Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____
(рівень вищої освіти)

Спеціалізовані комп'ютерні засоби для розпізнавання контурних зображень
(тема)

Виконав: студент 2 курсу, групи СКСм-20-2

_____Погрібний Є. Є._____
(прізвище, ініціали)

Спеціальність *123 _ Комп'ютерна інженерія*

Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_____

*Спеціалізовані комп'ютерні системи*_____
(повна назва освітньої прграми)

Керівник роботи проф. Кривуля Г.Ф.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«25» 03 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Погрібному Євгенію Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Спеціалізовані комп'ютерні засоби для розпізнавання контурних зображень

Specialized Computer Tools for Contour Image Recognition

затверджена наказом по університету від «24» 03 2022 р. № 408 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.05.2022

3. Вихідні дані до роботи (проекту)

Теоретичні відомості про платформу Android

Мова програмування Kotlin

Бібліотеки OpenCV та TensorFlow Lite

4. Перелік питань, що потрібно опрацювати у роботі

Аналіз проблемної галузі та постановка задачі

Аналіз використовуваних технологій

Опис використовуваних програмних засобів

Принцип роботи та аналіз мобільного додатку

Тестування та відлагодження програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _17 слайдів_ _____ _____

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 20.01.2022_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми	20.01.2022-25.01.2022	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів	01.03.2022-09.03.2022	
3	Аналіз існуючих технологій виділення контурів на зображенні	10.03.2022-19.03.2022	
4	Аналіз існуючих технологій розпізнавання зображень	20.03.2022-31.03.2022	
5	Розробка інтерфейсу мобільного додатку	01.04.2022-04.04.2022	
6	Розробка мобільного додатку	05.04.2022-24.04.2022	
7	Тестування та відлагодження мобільного додатку	25.04.2022-30.04.2022	
8	Оформлення пояснювальної записки	01.05.2022-20.05.2022	
9	Захист проекту	21.05.2022-25.05.2022	

Студент _____ _____
(підпис)

Керівник роботи (проекту) _____ _____ проф. Кривуля Г. Ф.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 76 сторінок, 30 рисунків, 20 джерел за переліком посилань.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МОБІЛЬНИЙ ДОДАТОК, ANDROID, KOTLIN, КОНТУРНІ ЗОБРАЖЕННЯ, COMPUTER VISION, MACHINE LEARNING, OPENCV, TENSORFLOW.

Метою даної роботи є розробка програмного забезпечення для розпізнавання контурних зображень, що дозволяє виділити контур на зображенні та розпізнати об’єкт на фото.

У процесі виконання даної роботи було розроблено мобільний додаток, який дозволяє, використовуючи спеціальний алгоритм, підкреслити контур об’єкта, зображеного на фото, а також встановити який саме об’єкт зображено.

Для реалізації алгоритму виділення контуру зображення були використані засоби технології Computer Vision, а саме бібліотеку OpenCV для Android, а для допомоги у розпізнаванні об’єкта на фото були використані елементи Machine Learning, а саме бібліотека TensorFlow Lite, розроблена компанією Google для розгортування моделей на мобільних пристроях, мікроконтролерах та інших периферійних пристроях.

ABSTRACT

The explanatory note contains 76 pages, 30 figures, 20 sources.

SOFTWARE, MOBILE APPLICATION, ANDROID, KOTLIN, CONTOUR IMAGES, COMPUTER VISION, MACHINE LEARNING, OPENCV, TENSORFLOW.

The main goal of this project is to develop software for contour image recognition, which allows you to highlight the contour on the image and recognize the object in the photo.

During the implementation of this work, a mobile application was developed, which allows, using a special algorithm, to highlight the contour of the object depicted in the photo, as well as to recognize which object is depicted.

Computer Vision technology, namely the OpenCV library for Android, was used to implement the image contour highlighting algorithm, and Machine Learning elements, namely the TensorFlow Lite library, developed by Google for deploying models on mobile, microcontrollers and other edge devices.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CV – Computer Vision;

ML – Machine Learning;

КА – контурний аналіз;

ЕВ – елементарний вектор;

ВК – вектор-контур;

Індустрія 4.0 (або четверта промислова революція) – це новий підхід до виробництва, заснований на масовому впровадженні інформаційних технологій у промисловість, автоматизацію бізнес-процесів та поширення штучного інтелекту;

ШІ – штучний інтелект;

CNN (convolutional neural network) – згорткова нейронна мережа;

RNN (recurrent neural network) – рекурентна нейронна мережа;

PDE (partial differential equation) – диференціальне рівняння з частинними похідними;

JVM – Java Virtual Machine;

SDK – Software Development Kit;

APK (Android Package) – формат файлу, який використовується операційною системою Android для розповсюдження та встановлення мобільних програм та ігор;

ADB (Android Debug Bridge) – програма, яка дозволяє Android Studio комунікувати з будь-яким пристроєм на системі Android;

XML (Extensible Markup Language) – це мова розмітки та формат файлу для зберігання, передачі та відновлення довільних даних.

ВСТУП

Розробка програмного забезпечення з використанням сучасних технологій, таких як Computer Vision та Machine Learning, на сьогоднішній день є доволі актуальною темою на ринку в індустрії інформаційних технологій. Це зумовлено тим, що чим більше ці технології досліджуються та розвиваються, тим більше з’являється сфер та ідей для їх застосування.

Фіксація порушників ПДР на камери, трансляція зображення дронами, охоронні системи, і навіть маски з різноманітними ефектами – все це приклади застосування комп'ютерного зору. Також, дана технологія може бути використана у сферах промисловості, медицині, торгівлі, системах керування автомобілем та ще велика кількість цікавих прикладів.

Щодо Machine Learning, то використання цієї технології можна знайти у робототехніці, сфері маркетингу, соціальних мережах, системах розпізнавання облич, медицині і т.д. Все це можливо за допомогою аналізу даних.

Актуальною задачею даного проекту є розробка мобільного додатку, який здатен розпізнавати контурні зображення на фото, що дозволяє виділити контур зображеного об’єкта та більш точно розпізнати його, використовуючи технології Computer Vision та Machine Learning.

Мобільний додаток розробляється на базі операційної системи Android, за допомогою мови програмування Kotlin. Він повинен працювати на будь-якому мобільному пристрої з версією Android 5.0 та вище. Також додаток потребує наявності зовнішньої камери на пристрої, а також доступу до внутрішньої галереї зображень.

Інтерфейс додатку повинен мати можливість завантажити зображення із галереї для аналізу, зробити нове фото для аналізу, відобразити результат та зберегти проаналізоване зображення з виділеним контуром до галереї.

Для правильної роботи додатку, необхідно розробити алгоритм

виділення контуру об’єкта на зображенні, тобто його видимий край, який відокремлює об’єкт від фону. Для вирішення цієї проблеми використовується бібліотека OpenCV для платформи Android. Маючи готові контури можна приступати до подальшого аналізу. А для розпізнавання об’єкту на зображенні використовується бібліотека TensorFlow Lite від компанії Google.

1 ЗАГАЛЬНІ ПОНЯТТЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Контурний аналіз для розпізнавання зображень

При формуванні зорового образу в свідомості людини зоровий апарат здійснює відстеження лінії контуру об’єкта, внаслідок чого в свідомості відзначаються його характерні деталі. Необхідним етапом сприйняття вважається сканування по лінії контуру з метою створення образу об'єкта для подальшого розпізнавання. Це розкриває важливу роль контурів при розпізнаванні зображення.

Контурний аналіз, під яким слід розуміти сукупність методів виділення, опису та перетворення контурів зображення, є важливим етапом обробки зображень і розпізнавання зорових образів. Контур цілком визначає форму зображення і містить всю необхідну інформацію для розпізнавання зображень за їх формою.

КА дозволяє описувати, зберігати, порівнювати і здійснювати пошук об’єктів, представлених у вигляді своїх зовнішніх обрисів – контурів. Внутрішні точки об’єкта до уваги не беруться. Це обмежує сферу застосовуваності алгоритмів КА, але розгляд самих лише контурів дозволяє перейти від двовимірного простору зображення до простору контурів і тим самим знизити обчислювальну та алгоритмічну складність. КА дозволяє ефективно вирішувати основні проблеми розпізнавання образів – перенесення, повороту і зміни масштабу зображення об’єкта. Методи КА є інваріантними щодо цих перетворень.

Що ж таке контур об’єкта? Контур – це межа об’єкта, сукупність точок (пікселів), що відокремлюють об’єкт від фону. У системах комп’ютерного зору використовується кілька способів кодування контуру. Найбільш відомими є код Фрімена, двовимірне кодування, полігональне кодування. Проте всі ці способи кодування не використовуються в КА. Натомість у КА

контур кодується послідовністю, що складається з комплексних чисел. На контурі фіксується точка, яка називається початковою точкою. Потім контур обходиться (за годинниковою стрілкою), і кожен вектор зміщення записується комплексним числом $a + ib$, де a – зміщення точки по осі X, а b – зсув по осі Y. Зсув вираховується щодо попередньої точки.

Через фізичну природу тривимірних об’єктів їх контури завжди замкнені і не можуть мати самоперетину. Це дозволяє однозначно визначити шлях обходу контуру (з точністю до напряму – за або проти годинникової стрілки). Останній вектор контуру завжди приводить до початкової точки.

Кожен вектор контуру називається елементарним вектором (ЕВ). А сама послідовність комплекснозначних чисел – вектор-контуром (ВК). Вектор-контури позначаються великими грецькими літерами (N), а їх елементарні вектори – малими грецькими буквами (ν)

Чому в КА використовується саме комплекснозначне кодування? Тому що операції над контуром саме як над вектором комплексних чисел характеризуються чудовими математичними властивостями, порівняно з іншими способами кодування.

В принципі, комплексне кодування є близьким до двовимірного, де контур визначається як сукупність ЕВ, представлених своїми двовимірними координатами. Різниця ж полягає в тому, що операції скалярного добутку для векторів і для комплексних чисел є різними. Саме ця обставина й надає перевагу методам КА.

Методи контурного аналізу актуальні й на сучасному етапі у сфері розпізнавання об'єктів. Наприклад у програмах розпізнавання тексту, такі програми зараз значно поширені навіть у повсякденному житті. Окрім розпізнавання тексту, контурний аналіз використовують у системах розпізнавання об'єктів потокового відео. Такі системи на сучасному етапі використовують у сфері безпеки, для автопілотів автомобілів тощо. Під час використання методів контурного аналізу можна підраховувати кількість потрібних об'єктів на зображенні, такі системи використовуються на складі

підприємств для проведення аудитів товарів.

Методи контурного аналізу часто використовують в Інустрії 4.0. для моделювання та візуалізації етапів виробничого процесу. Окрім розпізнавання тексту на зображенні для оцінювання відповідності об'єкта певним вимогам, існує можливість розпізнавання символів за допомогою Web-камери.

Алгоритми виділення меж розглядають зміни яскравості на знімку як деяку безперервну функцію і використовують операції з похідними 1-го і 2-го порядку. Найбільш популярними методами виявлення контурів є високочастотні та фільтри з операторам Робертса, Собеля, Превітта, Кірша. Але вони не є стійкими до шумів і дають розривну межу. Більш складний метод Канні забезпечує безперервність меж у складних умовах зашумленого знімка.

Усі методи контурного аналізу базуються на одній із властивостей яскравості – розривності. Для пошуку цієї розривності використовують так звану ковзну маску, що є квадратною матрицею коефіцієнтів, відповідну до групи пікселів вхідного зображення. Цю матрицю використовують для виконання просторової фільтрації зображення.

Просторова фільтрація виконується звичайним переміщенням маски по зображенню та обрахуванням у кожній точці градієнта рівня яскравості. Однотонні простори зображення будуть мати низький градієнт рівня яскравості, у вихідному зображенні такі простори зображення темнішають, а у випадку високого рівня градієнта на вихідному зображенні ці простори будуть мати яскравіші лінії. Перед обрахунком градієнта потрібно вирахувати відгук R фільтрації в точці (x, y):

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + ...$$
$$+w(0,0)f(x,y) + ... + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1),$$

тобто це сума добутку коефіцієнтів маски на значення пікселів, до яких застосовується маска. Для визначення градієнта яскравості використовуються дискретні аналоги похідних першого і другого порядку.

Перша похідна f(x) визначається як різниця значень сусідніх пікселів, а друга похідна – як різниця значень сусідніх значень першої похідної

$$\frac{\delta f}{\delta x} = f(x+1) - f(x),\ \frac{\delta^2 f}{\delta x^2} = f(x+1) + f(x-1) - 2f(x).$$

Піксель зображення звичайно має дві координати, а отже, у випадку двох змінних (*x, y*) потрібно просто обрахувати часткові похідні за двома просторовими осями. Обчислення першої похідної зображення виконують на дискретних наближеннях двовимірного градієнта. Градієнт зображення – це вектор

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix}.$$

У методах контурного аналізу важливим є модуль вектора градієнта функції:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2}.$$

Напрямок вектора градієнта теж є основною характеристикою під час знаходження контурів:

$$a(x,y) = arctg\left(\frac{G_y}{G_x}\right).$$

Напрямок контура в точці (*x, y*) буде перпендикулярний напрямку вектора градієнта.

Розглянемо і проаналізуємо існуючі алгоритми визначення контурів зображень та порівняємо їх ефективності:

1. **Оператор Прюітта** використовує маску 3×3 для просторової фільтрації.

$$\begin{aligned} G_x &= (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \\ G_y &= (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \end{aligned}.$$

У цих функціях різниця між сумами по верхній і нижній строками області 3×3 є наближеними значеннями похідної по осі *x*, а різниця між

сумами по першому та останньому стовпцям – похідна по осі *y*. Реалізуються ці функції за допомогою маски фільтрації по осях *x* та *y*

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}; \; G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

Виконаємо оператор Прюітта на тестовому зображенні роздільної здатності 6016×4000 px (рис. 1.1 і 1.2).

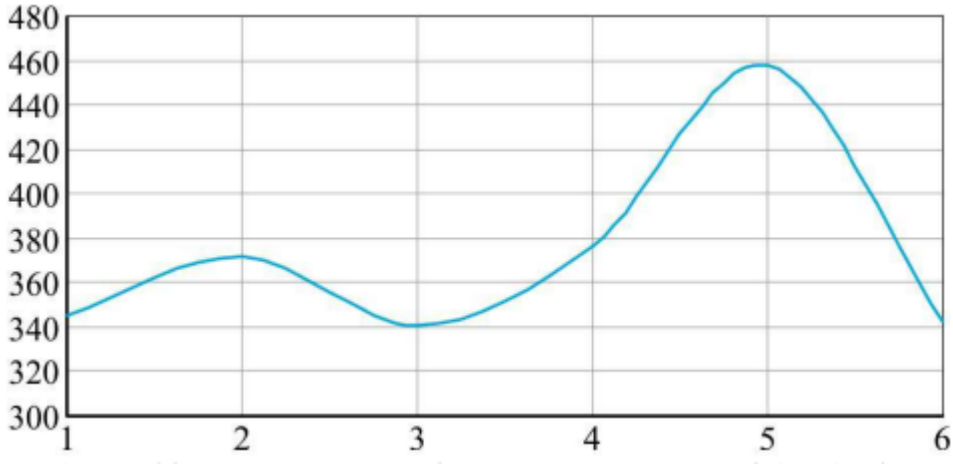

Рисунок 1.1 – Графік завантаженості оперативної пам'яті (МБ) під час виконання оператора Прюітта

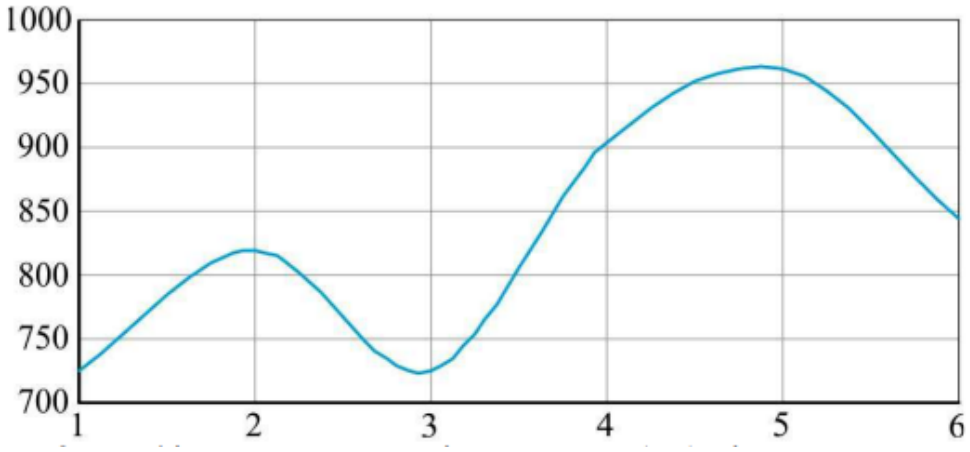

Рисунок 1.2 – Графік завантаженості процесора (Гц) під час виконання оператора Прюітта

Результати роботи оператора Прюітта (рис. 1.3).

Рисунок 1.3 – Оригінальне зображення (зліва). Результат виконання оператора Прюітта (справа)

2. **Оператор Собеля** також використовує маску 3×3 для просторової фільтрації.

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}; \; G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

Однак його особливістю є використання вагового коефіцієнта для значень середніх елементів:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7).$$

Для перевірки роботи оператора Собеля використаємо те саме тестове зображення (рис. 1.4 і 1.5).

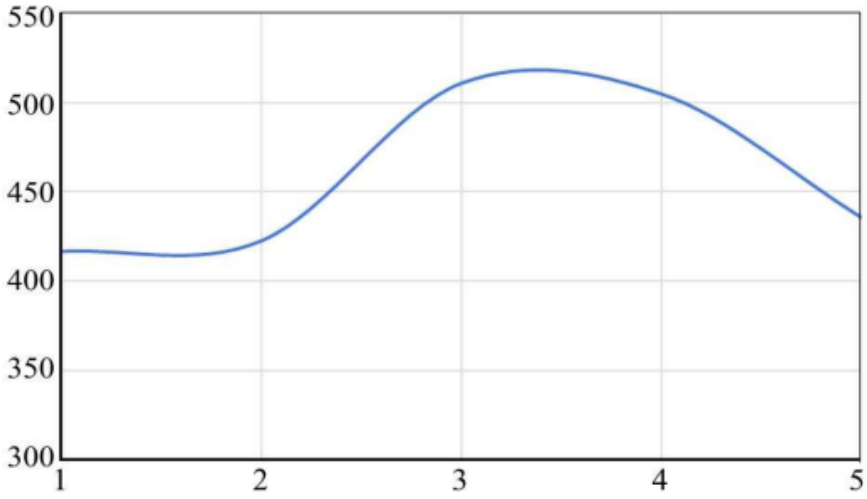

Рисунок 1.4 – Графік завантаженості оперативної пам'яті (МБ) під час виконання оператора Собеля

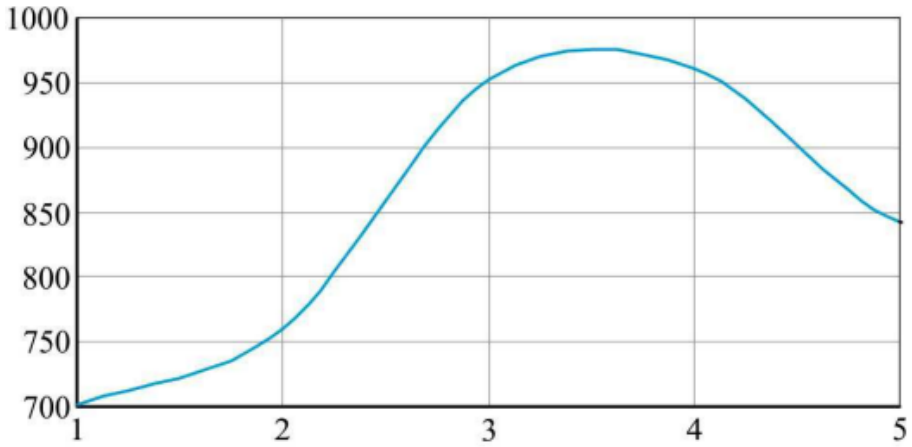

Рисунок 1.5 – Графік завантаженості процесора (Гц) під час виконання оператора Собеля

Результати роботи оператора Собеля (рис. 1.6).

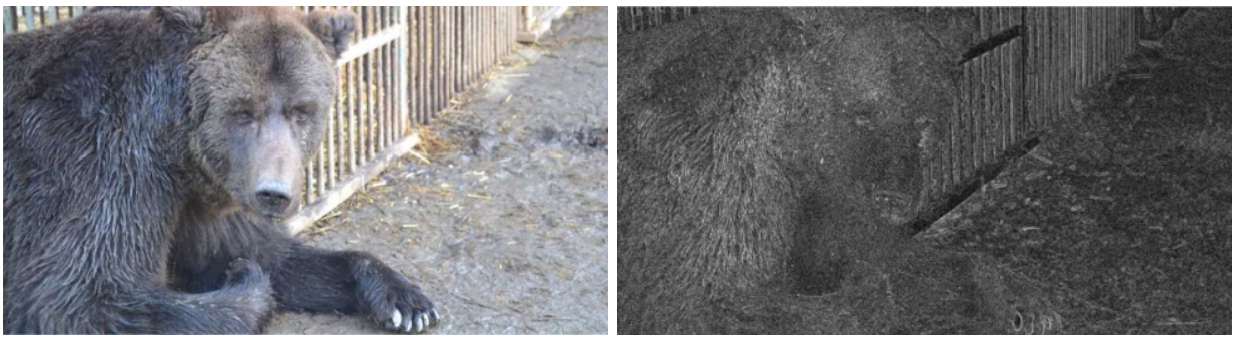

Рисунок 1.6 – Оригінальне зображення (зліва). Результат виконання оператора Собеля (справа)

Як видно з результатів, оператор Собеля більш чутливий до перепадів яскравості на зображенні.

3. **Оператор Кірша** використовує коефіцієнти для всіх значень, окрім середнього. Маска оператор Кірша (рис. 1.7 і 1.8).

$$G_x = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}; \; G_y = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & 3 & -3 \end{bmatrix}.$$

Як видно з результатів виконання оператора Кірша, вагові коефіцієнти дають дуже сильну чутливість до перепадів яскравості.

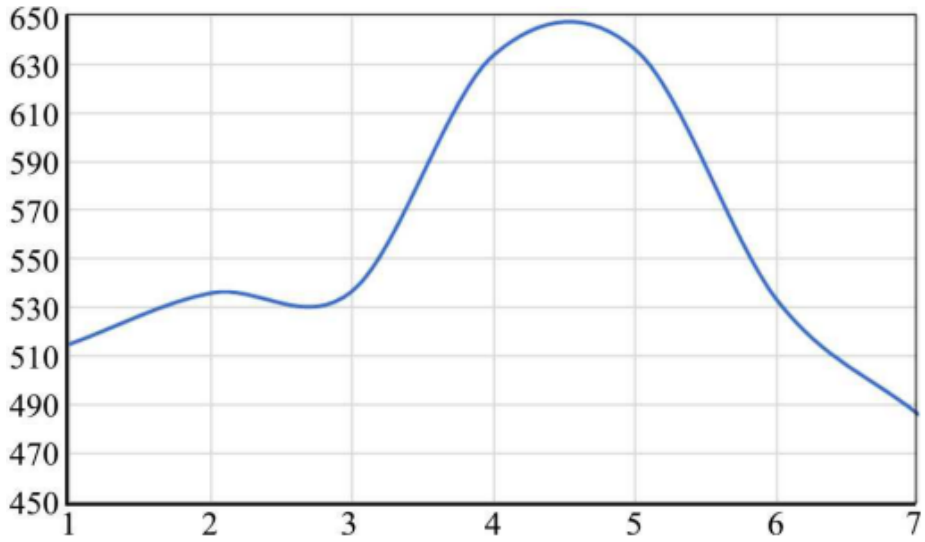

Рисунок 1.7 – Графік завантаженості оперативної пам'яті (МБ) під час виконання оператора Кірша

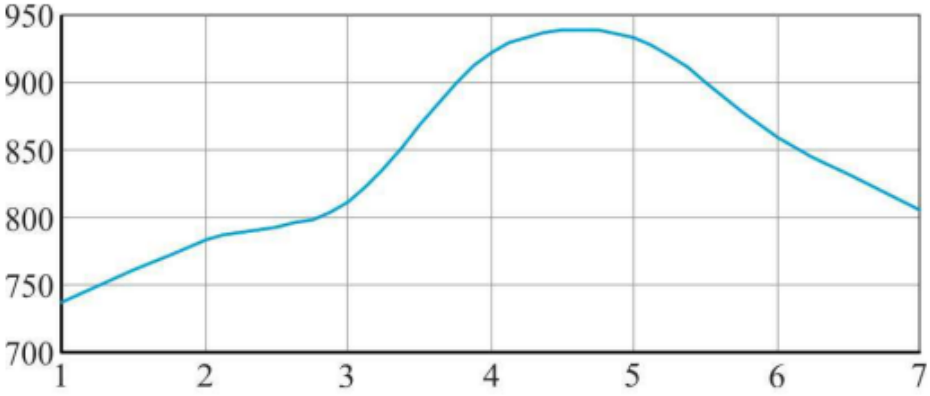

Рисунок 1.8 – Графік завантаженості процесора (Гц) під час виконання оператора Кірша

Результати роботи оператора Кірша наведено на рис. 9.

Рисунок 1.9 – Оригінальне зображення (зліва). Результат виконання оператора Кірша (справа)

4. **Оператор Лапласа (5×5).** Оператор Лапласа, що використовують для виділення контурів, це розширення векторного оператора Лапласа. Цей оператор має однакову маску для просторової фільтрації як для осі *x*, так і для осі *y*.

$$G_{(x,y)} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}.$$

Таку маску можна інтерпретувати як суму різниць центрального елемента з кожним із 24 інших елементів. Таким чином, однаково застосовують можливі перепади яскравості у всіх напрямках (рис. 1.10–1.12).

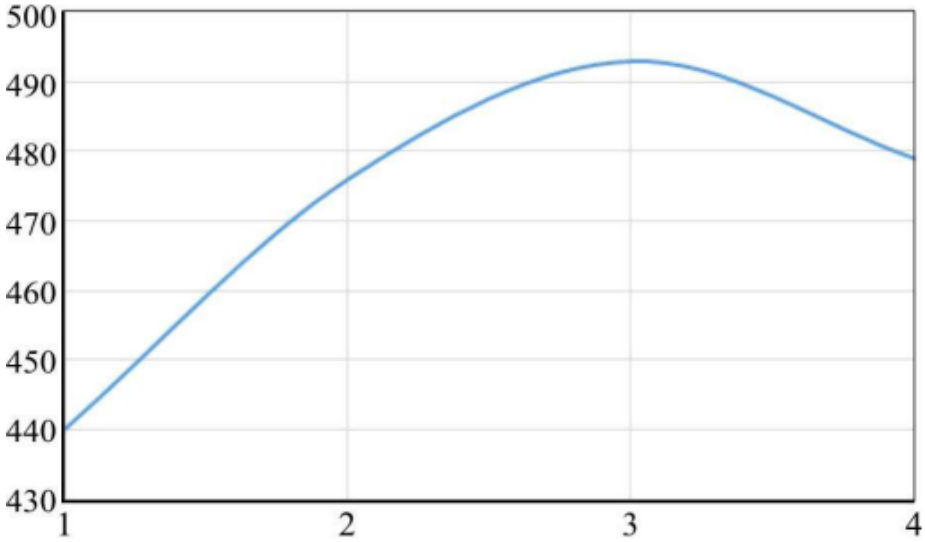

Рисунок 1.10 – Графік завантаженості оперативної пам'яті (МБ) під час виконання оператора Лапласа

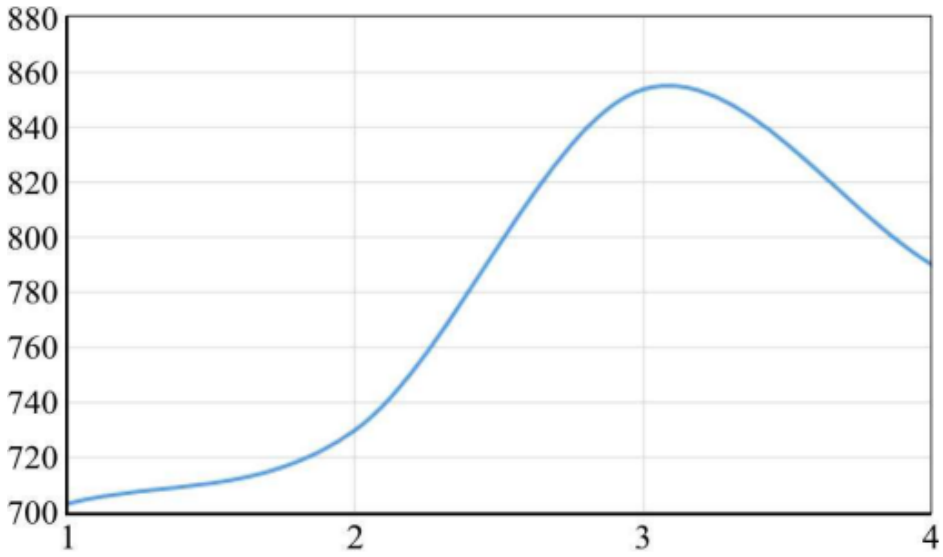

Рисунок 1.11 – Графік завантаженості процесора (Гц) під час виконання оператора Лапласа

Рисунок 1.12 – Оригінальне зображення (зліва). Результат виконання оператора (справа)

Проаналізувавши отримані результати, можна зробити висновок, що оператор Прюітта найменш ресурсно-витратний, але при цьому більшість контурів на зображенні просто не виділяються, але дуже добре підійде для зображень з високою загальною яскравістю, тому що в такому випадку менша імовірність прояву контуру, якого не існує на вхідному зображенні, це зумовлене менш чутливою маскою. Оператор Кірша є найбільш ресурсно-витратним методом серед розглянутих, але при цьому його перевага в дуже чутливій масці, що виділяє цей метод для використання навіть за найнижчої загальної яскравості зображення. Оптимальним методом щодо ресурсно-витратам та знаходження контурів є оператор Собеля. Це зумовлено використанням маски з коефіцієнтами тільки для середніх значень. Окремо

можна виділити оператора Лапласа. Цей метод виконується швидше і має меншу обчислювальну вартість. При цьому дає не набагато гірший за інші методи результат. Цей метод добре використовувати, якщо обчислювальні потужності є не високими.

Окрім описанних вище методів виділення контуру, значної популярності набув оператор Канні. **Оператор Канні** – оператор виявлення контурів зображення. Був розроблений в 1986 році Джоном Канні, який використовує багатоступінчастий алгоритм для виявлення широкого спектра контурів на зображеннях. Канні вивчив математичну проблему отримання фільтра, за оптимальними критеріями виділення, локалізація і мінімізації. Він показав, що шуканий фільтр є сумою чотирьох експонент і може бути добре наближений першою похідною Гауссіана. Канні ввів поняття Non-Maximum, яке означає, що пікселями контурів оголошуються пікселі, в яких досягається локальний максимум градієнта в напрямку вектора градієнта. Хоча його робота була проведена на початках розвитку комп'ютерного зору, детектор контурів Канні досі є одним із кращих детекторів, а також, він базується на попередніх методах, не потребує великих обчислювальних витрат та виділяє тільки значимі контури, що дає добрий показних співвідношення корисного сигналу і шуму, тому саме цей метод було використано при програмній реалізації даного атестаційного проекту.

Алгоритм Кенни складається з п’яти окремих кроків:

- Згладжування – розмиття зображення для видалення шуму;
- Пошук градієнта – межі виділяються там, де градієнт зораження набуває максимального значення;
- Придушення «Не максимумів» – тільки локальні максимуми відзначаються як межі;
- Подвійна порогова фільтрація – потенційні межі виділяються порогами;
- Трасування області неоднозначності – підсумкові межі визначаються шляхом придушення всіх меж, що не зв’язані з «сильними» межами.

Перед застосуванням детектора, слід перетворити зображення на відтінки сірого, щоб зменшити обчислювальні витрати. Цей етап характерний для багатьох методів обробки зображень.

Метод Канні легко адаптується до різних задач. Його параметри дозволяють адаптувати його до виявлення контурів на зображеннях з різними характеристиками в залежності від конкретних вимог.

Результатом робити описаних алгоритмів є набір незв'язних областей. Для отримання зв'язного контуру необхідно провести додаткову обробку, наприклад морфологічну для отримання зв'язного краю об'єкта, який і називається контуром об'єкта.

Необхідно зазначити, що контурний аналіз, також, має і недоліки. Розглянемо їх детальніше.

КА має дві групи факторів, які негативно позначаються на результатах розпізнавання.

Перша група факторів пов'язана з проблемою виділення контуру на зображеннях. Контур – це суворо визначена дискретна структура. Проте велике число реальних зображень мають об'єкти, слабко виражені на навколишньому фоні. Об'єкт може не мати чіткої межі, він може бути однаковий за яскравістю й кольором з фоном, може бути зашумлений перешкодами і так далі. Всі ці фактори призводять до того, що контур або неможливо виділити взагалі, або він виділяється неправильно й не відповідає межі об'єктів.

На рис. 1.13 показано бінаризоване зображення. Невеликі «містки» між зображенням цифри і навколишнім фоном роблять контур цифри нерозпізнаваним методами КА.

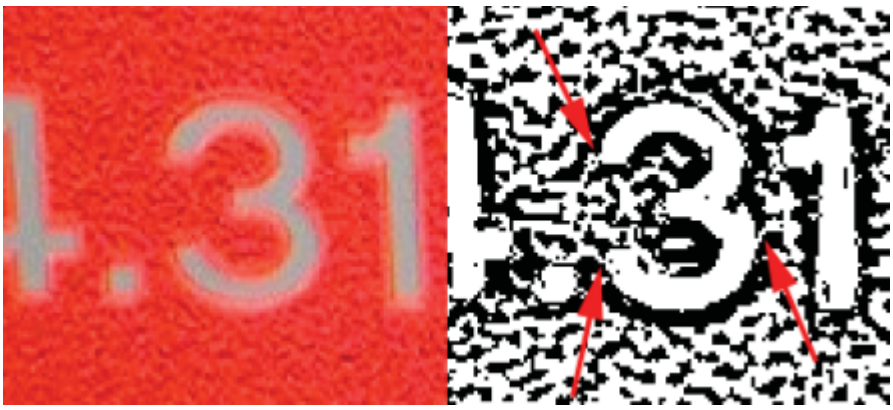

Рисунок 1.13 – Контурний аналіз зображення з нерозпізнаваним контуром

Такі випадки дуже складні для КА. Адже КА має сенс тільки в тому випадку, коли контур об’єкта визначений однозначно правильно у всіх своїх точках.

Друга група факторів, що ускладнюють КА, пов’язана з принципами контурного аналізу. Методи КА допускають те, що контур описує весь об’єкт повністю й не допускає жодних перетинів з іншими об’єктами або неповної видимості об’єкта.

На рис. 1.14 видно, як блік фотографії, що йде горизонтальною рискою, робить букви нерозрізнюваними для КА.

Рисунок 1.14 – Контурний аналіз зображення з горизонтальним бліком

Таким чином, КА має слабку стійкість до перешкод, не допускає перетину або часткової видимості об’єкта.

Але, незважаючи на недоліки, методи КА привабливі своєю простотою і швидкодією. За наявності чітко вираженого об’єкта на контрастному тлі і відсутності перешкод КА добре справляється з розпізнаванням.

1.2 Методи розпізнавання зображень

Дослідження методів розпізнавання зображень називають теорією розпізнавання образів. **Теорія розпізнавання образів** – розділ кібернетики, що розвиває теоретичні основи й методи класифікації і ідентифікації предметів, явищ, процесів, сигналів, ситуацій і т. п. об'єктів, які характеризуються скінченним набором деяких властивостей і ознак. Такі задачі вирішуються досить часто, наприклад, при переході або проїзді вулиці за сигналами світлофора. Розпізнавання кольору лампи світлофора, що засвітилася, і знання правил дорожнього руху дозволяє прийняти правильне рішення про те, можна, чи не можна переходити вулицю в цей момент.

Замість терміна «**розпізнавання**» часто вживається інший – «**класифікація**». Ці два терміни у багатьох випадках розглядаються як синоніми, але не є повністю взаємозамінюваними. Кожен з них має свої сфери застосування, і інтерпретація обох термінів часто залежить від специфіки конкретної задачі.

Іншими словами, **розпізнавання образів** (pattern recognition) – це розділ теорії штучного інтелекту, що вивчає методи класифікації об’єктів. За традицією об’єкт, що піддається класифікації, називається *образом* (pattern). Образом може бути цифрова фотографія (розпізнавання зображень), буква або цифра (розпізнавання символів), запис мови (розпізнавання мови) тощо.

В межах теорії штучного інтелекту розпізнавання образів включається в більш широку наукову дисципліну – *теорію машинного навчання* (machine learning), метою якої є розробка методів побудови алгоритмів, що здатні навчатися.

У більшості випадків, сприймаючи явища навколишнього світу, людина здійснює їх класифікацію, тобто розбиває ці явища (предмети, ситуації) на групи схожих явищ (саме схожих, а не тотожних). По тим чи іншим причинам необхідно віднести в одну групу у чомусь "подібні" явища чи предмети, які можуть при цьому значно відрізнятися один від одного.

Наприклад, усі фігури, зображені на наступному малюнку, ми називаємо «літерою А», незважаючи на їх значну відмінність у їх написанні.

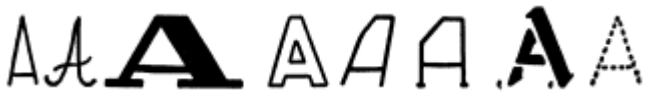

Рисунок 1.15 – Приклад класифікації об’єктів

Суттєвим є той факт, що виділивши такі групи (множини) об’єктів, ми отримуємо здатність "упізнавати", тобто встановлювати належність до вже відомої множини, нові об’єкти, які ще не зустрічалися нам раніше, наприклад впізнавати букви, написані новим для нас почерком.

Отримавши уявлення про те, що являє собою буква «А» на основі деякої, зазвичай невеликої кількості екземплярів цієї букви, ми спроможні упізнати як завгодно велику кількість інших її екземплярів.

Проте, далеко не всі множини об’єктів дають змогу на основі невеликої частини множини упізнати як завгодно багато інших невідомих нам її представників. Наприклад, фотографії студентів деякого вузу утворюють множину. Проте неможливо після ознайомлення зі скажімо десятьма фотографіями студентів визначити по новій фотографії є людина студентом цього вузу чи ні.

Таким чином існують множини деякого особливого типу. Ці множини мають *характерну властивість*, яка виявляється у тому, що після *ознайомлення із скінченною частиною об’єктів цих множин*, можна упізнавати як завгодно велику кількість інших їх представників. Множини такого типу і будемо називати **образами**.

Прикладами образів можуть бути такі множини: чоловіки, дитячі портрети, ссавці, картини Пікассо, цифри 5, зображення винищувачів МІГ. Застосовувати до них термін «образ» ми можемо тому, що ознайомлення з образом не пов’язано із запам’ятовуванням окремих об’єктів, а упізнавання нового об’єкта відбувається без безпосереднього порівняння із кожним раніше відомим.

Характерна властивість образів об’єктивна у тому сенсі, що різні люди

(живі істоти), які навчалися на різних групах об’єктів образа, у переважній більшості однаково і незалежно один від одного класифікують одні і ті самі нові об’єкти. Саме об’єктивність цієї властивості образів дає змогу людям, які вчилися у різних школах, успішно розпізнавати раніше невідомий їм почерк.

Створення штучних систем розпізнавання образів залишається складною теоретичною й технічною проблемою. Необхідність у такому розпізнаванні виникає в самих різних областях — від військової справи й систем безпеки до оцифрування різних аналогових сигналів.

Сприйняття явищ у формі образів відіграє надзвичайно важливу роль у процесах пізнання зовнішнього світу. У процесі біологічної еволюції багато тварин за допомогою зорового й слухового апарата вирішили задачу розпізнавання образів досить добре. Як випливає із самого означення образа "розпізнаванню" нових для нас об’єктів передує процес навчання. Під час навчання істоти ознайомлюються із деякою кількістю об’єктів і, крім цього, із якогось джерела (наприклад від батьків, старших і т.п.) отримують інформацію про те, до якого образу відноситься кожний із цих об’єктів. Цей процес отримав назву «**навчання з учителем**».

Більш загальний характер має "**навчання без учителя**", у процесі якого система вчиться спонтанно виконувати поставлене завдання без втручання з боку "вчителя". Навчання машин без вчителя формулюється як задача *кластерного аналізу*. Вибірка об’єктів розбивається на кластери (множини, що мають порожній перетин), таким чином, що кожний кластер складається із "схожих" об’єктів, а різні кластери "суттєво" відрізняються один від одного. Кластеризація часто використовується в якості допоміжного засобу розв’язування задач класифікації та регресійного аналізу. Деякі алгоритми розв’язування задач класифікації комбінують навчання з учителем та навчання без учителя (наприклад навчання мереж векторного квантування).

Традиційно задачі розпізнавання образів включають у коло задач

штучного інтелекту. Можна виділити два основних напрямки:

- Вивчення здібностей до розпізнавання, якими володіють живі істоти, їхнє пояснення й моделювання;

- Розвиток теорії й методів побудови пристроїв, призначених для розв'язання окремих задач у прикладних цілях.

Існує два підходи до навчання: індуктивне і дедуктивне. Індуктивне навчання, або навчання за прецедентами, засноване на виявленні загальних властивостей об’єктів на підставі неповної інформації, отриманих емпіричним шляхом. Дедуктивне навчання передбачає формалізацію знань експертів у вигляді баз знань (експертних систем тощо).

Слід зауважити, що, як кожна математична дисципліна, розпізнавання образів має власний математичний апарат, який включає математичну статистику, методи оптимізації, дискретну математику, алгебру і геометрію.

Розпізнавання образів має широке застосування і використовується при створенні усіх комп’ютерних систем, на які покладаються інтелектуальні функції, тобто функції, пов’язані із прийняттям рішень замість людини: медична діагностика, криміналістична експертиза, пошук інформації та інтелектуальний аналіз даних тощо.

Прецедент — це об’єкт, приналежність якого до заданого класу визначена заздалегідь. Прецедентом може бути, наприклад, набір ознак пацієнта із відомим діагнозом, з яким слід порівнювати набір ознак людини, діагноз якої ще невідомий.

Одне з найвдаліших формулювань ключової парадигми теорії розпізнавання таке: будь-який об’єкт у природі є унікальним, всі об’єкти є типізованими.

Зміст цієї парадигми такий. Кожний об’єкт характеризується тими чи іншими властивостями. Наявність чи відсутність таких властивостей, а також якісні та кількісні характеристики цих властивостей розглядаються як *ознаки об’єкта.* Унікальність будь-якого об’єкта означає те, що в природі не існує двох різних об’єктів, для яких збігаються абсолютно всі ознаки, а це

дозволяє, принаймні теоретично, відрізнити один об’єкт від іншого. Але деякі ознаки різних об’єктів можуть збігатися, і це дає підстави говорити про те, що ці об’єкти належать до одного типу або класу.

Кожний образ являє собою набір чисел, що описують його властивості і називаються *ознаками* (feature). Упорядкований набір ознак об’єкта називається *вектором ознак* (feature vector). Вектор ознак – це точка в *просторі ознак* (feature space).

Об’єктом у теорії розпізнавання прийнято називати будь-яку сутність, що існує або могла б існувати в реальному світі, а також будь-яке явище або процес.

Класом у теорії розпізнавання образів прийнято називати сукупність об’єктів, які мають ті чи інші спільні ознаки. Клас може об’єднувати фізично існуючі сутності (наприклад, людина, яблуко) або бути абстрактним поняттям (горе, економічний крах і т.п.).

Ознаки, що дають можливість відрізнити представників одного класу від іншого, прийнято називати *інформативними ознаками*.

Ознаки, спільні для всіх представників класу, називатимемо *інваріантами класу*.

Класифікатор, або *вирішальне правило* (decision rule) – це функція, яка ставить у відповідність вектору ознак образу клас, до якого він належить.

Задачу розпізнавання образів можна розділити на ряд підзадач.

1. *Генерування ознак* (feature generation) – вимірювання або обчислення числових ознак, що характеризують об’єкт.

2. *Вибір ознак* (feature selection) – визначення найбільш інформативних ознак для класифікації (в цей набір можуть входити не лише первинні ознаки, але й функції від них).

3. *Побудова класифікатора* (classifier construction) – конструювання вирішального правила, на підставі якого здійснюється класифікація.

4. *Оцінка якості класифікації* (classifier estimation) – обчислення показників правильності класифікації (точність, чутливість, специфічність,

помилки першого та другого роду).

Розпізнавання образів – це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних. При постановці задач розпізнавання намагаються користуватися математичною мовою.

Введемо позначення і сформулюємо математичну постановку задачі класифікації.

Нехай Ω – простір образів; $\omega \in \Omega$ – образ; $M = \{1, 2, ..., m\}$ – номери класів $\Omega_1, \Omega_2, ..., \Omega_m$, таких що $\Omega_i \bigcap \Omega_j$, якщо $i \neq j$ і $\bigcup_{i=1}^{m} \Omega_i = \Omega$; $g : \Omega \to M$ – індикаторна функція, що є невідомою; X — простір ознак, тобто векторний простір, точками якого є вектори ознак образів; $x : \Omega \to X$ – функція, що ставить у відповідність образу ω його вектор ознак $x(\omega)$; $K_1, K_2, ..., K_m$ – підмножини простору X, такі що $K_i \bigcap K_j$, якщо $i \neq j$ і $\bigcup_{i=1}^{m} K_i = X$; $\hat{g} : X \to M$ – вирішальне правило, яке ставить у відповідність вектору ознак образа номер класу, якому він належить.

Задача класифікації з учителем (supervised classification) полягає у тому, щоб на підставі множини прецедентів $\left(g_j, x_j\right)$, $j = 1, ..., N$, яка називається *навчальною вибіркою* (training sample) побудувати вирішальне правило $\hat{g}$, що мінімізує кількість помилок. Вчителем вважається або сама навчальна вибірка, або той, хто указав значення g_j.

Задача класифікації без учителя (unsupervised classification) часто називається *кластеризацією* (clusterization), або *кластерним аналізом* (cluster analysis). В цій задачі вибірка образів $x_j, j = 1, 2, ..., N$ розбивається на підмножини, що не перетинаються (кластери), які складаються із схожих один на одного об’єктів, до того ж вимагається, щоб об’єкти із різних кластерів істотно відрізнялися один від одного.

Розглянемо існуючі підходи до розпізнавання образів:

- Для оптичного розпізнавання образів можна застосувати метод перебору вигляду об'єкта під різними кутами, масштабами, зсувами й т. д. Для букв потрібно перебирати шрифт, властивості шрифту й т. д.;

- Другий підхід – знайти контур об'єкта й досліджувати його властивості (зв’язність, наявність кутів і т. д.);

- Ще один підхід – використовувати штучні нейронні мережі (багатошарові персептрони, мережі квантування, мапи Кохонена, рекурентні мережі). Цей метод вимагає або великої кількості прикладів задачі розпізнавання (із правильними відповідями), або спеціальної структури нейронної мережі, що враховує специфіку даної задачі.

Як один із методів розпізнавання обрізів розглядають так званий персептрон.

Ф. Розенблатт уводячи поняття про модель мозку, завдання якої полягає в тому, щоб показати, як у деякій фізичній системі, структура й функціональні властивості якої відомі, можуть виникати психологічні явища, – описав найпростіші експерименти з розрізнення. Дані експерименти цілком стосуються до методів розпізнавання образів, але відрізняються тим, що алгоритм розв'язання не детермінований.

Найпростіший експеримент, на основі якого можна одержати психологічно значиму інформацію про деяку систему, зводиться до того, що моделі пред'являються два різних стимули й потрібно, щоб вона реагувала на них різним чином. Метою такого експерименту може бути дослідження можливості спонтанного розрізнення стимулів системою при відсутності втручання з боку експериментатора, або, навпаки, вивчення примусового розрізнення, при якому експериментатор прагне навчити систему здійснювати необхідну класифікацію.

У досвіді з навчанням персептрону зазвичай пред'являється деяка послідовність образів, у яку входять представники кожного із класів, що підлягають розпізнаванню. Відповідно до деякого правила модифікації пам'яті правильний вибір реакції підкріплюється. Потім персептрону

пред'являється контрольний стимул і визначається ймовірність одержання правильної реакції для стимулів даного класу. Залежно від того, збігається чи не збігається обраний контрольний стимул з одним з образів, які використовувалися в навчальній послідовності, отримують різні результати.

Якщо контрольний стимул не збігається з жодним із навчальних стимулів, то експеримент пов'язаний не тільки з чистим розпізнаванням, але містить у собі й елементи узагальнення.

Якщо контрольний стимул збуджує деякий набір сенсорних елементів, цілком відмінних від тих елементів, які активізувалися при впливі раніше пред'явлених стимулів того ж класу, то експеримент є дослідженням чистого узагальнення.

Персептрони не мають здатності до чистого узагальнення, але вони цілком задовільно функціонують в експериментах із розпізнавання, особливо якщо контрольний стимул досить близько збігається з одним з образів, щодо яких персептрон уже нагромадив певний досвід.

Приклади задач розпізнавання образів:

- Розпізнавання літер;
- Розпізнавання штрих-кодів;
- Розпізнавання автомобільних номерів;
- Розпізнавання осіб;
- Розпізнавання мови;
- Розпізнавання зображень;
- Розпізнавання локальних ділянок земної кори, у яких знаходяться родовища корисних копалин.

Розпізнавання образів застосовується в наступних областях:

- Біоінформатика: пошук шаблонів в ДНК;
- Бази даних: пошук і класифікація;
- Обробка текстів: тематична класифікація;
- Аналіз зображень: розпізнавання символів, робота з картами, розпізнавання осіб, поділ об'єктів;

- Виробництво: контроль якості (візуальна перевірка коректності мікросхем);
- Пошук по мультимедіа: визначення жанрів;
- Біометрія: ідентифікація людини за відбитками пальців, по райдужній оболонці ока;
- Прогнозування: погода, сейсмологія, геологія;
- Обробка мови: переклад аудіо-сигналів в текст.

Повернемося до означення. *Розпізнавання образів* – процес віднесення об'єкта з фіксованою групою його властивостей до одного об'єкту з множини образів за заздалегідь обумовленим правилом. Наприклад, рибу у тенетах треба поділити на окунів і лососів. Припустимо, що це робиться по довжині риби. Тобто у нас є об'єкт "риба", і за значенням властивості "довжина" ми відносимо рибу або до образу "лосось", або до образу "окунь".

У процедурі розпізнавання образів можна виділити найбільш важливі кроки:

1. Сприйняття образу. На цьому етапі проводять отримання значень характеристичних властивостей об'єкта (вимірювання лінійних вимірів, фотографування, оцифровка звуку);
2. Попередня обробка. Видалення шумів, представлення зображення в чорно-білому варіанті, обрізання непотрібних частин зображення;
3. Виділення характеристик (індексація). На цьому етапі вимірюються характеристичні властивості об'єкта (вимірюємо довжину риби та її колір);
4. Класифікація (прийняття рішення).

Виділяють 4 групи методів розпізнавання:

1. Порівняння із зразком.

Застосовуємо геометричну нормалізацію і вважаємо відстань до прототипу. Найбільш наочно застосування цього методу в розпізнаванні тексту. Якщо маємо зображення відсканованого символу і колекція зображень зразків (усіх букв абетки), і потрібно визначити, якій букві алфавіту відповідає відскановане зображення, то для цього масштабуємо

зображення символу до розмірів зразків і вибираємо той, відстань до якого мінімальна;

2. Нейронні мережі.

Методи з використанням нейронних мереж, навідміну від інших, засновані на навчанні без учителя. При такому підході машина сама повинна визначити структуру класів і приналежність кожного образу певного класу. Нейронна мережа - це метод параметричної апроксимації, що довів свою корисність в побудові моделей щільності. Нейронні мережі зазвичай апроксимують деяку векторну функцію f деякого вхідного вектора рівнів x. Кожен рівень формує вектор виходів, кожен з яких представляє собою результат дії певної нелінійної функції.

При розпізнаванні методом найближчого сусіда результат методу був повністю визначений еталонними зразками, завантаженими в пам'ять перед обробкою. Тоді як при застосуванні нейронного методу розпізнавання кінцевий результат заздалегідь не відомий. Алгоритми нейронних мереж самі вирішують задачу вибору ознак. З цим пов'язаний один з недоліків - перенавчання.

Використання таких методів виглядає наступним чином: вибираємо вид мережі і налаштовуємо коефіцієнти, на вхід нейронної мережі подається об'єкт для розпізнавання, група рецепторів мережі відповідає за прийом своєї характеристичної властивості.

3. Статистичні методи.

Статистичні методи використовують похідну інформацію для вирішення задачі розпізнавання. В загальному випадку, даний метод зводиться до визначення ймовірності чи належить об'єкт до певного класу, якщо ознаки цього об'єкту отримали відповідні значення. Прикладом статистичного методу є байєсівські вирішальні правила.

Такі методи є одним із напрямків теорії розпізнавання образів, в основі якого лежить уявлення про клас розпізнаваних об'єктів як про ряд реалізацій деякої випадкової величини. Цю випадкову величину з більш-менш

визначеними статистичними характеристиками називають статистичною моделлю класу розпізнаваних об'єктів.

Якщо задані статистичні моделі, то методами математичної статистики можна побудувати алгоритм розпізнавання, оптимальний за тим чи іншим критерієм якості. У найбільш сприятливому випадку задана модель дозволяє вказати умови розподілу ймовірностей об'єктів кожного класу, що дає можливість використовувати для розпізнавання баєсове вирішальне правило або мінімаксне вирішальне правило.

У більш загальному випадку модель задається у вигляді випадкового поля, що залежить від цілого ряду постійних і змінних невідомих параметрів. Серед них цікаві лише значення параметра, що вказує клас кожного розпізнаваного об'єкта. Решта невідомих параметрів, переважно, заважають оцінці. Завдання визначення значень параметрів які заважають є завданням адаптації розпізнавання. При навчанні задається навчальна вибірка, що складається з об'єктів, класи яких вказані. На цій основі, в залежності від того, наскільки детально відомі статистичні характеристики даної моделі, будуються ті чи інші статистичні оцінки самих параметрів або певних функцій. Отримані оцінки потім використовуються в процесі вирішення завдання розпізнавання шляхом їх підстановки замість невідомих значень параметрів.

4. Структурні та синтаксичні методи.

Розбираємо об'єкт на елементи. Будуємо правило, в залежності від входження окремих елементів та їх послідовностей.

Структурний (лінгвістичній) підхід застосовується до задач розпізнавання образів, в яких важлива інформація про структуру конкретного об'єкта. Від процедури розпізнавання потрібна змога у визначенні об'єкту його класу та інформацію про об'єкт, який не дозволяє віднести його до іншого класу. Структурні методи ідентифікації об’єктів характеризуються своєю структурою значень і відношень. Структурний підхід ґрунтується на аналогії між синтаксисом та структурою реалізації образу.

Головна перевага структурних методів – це можливість створити велику кількість реалізацій представивши невелику множину елементів.

Недоліками цих методів є :

- обмеженість використання;
- прямі вирішальні правила не існують в даному методі.

Одним з структурних методів є класифікація відстані до найближчого сусіда, суть якого полягає в наступному, потрібно виділити невідомий вектор ознак x до такого класу, з яким даний вектор буде найбільш близький. Класифікація методу також може бути більш ефективна навіть якщо класи накладаються один на одний або мають складну структуру.

В стуктурному методі припущення про моделі розподілу векторів ознак в просторі зайві, а отже алгоритм використовує тільки інформацію з відомими еталонними зразками. Метод заснований на розрахунку відстані x до кожного зразка в базі даних і знаходження мінімальної відстані. Переваги такого методу наступні:

- ієрархічні структури даних дозволяють значно зменшити кількість обчислювань;
- в базу даних можна додати нові зразки.

Якщо в базі даних виконувати пошук k сусідів то рішення буде набагато краще. Отже при кількості $k > 1$ можна забезпечити найкращий варіант вибірки та розподілу векторів в просторі.

Методи машинного навчання і прийняття рішень є фінальною стадією в розпізнаванні образів. Вони знаходяться на стику математичної статистики, методів оптимізації та класичних математичних дисциплін, але має також і власну специфіку, пов'язану з проблемами обчислювальної ефективності та перенавчання. У більшості випадків суть навчання полягає в наступному: на основі навчальної вибірки з ознаками кожного класу побудувати таку модель, за допомогою якої машина зможе проаналізувати нове зображення і вирішити, який з об'єктів є на зображенні.

1.3 Сфери застосування розроблюваного проекту

На сьогоднішній день, можна спостерігати стрімкий розвиток сучасних технологій, тому з кожним днем з'являється все більше сфер для їх застосування.

Реалізація даного проекту заснована на поєднанні елементів двох технологій – Computer Vision та Machine Learning, а саме виділення контуру на зображенні та розпізнавання зображених об'єктів. Цей проект можна розглядати як повноцінний застосунок для розпізнавання контурних зображень, або як частину масивного алгоритму для його ефективного застосування в окремій вузьконаправленній спеціальності. Таким чином, можна зробити висновок, що до сфер застосування розроблюваного програмного засобу можна віднести ті, де доцільно використовувати саме ці технології.

Великою мірою, направленість застосування залежить від того, на розпізнавання яких об'єктів навчена використовувана модель TensorFlow Lite. Навчання моделі для точного розпізнавання будь-яких зображених об’єктів є доволі складною операцією, яка потребує великого об’єму роботу, часу, а також людських та апаратних витрат. Тому значно ефективніше буде застосувати дану технологію в окремій сфері та навчати модель розпізнавати відповідні об’єкти.

Розглянемо деякі приклади застосування розроблюваного програмного засобу:

- Військова розвідувальна техніка.

Ця сфера є доволі актуальною в наш час, її ефективне використання часто стає вирішальним у сучасних протистояннях з ворогом. В даному випадку, розроблювана технологія допоможе розпізнати ворожу техніку на полі бою. За допомогою технології Machine Learning, застосунок зможе розпізнати та відобразити яка саме техніка знаходиться на полі бою та навести деяку довідкову інформацію про неї, наприклад, тип техніки, калібр,

кількість особового складу тощо. А використання Computer Vision допоможе чітко розпізнати межі техніки, відмалювавши її контур, що значно спростить коригування вогню. Ефективним прикладом використанні розроблюваного проекту у даній сфері, можна назвати аналіз зображень отриманих з безпілотних літальних апаратів або дронів.

- Контроль якості у промисловості.

Зазвичай, у промисловості при виробництві тієї чи іншої продукції, після завершення виробу, продукція проходить процедуру контролю якості для того, щоб у продаж поступив товар належної якості. За допомогою розроблюваного проекту, можна автоматизувати даний процес. Модель TensorFlow Lite, спеціально навчена для відповідних цілей, зможе розпізнати на виробі різноманітні дефекти: подряпини, тріщини, сколи тощо. А розпізнавання контуру об’єкта допоможе наглядніше продемонструвати розташування, розмір та межі знайдених дефектів, а також проаналізувати правильність форми виробу. З цього можна зробити висновок, що використання такої системи значно підвищить якість кінцевої продукції.

- Автомобільна техніка.

Використанням сучасних технологій в автомобільній техніці вже нікого не здивуеш, вже багато років автомобілі випускаються оснащені камерами, різноманітними системами безпеки, датчиками тощо. Проте, використання розроблюваного програмного засобу, у якості частини алгоритму, допоможе покращити системи розпізнавання перешкод. Дана технологія зможе проаналізувати зображення з камер та попередити водія про перешкоду на дорозі. Розпізнавши перешкоду, для того, щоб не відволікати увагу водія, слід передати отрмину інформацію голосовому помічнику, який і попередить водія про небезпеку, або відобразити контур перешкоди на екрані бортового комп’ютера, якщо це, наприклад, система паркування.

Це лише декілька приклідів, однак по ним можна зробити чіткий висновок, що сфери застосування розроблюваної технології можуть бути абсолютно різні, а користь від її використання доволі значна.

1.4 Постановка задачі

Актуальною задачею даної роботи є розробка мобільного додатку на базі операційної системи Android, який поєднує в собі елементи технологій Computer Vision та Machine Learning, що дає можливість виділити контур на зображенні та розпізнати який саме об’єкт зображено на знімку.

Реалізація програмного засобу відбувається за допомогою мови програмування Kotlin у середовищі розробки Android Studio. Для розпізнавання контуру на зображенні необхідно використати бібліотеку OpenCV, яка призначена для виконання функцій комп’юетрного зору в режимі реального часу. А для розпізнавання об’ктів на зображенні слід використати бібліотеку TensorFlow Lite, яка розроблена для розгортування моделей машинного навчання на мобільних пристроях, мікроконтролерах та інших периферійних пристроях і перевагою якої є те, що вона здатна швидко виконувати розпізнавання без підключення до мережі інтернет.

Мобільний додаток повинен мати змогу безперебійно працювати на будь-якому пристрої з операційною системою Android версії 5.0 та вище. Крім цього, повинен мати змогу отримати доступ до зовнішньої камери та внутрішньої галереї зображень.

Інтерфейс розроблюваного додатку пивинен мати такі можливості:

- Відображувати зображення з видошукача зовнішньої камери та мати змогу виконувати усі її базові функції (автофокус, зум тощо);
- Зробити новий знімок та передати його для подальшого аналізу;
- Завантажити зображення для аналізу з внутрішньої галереї;
- Перехід на наступний екран для відображення результату аналізу;
- Відмалювання контуру зображення;
- Відображення результату розпізнавання зображених об’єктів з указанням проценту впевненності розпізнавання;
- Перемикання між режимами відображення (тільки контур або контур поверх зображення);

- Повернення до попереднього екрану;
- Збереження зображення з нанесеним контуром до внутрішньої галереї зображень.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1 Computer Vision для розпізнавання контурів на зображенні

Комп'ютерний зір – це область штучного інтелекту, яка навчає комп'ютери інтерпретувати і розуміти візуальний світ. Використовуючи цифрові зображення з камер, відео та моделі глибокого навчання, комп'ютери точно ідентифікують і класифікують об'єкти, а потім реагують, коли "бачать" їх знову.

Перші експерименти в області комп'ютерного зору проводилися в 1950-х роках. У них використовувалися деякі з перших нейронних мереж для виявлення країв об'єкта і для об'єднання простих об'єктів в категорії – кола і квадрати. У 1970-х роках перше комерційне використання комп'ютерного зору полягало в інтерпретації друкованого або рукописного тексту з використанням оптичного розпізнавання символів. Це просування було використано для інтерпретації письмового тексту для незрячих людей.

У міру того, як в 1990-х роках Інтернет розвивався, роблячи безліч зображень доступними онлайн для аналізу, програми розпізнавання осіб процвітали. Ці зростаючі обсяги даних допомогли машинам ідентифікувати конкретних людей на фотографіях і відео.

Сьогодні декілька факторів об'єдналися, щоб відродити комп'ютерний зір:

- Мобільні технології з вбудованими камерами наповнили світ фотографіями і відео;
- Обчислювальна потужність стала більш доступною;
- Обладнання для комп'ютерного зору і аналізу стало широко доступно;
- Нові алгоритми, такі як згорткові нейронні мережі, можуть використовувати апаратні і програмні можливості.

Вплив цих досягнень на область комп'ютерного зору вражає. Точність визначення і класифікації об'єктів менш ніж за десятиліття зросла з 50 до 99%. Сучасні системи швидше і більш коректно, ніж люди виявляють і реагують на візуальні сигнали. Від розпізнавання облич до обробки рухів на футбольному матчі, комп'ютерний зір перевершує візуальні здібності людини в багатьох областях.

Робота комп'ютерного зору складається з трьох основних етапів:

- Отримання зображення: зображення, навіть важкі, можна отримувати в режимі реального часу за допомогою відео, фотографій або 3D-технологій для аналізу;

- Обробка зображення: моделі глибокого навчання автоматизують більшу частину цього процесу, але моделі часто навчаються, спочатку отримуючи тисячі позначених або попередньо ідентифікованих зображень;

- Розуміння зображення: етап інтерпретації, коли об'єкт ідентифікується або класифікується.

Сьогоднішні системи ШІ йдуть далі і вживають заходи, засновані на розумінні образу. Існує багато типів комп'ютерного зору, які використовуються по-різному:

- Сегментація зображення розбиває зображення на кілька областей або фрагментів для окремого дослідження;

- Виявлення об'єкта ідентифікує конкретний об'єкт на зображенні. Розширене виявлення об'єктів розпізнає безліч об'єктів в одному зображенні: футбольне поле, нападник, захисник, м'яч і так далі. Ці моделі використовують координати X, Y, щоб створити обмежувальну рамку та ідентифікувати все всередині неї;

- Розпізнавання облич – це розширений тип виявлення об'єктів, який не тільки розпізнає людське обличчя на зображенні, але і ідентифікує конкретну людину;

- Виявлення контуру – це метод, який використовується для визначення зовнішнього краю об'єкта або ландшафту, щоб краще визначити,

що знаходиться на зображенні;

- Розпізнавання образів – це процес розпізнавання повторюваних форм, кольорів і інших візуальних індикаторів на зображеннях;
- Класифікація зображень групує зображення в різні категорії;
- Зіставлення ознак – це тип виявлення шаблонів, який зіставляє подібності в зображеннях, щоб допомогти їх класифікувати.

Прості програми комп'ютерного зору використовують тільки один з цих методів, але більш складні, такі як комп'ютерний зір для автомобілів з самостійним водінням, покладаються на різні методи для досягнення своєї мети.

Комп’ютерний зір працює майже так само, як людський зір, за винятком того, що люди мають фору. Людський зір має перевагу у часі життя контексту, щоб навчитись розрізняти предмети, як далеко вони знаходяться, чи рухаються вони і чи є щось не так у зображенні.

Комп’ютерний зір навчає машини виконувати ці функції, але він повинен це робити за значно менший час за допомогою камер, даних та алгоритмів, а не сітківки, зорових нервів та зорової кори. Оскільки система, навчена контролювати продукцію або спостерігати за виробничим активом, може щохвилини аналізувати тисячі продуктів або процесів, помічаючи непомітні дефекти або проблеми, вона може швидко перевершити людські можливості.

Комп’ютерному зору потрібно багато даних. Він проводить аналіз даних знову і знову, поки не розпізнає відмінності і в кінцевому рахунку розпізнає зображення. Наприклад, щоб навчити комп’ютер розпізнавати автомобільні шини, йому потрібно подавати величезну кількість зображень шин та предметів, пов’язаних із шинами, щоб дізнатися відмінності та розпізнати шину, особливо таку, що не має дефектів.

Для цього використовуються дві основні технології: глибоке навчання, або Deep Learning, та згорткова нейронна мережа (CNN).

CNN допомагає машинному навчанню або моделі глибокого навчання

«дивитись», розбиваючи зображення на пікселі, яким присвоюються теги або мітки. Вона використовує мітки для виконання звивин (математична операція над двома функціями для отримання третьої функції) і робить прогнози щодо того, що він «бачить». Нейронна мережа запускає згортки і перевіряє точність своїх прогнозів у серії ітерацій, поки прогнози не почнуть здійснюватися. Потім вона розпізнає або бачить зображення майже як людина.

CNN спочатку розрізняє жорсткі краї та прості форми, а потім заповнює інформацію, виконуючи ітерації своїх прогнозів. Вона використовується для розуміння окремих зображень, а для відеопрограм подібним чином використовується рекурентна нейронна мережа (RNN).

2.2 Machine Learning для розпізнавання зображених об’єктів

Машинне навчання (Machine Learning, ML) – це галузь штучного інтелекту (ШІ) та комп’ютерних наук, яка зосереджена на використанні даних та алгоритмах для імітації способу навчання людей, поступово покращуючи його точність.

Першу програму на основі алгоритмів, здатних самонавчатися, розробив Артур Самуель (Arthur Samuel) в 1952 році, призначена вона була для гри в шашки. Самуель дав і перше визначення терміну «машинне навчання»: це «область досліджень розробки машин, які не є заздалегідь запрограмованими». Більш точне визначення терміну «навчання» дав набагато пізніше Т. М. Мітчелл: кажуть, що комп'ютерна програма навчається на основі досвіду E по відношенню до деякого класу задач T і міри якості P, якщо якість вирішення завдань з T, виміряний на основі P, поліпшується з набуттям досвіду E.

Вже в 1957 році була запропонована перша модель нейронної мережі, що реалізує алгоритми машинного навчання, схожі на сучасні. В даний час ведеться розробка найрізноманітніших систем машинного навчання,

призначених для використання в таких технологіях майбутнього, як Інтернет Речей, Промисловий Інтернет Речей, в концепції «розумне» місто, при створенні безпілотного транспорту і в багатьох інших.

Розрізняють два типу машинного навчання. Навчання по прецедентах, або індуктивне навчання, засноване на виявленні загальних закономірностей по приватним емпіричним даним. Дедуктивне навчання передбачає формалізацію знань експертів і їх перенесення в комп'ютер у вигляді бази знань. Дедуктивне навчання прийнято відносити до області експертних систем, тому терміни машинне навчання і навчання по прецедентах можна вважати синонімами. Цей метод навчання зараз є доволі популярним, а ось експертні системи переживають кризу. Бази знань, що лежать в їх основі, важко узгоджувати з реляційною моделлю даних, тому промислові СУБД неможливо ефективно використовувати для наповнення баз знань експертних систем.

Навчання по прецедентах, в свою чергу, поділяють на три основних типи:

- контрольоване навчання, або навчання з учителем (supervised learning);
- неконтрольоване навчання (unsupervised learning), або навчання без учителя;
- навчання з підкріпленням (reinforcement learning).

Крім названих, розробляються і інші методи навчання: активне, багатозадачне, різноманітне, трансферне і т.д. Особливо успішно розвивається в останні роки «глибоке навчання», при використанні якого можуть успішно поєднуватися алгоритми навчання з вчителем і без вчителя.

Машинне навчання знаходиться на стику математичної статистики, методів оптимізації та класичних математичних дисциплін, але має також і власну специфіку, пов'язану з проблемами обчислювальної ефективності та перенавчання. Багато методів індуктивного навчання розроблялися як альтернатива класичним статистичним підходам. Велика кількість методів

тісно пов'язана з витягуванням інформації та інтелектуальним аналізом даних.

Машинне навчання – не тільки математична, а й практична, інженерна дисципліна. Чиста теорія, як правило, не призводить відразу до методів і алгоритмів, які можуть застосовуватися на практиці. Щоб змусити їх добре працювати, доводиться винаходити додаткові евристики, що компенсують невідповідність зроблених в теорії припущень умовам реальних завдань. Практично жодне дослідження в машинному навчанні не обходиться без експерименту на модельних або реальних даних, що підтверджує практичну працездатність методу.

Систему навчання алгоритму машинного навчання розбивають на три основні частини:

- Процес прийняття рішень: загалом алгоритми машинного навчання використовуються для прогнозування або класифікації. На основі деяких вхідних даних, які можуть бути позначені або не позначені, ваш алгоритм створить оцінку щодо шаблону в даних;

- Функція помилки: функція помилки служить для оцінки передбачення моделі. Якщо є відомі приклади, функція помилки може провести порівняння для оцінки точності моделі;

- Процес оптимізації моделі: якщо модель може краще відповідати точкам даних навчального набору, то ваги коригуються, щоб зменшити невідповідність між відомим прикладом та оцінкою моделі. Алгоритм буде повторювати цей процес оцінки та оптимізації, оновлюючи ваги автономно, доки не буде досягнуто поріг точності.

3 ОПИС ВИКОРИСТОВУВАНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1 Мова програмування Kotlin

Kotlin – це універсальна, кросс-платформена, безкоштовна, статично типізована “прагматична” мова програмування з відкритим кодом, яка була спочатку розроблена для JVM (Java Virtual Machine) та Android. Вона поєднує елементи об’єктно-орієнтованого та функціонального програмування. Kotlin орієнтований на сумісність, безпеку, чистоту та підтримку інструментів. Також виготовляються версії Kotlin, орієнтовані на JavaScript ES5.1 та нативний код (з використанням LLVM) для ряду процесорів. Kotlin виник в 2010 році в компанії JetBrains, що стоїть за створенням середовища розробки IntelliJ IDEA, і з 2012 року став мовою програмування з відкритим кодом.

На Google I/O 2019 було оголошено, що Android-розробка буде все більше орієнтована на Kotlin і що тепер віддається перевага саме цій мові програмування при розробці нових додатків для операційної системи Android. Таким чином, Kotlin став офіційною мовою програмування для Android розробників. Kotlin – це виразна та стисла мова програмування, яка зменшує кількість загальних помилок в коді та легко інтегрується в існуючі програми. Тому при створенні нового додатку для Android, Google радить починати з Kotlin, щоб скористатися його найкращими у своєму класі функціями.

Ось які основні переваги виділяють розробники при написанні програмного коду на Kotlin:

- Виразність і лаконічність: Ви можете робити більше з меншою кількістю коду. Можна висловлювати свої ідеї та зменшити кількість шаблонного коду. 67% професійних розробників кажуть, що Kotlin збільшив їх продуктивність.

- Безпечніший код: Kotlin має безліч мовних функцій, які допоможуть вам уникнути поширених помилок програмування, таких як NullPointerException. Додатки Android, які написані на Kotlin, мають на 20% менше шансів вийти з ладу.

- Сумісність: Можливо викликати код на базі Java з Kotlin або викликати Kotlin з коду на основі Java. Kotlin на 100% взаємодіє з мовою програмування Java, тому ви можете стільки коду на Kotlin у своєму проекті, скільки завгодно.

- Структурована багатопоточність: програми Kotlin роблять асинхронний код таким же простим у роботі, як і блокуючий код. Програми суттєво спрощують управління фоновими задачами для всього, від мережевих викликів до доступу до локальних даних.

Серед недоліків можна виділити низьку швидкість. Найчастіше розробники скаржаться на непередбачувану швидкість компіляції. За швидкодією Kotlin поступається Java, оскільки в його основі лежить віртуальна машина JVM – фундаментальна програма, випущена спеціально під Java, а не під Kotlin.

Крім цього, Kotlin має всі особливості та переваги функціональної мови. Дозвіл функцій верхнього рівня – це лише початок історії функціонального програмування для Kotlin. Мова також підтримує функції вищого порядку, анонімні функції, лямбди, вбудовані функції, closures, хвостові рекурсії та узагальнення.

Kotlin – це наступний етап розвитку Java, з якої він повністю сумісний. Це робить його відмінним інструментом для мобільних та ентерпрайз-додатків. А також, його було визнано офіційною мовою програмування Android. Тому саме цю мову було обрано основною для реалізації даного проекту.

3.2 Середовище розробки Android Studio

Найкращий спосіб розпочати розробку Android-додатків на мові програмування Kotlin – встановити середовище розробки Android Studio, в комплекті з якою поставляється Android SDK (Software Development Kit), це дасть все необхідне для початку роботи.

Android Studio – це офіційне інтегроване середовище розробки (IDE) для розробки додатків для Android, створене на основі IntelliJ IDEA. На додаток до потужного редактора коду та інструментів розробника IntelliJ, Android Studio пропонує ще більше функцій, що підвищують вашу продуктивність при створенні програм для Android, таких як:

- Гнучка система побудови на основі Gradle;
- Швидкий і багатофункціональний емулятор;
- Єдине середовище, де можна розробляти для всіх Android пристроїв;
- Можливість застосовувати зміни коду та ресурсів до запущеної програми без її перезапуску;
- Шаблони коду та інтеграція з GitHub, щоб допомогти вам створити загальні функції програми та імпортувати зразок коду;
- Широкі інструменти та фреймворки для тестування;
- Інструменти Lint для виявлення проблем продуктивності, зручності використання, сумісності версій та інших проблем;
- Підтримка C ++ та NDK;
- Вбудована підтримка Google Cloud Platform, що спрощує інтеграцію з Google Cloud Messaging та App Engine.

Головне вікно Android Studio складається з декількох логічних областей, позначених на рисунку 3.1.

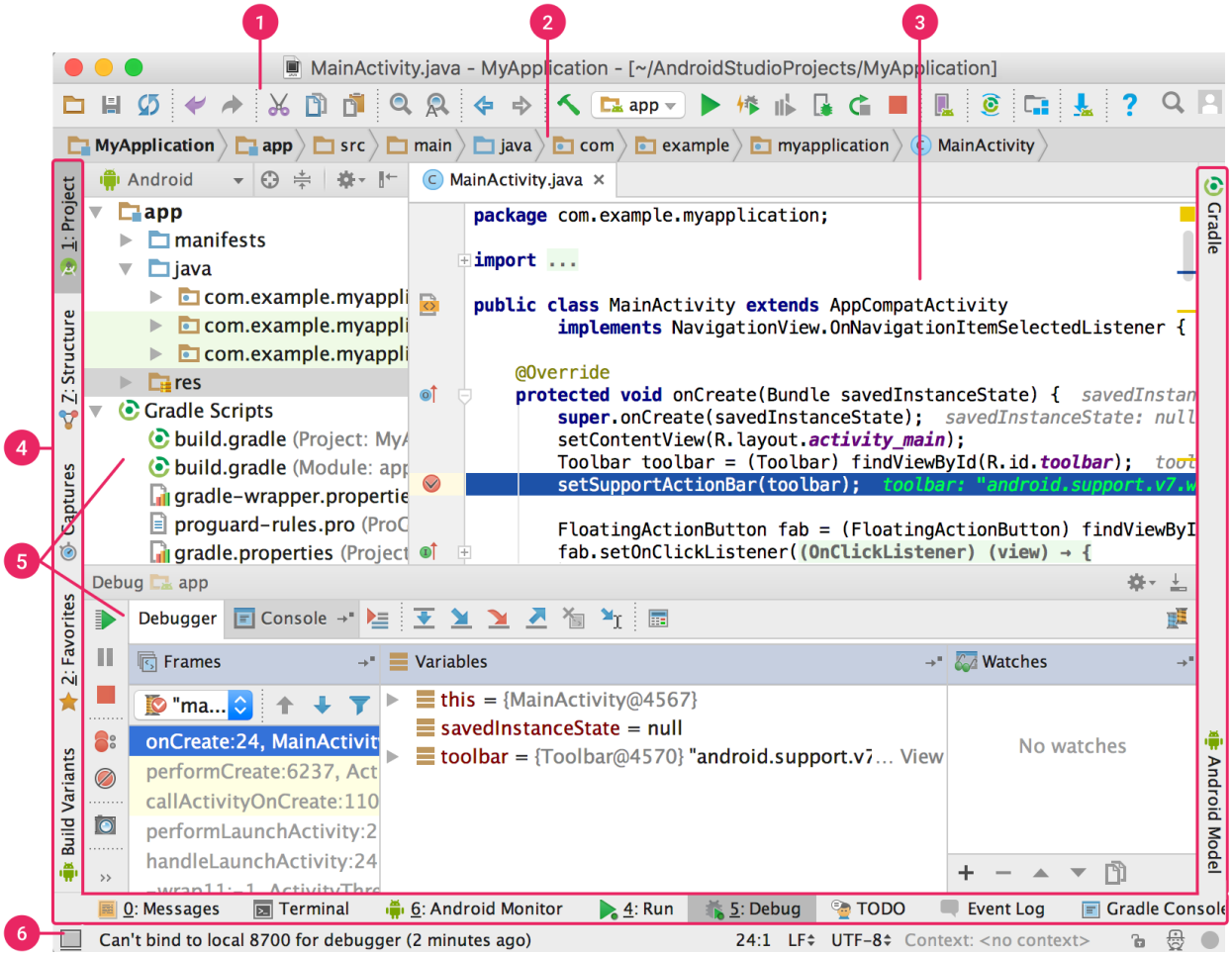

Рисунок 3.1 – Головне вікно Android Studio

1. Панель інструментів, що дозволяє виконувати широкий спектр дій, включаючи запуск програми та запуск інструментів Android.

2. Панель навігації допомагає переміщатися по проекту та відкривати файли для редагування. Це забезпечує більш компактний вигляд структури, видимої у вікні проекту.

3. Вікні редактора, де створюється та змінюється код. Залежно від поточного типу файлу редактор може змінюватися. Наприклад, під час перегляду файлу макета редактор відображає редактор макетів.

4. Панель вікон інструментів розташована на зовнішній частині вікна IDE і містить кнопки, що розгортають або згортають окремі вікна інструментів.

5. Вікна інструментів дають доступ до спеціальних задач, таких як управління проектами, пошук, контроль версій тощо.

6. Рядок стану відображає стан вашого проекту та самого IDE, а також будь-які попередження або повідомлення.

Важливою частиною розробки є Android SDK. Це набір укомплектованих файлів, які необхідні для початку створення Android-додатків. Він містить такі інструменти, як віртуальний менеджер пристроїв (для запуску емуляторів) та ADB міст, а також бібліотека додаткового коду для того, щоб програми Kotlin або Java працювали з платформою Android.

Важливою частиною цього пакету є інструменти SDK-tools. Це те, що фактично створює APK – перетворює Kotlin або Java програму на Android-додаток, який можна запустити на телефоні. APK – це формат файлу, який використовується системою Android для розповсюдження та встановлення мобільних додатків. До SDK-tools відносяться інструменти збирання проекту, інструменти відладки та інструменти для зображень.

Також важливим компонентом є ADB (Android Debug Bridge) – це програма, яка дозволяє комунікувати з будь-яким пристроєм на системі Android. Він покладається на інструменти платформи для того, щоб зрозуміти яка версія Android використовується на вказаному пристрої. ADB можна використовувати для доступу до інструментів оболонки, для запиту ідентифікатора пристрою або навіть для встановлення програм.

Ще одним корисним інструментом при розробці мобільного додатку в Android Studio є Android Lint – це статичний аналізатор коду. Статичні аналізатори перевіряють код на наявність дефектів, не виконуючи його. Android Lint використовує своє знання інфраструктури Android для перевірки коду та виявлення проблем, які компілятор виявити не може. Lint виділяє рядок в коді, щоб повідомити про наявність в ньому проблем. Як правило, до рекомендацій Android Lint варто прислухатися.

Емулятор Android – це те, що дозволяє тестувати та відстежувати мобільні додатки на ПК, тобто не обов’язково мати пристрій у наявності. Таким чином, можна отримати віртуальний пристрій Android прямо на вашому комп’ютері. Для вибору версії Android та технічних характеристик

пристрою використовується диспетчер віртуальних пристроїв Android.

Як основу системи збірки Android Studio використовує Gradle, з додатковими функціями Android, які надає спеціальний плагін. Ця система побудови працює як інтегрований інструмент із меню Android Studio та не залежить від командного рядка. Застосовуючи гнучкість Gradle, можна виконувати різноманітні налаштування процесу збірки, не змінюючи основні вихідні файли програми. Файли збірки Android Studio називаються build.gradle. Вони являють собою текстові файли, які використовують синтаксис Groovy для налаштування збірки з елементами, наданими плагіном для Gradle.

Візуальна частина мобільного додатку створюється за допомогою мови розмітки XML, що визначає макет документу – так само, як HTML, який використовується для створення веб-сайтів. Елементи макету екрана мобільного додатку називають віджетами. Віджети – це структурні елементи, із яких створюється інтерфейс користувача. Віджет може відображувати текст чи графіку, взаємодіяти із користувачем або розміщувати інші віджети на екрані. Кнопки, текстові поля, прапорці – все це різновиди віджетів.

Android SDK включає в себе безліч віджетів, які можна налаштовувати для отримання необхідного оформлення та поведінки. Кожен віджет є екземпляром класу View або одного із його підкласів. Додавати віджети можна за допомогою візуального редактора або безпосередньо в XML розмітку. Кожному віджету в макеті відповідає елемент XML. Ім’я елемента визначає тип віджета. Кожен елемент володіє набором атрибутів XML, які можна розглядати як інструкції для налаштування віджетів. В макеті віджети розміщуються в ієрархічному вигляді. Таким чином, XML розмітка відповідає за те, як віджети будуть відображуватися на екрані, а їх функціональна частина вже описується мовою програмування Java.

При збиранні проекту вміст макетів XML перетворюється в об’єкти класа View. За цей процес відповідає утиліта aapt (Android Asset Packaging Tool), яка в процесі збирання компілює ресурси файлів макетів в більш

компактний формат. Відкомпільовані ресурси пакуються в файл APK.

Отже, за допомогою середовища розробки Android Studio та мови програмування Kotlin можна легко створювати, запускати, тестувати та відстежувати власні Android-додатки.

3.3 OpenCV

OpenCV – це бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом, розроблена компанією Intel на мові програмування C/C++. Також, вона існує для деяких інших мов, наприклад, для Java. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

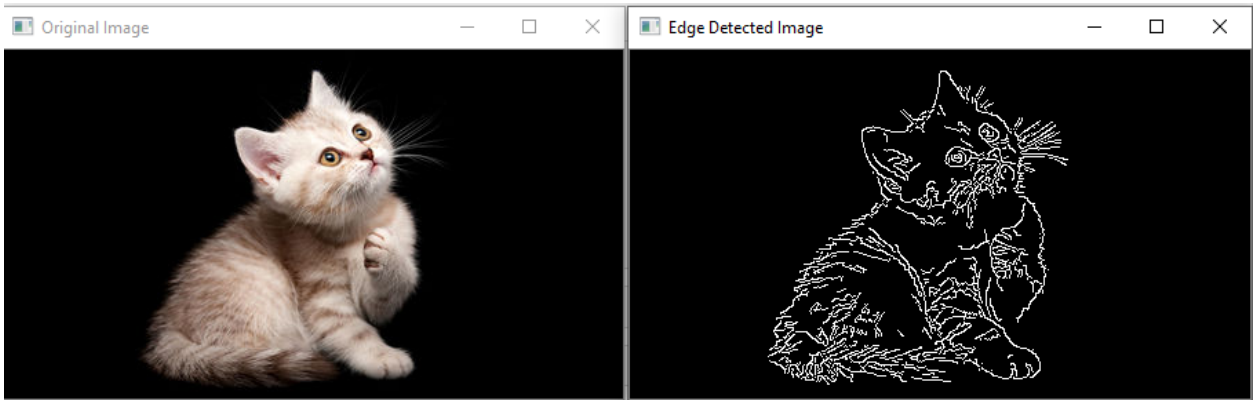

Рисунок 3.2 – Приклад застосування бібліотеки OpenCV

Офіційно проект OpenCV був запущений у 1999 році за ініціативою Intel Research з ціллю розвивати CPU-ресурсомісткі додатки. Основними вкладниками у проект була Intel's Performance Library Team та певна кількість експертів з оптимізації у Intel Russia. На перших етапах розвитку OpenCV основними задачами бібліотеки були:

- Розвивати дослідження у напрямку комп'ютерного зору, забезпечуючи добре оптимізований та відкритий код бібліотеки.

- Поширювати знання у сфері комп'ютерного зору, забезпечуючи загальну інфраструктуру, яку б могли розвивати розробники, таким чином код ставатиме більш легким для сприйняття та обміну.

- Розвивати засновані на роботі з комп'ютерним зором комерційні додатки, створюючи не залежну від платформи, оптимізовану та безкоштовну бібліотеку. Для цього використовувалася ліцензія, яка не вимагала від таких комерційних додатків бути відкритими.

Бібліотека має понад 2500 оптимізованих алгоритмів, що включає повний набір як класичних, так і найсучасніших алгоритмів комп’ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації людських дій у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок зі стереокамер, зшивання зображень для отримання високої роздільної здатності зображення цілої сцени, знайти подібні зображення з бази даних зображень, видалити червоні очі із зображень, зроблених за допомогою спалаху, слідкувати за рухами очей, розпізнавати пейзаж та встановлювати маркери, щоб накласти їх на доповнену реальність тощо. OpenCV має понад 47 тисяч користувачів користувачів спільноти та передбачає кількість завантажень, що перевищує 18 мільйонів. Бібліотека широко використовується у компаніях, дослідницьких групах та державних органах.

Вона має інтерфейси C ++, Python, Java та MATLAB та підтримує Windows, Linux, Android та Mac OS. OpenCV схиляється здебільшого до програм для зору в реальному часі та використовує переваги інструкцій MMX та SSE, коли вони доступні. Зараз розробляються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми.

3.4 Бібліотека TensorFlow Lite

TensorFlow – це наскрізна платформа з відкритим кодом для машинного навчання. Вона має всеосяжну, гнучку екосистему інструментів, бібліотек і ресурсів спільноти, що дозволяє дослідникам використовувати найсучасніші технології машинного навчання, а розробникам легко створювати й розгортати програми, що працюють на ML.

Від самого початку TensorFlow був розроблений дослідниками та інженерами, які працювали в команді Google Brain в організації Machine Intelligence Research від Google для проведення машинного навчання та дослідження глибоких нейронних мереж. Але система досить універсальна, щоб її можна було застосувати і в багатьох інших областях.

TensorFlow надає набір робочих процесів з інтуїтивно зрозумілими високорівневими API як для початківців, так і для експертів для створення моделей машинного навчання багатьма мовами. Розробники мають можливість розгортати моделі на ряді платформ, наприклад на серверах, у хмарі, на мобільних і периферійних пристроях, у браузерах та на багатьох інших платформах JavaScript. Це дозволяє розробникам набагато легше переходити від створення моделі та навчання до розгортання.

Переваги платформи TensorFlow:

- Проста побудова моделі.

Можливість легко створювати та навчати моделі ML, використовуючи інтуїтивно зрозумілі API високого рівня, такі як, наприклад, Keras, із швидким виконанням, що забезпечує негайну ітерацію моделі та легке відлагодження;

- Надійна робота ML в будь-якому місці.

Можливість легко навчати й розгортати моделі в хмарі, локально, у веб-браузері чи на пристрої, незалежно від мови, яку ви використовуєте;

- Потужні експерименти для дослідження.

Проста та гнучка архітектура для швидшого перенесення нових ідей від

концепції до коду, до найсучасніших моделей та до публікації.

Також, варто відзначити, що TensorFlow використовується багатьма внутрішніми продуктами та командами Google, зокрема: Search, Gmail, Translate, Maps, Android, Photos, Speech, YouTube та Play.

TensorFlow може навчати та запускати глибокі нейронні мережі для рукописної класифікації цифр, розпізнавання зображень, вбудовування слів, рекурентних нейронних мереж, sequence-to-sequence моделей для машинного перекладу, обробки природною мовою та моделювання на основі PDE (діференціальне рівняння з частинними похідними). Найкраще те, що TensorFlow підтримує прогнозування роботи в масштабі, з тими ж моделями, що використовуються для навчання.

TensorFlow дозволяє створювати графіки потоків даних і структури, щоб визначити, як дані переміщуються по графіку, беручи вхідні дані як багатовимірний масив, який називається Tensor. Це дозволяє вам побудувати блок-схему операцій, які можна виконати над цими вхідними даними, які входять на одному кінці і виходить на іншому кінці як вихідні дані.

Ось чому ця платформа називається TensorFlow, оскільки в ній тензор проходить через список операцій (flowchart), а потім виходить з іншого боку.

Рисунок 3.3 – Діаграма TensorFlow

TensorFlow Lite – це мобільна бібліотека для розгортування моделей на мобільних, мікроконтролерних та інших периферійних пристроях.

Ключові особливості TensorFlow Lite:

- Оптимізовано для машинного навчання на пристрої, усуваючи 5 ключових обмежень: затримка (немає зворотного зв'язку до сервера), конфіденційність (особисті дані не залишають пристрій), підключення (підключення до Інтернету не потрібне), розмір (зменшена модель і бінарний розмір) і енергоспоживання (ефективний висновок і відсутність мережевих підключень);

- Підтримка кількох платформ, яка охоплює пристрої з операційною системою Android та iOS, вбудовувані системи Linux та мікроконтролери;

- Підтримка різноманітних мов програмування, що включає Java, Swift, Objective-C, C++ та Python;

- Висока продуктивність з апаратним прискоренням і оптимізацією моделі;

- Наскрізні приклади на кількох платформах для звичайних завдань машинного навчання, таких як класифікація зображень, виявлення об'єктів, оцінка пози, відповіді на запитання, класифікація тексту тощо.

Рисунок 3.4 – Приклад використання TensorFlow Lite

TensorFlow Lite – це полегшена версія оригінального TensorFlow (TF). TF Lite спеціально розроблений для мобільних обчислювальних платформ і

вбудованих пристроїв, ігрових консолей і цифрових камер та інших периферійних пристроїв. Передбачається, що TensorFlow Lite надає можливість виконувати передбачення на вже навченій моделі (завдання логічного висновку).

TensorFlow, з іншого боку, використовується для побудови та навчання моделі ML. Іншими словами, TensorFlow призначений для навчальних моделей, тоді як TensorFlow Lite більш корисний для логічних висновків та периферійних пристроїв. TensorFlow Lite також оптимізує навчену модель за допомогою методів квантування, що зменшує необхідне використання пам’яті, а також обчислювальні витрати на використання нейронних мереж.

4 ПРИНЦИП РОБОТИ ТА АНАЛІЗ МОБІЛЬНОГО ДОДАТКУ

4.1 Алгоритм виявлення контуру

Контур об'єкта – це його видимий край, який відокремлює об'єкт від фону. Насправді, більшість методів аналізу зображень працюють саме з контурами, а не з пікселями як такими. Сукупність методів роботи з контурами називається контурним аналізом. При проведенні контурного аналізу вважається, що контур містить достатню інформацію про форму об’єкта, а його внутрішні точки не враховуються.

Існують деякі обмеження на область застосування контурного аналізу, які, в основному, пов'язані з проблемами виділення контуру на зображеннях:

- Через однакову яскравість з фоном об'єкт може не мати чіткої межі, або може бути зашумлений перешкодами, що призводить до неможливості виділення контуру;
- Перекриття об'єктів або їх групування призводить до того, що контур виділяється неправильно і не відповідає межам об'єкта.

Однак, перехід до розгляду тільки контурів об'єктів дозволяє піти від простору зображення – до простору контурів, що істотно знижує складність алгоритмів і обчислень.

Таким чином, контурний аналіз має досить слабку стійкість до перешкод, і будь-який перетин або лише часткова видимість об'єкта призводить або до неможливості детектування, або до помилкових спрацьовувань, але простота і швидкодія контурного аналізу, дозволяють цілком успішно застосовувати даний підхід при чітко вираженому об'єкті на контрастному тлі і відсутності перешкод.

Для чого на зображенні виділяти контур? Припустимо, що необхідно вирішити задачу пошуку на фотографії обличчя людини. Можна спробувати вирішити цю задачу простим перебором, взявши зображення обличчя і

попіксельно порівнюючи його із заданим зображенням. Але, очевидно, що такий алгоритм буде працювати дуже довго та зможе знайти тільки конкретне обличчя, а не будь-яке. Крім цього, якщо зображення трохи повернути або змінити масштаб, то пошук перестане працювати. З іншого боку, маючи контур зображення і контур обличчя можна лінії контуру описати якимось іншим способом, окрім растрової картинки, наприклад, у вигляді списку координат його точок, у вигляді групи ліній, описаних різними математичними формулами. Інакше кажучи, виділив контур, можна його векторизовати і виконувати вже не пошук растра серед растра, а векторного об'єкта серед векторних об'єктів. Це буде набагато швидше, а також опис об'єктів може бути інваріантним до поворотів або масштабування, тобто з’явиться можливість знаходити об'єкти навіть якщо вони повернені або масштабовані.

Для виділення контуру, як правило, спочатку отримують так званий контурний препарат, найчастіше це градієнт (швидкість зміни яскравості). Тобто, отримавши градієнт зображення, буде виділено білим кольором ті області, де існують різкі перепади яскравості, і чорними де яскравість змінюється плавно або взагалі не змінюється. Іншими словами, всі межі будуть виділені білими смугами. Далі ці білі смуги звужуються і отримуємо контур. На даний момент існує ряд стандартних алгоритмів виділення контуру, наприклад, алгоритм Кенні, який реалізований в бібліотеці OpenCV та був використаний для знаходження контурів в розроблюваному мобільному додатку.

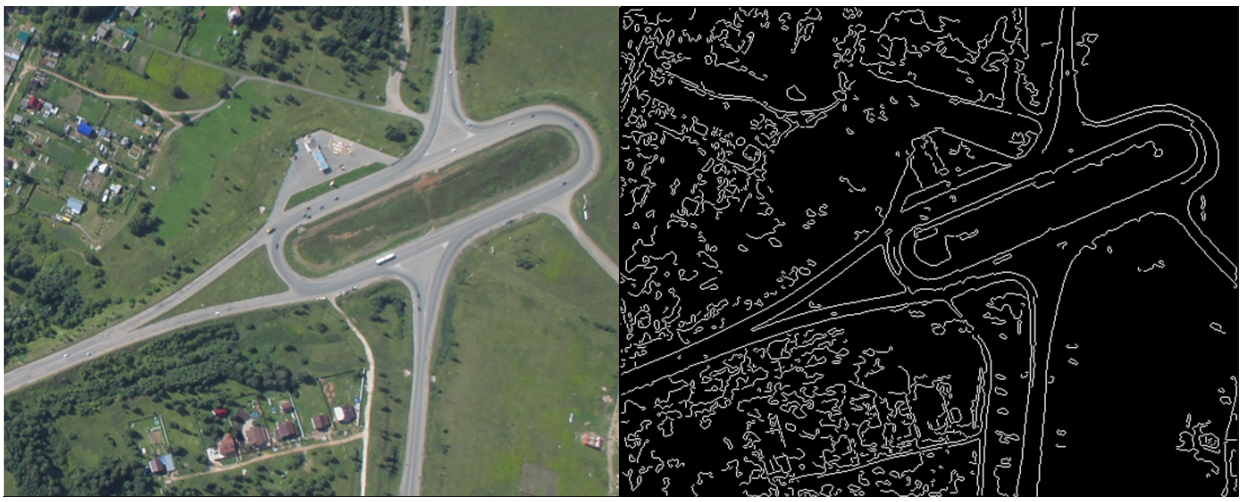

Рисунок 4.1 – Приклад виділення контура

Для реалізації виділення контурів об’єктів на зображенні, в даній роботі була використана бібліотека OpenCV. У цій бібліотеці для пошуку контурів є функція findContours(…), яка має вигляд: findContours(кадр, контури, ієрархія, режим_групування, метод_пакування, зсув), де:

- кадр – належним чином підготовлена для аналізу картинка. Це повинно бути 8-бітове зображення, на яке було застосовано кольоровий фільтр так, щоб об'єкт став абсолютно білим, а фон – чорним. Пошук контурів використовує для роботи монохромне зображення, так що всі пікселі картинки з ненульовим кольором будуть інтерпретуватися як 1, а всі нульові залишаться нулями;

- контури – список для збереження всіх знайдених контурів, представлених у вигляді векторів;

- ієрархія – інформація про топологію контурів;

- режим_групування – один із п’яти режимів угруповання знайдених контурів (RETR_LIST, RETR_EXTERNAL, RETR_CCOMP, RETR_TREE, RETR_FLOODFILL);

- метод_пакування – один із чотирьох запропонованих методів упаковки контурів (CHAIN_APPROX_NONE, CHAIN_APPROX_SIMPLE, CHAIN_APPROX_TC89_L1, CHAIN_APPROX_TC89_KCOS);

- зсув – величина зміщення точок контуру.

Отримані за допомогою функції findContours(…) контури можна відобразити в кадрі. Це допоможе візуально зрозуміти як виглядають знайдені алгоритмом контури. Це можливо за допомогою ще однієї корисної функції – drawContours(кадр, контури, індекс, колір, товщина, тип_лінії, ієрархія, макс_шар, зсув), де:

- кадр – зображення, поверх якого необхідно відобразити контури;
- контури – ті самі контури, знайдені функцією findContours(…);
- індекс – індекс контура, який слід відобразити;
- колір – колір контура;
- товщина – товщина лінії контура;
- тип_лінії – тип з'єднання точок вектора;
- ієрархія – інформація про ієрархію контурів;
- макс_шар – індекс шару, який потрібно відобразити;
- зсув – величина зміщення точок контуру.

Таким чином, в результаті роботи програми можна отримати зображення з виділеними внутрішніми і зовнішніми межами. Маючи готові контури можна приступити до подальшого аналізу зображення.

Розглянемо детальніше реалізацію алгоритму знаходження контурів та нанесення їх на зображення у розроблваному мобільному додатку, написанному мовою програмування Kotlin.

Перед початком роботи із зображеннями, необхідно підключити OpenCV SDK. Для цього потрібно завантажити останню версію SDK з офіційного сайту OpenCV та скопіювати його у проект окремим модулем, після чого підключити цей модуль у файлі build.gradle додатку.

Лістинг 4.1 – Підключення OpenCV SDK

```
implementation project(path: ':opencv')
```

Після цього можна приступати до обробки зображення. Основна робота по знаходженню та відрисовці контурів виконується у методі findContours(originalMat: Mat, checked: Boolean).

Лістинг 4.2 – Метод знаходження контурів

```
      private fun findContours(originalMat: Mat, checked: Boolean) {
              val  finalMat  =  Mat(originalMat.rows(),  originalMat.cols(),
originalMat.type())
              val tempImage = Mat()
              val contours = ArrayList<MatOfPoint>()
              val hierarchy = Mat()
              val color = Scalar(255.0, 0.0, 0.0)

              Imgproc.cvtColor(originalMat, tempImage, Imgproc.COLOR_BGR2GRAY)
              Imgproc.blur(tempImage, tempImage, Size(4.0, 4.0))
              Imgproc.Canny(tempImage, tempImage, 40.0, 60.0)
              Imgproc.findContours(tempImage,      contours,      hierarchy,
Imgproc.RETR_TREE, Imgproc.CHAIN_APPROX_SIMPLE)
              val drawMat = if (checked) originalMat else finalMat
              Imgproc.drawContours(drawMat, contours, -1, color, 4)
              showMatImage(drawMat)
      }
```

У якості параметрів цьому методу передається оригінальне зображення (originalMat), представлене у вигляді матриці, та булеанове значення режиму відображення результату. Щоб отримати зображення у матричному вигляді, необхідно сконвертувати його методом, в залежності від вхідного типу даних. Якщо зображення отримане з камери, то воно подається у вигляді Bitmap, тому для його конвертації у тип Mat (матриця) застосовується метод Utils.bitmapToMat(…), а якщо зображення отримане з галереї, то передається шлях за яким воно зберігається, тому для отримання матриці необхідно завантажити зображення за допомогою метода Imgcodecs.imread(…).

Рисунок 4.2 – Оригінальне зображення

В першу чергу, отримане зображення необхідно перетворити у формат відтінків сірого. Цей крок виконується за допомогою метода Imgproc.cvtColor(…). Перетворення зображення в градацію сірого є дуже важливим, оскільки воно готує зображення до наступних кроків. Цей процес важливий для роботи з пороговими значеннями, що, у свою чергу, необхідно для належної роботи алгоритму визначення контуру.

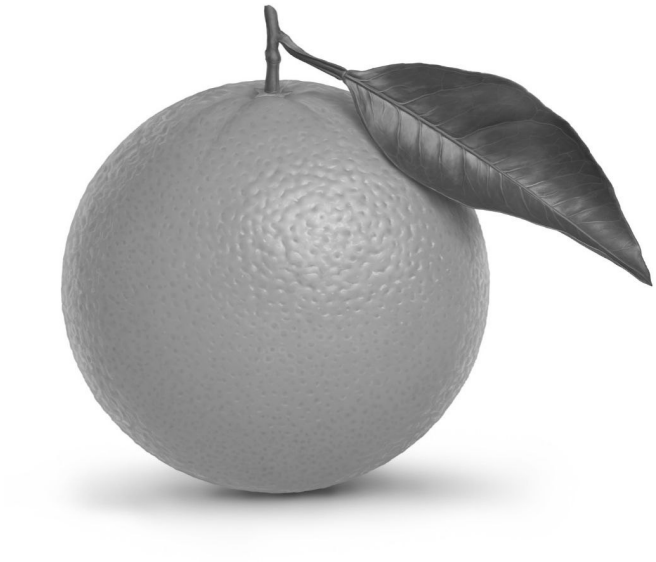

Рисунок 4.3 – Результат перетворення зображення у градацію сірого

Наступним кроком є застосування розмиття зображення для зменшення шумів на зображенні, щоб в результаті отримати більш чисте зображення контурів. Для цього використовується метод Imgproc.blur(…). На рисунку 4.4 можна побачити, що в результаті зображення стало меньш чітким та трохи розмитим, проте межі об’єкта залишаються на місці.

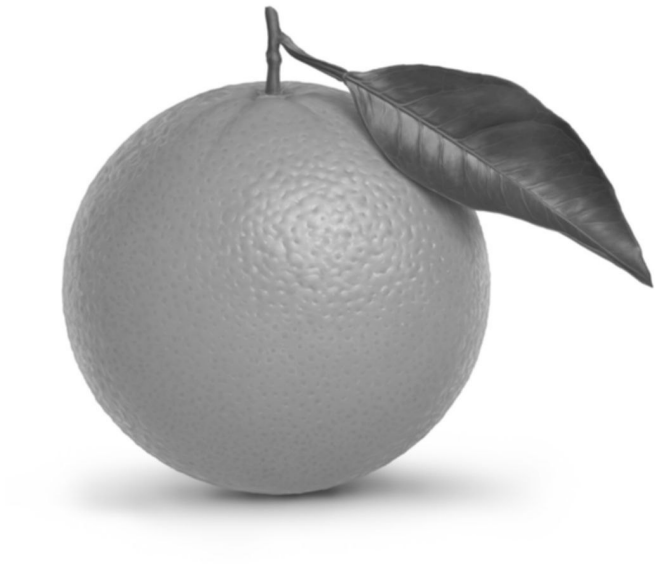

Рисунок 4.4 – Результат розмивання зображення

Після цього, можна вважати, що попередня обробка зображення завершена і можна переходити безпосередньо до знаходження контурів.

До підготовленого зображення застосовуємо алгоритм Канні, за допомогою метода Imgproc.Canny(…), що робить зображення чорно-білим, виділяючи об’єкти, що цікавлять, щоб полегшити роботу алгоритму визначення контуру. Використання порогових значення робить межу об’єкта на зображенні повністю білою, при цьому всі пікселі мають однакову інтенсивність. Тепер алгоритм може виявляти межі об’єктів з цих білих пікселів. Після цього залишається лише знайти та зберігти контури об’єкта, після чого нанести їх на зображення.

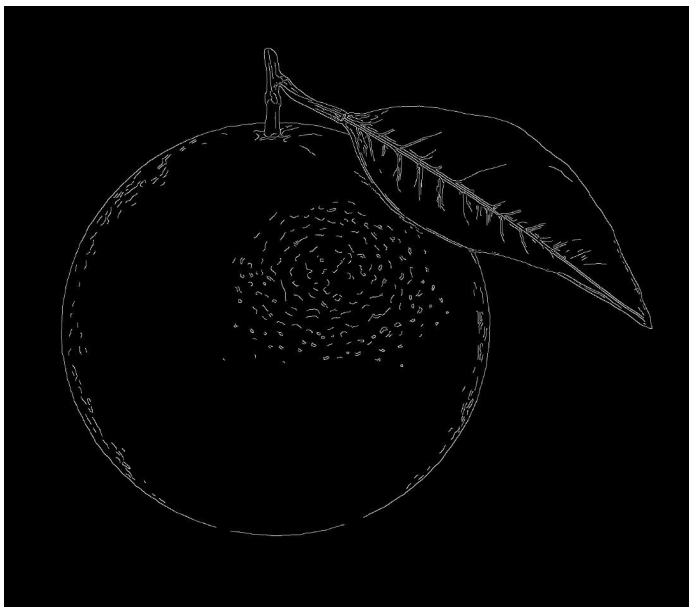

Рисунок 4.5 – Результат застосування алгоритма Канні

Далі передаємо отримане чорно-біле зображення, описанному вище, методу Imgproc.findContours(…), який знаходить та зберігає контури у вигляді списку. Після цього залишається лише нанести контури на результуюче зображення, в залежності від обранного режиму відображення. Якщо параметр checked має значення «true», то відобразяться тільки контури об’єктів на чорному фоні, якщо «false» – контури будуть нанесені поверх оригінального зображення. Нанесення контурів на зображення відбувається за допомогою методу Imgproc.drawContours(…), який також був описанний вище.

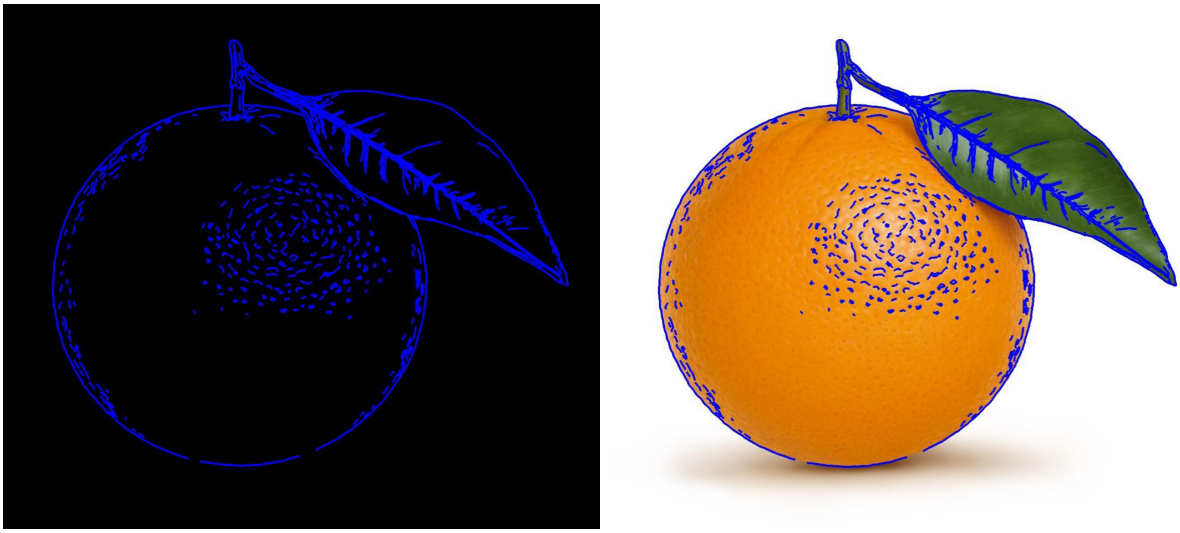

Рисунок 4.6 – Результат знаходження контурів на зображенні

Для відображення результату обробки зображення на екрані викликається метод showMatImage(mat: Mat), якому на вхід передається результуюча матриця зображення з нанесеними контурами.

Лістинг 4.3 – Метод відображення результату на екрані

```
    private fun showMatImage(mat: Mat) {
        Imgproc.cvtColor(mat, mat, Imgproc.COLOR_BGR2RGB)
        val bitmap = Bitmap.createBitmap(mat.width(), mat.height(),
Bitmap.Config.ARGB_8888)
        Utils.matToBitmap(mat, bitmap, true)
        _imageBitmap.value = bitmap
    }
```

Тут спочатку зображення перетворюється з формату BGR до формату RGB для корректного відображення кольору, для цього використовується метод Imgproc.cvtColor(…), так само як і у випадку з перетворенням зображення на градацію сірого.

Далі, за допомогою метода Bitmap.createBitmap(…), створюється чистий bitmap відповідного розміру і, за допомогою метода Utils.matToBitmap(…), результуюча матриця конвертується у створений bitmap.

Отриманий bitmap присвоюється полю _imageBitmap, після чого, завдяки технології зв’язування фреймворка DataBinding, зображення з’являється на екрані.

4.2 Розпізнавання зображених об’єктів на фото

В даній роботі розпізнавання зображених на фото об’єктів стає можливим завдяки використанню бібліотеки TensorFlow Lite, яка розроблена компанією Google спеціально для розгортування моделей машинного навчання на мобільних платформах та інших периферійних пристроях. В рамках даного проекту була підготовлена та навчена модель TensorFlow, яка здатна розпізнавати деякі об’єкти.

Рисунок 4.7 – Класифікація об’єктів за допомогою TensorFlow Lite

Модель TensorFlow Lite представлена у спеціальному ефективному портативному форматі, відомому як FlatBuffers (ідентифікований розширенням файлу .tflite). Це надає кілька переваг перед форматом моделі буфера протоколу TensorFlow, наприклад, зменшений розмір (невеликий розмір коду) і швидший висновок (доступ до даних здійснюється безпосередньо без додаткового аналізу/розпакування), що дозволяє TensorFlow Lite ефективно працювати на пристроях з обмеженими ресурсами обчислень і пам’яті.

Модель TensorFlow Lite може додатково включати метадані, які мають читабельний людиною опис моделі та машиночитані дані для автоматичного створення передобробних та постобробних конвеєрів під час логічного висновку на пристрої.

Модель TensorFlow Lite можна створити такими способами:

- Використати існуючу модель TensorFlow Lite: можна переглянути приклади TensorFlow Lite, щоб вибрати наявну модель. Моделі можуть містити або не містити метадані;

- Створення моделі TensorFlow Lite: Використання TensorFlow Lite Model Maker, щоб створити модель із власним набором даних. За замовчуванням усі моделі містять метадані;

- Перетворення моделі TensorFlow в модель TensorFlow Lite: використання конвертера TensorFlow Lite, щоб перетворити модель TensorFlow в модель TensorFlow Lite. Під час перетворення можна застосовувати такі оптимізації, як квантування, щоб зменшити розмір моделі

та затримку з мінімальною втратою точності або без неї. За замовчуванням усі моделі не містять метаданих.

Після створення моделі можна запускати програму пошуку логічного висновку. Під логічним висновком розуміється процес виконання моделі TensorFlow Lite на пристрої для прогнозування на основі вхідних даних. Висновок можна виконати наступними способами на основі типу моделі:

- Моделі без метаданих: використовується TensorFlow Lite Interpreter API, який підтримується на кількох платформах і мовах, таких як Java (або Kotlin), Swift, C++, Objective-C і Python;

- Моделі з метаданими: розробники можуть використовувати готові API за допомогою бібліотеки завдань TensorFlow Lite або створювати власні конвеєри висновків за допомогою бібліотеки підтримки TensorFlow Lite. На пристроях Android користувачі можуть автоматично створювати обгортки коду за допомогою прив’язки моделі ML в Android Studio або генератора коду TensorFlow Lite. Підтримується лише на Java (Android), поки для Swift (iOS) і C++ на стадії розробки.

На пристроях Android та iOS можна підвищити продуктивність за допомогою апаратного прискорення. На обох платформах можна використовувати GPU Delegate, на Android можна використовувати NNAPI Delegate (для нових пристроїв) або Hexagon Delegate (на старих пристроях), а на iOS можна використовувати Core ML Delegate. Щоб додати підтримку нових апаратних прискорювачів, можна визначити власного делегата.

Модель виявлення об’єктів бере дані зображення в певному форматі, аналізує їх і намагається класифікувати елементи зображення як один із набору відомих класів, які модель навчена розпізнавати. Швидкість, з якою модель може ідентифікувати відомий об’єкт (так званий прогноз або висновок об’єкта), зазвичай вимірюється в мілісекундах. На практиці швидкість висновку залежить від обладнання, на якому розміщена модель, розміру даних, що обробляються, і розміру моделі машинного навчання.

Для початку роботи з бібліотекою TensorFlow Lite необхідно

підключити її, додавши залежності модуля в файлі build.gradle.

Лістинг 4.4 – Підключення бібліотеки TensorFlow Lite

```
implementation 'org.tensorflow:tensorflow-lite:2.4.0'
implementation 'org.tensorflow:tensorflow-lite-support:0.1.0'
implementation 'org.tensorflow:tensorflow-lite-metadata:0.1.0'
```

Після цього можно розпочинати роботу над розпізнаванням зображень. У розроблюваному мобільному додатку необхідно ініціалізувати інтерпретатор моделі машинного навчання TensorFlow Lite з параметрами, перш ніж запускати передбачення з моделлю. Ці параметри ініціалізації залежать від моделі, яку ви використовуєте, і можуть включати такі параметри, як мінімальні пороги точності для передбачень і мітки для ідентифікованих класів об’єктів.

Модель TensorFlow Lite містить файл .tflite, що містить код моделі, і часто містить файл міток, що містить імена класів, передбачених моделлю. У разі виявлення об’єктів класами є такі об’єкти, як людина, собака, кішка або автомобіль. Моделі зберігаються в каталозі src/main/ml основного модуля.

Для зручності та читабельності коду в прикладі оголошується об’єкт-супутник, який містить необхідні статичні константи і визначає параметри моделі.

Лістинг 4.5 – Оголошення параметрів моделі

```
companion object {
    private const val MODEL_PATH = "mobilenet_quant_v1_224.tflite"
    private const val QUANT = true
    private const val LABEL_PATH = "labels.txt"
    private const val INPUT_SIZE = 224
}
```

Використовуючи ці параметри створюється об’єкт TensorFlow Lite Interpreter, який безпосередньо приймає на вхід модель та працює з нею для виконання операції розпізнавання, а також виконується завантаження списку міток зі іменами класів.

Лістинг 4.6 – Початкова ініціалізація

```
    init {
        interpreter = Interpreter(loadModelFile(assetManager, modelPath),
Interpreter.Options())
        labelList = loadLabelList(assetManager, labelPath)
    }
```

Основна робота по розпізнаванню об’єктів виконується у методі recognizeImage(bitmap: Bitmap), якому на вхід передається зображення у форматі bitmap.

Лістинг 4.7 – Метод recognizeImage(bitmap: Bitmap)

```
    fun recognizeImage(bitmap: Bitmap): List<RecognizeResult> {
        val scaledBitmap = Bitmap.createScaledBitmap(bitmap, inputSize,
inputSize, false)
        val byteBuffer = convertBitmapToByteBuffer(scaledBitmap)
        return if (quant) {
            val result = Array(1) {
                ByteArray(labelList.size)
            }
            interpreter?.run(byteBuffer, result)
            getSortedResultByte(result)
        } else {
            val result = Array(1) {
                FloatArray(labelList.size)
            }
            interpreter?.run(byteBuffer, result)
            getSortedResultFloat(result)
        }
    }
```

Спочатку отримане зображення для розпізнавання передається методу Bitmap.createScaledBitmap(…). Цей метод виконує масштабування зображення для зменьшення його розміру.

Після цього зображення піддається конвертації для перетворення bitmap на буфер байтів (ByteBuffer), для цієї задачі було розроблено метод convertBitmapToByteBuffer(…). Дане перетворення необхідне, так як Interpreter бібліотеки TensorFlow Lite працює саме з цим типом даних.

Далі виконується блок, в залежності від параметру quant, що впливає на тип даних результату розпізнавання, масив байтів (ByteArray) або масив чисел з плаваючою точкою (FloatArray).

В кожному з блоків процес розпізнавання майже однаковий. Спочатку

створюється масив для запису результуючих даних, після чого, отриманий на попередньому кроці, буфер байтів та масив для результату передається інтерпритатору для виконання розпізнавання. Запуск інтерпритатора виконується за допомогою метода interpreter.run(…). Interpreter виконує розпізнавання, використовуючи модель TensorFlow Lite та записує результат операції в переданий йому масив.

На останньому кроці виконується перетворення отриманих даних у читабельний вигляд, за допомогою метода getSortedResultByte(…) або getSortedResultFloat(…) виконується сортування по відсотку впевненності розпізнавання та конвертація результуючого масиву у список об’єктів класу RecognizeResult, які завдяки реалізації методу toString() легко перетворюються у текствовий вигляд та можуть бути відображені на екрані у вигляді віджету TextView. Результат розпізнавання з’явиться у верхньому правому куті інтерфейсу розроблюваного мобільного додатку.

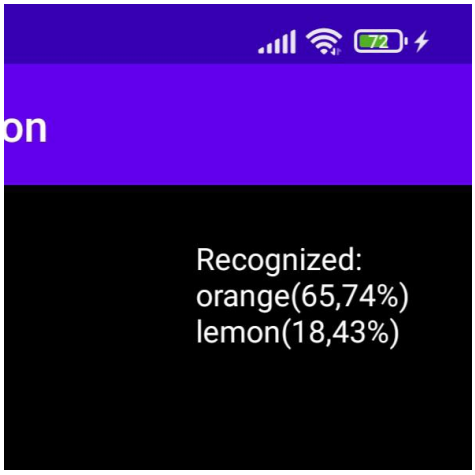

Рисунок 4.8 – Результат розпізнавання

4.3 Інтерфейс мобільного додатку

Інтерфейс розроблюваного мобільного додатку для розпізнавання контурних зображень предаставлений у вигляді двох екранів: CameraFragment та ImageFragment.

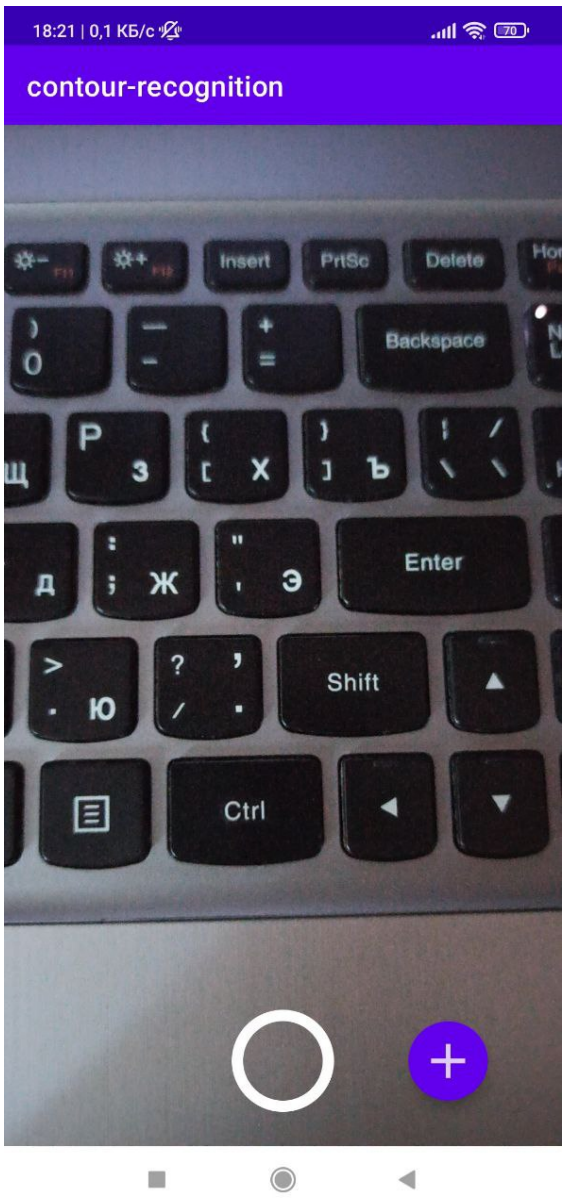

Рисунок 4.9 – Екран CameraFragment

Перший екран (CameraFragment) представлений у вигляді вікна камери з двома кнопками знизу. У цьому вікні можна використовувати всі базові функції камери (зробити знімок, автофокус, зум тощо).

Кнопка з позначкою «+» в нижньому правому кутку дозволяє відкрити галерею зображень для вибору. Обране зображення передається на обробку, описаними вище алгоритмами, у вигляді шляху де воно зберігається.

Інша кнопка, яка розташована в нижній частині посередині, дозволяє зробити новий знімок та відправити його на обробку алгоритмами. Нове зображення передається у форматі bitmap.

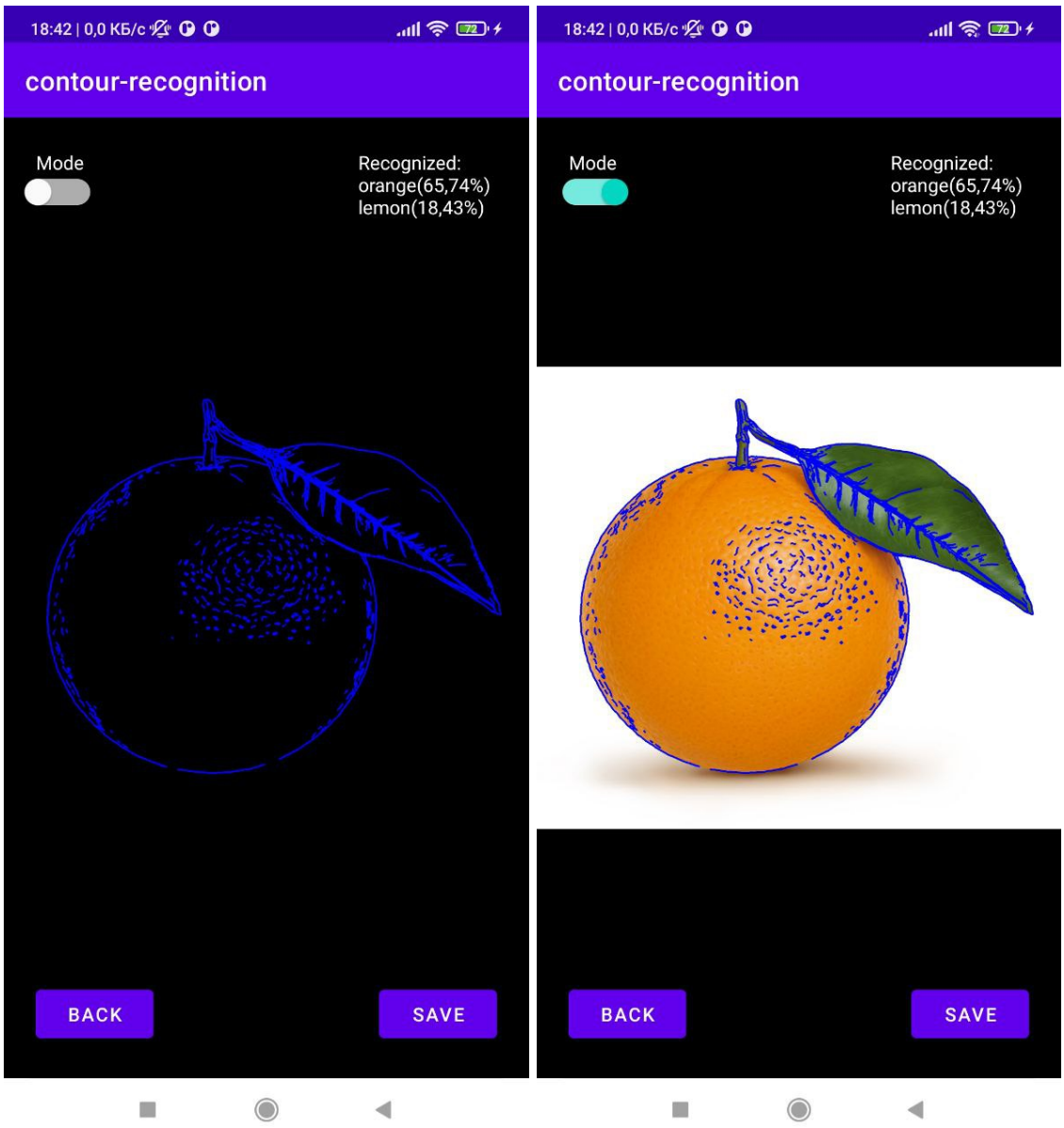

Рисунок 4.10 – Екран ImageFragment у двох режимах відображення

Після вибору та обробки зображення відкривається другий екран для відображення результату – ImageFragment. На цьому екрані можна побачити результат знаходження контурів – початкове зображення з виділеними контурами або тільки контури зображених об’єктів, в залежності від обраного режиму відображення.

У верхньому лівому кутку розташований перемикач, у вигляді віджета Switch, за допомогою якого і відбувається зміна режиму відображення – тільки контури об’єктів або виділення контурів на оригінальному зображенні.

У верхньому правому куті з’являється результат розпізнавання об’єктів за допомогою TensorFlow Lite у вигляді текстової інформації про класифікацію об’єкта та проценту впевненості у правильності розпізнавання. Як можна побачити, програма розпізнавання на 65,74% вневнена, що об'єкт на зображенні – це апельсин і на 18,43% що це лимон. Більший процент показує який із висновків можна вважати вірним.

У нижній частині екрану можна побачити дві кнопки «Back» та «Save». Натиснувши першу кнопку, відбувається повернення до екрану камери, а кнопка «Save» зберігає оброблене зображення до зовнішньої галереї зображень.

Також, варто зазначити, що при першому запуску додатку на екрані з’явиться діалогове вікно із запитом на дозвіл використання камери, для повноцінної роботи додатку необхідно надати позитивну відповідь.

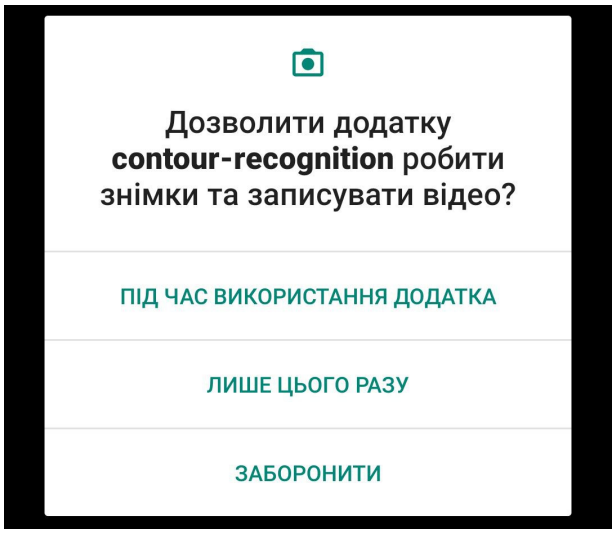

Рисунок 4.11 – Діалогове вікно запиту на дозвіл використання камери

При виборі пункту «Під час використання додатка» дозвіл буде надано і на всі наступні запуски додатку, пункт «Лише цього разу» надає дозвіл тільки до закриття додатку, тому при наступному запуску знову з’явиться запит, а при виборі «Заборонити» дозвіл не буде надано і камера не буде працювати.

ВИСНОВКИ

У процесі виконання даного атестаційного проекту на тему «Спеціалізовані комп’ютерні засоби для розпізнавання контурних зображень» було розроблено мобільний додаток на базі операційної системи Android, який має можливість виділити контур об’єктів на зображенні, виявити їх розташування та розпізнати їх класифікацію.

При реалізації даного проекту були використані сучасні технології, такі як Computer Vision та Machine Learning. Виділення контурів об’єктів на зображенні було втілено за допомогою бібліотеки OpenCV, а за виявлення та розпізнавання об’єктів відповідає бібліотека TensorFlow Lite, яка використовує, спеціально навчену на розпізнавання списку деяких об’ктів, модель. Дана розробка повністю відповідає тематиці проекту, а всі поставлені в роботі задачі було виконано в повному обсязі.

Подальший розвиток розробленого мобільного додатку можна забезпечити за допомогою виконання наступних покращень:

- Додати можливість аналізу кадрів в режимі реального часу;
- Додати можливість аналізу відео;
- Оптимізація та збільшення швидкості роботи алгоритмів;
- Збільшення точності розпізнавання об’єктів;
- Розширення фунціоналу;
- Реалізація додатку для інших мобільних платформ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Computer Vision: what it is and why it matters [Електронний ресурс]. – Режим доступу : www/ URL: https://www.sas.com/ru_ru/insights/analytics/computer-vision.html – Загол. з екрану

2. Computer Vision [Електронний ресурс]. – Режим доступу : www/ URL: https://www.ibm.com/topics/computer-vision – Загол. з екрану

3. Machine Learning [Електронний ресурс]. – Режим доступу : www/ URL: https://www.ibm.com/cloud/learn/machine-learning – Загол. з екрану

4. Машинное обучение [Електронний ресурс]. – Режим доступу : www/ URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение – Загол. з екрану

5. Android's Kotlin-first approach [Електронний ресурс]. – Режим доступу : www/ URL: https://developer.android.com/kotlin/first – Загол. з екрану

6. Meet Android Studio [Електронний ресурс]. – Режим доступу : www/ URL: https://developer.android.com/studio/intro – Загол. з екрану

7. OpenCV [Електронний ресурс]. – Режим доступу : www/ URL: https://opencv.org/about/ – Загол. з екрану

8. Компьютерное зрение (Computer Vision) [Електронний ресурс]. – Режим доступу : www/ URL: https://wiki.programstore.ru/kompyuternoe-zrenie-comuter-vision/ – Загол. з екрану

9. Documentation for app developers [Електронний ресурс]. – Режим доступу : www/ URL: https://developer.android.com/docs – Загол. з екрану

10. Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов [Текст] : пер. с англ. Матвеев Е. – СПб. : Питер, 2013. – 688 с.

11. Шамуратов О. Ю. Алгоритми контурного аналізу зображень [Текст] / О. Ю. Шамуратов, Н. Б. Шаховська. // Журн. Науковий вісник НЛТУ України. – 2019. – №6. – С. 123–127.

12. Богдан В. П. Інтелектуальний аналіз відеоданих. Контурний аналіз – вид комп'ютерної обробки зображень [Текст] / В. П. Богдан. // Журн. Сучасна спеціальна техніка. – 2012. – №1. – С. 61–68.

13. Кривуля Г. Ф. Автономна система моніторингу та контролю технічних об'єктів з використанням безпровідних сенсорних мереж [Текст] / Г. Ф. Кривуля, В. А. Власов. // Вісник Харківського національного технічного університету сільського господарства імені Петра Василенка. – 2017. – №187. – С. 64–67.

14. Методи та засоби ідентифікації та класифікації об’єктів за характерними точками їх контурів: монографія [Текст] / Д. І. Загородня, П. Є. Биковий, Х. В. Ліп’яніна-Гончаренко та ін. – Тернопіль: ЗУНУ, 2020. – 165 с.

15. Теорія розпізнавання образів [Електронний ресурс] – Режим доступу : www/URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/16460/1/Теорія%20розпізнавання%20образів.%20Лекції.pdf– Загол. з екрану

16. Основні поняття розпізнавання образів [Електронний ресурс] – Режим доступу : www/URL: http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_01.pdf – Загол. з екрану

17. Копча-Горячкіна Г. Е. ТЕОРІЯ РОЗПІЗНАВАННЯ ОБРАЗІВ. Частина І: Навчально-методичний посібник для студентів факультету інформаційних технологій напрямів „Комп’ютерні науки” та „Програмна інженерія” [Текст] / Галина Ернестівна Копча-Горячкіна. – Ужгород: ДВНЗ «Ужгородського національного університету», 2016. – 59 с.

18. Gaudenz Boesch TensorFlow Lite – Real-Time Computer Vision on Edge Devices [Електронний ресурс] / Gaudenz Boesch. – 2022. – Режим доступу : www/URL: https://viso.ai/edge-ai/tensorflow-lite/ – Загол. з екрану

19. TensorFlow Lite [Електронний ресурс]. – 2022. – Режим доступу : www/URL: https://www.tensorflow.org/lite/guide – Загол. з екрану

20. Quickstart for Android | TensorFlow Lite [Електронний ресурс]. – 2022. – Режим доступу : www/URL: https://www.tensorflow.org/lite/android/quickstart – Загол. з екрану