Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ АРХІТЕКТУРНИХ ПАМ'ЯТОК УКРАЇНИ НА ЗОБРАЖЕННЯХ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1
Тютюнник Я. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І. С.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____Тютюннику Ярославу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення мобільного застосунку для розпізнавання архітектурних пам'яток України на зображеннях

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 23 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека машинного навчання з відкритим кодом TensorFlow.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих мобільних застосунків для розпізнавання об'єктів на зображеннях.
2. Особливості розроблення мобільного застосунку для розпізнавання архітектурних пам'яток України на зображеннях.
3. Етапи розроблення мобільного застосунку для розпізнавання архітектурних пам'яток України на зображеннях.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розробки зображень архітектурних пам'яток України, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-13.04.25	
3	Аналіз літератури з досліджуваної проблеми	14.04.25-17.04.25	
4	Аналіз технічних засобів	18.04.25-24.04.25	
5	Розробка методу	25.04.25-30.04.25	
6	Програмна реалізація	01.05.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Творошенко І. С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 76 с., 23 рис., 36 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, АРХІТЕКТУРНІ ПАМ'ЯТКИ УКРАЇНИ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ, TENSORFLOW LITE, ANDROID STUDIO.

Об'єктом роботи є процес розпізнавання архітектурних пам'яток України на фотографіях за допомогою мобільного застосунку.

Метою роботи є розроблення мобільного застосунку, який здійснює розпізнавання архітектурних пам'яток України на зображеннях з використанням згорткових нейронних мереж.

У роботі використано методи машинного навчання, комп'ютерного зору та розробки мобільних застосунків. Для реалізації задачі класифікації зображень було створено та навчено модель згорткової нейронної мережі, що здатна ідентифікувати 18 класів архітектурних пам'яток України. У результаті виконаної роботи створено мобільний застосунок, що дозволяє користувачеві розпізнавати архітектурні пам'ятки України за допомогою камери смартфона. Проведено тестування точності моделі, що показало прийнятні результати для практичного використання застосунку.

Розроблений застосунок може бути використаний у сфері туризму, освіти та для культурно-просвітницьких цілей.

MOBILE APPLICATION, IMAGE RECOGNITION, ARCHITECTURAL MONUMENTS OF UKRAINE, CONVULSIVE NEURAL NETWORKS, ARTIFICIAL INTELLIGENCE, TENSORFLOW LITE, ANDROID STUDIO.

The object of the work is the process of recognizing architectural monuments of Ukraine in photographs using a mobile application.

The purpose of the work is to develop a mobile application that recognizes architectural monuments of Ukraine in images using convolutional neural networks.

The work uses methods of machine learning, computer vision and mobile application development. To implement the image classification task, a convolutional neural network model was created and trained, which is capable of identifying 18 classes of architectural monuments of Ukraine. As a result of the work performed, a mobile application was created that allows the user to recognize architectural monuments of Ukraine using a smartphone camera. The accuracy of the model was tested, which showed acceptable results for the practical use of the application.

The developed application can be used in the field of tourism, education, and for cultural and educational purposes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Network (згорткова нейронна мережа)

VIP – Visually Impaired Persons (люди з вадами зору)

ML – Machine Learning (машинне навчання)

ООН – Організація Об’єднаних Націй

FAO – Food and Agriculture Organization (продовольча та сільськогосподарська організація ООН)

NLP – Natural Language Processing (обробка природної мови)

SSD – Single Shot Multi-Box Detector (одиночний детектор Multibox)

BP – Back Propagation (метод зворотного поширення помилки)

YOLO – You Only Look Once (ти дивишся тільки один раз)

HOG – Histogram of Oriented Gradients (гістограма напрямлених градієнтів)

SIFT – Scale-Invariant Feature Transform (масштабоінваріантне ознакове перетворення)

SURF – Speeded Up Robust Features (прискорені стійкі ознаки)

DPM – Deformable Part Models (моделі деформованих частин)

SIFT – Scale-Invariant Feature Transform (масштабоінваріантне ознакове перетворення)

SURF – Speeded Up Robust Features (прискорені стійкі ознаки)

ONNX – Open Neural Network Exchange (відкрита бібліотека програмного забезпечення для побудови нейронних мереж глибокого навчання)

GPU – Graphics Processing Unit (графічний процесор)

TPU – Tensor Processing Unit (тензорний блок обробки)

NPU – Neural Processing Unit (нейронний процесор)

ResNet – Residual Neural Network (залишкова нейронна мережа)

FPN – Feature Pyramid Network (мережа пірамідальних ознак)

ВСТУП

У XXI столітті спостерігається стрімкий розвиток технологій штучного інтелекту: методів машинного і глибокого навчання. Досягнення відкривають нові можливості у сфері обробки та інтерпретації зображень, що є важливою складовою численних галузей – від медицини до цифрової гуманітаристики. Одним із перспективних напрямів є автоматичне розпізнавання об’єктів культурної спадщини, зокрема архітектурних пам’яток, на основі фотографій, зроблених за допомогою мобільних пристроїв.

Україна має багатий архітектурний спадок, який є надзвичайно важливим для формування національної ідентичності, розвитку туризму та збереження історичної пам’яті. Проте велика кількість об’єктів лишається маловідомою широкому загалу, а доступ до актуальної інформації про них обмежений. У цьому контексті актуальним є створення інструментів, що можуть полегшити ідентифікацію таких пам’яток за допомогою сучасних мобільних технологій.

Одним із найефективніших підходів до задачі класифікації зображень є використання згорткових нейронних мереж, які здатні самостійно навчатися виділяти важливі ознаки об’єктів на фото. Завдяки цьому такі моделі можуть бути адаптовані для розпізнавання архітектурних елементів, характерних для історичних будівель, церков, фортець, палаців тощо.

Актуальність роботи полягає у зростаючому інтересі до збереження та популяризації культурної спадщини України, зокрема архітектурних пам’яток. У сучасному світі мобільні пристрої стали невід’ємною частиною повсякденного життя, а штучний інтелект – потужним інструментом для автоматизації рутинних задач. Поєднання цих складових дозволяє створити інноваційний інструмент для ідентифікації об’єктів архітектури за фотографіями в режимі реального часу.

1 АНАЛІЗ ІСНУЮЧИХ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ

1.1 Аналіз сучасних мобільних застосунків для розпізнавання об'єктів на зображеннях в Україні та за кордоном

Google Об'єктив – технологія розпізнавання зображень, розроблена компанією Google, покликана збирати відповідну інформацію, що стосується об'єктів, які вона ідентифікує за допомогою візуального аналізу на основі нейронної мережі (рис. 1.1) [1].

Рисунок 1.1 – Розпізнавання штрихкодів за допомогою застосунку Google Об'єктив

Особливості застосунку Google Об’єктив.

Направляючи камеру телефону на об’єкт, Google Об’єктив намагатиметься ідентифікувати об’єкт, читаючи штрих-коди, QR-коди, мітки та текст, а також показувати відповідні результати пошуку, вебсторінки та інформацію. Наприклад, при вказівці камери пристрою на мітку Wi-Fi, що містить ім’я мережі та пароль, він автоматично підключиться до сканованого джерела Wi-Fi. Об’єктив також інтегрований із програмами Google Фото та Google Assistant.

Об’єктив Google – зручний інструмент, щоб пізнавати світ навколо за допомогою камери або фото.

Функції застосунку Google Об’єктив:

– сканування і переклад текстів. Застосунок надає можливість перекладати незнайомі слова, зберігати візитні картки одразу в контакти, додавати заходи з плакатів та копіювати складні коди або довгі абзаци в пам’ять смартфону – і все це швидко і легко;

– отримання назв рослин і тварин. Інструментальний засіб надає можливість досліджувати флору і фауну навколо, наприклад, дізнатись назву привабливої квітки на клумбі або породу собаки, яку побачили в парку;

– дослідження місця навколо нас. Програма дозволяє дізнаватися цікаві факти про пам’ятки, знаходити рейтинги та години роботи закладів, а також інші відомості про навколишні об’єкти;

– пошук привабливих речей. При наведенні камери на певну річ інтер’єру чи одяг, застосунок видає фото подібних речей та URL-адреси вебсайтів інтернет-магазинів, де можна придбати дані речі;

– отримання рекомендацій за відгуками людей. Застосунок надає можливість дізнаватись про популярні страви в меню будь-якого ресторану на основі відгуків у Google Картах;

– сканування кодів. Інструментальний засіб може легко та швидко сканувати QR-коди та штрих-коди. Для сканування кодів необхідно навести камеру смартфону на QR-код чи штрих код.

Переваги застосунку Google Об’єктив:

– розпізнавання тексту та його копіювання. Google Об’єктив дозволяє сканувати текст з будь-якого зображення чи документа та конвертувати його у цифровий формат;

– переклад тексту в реальному часі. Застосунок дозволяє миттєво перекладати текст із понад 100 мов, накладаючи переклад на оригінальне зображення. Це дуже зручно під час подорожей або роботи з іноземними текстами;

– пошук за зображенням. Можливість знаходити інформацію про предмети, пам’ятки, книги, рослини, тварин тощо. Наприклад, якщо ви бачите невідомий об’єкт або витвір мистецтва, Google Об’єктив допоможе знайти детальну інформацію про нього;

– допомога у шопінгу. Застосунок дозволяє знаходити товари в інтернет-магазинах за їхнім зображенням, порівнювати ціни та знаходити схожі моделі. Це особливо корисно для тих, хто шукає найкращі пропозиції онлайн;

– сканування QR-кодів та штрихкодів. Вбудована функція розпізнавання QR-кодів дозволяє швидко відкривати сайти, переглядати інформацію про продукти та використовувати знижкові купони;

– взаємодія з рукописним текстом. Застосунок надає можливість розпізнавати навіть рукописний текст і копіювати його для подальшого використання;

– безкоштовний доступ та інтеграція з іншими сервісами Google. Google Об’єктив доступний безкоштовно і легко інтегрується з Google Перекладачем, Google Фото та Google Chrome, що робить його ще більш універсальним.

Недоліки застосунку Google Об’єктив:

– залежність від інтернету. Деякі функції (наприклад, пошук за зображенням) потребують підключення до інтернету, що може бути незручно у місцях без стабільного інтернет-з’єднання;

– помилки у розпізнаванні тексту. Хоча технологія розпізнавання тексту дуже потужна, вона може давати неточності при скануванні нерозбірливого або декоративного шрифту, рукописних записів чи тексту на складному фоні;

– обмеження у пошуку предметів. Іноді Google Об’єктив не може точно визначити рідкісні або унікальні предмети та видає схожі, але не ідентичні результати;

– витрата заряду батареї. Оскільки застосунок використовує камеру та штучний інтелект у реальному часі, це може призводити до швидшого розряджання акумулятора смартфону;

– питання конфіденційності. Оскільки Google Об’єктив обробляє зображення через сервери Google, це може викликати занепокоєння щодо збереження особистих даних, особливо якщо використовується розпізнавання документів або особистої інформації;

– не завжди точний переклад. Автоматичний переклад інколи може бути неточним або змінювати зміст тексту, особливо при роботі з мовами, що мають складну граматику чи ідіоматичні вирази.

Seeing AI – це безкоштовна програма, яка описує голосом навколишній світ. Це дослідницький проєкт, розроблений для незрячих людей і людей зі слабким зором за їх участю, який використовує можливості штучного інтелекту для відкриття цим людям візуального світу шляхом опису людей, тексту та об’єктів, що знаходяться поблизу (рис. 1.2) [2].

Рисунок 1.2 – Інтерфейс мобільного застосунку Seeing AI

Seeing AI містить засоби, які допомагають справлятися з різними повсякденними завданнями:

– озвучування короткого тексту. Застосунок вимовляє текст, коли він з’являється перед камерою;

– допомога в скануванні документів. Інструментальний засіб надає аудіоінструкції для зйомки друкованої сторінки та розпізнає текст з урахуванням вихідного форматування. Застосунок надає можливість поставити запитання про вміст, щоб допомогти легко знайти необхідну інформацію;

– сканування продуктів. Мобільний застосунок здійснює сканування штрих-кодів або QR-кодів з підтримкою спеціальних можливостей за допомогою звукових сигналів в якості покажчиків і дозволяє прослуховувати назви продуктів та відомості з упаковки, якщо вони доступні;

– аналіз навколишніх сцен. Застосунок надає можливість прослухати загальний опис зафіксованої сцени. Щоб прослухати докладніший опис, необхідно натиснути кнопку «Додаткові відомості». У користувача також є можливість вивчити фотографію, переміщуючи палець по екрану, щоб почути інформацію про розташування різних об’єктів;

– аналіз облич людей. Інструментальний засіб зберігає обличчя людей, допомагаючи користувачу впізнавати їх, а також приблизно визначає їх вік, стать і вираз обличчя;

– сканування валюти. Мобільний застосунок може розпізнавати банкноти при наведенні камери на купюри;

– розпізнавання кольорів. Застосунок вміє визначати кольори при наведенні на них камери користувачем;

– сканування рукописного тексту. Інструментальний засіб читає та відтворює голосом рукописний текст, наприклад, у вітальних листівках (доступно кількома мовами);

– аналіз яскравості навколишнього середовища. Застосунок видає звуковий сигнал, який відповідає яскравості навколишнього середовища;

– аналіз зображень в інших програмах. Інструментальний засіб може розпізнавати та аналізувати фотографії з інших застосунків, наприклад, з «Галереї» чи соціальних мереж.

Переваги застосунку Seeing AI:

– озвучування тексту в реальному часі. Seeing AI дозволяє швидко розпізнавати та озвучувати текст із документів, етикеток, плакатів, рукописних записів тощо. Це дає змогу людям із порушеннями зору легко отримувати інформацію з друкованих матеріалів;

– розпізнавання об’єктів та опис сцен. Застосунок може ідентифікувати предмети та озвучувати їхній опис. Наприклад, він може сказати: «Перед вами стілець» або «На столі лежить книга». Це значно спрощує орієнтацію у просторі;

– розпізнавання банкнот. Seeing AI допомагає розпізнавати купюри різних валют, що є особливо корисним у повсякденному житті, коли потрібно розрахуватися готівкою;

– ідентифікація осіб. Застосунок може розпізнавати обличчя знайомих людей, зберігати їхні імена та навіть повідомляти про емоційний стан співрозмовника, що допомагає у соціальній взаємодії;

– сканування штрихкодів. Seeing AI може сканувати штрихкоди на товарах та озвучувати їхню назву й опис, що дозволяє людям із порушеннями зору самостійно здійснювати покупки;

– робота без підключення до Інтернету. Деякі функції, такі як розпізнавання короткого тексту, працюють автономно, що дуже зручно в умовах відсутності стабільного інтернет-з’єднання;

– безкоштовне використання. Seeing AI доступний безкоштовно для користувачів iOS, що робить його доступним для широкої аудиторії.

Недоліки застосунку Seeing AI:

– обмежена доступність на Android. Наразі Seeing AI офіційно доступний лише для пристроїв на iOS, що є суттєвим недоліком для користувачів Android;

– не завжди точне розпізнавання. Хоча штучний інтелект працює досить добре, він може помилятися при ідентифікації об’єктів, людей або сцен, особливо в умовах поганого освітлення;

– проблеми з рукописним текстом. Seeing AI може розпізнавати рукописний текст, але не завжди точно, особливо якщо почерк нерозбірливий або нестандартний;

– обмеження у розпізнаванні мов. Деякі функції, такі як розпізнавання рукописного тексту або опис сцен, можуть працювати менш точно для рідкісних мов або діалектів;

– високе енергоспоживання. Оскільки застосунок активно використовує камеру та штучний інтелект, він може швидко розряджати акумулятор смартфона;

– питання конфіденційності. Seeing AI передає деякі дані на сервери Microsoft для обробки, що може викликати побоювання щодо збереження особистої інформації.

PlantSnap – високотехнологічний, комплексний та точний мобільний застосунок для ідентифікації рослин (рис. 1.3) [3].

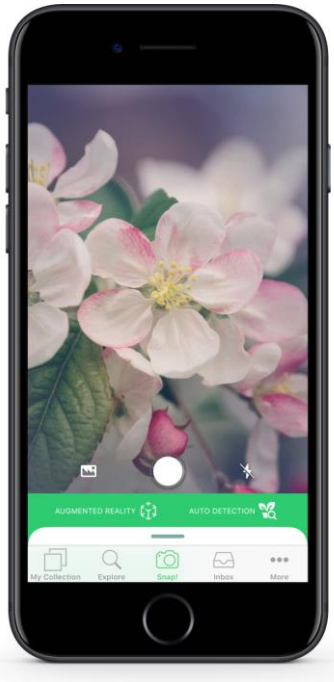


Рисунок 1.3 – Інтерфейс мобільного застосунку PlantSnap

Функції мобільного застосунку PlantSnap:

– розпізнавання рослин за зображенням. Надає всю необхідну інформацію про рослину, яка попадає в об’єктив камери смартфону при використанні мобільного застосунку. Програма містить базу даних з інформацією про рослини, завдяки чому знаходження інформації про рослини відбувається миттєво;

– публікування ключової інформації про рослини. Після ідентифікації рослини застосунок надає інформацію про її таксономію та повний опис рослини, незалежно від її виду. PlantSnap також розповідає, як доглядати за рослинами та вирощувати їх;

– пошук рослини за назвою. Інструментальний засіб містить пошуковий рядок, в якому користувач може записати назву рослини та отримати детальну інформацію про неї. У базі даних платформи міститься інформація про чимало видів рослин: дерева, кімнатні рослини, квіти, кущі тощо;

– аналіз фотографій рослин з усього світу. Застосунок містить мапу світу SnapMap. За допомогою функції «Дослідити» користувач може використовувати SnapMap, щоб знайти ідентифіковані рослини в будь-якій точці планети. Надана можливість подивитись анонімні фотографії, зроблені за допомогою PlantSnap, і відкрити для себе різні види квітів, листя, дерев, грибів і кактусів, поширених по всьому світу;

– створення власної колекції рослин. Застосунок надає можливість користувачу зберігати всі свої відкриття в одному місці та легко отримувати до них доступ в будь-який момент часу. Користувач може створити власну бібліотеку квітів, грибів і дерев;

– перегляд фото рослин на будь-якому пристрої. Інструментальний засіб надає можливість користувачу переглядати власні фотографії на будь-якому пристрої, через те, що зроблені фотографії зберігаються в Інтернеті, у хмарному сховищі PlantSnap. За допомогою ідентифікатора рослин PlantSnap користувач також може збільшити масштаб фотографій, щоб побачити кожну

деталь квітів, листя, кімнатних рослин, грибів, кактусів, декоративних рослин, м’ясоїдних рослин і сукулентів, ідентифікованих у всьому світі;

– навчання догляду за рослинами. PlantSnap може навчити користувача як доглядати за рослинами та квітами, як садити дерева, як доглядати за орхідеями та багато інших порад щодо садівництва.

Переваги застосунку Seeing AI:

– швидке розпізнавання рослин. PlantSnap дозволяє миттєво визначати види рослин за фотографією. Достатньо зробити знімок листя, квітки або стебла, і застосунок надасть детальну інформацію;

– велика база даних. Застосунок містить понад 600000 видів рослин, що робить його одним із найпотужніших інструментів для ботаніків, садівників та любителів природи;

– освітня цінність. Кожна рослина супроводжується детальним описом, включаючи наукову назву, середовище проживання, особливості догляду та цікаві факти;

– геолокація для покращення точності. Використання GPS допомагає визначити ймовірні види рослин залежно від вашого регіону, що підвищує точність ідентифікації;

– можливість створення власної бібліотеки рослин. Користувачі можуть зберігати знайдені рослини у власному каталозі, що допомагає вести особистий ботанічний щоденник;

– соціальна взаємодія. Застосунок дозволяє спілкуватися з іншими любителями рослин у спільноті, обмінюватися фотографіями та інформацією;

– підтримка різних платформ. PlantSnap доступний як для iOS, так і для Android, що робить його зручним для більшості користувачів.

Недоліки застосунку Seeing AI:

– потреба у стабільному інтернет-з’єднанні. Для точної ідентифікації застосунок використовує онлайн-базу даних, тому без доступу до інтернету його функціональність значно обмежується;

– не завжди точне визначення рослин. Іноді алгоритм може помилятися, особливо якщо фотографія зроблена в поганих умовах освітлення або з неправильного ракурсу;

– платна версія з розширеним функціоналом. Безкоштовна версія має обмежену кількість ідентифікацій на день, а для отримання повного доступу до всіх можливостей потрібна підписка;

– високе енергоспоживання. Постійне використання камери та GPS може призводити до швидкого розряджання акумулятора мобільного пристрою;

– проблеми з рідкісними видами. Якщо рослина є рідкісною або локальною для певного регіону, застосунок може не знайти її у своїй базі або видати неправильні результати;

– реклама у безкоштовній версії. Користувачі без преміум-підписки можуть стикатися з рекламою, що може бути незручним під час використання застосунку.

У результаті аналізу існуючих мобільних застосунків для розпізнавання об’єктів на зображеннях в Україні та за кордоном зроблено висновок, що хоча подібні інструменти демонструють високий рівень функціональності, точності та зручності використання, жоден із них не спеціалізується безпосередньо на розпізнаванні архітектурних пам’яток України. Більшість представлених програм, таких як Google Об’єктив, орієнтовані на універсальне розпізнавання об’єктів, що включає текст, рослини, тварин, предмети побуту, але не мають точкової бази даних архітектурних об’єктів культурної спадщини України. Seeing AI зосереджений передусім на допомозі людям з порушенням зору, а PlantSnap – на ідентифікації флори.

Зазначене вище, вказує на наявність значної ніші для створення спеціалізованого продукту, орієнтованого саме на українські архітектурні пам’ятки. Такий застосунок не лише заповнює прогалину в існуючому програмному забезпеченні, а й сприяє збереженню культурної спадщини, формуванню цифрової ідентичності нації та популяризації архітектурного

туризму в Україні. Отже, розроблення мобільного застосунку з функцією розпізнавання архітектурних об’єктів України на основі глибокого навчання є своєчасним, актуальним і соціально значущим завданням.

1.2 Аналіз літературних джерел щодо апробації результатів розроблення мобільних застосунків для розпізнавання об’єктів на зображеннях

Як зазначають автори статті [4], люди з вадами зору (Visually Impaired Persons, VIP) складають значну частину населення, і вони присутні по всьому світу та в будь-якій частині світу. Останнім часом технології довели свою присутність у кожній сфері, а інноваційні пристрої допомагають людям у повсякденному житті. У цій роботі розумна та інтелектуальна система розроблена для VIP-персон, щоб допомогти мобільності та забезпечити їх безпеку. Пропонована система забезпечує навігацію в режимі реального часу за допомогою автоматизованого голосу. Хоча VIP-персони не зможуть бачити об’єкти навколо себе, вони можуть відчути та візуалізувати середовище роумінгу. Крім того, для забезпечення їх безпеки розроблено вебзастосунок. Користувач цієї програми може ввімкнути функцію на вимогу для обміну своїм місцезнаходженням із сім’єю, порушуючи конфіденційність. За допомогою цього застосунку члени сімей VIP-персон зможуть відстежувати їх пересування (отримувати місцезнаходження та знімки), перебуваючи вдома. Таким чином, пристрій дозволяє VIP-персонам візуалізувати навколишнє середовище та забезпечити свою безпеку. Такий комплексний пристрій був відсутньою ланкою в існуючій літературі. Застосунок використовує архітектуру MobileNet через низьку обчислювальну складність для роботи на кінцевих пристроях з низьким енергоспоживанням. Для оцінки ефективності запропонованої системи було проведено шість пілотних досліджень, які показали задовільні результати. Для виявлення та

розпізнавання об’єктів використовується модель глибокої згорткової нейронної мережі (Convolutional Neural Network, CNN) із точністю 83,3%, тоді як набір даних містить понад 1000 категорій. Крім того, виконується кількісний порівняльний аналіз на основі балів з використанням підтримуваних функцій пристроїв. Виявлено, що запропонована система перевершила існуючі пристрої, отримавши загальну оцінку 9,1/10, що на 8% вище, ніж друга найкраща.

Автори статті [5] повідомляють, що після появи AlexNet методи, засновані на CNN, стали основним підходом у сфері комп’ютерного зору. Це спричинило появу численних досліджень у галузі нейронних мереж та різних модифікацій алгоритмічних структур. Проте для досягнення швидкого та точного виявлення необхідно вийти за межі існуючої архітектури CNN, що є великою проблемою. Відносно зріла теоретична база та технологічний розвиток трансформерів у сфері обробки природної мови (Natural Language Processing, NLP) привернули увагу дослідників. Було доведено, що метод трансформерів можна застосовувати для завдань комп’ютерного зору, і що в деяких випадках він перевершує традиційні методи CNN. Щоб допомогти дослідникам краще зрозуміти процес розвитку методів виявлення об’єктів, їхні особливості, існуючі підходи, різні архітектури, виклики та перспективи розвитку, у цій статті розглядаються класичні історичні методи виявлення об’єктів, що використовують CNN. Обговорюються ключові особливості, переваги та недоліки цих алгоритмів. На основі аналізу великої кількості наукових робіт здійснено порівняння методів виявлення об’єктів, заснованих на CNN і трансформерах. Для справедливого аналізу обрано 13 основних методів, які мають значний вплив у цій сфері та є найпопулярнішими й перспективними. Отримані порівняльні дані підтверджують перспективність розвитку трансформерів і можливість їхньої інтеграції з іншими методами. Окрім того, представлено останні інноваційні підходи до використання трансформерів у завданнях комп’ютерного зору. Наприкінці статті

підсумовуються основні виклики, можливості та перспективи розвитку цієї сфери.

Автори статті [6] говорять про те, що сільськогосподарські шкідники спричиняють від 20% до 40% втрат у світовому виробництві сільськогосподарських культур щороку, як повідомляє Продовольча та сільськогосподарська організація ООН (Food and Agriculture Organization, FAO). Тому розумне сільське господарство є найкращим варіантом для фермерів, що дозволяє застосовувати методи штучного інтелекту, інтегровані з сучасними інформаційними та комунікаційними технологіями, для боротьби з небезпечними комахами-шкідниками. Це, своєю чергою, сприятиме підвищенню продуктивності сільськогосподарських культур. У зв'язку з цим у цій статті представлено новий мобільний застосунок, який автоматично класифікує шкідників за допомогою глибокого навчання для підтримки роботи фахівців і фермерів. Розроблений застосунок використовує Faster R-CNN (покращену регіональну згорткову нейронну мережу) для виконання завдання розпізнавання комах-шкідників на основі хмарних обчислень. Крім того, база даних рекомендованих пестицидів пов'язана з виявленими шкідниками сільськогосподарських культур, щоб надавати фермерам корисні рекомендації. Дослідження було успішно протестовано на п'яти групах шкідників: попелиця (Aphids), цикадки (Cicadellidae), льоновий бруньковий червець (Flax Budworm), земляні блішки (Flea Beetles) та червоний павук (Red Spider). Запропонована модель Faster R-CNN продемонструвала найвищу точність розпізнавання – 99,0% для всіх тестових зображень шкідників. Крім того, наша модель глибокого навчання перевершує інші методи розпізнавання, зокрема Single Shot Multi-Box Detector (SSD) MobileNet та традиційні зворотні нейронні мережі (BP – Back Propagation). Основна перспектива цього дослідження – впровадження розробленого мобільного застосунку для онлайн-розпізнавання сільськогосподарських шкідників як у відкритих польових умовах (великих фермах), так і в теплицях для конкретних культур.

У статті [7] розглядаються актуальні завдання комп’ютерного зору – розпізнавання та класифікація об’єктів на зображеннях. Як альтернатива існуючим методам запропоновано використання методів глибокого навчання на основі нейронних мереж. Розроблено новий підхід для ефективного розпізнавання та класифікації кулінарних страв на зображеннях, який передбачає використання можливостей бібліотеки глибокого навчання TensorFlow та особливостей CNN. Було створено програмний застосунок для розпізнавання та класифікації кулінарних страв. Тестування запропонованого інструменту показало, що після 7 повних епох навчання моделі було досягнуто 77% точності, що є доволі хорошим результатом, враховуючи одну з основних проблем у розпізнаванні кулінарних страв – міжкласову схожість. Перспективним напрямком подальших досліджень є створення підкласів для існуючих загальних класів страв.

Автори статті [8] зазначають, що економіка Тайваню значною мірою залежить від експорту сільськогосподарської продукції. Якщо після експорту навіть виникає підозра щодо наявності шкідників у сільськогосподарській продукції, не лише відбувається повернення товару, але й уся партія врожаю підлягає знищенню, що спричиняє значні збитки. Основні шкідники лускокрилих комах у Тайвані – борошнисті червці (Mealybugs), щитівки (Coccidae) та діаспідіди (Diaspididae) – не лише завдають серйозної шкоди рослинам, а й суттєво впливають на сільськогосподарське виробництво. Тому виявлення цих шкідників є критично важливим завданням у сфері сільського господарства Тайваню. У цьому дослідженні запропоновано систему виявлення шкідників на основі штучного інтелекту, яка дозволяє автоматично ідентифікувати лускокрилих шкідників за зображеннями. Для розпізнавання використовуються моделі глибокого навчання для виявлення об’єктів, зокрема: Faster R-CNN, Single-Shot Multibox Detector (SSD), You Only Look Once v4 (YOLO v4). Результати експериментів показали, що YOLO v4 продемонстрував найвищу точність класифікації серед усіх алгоритмів: 100% для борошнистих червців (Mealybugs), 89% для щитівок

(Coccidae), 97% для діаспідід (Diaspididae). Крім того, висока швидкість обчислень моделі YOLO v4 підтверджує її придатність для реального часу. Результати розпізнавання цієї моделі також допомагають кінцевим користувачам у прийнятті рішень. Для спрощення ідентифікації шкідників у польових умовах було розроблено мобільний застосунок, який використовує навчену модель розпізнавання шкідників. Це рішення допомагає вчасно застосовувати відповідні пестициди, що сприяє зменшенню втрат урожаю.

Автори статті [9] зазначають, що рослинні хвороби є однією з головних проблем, з якими стикається сільськогосподарський сектор у всьому світі. У Сполучених Штатах Америки захворювання сільськогосподарських культур призводять до втрат майже третини врожаю щороку. Попри важливість своєчасної діагностики, визначення хвороб культур на основі візуального огляду листя є складним завданням, особливо для фермерів із обмеженими ресурсами. Тому існує нагальна потреба в значно кращих методах виявлення, моніторингу та прогнозування хвороб рослин, щоб мінімізувати втрати у сільському господарстві. Комп’ютерний зір у поєднанні з машинним навчанням (Machine Learning, ML) має величезний потенціал для покращення контролю за станом посівів у масштабах великих господарств. У цій статті представлено мобільну систему на основі машинного навчання для автоматизованої діагностики хвороб листя рослин. Розроблена система використовує CNN як основний механізм глибокого навчання для класифікації 38 категорій хвороб. Для тренування, валідації та тестування моделі CNN було зібрано набір зображень, що містить 96206 фотографій листя рослин, як здорових, так і уражених хворобами. Інтерфейс системи реалізований у вигляді мобільного застосунку для Android, який дозволяє фермерам зробити фото ураженого листя рослини. Після цього застосунок визначає категорію хвороби та відображає відсоток впевненості в класифікації. Очікується, що цей інструмент допоможе фермерам зберігати врожай здоровим і запобігати використанню неправильних добрив, які можуть завдати додаткового стресу рослинам. Нарешті, система була оцінена

за різними показниками продуктивності, такими як точність класифікації та час обробки. Результати показали, що запропонована модель досягає загальної точності 94% у розпізнаванні 38 найпоширеніших класів хвороб у 14 видах сільськогосподарських культур.

Як зазначають автори статті [10], останнім часом у багатьох країнах повідомлялося про спалахи віспи мавп серед людей. Згідно зі звітами та дослідженнями, швидке виявлення та ізоляція інфікованих людей мають важливе значення для зниження швидкості поширення. Це дослідження представляє мобільний застосунок для Android, який використовує глибоке навчання, щоб допомогти в цій ситуації. Застосунок розроблено за допомогою Android Studio з використанням мови програмування Java та Android SDK 12. Відеозображення, зібрані камерою мобільного пристрою, надсилаються до глибокої згорткової нейронної мережі, яка працює на тому самому пристрої. Camera2 API платформи Android використовується для доступу до камери та операцій. Потім мережа класифікує зображення як позитивні або негативні для виявлення мавпячої віспи. Навчання мережі було проведено з використанням зображень уражень шкіри людей, інфікованих віспою мавп, та інших зображень уражень шкіри. Для цієї мети було використано загальнодоступний набір даних і підхід глибокого перенесення. Усі етапи навчання та тестування були застосовані в Matlab з використанням різних попередньо навчених мереж. Потім мережа з найкращою точністю була відтворена та навчена за допомогою TensorFlow. Модель TensorFlow була адаптована для мобільних пристроїв шляхом перетворення на модель TensorFlow Lite. Потім модель TensorFlow Lite було вбудовано в мобільний застосунок разом із бібліотекою TensorFlow Lite для виявлення віспи мавп. Застосунок успішно запущено на трьох пристроях. Під час виконання було зібрано час висновку. Середній час висновку становив 197 мс, 91 мс і 138 мс. Представлена система дозволяє людям з ураженням організму швидко поставити попередній діагноз. Таким чином, людей, інфікованих віспою мавп, можна заохотити діяти швидко, щоб звернутися до експерта для

остаточного діагнозу. Згідно з результатами тесту, система може класифікувати зображення з точністю 91,11%. Крім того, запропонований мобільний застосунок можна навчити для попередньої діагностики інших шкірних захворювань.

Аналіз літературних джерел щодо апробації результатів розроблення мобільних застосунків для розпізнавання об’єктів на зображеннях [4–10] показав, що дослідження у цій галузі охоплюють широкий спектр практичних задач – від підтримки людей з вадами зору до аграрного сектора і медичної діагностики. У більшості робіт акцент зроблено на використанні глибоких згорткових нейронних мереж або їхніх модифікацій, таких як Faster R-CNN, YOLO v4, MobileNet, що підтверджує їхню ефективність у задачах виявлення та класифікації об’єктів на зображеннях.

Спільною рисою успішних реалізацій є орієнтація на мобільні платформи, оптимізація моделей для швидкодії, а також застосування підходів трансферного навчання з використанням наявних відкритих наборів даних. Важливим аспектом апробації є тестування в реальних умовах використання кінцевими користувачами, що забезпечує релевантну оцінку точності, швидкості роботи та зручності інтерфейсу.

У контексті кваліфікаційної роботи зазначені підходи є надзвичайно актуальними, оскільки розроблюваний мобільний застосунок для розпізнавання архітектурних пам’яток України також ґрунтується на згорткових нейронних мережах.

Урахування найкращих практик, описаних у проаналізованих джерелах, дозволяє адаптувати сучасні методи глибинного навчання до специфіки завдання – ідентифікації архітектурних об’єктів на зображеннях, що є важливим кроком для збереження культурної спадщини та популяризації історичних пам’яток нашої країни серед широкої аудиторії.

1.3 Постановка задачі

Таким чином, автоматичне розпізнавання архітектурних пам’яток на зображеннях є актуальним завданням у сфері комп’ютерного зору, що має прикладне значення для збереження культурної спадщини, розвитку туризму та мобільних інформаційних технологій. Тому ставиться задача розроблення мобільного застосунку, який здатний розпізнавати визначні архітектурні пам’ятки України за допомогою методів глибокого навчання, зокрема згорткових нейронних мереж.

Об’єктом роботи є процес розпізнавання архітектурних пам’яток України на фотографіях за допомогою мобільного застосунку.

Метою роботи є розроблення мобільного застосунку, який здійснює розпізнавання архітектурних пам’яток України на зображеннях з використанням згорткових нейронних мереж.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– здійснити аналіз існуючих підходів до класифікації зображень з використанням CNN у мобільних застосунках;

– сформувати вибірку зображень 18 архітектурних пам’яток України для тренування та тестування моделі;

– підготувати навчальні дані та здійснити тренування згорткової нейронної мережі;

– реалізувати мобільний застосунок у середовищі Android Studio з використанням TensorFlow Lite;

– забезпечити інтеграцію навченої моделі CNN у мобільний застосунок;

– провести тестування застосунку на реальних зображеннях та оцінити точність і швидкодію системи;

– проаналізувати результати та визначити шляхи подальшого вдосконалення системи.

2 ОСОБЛИВОСТІ РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ АРХІТЕКТУРНИХ ПАМ'ЯТОК УКРАЇНИ НА ЗОБРАЖЕННЯХ

2.1 Особливості сучасних методів розпізнавання об'єктів на зображеннях

Сучасні методи розпізнавання об'єктів на зображеннях значно еволюціонували за останнє десятиліття завдяки розвитку обчислювальної техніки, збільшенню обсягів доступних даних та удосконаленню алгоритмів штучного інтелекту. Якщо раніше розпізнавання об'єктів базувалося на класичних підходах комп'ютерного зору, таких як гістограми градієнтів (Histogram of Oriented Gradients, HOG), детектори ознак (Scale-Invariant Feature Transform, Speeded Up Robust Features) та каскади Хаара, то нині домінують методи, засновані на глибокому навчанні, зокрема згорткових нейронних мережах (Convolutional Neural Network, CNN).

Однією з головних особливостей сучасних методів є їх здатність самостійно виділяти релевантні ознаки із зображення під час процесу навчання. Це суттєво відрізняється від традиційних підходів, де інженери вручну визначали набір ознак для класифікації. У згорткових нейронних мережах (CNN) ознаки автоматично витягуються на кількох рівнях абстракції: від простих (лінії, кути) до складніших (частини об'єктів, повні форми). Це дозволяє досягти високої точності розпізнавання навіть у складних умовах – при різному освітленні, часткових перекриттях, поворотах, масштабуванні та фоновому шумі.

Ще однією перевагою є масштабованість таких методів. Моделі типу YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) та Faster R-CNN дозволяють не лише класифікувати об'єкти, а й знаходити їх точне положення на зображенні. Більше того, завдяки передтренованим моделям (наприклад, MobileNet, ResNet, Inception), можливо швидко адаптувати

системи до нових задач методом донавчання (fine-tuning), використовуючи обмежені вибірки специфічних зображень.

Істотним досягненням є також інтеграція моделей розпізнавання в мобільні пристрої завдяки оптимізованим форматам (наприклад, TensorFlow Lite, ONNX) та апаратному прискоренню (Graphics Processing Unit, Tensor Processing Unit, Neural Processing Unit). Це дозволяє здійснювати розпізнавання в реальному часі прямо на смартфоні без необхідності підключення до Інтернету, що критично важливо для швидкої реакції систем та конфіденційності даних.

Попри значні переваги, сучасні методи мають і певні обмеження. Вони потребують великої кількості даних для навчання, високих обчислювальних ресурсів на етапі тренування та ретельного контролю за перенавчанням (overfitting). Також вони можуть бути вразливими до атак типу «adversarial examples», коли незначні зміни в пікселях вводять модель в оману.

Загалом, сучасні методи розпізнавання об’єктів на зображеннях є потужним інструментом, що дозволяє реалізовувати високоточні та масштабовані системи для широкого спектру завдань – від автономного водіння до медичної діагностики, від охорони правопорядку до цифрової ідентифікації культурної спадщини.

До появи підходів, заснованих на глибокому навчанні, традиційні методи виявлення об’єктів спиралися на створені вручну функції та спеціальні алгоритми [11].

Нижче наведені деякі з відомих традиційних підходів до виявлення об’єктів.

Гістограма орієнтованих градієнтів (Histogram of Oriented Gradients, HOG) – це дескриптор функції, який використовується для представлення інформації про локальну текстуру та форму зображення. Він фіксує інформацію про орієнтацію градієнта та обчислює гістограми напрямків градієнта для виявлення країв і меж об’єктів. HOG широко використовується для виявлення пішоходів та інших завдань виявлення об’єктів.

Хаар-подібні функції – це прості прямокутні фільтри, які використовуються в алгоритмі Віоли-Джонса для виявлення об’єктів. Ці функції вловлюють варіації інтенсивності в певних областях зображення та є обчислювально ефективними для застосунків у реальному часі. Алгоритм Віоли-Джонса відомий своїми можливостями швидкого виявлення облич.

Відповідність функцій: методи зіставлення функцій, такі як Scale-Invariant Feature Transform (SIFT) і Speeded-Up Robust Features (SURF), виявляють характерні локальні особливості на зображенні та зіставляють їх між кадрами для розпізнавання та відстеження об’єктів. Ці методи використовуються для виявлення об’єктів і вирівнювання зображення.

Моделі деформованих частин (Deformable Part Models, DPM) – це класичний фреймворк для виявлення об’єктів, який представляє об’єкти як набір деформованих частин. Він моделює просторове співвідношення між частинами та фіксує варіації зовнішнього вигляду об’єкта для підвищення точності виявлення. DPM використовувався для виявлення об’єктів із шарнірними конструкціями.

Вибірковий пошук – це метод генерації пропозицій, який використовується під час виявлення об’єктів для створення регіонів-кандидатів, які ймовірно містять об’єкти. Він сегментує зображення на основі кольору, текстури та розміру, щоб отримати потенційні області об’єкта для подальшої обробки [12].

Хоча традиційні підходи до виявлення об’єктів були ефективними в певних сценаріях, методи глибокого навчання, такі як YOLO та SSD, перевершили їх з точки зору точності та ефективності, особливо в задачах виявлення об’єктів у реальному часі.

Методи на основі ковзних вікон були одними з ранніх традиційних підходів до виявлення об’єктів. Ці методи включають переміщення вікна фіксованого розміру по зображенню в різних масштабах для виявлення об’єктів у різних місцях і розмірах. Хоча підходи ковзного вікна були

значною мірою замінені методами глибокого навчання, вони дають цінну інформацію про еволюцію цих методів.

Методи ковзного вікна:

– гістограми орієнтованих градієнтів для виявлення людини. Було представлено дескриптор функцій, який став ключовим компонентом багатьох детекторів об’єктів на основі ковзного вікна. Дескриптор HOG фіксує інформацію про орієнтацію локальних градієнтів для представлення країв об’єктів і широко використовується для виявлення пішоходів [13];

– виявлення об’єктів за допомогою дискримінаційно навчених моделей на основі частин. Представлено структуру моделей деформованих частин (DPM) для виявлення об’єктів. DPM використовує підхід ковзного вікна для пошуку частин об’єкта, моделюючи просторові відносини між частинами для кращої точності виявлення;

– відмінні риси зображення від масштабно-інваріантних ключових моментів. Масштабно-інваріантне перетворення ознак (SIFT), широко використовується для завдань зіставлення ознак і розпізнавання об’єктів. Ковзні вікна часто використовуються в методах на основі SIFT для виявлення ключових точок і виконання зіставлення ознак у різних масштабах зображення;

– швидке виявлення об’єктів за допомогою розширеного каскаду простих функцій. Алгоритм Віоли-Джонса, який є одним із найперших і успішних методів виявлення об’єктів у реальному часі, заснованих на функціях, подібних до Хаара. Техніка ковзного вікна використовується в алгоритмі Віоли-Джонса для сканування всього зображення на потенційні місця розташування об’єктів.

Методи на основі ковзних вікон були обмежені їхньою обчислювальною складністю, оскільки вони включали вичерпну оцінку ковзних вікон у кількох масштабах, що призводило до тривалого обчислення.

Розробка підходів на основі глибокого навчання, таких як YOLO та SSD, значно підвищила швидкість і точність виявлення об’єктів шляхом

впровадження наскрізного навчання та нових архітектур. Ці сучасні методи значною мірою замінили підходи на основі ковзних вікон у практичних застосуваннях, оскільки вони досягають продуктивності в реальному часі без необхідності явного сканування вікон.

Підходи, засновані на функціях відіграли значну роль у ранньому розвитку методів виявлення об’єктів і були основоположними в історії комп’ютерного зору. Ці методи спиралися на ручне виділення ознак і спеціалізовані алгоритми для ідентифікації об’єктів на зображеннях. Хоча їх значною мірою перевершили методи, засновані на глибокому навчанні, підходи, засновані на функціях, проклали шлях до більш просунутих методів.

Методи, засновані на функціях дали цінну інформацію та заклали основу для дослідження виявлення об’єктів. Однак у них були обмеження, такі як потреба в функціях ручної роботи та великих обчислювальних ресурсах. Розвиток підходів до глибокого навчання та наскрізного навчання призвів до суттєвих покращень у точності та ефективності виявлення об’єктів, завдяки чому методи, засновані на функціях, рідше використовуються в сучасних програмах. Перехід до методів, заснованих на глибокому навчанні, дозволив автоматично вивчати функції, зменшуючи залежність від створених вручну функцій і створюючи більш складні та точні системи виявлення об’єктів [14].

Досягнення функціональності в реальному часі потребує швидкої обробки кадрів або зображень, що створює складність через обчислювальну інтенсивність методологій глибокого навчання, таких як Faster R-CNN і YOLO. Щоб вирішити цю проблему, дослідники розробили полегшені архітектури, такі як SSD і YOLOv3-tiny, що знижує точність для швидшої обробки. Збільшення швидкості обчислень можливе за допомогою механізмів апаратного прискорення, таких як GPU (Graphics Processing Unit) або TPU (Tensor Processing Unit).

Підтримка точності виявлення має вирішальне значення, але прискорена обробка може призвести до зниження точності порівняно з

повільнішими, але більш точними методами. Щоб пом’якшити зниження точності, можна використовувати розширені архітектури, складні магістралі, як-от залишкова нейронна мережа (Residual Neural Network, ResNet), і оптимізацію гіперпараметрів під час навчання. Розглядання різноманітності розміру об’єкта та співвідношення сторін у сценах реального світу потребує таких методів, як мережа пірамід функцій (Feature Pyramid Network, FPN) і блоки прив’язки. FPN фіксує багатомасштабні об’єкти, тоді як блоки прив’язки полегшують прогнозування для об’єктів різного розміру.

Часткова оклюзія та безлад у реальних сценах ускладнюють виявлення. Можна розробити стійкі моделі виявлення об’єктів, які обробляють перекриття та перешкоди, використовуючи контекстну інформацію або часову узгодженість між кадрами для підвищення точності.

Обмеження ресурсів у периферійних пристроях або вбудованих системах можна вирішити за допомогою спрощених архітектур і методів квантування моделей.

Навчання моделей виявлення об’єктів у реальному часі вимагає значної кількості анотованих даних. Ефективність можна підвищити завдяки передачі навчання та доповненню даних, використовуючи попередньо навчені моделі та синтетичні дані для зменшення вимог до анотацій.

Адаптація моделей з одного середовища до різних з різними умовами вимагає асиміляції різноманітних навчальних даних і застосування методів адаптації домену для адаптації.

Конвергенція вдосконалених алгоритмів, оптимізації апаратного забезпечення та підібраних наборів даних має ключове значення для подолання цих багатогранних викликів. Вчені та практики постійно досліджують інноваційні методи вдосконалення виявлення об’єктів у режимі реального часу для застосунків, що охоплюють робототехніку, спостереження, автономні транспортні засоби тощо.

Виявлення об’єктів у реальному часі є наріжною технологією в сферах комп’ютерного зору та штучного інтелекту, наділяючи машини здатністю

миттєво сприймати та розуміти оточення. Об’єднання передових алгоритмів глибокого навчання, спрощена архітектура та механізми апаратного прискорення породили парадигматичну трансформацію, катапультуючи виявлення об’єктів у реальному часі в різноманітні програми, що охоплюють автономні транспортні засоби, спостереження, робототехніку тощо.

Еволюція виявлення об’єктів у реальному часі супроводжувалася низкою проблем, кожна з яких зустрічалася з інноваційними рішеннями. Від появи одноетапних архітектур, таких як YOLO та SSD, до оркестровки периферійних обчислювальних парадигм і апаратних прискорювачів. Шлях виявлення об’єктів у реальному часі характеризується постійним удосконаленням і прогресом [15]. Майбутні напрямки включають виявлення тривимірних об’єктів у реальному часі, ефективні апаратні архітектури, мультимодальне злиття та поступове навчання. Проблеми охоплюють обробку складних сцен, обмеження ресурсів і зміщення даних, що підкреслює необхідність постійних досліджень і розробок для підвищення точності, ефективності та надійності виявлення об’єктів у реальному часі в програмах і доменах, що розвиваються.

2.2 Формування методики розпізнавання архітектурних пам’яток України на зображеннях

Розроблення методики розпізнавання архітектурних пам’яток України на зображеннях є складною багатоетапною задачею, що охоплює збирання та підготовку даних, проєктування та навчання моделі глибокого навчання, її оптимізацію для мобільного середовища, а також інтеграцію в мобільний застосунок. Для реалізації цього завдання в межах кваліфікаційної роботи було використано один із найефективніших підходів у сучасному комп’ютерному зорі – згорткові нейронні мережі, які демонструють високу точність у задачах класифікації та розпізнавання зображень [16, 17].

Згорткові нейронні мережі належать до класу глибоких нейронних мереж і призначені спеціально для обробки двовимірних вхідних даних, зокрема зображень. Їх головною особливістю є наявність згорткових шарів, які виконують операцію згортки з фільтрами (ядрами), дозволяючи автоматично виявляти характерні ознаки зображення на різних рівнях абстракції. На ранніх шарах модель виявляє прості структури – краї, лінії, кути; на глибших – більш складні деталі, такі як форми, орнаменти, візерунки, а зрештою – цілісні елементи об’єкта [18].

Принцип роботи згорткових нейронних мереж (Convolutional Neural Networks, CNN) показано на рисунку 2.1 [19].

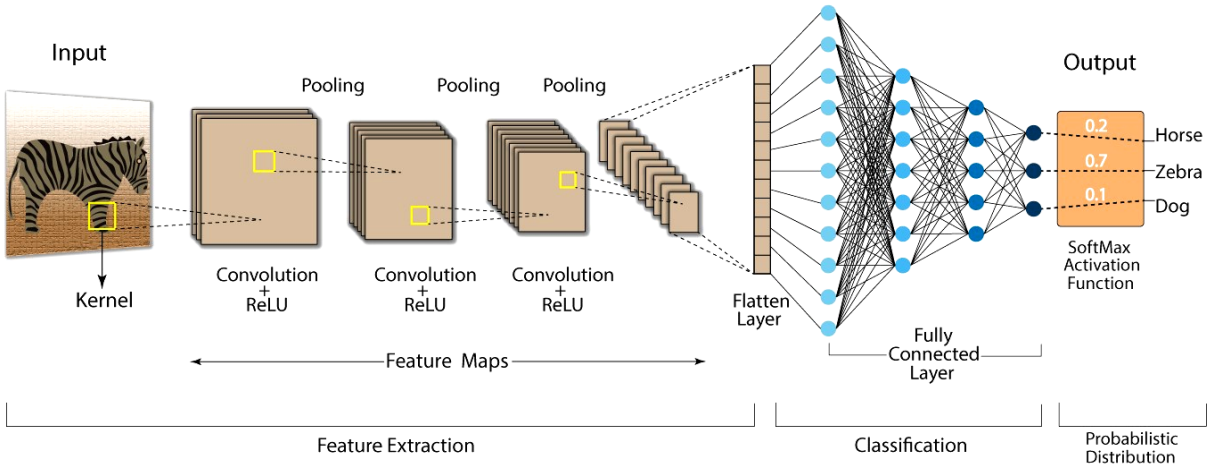


Рисунок 2.1 – Принцип роботи згорткових нейронних мереж

Для побудови ефективної системи розпізнавання архітектурних пам’яток України першим важливим кроком стало формування якісного набору даних. Було зібрано зображення вісімнадцяти архітектурних об’єктів, які представляють різні регіони України. Джерелами даних слугували як відкриті фотобанки та туристичні портали, так і власноруч зроблені знімки. Особлива увага приділялась різноманітності зображень – як за ракурсом, так і за погодними умовами, освітленням, порою року та наявністю додаткових об’єктів на фоні. Таке урізноманітнення дозволяє моделі навчитися узагальнювати ознаки та підвищує її стійкість до зміни умов зйомки.

Перед подачею в модель усі зображення були приведені до уніфікованого формату, нормалізовані та доповнені аугментаціями – випадковими обертаннями, дзеркальними відображеннями, масштабуванням, регулюванням яскравості та контрасту. Аугментація відіграє ключову роль у збільшенні варіативності даних і допомагає уникнути перенавчання моделі на обмеженому наборі прикладів [20].

Для навчання моделі було обрано архітектуру MobileNetV2 – легку, оптимізовану для мобільних пристроїв нейронну мережу, що демонструє хороший компроміс між точністю і швидкодією. Завдяки використанню глибинних згорток, MobileNetV2 забезпечує значне зменшення кількості параметрів і обчислювальних витрат при збереженні здатності до розпізнавання складних ознак [21].

Архітектуру мережі MobileNetV2 показано на рисунку 2.2 [22].

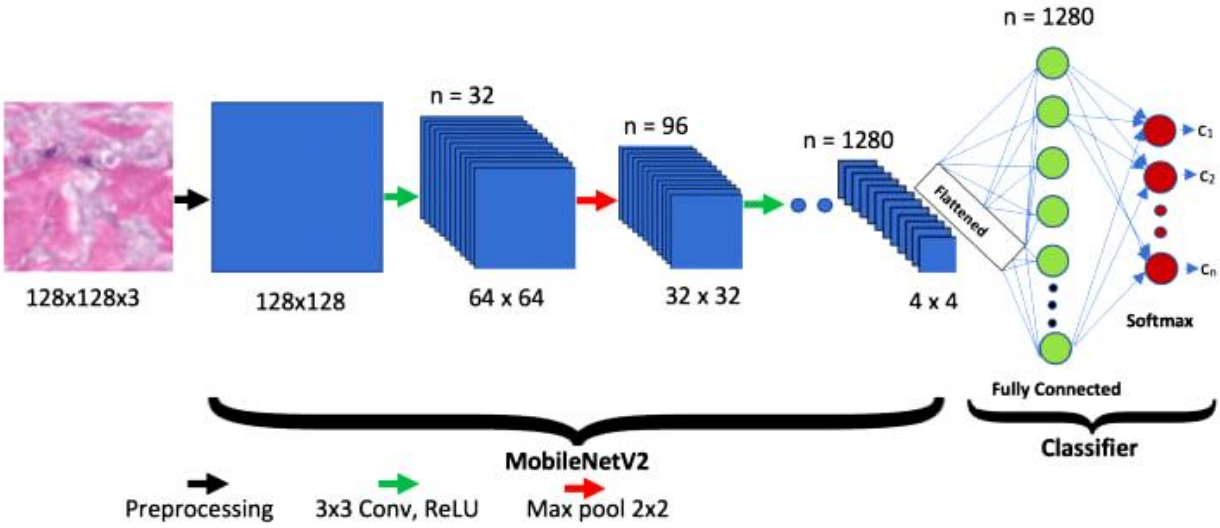
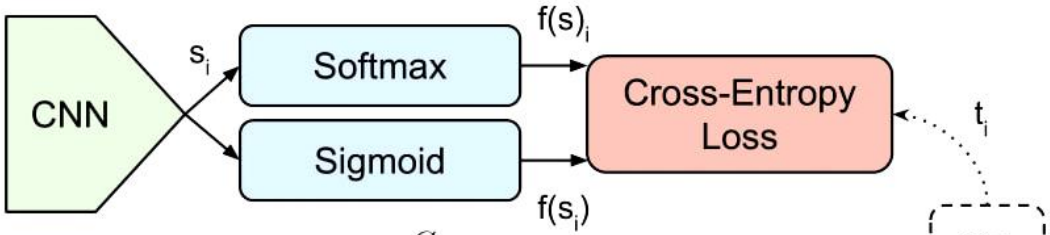


Рисунок 2.2 – Архітектура мережі MobileNetV2

Процес навчання тривав кілька епох із використанням функції втрат категорійної кросентропії та оптимізатора Adam. Функція втрат категорійної кросентропії – це одна з найпоширеніших функцій втрат, яка використовується для задач багатокласової класифікації у нейронних мережах, зокрема згорткових нейронних мережах.

Оптимізатор Adam – це один з найпопулярніших та найефективніших алгоритмів оптимізації у глибокому навчанні, який використовується для оновлення ваг нейронної мережі під час навчання [23].

Принцип роботи функції втрат категорійної кросентропії зображено на рисунку 2.3 [24].



$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1-t_1) log(1-f(s_1))$$

Рисунок 2.3 – Принцип роботи функції втрат категорійної кросентропії

Принцип роботи функції оптимізатора Adam зображено на рисунку 2.4 [25].

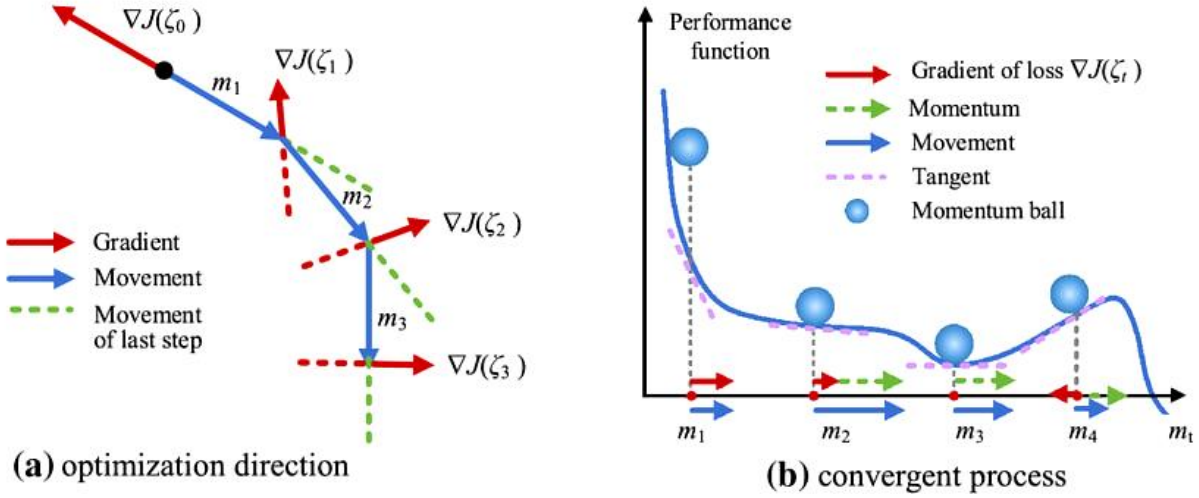


Рисунок 2.4 – Принцип роботи оптимізатора Adam

Графічний процесор (Graphics Processing Unit, GPU) – це спеціалізований мікропроцесор, який спочатку був розроблений для прискорення обробки графіки, зокрема рендерингу зображень, відео та тривимірної анімації. Однак у сучасному машинному навчанні, особливо при роботі з нейронними мережами, GPU став одним із ключових компонентів для обробки великих обсягів обчислень. Графічний процесор (GPU) складається з сотень або тисяч менш потужних, але високо паралельних ядер. Така архітектура ідеально підходить для задач, які вимагають однотипних операцій над великими обсягами даних – саме те, що потрібно при тренуванні згорткових нейронних мереж (CNN) [26].

Модель навчалася на графічному процесорі (Graphics Processing Unit, GPU) з контролем точності за допомогою валідаційної вибірки та з використанням механізму ранньої зупинки. Після завершення навчання було досягнуто точності 73,43% на тестовій вибірці, що є достатньо високим результатом для задачі з 18 класами (пам’яток архітектури). Графіки з детальною інформацією щодо тренування моделі наведені на рисунку 2.5.

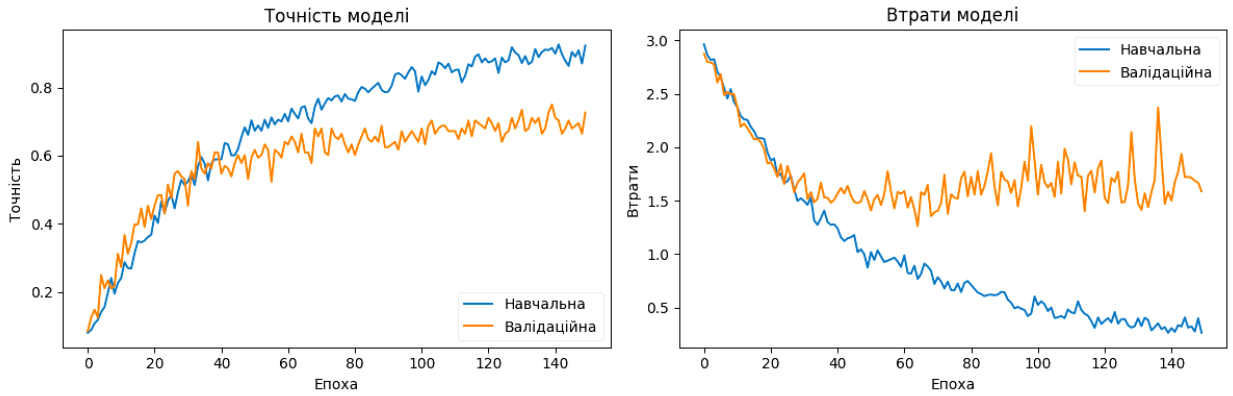


Рисунок 2.5 – Історія тренування моделі CNN

TensorFlow Lite – це легковаговий формат моделі машинного навчання, розроблений компанією Google, спеціально для запуску нейронних мереж на мобільних, вбудованих та пристроях з обмеженими ресурсами [27].

Робочий процес для випадків використання TensorFlow Lite наведено на рисунку 2.6 [28].

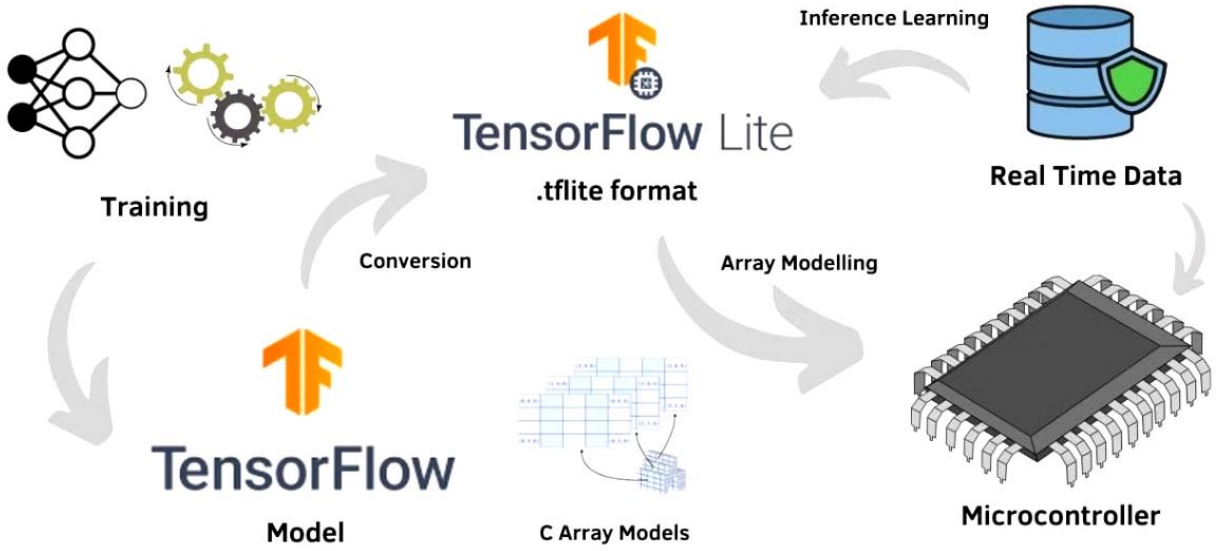


Рисунок 2.6 – Робочий процес використання TensorFlow Lite

Наступним етапом стало перетворення моделі у формат TensorFlow Lite. Ця операція необхідна для запуску моделі безпосередньо на мобільному пристрої без потреби звернення до серверу [29]. Таке рішення не тільки покращує швидкість роботи застосунку, а й гарантує конфіденційність персональних даних користувача. Крім того, було застосовано квантування моделі (quantization), яке ще більше зменшує розмір файлу і пришвидшує обчислення без суттєвого впливу на точність класифікації.

Розроблення мобільного застосунку здійснювалося у середовищі Android Studio з використанням мови програмування Kotlin. Android Studio – це офіційне інтегроване середовище розробки для створення мобільних застосунків під операційну систему Android, яке підтримується і розробляється компанією Google. Воно базується на IntelliJ IDEA – популярному середовищі розробки від компанії JetBrains, адаптованій для Android-розробки.

Kotlin – це сучасна мова програмування, яка офіційно підтримується Google для розробки застосунків під Android.

Kotlin була створена компанією JetBrains у 2011 році як більш лаконічна, безпечна та ефективна альтернатива Java, і стала офіційною мовою Android-розробки у 2017 році. Інтерфейс середовища розробки Android Studio показано на рисунку 2.7.

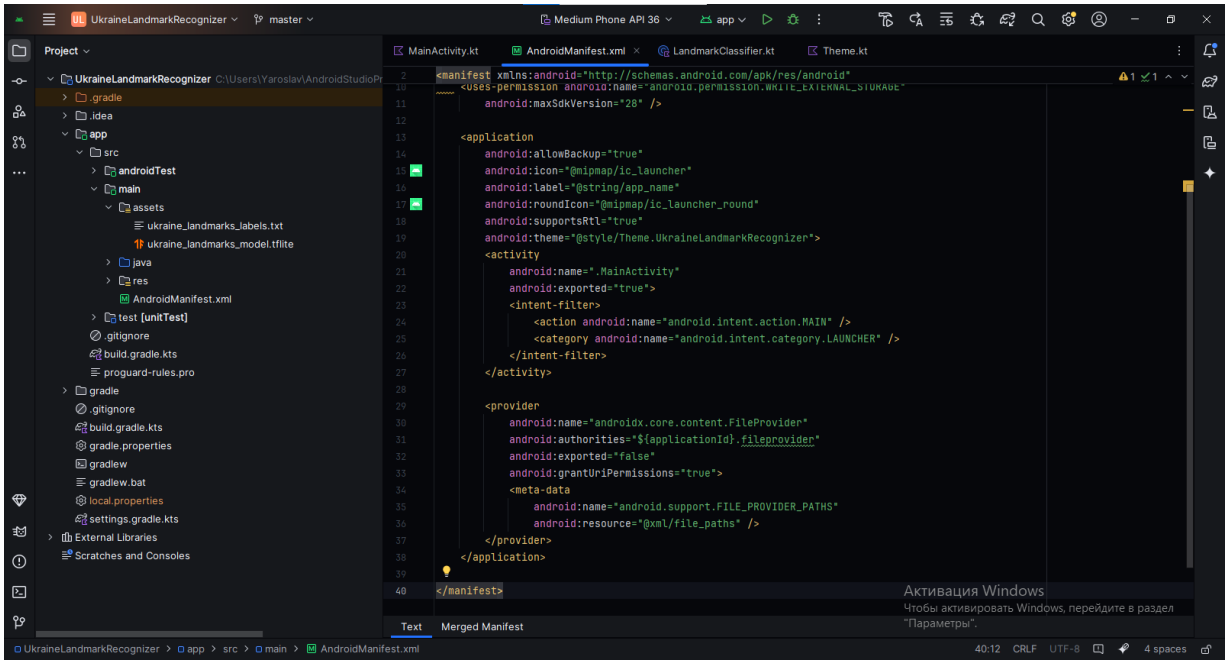


Рисунок 2.7 – Інтерфейс середовища розробки Android Studio

Було реалізовано простий та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачеві зробити фото або обрати його з галереї, запустити розпізнавання та отримати результат із назвою пам’ятки. Усі обчислення відбуваються локально, що забезпечує швидкий відгук системи та мінімізує використання ресурсів пристрою [30].

Результати реалізації методики підтвердили її ефективність: мобільний застосунок коректно визначає більшість архітектурних об’єктів, навіть якщо вони зняті під різним кутом або на фоні інших будівель. У перспективі можливе розширення кількості класів, доповнення бази пам’яток новими об’єктами та використання мультимодальних даних, таких як текстові підказки або аудіогіди. Таким чином, розроблена методика є універсальною основою для створення розумних туристичних гідів, освітніх інструментів та культурних платформ у цифровій формі.

2.3 Моделювання структури мобільного застосунку для розпізнавання архітектурних пам’яток України на зображеннях

Процес створення мобільного застосунку вимагає попереднього проєктування його структури та поведінки, що дозволяє формалізувати функціональні вимоги до системи, оптимізувати архітектурні рішення та уніфікувати комунікацію між окремими компонентами.

Для візуалізації основних аспектів взаємодії користувача із системою та міжкомпонентної логіки роботи застосунку були побудовані діаграми в нотації UML (Unified Modeling Language):

– діаграма варіантів використання (use case);

– діаграма послідовностей (sequence diagram);

– діаграма кооперації (collaboration diagram).

Використання цих діаграм дозволяє чітко відобразити функціональність і внутрішню структуру системи на етапі проєктування.

На діаграмі варіантів використання відображено взаємодію користувача із мобільним застосунком. Основними діючими особами є клієнт та система розпізнавання архітектурних пам’яток України.

Застосунок надає користувачу функціональні можливості, зокрема: зйомку зображення або вибір його з галереї, запуск процесу розпізнавання архітектурної пам’ятки, а також отримання та перегляд результату класифікації. Діаграма чітко ілюструє логіку поведінки системи у відповідь на дії користувача, визначаючи основні сценарії використання.

Діаграма варіантів використання зображена на рисунку 2.8.

Діаграма послідовностей описує часову послідовність взаємодії між об’єктами системи під час виконання ключового сценарію – розпізнавання архітектурного об’єкта. Тут показано, як користувач ініціює захоплення зображення, яке передається до модуля інтерфейсу машинного навчання, де відбувається інференс моделі (виведення результату класифікації).

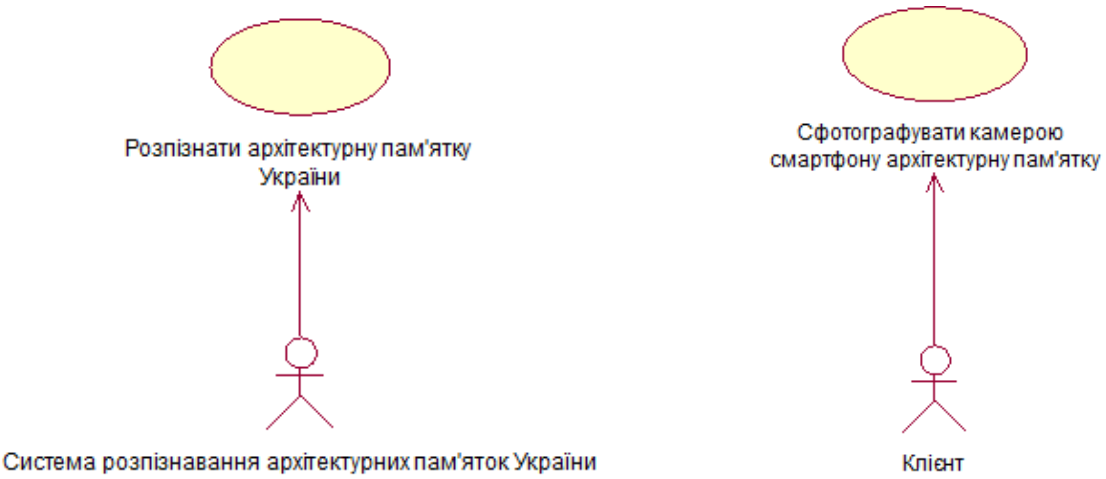


Рисунок 2.8 – Діаграма варіантів використання мобільного застосунку для розпізнавання архітектурних пам’яток України

Після завершення обробки результат передається інтерфейсу користувача для виводу на екран. Дана діаграма дозволяє оцінити порядок викликів, асинхронність деяких процесів, а також важливі точки обміну даними між модулями.

Діаграма послідовностей зображена на рисунку 2.9.

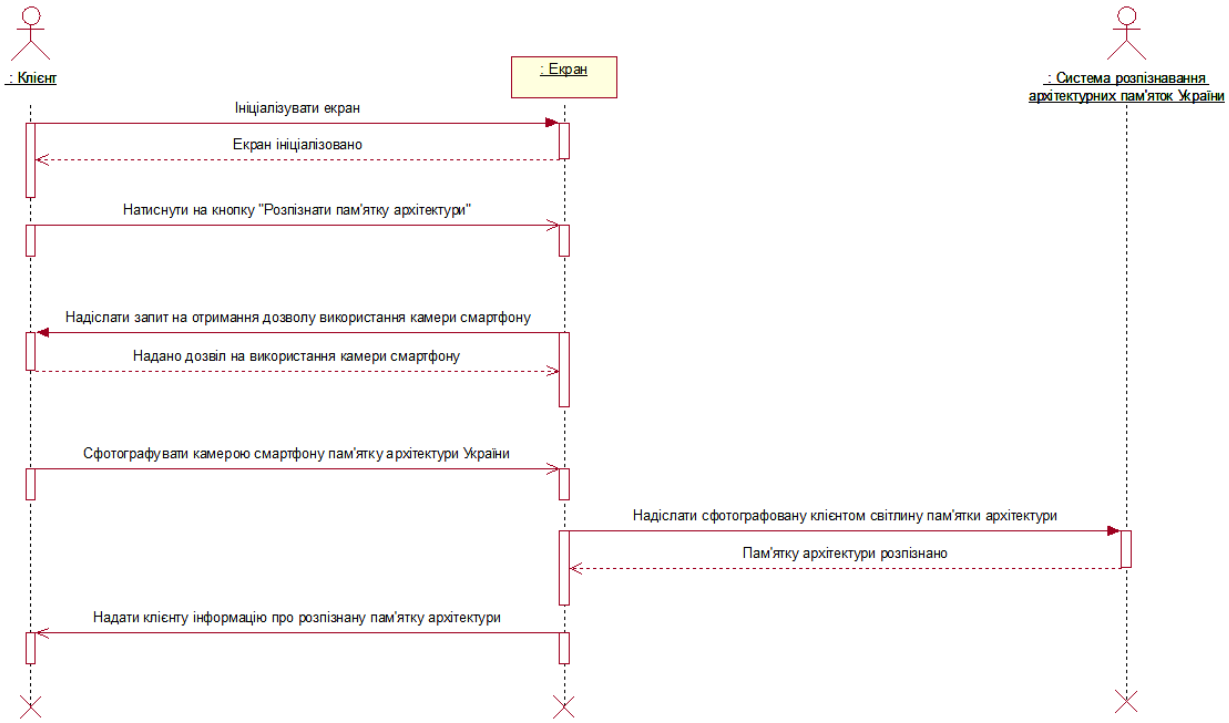


Рисунок 2.9 – Діаграма послідовностей мобільного застосунку для розпізнавання архітектурних пам’яток України

Діаграма кооперації деталізує взаємодію між структурними компонентами системи, підкреслюючи їхні ролі в загальній архітектурі.

Кооперація між об’єктами на діаграмі реалізується через обмін повідомленнями, який дозволяє зрозуміти, як саме реалізується логіка застосунку, які об’єкти комунікують безпосередньо, а які – опосередковано. Такий підхід дозволяє виявити можливі вузькі місця або залежності між частинами системи ще до її реалізації.

Діаграма кооперації відображена на рисунку 2.10.

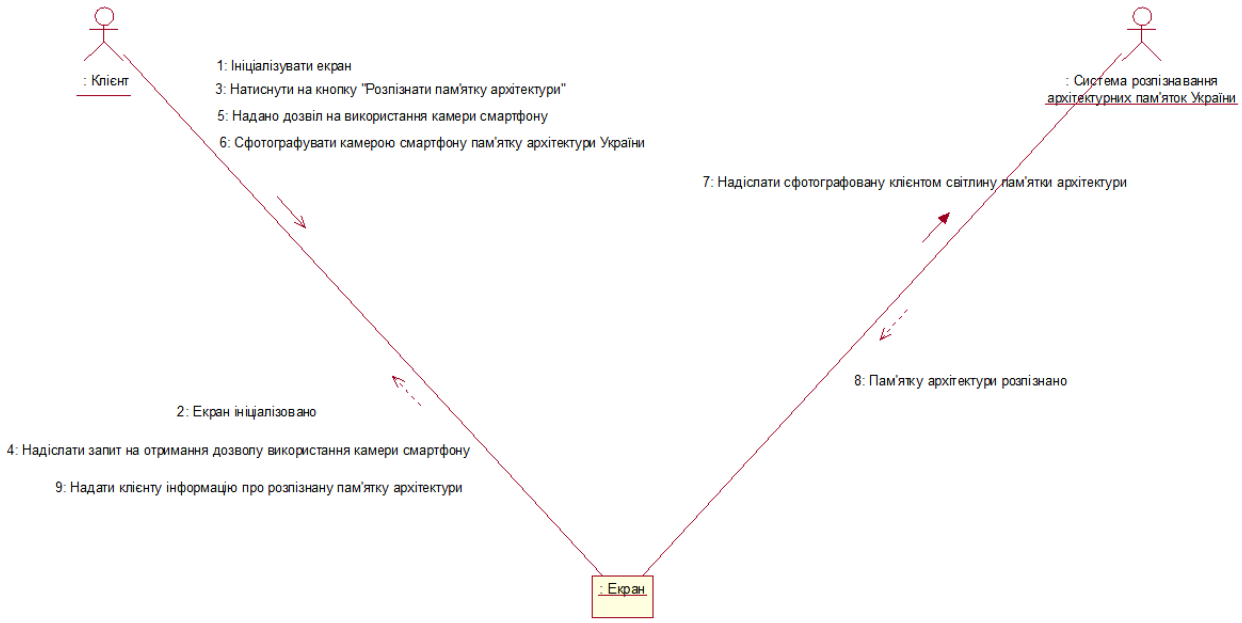


Рисунок 2.10 – Діаграма кооперації мобільного застосунку для розпізнавання архітектурних пам’яток України

3 ЕТАПИ РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ АРХІТЕКТУРНИХ ПАМ'ЯТОК УКРАЇНИ НА ЗОБРАЖЕННЯХ

3.1 Вибір інструментальних засобів реалізації мобільного застосунку для розпізнавання архітектурних пам'яток України на зображеннях

Для реалізації мобільного застосунку для розпізнавання архітектурних пам'яток України на зображеннях було обрано середовище Android Studio. Це офіційне інтегроване середовище розробки для створення застосунків під операційну систему Android. Середовище Android Studio було представлене компанією Google у 2013 році та є основною платформою для створення, тестування й розгортання Android-застосунків як на телефонах, планшетах, розумних годинах, так і на телевізорах з Android TV.

Основні можливості Android Studio:

– повна підтримка мови програмування Kotlin. Android Studio надає повну підтримку для Kotlin – рекомендованої мови програмування для створення застосунків для пристроїв з операційною системою Android. Включено підсвітку синтаксису, автодоповнення, рефакторинг, інтеграцію з системою автоматичної збірки Gradle та налагодження;

– наявність Android Emulator. Інструмент, який дозволяє запускати та тестувати застосунок на віртуальному пристрої, не маючи під рукою реального смартфона. Емулятор підтримує кілька конфігурацій пристроїв і Android-версій;

– наявність інструментів UI Designer і Compose Preview. Android Studio підтримує візуальне редагування XML-інтерфейсу, а також візуалізацію інтерфейсів, написаних через декларативну структуру інтерфейсу користувача з відкритим кодом Jetpack Compose. Анотація Compose Preview дозволяє побачити, як виглядатиме інтерфейс програмного застосунку у реальному часі;

– інтеграція з Gradle. Система збірки Gradle дозволяє гнучко керувати залежностями, налаштовувати типи збірок (debug, release), здійснювати обфускацію коду через утиліту командного рядка із відкритим програмним кодом ProGuard, а також додавати скрипти для автогенерації коду або ресурсу;

– наявність інструментів для налагодження та відладки коду. Android Studio містить зручну систему зупинок виконання коду, перегляду значень змінних, аналізу стеку викликів, перевірки стану оперативної пам’яті під час виконання коду програми, а також ведення журналу повідомлень через засіб читання і запису Logcat. Це дає змогу ефективно знаходити й виправляти помилки в роботі застосунку у реальному часі;

– наявність інструментів продуктивності. Android Profiler дозволяє відстежувати споживання ресурсу центрального процесору, графічного процесору, пам’яті пристрою, а також використання мережі. Це корисно для оптимізації продуктивності, особливо для застосунків із машинним навчанням;

– підтримка використання модульної структури. Android Studio дозволяє розбити застосунок на модулі (модуль app, модуль бібліотеки, модуль для тестування тощо), що сприяє масштабуванню та повторному використанню коду;

– інтеграція з платформою розробки мобільних застосунків Firebase та ML Kit. Середовище забезпечує пряме підключення до сервісів Google, таких як Firebase (аналітика, автентифікація, хмарне збереження), а також ML Kit – набір інструментів машинного навчання;

– інтеграція з Git та іншими системами керування версіями. Android Studio підтримує Git, GitHub, SVN та інші системи контролю версій. Це дозволяє вести колективну розробку, зберігати історію змін, створювати гілки, виконувати злиття тощо;

– інтеграція з Jetpack. Jetpack – це набір бібліотек Android, таких як LiveData, ViewModel, Navigation, Compose, CameraX, Room тощо, які полегшують і прискорюють розробку мобільних засосунків. Android Studio має вбудовану підтримку для всіх компонентів набору бібліотек Jetpack.

Програмування мобільного застосунку для розпізнавання архітектурних пам’яток України здійснювалося мовою програмування Kotlin. Kotlin – це сучасна, статично типізована мова програмування, розроблена компанією JetBrains і офіційно підтримувана Google як основна мова для розробки під пристрої з операційною системою Android. Kotlin створено як більш лаконічну, безпечну й виразну альтернативу мові Java, з повною сумісністю з існуючим Java-кодом.

Основні характеристики Kotlin:

– лаконічність у порівнянні з іншими мовами програмування. У порівнянні з Java, код на Kotlin зазвичай на 30%–40% коротший, що досягається завдяки розширенням, виводом типів, виразним функціям та скороченим конструкціям;

– безпека щодо null-посилань. Kotlin включає механізм null-безпеки, який зменшує ризик виникнення помилок типу NullPointerException на етапі виконання. Наприклад, розділення типів на nullable та non-nullable;

– повна сумісність із мовою програмування Java. Kotlin може використовувати будь-які існуючі Java-бібліотеки або фреймворки, а також бути інтегрованим у вже існуючі Java-проєкти;

– розширені функціональні можливості. Підтримка лямбда-виразів, високорівневих функцій, колекційних операцій, функцій як об’єктів – усе це робить Kotlin дуже зручним для написання реактивного та декларативного коду (зокрема з Jetpack Compose);

– можливість роботи із співпрограмами та співпроцедурами. Kotlin підтримує співпрограми – легковагові потоки, які дозволяють писати асинхронний код просто і читабельно. Це особливо корисно в операційні системі Android для операцій з мережею, введенням або виведенням і машинним навчанням;

– наявність розширення (extension functions). Розширення мови Kotlin дають змогу додавати нову функціональність до існуючих класів без їх модифікації. Наприклад, розширення для сервісів ImageProxy або Bitmap;

– інтероперабельність з інтерфейсом прикладного програмування Android. Усі Android класи доступні в Kotlin без додаткової обгортки, при цьому інтерфейси прикладного програмування Jetpack (ViewModel, LiveData, Compose) мають Kotlin-орієнтовані реалізації;

– підтримка середовища Android Studio. Android Studio має вбудовану підтримку Kotlin: автодоповнення, налагодження, аналіз коду, рефакторинг, компіляція, перевірка помилок тощо.

У процесі розробки мобільного застосунку для розпізнавання архітектурних пам’яток України було використано низку сучасних бібліотек, які істотно полегшують реалізацію функціональності, пришвидшують розробку та забезпечують інтеграцію новітніх технологій Android та машинного навчання.

Основою графічного інтерфейсу став фреймворк Jetpack Compose – декларативна бібліотека для створення дизайну користувацького інтерфейсу, розроблена компанією Google. Вона дозволяє описувати зовнішній вигляд елементів інтерфейсу в Kotlin-коді без використання розширюваної мови розмітки (EXtensible Markup Language, XML), що спрощує підтримку й масштабування застосунку.

За допомогою декларативного користувацького інтерфейсу Jetpack Compose було реалізовано основний екран, кнопки взаємодії, картки результатів класифікації та інші компоненти [31].

Для інтеграції камери в реальному часі було використано бібліотеку CameraX. Це сучасний інтерфейс прикладного програмування, який входить до складу набору бібліотек та інструментів Android Jetpack і надає спрощений та уніфікований доступ до апаратного забезпечення камери. CameraX автоматично адаптується до різних пристроїв та версій Android, забезпечуючи стабільну роботу камери, підтримку прев’ю, захоплення зображень та їх обробку у вигляді потоків. Зокрема, у проєкті використано класи Preview, ImageCapture, ImageAnalysis та PreviewView.

Ключовою для реалізації машинного навчання стала бібліотека TensorFlow Lite – спеціалізований інструмент для виконання моделей глибокого навчання на мобільних пристроях. Зокрема, в проєкті використано бібліотеки tensorflow-lite, tensorflow-lite-support, а також tensorflow-lite-task-vision. Вони забезпечують доступ до основних компонентів машинного навчання, роботу з тензорами, масштабування зображень, обробку результатів класифікації та інтеграцію моделей у форматі .tflite.

Також використано бібліотеку Coil (Coil Compose) – легковаговий фреймворк для завантаження та відображення зображень у дизайні користувацького інтерфейсу Compose [32]. Хоча застосунок переважно працює з локальними зображеннями, бібліотека Coil забезпечує зручну інтеграцію із вбудованим компонентом Image у фреймворці Compose і підтримує відображення низькорівневого графічного подання зображення Bitmap у вигляді представлення ImageBitmap.

Крім того, було підключено стандартні бібліотеки Jetpack: androidx.core, androidx.lifecycle, androidx.activity, а також бібліотеки для юніт-тестування (junit) і UI-тестування (androidx.ui.test.junit4). Ці компоненти відповідають за життєвий цикл активностей, обробку дозволів, підтримку тем оформлення та інші базові функції Android-застосунків.

Завдяки гармонійній взаємодії зазначених бібліотек стало можливим створити сучасний, швидкий, стабільний і зручний у використанні мобільний застосунок, здатний ефективно працювати з камерою смартфону, класифікувати зображення та відображати результати без затримок і складностей для користувача.

У розробці мобільного застосунку для розпізнавання архітектурних пам’яток України було використано фреймворк машинного навчання TensorFlow, який є одним із найпопулярніших інструментів для створення моделей штучного інтелекту. Фреймворк TensorFlow був розроблений компанією Google і з моменту свого запуску у 2015 році став одним з основних інструментів у сфері глибокого навчання та обробки даних. Його потужність, універсальність і підтримка великої спільноти зробили його лідером серед платформ для реалізації нейронних мереж.

Оскільки застосунок розроблявся для мобільної платформи Android, було використано оптимізовану версію фреймворку TensorFlow під назвою TensorFlow Lite. Цей інструментарій спеціально створений для мобільних і вбудованих пристроїв, які мають обмежені обчислювальні ресурси, невеликий обсяг оперативної пам’яті та потребують економного енергоспоживання. TensorFlow Lite забезпечує можливість виконання інференсу – процесу отримання передбачень моделі безпосередньо на пристрої, без підключення до інтернету або зовнішніх серверів. Завдяки цьому досягається висока швидкість роботи та повна автономність застосунку.

У даному проєкті модель згорткової нейронної мережі була попередньо навчена з використанням фреймворку TensorFlow у середовищі Python, після чого збережена у форматі .tflite, придатному для використання на Android-пристроях. Вхідними даними для моделі є зображення у форматі RGB (Red, Green, Blue) розміром 224×224 пікселі, які попередньо масштабуються та нормалізуються у мобільному застосунку. Результатом роботи моделі є набір числових значень – ймовірностей належності зображення до кожного з можливих класів. У цьому випадку класи відповідають 18 найвідомішим архітектурним пам’яткам України.

Для виконання інференсу в Kotlin-коді застосовується інтерфейс Interpreter з пакету org.tensorflow.lite.Interpreter. У застосунку цей інтерпретатор створюється один раз під час запуску і працює з буферами вхідних і вихідних тензорів. Зображення користувача перетворюється на тензор (ByteBuffer) і подається на вхід моделі, після чого оброблений результат зчитується з вихідного буфера. Найімовірніша категорія вибирається як кінцевий результат, що демонструється користувачеві на екрані.

Використання фреймворку TensorFlow Lite має низку переваг.

По-перше, це швидке виконання навіть на пристроях середнього рівня, що критично важливо для взаємодії в реальному часі.

По-друге, обробка даних відбувається повністю локально на пристрої, що гарантує конфіденційність персональної інформації.

По-третє, формат .tflite дозволяє зменшити обсяг моделі, що сприяє економії ресурсів і пришвидшує завантаження.

У проєкті також передбачено можливість подальшої оптимізації моделі, наприклад, через квантування (перетворення чисел з формату float32 до int8), що дозволить ще більше зменшити розмір і пришвидшити виконання без значної втрати точності.

Отже, фреймворк TensorFlow Lite виступає ключовим компонентом розробленого мобільного застосунку. Він забезпечує ефективну інтеграцію глибокого навчання в Android-застосунок, дозволяючи реалізувати складну задачу комп'ютерного зору – розпізнавання архітектурних об'єктів без необхідності у складних серверних рішеннях або підключенні до Інтернету. Це робить застосунок універсальним, зручним для кінцевого користувача і таким, що працює стабільно в автономному режимі.

3.2 Етапи програмної реалізації розпізнавання архітектурних пам'яток України на зображеннях

Розробка мобільного застосунку для розпізнавання архітектурних пам'яток України на зображеннях є багатоступеневим процесом, який поєднує в собі машинне навчання, роботу з камерою пристрою, обробку зображень, створення користувацького інтерфейсу та програмну логіку на мові Kotlin у середовищі Android Studio. Програмна реалізація передбачала не лише створення мобільного застосунку, але й попередню підготовку даних і навчання моделі нейронної мережі.

На першому етапі було здійснено збір і підготовку зображень архітектурних пам'яток України. Було сформовано навчальну вибірку, що включала зображення 18 різних архітектурних об'єктів України.

Архітектурні пам’ятки України, які розпізнає створений мобільний застосунок:

– Андріївська церква м. Київ;

– Благовіщенський собор м. Харків;

– Будинок із химерами м. Київ;

– Держпром м. Харків;

– Дзеркальний струмінь м. Харків;

– Замок Любарта м. Луцьк;

– Золоті ворота м. Київ;

– Кам’янець-Подільська фортеця м. Кам’янець-Подільський;

– Успенський собор Києво-Печерської Лаври м. Київ;

– Львівська Національна Опера м. Львів;

– Маріїнський палац м. Київ;

– Михайлівський Золотоверхий собор м. Київ;

– Оперний театр м. Одеса;

– Полтавський краєзнавчий музей м. Полтава;

– Собор Святого Юра м. Львів;

– Софійський собор м. Київ;

– ХНАТОБ м. Харків;

– Чернівецький Національний університет м. Чернівці.

Для кожного класу було зібрано від 100 до 200 зображень у різних ракурсах, освітленні та з різною якістю. Зображення були нормалізовані за розміром, масштабовані до 224×224 пікселі та збережені у відповідній структурі директорій.

Далі було створено модель згорткової нейронної мережі на базі архітектури MobileNet або схожої полегшеної структури, що дозволяє ефективно працювати на мобільних пристроях. Модель було навчено за допомогою бібліотек TensorFlow та Keras у середовищі Python.

Keras – це високорівнева бібліотека глибокого навчання з відкритим вихідним кодом, написана на мові програмування Python.

Вона надає зручний і зрозумілий інтерфейс для побудови, навчання й оцінювання штучних нейронних мереж. Keras є частиною екосистеми TensorFlow та з 2017 року офіційно інтегрована як її модуль. Завдяки простому та інтуїтивному синтаксису, Keras дозволяє швидко експериментувати з різними архітектурами нейронних мереж, що робить її ідеальним вибором як для початківців, так і для досвідчених розробників у галузі штучного інтелекту [33].

Однією з ключових переваг Keras є її модульність. Будь-яку модель можна будувати з окремих шарів (наприклад, Conv2D, MaxPooling2D, Dense, Dropout), об’єднуючи їх послідовно або за допомогою функціонального інтерфейсу прикладного програмування. Такий підхід надає розробнику гнучкість у створенні як простих згорткових мереж для класифікації зображень, так і складних архітектур з кількома входами, виходами або гілками .

У межах даного проєкту Keras використовувався для створення та навчання згорткової нейронної мережі, яка виконує класифікацію зображень архітектурних пам’яток України. Було реалізовано послідовну модель, яка містить шари згортки, пулінгу, нормалізації, регуляризації та повнозв’язні шари. Завдяки модулю ImageDataGenerator було забезпечено автоматичне зчитування зображень із директорій, їх масштабування та аугментацію (обертання, зсув), що дозволяє зменшити перенавчання та покращити узагальнюючу здатність моделі.

Навчання моделі відбувалося з використанням оптимізатора Adam, функції втрат категорійної кросентропії, а також метрики точності. У процесі тренування застосовувались зворотні виклики – рання зупинка та збереження найкращої моделі на основі значення валідаційної точності.

Після навчання модель, побудовану у Keras, було конвертовано у формат TensorFlow Lite за допомогою методу tf.lite.TFLiteConverter.from_keras_model().

Таким чином, Keras виступив не лише як інструмент для проєктування і навчання мережі, але й як перехідний етап між підготовкою моделі на комп’ютері та її розгортанням у мобільному застосунку на Android.

Завдяки Keras процес створення моделі стає прозорим, логічним і добре структурованим. Наявність розширеної документації, прикладів, активної спільноти та інтеграція із середовищем Google Colab дозволяє значно спростити процес розробки систем розпізнавання образів, зокрема таких, як класифікація архітектурних пам’яток за зображенням.

Процес навчання включав поділ даних на навчальну, валідаційну та тестову вибірки, компіляцію моделі з оптимізатором Adam, використання функції втрат категорійної кросентропії, а також моніторинг точності та втрат на кожній епосі. Було реалізовано ранню зупинку навчання для запобігання перенавчанню. Після завершення навчання модель досягла точності 73,43% на тестовій вибірці, що є задовільним результатом для задачі багатокласової класифікації з 18 категоріями.

Після навчання модель було експортовано у формат .tflite – TensorFlow Lite, який оптимізовано для роботи на мобільних пристроях. Було також збережено файл міток (ukraine_landmarks_labels.txt), який містить назви відповідних класів у тому ж порядку, в якому вони були представлені в моделі.

Наступним етапом стала розробка самого мобільного застосунку в середовищі Android Studio. Спершу було створено новий проєкт із використанням шаблону Empty Compose Activity.

Empty Compose Activity – це стандартний шаблон проєкту, який надається в середовищі розробки Android Studio для створення застосунків із використанням Jetpack Compose як основного фреймворку для побудови інтерфейсу користувача. Цей шаблон дозволяє розпочати розробку з повністю налаштованим середовищем, без необхідності створювати структуру вручну.

Процес вибору шаблону Empty Compose Activity в середовищі розроблення Android Studio показано на рисунку 3.1.

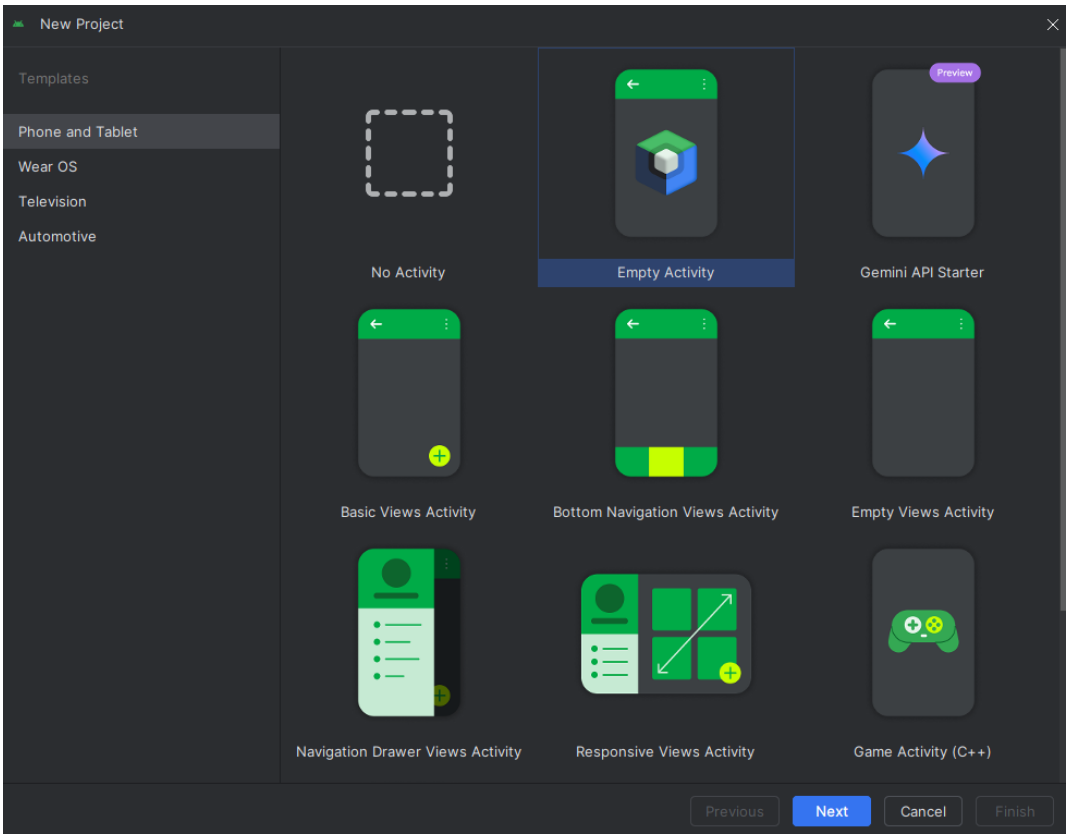


Рисунок 3.1 – Вибір шаблону Empty Compose Activity в середовищі Android Studio

У файлі build.gradle.kts (рівень модуля app) було підключено всі необхідні залежності, що забезпечують повноцінне функціонування застосунку. Зокрема, для побудови інтерфейсу користувача було додано бібліотеки Jetpack Compose, які надають інструменти для декларативного програмування дизайну інтерфейсу користувача на мові Kotlin, підтримку Material Design 3, типографіки, темної та світлої теми та анімацій.

Для реалізації роботи з камерою було підключено компоненти CameraX, включаючи модулі camera-core, camera-camera2, camera-lifecycle та camera-view, які спрощують ініціалізацію камери, відображення попереднього перегляду (PreviewView) та захоплення зображень (ImageCapture) у стабільний і кросплатформовий спосіб.

Основні залежності, додані до файлу build.gradle.kts (рівень модуля app) показано на рисунку 3.2.

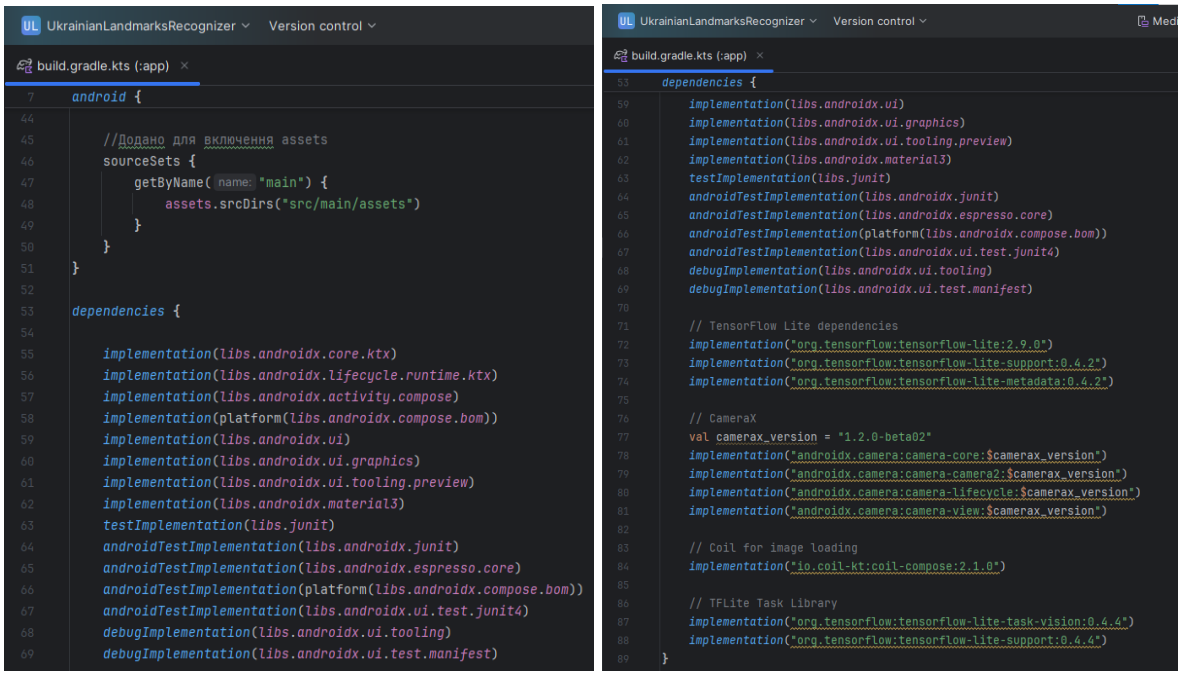


Рисунок 3.2 – Основні залежності, додані до файлу build.gradle.kts (рівень модуля app)

Каталог assets було створено для розміщення моделі та списку міток, з відповідним налаштуванням noCompress для форматів .tflite і .txt. Розташування і вміст папки assets показано на рисунку 3.3.

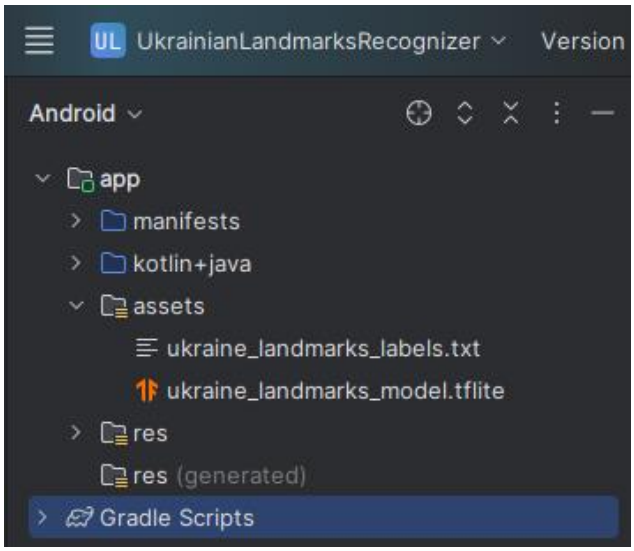


Рисунок 3.3 – Розташування і вміст каталогу assets

Основні модулі проєкту, в яких реалізована функціональність застосунку (класи ImageUtils.kt, LandmarkRecognizer.kt та файл MainActivity.kt) розміщено в пакеті com.example.ukrainianlandmarksrecognizer. Розташування основних класів застосунку продемонстровано на рисунку 3.4.

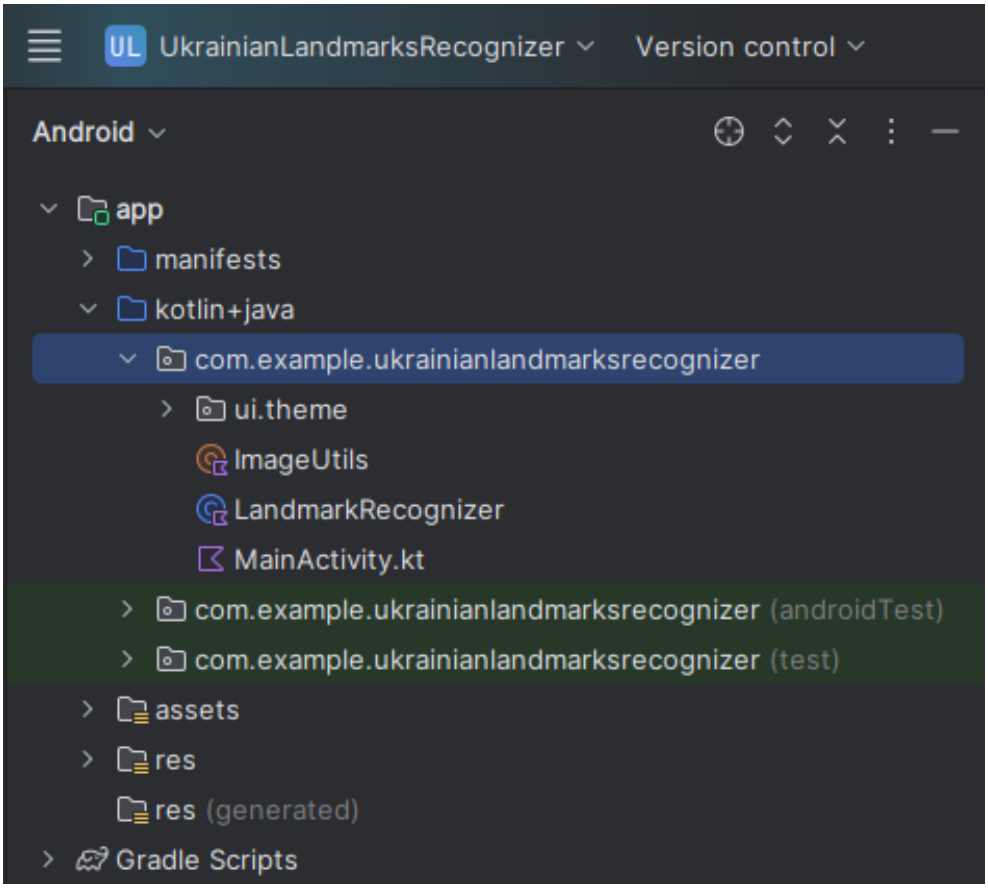


Рисунок 3.4 – Розташування основних класів застосунку

Уся логіка розпізнавання реалізована в класі LandmarkRecognizer.

Під час ініціалізації застосунку завантажується модель ukraine_landmarks_model.tflite, створюється інтерпретатор TensorFlow Lite, а також зчитується файл з мітками. Метод classifyImage (bitmap: Bitmap) приймає зображення, масштабує його до необхідного розміру, нормалізує значення пікселів і перетворює його на тензор у форматі ByteBuffer.

Після запуску інференсу вихідний буфер з імовірностями зчитується, і вибирається клас з найвищою ймовірністю, після чого повертається його мітка та відсоток достовірності.

Клас LandmarkRecognizer є центральним компонентом застосунку, що відповідає за завантаження моделі машинного навчання у форматі TensorFlow Lite, її ініціалізацію та виконання класифікації зображень архітектурних пам’яток України. Цей клас реалізує повний цикл інференсу: від підготовки вхідного зображення до повернення результату розпізнавання у вигляді текстової мітки з відповідною ймовірністю.

Під час створення об’єкта класу LandmarkRecognizer у його конструкторі відбувається ініціалізація інтерпретатора моделі Interpreter, що імпортується з пакету org.tensorflow.lite.Interpreter. Модель у форматі .tflite завантажується з каталогу assets за допомогою функції FileUtil.loadMappedFile(), що забезпечує швидкий доступ до вмісту файлу у вигляді пам’яті, відображеної на файл (memory-mapped file).

Одразу після завантаження моделі зчитується форма вхідного тензора (inputTensor.shape()), що дозволяє автоматично визначити необхідні розміри вхідного зображення: ширину, висоту та кількість каналів (як правило, три для RGB). Це гарантує, що модель отримає вхідні дані у правильному форматі. Паралельно завантажуються текстові мітки з файлу ukraine_landmarks_labels.txt, які відповідають класам архітектурних пам’яток.

Клас також містить механізми обробки винятків: у разі помилки (наприклад, неправильний розмір вхідного зображення або неможливість завантажити файл) виводиться повідомлення до системного журналу (Log.e) та повертається повідомлення «Помилка» або «Невідома пам’ятка».

Клас LandmarkRecognizer є повністю автономним: після завантаження моделі не потребує підключення до Інтернету або зовнішніх інтерфейсів прикладного програмування. Завдяки цьому він може бути використаний у будь-яких умовах, забезпечуючи миттєву класифікацію зображень прямо на пристрої. Такий підхід також підвищує конфіденційність користувача, оскільки обробка зображень відбувається локально.

Завдяки чітко розділеній відповідальності, клас легко тестується, масштабується (наприклад, для заміни моделі або зміни кількості класів) та використовується повторно у будь-яких активностях або компонентах Compose-інтерфейсу.

Саме LandmarkRecognizer забезпечує основну функціональність мобільного застосунку – розпізнавання архітектурних об’єктів на зображеннях у реальному часі.

Для роботи з камерою використовується компонент CameraPreview, побудований за допомогою бібліотеки CameraX. Інтерфейс містить PreviewView, на якому виводиться зображення з камери в реальному часі. При натисканні кнопки «Фото» викликається метод takePicture(), який захоплює кадр і повертає об’єкт ImageProxy. Зображення з ImageProxy конвертується у формат Bitmap за допомогою класу ImageUtils, де воно попередньо перетворюється з формату YUV у формат JPEG, декодується у Bitmap, і, за необхідності, повертається у правильну орієнтацію.

Головна активність застосунку (MainActivity) реалізує логіку взаємодії з камерою та галереєю, керує інтерфейсом і результатами класифікації. Користувач може обрати зображення з галереї або зробити знімок у режимі реального часу. Після отримання зображення програма автоматично передає його до класифікатора, і виводить назву розпізнаної пам’ятки з відповідною ймовірністю.

Клас MainActivity є основною активністю мобільного застосунку, яка виконує роль точки входу, організовує логіку роботи з камерою та галереєю, координує процес розпізнавання зображень і відповідає за взаємодію з користувачем через інтерфейс, побудований за допомогою Jetpack Compose.

Після запуску застосунку клас MainActivity, як нащадок ComponentActivity, виконує перевірку дозволу на доступ до камери. У разі відсутності дозволу застосунок автоматично запитує його через компонент ActivityResultContracts.RequestPermission, що є частиною сучасної системи безпечної обробки дозволів Android.

Після отримання дозволу викликається метод setContent, який ініціалізує інтерфейс користувача з використанням теми UkrainianLandmarksRecognizerTheme та завантажує основний екран LandmarkRecognizerScreen.

У MainActivity створюється об’єкт класу LandmarkRecognizer, що реалізує основну функціональність машинного навчання: завантаження моделі, підготовку вхідного зображення, виконання інференсу та повернення результатів класифікації. Усі помилки, які можуть виникнути на цьому етапі (наприклад, відсутність файлу моделі), обробляються через блок try-catch із виведенням повідомлень у консоль (Log.e) та інформуванням користувача за допомогою Toast.

Основний інтерфейс застосунку побудовано у функції LandmarkRecognizerScreen(), яка використовує реактивну архітектуру Jetpack Compose. Тут оголошуються змінні стану recognizedLandmark, capturedImage та isCameraActive, що керують поведінкою застосунку: відображенням камери, галереї або результату класифікації. Компонент містить візуальну частину (виведення зображення або трансляція з камери), текстовий блок із результатом розпізнавання, а також дві кнопки: «Камера» та «Галерея», які дозволяють перемикатися між режимами вводу зображення.

Окремо реалізовано компонент CameraPreview, який відповідає за відображення потоку з камери у реальному часі та захоплення зображення після натискання кнопки «Фото». Зображення захоплюється через API ImageCapture бібліотеки CameraX, після чого передається у вигляді ImageProxy до функції обробки. Там зображення конвертується у формат Bitmap за допомогою допоміжного класу ImageUtils, орієнтується відповідно до положення пристрою та передається в LandmarkRecognizer для класифікації.

Також реалізовано інтеграцію з галереєю пристрою: користувач може обрати файл із зображенням, який буде автоматично декодовано у формат Bitmap (із підтримкою різних версій Android) та передано для розпізнавання. Таким чином, MainActivity забезпечує двосторонню взаємодію з користувачем: через камеру або вибір готового зображення.

Особливістю реалізації класу MainActivity є логічне розділення відповідальностей: управління станом, ініціалізація класифікатора, обробка зображень і побудова дизайну користувацького інтерфейсу реалізовані у вигляді окремих функцій.

Така структура забезпечує зручність у підтримці та розширенні коду. Наприклад, додавання нових джерел зображень або впровадження багатомовної підтримки можна реалізувати без втручання у логіку класифікації.

Завдяки поєднанню функціональних можливостей Android (робота з камерою, вибір файлів, інтерфейс користувача) та компонентів машинного навчання (LandmarkRecognizer), клас MainActivity є центральною частиною застосунку, що забезпечує основну бізнес-логіку, взаємодію з TensorFlow Lite-моделлю та керування станом інтерфейсу в режимі реального часу.

Файл ImageUtils.kt є допоміжним утилітним модулем застосунку, який відповідає за обробку й конвертацію зображень, отриманих від камери у форматі ImageProxy, до стандартного формату Bitmap, зручного для подальшого використання у нейронній мережі. Він реалізований як об’єкт object ImageUtils і містить одну основну функцію – imageProxyToBitmap(), яка є незамінною в роботі з бібліотекою CameraX.

Таким чином, ImageUtils.kt виступає важливим компонентом застосунку, що виконує низькорівневу обробку зображень і забезпечує правильну підготовку вхідних даних для згорткової нейронної мережі. Без якісного перетворення зображення у формат Bitmap модель не змогла б коректно обробити вхідні дані, а інтерфейс не міг би візуалізувати результат. Крім того, використання цього класу забезпечує стабільність і гнучкість застосунку, оскільки обробка зображень реалізована централізовано й може бути легко змінена або доповнена в разі потреби.

Файл AndroidManifest.xml є ключовим конфігураційним компонентом кожного Android-застосунку. Він визначає основні властивості застосунку, його дозволи, компоненти (активності, сервіси, ресивери), ресурси, доступ до апаратного забезпечення, точки входу, а також налаштування безпеки та сумісності. У проєкті мобільного застосунку для розпізнавання архітектурних пам’яток України цей файл було налаштовано відповідно до функціональних вимог та архітектури застосунку.

Файл AndroidManifest.xml відіграє критичну роль у правильному функціонуванні застосунку, особливо в контексті інтеграції камери, файлової системи, безпечного доступу до ресурсів і призначення точки входу. Без правильного опису дозволів і провайдерів застосунок не зміг би взаємодіяти з апаратними компонентами пристрою та системними сервісами Android.

Інтерфейс користувача створено за допомогою Jetpack Compose. Jetpack Compose – це сучасна бібліотека для побудови інтерфейсу користувача в Android-застосунках, розроблена компанією Google. Вона є частиною екосистеми Jetpack і призначена для декларативного створення UI-компонентів без необхідності використання традиційного підходу на основі XML [34].

На відміну від класичного імперативного підходу, де кожен елемент інтерфейсу описується вручну у вигляді XML-файлів і пов’язується з кодом через findViewById, у Jetpack Compose інтерфейс описується мовою Kotlin у вигляді функцій-компонентів. Такий підхід дозволяє значно зменшити обсяг коду, підвищити його читабельність та спростити логіку керування станом.

Однією з ключових переваг Jetpack Compose є реактивність: зміна стану (наприклад, вмісту поля, натиснутої кнопки чи отриманого результату класифікації) автоматично призводить до перерахування й оновлення лише тих частин інтерфейсу, які залежать від цього стану. Це реалізується за допомогою спеціальних конструкцій Kotlin, зокрема remember, mutableStateOf, derivedStateOf, які дозволяють ефективно зберігати та оновлювати змінні в життєвому циклі дизайну користувацького інтерфейсу.

Jetpack Compose підтримує повну інтеграцію з іншими компонентами Android Jetpack, такими як ViewModel, Navigation, LiveData, Hilt тощо. Це дозволяє використовувати його в сучасній архітектурі Android-застосунків без додаткових адаптацій.

Ще однією важливою перевагою є підтримка Compose Preview – інтерактивного перегляду інтерфейсу без запуску емулятора. Розробник може миттєво побачити результат змін прямо в Android Studio, що значно пришвидшує розробку.

У межах розробки мобільного застосунку для розпізнавання архітектурних пам’яток України Jetpack Compose використовувався для створення головного інтерфейсу. Зокрема, за допомогою Compose реалізовано відображення камери, перемикання між режимами «Камера» та «Галерея», відображення зображення, отриманого від користувача, результат класифікації, а також інтерактивні кнопки. Уся логіка інтерфейсу базується на станах і змінних, які автоматично оновлюються, щойно змінюється джерело зображення чи результат розпізнавання.

Jetpack Compose активно розвивається, має широку підтримку спільноти та регулярні оновлення. Він є офіційно рекомендованим способом створення дизайну користувацького інтерфейсу для Android з 2021 року. Завдяки своїй простоті, потужності та інтеграції з мовою Kotlin, Compose є найефективнішим інструментом для побудови інтерфейсу у сучасних Android-застосунках [35].

Інтерфейс застосунку адаптивний, підтримує перемикання між камерою та переглядом фото, показує зображення, результат розпізнавання, а також кнопки для зміни режиму роботи. Візуальна частина повністю інтегрована в логіку роботи та реагує на зміну станів.

Велику увагу приділено обробці помилок та стабільності. Всі критичні виклики захищено блоками try-catch. Застосунок перевіряє наявність дозволу на камеру, обробляє винятки під час зчитування файлів і захоплення зображень, а також надає користувачеві текстові повідомлення у разі помилки.

Загалом програмна реалізація базується на сучасному архітектурному підході, що поєднує компоненти Jetpack, гнучкі бібліотеки набору із засобів розробки Android та фреймворки машинного навчання. Застосунок працює повністю автономно, не потребує підключення до мережі, має інтуїтивно зрозумілий інтерфейс і виконує завдання розпізнавання архітектурних пам’яток швидко й точно. Структура проєкту дозволяє легко масштабувати функціональність – додавати нові класи, покращувати модель або впроваджувати нові джерела зображень.

3.3 Тестування розробленого мобільного застосунку та аналіз результатів

Після завершення розробки мобільного застосунку було проведено всебічне тестування його функціональних можливостей на різних пристроях з операційною системою Android. Основна мета тестування полягала у перевірці коректності роботи камери, обробки зображень, виконання інференсу за допомогою моделі TensorFlow Lite, а також загальної стабільності інтерфейсу користувача, реалізованого з використанням Jetpack Compose.

Головний екран застосунку містить зручну кнопку «Фото» для фотографування пам’ятки архітектури, кнопку «Галерея» для вибору фото з галереї мобільного пристрою, а також область для відображення результату розпізнавання. Головний екран інструментального засобу, який відкривається при запуску представлено на рисунку 3.5.

Після натискання на кнопку «Фото» в інструментальному засобі виконується передача зображення у форматі Bitmap для подальшої обробки та класифікації. На рисунку 3.6 наведено вивід результату розпізнавання архітектурної пам’ятки України – Замок Любарта з відповідним рівнем впевненості у відсотках.

Окрім розпізнавання об’єктів у реальному часі з камери, також було протестовано функціональність обробки зображень, обраних із галереї мобільного пристрою. Після натискання кнопки «Галерея» користувач має змогу обрати зображення пам’ятки архітектури з галереї смартфону. Процес вибору фотографії з галереї показано на рисунку 3.7.

Результат розпізнавання архітектурної пам’ятки України (Андріївська церква) за фотографією з галереї продемонстровано на рисунку 3.8.

На рисунку 3.9 показано результат розпізнавання пам’ятки архітектури України (Львівська опера) за фотографією з галереї мобільного пристрою з упевненістю 95,2%.

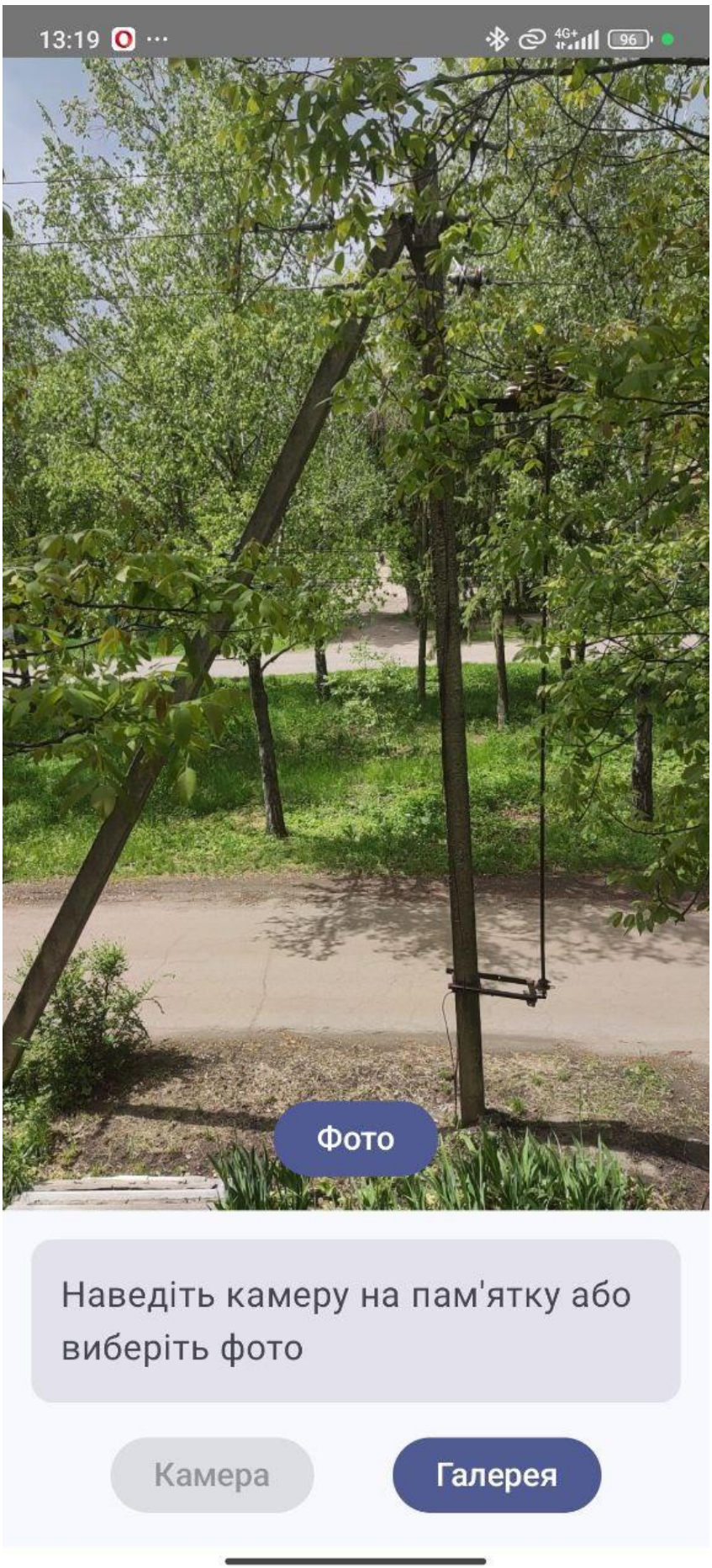


Рисунок 3.5 – Головний екран мобільного застосунку

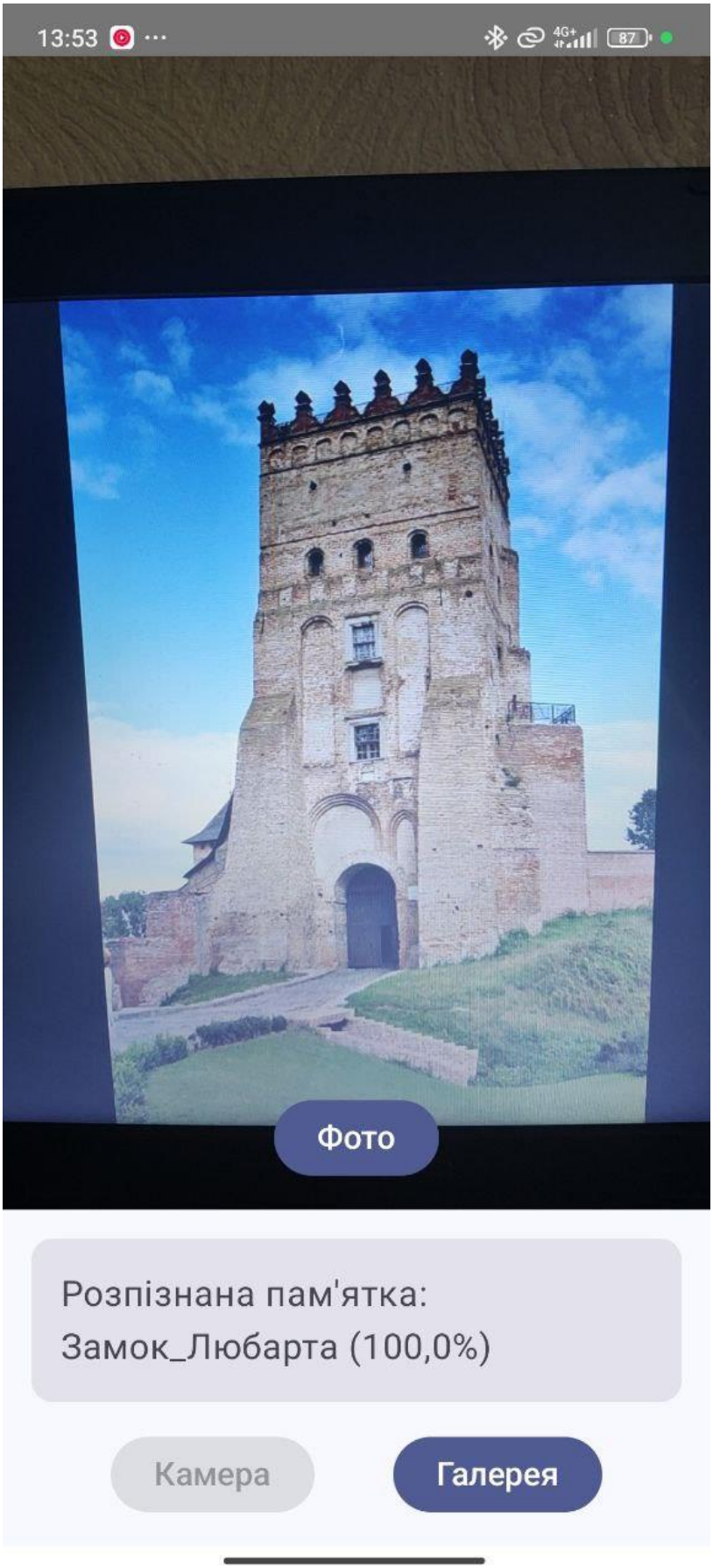


Рисунок 3.6 – Результат розпізнавання пам’ятки архітектури України за фотографією, зробленою в розробленому мобільному застосунку

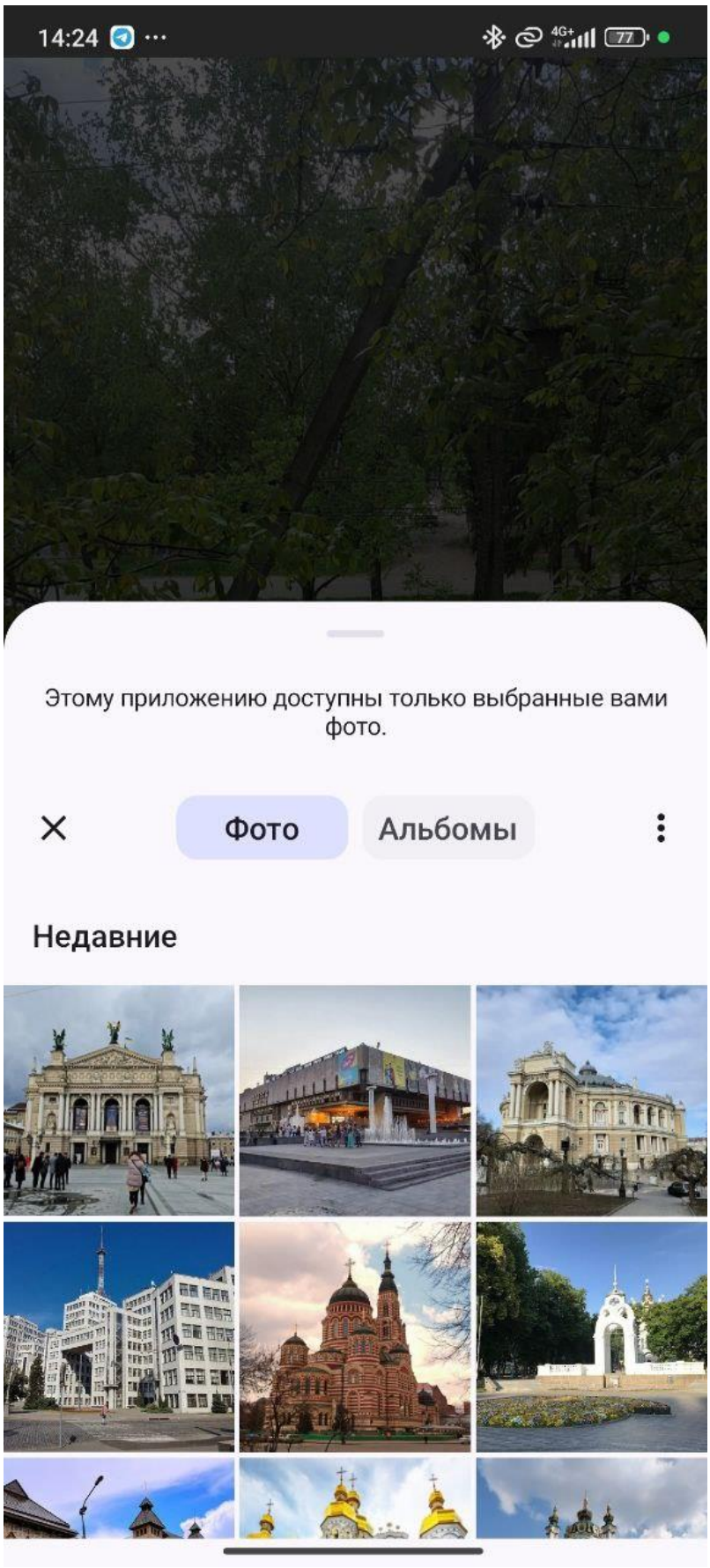


Рисунок 3.7 – Процес вибору фотографії пам’ятки архітектури з галереї для розпізнавання

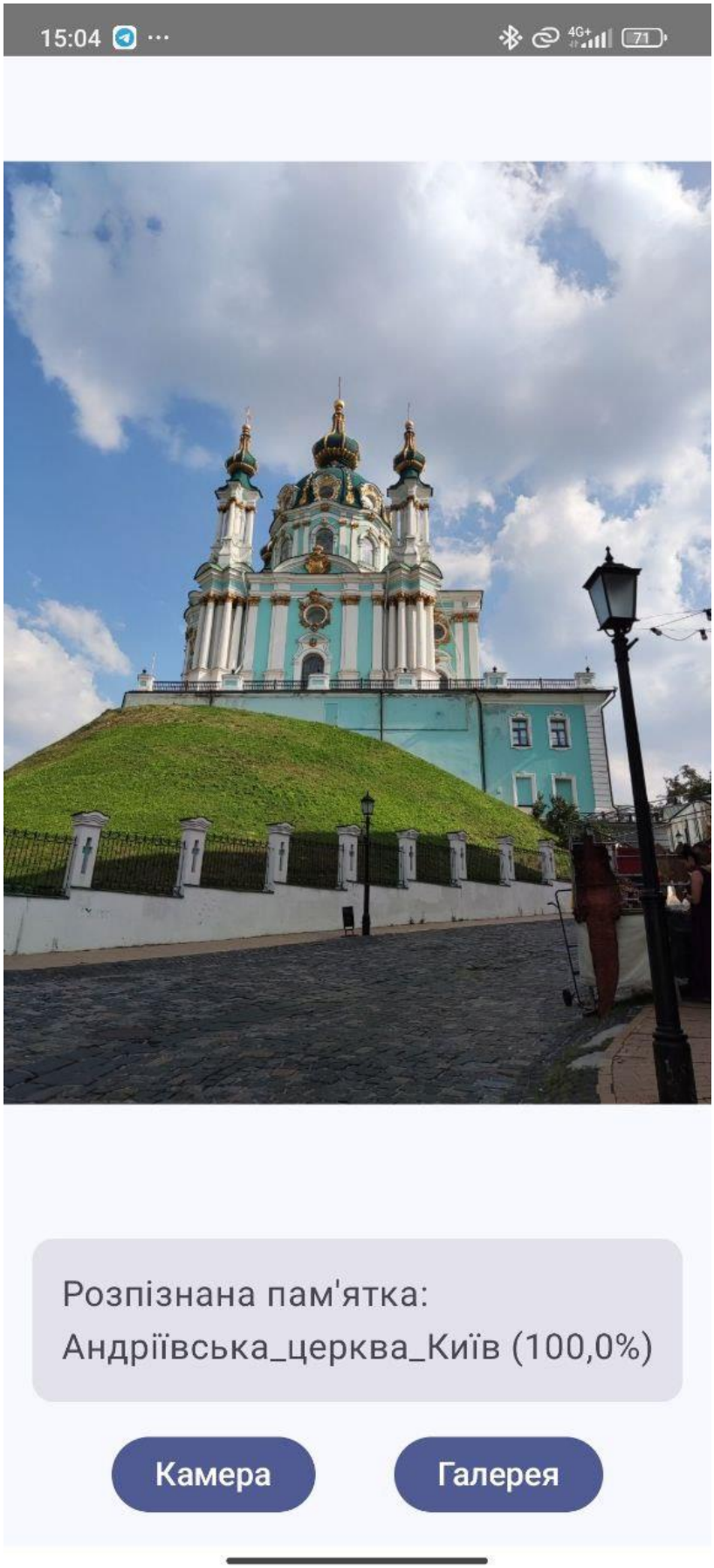


Рисунок 3.8 – Результат розпізнавання пам’ятки архітектури України за фотографією, обраною з галереї смартфону

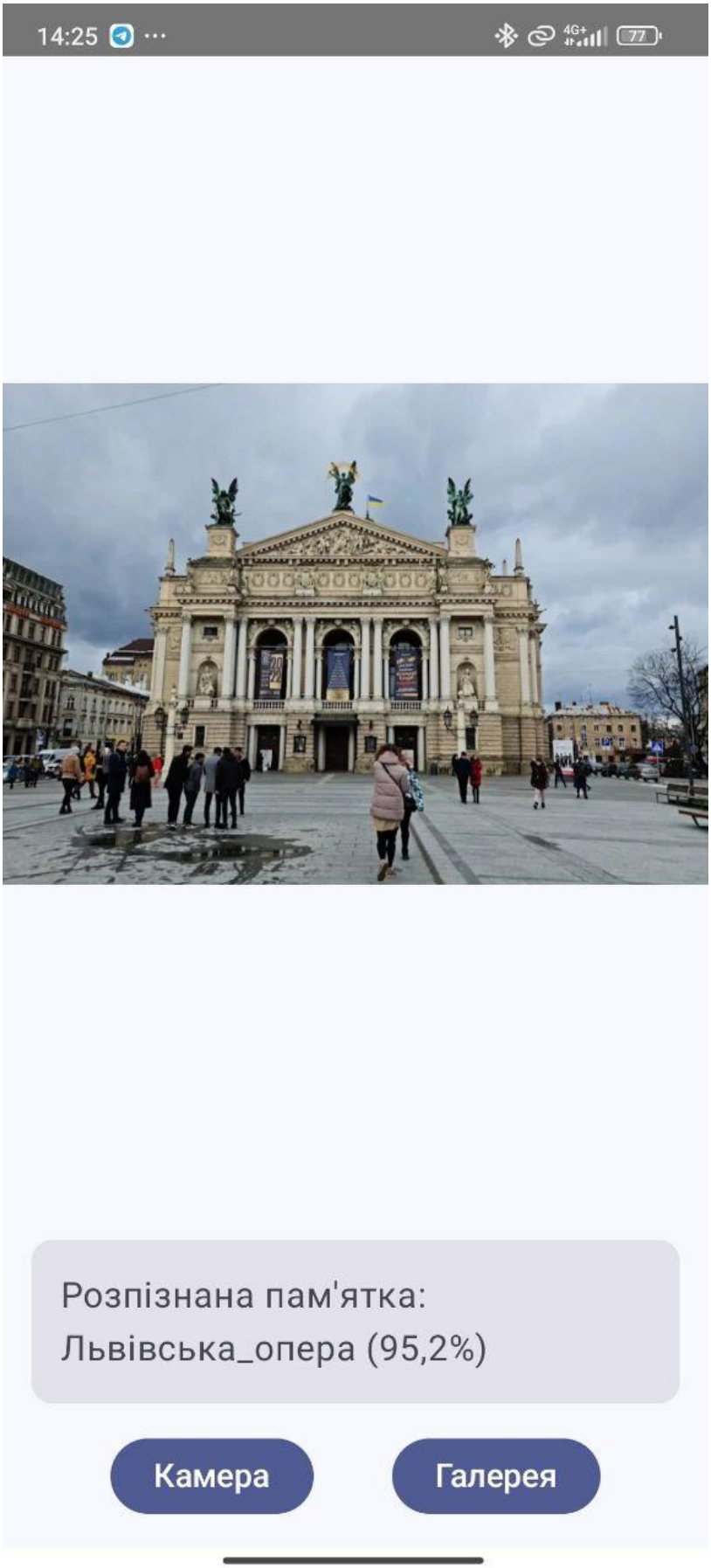


Рисунок 3.9 – Результат розпізнавання пам’ятки архітекури України з упевненістю 95,2 %

Якщо користувач спробує розпізнати пам’ятку архітектури України, розпізнавання якої не передбачене даним інструментальним засобом, буде показано відповідне повідомлення. На рисунку 3.10 показано результат розпізнавання пам’ятки архітектури України (Володимирський собор м. Київ), яка не входить в список пам’яток архітектури України, які розпізнає даний застосунок за фотографією з галереї мобільного пристрою.

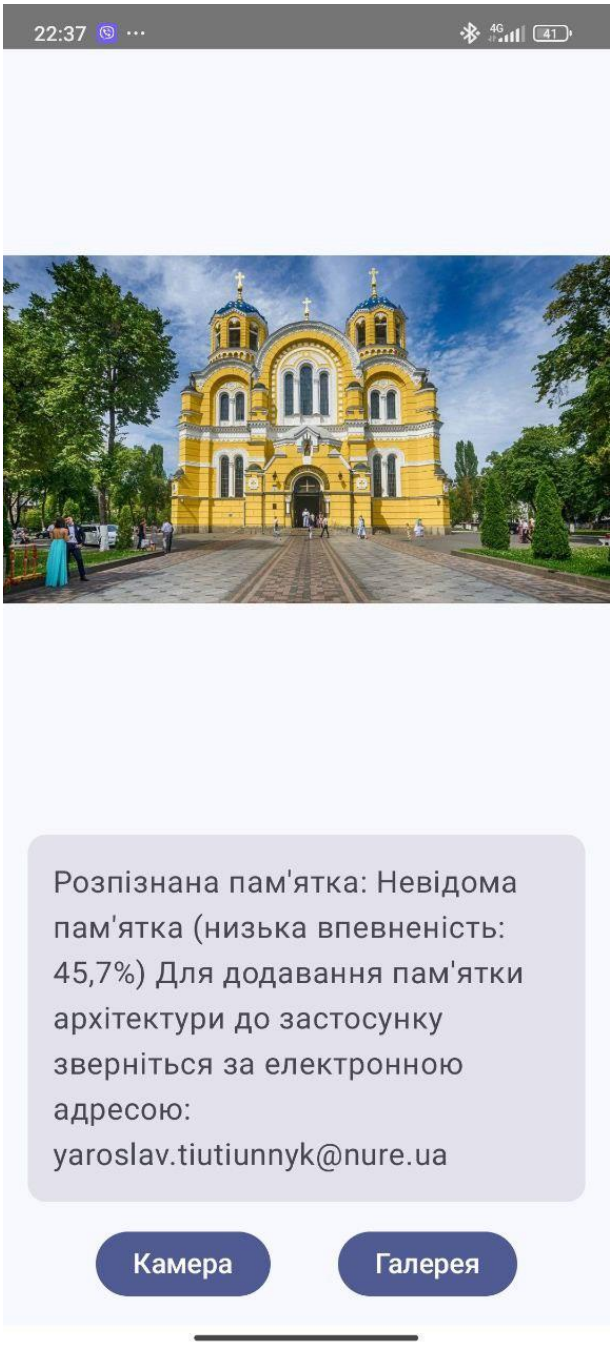


Рисунок 3.10 – Результат розпізнавання пам’ятки архітекури України, розпізнавання якої не передбачене інструментальним засобом

У процесі тестування було перевірено коректність відображення інтерфейсу на різних розмірах екранів, а також стабільність роботи під час багаторазових викликів камери. Додатково було проаналізовано час відгуку після виконання знімка до отримання результату класифікації – у середньому він становив від 1 до 2 секунд, що є прийнятним для мобільного застосунку.

Також було протестовано модель на різних зображеннях з пам’ятками архітектури у різних умовах освітлення. Інструментальний засіб показав хорошу стабільність та достатню точність при класифікації об’єктів, які були представлені у навчальній вибірці. Випадки з невизначеним фоном або частковим перекриттям об’єкта могли впливати на точність результату, проте загальна ефективність системи була задовільною.

Таким чином, результати тестування засвідчили працездатність і функціональну повноцінність мобільного застосунку. Надалі можливе розширення набору класів, оптимізація моделі та інтеграція можливостей обробки зображень з галереї користувача.

3.4 Перспективи подальшої роботи

Розроблений мобільний застосунок для розпізнавання архітектурних пам’яток України на зображеннях є стабільною основою для подальшого розширення функціональності, підвищення точності, оптимізації продуктивності та масштабування на інші напрямки. У рамках кваліфікаційної роботи було реалізовано базову архітектуру системи, інтегровано модель згорткової нейронної мережі, налагоджено обробку зображень із камери та галереї, а також побудовано зручний користувацький інтерфейс. Проте у процесі використання та тестування застосунку були виявлені потенційні напрями для вдосконалення.

Одним із першочергових завдань подальшої роботи є розширення набору класів архітектурних пам’яток, які може розпізнавати модель.

У поточній версії використано 18 класів, однак Україна має значно більше унікальних архітектурних об’єктів, які можна включити в модель після збирання додаткових даних. Для цього доцільно створити масштабовану базу зображень із залученням відкритих джерел.

Також доцільним є удосконалення моделі машинного навчання. Поточна версія має точність близько 73%, що є прийнятним результатом, але може бути поліпшена шляхом використання сучасніших архітектур, таких як EfficientNet, або попереднього навчання (transfer learning) на більших наборах даних. Крім того, можна розглянути додаткову обробку зображень, аугментацію, балансування класів та застосування кращих стратегій регуляризації.

Іншим важливим напрямом є оптимізація продуктивності застосунку на слабших пристроях. Хоча модель виконується локально за допомогою TensorFlow Lite, варто розглянути використання апаратного прискорення (наприклад, GPU Delegate або NNAPI Delegate), щоб зменшити час відповіді та енергоспоживання. Також є можливість використовувати динамічне завантаження моделі з сервера, що дозволить оновлювати її без потреби перевипуску всього застосунку.

З точки зору функціональності перспективним є впровадження геолокації, що дозволить автоматично обмежувати пошук пам’яток залежно від місцезнаходження користувача. Це значно підвищить точність і релевантність розпізнавання, оскільки багато архітектурних об’єктів є регіонально специфічними.

Іншим можливим удосконаленням є створення історії класифікацій або галереї розпізнаних об’єктів, яка зберігатиме дані про попередні результати та дозволить користувачеві переглядати розпізнані пам’ятки в інтерактивному вигляді. Це не лише підвищить зручність використання, а й дозволить накопичувати особисту колекцію побачених об’єктів.

Також перспективним є впровадження багатомовного інтерфейсу, зокрема англійською мовою, що дозволить використовувати застосунок не лише українцям, а й туристам, які подорожують країною.

Крім цього, варто передбачити можливість інтеграції з популярними сервісами, такими як Google Maps чи Wikipedia, для перегляду додаткової інформації про розпізнану пам’ятку.

Зрештою, одним із стратегічних напрямів розвитку є перенесення моделі розпізнавання на серверну інфраструктуру, що дозволить виконувати більш складні обчислення, акумулювати статистику, будувати рекомендаційні системи та інтегрувати аналітику використання.

Таким чином, розроблений мобільний застосунок має значний потенціал для розширення – як у технічному, так і в інформаційному аспектах. Подальший розвиток може перетворити його на повноцінний туристичний навігатор, освітній ресурс або елемент національної цифрової спадщини, що популяризуватиме українську архітектуру на глобальному рівні.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було розроблено мобільний застосунок для розпізнавання архітектурних пам’яток України на зображеннях, який поєднує сучасні методи глибокого навчання, бібліотеки Android Jetpack, та технології побудови інтерфейсу Jetpack Compose. Основна мета проєкту – створення інструменту, що здатен визначати архітектурні об’єкти на фотографіях, – була успішно досягнута.

У процесі розробки проведено ґрунтовний аналіз існуючих рішень у сфері розпізнавання об’єктів і мобільної розробки, що дозволило обґрунтовано обрати відповідні інструментальні засоби. Для реалізації класифікації було підготовлено набір зображень, побудовано й навчено згорткову нейронну мережу, яку згодом конвертовано у формат TensorFlow Lite для ефективного використання на мобільних пристроях.

Мобільний застосунок було реалізовано в середовищі Android Studio з використанням мови програмування Kotlin, компонентів бібліотеки CameraX для роботи з камерою, а також фреймворку TensorFlow Lite для виконання інференсу моделі без підключення до інтернету. Було реалізовано підтримку введення зображень як із камери, так і з галереї мобільного пристрою, автоматичну обробку та класифікацію фото, а також динамічне відображення результату в інтерфейсі користувача.

Модель досягла точності понад 73% на тестовій вибірці, що є прийнятним результатом для задачі багатокласової класифікації. Весь процес виконується на пристрої в реальному часі, що гарантує швидкодію та автономність застосунку.

У результаті роботи було створено повноцінний застосунок, який можна використовувати як у навчальних, культурних, так і в туристичних цілях. Окрім того, у роботі окреслено перспективи подальшого розвитку системи: розширення набору зображень, оптимізація моделі, впровадження геолокації, багатомовної підтримки та інтеграції з онлайн-сервісами.

Загалом виконання даної кваліфікаційної роботи дозволило застосувати на практиці знання з програмування, обробки зображень, машинного навчання та розробки інтерфейсів, а також продемонструвало можливості сучасних мобільних технологій у сфері культурної спадщини.

Результати роботи апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [36].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Moyo.ua. Що таке Google Об’єктив – на прикладі 10 функцій. Moyo.ua. URL: https://www.moyo.ua/ua/news/google_lens_chto_eto_v_2_slovakh_osobennosti_ustanovki_i_10_funktsiy_prilozheniya.html (дата звернення 20.04.2025).

2. Microsoft Corporation. Seeing AI. Google Play. URL: https://play.google.com/store/apps/details?id=com.microsoft.seeingai&hl=uk (дата звернення 20.04.2025).

3. PlantSnap, Inc. (n.d.). PlantSnap – Identify Plants, Trees, Mushrooms With An App. URL: https://www.plantsnap.com (дата звернення 20.04.2025).

4. Ashiq, F., Asif, M., Ahmad, M. B., Zafar, S., Masood, K., Mahmood, T., ... & Lee, I. H. (2022). CNN-based object recognition and tracking system to assist visually impaired people. *IEEE ACCESS*, 10, 14819-14834.

5. Arkin, E., Yadikar, N., Xu, X., Aysa, A., & Ubul, K. (2023). A survey: object detection methods from CNN to transformer. *Multimedia Tools and Applications*, 82(14), 21353-21383.

6. Karar, M. E., Alsunaydi, F., Albusaymi, S., & Alotaibi, S. (2021). A new mobile application of agricultural pests recognition using deep learning in cloud computing system. *Alexandria Engineering Journal*, 60(5), 4423-4432.

7. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

8. Chen, J. W., Lin, W. J., Cheng, H. J., Hung, C. L., Lin, C. Y., & Chen, S. P. (2021). A smartphone-based application for scale pest detection using multiple-object detection methods. *Electronics*, 10(4), 372.

9. Ahmed, A. A., & Reddy, G. H. (2021). A mobile-based system for detecting plant leaf diseases using deep learning. *AgriEngineering*, 3(3), 478-493.

10. Sahin, V. H., Oztel, I., & Yolcu Oztel, G. (2022). Human monkeypox classification from skin lesion images with deep pre-trained network using mobile application. *Journal of medical systems*, 46(11), 79.

11. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

12. Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1701-1708).

13. Rosenberg, C. (2013). Improving photo search: A step across the semantic gap. Google Research Blog, 12.

14. Maltoni, D. (2013). Pattern recognition by means of hierarchical temporal memory. *Technical report*, DEIS - University of Bologna.

15. LeCun, Y., Huang, F. J., & Bottou, L. (2004, June). Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* (Vol. 2, pp. II-104). IEEE.

16. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2). Cambridge: MIT press.

17. Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

18. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

19. Developers Breach. Convolutional neural network | Deep learning. URL: https://developersbreach.com/convolution-neural-network-deep-learning/ (дата звернення 23.04.2025).

20. Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, *29*(9), 2352-2449.

21. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.

22. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

23. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

24. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

25. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

26. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

27. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

28. Gorokhovatskyi V., Chmutov Y., Tvoroshenko I., and Kobylin O. (2025) Reducing computational costs by compressing the structural description in image classification methods, *Advanced Information Systems*, vol. 9, no. 1, pp. 5–12.

29. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

30. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, *International Journal of Academic and Applied Research*, 7(11), pp. 134-145.

31. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine*, pp. 25-27.

32. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

33. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

34. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.

35. Yakovleva O., Matúšová S., Tvoroshenko I., and Isaiev Y. (2024) Visitor counting based on video stream analysis from surveillance cameras to solve various business problems, *Verejná správa a regionálny rozvoj ekonómia, manažment a marketing*, XX(1), pp. 67-87.

36. Тютюнник Я.О. (2025) Аналіз особливостей сучасного мобільного застосунку Seeing AI для розпізнавання об’єктів на зображеннях. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.).* Харків: ХНУРЕ, 2025. Т. 7. С. 148-150.