daniel Ruby, 2023 – Режим доступу: https://www.demandsage.com/android-statistics/ (дата звернення: 23.04.2023)
3. Why On-Device Machine Learning? – [Електронний ресурс] / Google for Developers – Режим доступу: https://developers.google.com/learn/topics/on-device-ml/learn-more (дата звернення: 20.04.2023)
4. CNN Basic Architecture for Classification & Segmentation – [Електронний ресурс] / March 26, 2023 by Ajitesh Kumar – Режим доступу: https://vitalflux.com/cnn-basic-architecture-for-classification-segmentation/ (дата звернення: 13.04.2023)
5. Translation Invariance in Convolutional Neural Networks – [Електронний ресурс]/ Nov 14, 2019 by Divyanshu Soni – Режим доступу: https://divsoni2012.medium.com/translation-invariance-in-convolutional-neural-networks-61d9b6fa03df (дата звернення: 13.04.2023)
6. These Machine Learning Techniques Make Google Lens A Success – [Електронний ресурс] / Analytics India Magazine – Режим доступу: https://analyticsindiamag.com/these-machine-learning-techniques-make-google-lens-a-success/ (дата звернення: 15.03.2023)
7. The AI Behind FaceApp – [Електронний ресурс] / June 29, 2020 by SEJUTI DAS – Режим доступу: https://analyticsindiamag.com/the-ai-behind-faceapp/ (дата звернення: 12.02.2023)

8. Machine learning – [Електронний ресурс] / by Ed Burns – Режим доступу: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (дата звернення: 13.02.2023)
9. – [Електронний ресурс] – Режим доступу: http://www.mmf.lnu.edu.ua/ar/1739 (дата звернення: 23.04.2023)
10. Машинне Навчання Простими Словами. Частина 1 – [Електронний ресурс] 15 лютого 2022 yaneya – Режим доступу: https://www.javatpoint.com/machine-learning-models (дата звернення: 12.02.2023)
11. Як працює machine learning та його застосування на практиці – [Електронний ресурс] / 31.01.2019, 13:41 Аліна Присяжнюк – Режим доступу: https://nachasi.com/tech/2019/01/31/yak-pratsyuye-machine-learning/ (дата звернення: 15.01.2023)
12. What is Classification in Machine Learning and Why is it Important? – [Електронний ресурс]/ 23 November 2022 by Krati Joshi – Режим доступу: https://emeritus.org/blog/artificial-intelligence-and-machine-learning-classification-in-machine-learning/ (дата звернення: 13.01.2023)
13. Artificial Neural Network Tutorial – [Електронний ресурс] / – Режим доступу: https://www.javatpoint.com/artificial-neural-network (дата звернення: 01.02.2023)
14. Activation functions in Neural Networks – [Електронний ресурс] / Sakshi Tiwari – Режим доступу: https://www.geeksforgeeks.org/activation-functions-neural-networks/ (дата звернення: 13.02.2023)
15. How does Backward Propagation Work in Neural Networks? – [Електронний ресурс] / Neha Seth — Published On June 1, 2021 and Last Modified On June 8th, 2021 – Режим доступу: https://www.analyticsvidhya.com/blog/2021/06/how-does-backward-propagation-work-in-neural-networks/ (дата звернення: 23.04.2023)

16. Convolutional Neural Network: Benefits, Types, and Applications – [Електронний ресурс] – Режим доступу: https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/ (дата звернення: 20.04.2023)
17. Transfer Learning with Convolutional Neural Networks in PyTorch – [Електронний ресурс] /Will Koehrsen Nov 26, 2018 Режим доступу: https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce (дата звернення: 25.04.2023)
18. ImageNet – [Електронний ресурс]/ Jia Deng – Режим доступу: https://paperswithcode.com/dataset/imagenet (дата звернення: 12.02.2023)
19. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) – [Електронний ресурс] – Режим доступу: https://www.image-net.org/challenges/LSVRC/ (дата звернення: 12.02.2023)
20. Higher accuracy on vision models with EfficientNet-Lite – [Електронний ресурс]/ March 16, 2020 by Posted by Renjie Liu, Software Engineer – Режим доступу: https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html (дата звернення: 20.03.2023)
21. Performance – [Електронний ресурс] – Режим доступу: https://pocketflow.github.io/performance/ (про MobileNetV1) (дата звернення: 13.03.2023)
22. Image Classification on ImageNet – [Електронний ресурс] – Режим доступу: https://paperswithcode.com/sota/image-classification-on-imagenet (дата звернення: 20.03.2023)
23. MobileNet V3 – [Електронний ресурс] / Feb 12, 2021 – Режим доступу: https://paperswithcode.com/lib/torchvision/mobilenet-v3 (дата звернення: 20.03.2023)
24. TF EfficientNet Lite – [Електронний ресурс] /Feb 14, 2021 – Режим доступу: https://paperswithcode.com/model/tf-efficientnet-lite?variant=tf-efficientnet-lite0 (дата звернення: 20.03.2023)

25. EfficientNet-lite – [Електронний ресурс] / Apr 21, 2020 – Режим доступу: https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/lite/README.md (дата звернення: 23.03.2023)
26. Performance Metrics in Machine Learning – [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/performance-metrics-in-machine-learning (дата звернення: 20.03.2023)
27. Improving Performance of Convolutional Neural Network! – [Електронний ресурс] / Aug 14, 2018 by Dipti Pawar – Режим доступу: https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7 (дата звернення: 02.04.2023)
28. Meet Android Studio – [Електронний ресурс] – Режим доступу: https://developer.android.com/studio/intro (дата звернення: 03.01.2023)
29. Application fundamentals– [Електронний ресурс] – Режим доступу: https://developer.android.com/guide/components/fundamentals (дата звернення: 13.01.2023)
30. TensorFlow Lite – [Електронний ресурс] – Режим доступу: https://www.tensorflow.org/lite/guide (дата звернення: 17.03.2023)
31. Firebase – [Електронний ресурс] – Режим доступу: https://firebase.google.com/ (дата звернення: 22.02.2023)
32. PlantNet: transfer learning-based fine-grained network for high-throughput plants recognition – [Електронний ресурс]/ 05 January 2022 by Ziying Yang, Wenyan He, Xijian Fan – Режим доступу: https://link.springer.com/article/10.1007/s00500-021-06689-y (дата звернення: 13.02.2023)
33. Pl@ntNet-300K image dataset – [Електронний ресурс] / Camille Garcin IMAG, Univ Montpellier, Inria, CNRS; Alexis Joly; Pierre Bonnet; Antoine Affouard; Jean-Christophe Lombardo; Mathias Chouet; Maximilien Servajean; Titouan Lorieul; Joseph Salmon– Режим доступу: https://zenodo.org/record/5645731#.ZGX1uXZBxrp (дата звернення: 16.01.2023)

34. PictureThis - Plant Identifier– [Електронний ресурс] – Режим доступу: https://apps.apple.com/us/app/picturethis-plant-identifier/id1252497129 (дата звернення: 13.01.2023)
35. Flora Incognita– [Електронний ресурс] – Режим доступу: https://floraincognita.com/flora-incognita/ (дата звернення: 13.01.2023)

Список використаних зображень

з1 – [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/328733599/figure/fig2/AS:734165188218886@1552050029499/The-structure-of-the-artificial-neuron.png

з2 – [Електронний ресурс] – Режим доступу:

https://www.researchgate.net/publication/354817375/figure/fig2/AS:1071622807097344@1632506195651/Multi-layer-perceptron-MLP-NN-basic-Architecture.jpg

з3 – [Електронний ресурс] – Режим доступу: https://pub.mdpi-res.com/applsci/applsci-09-04500/article_deploy/html/images/applsci-09-04500-g001-550.jpg?1573807881

з4 – [Електронний ресурс] – Режим доступу: https://production-media.paperswithcode.com/methods/Screen_Shot_2020-06-06_at_10.37.14_PM.png

з5 – [Електронний ресурс] – Режим доступу: https://4.bp.blogspot.com/-K6QM83BuiDw/Xm-jIy3K5OI/AAAAAAAAC28/OzMyVWwYgD4LhZt5CflmjLW_e_iydJEWgCLcBGAsYHQ/s1600/figure1a.png

з6 – [Електронний ресурс] – Режим доступу: https://1.bp.blogspot.com/-Rx3Zf6EoPJw/Xm-

jGARge9I/AAAAAAAAC24/Gu12faMjGVERKFWkwfZTghUxipDZtOvwgCLcBGAsYHQ/s1600/figure1b.png