

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО
ЗАСТОСУНКУ ДЛЯ ІДЕНТИФІКАЦІЇ ПОРІД ТВАРИН. RESEARCH AND
SOFTWARE IMPLEMENTATION OF A MOBILE APPLICATION FOR ANIMAL
BREED IDENTIFICATION.

Виконав: студент(ка) 2 курсу, групи КНТ-124м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ВИНОГРАДОВ М.М.

(ПРИЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

ВИНОГРАДОВА Миколи Миколайовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проекту (роботи) Дослідження та програмна реалізація мобільного застосунку для ідентифікації порід тварин. Research and Software Implementation of a Mobile Application for Animal Breed Identification
 керівник проекту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович,
(ПРИЗВИЩЕ, ім'я, по батькові, науковий ступінь, вчене звання)
 затверджені наказом закладу вищої освіти від “30” вересня 2025 року № 447
2. Строк подання студентом проекту (роботи) 04 листопада 2025 року
3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження мобільного застосунку.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викл.		

7. Дата видачі завдання “01” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4–5 тижні	Розділ 2
4	Розробка програми.	6–8 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	9–10 тижні	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	11 тиждень	Додатки
7	Нормоконтроль та рецензування	12 тиждень	
8	Захист роботи.	12 тиждень	

Студент (ка)



(підпис)

Микола ВІНОГРАДОВ

(Ім'я ПРІЗВИЩЕ)

Керівник проекту (роботи)

Підпис

(підпис)

Андрій ОЛІЙНИК

(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 80 с., 6 табл., 43 рис., 3 дод., 26 джерел.

МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, МОБІЛЬНІ ДОДАТКИ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ ПОРОДИ.

Об'єкт дослідження – процес розробки програмного забезпечення для розпізнавання порід тварин.

Предмет дослідження – методи та програмні засоби для розпізнавання зображень.

Мета роботи – розробка мобільного застосунку, що забезпечує розпізнавання порід тварин.

Матеріали, методи та технічні засоби: мови програмування Java та Python, інтегровані середовища розробки Android Studio та PyCharm Community Edition, персональний комп'ютер з процесором не менше 2.2 ГГц під управлінням операційної системи Windows 10 (64 bit).

Результати. Створено мобільний застосунок, що використовує різні моделі нейронних мереж для розпізнавання порід тварин за зображенням.

Висновки. Розроблений застосунок значно спрощує процес ідентифікації порід котів і собак, та може бути корисним для власників, ветеринарів і організацій, що займаються тваринами, зокрема притулки та зоомагазини.

Галузь використання – ветеринарна медицина, зоомагазини, кінологічні організації, притулки для тварин.

ABSTRACT

Explanatory note to the final qualifying work of the master: 80 pages, 6 tables, 43 figures, 3 appendixes, 26 sources.

MACHINE LEARNING, NEURAL NETWORK, MOBILE APPLICATIONS, IMAGE CLASSIFICATION, BREEDING RECOGNITION.

The object of research is the process of developing software for recognizing animal breeds.

The subject of research is methods and software for image recognition.

The purpose of the work is to develop a mobile application that provides animal breed recognition.

Materials, methods and technical means: Java and Python programming languages, Android Studio and PyCharm Community Edition integrated development environments, a personal computer with a processor of at least 2.2 GHz running the Windows 10 (64 bit) operating system.

Results. A mobile application has been created that uses various neural network models to recognize animal breeds by image.

Conclusions. The developed application greatly simplifies the process of identifying cat and dog breeds, and can be useful for owners, veterinarians and organizations dealing with animals, in particular shelters and pet stores.

Field of use - veterinary medicine, pet stores, dog organizations, animal shelters.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Короткий опис досліджуваної предметної області.....	10
1.2 Огляд існуючих рішень проблеми.....	10
1.2.1 Google Lens.....	10
1.2.2 Petfinder.....	11
1.2.3 Pawscout.....	12
1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів.....	13
1.3 Висновки за розділом 1.....	15
2 Матеріали і методи.....	17
2.1 Вибір мови програмування.....	17
2.2 Вибір середовища розробки для створення програмного забезпечення... 19	
2.2.1 Середовище розробки для Python.....	19
2.2.2 Середовище розробки для Android і Java.....	21
2.3 Набір даних.....	23
2.4 Вибір бібліотеки для машинного навчання.....	24
2.5 Висновки за розділом 2.....	27
3 Опис програми.....	29
3.1 Структура мобільного застосунку для ідентифікації порід тварин.....	29
3.2 Функціонування мобільного застосунку для ідентифікації порід тварин	30
3.3 Проектування інтерфейсу мобільного застосунку для ідентифікації порід тварин.....	34
3.4 Архітектура створених моделей нейронних мереж.....	36
3.4.1 MobileNetV2.....	37
3.4.2 Convolutional neural network.....	38
3.4.3 ResNet50.....	40

3.4.4 EfficientNetB0.....	41
3.4.5 DenseNet121.....	42
3.5 Висновки за розділом 3.....	43
4. Експлуатація, тестування та експериментальне дослідження мобільного застосунку.....	44
4.1 Призначення й умови застосування мобільного застосунку.....	44
4.2 Характеристики мобільного застосунку для ідентифікації порід тварин.....	45
4.3 Інструкція по експлуатації програми.....	46
4.3.1 Звернення до програми.....	46
4.3.2 Вхідні й вихідні дані.....	46
4.3.3 Повідомлення.....	46
4.4 Виконання мобільного застосунку для ідентифікації порід тварин.....	47
4.5 Порівняння ефективності моделей нейронних мереж.....	53
4.6 Тестування програмного забезпечення.....	59
4.7 Висновки за розділом 4.....	59
Висновки.....	60
Перелік джерел посилання.....	62
Додаток А Технічне завдання.....	64
Додаток Б Фрагмент тексту програми.....	68
Додаток В Слайди презентації.....	74

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
APK	– Android Package;
CNN	– Convolutional Neural Network;
GPU	– Graphics Processing Unit;
IDE	– Integrated Development Environment;
NDK	– Native Development Kit;
SDK	– Software Development Kit;
UI	– User Interface;
URL	– Uniform Resource Locator;
HM	– нейронна мережа.

ВСТУП

Сучасні досягнення в галузі штучного інтелекту та машинного навчання все активніше впроваджуються у сферу мобільних технологій, відкриваючи користувачам смартфонів нові функціональні можливості. Одним із важливих напрямів використання є аналіз зображень, зокрема визначення порід домашніх тварин за допомогою камери мобільного пристрою. Завдяки розвитку методів глибинного навчання та широкій доступності мобільних платформ стало можливим створення застосунків, що дозволяють з високою точністю розпізнавати породи котів і собак безпосередньо зі знімків, зроблених телефоном [1].

Зростання кількості власників домашніх улюбленців та потреба у швидкому й надійному розпізнаванні порід стимулюють появу мобільних застосунків, які автоматизують цей процес. Подібні рішення можуть бути корисними у ветеринарних клініках, притулках для тварин, зоомагазинах, а також стати зручним інструментом для власників тварин. Крім того, такі застосунки сприяють формуванню відповідального ставлення до тварин і можуть допомогти у виявленні безпритульних котів і собак [2].

Водночас наявні мобільні рішення не завжди забезпечують стабільне та точне розпізнавання порід у реальних умовах використання. Вони можуть давати похибки при недостатньому освітленні, нестандартних кутах зйомки або наявності зайвих об'єктів на зображенні. До того ж чимало застосунків мають обмежений каталог порід, не підтримують офлайн-режим або потребують доступу до серверів із високою обчислювальною потужністю. Ще однією суттєвою проблемою є необхідність оптимізації алгоритмів машинного навчання для ефективної роботи на мобільних пристроях із невеликими ресурсами. Саме тому обрана тема дослідження є надзвичайно актуальною [3].

Мета роботи – розробка мобільного застосунку, що дозволяє користувачам автоматично визначати порід котів і собак за фотографією.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Короткий опис досліджуваної предметної області

Мобільний застосунок для визначення порід домашніх тварин (котів і собак) є програмним продуктом, що застосовує методи комп'ютерного зору та алгоритми машинного навчання для розпізнавання породи за зображенням, отриманим за допомогою камери смартфона [4].

Подібні рішення стають все більш популярними завдяки простоті використання, поширеності мобільних пристроїв та важливості завдання ідентифікації тварин у повсякденному житті. Вони можуть використовуватися власниками тварин, ветеринарними спеціалістами, працівниками притулків і зоомагазинів, а також слугувати інструментом для розпізнавання безпритульних тварин і організації їхнього обліку [5].

Програмне забезпечення такого характеру зазвичай має такі функції:

- завантаження зображення для розпізнавання;
- розпізнавання (класифікація) породи домашнього улюбленця;
- перегляд списку порід, які можуть бути розпізнані.

1.2 Огляд існуючих рішень проблеми

Розглянемо існуючі аналоги, які є найбільш поширеними та використовуються на даний момент для розпізнавання порід тварин, таведемо їх переваги та недоліки за вищезазначеними критеріями.

1.2.1 Google Lens

Google Lens – це Додаток Google, який використовує машинне навчання та штучний інтелект для розпізнавання об'єктів, включаючи породи тварин. Можна просто сфотографувати тварину, і Google Lens спробує визначити її породу (рис 1.1) [6].

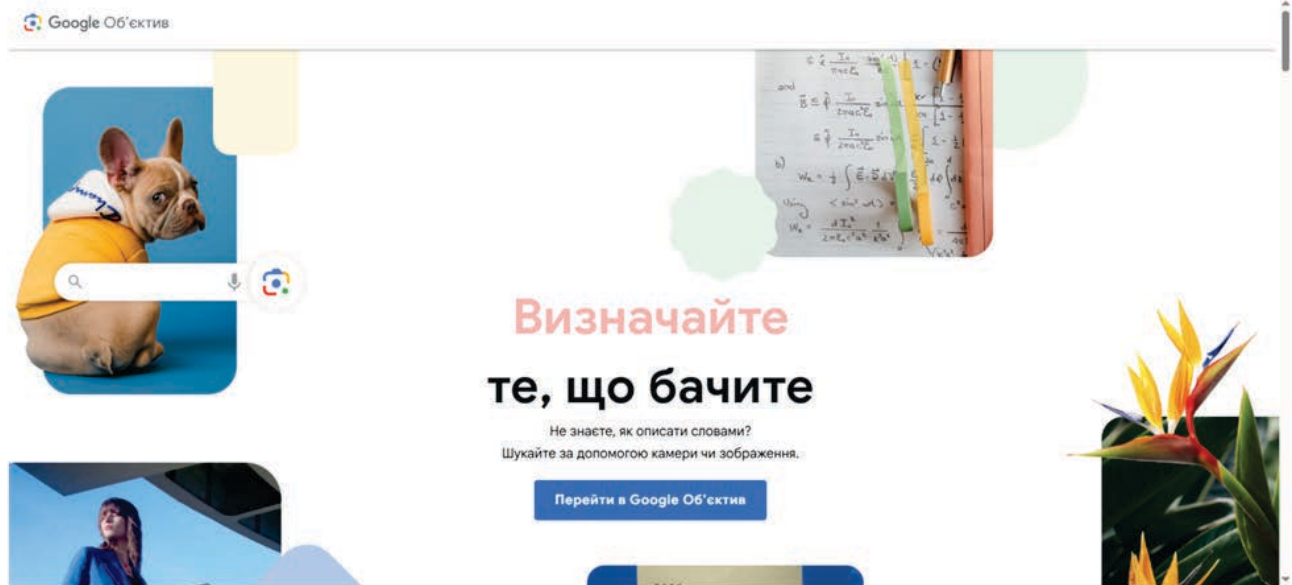


Рисунок 1.1 – Google Lens [6]

Основні переваги:

- визначає породу котів і собак;
- можливість голосового вводу;
- велика база даних – використовує величезні обсяги інформації від Google, що підвищує точність розпізнавання.

1.2.2 Petfinder

Petfinder – це програма для пошуку домашніх тварин, яка також включає функцію розпізнавання породи на основі фотографії. Вона використовує технології машинного навчання для ідентифікації порід собак та котів (рис. 1.2) [7].

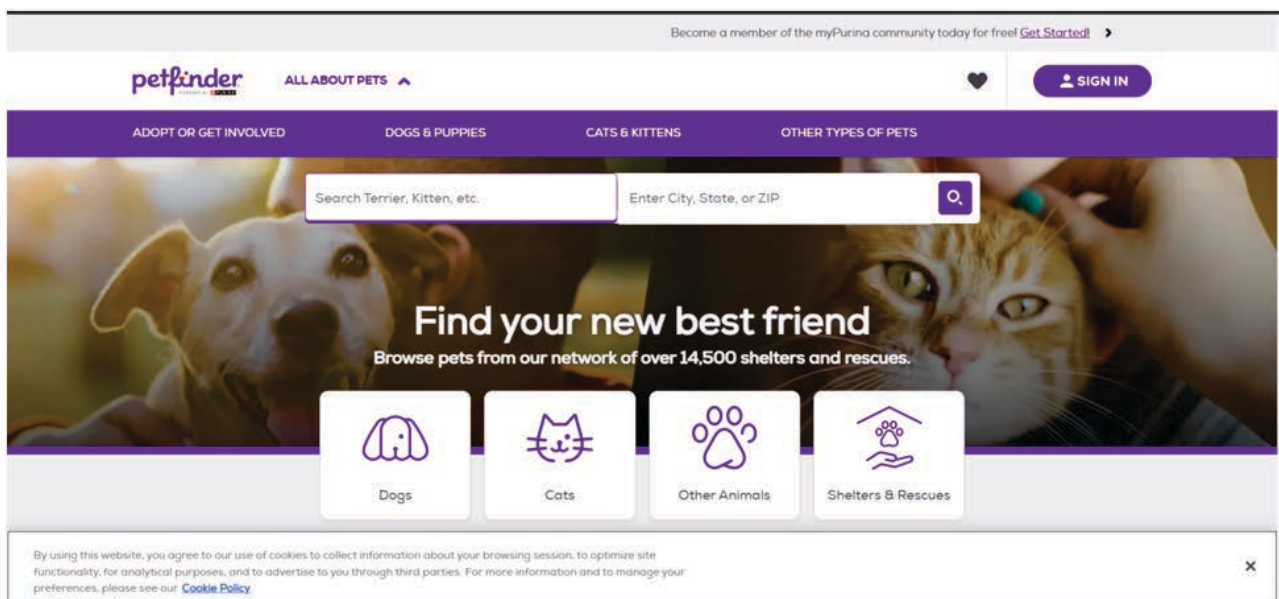


Рисунок 1.2 – Petfinder [7]

Основні переваги:

- спеціалізація на домашніх тваринах;
- база даних з притулків;
- додаткова інформація про породу.

1.2.3 Pawscout

Pawscout – це додаток з функцією розпізнавання порід собак і кішок, а також з можливістю пошуку втрачених вихованців (рис 1.3) [8].

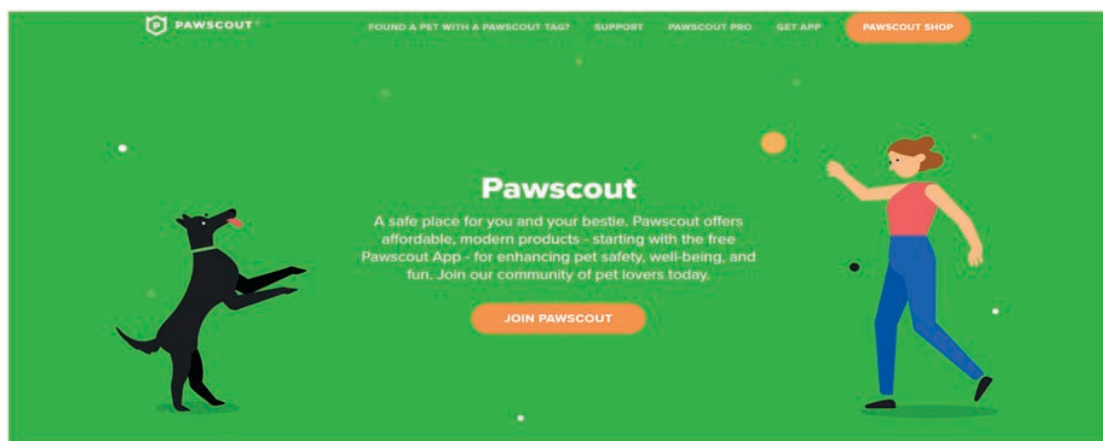


Рисунок 1.3 – Pawscout [8]

Основні переваги:

- спільнота власників тварин;
- цифровий профіль тварини;
- трекінг місцеперебування.

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день програми та додатки для розпізнавання породи домашнього улюбленця по зображенню.

Але крім основних переваг, усі вони мають і недоліки. Наприклад:

- неточність розпізнавання – жоден додаток не гарантує 100% точності визначення породи, особливо у випадку змішаних порід;
- обмежена база даних – деякі додатки можуть не містити інформації про рідкісні або нові породи;
- потреба в інтернет-з'єднанні – більшість таких програм використовують хмарні технології, що унеможлиблює їх роботу в офлайн-режимі.

Також дійсно існують переваги, характерні для всіх програм, а саме можливість швидкого доступу до інформації про породу, її особливості та догляд, а також інтеграція з іншими сервісами, такими як пошук тварин для усиновлення чи безпекові функції.

У ході виконання дипломної роботи буде реалізовано програмний продукт, що буде мати наступні переваги, на відміну від існуючих аналогів:

- безкоштовність та доступність;
- можливість часткової роботи в оффлайн режимі;
- спеціалізація на ідентифікації породи;
- перегляд списку порід, які можуть бути розпізнані.

В таблиці 1.1 наведено порівняння вже існуючих програмних продуктів та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.1 – Порівняння програмних продуктів

Характеристика	Google Lens	Petfinder	Pawscout
Доступність	+	+	-
Спеціалізація на ідентифікації	+	-	-
Необхідність інтернет-з'єднання	-	-	-
Точність ідентифікації	+	-	-
Перегляд списку порід, які можуть бути розпізнані	-	+	-

Як можна побачити, серед аналогів найкращим є “Google Lens”, оскільки він більш доступний, спеціалізується на ідентифікації та має дуже велику точність розпізнавання порід тварин, в той час як аналоги “Petfinder”, та “Pawscout” не підходять для виконання поставленого завдання.

При розробці мобільного застосунку для ідентифікації порід тварин необхідно забезпечити такі функціональні вимоги:

- підтримка можливості завантаження зображення тварини з камери або галереї;
- можливість попередньої обробки зображення для підготовки до класифікації;
- можливість передачі зображення на сервер для обробки моделлю нейронної мережі;
- підтримка класифікації породи тварини за допомогою алгоритмів машинного навчання;
- підтримка повернення результатів класифікації з рівнем достовірності;

- можливість відображення користувачу опису та характеристик визначеної породи;
- підтримка перегляду списку доступних порід;
- можливість завантаження зображення через URL-посилання;
- можливість змінити мову програму на українську та англійську.

1.3 Висновки за розділом 1

Як показало дослідження предметної області, на сьогоднішній день задача розпізнавання породи тварин є цікавою і корисною й в ній потребують як власники тварин так і професіонали. Тому розробляється безліч та вже розроблено мобільних застосунків. Найбільш відомими з яких є: Google Lens, Petfinder, Pawscout. Однак, у ході аналізу було виявлено, що існуючі аналоги хоча і мають певні переваги у своїй сфері використання, але не здатні поєднати весь необхідний функціонал для забезпечення повної роботи для ідентифікації породи домашнього улюбленця.

Тому, тема роботи є актуальною, оскільки вона спрямована на вирішення практичної задачі, яка має попит серед широкого кола користувачів. Автоматизація процесу ідентифікації породи домашніх тварин дає змогу зменшити час на пошук інформації, підвищує зручність користування сучасними технологіями та може бути корисною як для власників тварин, так і для ветеринарів, заводчиків і спеціалістів у сфері догляду за тваринами. Крім того, розвиток мобільних застосунків із використанням технологій глибокого навчання відповідає сучасним тенденціям у сфері штучного інтелекту та цифровізації побутових сервісів.

Метою роботи є розробка мобільного застосунку, що дозволяє користувачам автоматично визначати порід котів і собак за фотографією

Завдання роботи:

- виконати аналіз вимог до розроблюваного програмного забезпечення;

- обрати та обґрунтувати інструментарій розробки;
- розробити архітектуру системи;
- виконати програмну реалізацію
- виконати тестування.

Сформульовано функціональні вимоги до мобільного застосунку для ідентифікації порід тварин.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

У сучасній Android-розробці використовується кілька мов програмування, кожна з яких має свої особливості та сфери застосування. Вибір мови залежить від вимог до продуктивності, зручності підтримки коду та сумісності з наявними бібліотеками. Найпоширенішими серед них є Java, Kotlin, C++ та Python [9].

Java. Це одна з головних мов для створення Android-додатків, яка багато років була основним стандартом платформи. Java є статично типізованою мовою з великою кількістю бібліотек і добре розвиненою екосистемою, а також має офіційну підтримку від Google. Її стабільність та зрозумілий синтаксис роблять Java зручним вибором як для початківців, так і для досвідчених розробників. Крім того, вона широко використовується в корпоративних застосунках, що забезпечує велику кількість готових рішень і прикладів [10].

Kotlin. Мова, розроблена компанією JetBrains та офіційно рекомендована для Android-розробки. Вона поєднує переваги статичної типізації з сучасним і лаконічним синтаксисом, що дозволяє писати зрозуміліший та компактніший код. Kotlin сумісний із Java-бібліотеками, тому активно застосовується як у нових проєктах, так і в модернізації старих. Крім того, мова має зручні інструменти для роботи з асинхронними процесами. Завдяки активній підтримці спільноти Kotlin швидко розвивається та отримує регулярні оновлення [11].

C++. Використовується у випадках, коли необхідна висока продуктивність, наприклад у мобільних іграх або програмах для обробки мультимедійних даних. За допомогою Android NDK можна інтегрувати низькорівневі компоненти на C++ у звичайні Android-додатки. Це дає змогу отримати кращу швидкодію, але водночас ускладнює процес розробки. C++ забезпечує прямий доступ до апаратних ресурсів пристрою, що робить його незамінним у критично важливих обчисленнях [12].

Python. Хоча Python рідко застосовується для прямої розробки Android-додатків, він корисний для створення серверної частини, роботи з базами даних та алгоритмами машинного навчання. Завдяки своїй простоті й великій кількості бібліотек Python часто використовується як допоміжний інструмент під час розробки мобільних рішень. Існують також кросплатформні рішення, що дозволяють створювати мобільні додатки на Python, проте вони менш продуктивні порівняно з Java чи Kotlin. Python особливо цінується за швидку розробку прототипів та інтеграцію з сучасними технологіями штучного інтелекту [13].

Порівняльна характеристика мов програмування наведена у таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Характеристика	Java	Kotlin	C++	Python
Підтримка Android SDK	+	+	Часткова	Часткова
Офіційна підтримка Google	+	+	-	-
Продуктивність	+	+	+	-
Простота вивчення	+	+	-	+
Багатопоточність	+	+	+	+
Типізація	Статична	Статична	Статична	Динамічна

На основі порівняння мов програмування було прийнято рішення обрати Java як оптимальний варіант для розробки Android-додатків. Вона поєднує стабільність і сумісність із різними версіями операційної системи Android, має велику екосистему бібліотек та компонентів, що дозволяє швидко реалізовувати різноманітні функції застосунку. Крім того, Java є офіційною мовою для Android, має добре документовану підтримку та активну спільноту розробників, що спрощує вирішення технічних проблем і підвищує ефективність процесу розробки. Ці фактори роблять Java найкращим вибором для створення надійної, масштабованої та підтримуваної клієнтської частини мобільного застосунку.

Для реалізації серверної частини було обрано Python, оскільки ця мова відзначається простотою синтаксису та широкими можливостями для роботи з даними і API. Python має великий набір бібліотек для обробки зображень, роботи з мережею та інтеграції з базами даних, що дозволяє ефективно реалізовувати функціонал серверної частини. Особливо важливим для цього проєкту є використання можливостей Python для машинного навчання, включаючи побудову та навчання моделей нейронних мереж для ідентифікації порід тварин. Такий вибір забезпечує високу продуктивність, гнучкість і спрощує масштабування системи при зростанні навантаження.

Завдяки комбінації Java для клієнтської частини та Python для серверної реалізації досягається оптимальний баланс між зручністю розробки, продуктивністю та масштабованістю застосунку. Це дозволяє забезпечити надійну взаємодію між користувачем і системою, ефективну обробку даних і швидку реалізацію алгоритмів машинного навчання, що є ключовим для досягнення мети проєкту.

2.2 Вибір середовища розробки для створення програмного забезпечення

2.2.1 Середовище розробки для Python

Існує значна кількість інтегрованих середовищ розробки (IDE), що забезпечують програмістів необхідними інструментами для роботи з мовою Python. Серед найпоширеніших можна виділити PyCharm, Visual Studio Code та IDLE [14].

- Visual Studio Code (VS Code). Легке та високопродуктивне середовище розробки, яке підтримує Python за допомогою спеціальних розширень. Відзначається швидкістю роботи, гнучкістю та широкими можливостями налаштування [15];

- PyCharm. Потужне IDE, спеціально орієнтоване на мову Python. Воно пропонує широкий набір можливостей: автодоповнення коду,

підсвічування синтаксису, вбудовані інструменти для тестування, а також інтеграцію з системами контролю версій, зокрема Git [16];

– IDLE. Просте середовище, що входить до стандартного пакета Python. Завдяки своїй простоті воно підходить для початківців, проте за функціональністю поступається іншим сучасним IDE [17].

Порівняння характеристик середовищ розробки наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння середовищ розробки

Характеристика	PyCharm	VS Code	IDLE
Автодоповнення коду	Наявність ІІІ асистента	В залежності від розширень	Мінімальне
Дебагінг	Так	Так	Мінімальний
Інструменти для аналізу коду	Вбудовані	Через розширення	Відсутні
Швидкість роботи	Середня	Висока	Висока
Підтримка Git	Вбудовано	Через розширення	Ні

Після детального аналізу популярних середовищ розробки для Python було обрано IDE PyCharm, оскільки воно пропонує широкий набір потужних інструментів, що значно спрощують процес розробки складних програмних систем. Серед ключових переваг PyCharm варто відзначити вбудовані засоби для тестування та налагодження коду, автоматичне визначення помилок і пропозиції щодо їх виправлення, а також інтеграцію з системами контролю версій, що забезпечує ефективне керування проєктом і командну роботу над кодом. Крім того, IDE підтримує інтеграцію з популярними бібліотеками та фреймворками для машинного навчання, включаючи TensorFlow і PyTorch, що робить його оптимальним вибором для реалізації серверної частини мобільного застосунку.

Хоча інші середовища розробки, такі як VS Code, також є зручними та мають гнучкі можливості налаштування, PyCharm відрізняється більш

повноцінною підтримкою Python і додатковими інструментами для професійної розробки. Це дозволяє значно скоротити час на написання та тестування коду, підвищує ефективність розробників і забезпечує високу якість програмного забезпечення.

Таким чином, вибір PyCharm як основного середовища розробки Python-коду забезпечує оптимальний баланс між продуктивністю, зручністю використання та функціональністю, що особливо важливо для проєктів, які передбачають роботу з алгоритмами машинного навчання та обробку великих обсягів даних.

2.2.2 Середовище розробки для Android і Java

Для створення Android-додатків застосовуються спеціалізовані інтегровані середовища розробки (IDE), які надають програмісту всі необхідні інструменти – від написання коду та налагодження до тестування і формування APK-файлів. Використання IDE значно підвищує ефективність роботи, адже вони поєднують у собі редактор коду, емулятор пристроїв, інструменти для відстеження помилок і керування залежностями. Серед найпопулярніших середовищ для Android-розробки можна виділити Android Studio, Visual Studio Code та IntelliJ IDEA [18].

- Android Studio – офіційне середовище розробки від Google, орієнтоване на Android. Підтримує мови Java, Kotlin та C++, містить вбудований емулятор, редактор користувацького інтерфейсу, засоби для налагодження й профілювання, а також забезпечує повну інтеграцію з Android SDK та Gradle [19];

- Visual Studio Code (VS Code) – легковаговий редактор коду, який розширюється за допомогою плагінів. Для Android-розробки найчастіше використовується у поєднанні з фреймворком Flutter та мовою Dart. Завдяки розширенням підтримує налагодження, роботу з емуляторами та автоматичне збирання проєктів [20];

– IntelliJ IDEA – повнофункціональне середовище розробки від компанії JetBrains, на основі якого створено Android Studio. Підтримує мови Java та Kotlin, має розвинені засоби для автодоповнення коду, рефакторингу, роботи з Git, а також інструменти для проєктування інтерфейсу (у версії Ultimate) [21].

Порівняння характеристик середовищ розробки наведено у таблиці 2.3.

Таблиця 2.3 – Порівняння середовищ розробки

Характеристика	Android Studio	VS Code	IntelliJ IDEA
Емулятор Android	Вбудований	Через розширення	Через плагіни
Автодоповнення коду	+	+	+
Дебагінг	Вбудований	Через розширення	Вбудований
Інтеграція з Git	Вбудована	Через розширення	Вбудована
Продуктивність	Середня	Висока	Середня
UI-дизайнер	Вбудований	Відсутній	Лише в Ultimate

Для розробки мобільного застосунку під Android було обрано Android Studio як основне середовище розробки. Це офіційне IDE від компанії Google, яке забезпечує повну інтеграцію з Android SDK, що дозволяє розробникам використовувати всі можливості платформи та нові функції операційної системи без додаткових налаштувань. Android Studio має потужний UI-редактор, що дозволяє швидко створювати інтуїтивно зрозумілі інтерфейси користувача з використанням drag-and-drop елементів, а також підтримує адаптивний дизайн для різних розмірів екрану та типів пристроїв.

Серед інших важливих можливостей IDE варто виділити вбудований емулятор, що дозволяє тестувати застосунки на різних версіях Android без фізичних пристроїв, а також розширені інструменти для налагодження та

профілювання продуктивності, що спрощують пошук помилок і оптимізацію коду. Android Studio також забезпечує підтримку автоматизованого збирання проєктів, інтеграцію з системами контролю версій і плагінами для тестування, що дозволяє організувати повний цикл розробки без потреби у використанні сторонніх інструментів.

Таким чином, використання Android Studio як основного середовища розробки забезпечує високу ефективність створення нативних Android-застосунків, дозволяє реалізувати складні інтерфейси користувача, підвищує якість коду та скорочує час розробки. Завдяки поєднанню багатофункціональних інструментів, підтримки новітніх технологій Android і можливостей для тестування, Android Studio є оптимальним вибором для професійної мобільної розробки.

2.3 Набір даних

Для навчання моделей нейронних мереж використовувався набір даних “oxford_iiit_pet”. Це набір даних про домашніх тварин являє собою дані зображень домашніх тварин з 37 категорій, що містить приблизно 200 зображень для кожного класу. Зображення мають великі відмінності в масштабі, позі та освітленні. Всі зображення мають відповідну інструкцію про породу та вид. Крім того, для навчального поділу передбачені рамки голови, що обмежують, що дозволяє використовувати цей набір даних для простих завдань виявлення об'єктів. У тестовому поділі рамки, що обмежують, порожні (рис 2.1) [22].

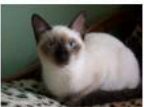
file_name	head_bbox	image	label	segmentation_mask	species
Sphynx_158.jpg	array([[0., 0., 0., 0.]], dtype=float32)		33 (Sphynx)		0 (Cat)
english_cocker_spaniel_135.jpg	array([[0., 0., 0., 0.]], dtype=float32)		12 (english_cocker_spaniel)		1 (Dog)
British_Shorthair_181.jpg	array([[0., 0., 0., 0.]], dtype=float32)		9 (British_Shorthair)		0 (Cat)
Siamese_161.jpg	array([[0., 0., 0., 0.]], dtype=float32)		32 (Siamese)		0 (Cat)
Sphynx_178.jpg	array([[0., 0., 0., 0.]], dtype=float32)		33 (Sphynx)		0 (Cat)
american_pit_bull_terrier_139.jpg	array([[0., 0., 0., 0.]], dtype=float32)		2 (american_pit_bull_terrier)		1 (Dog)
newfoundland_122.jpg	array([[0., 0., 0., 0.]], dtype=float32)		22 (newfoundland)		1 (Dog)

Рисунок 2.1 – Фрагмент набору даних

2.4 Вибір бібліотеки для машинного навчання

Python пропонує широкий вибір бібліотек для реалізації алгоритмів машинного навчання, які охоплюють як класичні статистичні методи, так і сучасні підходи на базі нейронних мереж. Завдяки цьому він став однією з ключових мов програмування у сфері штучного інтелекту та аналізу даних. Використання готових фреймворків значно спрощує процес побудови моделей, прискорює експерименти та дозволяє зосередитися на вирішенні прикладних завдань, а не на реалізації алгоритмів «з нуля» [23].

Найпоширеніші бібліотеки:

– Scikit-learn. Ця бібліотека є однією з найпопулярніших для роботи з класичними алгоритмами машинного навчання. Вона містить реалізації методів класифікації, регресії, кластеризації, зниження розмірності та попередньої обробки даних. Завдяки єдиному інтерфейсу для різних алгоритмів Scikit-learn забезпечує простоту використання та швидке навчання нових моделей. Крім того, вона добре інтегрується з іншими інструментами для роботи з даними, такими як NumPy, SciPy та Pandas, що робить її незамінною у прикладних проєктах і для швидкого прототипування. Основним обмеженням є відсутність підтримки глибоких нейронних мереж і прискорених обчислень на GPU, проте для класичних задач машинного навчання вона залишається еталонним інструментом [24];

– PyTorch. Цей фреймворк було створено компанією Meta (Facebook), і він швидко здобув популярність завдяки своїй динамічній обчислювальній графіці, що забезпечує гнучкість і зручність у процесі розробки моделей. PyTorch надає дослідникам можливість експериментувати у режимі реального часу, спрощуючи процес налагодження та тестування моделей. Бібліотека активно підтримує інтеграцію з інструментами комп'ютерного зору (TorchVision), обробки природної мови (TorchText) та рекомендаційних систем (TorchRec). Крім того, PyTorch має потужну екосистему додаткових інструментів, включаючи PyTorch Lightning для структурованої побудови моделей та Hugging Face Transformers для роботи з сучасними мовними моделями. Завдяки активному розвитку й підтримці спільноти PyTorch став де-факто стандартом у наукових дослідженнях і все частіше використовується у виробничих рішеннях [25];

– TensorFlow. Це один із найпотужніших і наймасштабованіших фреймворків для глибокого навчання, розроблений компанією Google. Його архітектура дозволяє реалізовувати як прості експериментальні моделі, так і складні системи з мільйонами параметрів, здатні працювати на різних платформах – від мобільних пристроїв до розподілених обчислювальних кластерів. TensorFlow підтримує не лише гнучке створення моделей, а й їх

подальше розгортання завдяки сервісам TensorFlow Serving і TensorFlow Lite. Вбудовані інструменти, такі як TensorBoard, надають можливість візуалізації процесу навчання, що є важливим для аналізу та оптимізації моделей. Крім того, TensorFlow має багатий набір попередньо навчених моделей та інтеграцію з бібліотекою Keras, яка дозволяє швидко створювати та тестувати нейронні мережі за допомогою високорівневого API. Завдяки цьому TensorFlow широко застосовується як у дослідженнях, так і у промислових системах, де потрібна масштабованість та продуктивність [26].

Порівняння характеристик бібліотек для машинного навчання наведено у таблиці 2.4.

Таблиця 2.4 – Порівняння бібліотек для машинного навчання

Характеристика	Scikit-learn	TensorFlow	PyTorch
Призначення	Класичні ML-алгоритми	Глибоке навчання	Глибоке навчання, дослідження
Простота використання	Висока	Середня	Висока
Підтримка GPU	Ні	Так	Так
Гнучкість	Обмежена	Висока	Висока
Ліцензія	BSD	Apache 2.0	BSD

З огляду на вимоги до гнучкості, можливість використання нейронних мереж, а також підтримку продуктивного середовища, було обрано TensorFlow. Цей фреймворк має потужний інструментарій для побудови моделей глибокого навчання, підтримку GPU для прискорення обчислень, а також зручну інтеграцію з високорівневим API Keras, що суттєво спрощує створення та навчання моделей.

2.5 Висновки за розділом 2

Для реалізації мобільного застосунку для ідентифікації породи тварин було обрано мову програмування Java з використанням IDE Android Studio для клієнтської частини та Python з використанням IDE PyCharm для серверної. Такий підхід дозволяє поєднати перевірені та стабільні інструменти для розробки інтерфейсу користувача з потужними можливостями для реалізації алгоритмів штучного інтелекту та машинного навчання на стороні сервера. Це рішення забезпечує гнучкість архітектури програмного забезпечення, спрощує інтеграцію клієнтської та серверної частин, а також дає можливість ефективно розподілити обчислювальні навантаження між мобільним пристроєм і сервером.

Вибір мов програмування був зумовлений їхніми сильними сторонами та оптимальністю для конкретних завдань. Java виступає однією з базових мов Android-розробки, має офіційну підтримку від Google та забезпечує сумісність із різними версіями операційної системи. Завдяки великій екосистемі бібліотек і готових рішень вона дозволяє швидко створювати інтуїтивний та продуктивний інтерфейс користувача. Використання Java також гарантує стабільність і надійність клієнтської частини застосунку, що є критично важливим для взаємодії користувача з системою в реальному часі. Python, своєю чергою, вирізняється простотою синтаксису та широким вибором бібліотек для реалізації алгоритмів машинного навчання. Його потужність особливо проявляється у завданнях, пов'язаних із комп'ютерним зором, аналізом зображень і застосуванням нейронних мереж, що робить його найкращим вибором для серверної частини системи.

Не менш важливим є правильний вибір інтегрованих середовищ розробки. Android Studio як офіційне IDE для Android забезпечує повний цикл створення застосунку: від проєктування інтерфейсу та написання коду до налагодження, тестування й розгортання на реальних або віртуальних пристроях. Його функціонал, зокрема вбудований емулятор, система підказок і

засоби інтеграції з інструментами контролю версій, значно підвищує ефективність роботи над клієнтською частиною. PyCharm надає розробнику широкий набір інструментів для роботи з Python, включаючи зручне управління залежностями, інтеграцію з фреймворками штучного інтелекту (такими як TensorFlow чи PyTorch), а також розширені можливості налагодження. Завдяки цьому процес створення, тестування й оптимізації серверної частини стає більш структурованим і продуктивним. Поєднання цих IDE дозволяє не лише прискорити розробку, а й підвищити якість програмного забезпечення, забезпечивши його гнучкість, масштабованість і зручність подальшої підтримки.

3 ОПИС ПРОГРАМИ

3.1 Структура мобільного застосунку для ідентифікації порід тварин

Структура мобільного застосунку для ідентифікації порід тварин наведена на рис. 3.1.

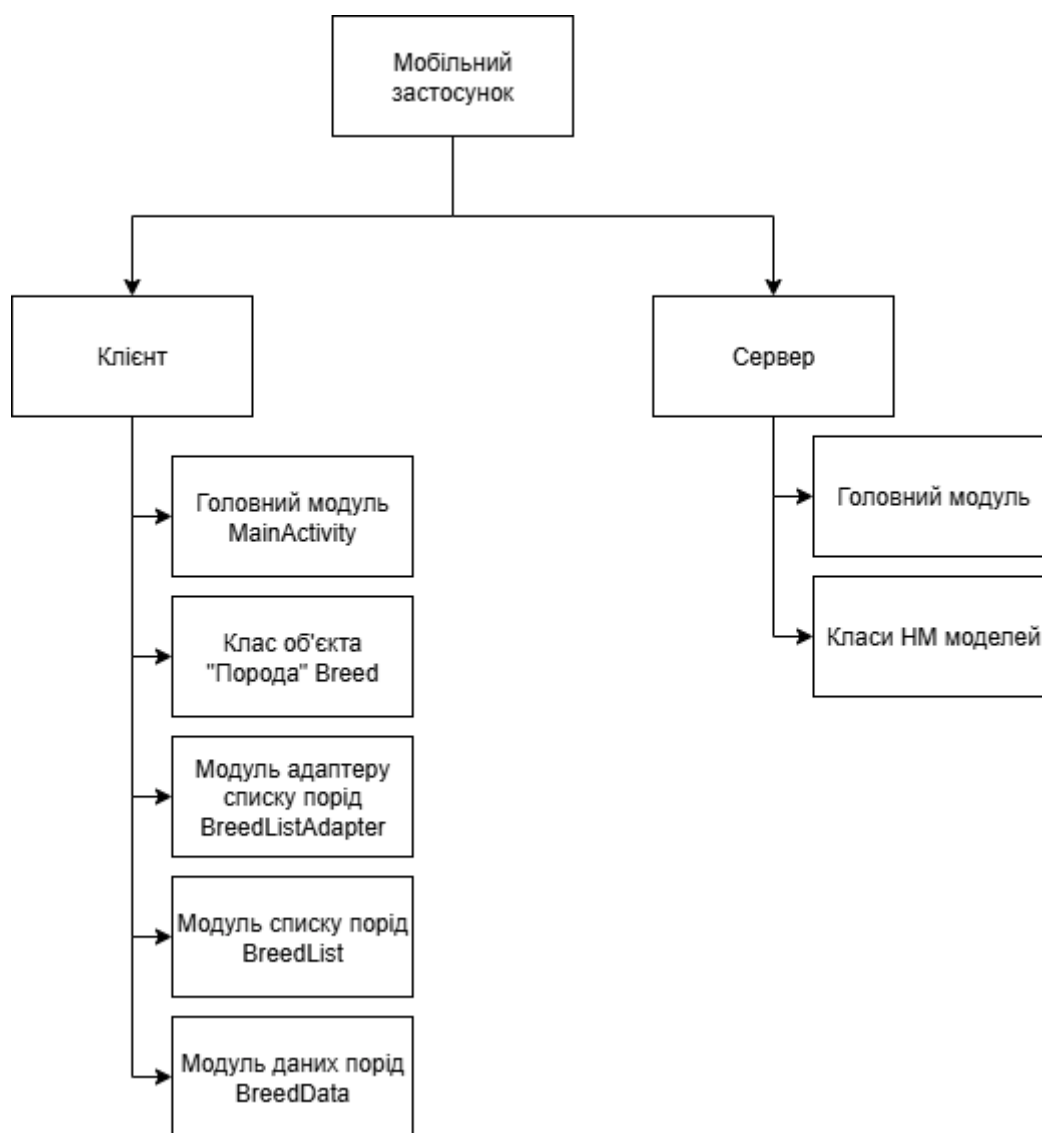


Рисунок 3.1 – Структура мобільного застосунку для ідентифікації порід тварин

Як можна побачити, мобільний застосунок складається з двох частин: клієнтської та серверної.

Клієнтська частина написана на Android з використанням мови програмування Java та середовища розробки Android Studio. Вона складається з п'яти модулів:

- головний модуль MainActivity. Виконує основну роль у роботі клієнтської частини застосунку. Відповідає за вибір користувачем зображення для розпізнавання, його попередню обробку, відправку та отримання результату;
- клас об'єкта “Порода” Breed. Клас, що використовується для зберігання даних про породу тварини;
- модуль адаптеру списку порід “BreedListAdapter”. Адаптер списку порід. Необхідний для правильного відображення списку порід;
- модуль списку порід BreedList. Необхідний для демонстрації списку порід, що можуть бути розпізнані мобільним застосунком;
- модуль даних порід BreedData. Слугує для збереження даних списку порід які можуть бути розпізнані мобільним застосунком.

3.2 Функціонування мобільного застосунку для ідентифікації порід тварин

Алгоритм функціонування клієнтської частини мобільного застосунку наведено на рисунку 3.2.

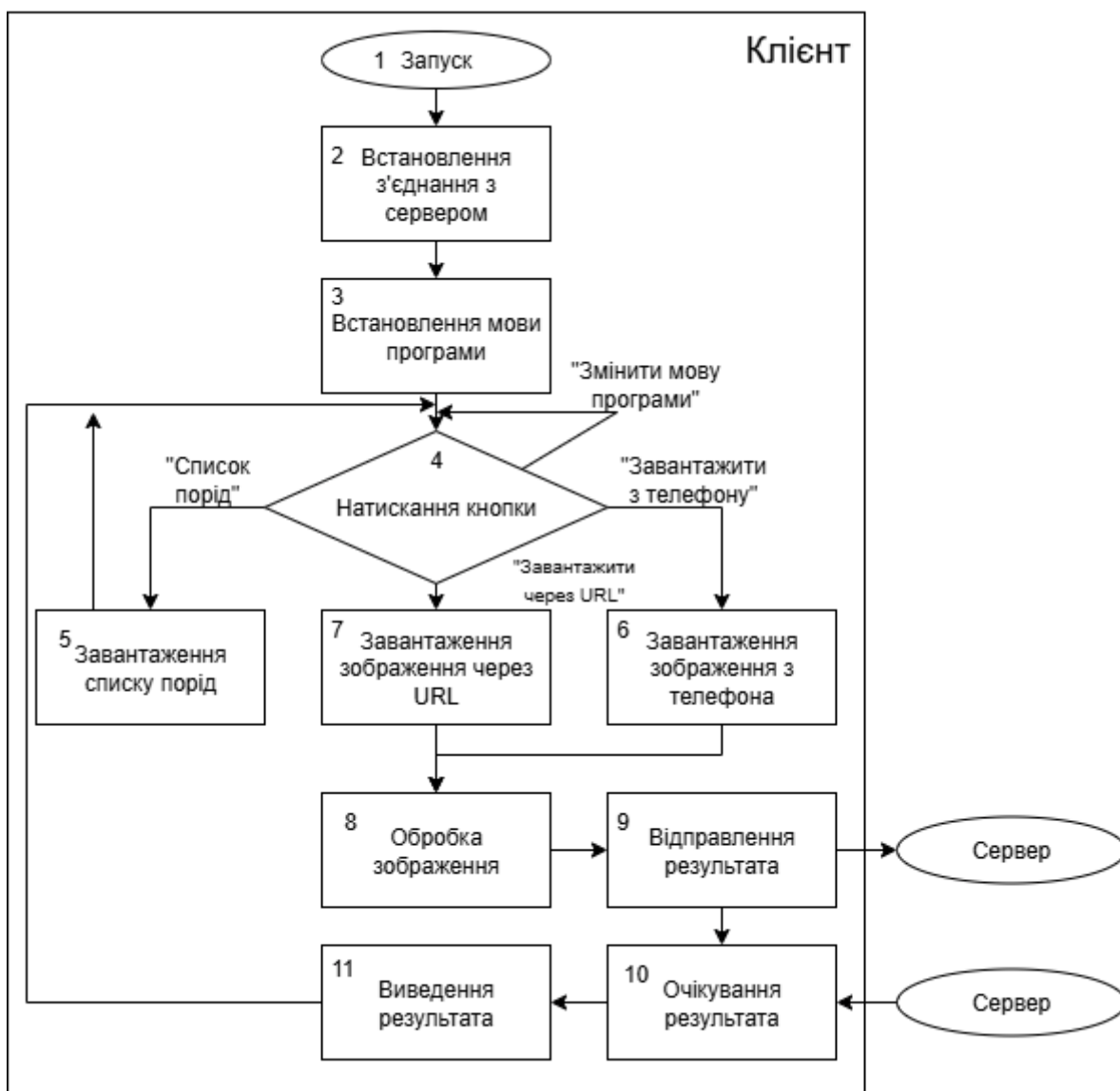


Рисунок 3.2 – Алгоритм функціонування клієнтської частини мобільного застосунку наведено

Як видно зі схеми, клієнтська частина містить 12 блоків.

Блок 1. Запуск програми.

Блок 2. Виконується встановлення з'єднання з сервером.

Блок 3. На цьому етапі виконується встановлення мови програми, це або українська, або англійська, в залежності від налаштувань телефона користувача

Блок 4. В залежності від кнопки яку натискає користувач, виконується перехід до блоку 5, якщо натиснута кнопка “Список порід”, блоку 7, якщо

натиснута кнопка “Завантажити через URL”, та блоку 6, якщо натиснута кнопка “Завантажити з телефону”. Якщо натиснута кнопка “Змінити мову програми” відбувається перехід до цього ж блоку та встановлюється наступна по списку мова програми.

Блок 5. Виконує завантаження та відображення списку порід, які можуть бути розпізнані застосунком. При закритті (кнопка на формі “назад”, або вбудована в телефон функція) виконується перехід до блоку 4.

Блок 6. Відкриває вікно з текстовим рядком, у який користувач повинен вставити URL зображення та натиснути кнопки “Load” для завантаження зображення у програму. Перехід до блоку 8.

Блок 7. Відкриває стандартне сховище зберігання даних. Для завантаження доступні лише файли формату “.jpg/.jpeg” та “.png”. Для завантаження користувач натиснути на відповідне зображення. Перехід до блоку 8

Блок 8. На даному етапів виконується обробка зображення, яка включає наступні дії: встановлення розміру 800:600, зміна формату на “.png” та перетворення у масив байтів. Перехід до блоку 9

Блок 9. Компонує запит для відправки на сервер, що складається з масиву байтів зображення, ідентифікатора користувача, мови програми та обраної моделі розпізнавання. Після цього надсилається результат на сервер. Перехід до блоку 10.

Блок 10. Очікує відповідь від сервера. Після отримання відповіді виконується перехід до блоку 11.

Блок 11. Виводить у текстове поле отриману від сервера інформацію про породу, яка була розпізнана та її особливості. Перехід до блоку 4.

Алгоритм функціонування серверної частини мобільного застосунку наведено на рисунку 3.3.

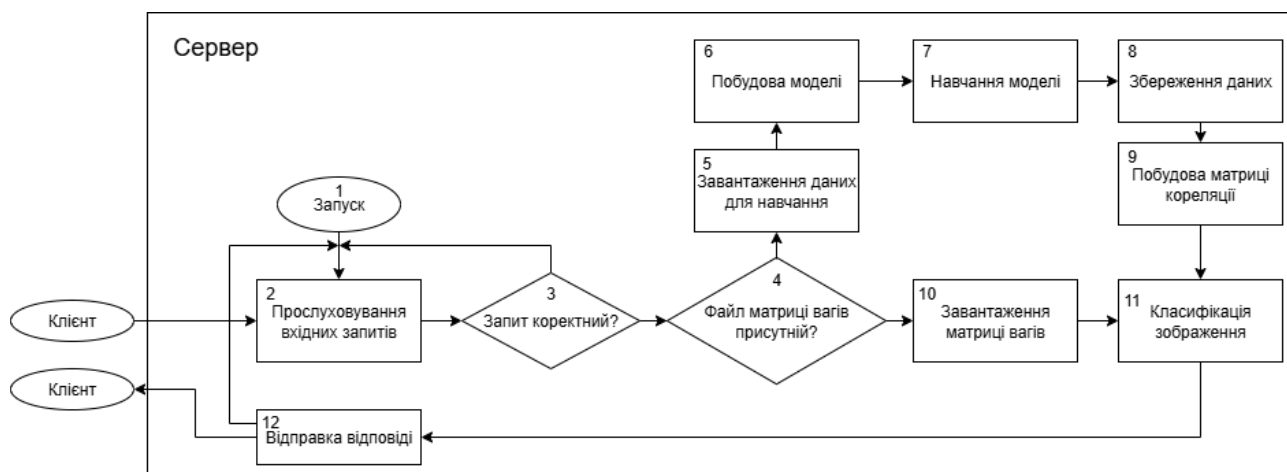


Рисунок 3.3 – Алгоритм функціонування серверної частини мобільного застосунку наведено

Як видно зі схеми, серверна частина містить 12 блоків.

Блок 1. Запуск програми.

Блок 2. Сервер постійно прослуховує вхідні запити користувачів і як тільки вони надходять, демонструє його на формі та виконує перехід до блоку 3.

Блок 3. Якщо запит від користувача містить його ідентифікатор, зображення, мову програми та ідентифікатор обраної користувачем моделі для розпізнавання, то виконується перехід до блоку 4, якщо ні, то виконується перехід до блоку 2.

Блок 4. Виконує перевірку на наявність файлу матриці вагів на сервері. Якщо такий файл не знайдений виконується перехід до кроку 5, якщо файл є виконується перехід до блоку 10.

Блок 5. Завантажує набір даних, що будуть використовуватися для навчання моделі. Розподіляє їх на навчальні та тестові. Перехід до блоку 6.

Блок 6. В залежності від обраної користувачем моделі, виконується її побудова та компіляція. Перехід до блоку 7.

Блок 7. Навчання моделі. В залежності від моделі, виконується у 5 або 30 епох. Перехід до блоку 8.

Блок 8. Результат навчання у вигляді матриці вагів зберігається у файлі. Перехід до блоку 9.

Блок 9. Виконує побудову матриці кореляції для графічного відображення точності навченої моделі. Перехід до блоку 11.

Блок 10. Завантажує матрицю вагів навченої моделі для подальшої роботи. Перехід до блоку 11.

Блок 11. Для класифікації, вхідне зображення зменшується до розмірів 128 на 128 пікселів (в деяких моделях до 224 на 224 пікселів) та нормалізується.. Після цього виконується класифікація зображення. Далі формує результат, що складається з назви класу та точності даного розпізнавання. Перехід до блоку 12.

Блок 12. Формує результат для відправки користувача, що складається з назви породи, та обраної зі словника, особливості даної породи. Перехід до блоку 2.

3.3 Проектування інтерфейсу мобільного застосунку для ідентифікації порід тварин

Графічний інтерфейс мобільного застосунку складається з двох форм: головна форма та список порід.

Головна форма мобільного застосунку складається з чотирьох основних кнопок, кожна з яких відповідає за певну функціональність: завантаження зображення з пам'яті телефону, введення URL для завантаження зображення з Інтернету, відкриття списку доступних порід тварин та зміну мови інтерфейсу програми. Крім того, форма містить список із назвами моделей розпізнавання, що дозволяє користувачу обирати конкретну модель для обробки зображень.

Важливою частиною інтерфейсу є текстове поле, в якому відображається результат розпізнавання породи тварини на основі завантаженого або обраного зображення. Це забезпечує зручність і наочність для користувача, оскільки результат стає доступним безпосередньо на головній формі. Крім того, на формі відображається саме зображення, обране або завантажене користувачем, що

дозволяє порівняти оригінал з отриманим результатом і підвищує наочність роботи застосунку. Макет даної форми продемонстровано на рисунку 3.4.

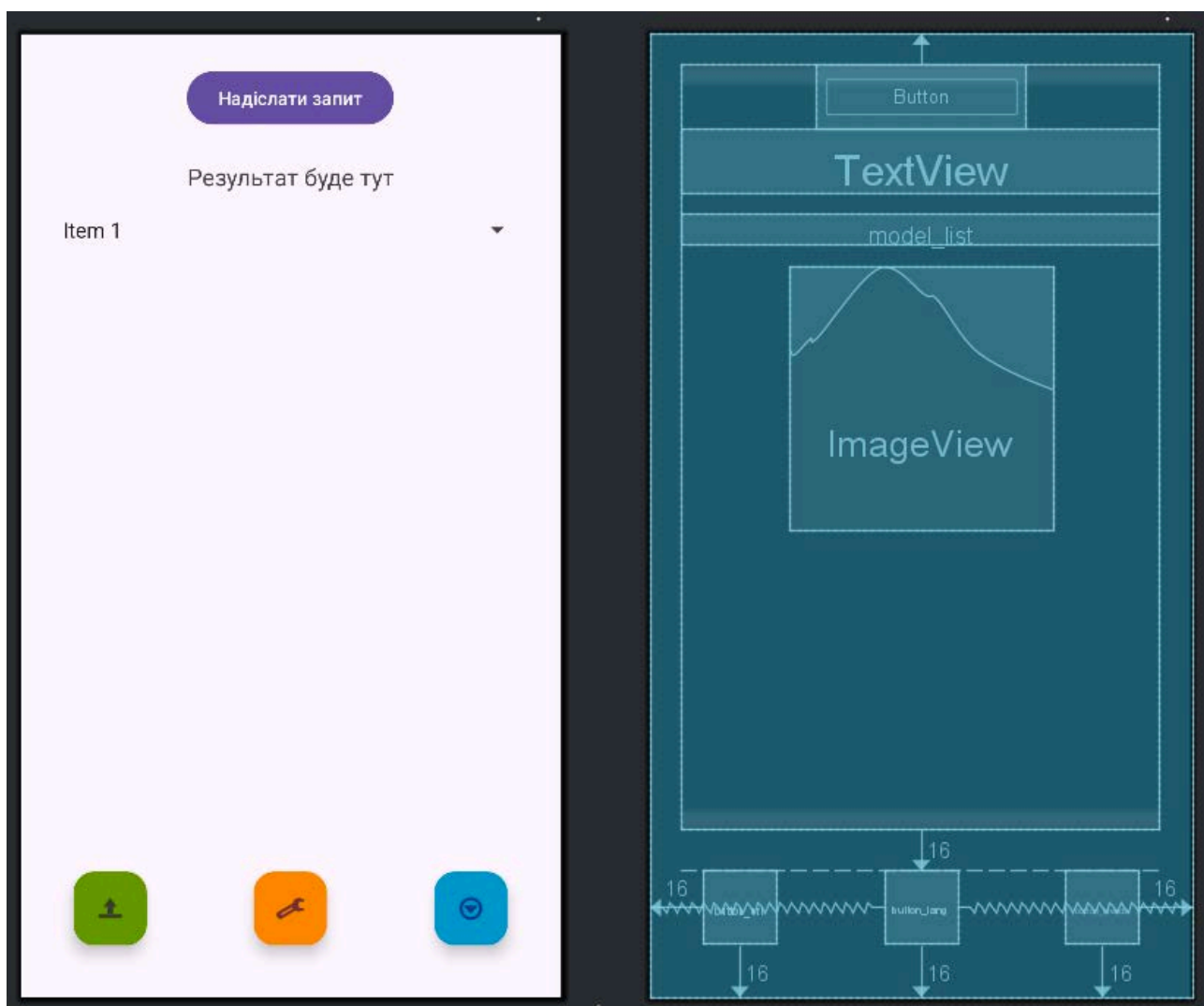


Рисунок 3.4 – Головна форма програми

Форма списку порід складається з кнопки “Назад” та списку, кожен елемент якого складається з зображення тварини, назви породи та її опису. Макет даної форми продемонстровано на рисунку 3.5.

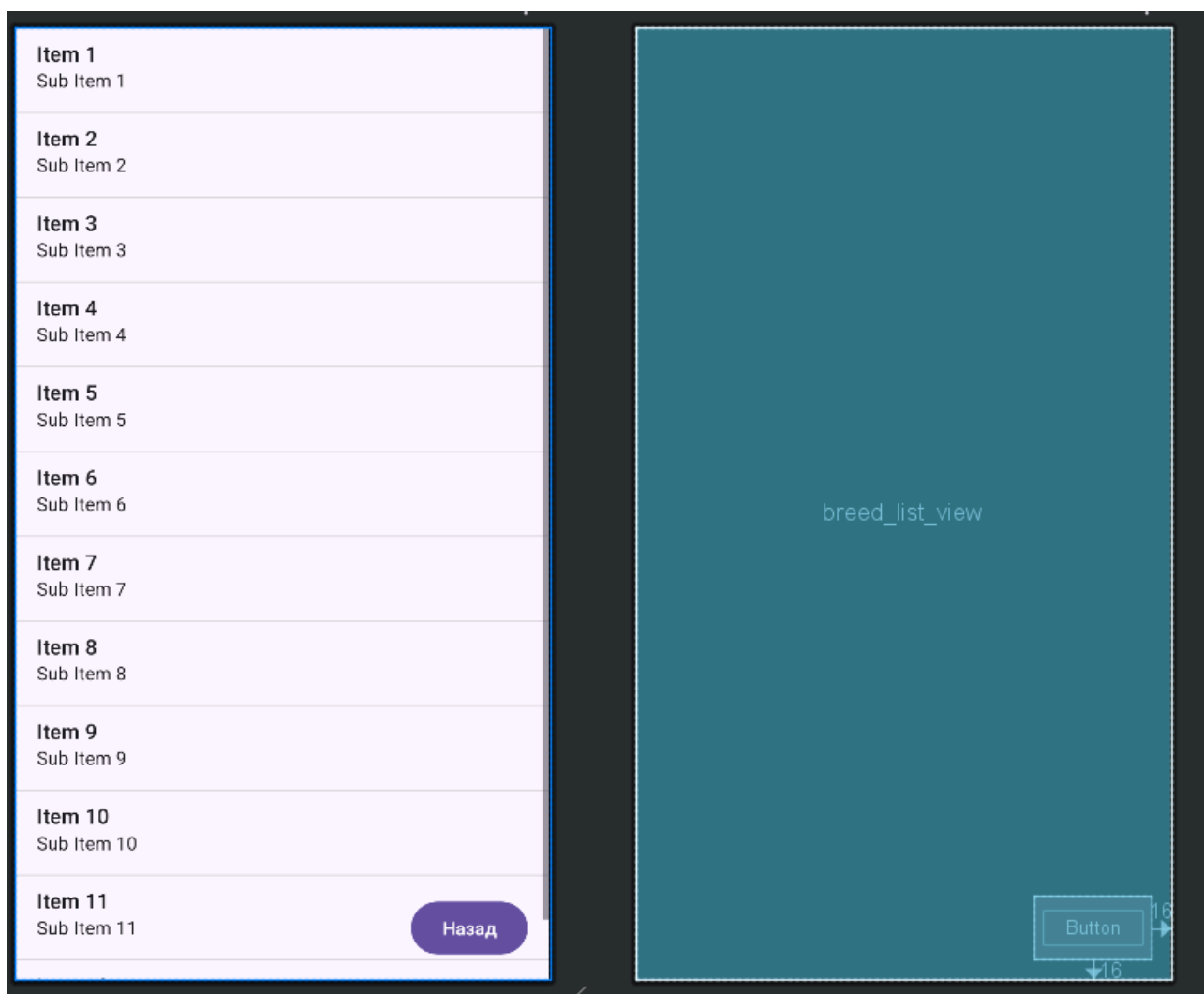


Рисунок 3.5 – Форма списку порід

Серверна частина мобільного застосунку на містить графічного інтерфейсу користувача й тільки виводить на форму останнє отримане від клієнта зображення.

3.4 Архітектура створених моделей нейронних мереж

В рамках виконання роботи для розпізнавання породи тварини, було використано п'ять моделей: MobileNetV2, ResNet50, EfficientNetB0, DenseNet121, та власноруч створена модель згорткової НМ (CNN).

3.4.1 MobileNetV2

Архітектура моделі MobileNetV2 наведена на рисунку 3.6.

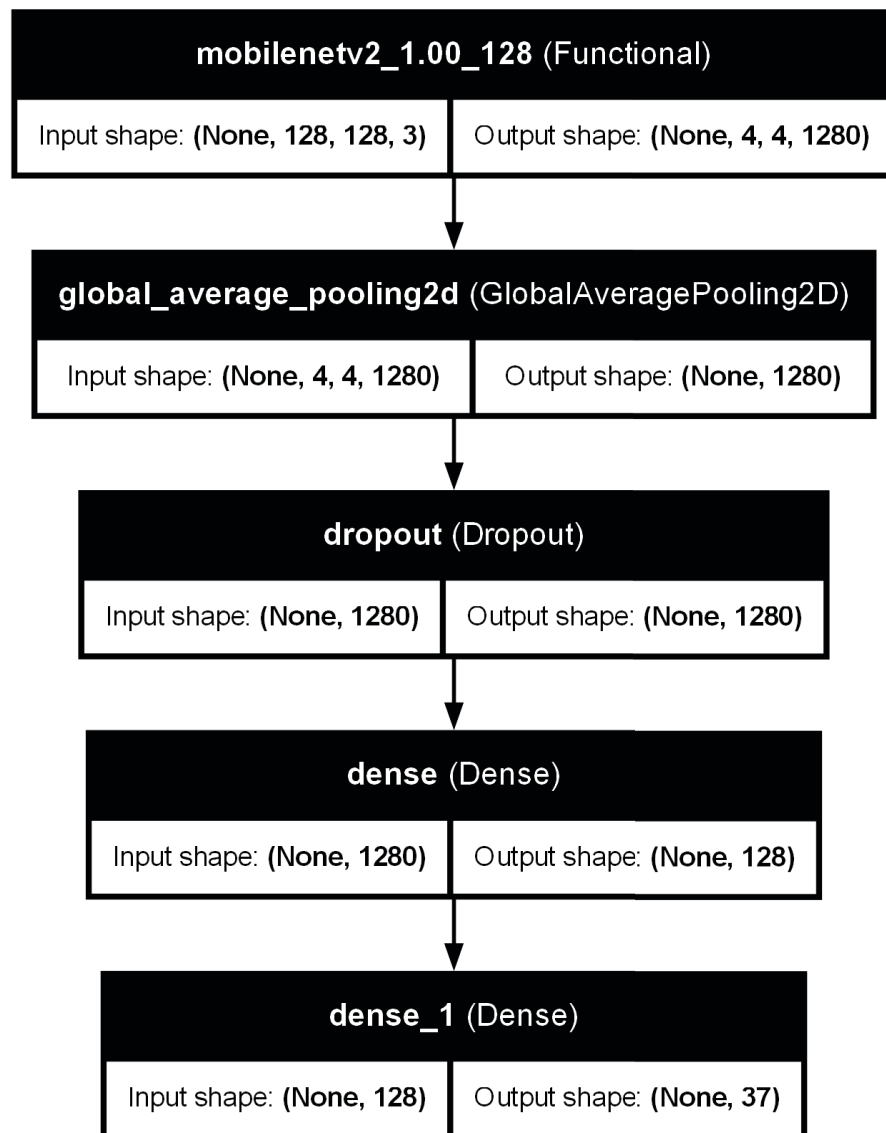


Рисунок 3.6 – Архітектура моделі MobileNetV2.

Дана модель побудована на архітектурі MobileNetV2, яка оптимізована для ефективного розпізнавання зображень з невеликими обчислювальними витратами. На вхід подається зображення розміром $128 \times 128 \times 3$, після чого базова частина MobileNetV2 виконує багаторівневу обробку та виділення ознак, формуючи тензор розмірності $4 \times 4 \times 1280$. Далі застосовується шар Global

Average Pooling, що згортає просторову інформацію у вектор ознак довжиною 1280. Для зменшення ризику перенавчання використовується шар Dropout, після якого розташований повнозв'язний шар (Dense) на 128 нейронів. Завершальним є вихідний повнозв'язний шар із 37 нейронів, кожен з яких відповідає певному класу та повертає ймовірність належності зображення до цього класу.

3.4.2 Convolutional neural network

Архітектура моделі CNN зображена на рисунку 3.7.

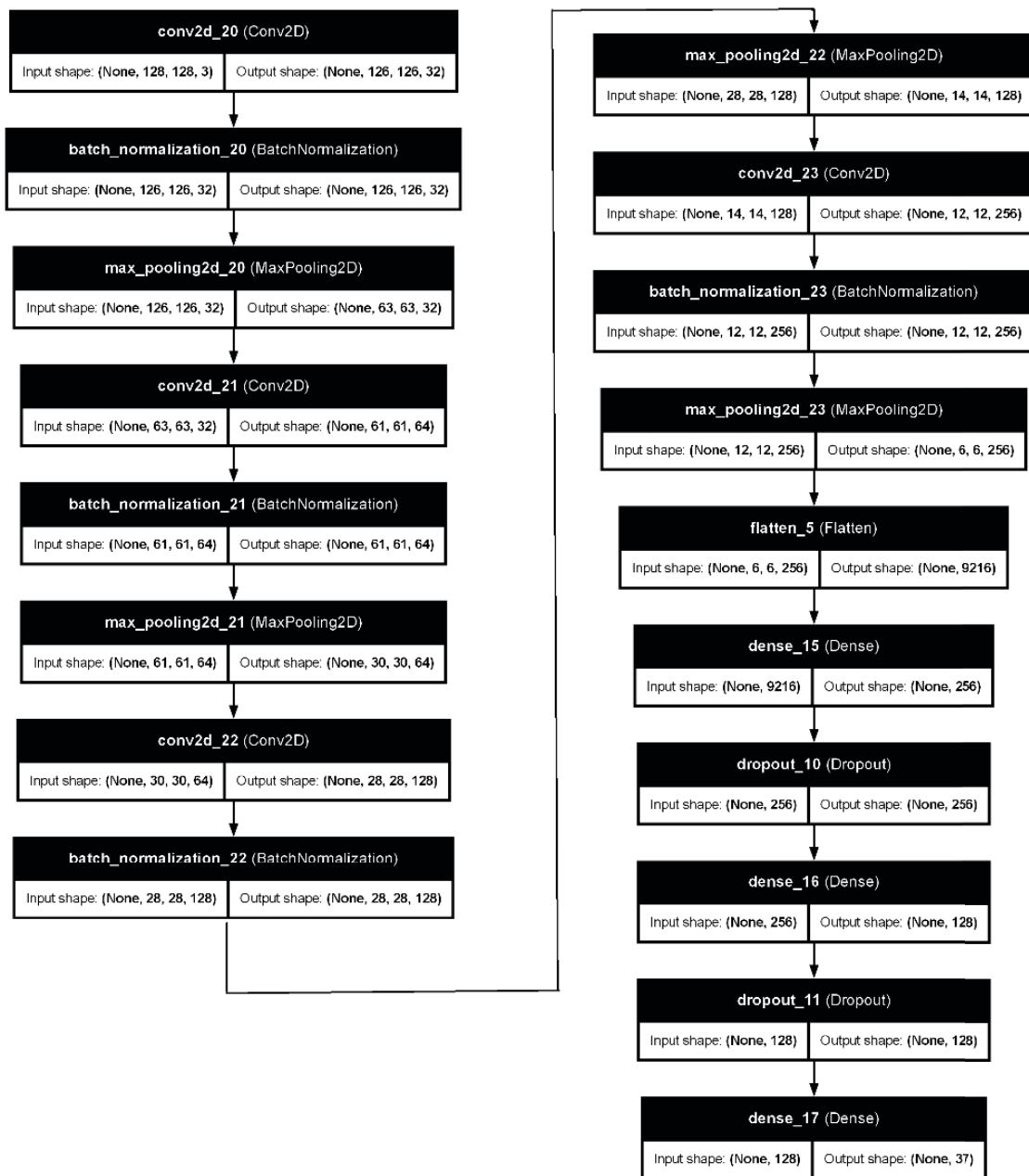


Рисунок 3.7 – Архітектура моделі Convolutional Neural Network.

Ця модель є класичною згортковою нейронною мережею (CNN) для класифікації зображень. На вхід подається зображення розміром $128 \times 128 \times 3$, яке послідовно проходить через кілька блоків: згорткові шари (Conv2D) для виділення локальних ознак, Batch Normalization для стабілізації та прискорення навчання, а також MaxPooling2D для зменшення просторових розмірів і концентрації на найбільш важливих характеристиках. У процесі збільшується кількість каналів ознак (від 32 до 256), що дозволяє мережі виявляти дедалі

складніші патерни. Після останнього шару згортки та пулінгу вихідні дані перетворюються у вектор за допомогою шару Flatten, який подається на кілька повнозв'язних шарів (Dense). Для запобігання перенавчанню застосовуються Dropout-шари. Завершальний шар містить 37 нейронів, кожен з яких відповідає певному класу, а його вихід відображає ймовірність належності зображення до цього класу.

3.4.3 ResNet50

Архітектура моделі ResNet50 зображена на рисунку 3.8.

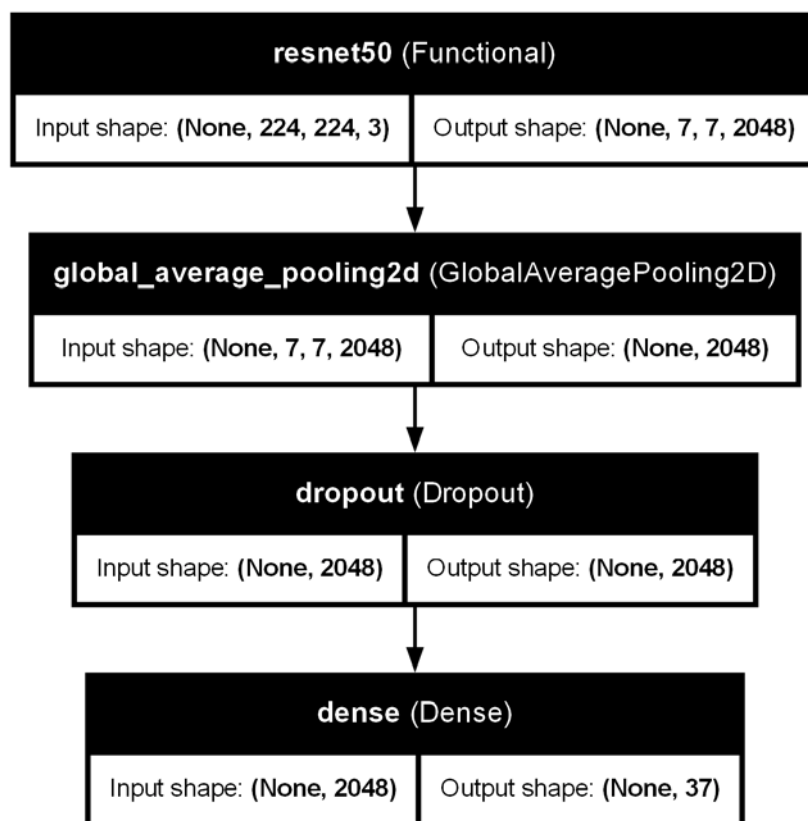


Рисунок 3.8 – Архітектура моделі ResNet50.

Модель побудована на архітектурі ResNet50, яка використовує механізм залишкових зв'язків для глибокого навчання та ефективного виділення ознак. На вхід подається зображення розміром $224 \times 224 \times 3$, яке обробляється базовою

частиною ResNet50, що формує тензор розміром $7 \times 7 \times 2048$. Далі застосовується шар Global Average Pooling, який перетворює просторові ознаки у вектор довжиною 2048. Для зменшення ризику перенавчання додається шар Dropout, після чого використовується вихідний повнозв'язний шар (Dense) із 37 нейронів. Кожен вихідний нейрон відповідає певному класу та відображає ймовірність належності вхідного зображення до цього класу.

3.4.4 EfficientNetB0

Архітектура моделі EfficientNetB0 зображена на рисунку 3.9.

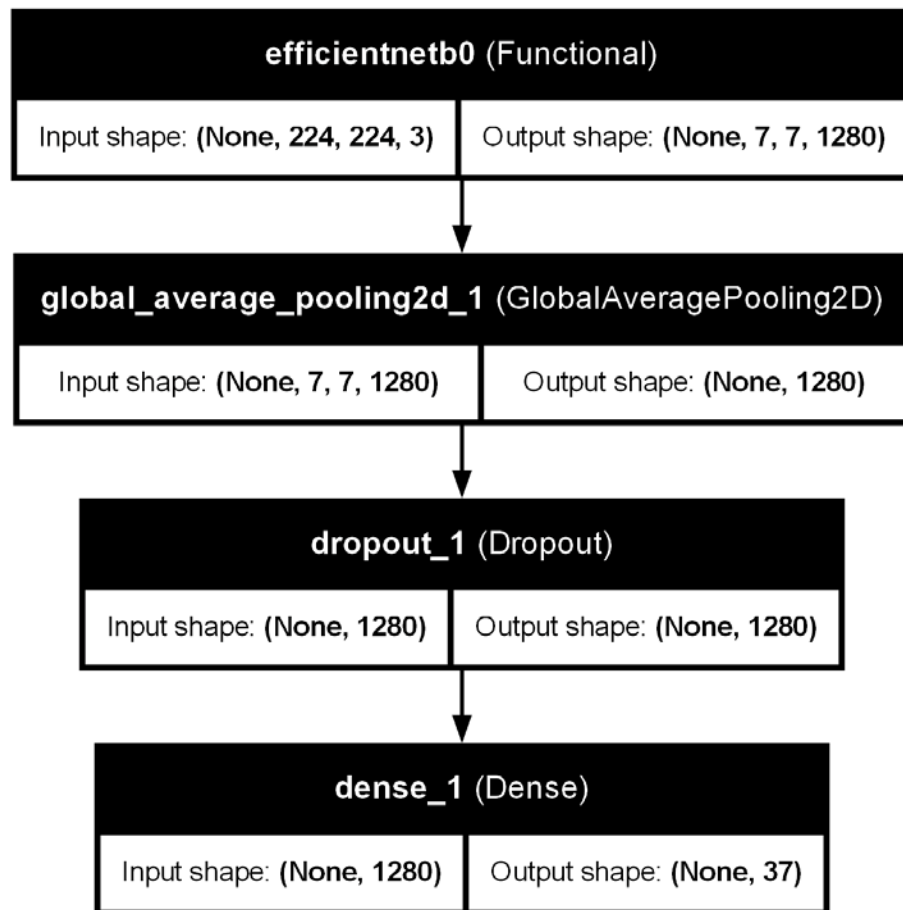


Рисунок 3.9 – Архітектура моделі EfficientNetB0.

Архітектура моделі побудована на основі попередньо навченої нейронної мережі EfficientNetB0, яка використовується для витягання ознак із вхідних

зображень розміром 224×224 пікселі з трьома каналами. Після основної частини мережі застосовується шар `GlobalAveragePooling2D`, що зменшує просторові виміри та перетворює карти ознак у вектор фіксованої довжини 1280. Для запобігання перенавчанню використовується шар `Dropout`, після чого вихідні дані надходять на фінальний `Dense`-шар з 37 нейронами, який виконує класифікацію за кількістю заданих класів.

3.4.5 DenseNet121

Архітектура моделі `DenseNet121` зображена на рисунку 3.10.

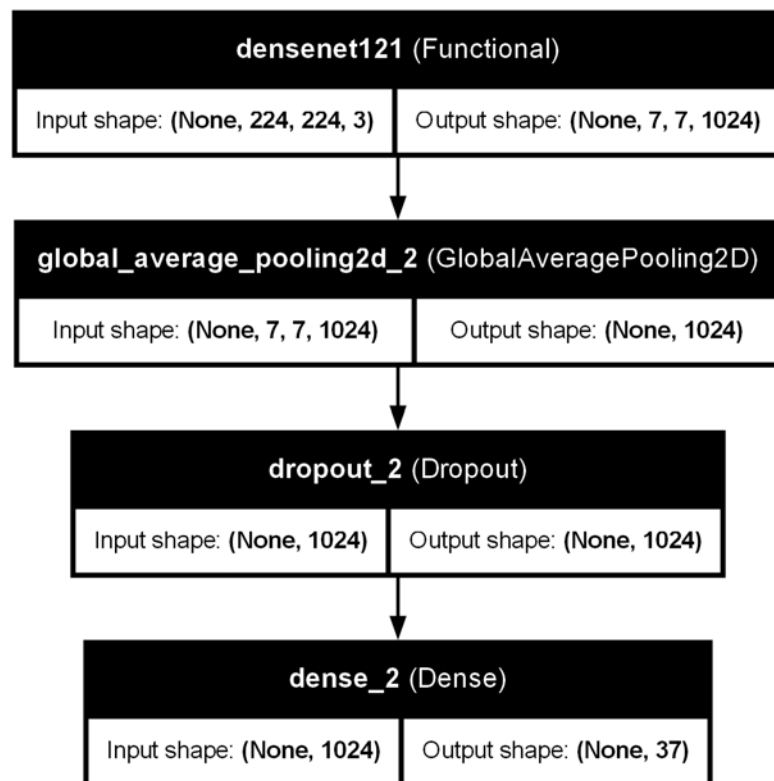


Рисунок 3.10 – Архітектура моделі `DenseNet121`.

Архітектура моделі побудована на основі попередньо навчений нейронної мережі `DenseNet121`, яка приймає зображення розміром 224×224 пікселі з трьома каналами та генерує тензор ознак розміром $7 \times 7 \times 1024$. Далі застосовується шар `GlobalAveragePooling2D`, який перетворює отримані карти

ознак у вектор довжиною 1024. Для зменшення ризику перенавчання використовується шар Dropout, після чого вектор подається на вихідний Dense-шар із 37 нейронами, що відповідають кількості класів класифікації.

3.5 Висновки за розділом 3

Як результат проведеної роботи, у третьому розділі було детально графічно відображено структуру мобільного застосунку, що дозволяє наочно побачити взаємозв'язки між клієнтською та серверною частинами системи, а також основні компоненти, що забезпечують його функціонування. Крім того, наведено та описано алгоритм роботи клієнта і сервера, який демонструє послідовність обробки даних, від завантаження зображення користувачем до отримання результату розпізнавання породи тварини.

У розділі також докладно описано інтерфейс мобільного додатка та серверної частини, що дозволяє зрозуміти, як користувач взаємодіє з системою та які інструменти застосовуються для обробки запитів. Висвітлено функціональні елементи головної форми застосунку, кнопки для завантаження зображень, вибору моделей розпізнавання та відображення результатів, а також інтерфейс серверної частини для обробки запитів і взаємодії з нейронними мережами.

Окрім цього, у розділі детально розглянуто архітектуру використаних моделей нейронних мереж. Було описано структуру шарів та принципи обробки вхідних даних, що забезпечують точне розпізнавання порід тварин.

4. ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Призначення й умови застосування мобільного застосунку

Мобільний застосунок призначений для ідентифікації порід тварин за зображенням, що завантажується користувачем. Клієнтська частина застосунку реалізована мовою програмування Java з використанням інтегрованого середовища розробки Android Studio, яке є офіційним інструментом для створення нативних Android-додатків і забезпечує повну інтеграцію з Android SDK. Серверна частина розроблена мовою Python у середовищі PyCharm, що надає зручні інструменти для роботи з бібліотеками машинного навчання та полегшує процес налагодження і тестування коду. Такий поділ на клієнтську та серверну частини дозволяє поєднати стабільність і швидкодію інтерфейсу користувача з гнучкістю і потужністю алгоритмів глибинного навчання.

Програмне забезпечення для ідентифікації порід тварин забезпечує виконання таких функцій:

- можливість завантаження зображення з галереї пристрою або за URL-адресою;
- автоматичне розпізнавання породи тварини за допомогою моделей нейронних мереж;
- відображення результату і ймовірності правильності класифікації;
- доступ до списку підтримуваних порід у зручному форматі;
- можливість зміни мови інтерфейсу мобільного застосунку;
- підтримка декількох моделей розпізнавання з можливістю їх вибору.

Для експлуатації мобільного застосунку необхідно таке апаратне та програмне забезпечення:

- смартфон або планшет з операційною системою Android 7.0 або новішою;
- персональний комп'ютер з доступом до мережі інтернет, процесором з тактовою частотою не менше 2.2 ГГц, з встановленим Python,

бібліотекою TensorFlow та достатнім обсягом оперативної пам'яті (від 4 ГБ) для обробки зображень та роботи моделей нейронних мереж.

Функціональні характеристики розробленого мобільного застосунку для ідентифікації порід тварин наведено у технічному завданні (додаток А). Фрагмент тексту мобільного застосунку для ідентифікації порід тварин за зображенням наведено у додатку Б.

4.2 Характеристики мобільного застосунку для ідентифікації порід тварин

Мобільний застосунок для ідентифікації порід тварин характеризується кількома ключовими аспектами, що забезпечують його ефективність і зручність використання. Основною функцією є можливість автоматичного визначення породи тварини за зображенням, завантаженим користувачем із галереї пристрою або за допомогою URL-адреси. Для цього застосунок використовує алгоритми глибокого навчання, що дозволяють здійснювати точну класифікацію та формувати результат із зазначенням рівня ймовірності.

Гнучкість застосунку полягає у можливості використання різних моделей нейронних мереж, серед яких користувач може обрати найбільш оптимальну. Крім того, передбачена підтримка багатомовного інтерфейсу, що робить програму доступною для ширшого кола користувачів. Система побудована за клієнт-серверною архітектурою, де мобільний додаток відповідає за зручну взаємодію з користувачем, а серверна частина забезпечує обробку даних та виконання ресурсомістких операцій. Такий підхід дозволяє поєднати простоту використання з високою точністю роботи моделей.

Важливими характеристиками також є наявність інтуїтивно зрозумілого інтерфейсу, що передбачає чітку структуру головної форми з кнопками завантаження зображення, відкриття списку порід та зміни мови застосунку. Додатково передбачено можливість відображення списку підтримуваних порід і результатів класифікації у зручному форматі. Автоматизація процесів обробки

зображень зменшує кількість дій з боку користувача та підвищує швидкість отримання результату. Система також підтримує графічне відображення обраного зображення, що робить роботу з додатком наочною та більш зручною.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску серверної частини мобільного застосунку для ідентифікації порід тварин необхідно запустити основний ру-файл. Це можна зробити у командному рядку, або в середовищі, що підтримує розробку програм на мові Python. Клієнтська частина застосунку запускається на смартфоні або планшеті з операційною системою Android, та що задовольняє апаратним вимогам.

4.3.2 Вхідні й вихідні дані

Вхідними даними до мобільного застосунку для ідентифікації порід тварин є зображення завантажене користувачем та набір даних для навчання моделей розпізнавання зображень.

Вихідними даними мобільного застосунку для ідентифікації порід тварин є результат розпізнавання породи тварини, що складається з її породи та опису.

4.3.3 Повідомлення

Користувач мобільного застосунку для ідентифікації порід тварин може отримати такі повідомлення: повідомлення про очікування результату розпізнавання породи тварини, повідомлення з результатом розпізнавання породи тварини.

4.4 Виконання мобільного застосунку для ідентифікації порід тварин

Після запуску мобільного застосунку користувача зустрічає головна форма програми, яка зображена на рисунку 4.1. На даній формі можна побачити чотири кнопки, кожна з яких відповідає за певну функціональність: завантаження зображення з пам'яті телефону, введення URL для завантаження зображення з Інтернету, відкриття списку доступних порід тварин та зміну мови інтерфейсу програми. Крім того, форма містить список із назвами моделей розпізнавання, що дозволяє користувачу обирати конкретну модель для обробки зображень..



Рисунок 4.1 – Головна форма програми

Коли користувач натискає кнопку “Надіслати запит” відкривається внутрішнє сховище, звідки користувач повинен обрати зображення, яке буде надіслане на сервер для розпізнавання породи тварини на ньому (рис 4.2).



Рисунок 4.2 – Відкриття внутрішнього сховища

Після того як користувач обрав зображення, воно надсилається на сервер й користувач має почекати поки виконується ідентифікація породи тварини (рис 4.3).

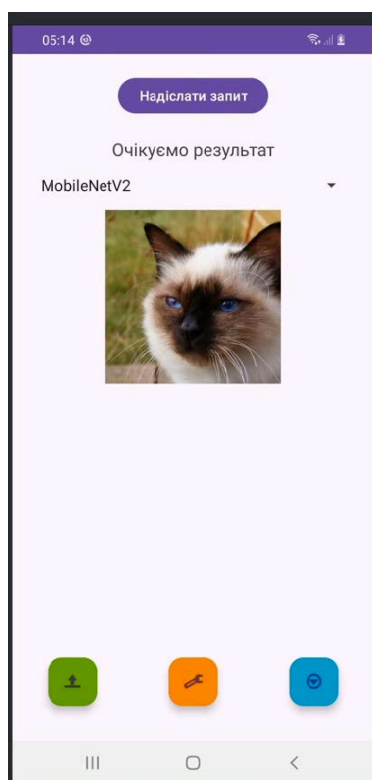


Рисунок 4.3 – Очікування результату

Після того як порода тварини ідентифікована користувач може побачити результат на головній формі програми (рис 4.4).

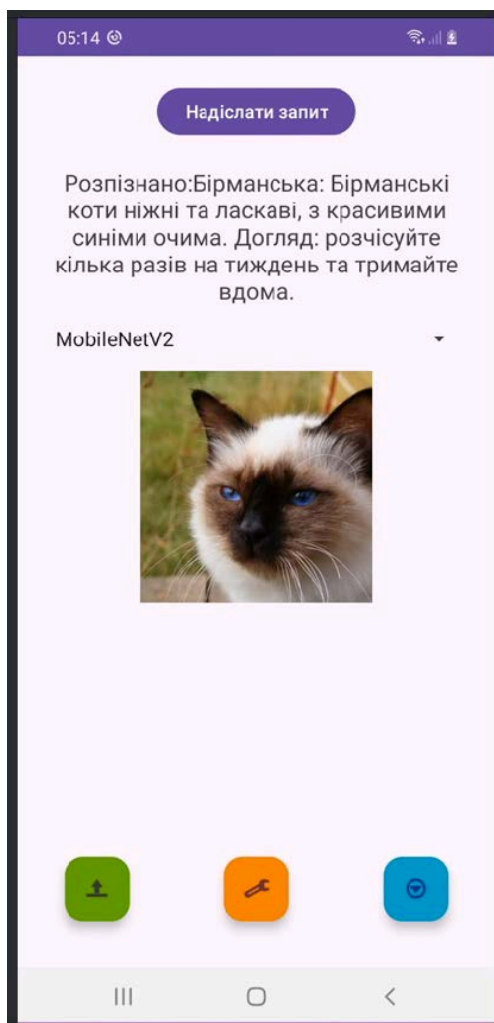


Рисунок 4.4 – Відображення результату

Натиснувши кнопку у лівому нижньому куті, користувач може завантажити бажане зображення через його URL-посилання (рис 4.5).



Рисунок 4.5 – Завантаження через URL

Натиснувши кнопку у правому нижньому куті, користувач може відкрити список порід, які можуть бути розпізнані програмою, та їх короткий опис.



Рисунок 4.6 – Форма списку порід

Натиснувши на головній формі кнопку, яка знаходиться посередині нижньої частини форми, користувач перемикне мову програми на наступну англійську, або українську в залежності від поточного стану (рис 4.7).

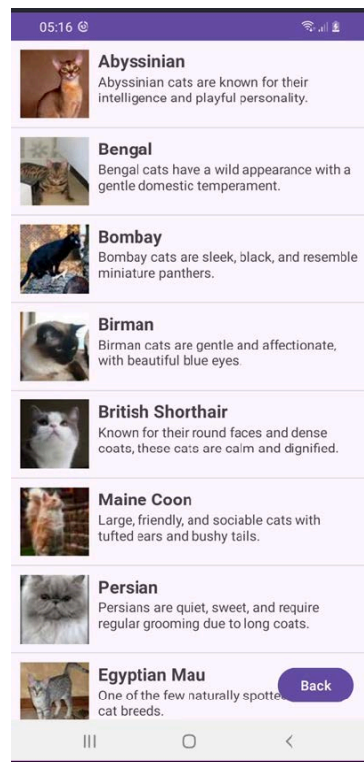


Рисунок 4.7 – Форма списку порід. Англійська локалізація

4.5 Порівняння ефективності моделей нейронних мереж

В рамках виконання роботи для розпізнавання породи тварини, було використано п'ять моделей: MobileNetV2, ResNet50, EfficientNetB0, DenseNet121, та власноруч створена модель згорткової НМ (CNN).

Ефективність розпізнавання моделі MobileNetV2 та її матрицю кореляції наведено на рисунку 4.8-4.9.

```
Weights file not found. Start training...
Epoch 1/5
2025-09-19 02:40:28.099142: I tensorflow/core/kernels/data/tf_record_dataset_op.cc:387] The default buffer size is 262144
115/115 ----- 150s 1s/step - accuracy: 0.2251 - loss: 3.0677 - val_accuracy: 0.6830 - val_loss: 1.1063
Epoch 2/5
115/115 ----- 118s 1s/step - accuracy: 0.8551 - loss: 0.6455 - val_accuracy: 0.7555 - val_loss: 0.8016
Epoch 3/5
115/115 ----- 115s 1s/step - accuracy: 0.9746 - loss: 0.1855 - val_accuracy: 0.7896 - val_loss: 0.6765
Epoch 4/5
115/115 ----- 113s 987ms/step - accuracy: 0.9975 - loss: 0.0628 - val_accuracy: 0.7991 - val_loss: 0.6634
Epoch 5/5
115/115 ----- 116s 1s/step - accuracy: 1.0000 - loss: 0.0296 - val_accuracy: 0.8095 - val_loss: 0.6324
```

Рисунок 4.8 – Ефективність моделі MobileNetV2

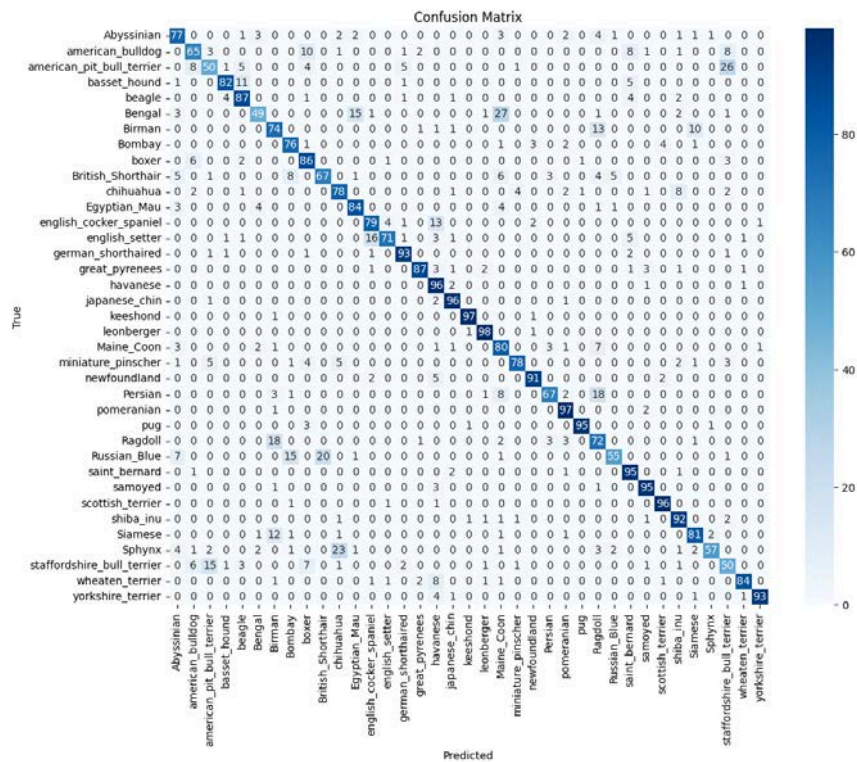


Рисунок 4.9 – Матриця кореляції MobileNetV2

Ефективність розпізнавання моделі CNN наведено на рисунку 4.10-4.11

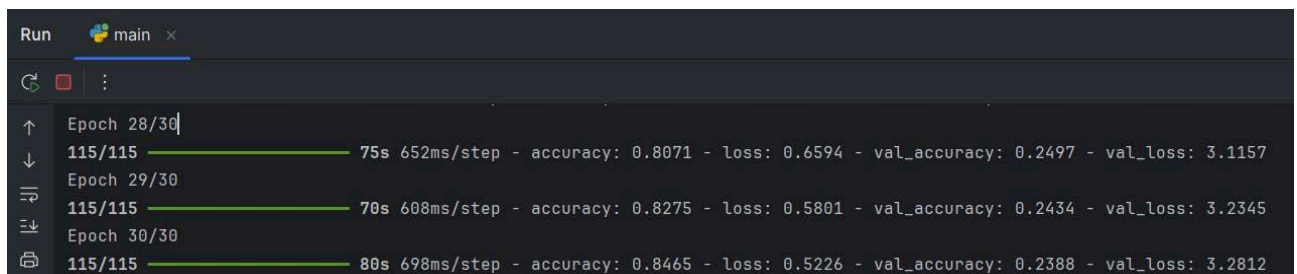


Рисунок 4.10 – Ефективність моделі CNN

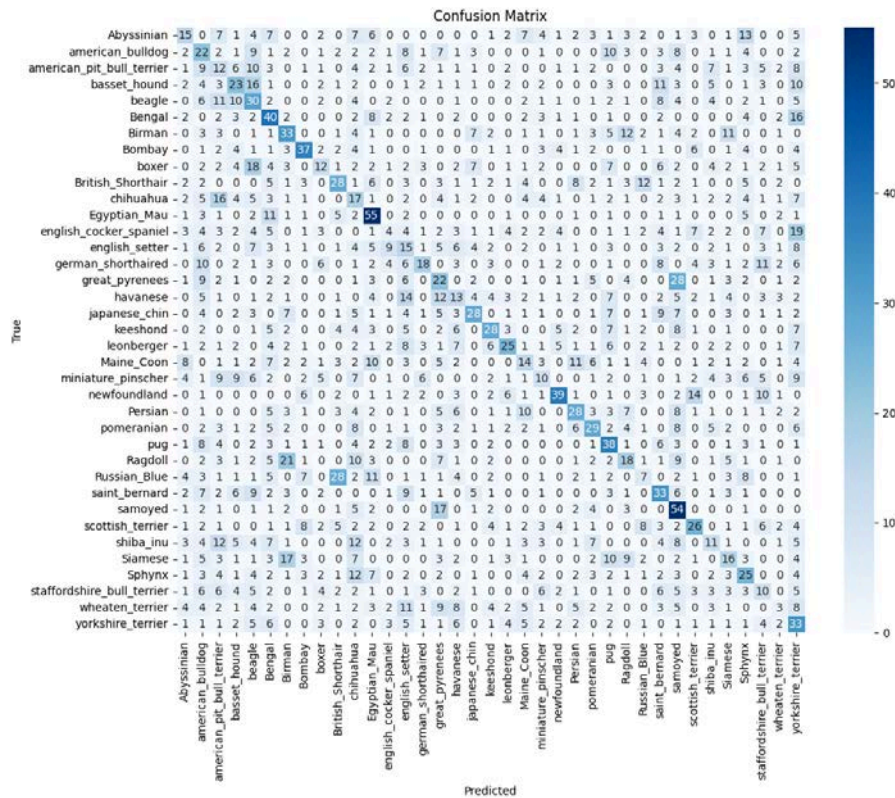


Рисунок 4.11 – Матриця кореляції CNN

Ефективність розпізнавання моделі ResNet50 та її матрицю кореляції наведено на рисунку 4.12-4.13

```

UK
Weights not found, training ResNet50...
Epoch 1/5
115/115 ————— 410s 4s/step - accuracy: 0.0550 - loss: 3.9737 - val_accuracy: 0.2935 - val_loss: 2.6249
Epoch 2/5
115/115 ————— 428s 4s/step - accuracy: 0.2941 - loss: 2.5084 - val_accuracy: 0.5691 - val_loss: 1.7822
Epoch 3/5
115/115 ————— 431s 4s/step - accuracy: 0.5460 - loss: 1.7148 - val_accuracy: 0.6996 - val_loss: 1.3184
Epoch 4/5
115/115 ————— 425s 4s/step - accuracy: 0.6508 - loss: 1.2793 - val_accuracy: 0.7626 - val_loss: 1.0540
Epoch 5/5
115/115 ————— 436s 4s/step - accuracy: 0.7413 - loss: 1.0016 - val_accuracy: 0.7975 - val_loss: 0.8889

```

Рисунок 4.12 – Ефективність моделі ResNet50

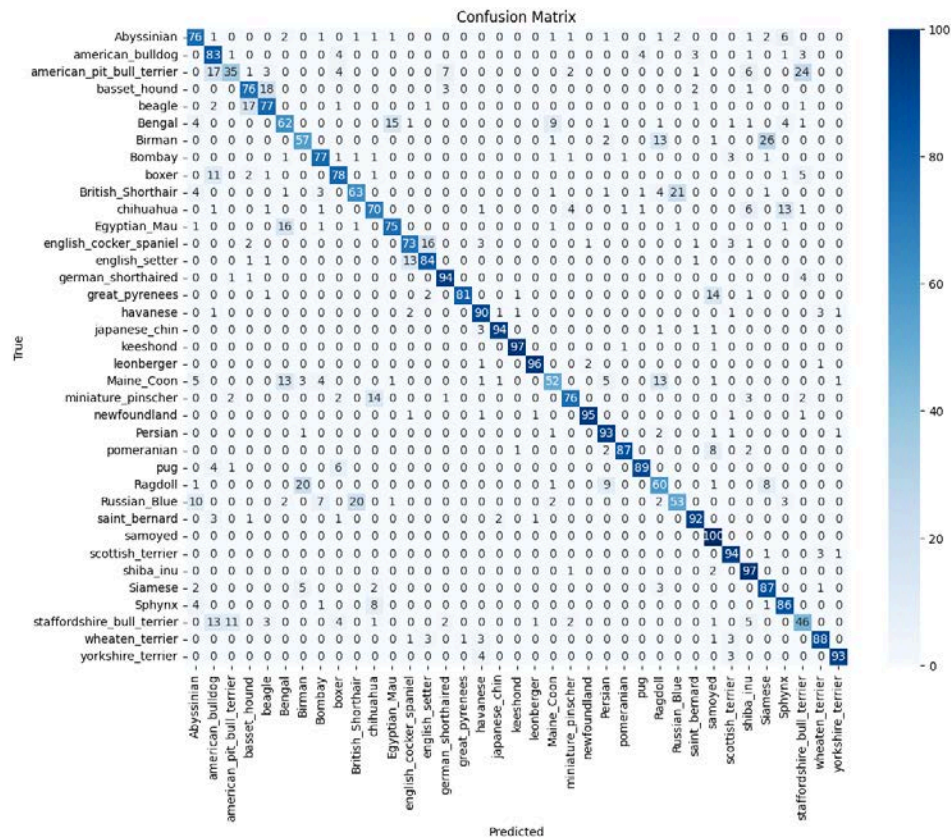


Рисунок 4.13 – Матриця кореляції ResNet50

Ефективність розпізнавання моделі EfficientNetB0 та її матрицю кореляції наведено на рисунку 4.14-4.15

```
EfficientNetB0
uk
Weights not found, training EfficientNetB0...
Epoch 1/5
115/115 ————— 174s 1s/step - accuracy: 0.0702 - loss: 3.5179 - val_accuracy: 0.4086 - val_loss: 2.8655
Epoch 2/5
115/115 ————— 158s 1s/step - accuracy: 0.3874 - loss: 2.7345 - val_accuracy: 0.6838 - val_loss: 2.2158
Epoch 3/5
115/115 ————— 154s 1s/step - accuracy: 0.6513 - loss: 2.1224 - val_accuracy: 0.7901 - val_loss: 1.7278
Epoch 4/5
115/115 ————— 157s 1s/step - accuracy: 0.7773 - loss: 1.6632 - val_accuracy: 0.8378 - val_loss: 1.3772
Epoch 5/5
115/115 ————— 159s 1s/step - accuracy: 0.8178 - loss: 1.3406 - val_accuracy: 0.8618 - val_loss: 1.1306
```

Рисунок 4.14 – Ефективність моделі EfficientNetB0

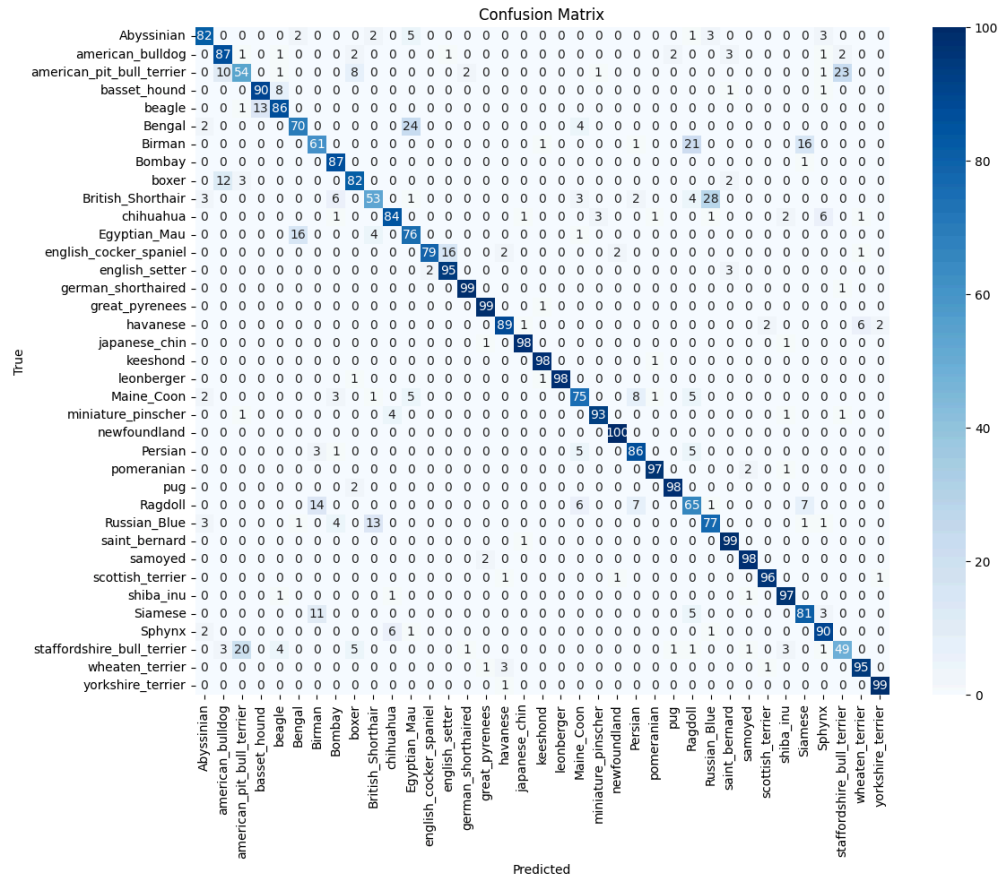


Рисунок 4.15 – Матриця кореляції EfficientNetB0

Ефективність розпізнавання моделі DenseNet121 та її матрицю кореляції наведено наведено н рисунку 4.16-4.17

```
Weights not found, training DenseNet121...
Epoch 1/5
2025-09-19 03:54:04.319213: I tensorflow/core/kernels/data/tf_record_dataset_op.cc:387] The default buffer size is 262
115/115 ----- 421s 4s/step - accuracy: 0.0299 - loss: 4.3580 - val_accuracy: 0.1101 - val_loss: 3.3612
Epoch 2/5
115/115 ----- 394s 3s/step - accuracy: 0.1074 - loss: 3.4701 - val_accuracy: 0.2769 - val_loss: 2.7536
Epoch 3/5
115/115 ----- 417s 4s/step - accuracy: 0.2090 - loss: 2.8868 - val_accuracy: 0.4271 - val_loss: 2.2716
Epoch 4/5
115/115 ----- 425s 4s/step - accuracy: 0.3471 - loss: 2.3615 - val_accuracy: 0.5604 - val_loss: 1.8943
Epoch 5/5
115/115 ----- 411s 4s/step - accuracy: 0.4462 - loss: 2.0163 - val_accuracy: 0.6533 - val_loss: 1.6026
```

Рисунок 4.16 – Ефективність моделі DenseNet121

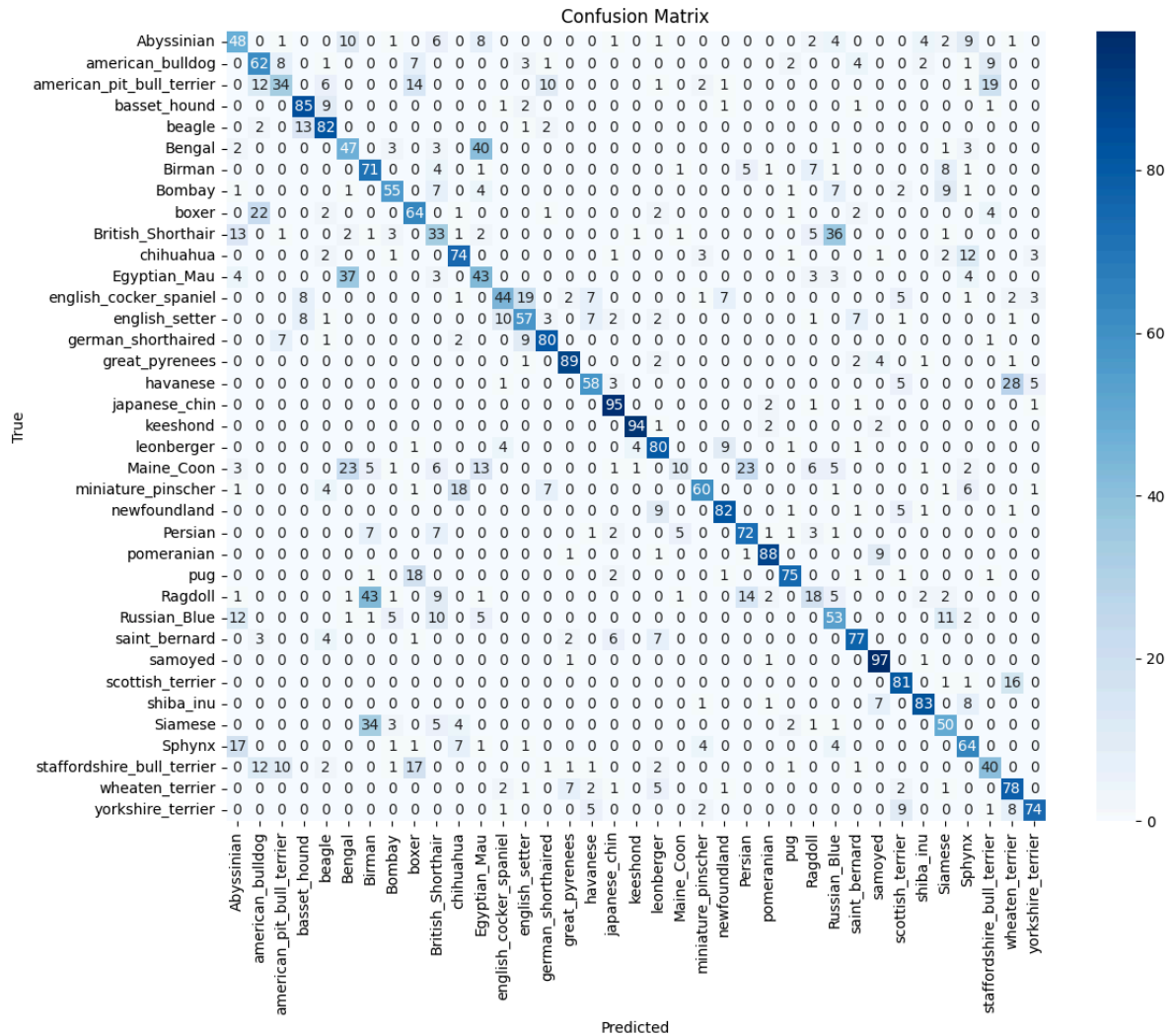


Рисунок 4.17 – Матриця кореляції DenseNet121

В таблиці 4.1 наведено порівняльну характеристику використаних моделей.

Таблиця 4.1 – Порівняльна характеристика моделей

Характеристика	CNN	MobileNetV2	DenseNet121	ResNet50	EfficientNetB0
Точність	0.2388	0.8095	0.6533	0.7975	0.8618
Час навчання, с	1715	612	2059	2130	802
Вага файлу вагів, Мб	32.731	28.711	29.023	93.505	16.984

Як можна побачити, власноруч розроблена модель поступається в своїй ефективності існуючим аналогам. Здебільшого це пояснюється тим, що дана модель не мала попередньо завантаженого набору вагів imagenet, який використовували вже існуючі моделі.

Слід зазначити, що моделі DenseNet121 та ResNet50 мають найбільший час навчання, при цьому показник точності залишається на рівні середнього.

Серед моделей застосованих в роботі найбільшу ефективність показала EfficientNetB0, оскільки вона має найбільший показник точності серед наведених моделей.

4.6 Тестування програмного забезпечення

Виконано тестування мобільного застосунку для ідентифікації порід тварин.

Виявлено, що мобільний застосунок для ідентифікації порід тварин функціонує правильно та злагоджено. Він реалізує всі функціональні вимоги та успішно виконує свою основну задачу ідентифікації порід тварин за зображенням.

Розроблений мобільний застосунок дозволяє забезпечувати спрощення та автоматизацію процесів, пов'язаних з ідентифікацією порід тварин.

4.7 Висновки за розділом 4

Описано мобільний застосунок для ідентифікації порід тварин. Виконано тестування розробленого мобільного застосунку для ідентифікації порід тварин. Результати тестування мобільного застосунку показали, що розроблена програма дозволяє забезпечити спрощення та автоматизацію процесів, пов'язаних з ідентифікацією порід тварин.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки мобільного застосунку для ідентифікації породи тварини за зображенням.

Визначено, що задача ідентифікації порід належить до галузі комп'ютерного зору і полягає у використанні алгоритмів машинного та глибинного навчання для аналізу візуальних ознак зображення з метою віднесення об'єкта до певного класу. Для цього застосовуються згорткові нейронні мережі та сучасні архітектури, такі як MobileNetV2, ResNet50, EfficientNetB0, DenseNet121, які забезпечують високу точність класифікації при збереженні продуктивності. Основною метою створення мобільного застосунку є автоматизація процесу визначення породи тварини та забезпечення швидкої й зручної взаємодії користувача із системою.

За результатами проведеного дослідження зроблено висновок, що на сьогодні існує значна кількість програмних засобів для класифікації зображень. Проте багато з них вимагають потужних обчислювальних ресурсів, що ускладнює їх застосування на мобільних пристроях. Крім того, частина систем має складні інтерфейси, які потребують додаткових знань від користувача. Тому актуальною є розробка мобільного застосунку для ідентифікації порід тварин, який поєднує точність, простоту використання та оптимальність у роботі з ресурсами.

Сформульовано функціональні вимоги до програмного забезпечення для ідентифікації породи тварини за зображенням.

Для реалізації мобільного застосунку було обрано мову програмування Python, яка має широкий набір бібліотек для роботи з нейронними мережами та обробки зображень (TensorFlow, Keras, OpenCV). Простий та зрозумілий синтаксис мови дозволив швидко реалізувати та протестувати кілька моделей, а також провести їх оптимізацію.

Для створення мобільного застосунку використано середовище розробки Android Studio у поєднанні з інструментами для інтеграції навченої моделі у мобільне середовище. Це дало можливість забезпечити запуск класифікатора безпосередньо на смартфоні, а також створити зручний графічний інтерфейс для користувача.

Новизна роботи полягає у створенні мобільного застосунку, що поєднує сучасні архітектури глибинного навчання з оптимізаційними підходами для роботи на пристроях із обмеженими ресурсами.

Практична цінність роботи полягає у тому, що розроблено програмне забезпечення для ідентифікації порід тварин за зображенням.

Запропоновано структуру мобільного застосунку для ідентифікації породи тварини, яка базується на клієнт–серверній архітектурі. Клієнтська частина включає головний модуль MainActivity, клас об'єкта Breed, модуль адаптеру списку порід BreedListAdapter, модуль списку порід BreedList та модуль даних BreedData, що забезпечує збереження та відображення інформації про породи. Серверна частина складається з головного модуля та класів моделей нейронних мереж, що реалізують процес класифікації зображень. Такий поділ дозволяє забезпечити ефективну взаємодію користувача із застосунком, гнучкість у модифікації моделей та масштабованість системи.

Описано функціонування програмного забезпечення для ідентифікації порід тварин, що передбачає завантаження зображення, його попередню обробку, класифікацію за допомогою нейромережевої моделі та відображення результатів у зручному вигляді.

Виконано проєктування інтерфейсу мобільного застосунку та його адаптацію для кінцевого користувача.

Виконано тестування розробленого мобільного застосунку для ідентифікації порід тварин. Результати тестування показали, що застосунок дозволяє забезпечити автоматизацію процесу класифікації зображень та може бути успішно застосований у практичних сценаріях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dog Breed Identifier With AI [Electronic resource]. – Access mode: https://play.google.com/store/apps/details?hl=en_US&id=com.famoconnect.dog.breed.identifier.scanner.mixed.breeds.
2. Snap-Dog - Cornell Tech [Electronic resource]. – Access mode: <https://tech.cornell.edu/built/snap-dog/>.
3. Dog Face Recognition and Identification for a Mobile App [Electronic resource]. – Access mode: <https://indatalabs.com/resources/dog-face-recognition>.
4. AI Dog Breed Identifier [Electronic resource]. – Access mode: <https://aidogbreedidentifier.com/>.
5. Petidentityapp: A Mobile Application With Emergency Rescue Features [Electronic resource]. – Access mode: <https://dl.acm.org/doi/abs/10.1145/3639233.3639325>.
6. Google Об'єктив - шукайте інформацію про речі навколо [Електронний ресурс]. – Режим доступу: <https://lens.google/intl/uk/>.
7. Urgent Need for Pet Adoption - Find Dogs & Cats & More | PetFinder [Electronic resource]. – Access mode: <https://www.petfinder.com/>.
8. Pawscout [Electronic resource]. – Access mode: <https://pawscout.com/>.
9. Programming languages in Android development [Electronic resource]. – Access mode: <https://developer.android.com/courses/pathways/android-basics-compose-unit-1-pathway-1>.
10. Java for Android Development [Electronic resource]. – Access mode: <https://developer.android.com/guide/platform>.
11. Kotlin and Android | Android Developers [Electronic resource]. – Access mode: <https://developer.android.com/kotlin>.
12. C++ and Android NDK | Android Developers [Electronic resource]. – Access mode: <https://developer.android.com/ndk>.

13. Python Mobile App Development: Key Benefits, How to Get Started [Electronic resource]. – Access mode: <https://gpsolutions.com/blog/custom-development/python-for-mobile-development/>.
14. Review: 7 Python IDEs compared [Electronic resource]. – Access mode: <https://www.infoworld.com/article/2250631/review-7-python-ides-compared.html>.
15. Get Started Tutorial for Python in Visual Studio Code [Electronic resource]. – Access mode: <https://code.visualstudio.com/docs/python/python-tutorial>.
16. PyCharm: the Python IDE for Professional Developers [Electronic resource]. – Access mode: <https://www.jetbrains.com/pycharm/>.
17. IDLE — Python 3.12. Documentation [Electronic resource]. – Access mode: <https://docs.python.org/3/library/idle.html>.
18. 8 Best Android Development IDEs to Know in 2025 [Electronic resource]. – Access mode: <https://www.spaceotechnologies.com/blog/android-development-ides/>.
19. Android Studio | Android Developers [Electronic resource]. – Access mode: <https://developer.android.com/studio>.
20. Set up an editor | Flutter documentation [Electronic resource]. – Access mode: <https://docs.flutter.dev/get-started/editor?tab=vscode>.
21. IntelliJ IDEA: The Leading Java and Kotlin IDE [Electronic resource]. – Access mode: <https://www.jetbrains.com/idea/>.
22. Visual Geometry Group - University of Oxford [Electronic resource]. – Access mode: <https://www.robots.ox.ac.uk/%7Evvgg/data/pets/>.
23. Python Libraries for Machine Learning [Electronic resource]. – Access mode: <https://www.simplilearn.com/python-machine-learning-libraries-article>.
24. Scikit-learn: Machine Learning in Python [Electronic resource]. – Access mode: <https://scikit-learn.org/stable/>.
25. PyTorch: An open source machine learning framework [Electronic resource]. – Access mode: <https://pytorch.org/>.
26. TensorFlow: An end-to-end open source platform for machine learning [Electronic resource]. – Access mode: <https://www.tensorflow.org/>.

ДОДАТОК А
Технічне завдання

Вступ

Мобільний застосунок може використовуватися для ідентифікації породи тварини за зображенням.

A.1 Підстава до розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему “Дослідження та програмна реалізація мобільного застосунку для ідентифікації порід тварин”, затверджене наказом Національного університету “Запорізька політехніка” № 447 від 30 вересня 2025 р.

A.2 Призначення розробки

Мобільний застосунок призначений для ідентифікації порід тварин за зображенням.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Мобільний застосунок для ідентифікації порід тварин забезпечує такі функціональні вимоги:

- підтримка можливості завантаження зображення тварини з камери або галереї;
- можливість попередньої обробки зображення для підготовки до класифікації;
- можливість передачі зображення на сервер для обробки моделлю нейронної мережі;
- підтримка класифікації породи тварини за допомогою алгоритмів машинного навчання;

- підтримка повернення результатів класифікації з рівнем достовірності;
- можливість відображення користувачу опису та характеристик визначеної породи;
- підтримка перегляду списку доступних порід;
- можливість завантаження зображення через URL-посилання;
- можливість змінити мову програму на українську та англійську.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс мобільного застосунку для ідентифікації порід тварин повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі

A.3.3 Вимоги до надійності

Мобільний застосунок для ідентифікації порід тварин повинен забезпечувати надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації мобільного застосунку для ідентифікації порід тварин необхідна наявність телефону та персонального комп'ютеру.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації мобільного застосунку для ідентифікації порід тварин необхідно таке апаратне та програмне забезпечення:

- смартфон або планшет з операційною системою Android 7.0 або новішою;

– персональний комп’ютер з доступом до мережі інтернет, процесором з тактовою частотою не менше 2.2 ГГц, з встановленим Python, бібліотекою TensorFlow та достатнім обсягом оперативної пам’яті (від 4 ГБ) для обробки зображень та роботи моделей нейронних мереж.

A.3.6 Вимоги до маркування і пакування

Мобільний застосунок для ідентифікації порід тварин може бути записаний на будь-якому носії інформації. На пакуванні повинна бути назва програми – «Мобільний застосунок для ідентифікації порід тварин».

A.4 Вхідні дані до роботи

Вхідними даними до мобільного застосунку для ідентифікації порід тварин є інформація про вибірку даних для навчання моделей та дані зображень, які входять до неї.

Вихідними даними мобільного застосунку для ідентифікації порід тварин є результат ідентифікації породи та навчання моделей.

ДОДАТОК Б
Фрагмент тексту програми

```

package com.example.client_app;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatActivityDelegate;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Base64;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.AdapterView;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.hivemq.client.mqtt.MqttClient;
import com.hivemq.client.mqtt.datatypes.MqttQos;

import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.Locale;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private final String serverUri = "test.mosquitto.org";
    private String userId;
    private final String topicPhoto = "animal/photo";
    private final String topicResultPrefix = "animal/result/";
    private SensorManager sensorManager;
    private Sensor lightSensor;
    private SensorEventListener lightListener;
    private boolean isDarkTheme = false;
    private Button buttonSend;
    private TextView textResult;
    private ImageView imageView;
    private byte[] selectedImageBytes = null;
    private Spinner list_of_models;
    private String selectedTest = "MobileNetV2";

    private final com.hivemq.client.mqtt.mqtt3.Mqtt3AsyncClient mqttClient =
MqttClient.builder()
        .useMqttVersion3()
        .serverHost(serverUri)
        .serverPort(1883)
        .identifier(UUID.randomUUID().toString())
        .buildAsync();

    private final ActivityResultLauncher<String> pickImageLauncher =

```

```

        registerForActivityResult(new ActivityResultContracts.GetContent(),
uri -> {
            if (uri != null) {
                loadImage(uri);
            }
        });
        private void setLocale(String langCode) {
            Locale locale = new Locale(langCode);
            Locale.setDefault(locale);

            android.content.res.Configuration config = new
android.content.res.Configuration();
            config.setLocale(locale);

            getResources().updateConfiguration(config,
getResources().getDisplayMetrics());

            recreate();
        }
        @Override
        protected void onCreate(Bundle savedInstanceState) {
AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_FOLLOW_SYSTEM);
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            userId = Settings.Secure.getString(
                getContentResolver(),
                Settings.Secure.ANDROID_ID
            );
            buttonSend = findViewById(R.id.button_send);
            textResult = findViewById(R.id.text_result);
            imageView = findViewById(R.id.image_view);
            list_of_models = findViewById(R.id.model_list);
            String systemLang = Locale.getDefault().getLanguage();

            if (systemLang.equals("ru")) {
                setLocale("uk");
            }
            FloatingActionButton buttonLang = findViewById(R.id.button_lang);
            buttonLang.setOnClickListener(view -> {

                String currentLang =
getResources().getConfiguration().getLocales().get(0).getLanguage();

                if (currentLang.equals("ru")) {
                    setLocale("en");
                }
                if (currentLang.equals("uk")) {
                    setLocale("en");
                }
                else {
                    setLocale("uk");
                }
            });

            String[] testOptions = {"MobileNetV2", "CNN", "ResNet50", "EfficientNetB0",
"DenseNet121"};

            ArrayAdapter<String> adapter = new ArrayAdapter<>(
                this,
                android.R.layout.simple_spinner_item,
                testOptions

```

```

    );

    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    list_of_models.setAdapter(adapter);

    list_of_models.setOnItemSelectedListener(new
    AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, android.view.View
        view, int position, long id) {
            selectedTest = parent.getItemAtPosition(position).toString();
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            selectedTest = "MobileNetV2";
        }
    });

    FloatingActionButton buttonBreeds = findViewById(R.id.button_breeds);
    buttonBreeds.setOnClickListener(view -> {
        Intent intent = new Intent(MainActivity.this,
    BreedListActivity.class);
        startActivity(intent);
    });
    FloatingActionButton buttonUrl = findViewById(R.id.button_url);
    buttonUrl.setOnClickListener(view -> {
        android.app.AlertDialog.Builder builder = new
    android.app.AlertDialog.Builder(this);
        builder.setTitle(R.string.paste_url);

        final android.widget.EditText input = new
    android.widget.EditText(this);
        input.setInputType(android.text.InputType.TYPE_CLASS_TEXT |
    android.text.InputType.TYPE_TEXT_VARIATION_URI);
        builder.setView(input);

        builder.setPositiveButton(R.string.load_url, (dialog, which) -> {
            String url = input.getText().toString();
            loadImageFromUrl(url);
        });
        builder.setNegativeButton(R.string.back_url, (dialog, which) ->
    dialog.cancel());

        builder.show();
    });

    connectToMqtt();

    buttonSend.setOnClickListener(view -> {
        pickImageLauncher.launch("image/*");
    });
}

private void connectToMqtt() {
    mqttClient.connect()
        .whenComplete((ack, throwable) -> {

```

```

        if (throwable == null) {
            subscribeForResult();
        } else {
            runOnUiThread(() -> Toast.makeText(this,
R.string.mqtt_connect, Toast.LENGTH_SHORT).show());
        }
    });
}

private void loadImage(Uri uri) {
    try {
        InputStream inputStream = getContentResolver().openInputStream(uri);
        selectedImageBytes = new byte[inputStream.available()];
        inputStream.read(selectedImageBytes);
        imageView.setImageURI(uri);

        sendPhotoRequest();
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(this, R.string.ImageError, Toast.LENGTH_SHORT).show();
    }
}

private void loadImageFromUrl(String urlString) {
    new Thread(() -> {
        try {
            URL url = new URL(urlString);
            InputStream inputStream = url.openStream();
            Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

            runOnUiThread(() -> {
                imageView.setImageBitmap(bitmap);

                ByteArrayOutputStream stream = new ByteArrayOutputStream();
                bitmap.compress(Bitmap.CompressFormat.PNG, 80, stream);
                selectedImageBytes = stream.toByteArray();

                sendPhotoRequest();
            });
        } catch (Exception e) {
            e.printStackTrace();
            runOnUiThread(() -> Toast.makeText(this, R.string.Cantloadurl,
Toast.LENGTH_SHORT).show());
        }
    }).start();
}

private void sendPhotoRequest() {
    if (selectedImageBytes == null) {
        Toast.makeText(this, R.string.ChooseImageFirst,
Toast.LENGTH_SHORT).show();
        return;
    }

    Bitmap bitmap = BitmapFactory.decodeByteArray(selectedImageBytes, 0,
selectedImageBytes.length);
    Bitmap scaledBitmap = Bitmap.createScaledBitmap(bitmap, 800, 600, true);

    ByteArrayOutputStream stream = new ByteArrayOutputStream();
    scaledBitmap.compress(Bitmap.CompressFormat.PNG, 80, stream);

    selectedImageBytes = stream.toByteArray();
}

```

```

        String imageBase64 = Base64.encodeToString(selectedImageBytes,
Base64.NO_WRAP);
        String currentLang = Locale.getDefault().getLanguage();
        if (currentLang=="ru") currentLang="uk";
        String message = userId + "||" + selectedTest + "||" + imageBase64 + "||" +
currentLang;

        runOnUiThread(() -> textResult.setText(R.string.waiting_for_result));
        mqttClient.publishWith()
            .topic(topicPhoto)
            .qos(MqttQos.AT_LEAST_ONCE)
            .payload(message.getBytes(StandardCharsets.UTF_8))
            .send();
    }

    private void subscribeForResult() {
        mqttClient.subscribeWith()
            .topicFilter(topicResultPrefix + userId)
            .qos(MqttQos.AT_LEAST_ONCE)
            .callback(publish -> {
                String result = new String(publish.getPayloadAsBytes(),
StandardCharsets.UTF_8);
                String res_rec = getString(R.string.recognition);
                runOnUiThread(() -> textResult.setText(res_rec + result));
            })
            .send();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (sensorManager != null && lightListener != null) {
            sensorManager.unregisterListener(lightListener);
        }
    }
}

```

ДОДАТОК В
Слайди презентації

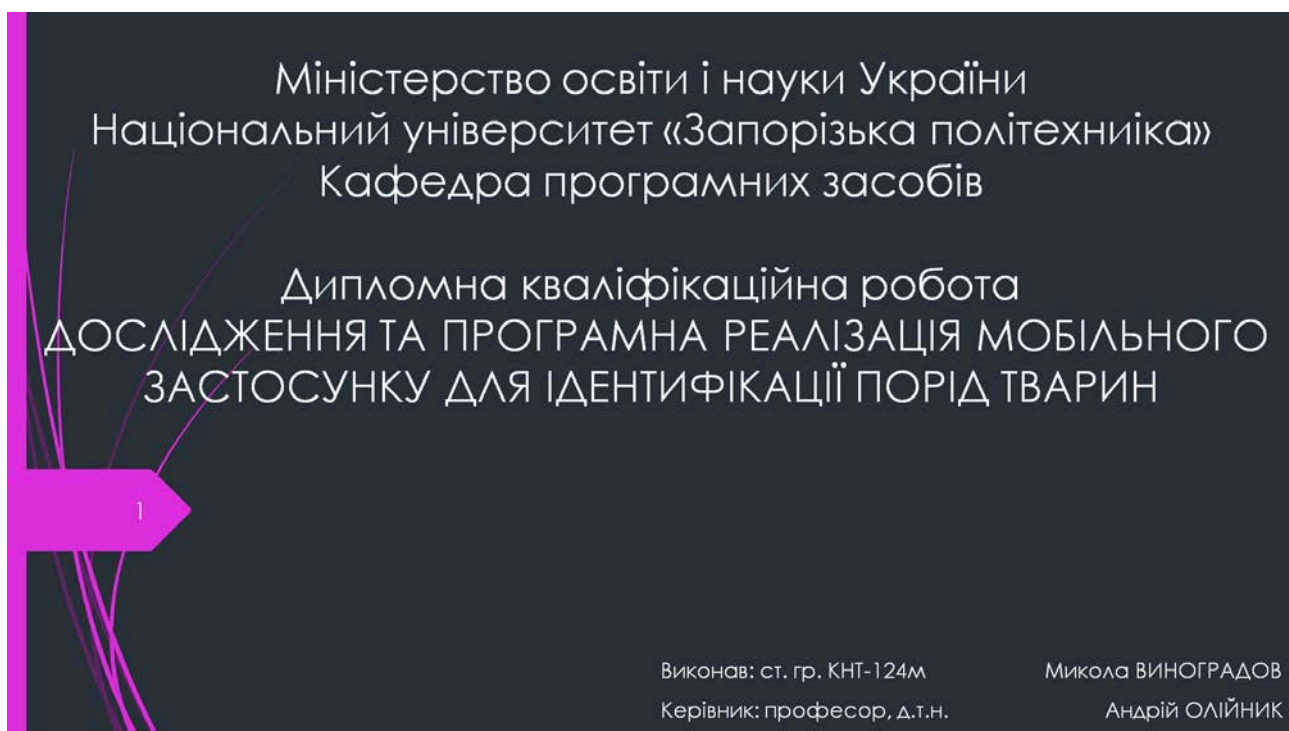


Рисунок В.1 – Слайд 1

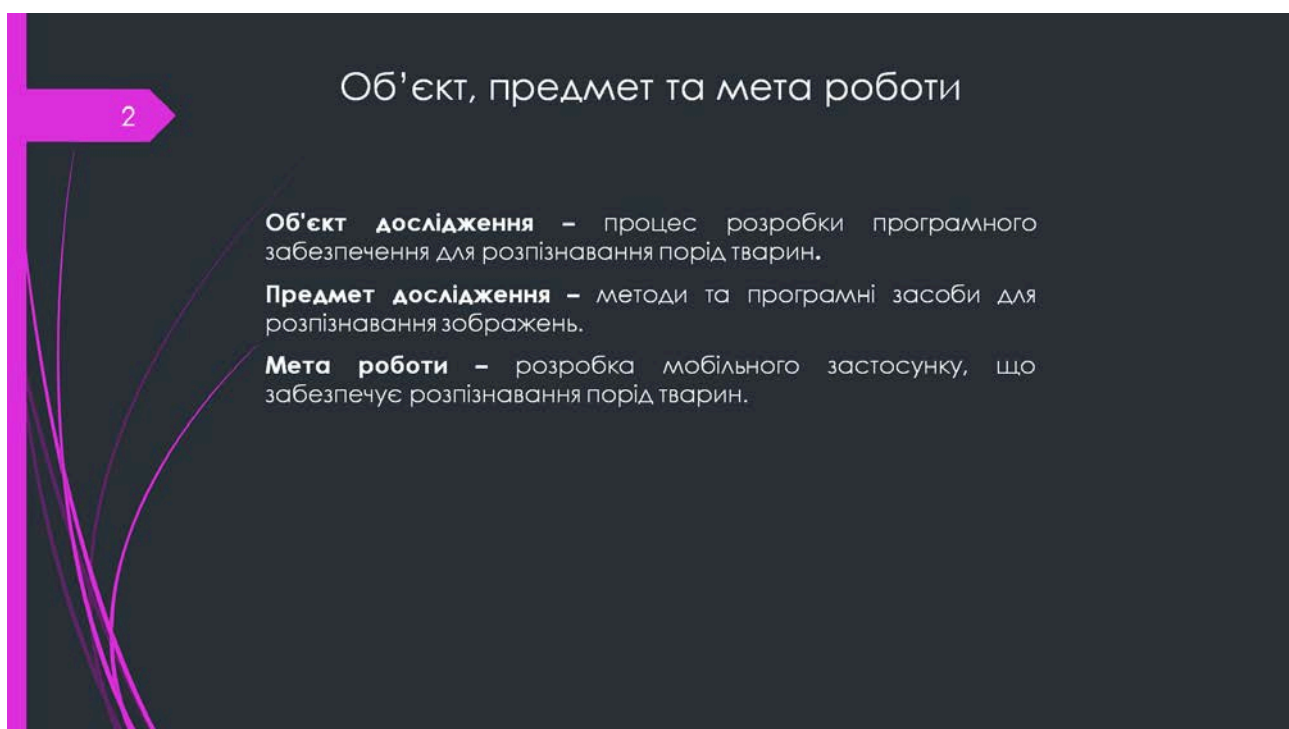


Рисунок В.2 – Слайд 2

Результати аналізу предметної області

3

В результаті аналізу, було виявлено, що програмне забезпечення даного характеру зазвичай має такі функції::

- Завантаження зображення.
- Розпізнавання породи домашнього улюбленця.
 - Навчання моделі розпізнавання.
 - Збереження даних навченої моделі.
- Перегляд списку порід, які можуть бути розпізнані.

Рисунок В.3 – Слайд 3

Аналіз програм аналогів

4

Програми аналоги:

- Google Lens.
- Petfinder.
- Pawscout.

Характеристика	Google Lens	Petfinder	Pawscout
Доступність	+	+	-
Спеціалізація на ідентифікації	+	-	-
Необхідність інтернет-з'єднання	-	-	-
Точність ідентифікації	+	-	-
Перегляд списку порід, які можуть бути розпізнані	-	-	-

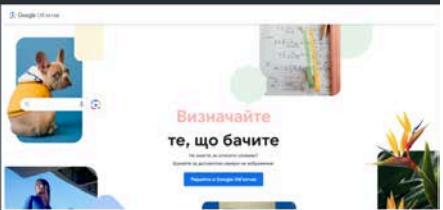
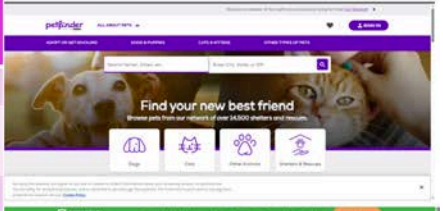




Рисунок В.4 – Слайд 4

Засоби розробки								
5								
1	2	3	4	5	Клієнтська частина застосунку: IDE: Android Studio. Мова програмування: Java.			
Підтримка Android SDK	+	+	Часткова	Часткова	Серверна частина застосунку: IDE: PyCharm. Мова програмування: Python.			
Офіційна підтримка Google	+	+	-	-	Бібліотека машинного навчання: Tensorflow			
Продуктивність	+	+	+	-	Характеристика	Scikit-learn	TensorFlow	PyTorch
Простота вивчення	+	+	-	+	Призначення	Класичні ML-алгоритми	Глибоке навчання	Глибоке навчання, дослідження
Багатопоточність	+	+	+	+	Простота використання	Висока	Середня	Висока
Типізація	Статична	Статична	Статична	Динамічна	Підтримка GPU	Ні	Так	Так
Характеристика	Android Studio	VS Code	IntelliJ IDEA		Гнучкість	Обмежена	Висока	Висока
Емулятор Android	Вбудований	Розширення	Плагіни		Ліцензія	BSD	Apache 2.0	BSD
Автодоповнення коду	+	+	+		Характеристика	PyCharm	VS Code	IDLE
Дебагінг	Вбудований	Розширення	Вбудований		Автодоповнення коду	Наявність ШІ асистента	В залежності від розширень	Мінімальне
Інтеграція з Git	Вбудована	Розширення	Вбудована		Дебагінг	Так	Так	Мінімальний
Продуктивність	Середня	Висока	Середня		Інструменти для аналізу коду	Вбудовані	Через розширення	Відсутні
UI-дизайнер	Вбудований	Відсутній	Лише Ultimate	в	Швидкість роботи	Середня	Висока	Висока
					Підтримка Git	Вбудовано	Через розширення	Ні

Рисунок В.5 – Слайд 5

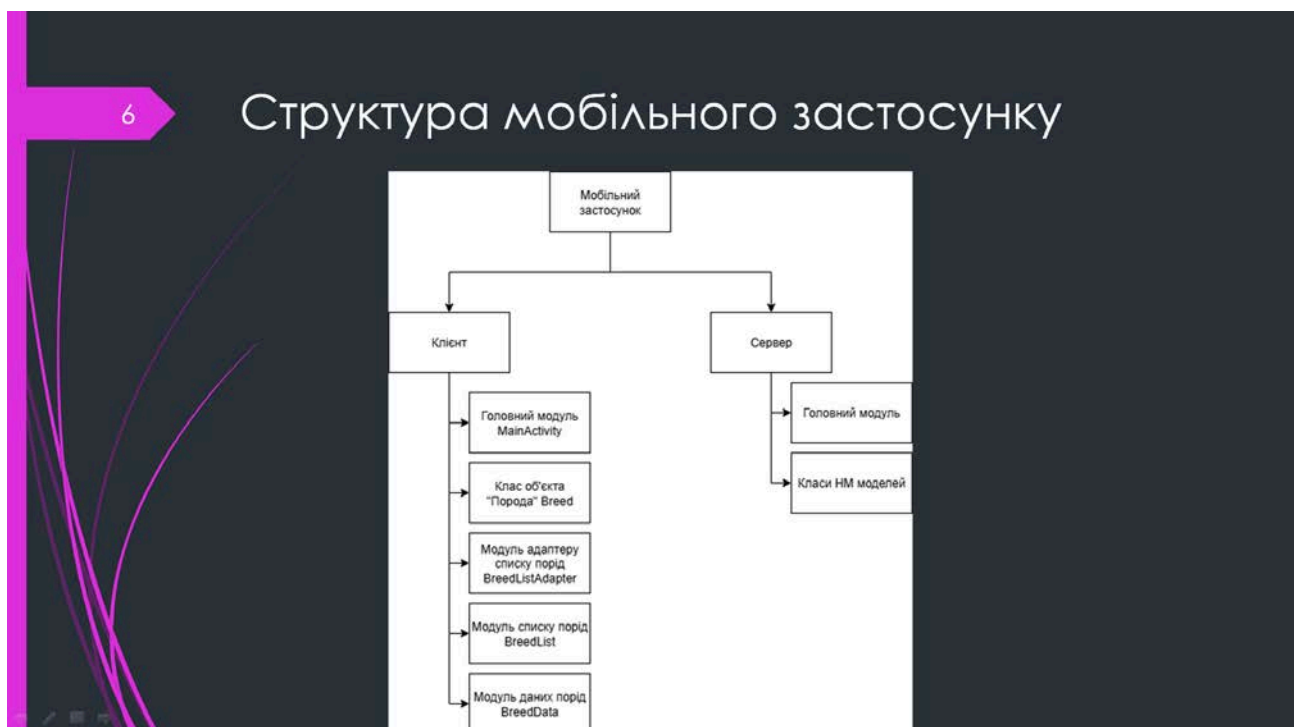


Рисунок В.6 – Слайд 6

Функціональна схема клієнтської частини

7

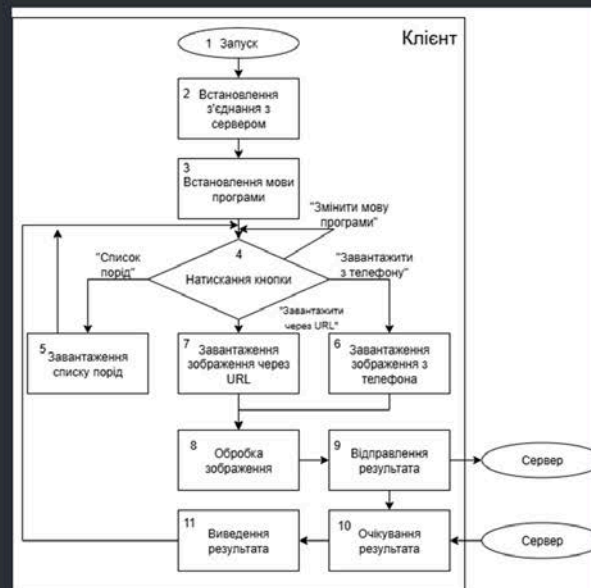


Рисунок В.7 – Слайд 7

Функціональна схема серверної частини

8

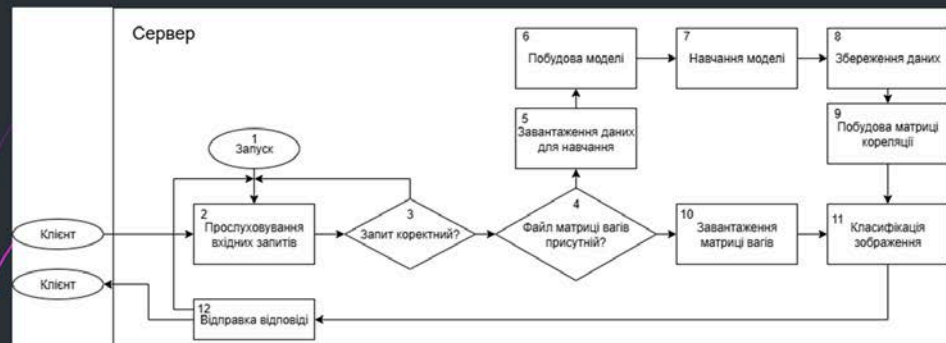


Рисунок В.8 – Слайд 8

Графічний інтерфейс мобільного застосунку

9



Мобільний застосунок складається з двох графічних форм: розпізнавання зображення та список порід, які можуть бути розпізнані

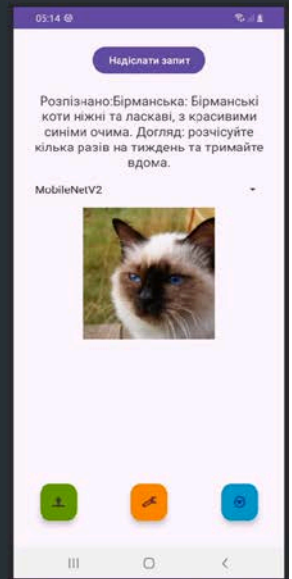
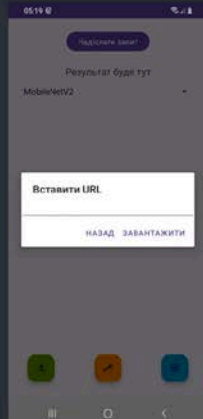


Рисунок В.9 – Слайд 9

Архітектура моделей

10

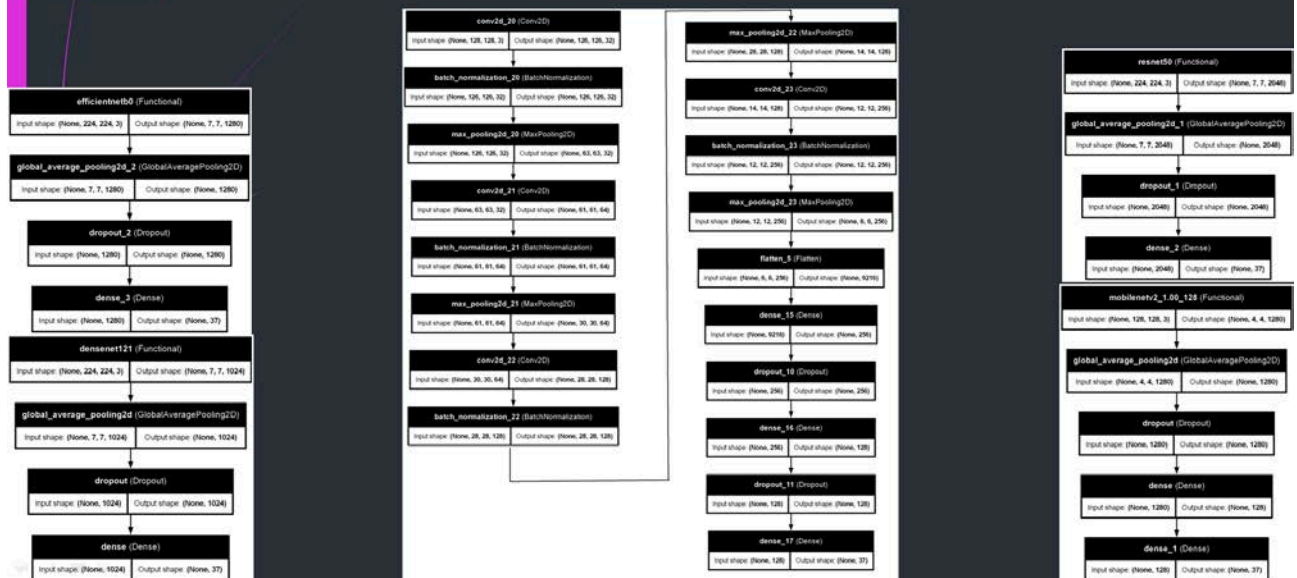


Рисунок В.10 – Слайд 10

Результати тестування моделей для ідентифікації порід тварин

11

Під час тестування було виявлено, що найточніша модель із застосованих – EfficientNetB0.

Характеристика	CNN	MobileNetV2	DenseNet121	ResNet50	EfficientNetB0
Точність	0.2388	0.8095	0.6533	0.7975	0.8618
Час навчання, с	1715	612	2059	2130	802
Вага файлу вагів, Мб	32.731	28.711	29.023	93.505	16.984

Рисунок В.11 – Слайд 11

12

Висновки

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки мобільного застосунку для ідентифікації порід тварин.

Розроблена система дозволяє покращити взаємодію між власниками тварин та професіоналами, а також оптимізує роботу спеціалістів.

Галузь використання: ветеринарна медицина, зоомагазини, кінологічні організації, притулки для тварин.

Рисунок В.12 – Слайд 12