КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

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

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«ANDROID - ЗАСТОСУНОК ДЛЯ РОЗПІЗНАВАННЯ ВИРАЗІВ ТА ПОЛОЖЕННЯ ОБЛИЧ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ»

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

Науковий керівник:
кандидат технічних наук, доцент
Олександр Самощенко _____

Рецензент:
_____

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

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

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

(підпис)

Київ – 2023

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра містить 52 сторінки, 22 рисунки, 1 таблицю, 2 додатки, використано 18 інформаційних джерел.

Об‘єкт дослідження – способи визначення облич на комп'ютерних зображеннях та в режимі реального часу.

Мета роботи – розробка мобільного застосунку для розпізнавання облич, здатний працювати без інтернет з'єднання, на комп'ютерних зображеннях та в режимі реального часу.

Проаналізовано мови програмування, такі як: Java та Kotlin, фреймворки та інструменти для роботи з нейронними мережами: OpenCV, ML Kit, PyTorch. Розроблено та опубліковано Android-застосунок, який може аналізувати зображення відповідно до мети дослідження.

ANDROID-ЗАСТОСУНОК, РОЗПІЗНАВАННЯ ОБЛИЧ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ANDROID, ANDROID SDK, KOTLIN, ML KIT, JETPACK COMPOSE, FIREBASE.

ЗМІСТ

ВСТУП

Технологія розпізнавання облич (face detection) має великий потенціал для використання в безпеці, біометрії, правоохоронній діяльності, розвагах, особистій безпеці та багатьох інших галузях. Виявлення облич може бути реалізовано до зображень (статично, умовно - офлайн) або в режимі реального часу (динамічно - онлайн).

Розпізнавання облич є однією з найбільш перспективних галузей комп'ютерної техніки, що знаходить широке застосування в різних галузях, таких як безпека, біометрика, медицина, особиста безпека, розваги тощо. Зокрема, може бути використана для ідентифікації людей на публічних місцях, для відстеження руху пасажирів у громадському транспорті, для контролю доступу в приміщення та комп'ютерні системи.

Amazon використовує технології виявлення облич для розпізнавання клієнтів у магазинах без касирів Amazon Go. Facebook використовує виявлення облич для автоматичного тегування на фотографіях та для підтвердження облікових записів.

РОЗДІЛ 1

ОСНОВНІ ВІДОМОСТІ ПРО НЕЙРОННІ МЕРЕЖІ

1.1 Класифікація нейронних мереж та їх властивості

Нейронні мережі розрізняють за топологічними типами відповідно до структури зв'язків між нейронами мережі, а також за типом формальних нейронів [3].

Нейронні мережі можна розглядати як спрямований граф зі зваженими зв'язками, у якому нейрони є вузлами. За архітектурою нейронні мережі можуть бути згруповані в два класи:

- мережі прямого поширення (в них графи не мають петель);
- рекурентні мережі (мережі зі зворотними зв'язками);

Також нейронні мережі поділяють за такими ознаками (рис.1.1).

За типом нейронів:

- однорідні;
- гібридні;

За методом навчання:

- навчання з вчителем;
- без вчителя;
- з підкріпленням;

За типом вхідних даних:

- аналогові;
- двійкові;
- образні;

За налаштуванням синапсів:

- фіксоване;
- динамічне;

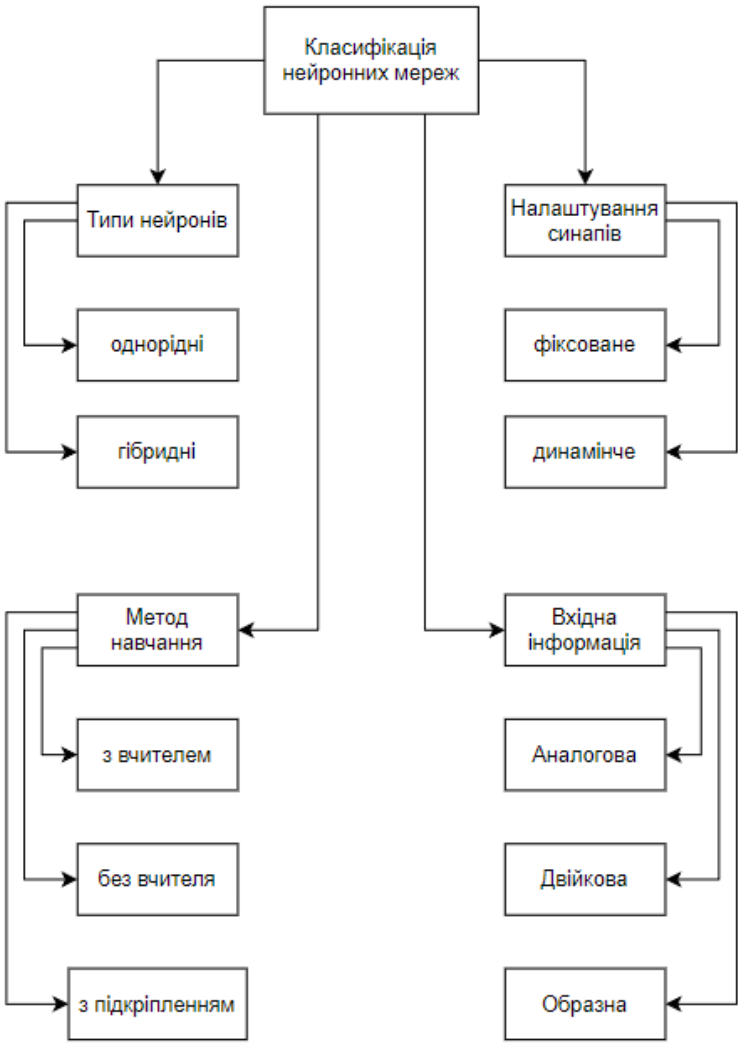


Рисунок 1.1 Класифікація нейронних мереж

1.2 Види нейронних мереж

Нейронні мережі не обмежуються тільки одним видом. Існує декілька видів нейронних мереж, які можна використовувати для різних задач. В даному розділі розглядаються основні відмінності найпоширеніших нейронних мереж [4].

Перцептрон (Perceptron)

Перцептрон - це проста форма штучної нейронної мережі (neural network), яка може виконувати бінарну класифікацію вхідних даних.

Винайдений Френком Розенблаттом в 1957 році, перцептрон був першим кроком до розуміння того, як працює мозок і як можна використовувати комп'ютер для моделювання поведінки мозку. [10]

Перцептрон складається зі штучних нейронів, кожен з яких приймає деякі вхідні дані, обчислює вагову суму цих даних і застосовує функцію активації для обчислення вихідних даних. Даний процес називається проходженням вперед через нейронну мережу.

Перцептрон зазвичай має один або кілька вхідних шарів, один або кілька прихованих шарів і один вихідний шар. Кожен нейрон в прихованому шарі приймає вхідні дані від кожного нейрона в попередньому шарі, обчислює вагову суму і застосовує функцію активації.

Функція активації може бути різної форми, але зазвичай використовують сигмоїдальну функцію, яка перетворює вхідні дані в діапазоні від 0 до 1. Це дозволяє визначити ймовірність того, що вхідні дані належать до певного класу.

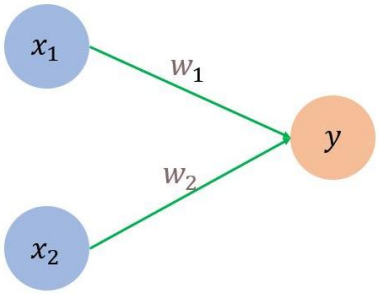


Рисунок 1.2 – Перцептрон

Багатошаровий перцептрон (Multi-Layer Perceptron)

Це тип нейронної мережі, який складається з багатьох шарів перцептронів. Кожен шар з'єднаний з попереднім та наступним шарами за допомогою зважених з'єднань.

Багатошаровий перцептрон має хоча б два прихованих шари, крім вхідного та вихідного шарів. Даний тип нейронних мереж може вирішувати складні завдання класифікації та регресії.

У кожному шарі багатошарового перцептрона використовується функція активації, яка застосовується до вихідних значень попереднього шару, перед тим як будуть передані наступному шару.

Тренування багатошарового перцептрону зазвичай здійснюється за допомогою методу зворотного поширення помилки (backpropagation). Під час цього процесу, мережа намагається зменшити помилку між передбаченими та очікуваними вихідними значеннями за допомогою оптимізаційних алгоритмів, таких як градієнтний спуск [18].

Багатошаровий перцептрон може бути використаний для розв'язання різних завдань машинного навчання, таких як класифікація зображень, розпізнавання мови, рекомендації та багато інших. Даний перцептрон є одним з найпоширеніших типів нейронних мереж та продовжує знаходити своє застосування у багатьох сферах.

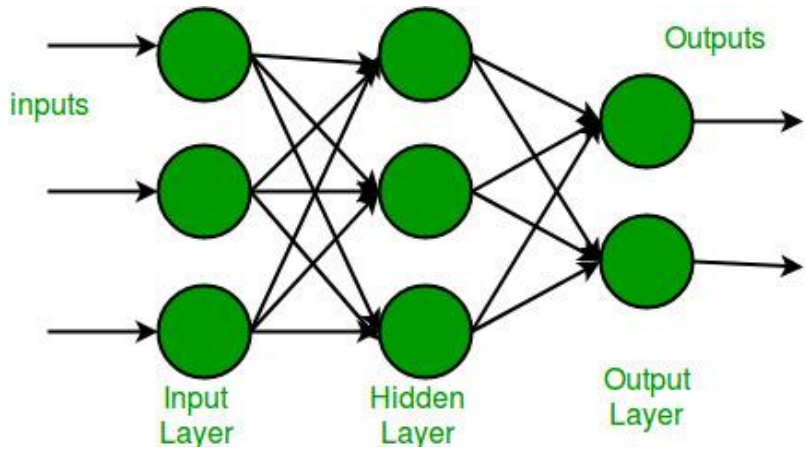


Рисунок 1.3 – багатошаровий перцептрон

Нейронна мережа прямого поширення (Feed Forward Neural Network)

Це тип нейронної мережі, яка складається з вхідного шару, одного або декількох прихованих шарів та вихідного шару. Кожен шар мережі складається з нейронів, які містять ваги та функцію активації.

Вхідний шар отримує вхідні дані та передає їх до прихованих шарів, де шари виробляються шляхом зміни вагів між нейронами. Кожен прихований шар виконує лінійні та нелінійні перетворення над вхідними даними, поки не отримається вихідний сигнал, який передається до наступного шару.

Вихідний шар отримує вихідні дані від останнього прихованого шару та виконує фінальну обробку, наприклад, класифікацію на кілька категорій або передбачення значення числа.

Нейронна мережа прямого поширення використовується для розв'язання різних задач машинного навчання, таких як класифікація, передбачення, регресія та інші. Даний вид нейронної мережі використовують для обробки різноманітних типів даних, в тому числі: зображення, звук, текст та інші. Крім того, нейронна мережа прямого поширення є базовим блоком для більш складних архітектур, таких як згорткові нейронні мережі та рекурентні нейронні мережі.

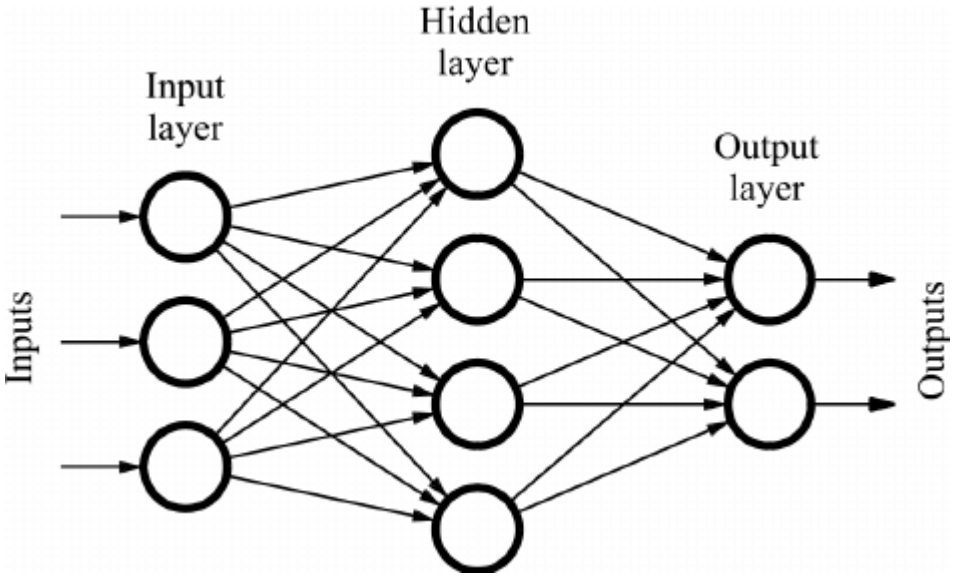


Рисунок 1.4 - Нейронна мережа прямого поширення

Згорткова нейронна мережа (Convolutional Neural Network)

На відміну від попередніх типів нейронних мереж, згорткові мережі використовують тривимірне розташування нейронів замість двовимірного розташування. Це один з найбільш популярних типів нейронних мереж, який

демонструє високу ефективність в розпізнаванні візуальних образів (відео та зображення), рекомендаційних системах та обробці мови.

- Згорткові нейронні мережі відмінно масштабуються і можуть використовуватися для розпізнавання образів, якого завгодно великого масштабу.
- У цих мережах використовуються об’ємні тривимірні нейрони. Всередині одного шару нейрони пов’язані лише невеликим полем, названі рецептивним шаром.
- Нейрони сусідніх шарів пов’язані за допомогою механізму просторової локалізації. Роботу безлічі таких шарів забезпечують особливі нелінійні фільтри, що реагують на все більше число пікселів.

Спочатку зображення з RGB перетворюється в шкалу сірого. Подальші зміни значень пікселів допоможуть виявити крайні точки, таким чином класифікуючи вхідні дані на різні класи. Даний рівень виконує функцію згортання перед передачею результату далі, тому ви можете спроектувати більш глибоку мережу з меншою кількістю вхідних параметрів, ніж у симуляції.

Комунікація в мережі може бути як однонаправленою, так і двонаправленою. В однонаправленій версії мережа має один або кілька згорткових шарів, які пізніше об’єднуються, тоді як у двонаправленій версії результати згорткових шарів надсилаються безпосередньо до нейронної мережі.

Згорткові нейронні мережі мають можливість проектувати глибоку мережу з невеликої кількості вхідних параметрів. Але до недоліків можна віднести те, що така нейромережа може бути повільною(все залежить від кількості прихованих шарів), а також те, що може бути досить складною в проектуванні.

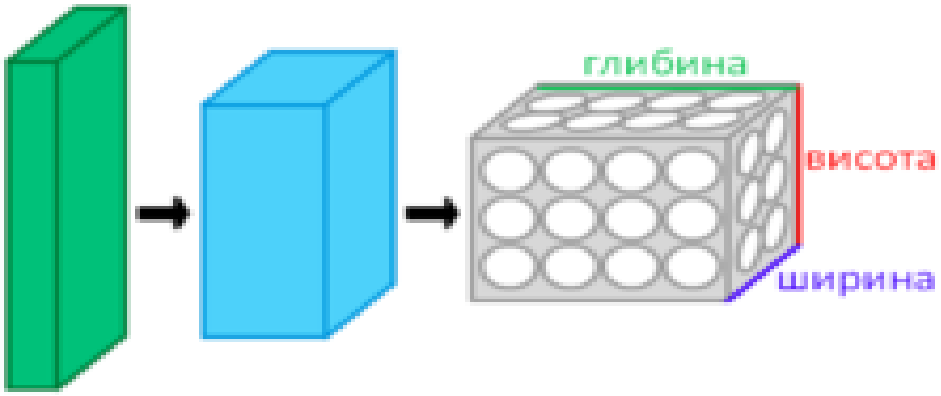


Рисунок 1.5 - Згорткова нейронна мережа

Мережа радіальних базисних функцій (Radial Basis Function Neural Network)

Нейронна мережа з радіально-базисними функціями (RBFNN) - це тип штучної нейронної мережі, яка використовує радіальні базисні функції як функції активації. Даний тип мереж зазвичай використовується для задач класифікації та регресії.

Базова структура RBFNN складається з трьох шарів: вхідного, прихованого та вихідного. Вхідний шар отримує вхідні дані і передає їх до прихованого шару, який складається з певної кількості вузлів або нейронів. Кожен нейрон у прихованому шарі пов'язаний з радіальною базисною функцією, яка відображає вхідні дані у високорозмірний простір.

Вихідний шар RBFNN складається з одного або декількох нейронів, залежно від завдання. Для завдання класифікації вихідний шар зазвичай має по одному нейрону для кожного класу, і вихід кожного нейрона представляє ймовірність того, що вхідні дані належать до цього класу. Для задачі регресії вихідний шар має лише один нейрон, а вихід представляє прогнозоване значення.

RBFNN має кілька переваг над іншими типами нейронних мереж, включаючи швидку збіжність, хорошу здатність до узагальнення та здатність

обробляти зашумлені дані. Також має деякі обмеження, такі як необхідність визначення оптимальної кількості радіальних базисних функцій, що може дорого коштувати з точки зору обчислень.

Рекурентна нейронна мережа (Recurrent Neural Network)

Рекурентними називають такі нейронні мережі, з'єднання між нейронами яких, утворюють орієнтовний цикл. Має такі характеристики:

- У кожного з'єднання є своя вага, чи пріоритет.
- Вузли поділяються на два типи, ввідні вузли і вузли приховані.
- Інформація у рекурентній нейронній мережі передається не тільки по прямій, шар за шаром, але і між самими нейронами.
- Важливою відмінною особливістю рекурентної нейронної мережі є наявність так званої «області уваги», коли машині можна задати певні фрагменти даних, що потребують посиленої обробки.

Рекурентні нейронні мережі застосовуються в розпізнаванні і обробці текстових даних.

Модульна нейронна мережа (Modular Neural Network)

Модульні нейронні мережі (МНМ) - це тип нейромережевої архітектури, що складається з декількох нейромережевих модулів, кожен з яких призначений для виконання певного завдання або підзадачі. Модульні нейронні мережі є високомодульними, тобто модулі можна легко комбінувати, замінювати або видаляти, що робить їх пристосованими до різних вимог.

Основна ідея MNN полягає у створенні архітектури нейронної мережі, яка складається з декількох менших мереж, кожна з яких може навчатися та оптимізуватися незалежно. Це дозволяє підвищити ефективність навчання, оскільки модулі можуть навчатися на різних наборах даних або за допомогою

різних методів оптимізації. Крім того, такий підхід полегшує масштабування мережі, додаючи нові модулі або збільшуючи розмір існуючих модулів.

Однією з головних переваг такої мережі є здатність вирішувати складні завдання, які вимагають скоординованого виконання кількох підзадач.

Ще однією перевагою МНМ є їхня здатність адаптуватися до різних вимог. Додаючи або видаляючи модулі, або модифікуючи самі модулі, її можна легко адаптувати до нових завдань або до змін у вхідних даних. Це робить МНМ дуже гнучкою та універсальною і дозволяє використовувати її у широкому спектрі застосувань.

Загалом, MNN є перспективним підходом до проектування нейронних мереж і має потенціал докорінно змінити спосіб використання нейронних мереж на практиці. Роблячи нейронні мережі більш модульними та адаптивними, MNN можуть допомогти поліпшити їхню продуктивність і зробити їх більш доступними для ширшого кола користувачів. На рисунку 1.6 зображено МНМ.

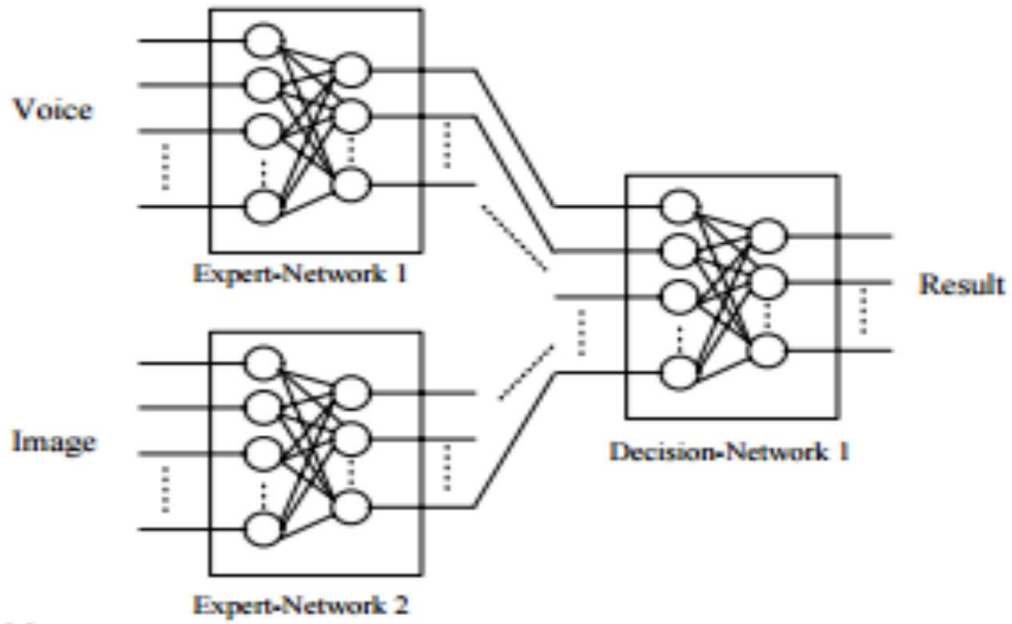


Рисунок 1.6 Модульна нейронна мережа

1.3 Класифікація зображень

Класифікація об'єкта є базовою задачею в машинному навчанні і призвана відповісти на запитання “Що саме знаходиться на зображенні?”.

Наприклад, на рисунку 1.7 можна побачити, що це автомобіль. Ми побачили зображення, та класифікували клас, до якого воно належить. При цьому кожне зображення можна класифікувати до певної кількості n категорій.

Рисунок 1.7 Демонстрація роботи classification

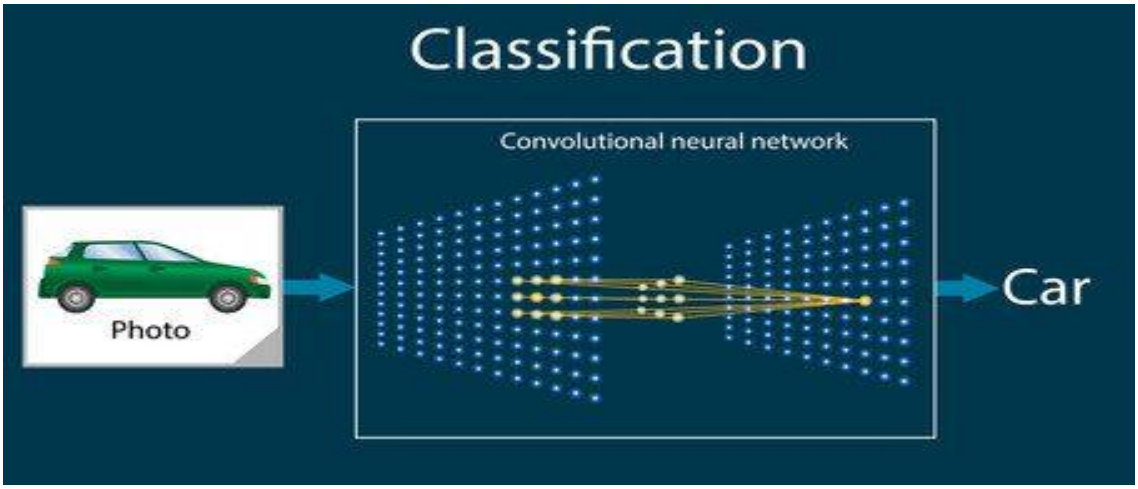


Рисунок 1.8 Демонстрація роботи classification

Передбачення розташування об'єкта на зображенні називається локалізацією об'єкта. Суть локалізації полягає в тому, що ми знаходимо розташування певного об'єкта на зображенні. Частіше всього ми отримуємо 4 точки, за допомогою яких можна визначити прямокутну область(bounding box), в якій знаходиться об'єкт [13].

Саме таку задачу виконуємо за допомогою object detection. Тобто виконується локалізація об'єктів на зображенні з визначеними раніше типу чи класу.

Instance segmentation (сегментація об'єктів) є досить схожою до виявлення об'єктів, але все таки має велику різницю. Замість пошуку 4 точок об'єкта чи bounding box, в даному методі знаходиться контур об'єкта, тим саме ми точніше визначаємо розташування об'єкта на зображенні, а також його точний контур.

На рисунку 1.9 наведена демонстрація перераховних характеристик.

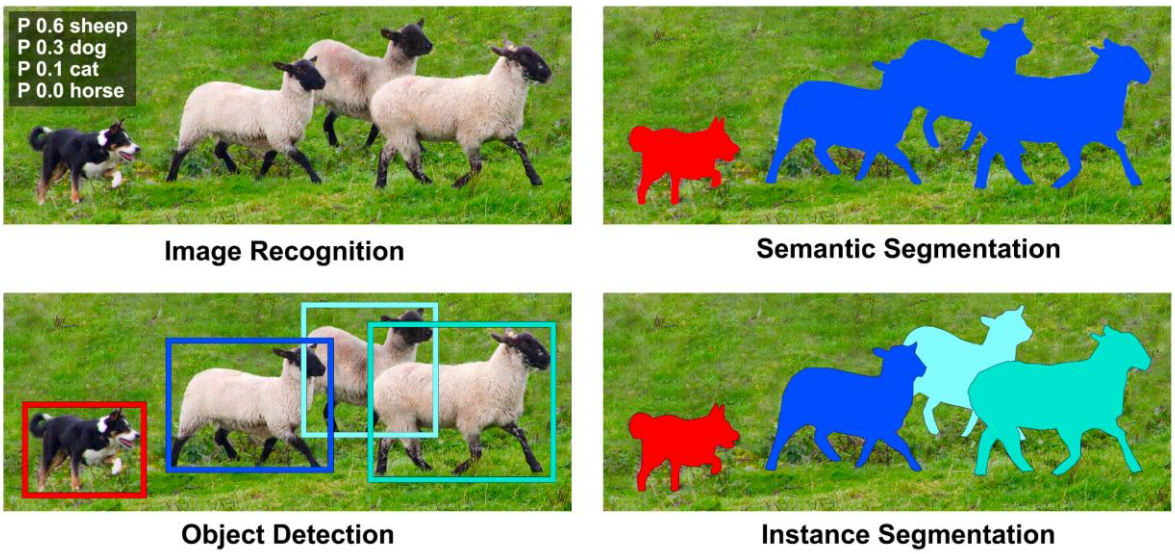


Рисунок 1.9 Recognition, Segmentation Detection

1.4 Архітектури на основі згорткових нейронних мереж

CNN - це тип нейронної мережі, який використовується для обробки даних зображення. CNN включає в себе послідовність згорткових шарів, які виявляють локальні функції на зображенні, та пулінгові шари, які зменшують розмір виходу зі згорткового шару. Після цього використовуються повнозв'язні шари, які об'єднують отримані ознаки та роблять прогнози для кінцевих класів.

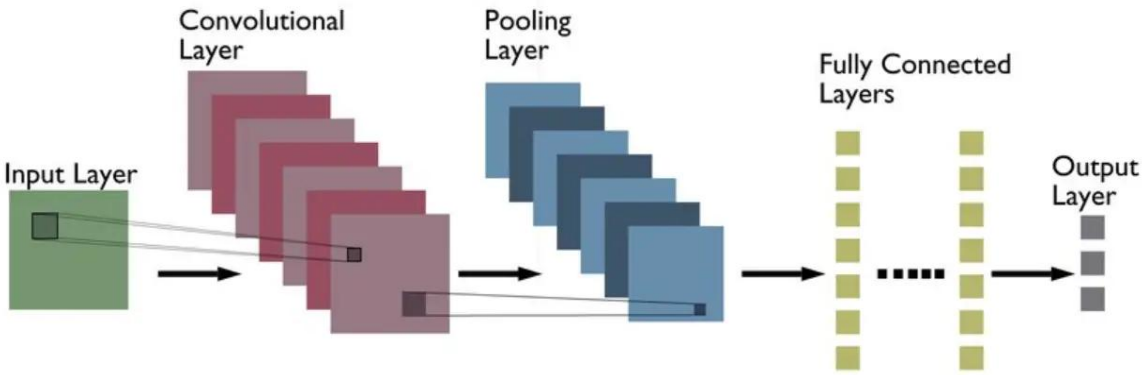


Рисунок 1.10 Стандартна архітектура CNN

Нижче наведено визначення різних шарів, показаних у наведеній архітектурі:

Згортковий шар(Convolutional layer) - це основний компонент згорткових нейронних мереж. Кожен згортковий шар складається з кількох фільтрів, кожен з яких складається з вагових коефіцієнтів. Згортковий шар пропускає вхідні дані через фільтри, щоб виділити різні функції або ознаки зображень, наприклад, ребра або кути. Вагові коефіцієнти фільтрів навчаються в процесі навчання моделі.

Об'єднувальний шар (Pooling layer) - це шар, що використовується для зменшення розмірності вихідних даних після згорткового шару. Під час пулінгу використовуються різні методи зведення, наприклад, максимальне або середнє

значення, для обчислення нових значень зі згорткового шару. Це допомагає зменшити кількість параметрів у моделі та ускладнити перенавчання.

Повністю з'єднаний шар (Fully connected layer) - це шар, в якому кожен нейрон пов'язаний з кожним нейроном попереднього шару. У повнозв'язному шарі використовуються вагові коефіцієнти та зсуви, які навчаються в процесі навчання моделі. Повнозв'язний шар використовується для остаточного класифікування зображень або інших вхідних даних за допомогою вихідного шару, що містить кількість нейронів, рівну кількості класів, які модель може передбачити.

Кожен нейрон у повнозв'язному шарі приймає вхідний вектор ознак з попереднього шару і виконує лінійну комбінацію з вагами та додаванням зміщення. Результат від цього додавання подається на вхід функції активації (наприклад, сигмоїд або ReLU), щоб отримати вихід нейрону. Повнозв'язний шар може мати декілька таких нейронів, і кожен з них виконує аналогічні обчислення.

Існує кілька популярних архітектур на основі згорткових нейронних мереж, кожна з яких має свої унікальні особливості та переваги [1]. Деякі з найвідоміших архітектур CNN включають:

LeNet-5 є однією з перших згорткових нейронних мереж (CNN), розробленою для розпізнавання рукописних цифр. Мережа складається з 7 шарів, які включають згорткові, пулінгові та повнозв'язні шари. Ця архітектура є одною з найстаріших і стала шаблоном для архітектур, які створювалися пізніше. Архітектура LeNet-5 зображена на схемі нижче.

В даний час LeNet-5 може використовуватись як стартова точка для розвитку більш складних згорткових нейронних мереж, а також для використання в задачах, де потрібно виконувати класифікацію зображень. Також, варіації LeNet-5 були використані для розв'язання інших задач, таких як розпізнавання облич та розпізнавання рукописного тексту.

Хоча модель LeNet-5 була розроблена понад 20 років тому, модель все ще залишається актуальною, оскільки деякі з її особливостей, таких як згорткові та пулінгові шари, є основоположними блоками сучасних згорткових нейронних мереж.

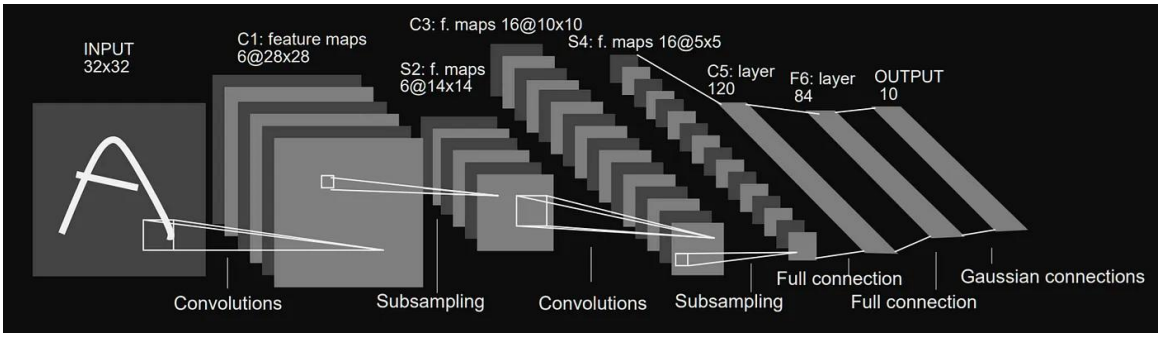


Рисунок 1.10 Архітектруа LeNet-5

AlexNet - це архітектура згорткової нейронної мережі (CNN), яка була розроблена 2012 році. Ця архітектура стала переможцем конкурсу ImageNet і першою глибокою нейронною мережею, яка досягла значного покращення точності класифікації зображень на наборі даних. При цьому також має 7 шарів як і LeNet-5. Являється першою архітектурою, яка реалізує випрямлену лінійну одиницю (ReLU) як функцію активації. Загалом, AlexNet має близько 60 мільйонів параметрів, що робить її відносно великою мережею для свого часу. Тому змогла досягти найсучасніших результатів на наборі даних ImageNet, і її успіх проклав шлях для подальшого прогресу в глибокому навчанні [8].

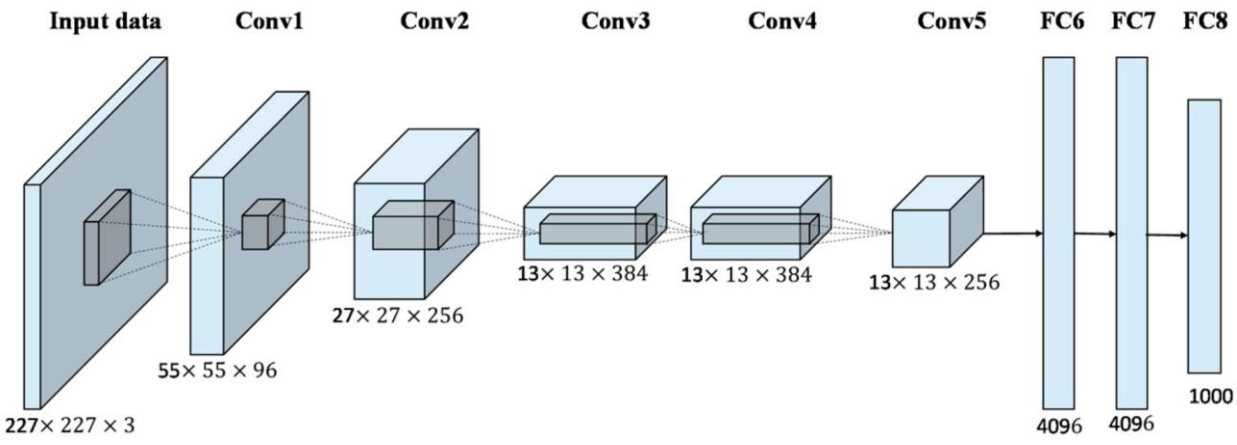


Рисунок 1.11 Архітектура AlexNet

Аналізуючи попередні архітектури, можна зробити висновок, що для покращення продуктивності нейронних мереж можна просто збільшити їхній розмір. Так у 2014 році в Оксфордському університеті групою візуальної геометрії (VGG) була розроблена архітектура *VGG-16*. На данний момент данна архітектура є однією з найбільш широко використовуваних архітектур нейронних мереж для класифікації зображень і розпізнавання об'єктів. Мережа складається з 16 шарів, які включають згорткові та повнозв'язні шари і використовує ReLU, що був розглянутий в архітектурі AlexNet. Загалом, VGG-16 має близько 138 мільйонів параметрів, що робить її відносно великою мережею. Також відома своєю простотою і, дає чудові результати для різноманітних завдань класифікації зображень [2].

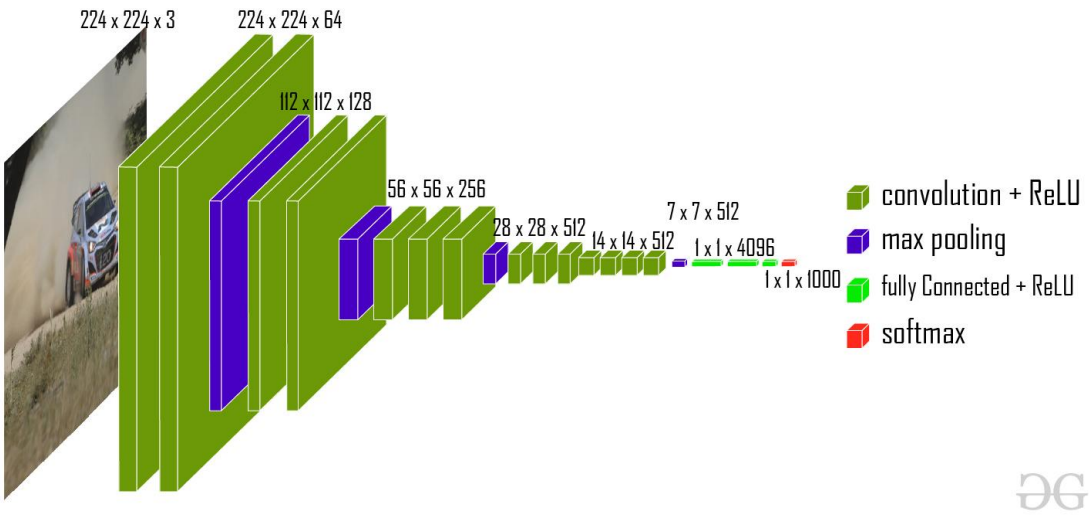


Рисунок 1.12 Архітектура VGG-16

Ще однією архітектурою є *Inception*, також відома як GoogLeNet, - це архітектура згорткової нейронної мережі (CNN), яку розробили дослідники Google у 2014 році. Архітектура відома своєю здатністю досягати високої точності з відносно невеликою кількістю параметрів, що робить її ефективною для реальних застосувань [17].

Суть полягає у використанні згорткових блоків 1x1 (NiN) для зменшення кількості властивостей перед подачею в "дорогі" паралельні блоки. Зазвичай цю частину називають bottleneck.

Inception містить кілька блоків, які дозволяють виконувати згортки різної розмірності та об'єднувати їх у єдиний потік обробки зображення. У кожному блоці використовуються різні типи згорток: 1x1, 3x3 та 5x5. Це дозволяє архітектурі Inception забезпечити більшу ефективність у порівнянні з традиційними згортковими мережами, такими як AlexNet і VGG.

Крім того, Inception містить так звані блоки "збільшення каналів", які дозволяють додавати додаткові функції у мережу, підвищуючи точність розпізнавання зображень. Також Inception використовує техніку "global average pooling", що дозволяє зменшити кількість параметрів мережі та зменшити ризик перенавчання.

Загалом, архітектура Inception має велику кількість параметрів (понад 20 мільйонів для Inception v3), але здатна досягати найсучасніших результатів у різноманітних задачах класифікації зображень, використовуючи меншу кількість параметрів, ніж інші популярні архітектури CNN.

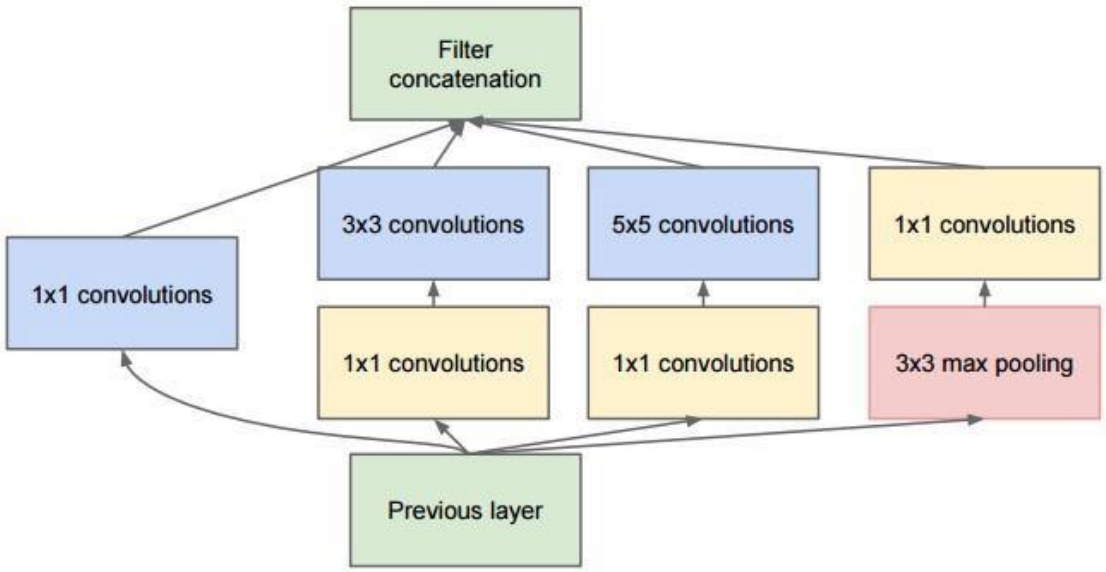


Рисунок 1.13 Архітектура модуля Inception

1.5 Висновки до розділу

Розглянуто класифікацію нейронних мереж та їх види, в першу чергу, згорткових нейронних мереж. Наведено опис архітектур згорткових нейронних мереж, зокрема AlexNet, VGG, LeNet та Inception. Кожна з цих архітектур має свої переваги та недоліки, що потрібно враховувати при їх використанні для різних завдань.

При виборі виду нейронної мережі, для розв'язання задачі, визначальним фактором є призначення кожної мережі. Для розв'язання задачі найбільше підходить згорткова нейронна мережа. Даний вид нейронних мереж є потужним інструментом для розв'язання задач обробки зображень, зокрема, для класифікації зображень, адже складаються з кількох шарів, кожен з яких складається з набору фільтрів (які виконують операції згортки та підсумовування), що дозволяють вилучати корисні ознаки зображення.

РОЗДІЛ 2

ВИБІР АРХІТЕКТУРИ ТА ЗАСОБІВ РЕАЛІЗАЦІЇ ANDROID-ЗАСТОСУНКУ

2.1 Огляд фреймворків.

Для реалізації мобільного застосунку спочатку потрібно визначитися з мовами програмування та фреймворками які будуть використовуватися. При виборі мови програмування потрібно приділити увагу наявності різних доступних фреймворків для розробки застосунку, складність реалізації, масштабування та подальшої підтримки застосунку. Такі вимоги дозволяють обрати найоптимальніше рішення для реалізації застосунку, адже різні фреймворки краще працюють для різних поставлених задач. Важливо щоб створений застосунок працював оптимально на різних пристроях та різних версіях ОС Android. Також важливу роль потрібно приділити роботі з камерою Android пристрою та файловою системою. На основі цього прийнято рішення розглянути популярні бібліотеки та фреймворки, що є відкритим програмним забезпеченням.

Android SDK - це набір інструментів, бібліотек та документації для розробки застосунків для платформи Android. Даний SDK містить інструменти для розробки, тестування та налагодження застосунків, бібліотеки, емулятори та документацію [16].

Основними цілями Android SDK є:

- Розробка застосунків для платформи Android;
- Розробка та налагодження Android-програм на таких мовах програмування, як Java або Kotlin;
- Розробка власних бібліотек та іншого програмного забезпечення для Android.

Переваги Android SDK:

- Надає широкі можливості для розробки застосунків на платформі Android;
- Дозволяє розробникам створювати різні типи застосунків, від простих до складних;
- Надає зручний інтерфейс для тестування застосунків та документацію з відкритим вихідним кодом;
- Дозволяє розробникам легко інтегрувати застосунки з іншими сервісами Google, такими як Google Maps, Google Drive тощо.

Недоліки Android SDK:

- Для повноцінної розробки застосунків необхідно володіти різними інструментами та технологіями;
- Підтримка деяких функцій SDK може бути обмежена на старих версіях Android.

Xamarin - це платформа розробки програмного забезпечення, яка дозволяє створювати кросплатформні застосунки для різних операційних систем, таких як iOS, Android та Windows, використовуючи мову програмування C# та середовище розробки Visual Studio [5].

Основні переваги Xamarin:

- Кросплатформність: дозволяє розробляти застосунки для різних операційних систем, що зменшує час та зусилля, необхідні для створення окремих застосунків для кожної платформи;
- Використання мови програмування C#: ця мова є дуже популярною серед розробників, що забезпечує більшу кількість інструментів, документації та підтримки спільноти;

- Більша ефективність: Xamarin дозволяє використовувати бібліотеки та інструменти, які використовуються в стандартному програмуванні на кожній з платформ;
- Підтримка інтеграції: Xamarin забезпечує зручну інтеграцію з популярними сервісами, такими як Microsoft Azure, SQLite, та інші.

Основні недоліки Xamarin:

- Обмежена підтримка сторонніх бібліотек: на відміну від нативного програмування на кожній платформі, Xamarin може мати обмежену підтримку сторонніх бібліотек;
- Розмір застосунків: застосунки, розроблені за допомогою Xamarin, можуть бути більшими в розмірі порівняно з застосунками, розробленими нативним способом;
- Швидкість: застосунки, розроблені за допомогою Xamarin, можуть мати більшу витрату ресурсів порівняно з нативними застосунками.

React Native - це кросплатформний фреймворк для розробки мобільних застосунків на базі JavaScript та React.js. React Native дозволяє розробляти застосунки для Android та iOS, використовуючи єдиний код [9].

Переваги React Native:

- Швидкість розробки: завдяки використанню єдиного коду для різних платформ, розробники можуть зосередитися на функціональності застосунка, а не на деталях різних платформ.
- Простий для розуміння: для розробки застосунків на React Native потрібно знати JavaScript та React.js, що робить його доступним для широкого кола розробників.
- Висока продуктивність: React Native дозволяє створювати нативні застосунки з високою продуктивністю та швидкістю роботи.

- Можливість використання сторонніх бібліотек: React Native має велику кількість сторонніх бібліотек та компонентів, що можна використовувати для розробки застосунків.
- Активна спільнота розробників: React Native має активну спільноту розробників, яка надає багато різноманітних бібліотек та рішень для розробки застосунків. Також допомагає розробникам у вирішенні проблем та підтримці застосунків.

Недоліки React Native:

- Обмежена підтримка певних функцій: React Native може не мати підтримки для деяких функцій, що доступні на нативних платформах.
- Високі вимоги до ресурсів: React Native вимагає більш високих ресурсів, ніж деякі інші кросплатформні фреймворки, що може призвести до погіршення продуктивності.
- Обмежені можливості настройки: React Native не дозволяє налаштовувати деякі аспекти мобільного застосунку, такі як віджети на головному екрані або робочий стіл.
- Обмежена підтримка певних функцій Android: React Native може не мати підтримки для деяких функцій, що доступні на нативній Android-платформі. Це може призвести до обмеження можливостей застосунку та зниження його продуктивності.
- Проблеми з взаємодією зі сторонніми бібліотеками: Деякі сторонні бібліотеки, які розроблені для Android-платформи, можуть не працювати з React Native. Це може призвести до обмеження можливостей застосунку або до необхідності змінювати код бібліотеки для його використання з React Native.
- Проблеми з продуктивністю: React Native може вимагати більше ресурсів, ніж деякі інші фреймворки, що може призвести до погіршення продуктивності застосунку. Деякі застосунки можуть бути повільними

або мати проблеми зі споживанням енергії на Android-платформі через використання React Native.

- Недостатні можливості настройки: React Native не дозволяє налаштовувати деякі аспекти мобільного застосунку, такі як віджети на головному екрані або робочий стіл, що може бути необхідним для деяких Android-застосунків.
- Проблеми з сумісністю: React Native може мати проблеми зі сумісністю з певними версіями Android-платформи. Це може призвести до проблем з розгортанням та використанням застосунку на певних пристроях або версіях Android.

Flutter - це відкрита та безкоштовна платформа розробки мобільних застосунків та веб-застосунків, створена компанією Google на базі мови програмування Dart. Завдяки вбудованій бібліотеці матеріального дизайну в Flutter можна легко створювати застосунки з гарним інтерфейсом користувача [14].

Основні переваги Flutter для розробки мобільних застосунків на Android:

- Швидкість розробки: Flutter дозволяє розробляти застосунки швидко завдяки широкій бібліотеці готових компонентів та інструментів, таких як гаряча перезавантаження, яке дозволяє розробникам візуально відстежувати зміни під час розробки.
- Висока продуктивність: Flutter надає високу продуктивність завдяки використанню свого власного движка рендеринга, який працює безпосередньо з апаратними можливостями пристроїв. Це дозволяє зменшити час відгуку застосунка та забезпечити швидке відображення графічних елементів.
- Кросплатформена підтримка: Flutter дозволяє розробляти застосунки для різних платформ, включаючи Android, iOS, Web та Desktop, за допомогою одного коду. Це зменшує час та зусилля, необхідні для розробки застосунків на різних платформах.

- Гнучкість та налаштування: Flutter дозволяє розробникам налаштовувати та масштабувати застосунки згідно зі своїми потребами. Також дозволяє використовувати сторонні бібліотеки та розширення для розширення функціональності застосунка.
- Гарний UI: Flutter надає високу якість візуальної графіки та анімації, що робить застосунки більш привабливими для користувачів. Завдяки вбудованій бібліотеці матеріального дизайну можна легко створювати застосунки зі схожим дизайном на різних платформах.

Основні недоліки Flutter для розробки мобільних застосунків на Android:

- Розмір застосунків: Один з найбільших недоліків Flutter - це розмір застосунків. Для розробки застосунків потрібно включити в проект весь необхідний код та бібліотеки, що може збільшити розмір застосунку.
- Не всі можливості Android SDK: Незважаючи на те, що Flutter надає багато можливостей для розробки застосунків, не включає всі можливості Android SDK. Якщо для застосунка потрібні конкретні можливості Android, можуть знадобитися додаткові зусилля для їх реалізації.
- Відсутність готових рішень для певних завдань: Незважаючи на широкий спектр бібліотек та інструментів, що надає Flutter, іноді можуть виникнути завдання, для яких немає готового рішення. У такому випадку розробнику доведеться розробляти рішення самостійно.
- Проблеми з взаємодією зі сторонніми бібліотеками: Flutter може мати проблеми зі взаємодією зі сторонніми бібліотеками, що може призвести до проблем зі стабільністю та продуктивністю застосунку.

Apache Cordova - це відкрите програмне забезпечення для розробки мобільних застосунків з використанням веб-технологій, таких як HTML, CSS та JavaScript. Cordova дозволяє розробникам створювати мобільні застосунки, які можна запустити на різних платформах, включаючи Android, iOS, Windows та більшість інших мобільних платформ.

Основні переваги Apache Cordova:

- Багатоплатформність: За допомогою Cordova можна створювати мобільні застосунки для різних платформ, включаючи Android, iOS та Windows.
- Використання веб-технологій: Розробка застосунків з використанням веб-технологій, таких як HTML, CSS та JavaScript, дозволяє розробникам використовувати знайомі інструменти та мови програмування.
- Наявність бібліотек та інструментів: Cordova надає багато готових бібліотек та інструментів для розробки мобільних застосунків.
- Підтримка плагінів: Cordova дозволяє додавати плагіни, що розширюють можливості застосунків.

Недоліки Apache Cordova:

- Швидкість та продуктивність: застосунки, розроблені з використанням Cordova, можуть бути менш продуктивними, ніж застосунки, розроблені з використанням нативних мов програмування.
- Обмежені можливості: Cordova не надає доступ до всіх можливостей пристрою, що може бути обмеженням для деяких типів застосунків.
- Залежність від зовнішніх бібліотек: Для розробки застосунків з використанням Cordova можуть знадобитися зовнішні бібліотеки, що можуть створювати проблеми зі сумісністю.

Apache Cordova може бути відмінним вибором для розробки мобільних застосунків, якщо ви знайомі з веб-технологіями та хочете створити багатоплатформний застосунок з використанням знайомих інструментів та мов програмування. Завдяки Cordova можна зекономити час та зусилля при розробці мобільного застосунку, оскільки не потрібно розробляти окремі версії для різних платформ.

Крім того, Cordova має активну спільноту розробників, яка надає підтримку та допомогу при виникненні проблем під час розробки застосунків.

Cordova не є ідеальним вибором для всіх типів мобільних застосунків. Якщо необхідні повні доступ до функцій пристрою, максимальна продуктивність та швидкість, то Cordova може не задовольнити вимоги. Крім того, можуть виникнути проблеми із сумісністю з зовнішніми бібліотеками та іншими інструментами, що може затримати розробку та збільшити складність проекту.

Тому, розглядаючи всі вище перелічені фреймворки, обрано використання Android SDK, для подальшої реалізації Android - застосунку. Android SDK основним інструментом для розробки, має велику кількість бібліотек для зручної роботи з різними сенсорами в Android. Також має величезне community. Основними мовами є Java та Kotlin, які завичай працюють дуже швидко. Ці мови є простими для розуміння та вивчення.

2.2 Огляд мов програмування.

Java та Kotlin є двома популярними мовами програмування для розробки Android-застосунків, відомі своїми функціональними можливостями. Обидві мови мають свої переваги та недоліки, про які потрібно знати на етапі підготовки до розробки [6].

Kotlin - це мова програмування, розроблена компанією JetBrains у 2010 році, з відкритим вихідним кодом. Kotlin можна використовувати в проектах з розробки застосунків для Android, iOS, а також десктопних застосунків. Kotlin забезпечує сумісність з Java, а це означає, що можна використовувати наявні навички та знання Java; це також дозволяє використовувати Kotlin для проектів з розробки застосунків для Android.

Основні можливості Kotlin:

- Null Safety: Kotlin є нуль-безпечним, що запобігає помилкам, які зазвичай спричинені відсутністю типів або неправильно ініціалізованими параметрами.
- Типи, які можна занулити: У Kotlin немає потреби в "null" або "nil". Це запобігає помилкам, спричиненим відсутніми типами чи параметрами, а також робить код більш читабельним.
- Властивості з автоматично створеними геттерами та сеттерами: Kotlin постачається з властивостями, які автоматично генерують геттери та сеттери.
- Стислий синтаксис: Немає зайвих ключових слів, що допомагає зробити код лаконічним і легким для сприйняття розробниками.

Java була створена компанією Sun Microsystems у 1995 році. Є об'єктно-орієнтованою, імперативною та паралельною з автоматичним керуванням пам'яттю. Java можна використовувати для проектів з розробки застосунків для Android, а також десктопних застосунків, але для цього потрібно імпортувати java-класи, що робить код дещо складнішим, ніж у Kotlin.

Основні особливості Java:

- Автоматичне очищення пам'яті: Завдяки автоматичному управлінню пам'яттю розробникам не потрібно турбуватися про виділення та звільнення пам'яті у своєму коді, оскільки Garbage collector (збирач сміття) подбає про це автоматично.
- Підтримка різних платформ: Java підтримує декілька платформ, що означає, що розробники можуть почати кодування на Java, а потім перейти на Kotlin для проектів з розробки застосунків для Android.

Порівняння обох мов розробки застосунків для Android

- Простота у використанні: Kotlin набагато простіший у використанні та вивченні, ніж Java. Також має хорошу колекцію інструментів, IDE, а

також навчальних посібників, що полегшує розробку мобільних застосунків.

- Продуктивність: Java - це мова перевірена часом з відмінною продуктивністю. З іншого боку, Kotlin працює краще, ніж Java для розробки Android завдяки використанню незмінності та властивостей, а також 100% сумісності з Java.
- Популярність: Kotlin - це нова мова, але швидко стала однією з найпопулярніших для розробки застосунків для Android. Крім того, ця мова підтримується Google, а це означає, що з часом Kotlin буде отримувати все більше підтримки та вдосконалюватися. Але якщо говорити про поточну ситуацію, то Java популярніша за Kotlin.
- Кросплатформена розробка: Kotlin має можливість кроскомпіляції та запуску на декількох платформах, в той час як Java обмежена використанням байт-коду, який може бути скомпільований лише для однієї платформи одночасно. У мобільній розробці є багато випадків, коли розробникам потрібно створити версію для Android, а також версії для iOS або навіть для Windows.
- Наявність різноманітних бібліотек: Java має величезну колекцію бібліотек, фреймворків та інструментів, доступних для розробки під Android. Kotlin починає наздоганяти новіші бібліотеки, такі як KTX та Coroutines, але все ще має багато чого наздоганяти в цій сфері.
- Масштабованість: Одним з недоліків використання таких мов, як Java, є те, що не настільки масштабовані. Через це застосунок може іноді роздуватися і впливати на продуктивність, особливо на старих пристроях, таких як Android смартфони. В той час як у Kotlin більше уваги приділяється масштабованості, що покращує продуктивність програми за рахунок зменшення роздуття.
- Підтримка спільноти: Kotlin є відносно новою мовою, в той час як Java існує вже досить давно. Таким чином, існує більше експертів та ресурсів для розробки, з якими можна проконсультуватися при

використанні Kotlin, ніж Java, що призвело до того, що в останні роки стала кращим вибором для багатьох компаній.

Враховуючи особливості кожної мови програмування, обрано Kotlin, адже масштабованість та синтаксичні правила цієї мови програмування, а також є офіційною мовою для розробки мобільних застосунків на платформі Android, що означає, що Kotlin має широку підтримку та активну спільноту розробників. Це робить його більш привабливим вибором для тих, хто хоче розробляти мобільні застосунки для платформи Android.

2.3 Огляд засобів реалізації аналізу облич

Open CV (Open Source Computer Vision Library) - відкрите програмне забезпечення для розробки програм комп'ютерного зору і машинного навчання. OpenCV була розроблена з метою створення універсальної бібліотеки, яка може бути використана для вирішення різноманітних задач комп'ютерного зору. OpenCV забезпечує реалізацію багатьох алгоритмів і функцій, пов'язаних з обробкою зображень, включаючи: зчитування та запис зображень і відео, обробку зображень (фільтри, згладжування, контурні детектори, тощо), виявлення та відстеження об'єктів в режимі реального часу, виявлення та розпізнавання облич та інших об'єктів [12];

Бібліотека була розроблена в Intel в 1999 році і на сьогоднішній день є відкритим програмним забезпеченням. OpenCV була розроблена з використанням мов програмування C і C++, і підтримується на різних платформах, таких як Windows, Linux, Android і Mac OS. Крім того, OpenCV може бути використана з такими мовами програмування, як Python, Java, Matlab та інші. Це робить OpenCV дуже популярною у галузі машинного навчання.

Виявлення облич в Open CV працює за допомогою каскадного класифікатора. Тобто результуючий класифікатор складається з кількох простіших класифікаторів. Для кожного виконується швидкий тест, і якщо зображення проходить даний тест, то далі виконується наступний, більш

детальний тест і тд. Алгоритм має близько 30-50 таких каскадів, а обличчя виявляється тільки в разі успішного проходження всіх тестів.

Перевагою такого методу є те, що при проходженні такого тесту, якщо тест не вдалий, то повернеться негативний результат ще на ранніх етапах, що дозволить не витрачати час на всі наступні етапи. Це дозволяю використовувати розпізнавання облич в реальному часі.

Самі каскади це просто набір XML файлів, що містять певні дані Open CV. Тобто ми ініціалізуємо код потрібним каскадом, а потім виконає роботу за нас.

Також розпізнавання облич можна використовувати без інтернет з'єднання. Всі обчислення відбуваються локально. Проте з цього випливає мінус в тому, що самі класифікатори і натреновані моделі повинні зберігатись в памяті Android пристрою, що в рази збільшує розмір розмір APK файлу для програми.

Також в даній бібліотеці можна використовувати свій власний кастомний набір даних.

Firebase ML Kit – це потужний інструмент вбудовуваних функцій розпізнавання облич у мобільні застосунки. Являється продуктом Firebase - платформи для мобільної розробки, створеної компанією Google. Firebase ML Kit надає широкий спектр готових моделей машинного навчання, які можна легко інтегрувати у мобільний застосунок, включаючи виявлення, розпізнавання та відстеження облич [7].

ML Kit спрощує застосування методів ML у програмах, об’єднуючи технології ML від Google, такі як Google Cloud Vision API, Tensor Flow Lite та Android Neural Networks API, в одному SDK.

ML Kit може продовжувати працювати навіть без інтернет з'єднання, хмарної обробки. Також є можливість роботи в режимі реального часу моделей,

що є оптимізовані для мобільних пристроїв, адже хмарна обробка відбувається за рахунок Google Cloud, що значно прившивдшує швидкість роботи.

Також є можливість роботи з власними існуючими моделями TensorFlow Lite, якщо доступні реалізації не охоплюють існуючу проблему. Для запуску потрібно лише завантажити модель в Firebase, адже ML Kit спрощує роботу з запуском та використанням власної моделі.

Розпізнавання облич на основі ML Kit працює достатньо швидко та точно, що спрощує роботу з пошуком іншої моделі Tensor Flow Lite для тренування. TensorFlow Lite оптимізовано для машинного навчання на пристрої за рахунок вирішення 5 ключових обмежень: затримки (немає зворотного зв'язку з сервером), конфіденційності (особисті дані не залишають пристрій), підключення (підключення до Інтернету не потрібне), розмір (зменшена модель і двійковий розмір) і енергоспоживання (ефективний висновок і відсутність мережевих з'єднань).

Розпізнавання облич в ML Kit реалізовано по принципу object detection з 4 точками (bounding box).

Розмір ML Kit для розпізнавання менший за Open CV, а при створенні APK файлу та правильної обфускації можна досягти розмірів середньостатистичного застосунку з Play Store.

PyTorch - це бібліотека машинного навчання з відкритим вихідним кодом для мови програмування Python. Бібліотека розроблена на базі бібліотеки Torch, яка була створена для мови програмування Lua [11].

PyTorch базується на концепції тензорів, що представляють багатовимірні матриці, та використовує графи обчислень для ефективного обчислення градієнтів, необхідних для здійснення зворотного поширення помилки під час навчання нейронних мереж.

PyTorch надає простий і зручний інтерфейс для створення, тренування та застосування нейронних мереж. Бібліотека дозволяє швидко експериментувати

з різними архітектурами мереж, використовувати передові методи оптимізації та навчання з підсиленням, а також працювати з великими наборами даних.

PyTorch також має ряд інструментів для візуалізації та налагодження нейронних мереж, що дозволяє аналізувати та вдосконалювати їхню роботу.

Одним з головних переваг PyTorch є можливість використовувати автоматичне диференціювання, що робить його особливо підходящим для задач глибинного навчання. Крім того, PyTorch має активну спільноту користувачів та розробників, що підтримує розвиток бібліотеки та допомагає вирішувати проблеми.

PyTorch має деякі недоліки. Зокрема, в порівнянні з іншими бібліотеками машинного навчання, такими як TensorFlow, PyTorch має меншу швидкість роботи та використовує більше ресурсів системи. Крім того, на відміну від TensorFlow, PyTorch не має вбудованої можливості для розподіленого навчання на кластерах.

2.4 Висновки до розділу

Розглянуто фреймворки для реалізації мобільного застосунку для платформи Android:

- Android SDK
- Xamarin
- React Native
- Flutter
- Apache Cordova

За результатами огляду переваг та недоліків кожного фреймворку, для реалізації Android-застосунку найоптимальнішим рішенням є Android SDK. Адже використання даного фреймворку дозволяє створювати більш швидкі (нативний код є оптимізованим для платформи Android, а також використовується на нижчих рівнях абстрації) та продуктивні (

кросплатформені фреймворки можуть містити затримки з роботою анімації, збільшити нагрузку на аккумулятор девайсу, на відміну від нативних рішень) застосунки, в порівнянні з кросплатформеними фреймворками.

Мовою програмування для даного застосунку є Kotlin. В порівнянні з Java Kotlin є більш сучасною та продуктивною (оскільки має менше накладних витрат порівняно з Java, зокрема, завдяки використанню inline-функцій) мовою програмування для розробки Android-застосунків, яка дозволяє писати менше, зменшувати кількість помилок (Kotlin має більш продуману систему null safety, що дозволяє запобігати багам, пов'язаним з неправильним використанням null-значень, що може зменшити кількість помилок та забезпечити більш безпечний код.) та є більш лаконічним.

Для реалізації функціоналу з виявлення облич обрано бібліотеку ML Kit. У порівнянні з іншими існуючими засобами, варто підмітити, що ML Kit без зайвих проблем інтегрується в застосунок (основним застосуванням є мобільні пристрої), доступний у відкритому доступі, що дозволяє модифікувати його, створювати власний функціонал або використовувати власні натреновані моделі. Також моделі машинного навчання працюють локально на пристрої, не вимагаючи постійного підключення до інтернету.

РОЗДІЛ 3

РОЗРОБКА ANDROID-ЗАСТОСУНКУ

3.1 Архітектура застосунку

При створенні нового застосунку, перш за все потрібно обрати його архітектуру. Правильно вибрана архітектура дозволяє створити масштабований застосунок, який можна ефективно тестувати та відокремити його окремі компоненти.

Для реалізації поставленої задачі можна обрати MVVM, що є шаблоном проектування програмного забезпечення, який використовується для розробки в Android-програмуванні. Даний шаблон рекомендований для створення мобільних застосунків компанією Google. Архітектура ділиться на 3 основні частини [15]:

- Model - представляє дані застосунку, зазвичай це база даних, мережеві запити і т. д.
- View - надає інформацію користувачеві, тобто це екран, який відображається на пристрої.
- ViewModel - забезпечує зв'язок між Model і View. ViewModel створений для того, щоб зберігати та керувати даними, що пов'язані з View.

Структура застосунку наведена нижче у таблиці 1.

Таблиця 1

Назва папки	Призначення
di	Впровадження залежностей
extensions	Функції розширення
presentation	Екрани застосунку
system	Бізнес логіка застосунку
ui	Основні UI - елементи
utils	Спільні функції

3.2 Реалізація розпізнавання облич на зображенні

Для реалізації функціональності з розпізнавання та обробки облич розроблена використовуючи Google ML Kit. Даний фреймворк вже має існуючу натреновану модель, яка здатна виконувати поставлену задачу. Незважаючи на це, реалізація розпізнавання облич все ще потребує зусиль, адже реалізація є інкапсульованою.

Так як ML Kit є потужним інструментом для визначення облич, а також має такі існуючі функції:

- Відстеження облич у відео протягом будь-якого часу. Тобто, обличчя, виявлене в послідовних відеокадрах, можна ідентифікувати як одну і ту ж особу.
- Орієнтир - основні точки облич, такі як ліве та праве око, основа носа.
- Контур - набір точок, що повторюють форму риси облич.

- Класифікація визначає чи присутня певна характеристика обличчя. Наприклад, обличчя можна класифікувати за тим, відкриті чи закриті очі, чи посміхається людина.

На даному етапі використання контуру не є необхідною функцією при визначенню облич. Тому визначення контурів буде виключене в подальшому.

Кожне ідентифіковане обличчя повертає такий набір данних:

- Координати рамки обличчя;
- Id трекінгу обличчя (присвоюється кожному новому ідентифікованому обличчю);
- Ймовірність відкриття лівого та правого ока, а також ймовірність посмішки.

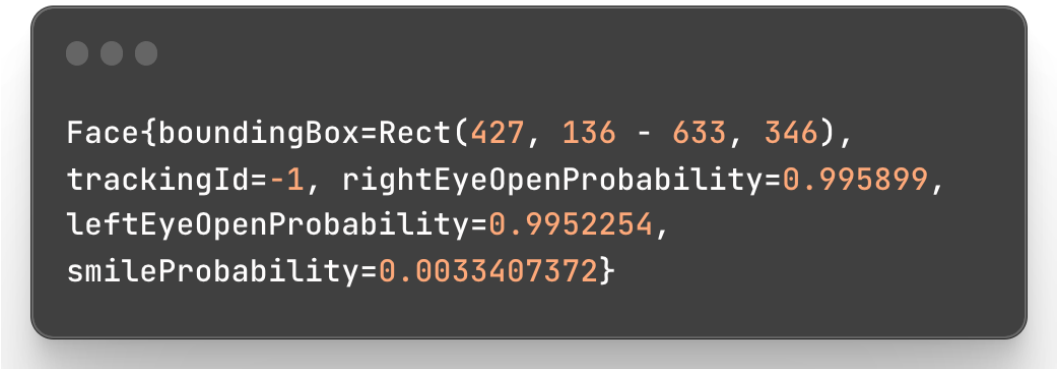


Рисунок 3.1 Приклад результату визначення облич

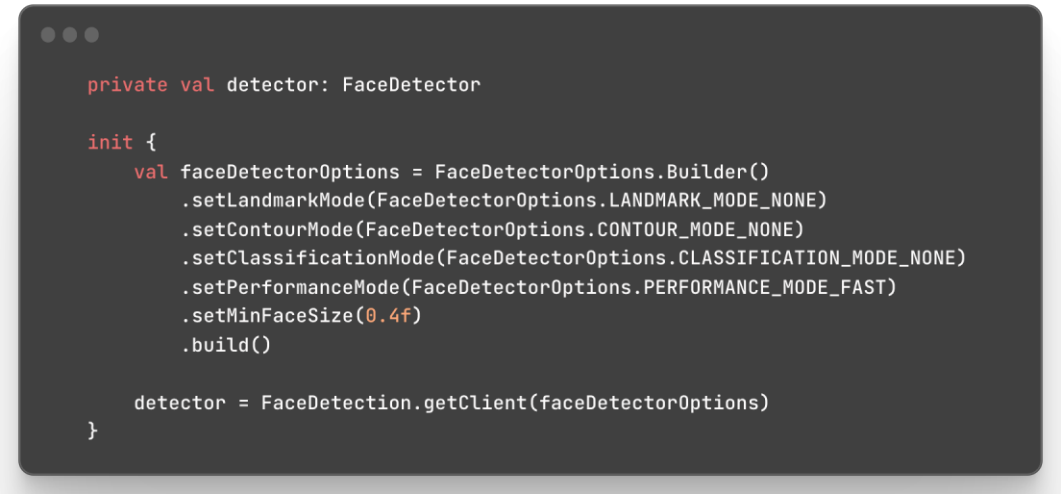
Для розпізнавання облич слід використовувати зображення розміром не менше 480x360 пікселів. Для того, щоб ML Kit міг точно розпізнати обличчя, вхідні зображення повинні містити обличчя, які представлені достатньою кількістю пікселів. Загалом, кожне обличчя повинно мати розмір не менше 100x100 пікселів. Щоб розпізнати контури облич, ML Kit вимагає більш високої роздільної здатності вхідних даних: кожне обличчя має бути не менше 200x200 пікселів.

Якщо розпізнавання облич відбувається в реальному часі, варто звернути увагу на загальні розміри вхідних зображень. Зображення меншого розміру

можуть оброблятися швидше, тому для зменшення затримок, зображення має бути з меншою роздільною здатністю, але слід пам'ятати про наведені вище вимоги до точності і переконатися, що обличчя об'єкта займає якомога більшу частину зображення.

Погане фокусування зображення і орієнтація обличчя відносно камери також може вплинути на точність розпізнавання.

Для реалізації детектування також створено клас *FaceDetectorProcessor*, для зручної роботи з ML Kit. Основним об'єктом в цьому класі є *FaceDetector*, для створення якого нам потрібно визначити опції детектора.



```
private val detector: FaceDetector

init {
    val faceDetectorOptions = FaceDetectorOptions.Builder()
        .setLandmarkMode(FaceDetectorOptions.LANDMARK_MODE_NONE)
        .setContourMode(FaceDetectorOptions.CONTOUR_MODE_NONE)
        .setClassificationMode(FaceDetectorOptions.CLASSIFICATION_MODE_NONE)
        .setPerformanceMode(FaceDetectorOptions.PERFORMANCE_MODE_FAST)
        .setMinFaceSize(0.4f)
        .build()

    detector = FaceDetection.getClient(faceDetectorOptions)
}
```

Рисунок 3.2 Ініціалізація FaceDetector з необхідними опціями

LandmarkMode - параметр, що відповідає за визначення орієнтирів обличчя: очі, вуха ніс.

ContourMode - параметр, що відповідає за виявлення контурів обличчя. Контури виявляються тільки для найвиразнішого обличчя.

ClassificationMode - параметр, що відповідає за класифікування обличчя за категоріями: ймовірність посмішки, ймовірність відкриття очей.

MinFaceSize - параметр, що встановлює, найменший бажаний розмір обличчя, виражений як співвідношення ширини голови до ширини зображення.

Для розпізнавання обличчя *FaceDetector* викликає функцію process, яка приймає на вхід зображення. Та містить лямбда функції для повернення результату, помилки чи завершення виконання розпізнавання.

```
fun processImageProxy(image: ImageProxy, onDetectionFinished: (List<Face>) -> Unit) {
        detector.process(InputImage.fromMediaImage(image.image!!,
 image.imageInfo.rotationDegrees))
            .addOnSuccessListener(executor) { results: List<Face> ->
                onDetectionFinished(results)
            }
            .addOnFailureListener(executor) { e: Exception ->
                Log.e("Camera", "Error detecting face", e)
            }
            .addOnCompleteListener { image.close() }
    }
```

Рисунок 3.3 Процес розпізнавання облич

3.3 Особливості застосунку

Застосунок виконано використовуючи архітектурний підхід для розробки застосунків під Android, що зветься Single Activity. Підхід полягає в тому, щоб весь застосунок був заснований на одній Activity, яка управляє фрагментами для відображення вмісту на екрані. В такій архітектурі всі взаємодії користувача з застосунком відбуваються в рамках цієї Activity. Використання однієї Activity замість декількох також допомагає зменшити ресурси, що використовуються застосунком і покращити продуктивність. Також покращується контроль навігації між екранами, адже навігація при такому підході здійснюється централізовано. Також розробка застосунку стає більш простою, адже не потрібно буде добавляти новостворені Activity до файлу маніфесту, щоб в подальшому відображати його.

Застосунок в цілому складається з 4 екранів, на яких виконується робота з розпізнавання:

- *Головний екран*
- *Екран з налаштуваннями*
- *Екран з камерою*
- *Екран з завантаженням зображення з файлової системи*

Для відображенням UI елементів на екрані обрано Jetpack Compose, адже має суттєві переваги над XML.

XML (Extensible Markup Language) - це стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками. Основними XML файлами є файли layout, string, color та drawable. Де layout - розмітка самого екрану чи його частини, string - файл з всіма текстовими ресурсами, color - кольори, та drawable - svg та png зображення. Сам синтаксис XML певною мірою нагадує створення веб сайтів, коли інтерфейс визначається в HTML.

Jetpack Compose це декларативний фреймворк. Декларативний підхід, який дозволяю нам створювати програми, не описуючи control flow. Якщо говорити іншими словами, то ми описуємо не “як”, а “що” ми хочемо бачити на екрані.

Також має й інші значні переваги в порівнянні з XML:

- Kotlin-only: не потрібно переключатись між класами та xml файлами, адже всю роботу можна зробити в Kotlin файлі.
- Композиційний підхід: кожен елемент інтерфейсу представляє собою Kotlin функцію, яка відповідає тільки за певний функціонал.
- Зворотна сумісність: дозволяє в одному проекті та на одному екрані використовувати як XML розмітку так і Composable функції.
- Менше коду: дозволяє для реалізації одного і того ж екрану зменшити кількість коду, та уникнути створення цілих класів, що могли призвести до помилок. Код є простий та легкий в обслуговуванні.

- Потужний: є оптимізованим під нативну розробку, може працювати в іншому потоці, а також позбавляє від зображення зайвих елементів на екрані.

Також за основу взято перевикористання деяких UI елементів, що є теж перевагою Jetpack Compose.

Головний екран містить два елементи, по кліку на яких можна перейти на екран з камерою або з вибору фото з файлової системи відповідно.

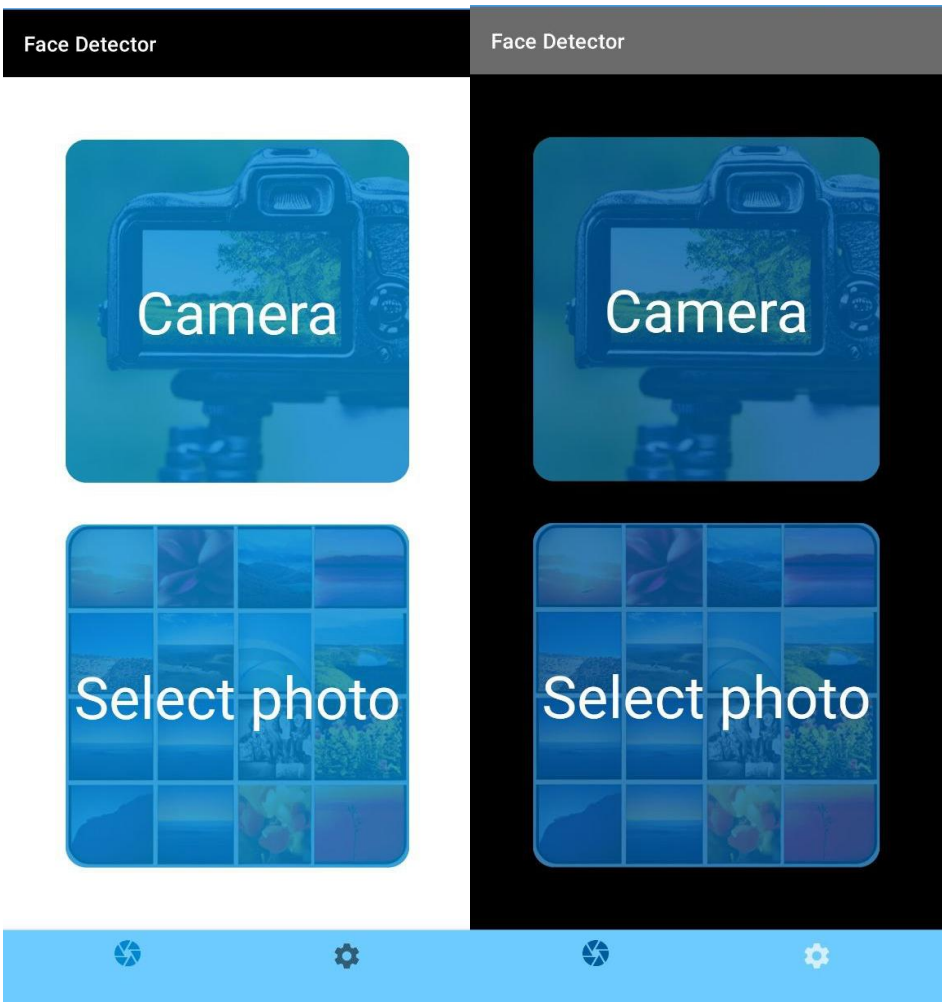


Рисунок 3.4 Головний екран

Екран налаштувань містить набір параметрів, за допомогою якого можна змінити поведінку роботи застосунку. А саме можна дозволити класифікацію за

ймовірності посмішки чи відкриття очей. Змінити товщину та колір рамки, в якій буде знаходитись обличчя, а також колір та розмір тексту біля рамки.

Крім того на цьому екрані міститься посилання на політику конфіденційності й інформацію про версію застосунку.

Дані про налаштування зберігаються в системному сховищі Android, що зветься SharedPreferences, що є механізмом для зберігання даних в приватному ключ-значення форматі. Що дозволяє зберігати прості типи даних, такі як String, Int чи Boolean, на постійному зберіганні на пристрою користувача. SharedPreferences зберігає дані в приватному файлі на пристрої користувача та забезпечує доступ до цих даних з будь-якого місця в застосунку. Це робить його дуже зручним для зберігання малих об'ємів даних, таких як налаштування.

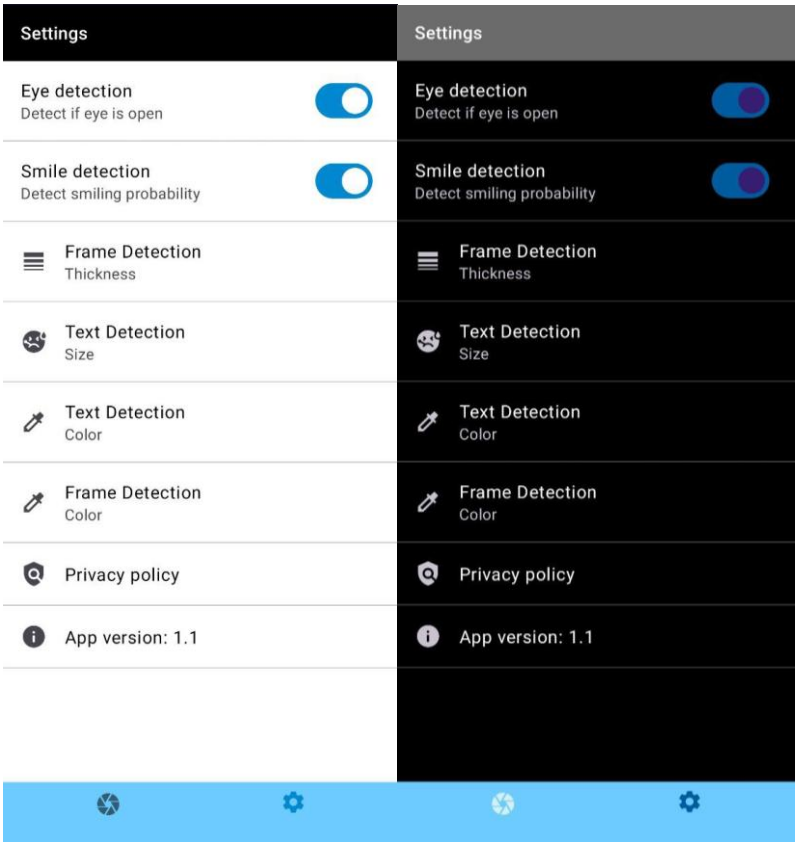


Рисунок 3.4 Екран налаштувань

Екран з завантаженням зображення з файлової системи - це екран, на якому відбувається розпізнавання облич на вже існуючих в файловій системі

Android зображеннях. Щоб потрапити на даний екран, користувач повинен надати доступ Android застосунку для запису та зчитуванню з файлової системи (до API 29). Після відкриття екрану з'являється елемент, при кліку на який можна вибрати зображення, для розпізнавання облич, після вибору відбувається процес розпізнавання та обробка результату.

Якщо на вхід подати зображення, що не містить ніяких облич, то просто повернеться та сама картинка. Якщо виявлено обличчя, то за допомогою методу обробки зображення, використовуючи Canvas, малюємо отримані обмежувальні рамки, та результати класифікації.

Після цього результат буде виведено на екран Android пристрою, і появляться кнопки для збереження зображення в файлову систему чи поділитися з друзями.

Навколо обмежувальної рамки smile - ймовірність посмішки, l та r - ймовірність відкриття лівого та правого ока відповідно.

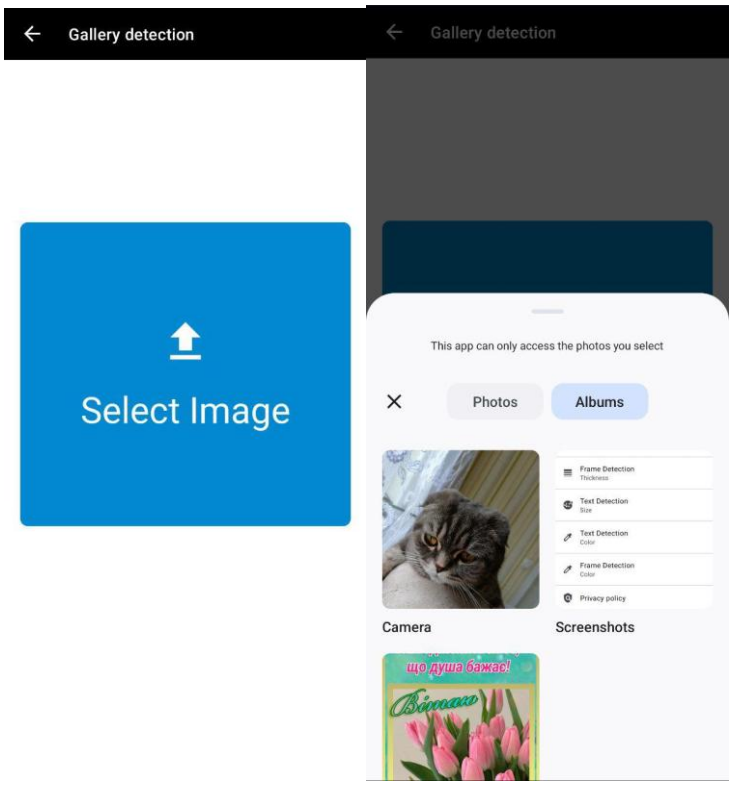


Рисунок 3.5 Екран з завантаженням зображення з файлової системи

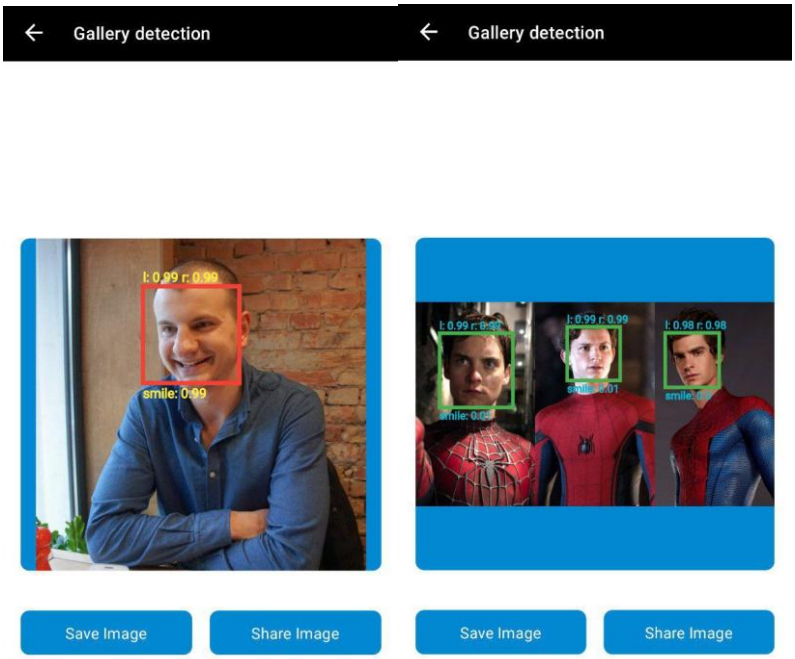


Рисунок 3.6 Приклад обробки заготовлених зображень

Екран з камерою - це екран, на якому відбувається розпізнавання облич в режимі реального часу, з подальшою можливістю зробити вже оброблене фото, яке збережеться в галерею. Щоб потрапити на даний екран необхідно надати дозвіл до камери і файлової системи. В файлі маніфесту також визначити, список апаратних функцій, які будуть використовуватися. Ними є камера, автофокус та спалах.

Для подальшої реалізації підключено бібліотеку для роботи з камерою на Android. Зображення виводиться в режимі реального часу разом з розпізнаванням облич.

Застосунок дає змогу використовувати фронтальну й основну камеру, а також має кнопку для фотографування.

Процес обробки зображення однаковий з екраном з завантаженням зображення з файлової системи. Прикладом результату роботи цього екрану є зображення нижче.

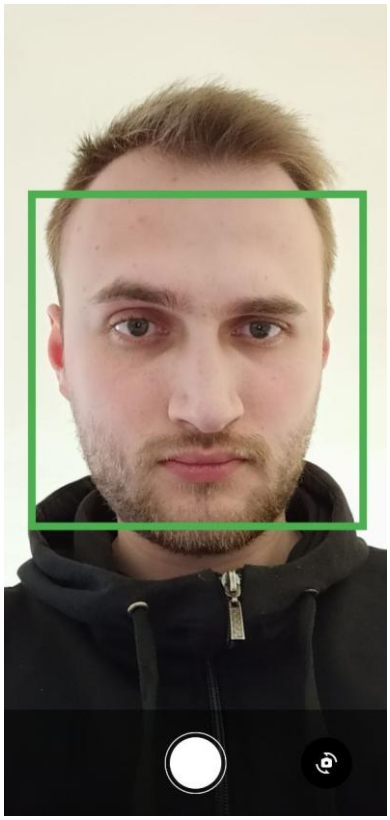


Рисунок 3.7 Результат обробки зображення на фронтальну камеру

3.4 Реліз застосунку

Після розробки застосунку та перевірки базового функціоналу потрібно зарелізити застосунок в Play Store.

Додано аналітику в застосунок. Існує багато інструментів для збору аналітики. Один з найпопулярніших - Firebase Analytics від Google. Для його використання потрібно створити проект Firebase, додати в нього застосунок та підключити необхідні бібліотеки до проекту.

Після підключення Firebase Analytics в застосунок, можна відстежувати різні події та властивості в застосунку, такі як кількість відвідувань екранів, час проведений користувачем на сторінці, кількість використання функцій застосунку тощо. Для цього потрібно додати відповідний код в різних частинах застосунку, які потрібно відслідковувати.

Для відстеження помилок в застосунку можна використовувати Crashlytics. Це інструмент, що дозволяє відстежувати та аналізувати помилки в застосунку, що допомагає знайти та виправити проблеми в застосунку перед тим, як спричинять проблеми для користувачів.

Підключення Crashlytics до проекту також відбувається за допомогою Firebase. Після підключення, при помилці застосунку, Crashlytics збирає інформацію про помилку та надсилає її до Firebase. В цьому випадку розробник отримує детальну інформацію про причини помилки та може виправити проблему.

Після чого створено аккаунт в Google Play Console. В середині консолі розробника створено новий застосунок, і заповнені всі необхідні поля. Потім завантажено APK файл застосунку та розгляд командою Google на ймовірні порушення застосунку. Після чого застосунок був успішно опублікований в Play Store.

3.5 Висновки до розділу

Розглянуто архітектуру застосунку, описана структура програми та призначення кожної директорії. Описана реалізація розпізнавання облич використовуючи ML Kit.

Описано особливості реалізації кожного з екранів застосунку, та наглядно показано UI цих сторінок: головний екран, екран з налаштуваннями, екран з камерою, екран з завантаженням зображення з файлової системи. Розписано деталі кожного з екранів, та робочу поведінку на кожному екрані. Проведена робота з камерою та файловою системою, обробкою результатів розпізнавання.

Після перевірки роботи застосунку, до нього підключено Analytics та Crashlytics. Також успішно проведено реліз застосунку в Google Play Store.

Створений застосунок відповідає поставленим вимогам по розпізнаванню облич.

ВИСНОВКИ

В першому розділі доглядно розглянуто види нейронних мереж, класифікацію. Більш детально розглянуто архітектури на основі згорткових нейронних мереж.

Після розгляду основних відомостей про нейронні мережі, проведено аналіз існуючих засобів для реалізації застосунку. Для реалізації використано:

- фреймворк Android SDK для створення застосунку
- мова програмування Kotlin
- фреймворк ML Kit для реалізації виявлення облич

Результатом виконання було створено Android-застосунок, для виявлення облич, використовуючи нейронні мережі. Створений застосунок має наступний функціонал:

- Виявлення облич використовуючи існуючі зображення
- Виявлення облич в режимі реального часу
- Збереження результату розпізнавання
- Баланс швидкодії та продуктивності
- Кастомізація отриманого результату
- Підтримка роботи без інтернет з'єднання
- Підтримка нічного режиму

Код програмного застосунку доступний у відкритому доступі, що дозволяє використання іншим розробникам для реалізації власних ідей. Також застосунок можна завантажити в Google Play store.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Different Types of CNN Architectures Explained: Examples [Електронний ресурс]. Режим доступу: https://vitalflux.com/different-types-of-cnn- architectures-explained-examples/
2. VGG - 16 CNN Model [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/vgg-16-cnn-model/
3. What is a neural network? Explanation and examples [Електронний ресурс]. Режим доступу: https://www.techtarget.com/searchenterpriseai/definition/ neural-network
4. A Basic Introduction To Neural Networks [Електронний ресурс]. Режим доступу: https://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html
5. Xamarin or Android Studio – Which one to choose for enterprise app development? [Електронний ресурс]. Режим доступу: https://i-verve.com/xamarin-vs-android-studio/
6. Kotlin vs Java: Which is the Best Choice for Android App Development? [Електронний ресурс]. Режим доступу: https://medium.com/javarevisited/kotlin-vs-java-which-is-the-best-choice-for-android-app-development-7c9fc782d2c9
7. Detect faces with ML Kit on Android [Електронний ресурс]. Режим доступу: https://developers.google.com/ml-kit/vision/face-detection/android
8. AlexNet: The Architecture that Challenged CNNs [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951
9. Dive into React Native for Android Development [Електронний ресурс]. Режим доступу: https://www.toptal.com/react-native/react-native-for-android-development
10. Perceptron Neural Networks [Електрониий ресурс]. Режим доступу: https://www.mathworks.com/help/deeplearning/ug/perceptron-neural-networks.html

11. What is PyTorch, and How Does It Work: All You Need to Know [Електронний ресурс]. Режим доступу: https://www.simplilearn.com/what-is-pytorch-article
12. Open CV Tutorial Gary Bradski [Електронний ресурс]. Режим доступу: http://roswiki.autolabor.com.cn/attachments/Events(2f)ICRA2010Tutorial/ICRA_2010_OpenCV_Tutorial.pdf
13. Classification Neural Network [Електронний ресурс]. Режим доступу: https://www.sciencedirect.com/topics/engineering/classification-neural-network
14. Mobile Deep Learning with TensorFlow Lite, ML Kit and Flutter [Електронний ресурс]. Режим доступу: http://surl.li/gjgfl
15. Analysis of Architectural Patterns for Android Development Nayab Akhtar, Sana Ghafoor [Електронний ресурс]. Режим доступу: https://www.researchgate.net/profile/Nayab-Akhtar-2/publication/352021976_Analysis_of_Architectural_Patterns_for_Android_Development/links/60b625d092851cde8847e819/Analysis-of-Architectural-Patterns-for-Android-Development.pdf
16. Head First Android Development: A Brain-Friendly Guide, Second Edition [Електронний ресурс]. Режим доступу: https://www.siirt.edu.tr/dosya/personel/android_kitabi-siirt-2020228161632918.pdf
17. Rethinking the Inception Architecture for Computer Vision [Електронний ресурс]. Режим доступу: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf
18. Types of Neural Networks and Definition of Neural Network [Електронний Ресурс]. Режим доступу: https://www.mygreatlearning.com/blog/types-of-neural-networks/

ДОДАТКИ

Додаток 1

Посилання на репозиторій

https://github.com/serhohuk/FaceDetector

Додаток 2

Посилання на Google Play Store

https://play.google.com/store/apps/details?id=com.serhohuk.facedetector