МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра Інформаційних технологій

Кваліфікаційна робота магістра

на тему: Розробка розширеного функціоналу та адміністративної системи застосунку для сповіщень про небезпечні об'єкти

Виконав студент групи МІС-22
спеціальності 122 Комп'ютерні науки
Тичіна Максим Сергійович

Керівник к.т.н., доцент
Фразе-Фразенко Олексій Олексійович

Рецензент регіональний
координатор програми EGAP
в Одеській області,
Копиченко Іван Юрійович

Одеса 2023

АНОТАЦІЯ

на магістерську кваліфікаційну роботу
«Розробка розширеного функціоналу та адміністративної системи застосунку для сповіщень про небезпечні об’єкти»
студента Тичіни Максима Сергійовича

Актуальність теми. Під час військового конфлікту небезпечні об’єкти можуть залишатися в різних місцях, і навіть випадковий дотик чи неправильне переміщення таких об'єктів може спричинити трагічні наслідки. За допомогою існуючого мобільного застосунку Danger Alerts користувачі можуть повідомляти про знаходження небезпечних об’єктів, але більшість людей не можуть самостійно розпізнавати тип таких об’єктів. Метою даної кваліфікаційної роботи є розробка мобільного застосунку з функцією автоматичного розпізнавання небезпечних об'єктів для полегшення повідомлень про їх знаходження та інформування про правильне поводження з ними.

Для досягнення даної мети були вирішені наступні задачі: виконаний аналіз предметної області використання цифрових помічників, порівняльний аналіз існуючих аналогів, вибір програмних засобів розробки, проектування системи, реалізація розширеного функціоналу застосунку та тестування його роботи.

Об’єкт дослідження: мобільний застосунок для сповіщень про небезпечні об’єкти з їх автоматичним розпізнаванням. Предмет дослідження: методи та засоби розробки функціоналу класифікації зображень з використанням технологій штучного інтелекту. Методи дослідження: аналіз та синтез для визначення переваг та недоліків існуючих аналогів, UML-проектування, методи та технології штучного інтелекту, експериментальне дослідження для перевірки отриманих результатів.

Результатом проведеної роботи є розширений функціонал застосунку «Danger Alerts». Практичне значення роботи полягає в покращенні зручності

процесу повідомлення про знайдені небезпечні об’єкти користувачам застосунку. Новизна полягає у застосуванні мереж глибокого навчання до ситуації мінної небезпеки в Україні, що дає можливість полегшити та убезпечити інформаційний процес.

Кваліфікаційна робота магістра складається з вступу, 5 розділів, висновків, переліку посилань на 17 найменувань. Повний обсяг проекту становить 73 сторінки, містить 30 рисунків.

Ключові слова: військовий конфлікт, небезпечні об’єкти, технології штучного інтелекту, розпізнавання зображень, класифікація об’єктів, TensorFlow.

SUMMARY

for a master's thesis

«Development of advanced functionality and administrative system of the application for notifications about dangerous objects»

by student Tychyna Maksym

Actuality of theme. During a military conflict, dangerous objects can remain in various places, and even an accidental touch or improper movement of such objects can cause tragic consequences. With the existing Danger Alerts mobile app, users can report when they find dangerous objects, but most people cannot recognize the type of objects themselves. The purpose of this qualification work is to develop a mobile application with the function of automatic recognition of dangerous objects to facilitate notifications about their finding and informing about their proper handling.

To achieve this goal, the following tasks were solved: analysis of the subject area of use of digital assistants, comparative analysis of existing analogues, selection of development software, system design, implementation of extended functionality of the application and testing of its operation.

Object of study: a mobile application for notifications about dangerous objects with their automatic recognition. The subject of research: methods and means of developing the functionality of image classification using artificial intelligence technologies. Research methods: analysis and synthesis to determine the advantages and disadvantages of existing analogues, UML design, methods and technologies of artificial intelligence, experimental research to verify the obtained results.

The result of the work is the extended functionality of the "Danger Alerts" application. The practical significance of the work is to improve the convenience of the process of notifying the users of the found dangerous objects. The novelty lies in the application of deep learning networks to the mine danger situation in Ukraine, which makes it possible to facilitate and secure the information process.

The master's thesis consists of an introduction, 5 chapters, conclusions, a list of references for 17 titles. The full scope of the project is 73 pages, 30 drawings.

Keywords: military conflict, dangerous objects, artificial intelligence technologies, image recognition, object classification, TensorFlow.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Adam	–	Adaptive moment estimation, оцінка адаптивного моменту
AI	–	Artificial intelligence, штучний інтелект
API	–	Application Programming Interface, програмний інтерфейс програми
CNN	–	Convolutional neural network, згорткова нейронна мережа
GPU	–	Graphics processing unit, графічний процесор
NLP	–	Natural language processing, обробка природної мови
NoSQL	–	Not Only SQL, нереляційна база даних
PCA	–	Principal component analysis, метод головних компонент
ReLU	–	Rectified Linear Unit, зрізаний лінійний вузол
RNN	–	Recurrent neural network, рекурентна нейронна мережа
SGD	–	Stochastic gradient descent, стохастичний градієнтний спуск
SQL	–	Structured Query Language, мова структурованих запитів
SVM	–	Support Vector Machines, метод опорних векторів
TFX	–	TensorFlow Extended, розширена версія бібліотеки TensorFlow
TPU	–	Tensor processing unit, тензорний блок обробки
UML	–	Unified Modeling Language, уніфікована мова моделювання
ДСНС	–	Державна служба з надзвичайних ситуацій
ОС	–	Операційна система
ШІ	–	Штучний інтелект

ВСТУП

Питання безпеки в зоні військового конфлікту стоїть як для військових, так і для мирного населення. Однією з найбільш серйозних проблем, з якою стикаються непосвячені військові та звичайні громадяни, є незнання того, як виглядають різні типи небезпечних предметів, таких як артилерійські снаряди, гранати, авіабомби та інші військові пристрої. Подібні предмети можуть бути дуже різноманітними за формою, розміром та кольором, і для звичайної людини складно або навіть неможливо визначити їх характеристики та небезпеку.

Цей недолік знань та досвіду може мати катастрофічні наслідки: неправильне поводження з небезпечними предметами може призвести до нещасних випадків, травм та вибухів. У зоні військового конфлікту навіть випадковий дотик чи неправильне переміщення таких об'єктів може спричинити трагічні наслідки.

Метою даної кваліфікаційної роботи є розробка мобільного застосунку з функцією автоматичного розпізнавання небезпечних об'єктів. З огляду на наведені факти ця задача є актуальною та вкрай важливою. Дозволяючи користувачам повідомляти про виявлені об'єкти, програма повинна мати можливість не тільки отримувати текстові та геолокаційні дані, але й використовувати штучний інтелект для аналізу прикріплених фотографій чи відео. Алгоритми комп'ютерного зору можуть розпізнавати різні типи небезпечних об'єктів за їх зовнішнім виглядом, що забезпечить швидке та точне визначення рівня загрози.

Така програма не тільки збільшить безпеку користувачів, а й допоможе військовим та рятувальникам швидше та точніше реагувати на загрози в зоні конфлікту. Шляхом впровадження автоматичного розпізнавання небезпечних об'єктів у розроблений застосунок Danger Alerts, можна значно покращити інформування та попередження у ситуаціях, пов'язаних із виявленням

небезпечних предметів, та зрештою сприяти зниженню людських жертв та травм у зонах воєнних конфліктів.

Для досягнення мети кваліфікаційної роботи були сформульовані такі завдання:

- аналіз предметної області;
- порівняльний аналіз існуючих програм-аналогів;
- обґрунтування вибору програмних засобів розробки та технологій;
- проектування системи;
- реалізація застосунку;
- тестування.

Структура кваліфікаційної роботи магістра складається з вступу, 5 розділів, висновків та переліку посилань на 17 найменувань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Під час військового конфлікту однією з серйозних та актуальних проблем є визначення та знешкодження небезпечних об'єктів, розміщених у різних місцях. Ці небезпечні об'єкти можуть включати не тільки озброєння, боєприпаси та вибухові речовини, але й міни, мінні поля, залишки озброєння, бомби, а також об'єкти інфраструктури, які були пошкоджені під час бойових дій.

Проблема полягає в тому, що ці небезпечні об'єкти можуть залишатися прихованими та невидимими для необізнаних осіб. Вони становлять серйозну загрозу для мирного населення, військових та рятувальників, і можуть призвести до загибелі людей, травм, руйнування інфраструктури та створення суспільного страху.

Подібні об'єкти розташовуються у різних місцях: у містах та сільських населених пунктах, на дорогах та мостах, у будівлях та спорудах, а також у прибережних зонах. Вони можуть залишатися в землі або під водою та становити небезпеку для цивільного населення, військових підрозділів та гуманітарних організацій, які надають допомогу постраждалим.

Розпізнавання та знешкодження цих небезпечних об'єктів є складними та небезпечними процесами, що потребують спеціалізованого обладнання, знань та досвіду. Недолік точної інформації про місцезнаходження та характеристики цих об'єктів створює додаткові труднощі для фахівців, які працюють на передовій лінії.

Розробка ефективних методів та технологій для виявлення та знешкодження небезпечних об'єктів у різних місцях під час військового конфлікту має критичне значення для забезпечення безпеки та порятунку життів.

Проблема, коли звичайні громадяни можуть не знати, як виглядають різні типи небезпечних предметів, такі як артилерійські снаряди, гранати, авіабомби та інші озброєння, має серйозні наслідки за часів воєнних конфліктів та конфліктних зон. Це веде до ризику для життя та здоров'я, оскільки люди можуть випадково підійти до небезпечних предметів, розглядати їх чи навіть переміщати, не розуміючи потенційної загрози.

У більшості випадків звичайні громадяни не мають навичок та знань для визначення небезпечних об'єктів, особливо в умовах стресу та паніки, які можуть виникнути за часів бойових дій. Вони можуть не впізнавати небезпечні предмети і навіть плутати їх із нешкідливими предметами. Це може призвести до нещасних випадків, включаючи вибухи, травми та загибель.

Тому додавання в розроблений мобільний застосунок, що дозволяє користувачам повідомляти про виявлені об'єкти, автоматичне розпізнавання небезпечних предметів є критично важливим заходом для забезпечення безпеки в умовах конфлікту. Автоматичне розпізнавання дозволить застосунку швидко та точно визначати небезпечні предмети на основі фотографій або відеозаписів, наданих користувачами.

Це дозволить надавати користувачам негайну інформацію про те, наскільки небезпечний об'єкт, який вони виявили, та рекомендації щодо того, як діяти в цій ситуації. Така програма стає важливим інструментом для навчання та інформування громадян про потенційні небезпеки в конфліктних зонах і може сприяти зниженню ризиків та порятунку життів.

Дана робота спрямована на розширення функціоналу наявного застосунку Danger Alerts для сповіщення про небезпечні об'єкти [1]. Основною метою роботи є впровадження методів штучного інтелекту для розпізнавання небезпечних об'єктів, а також створення адміністративної частини, що дозволяє ефективно управляти даними та забезпечувати швидшу реакцію на погрози.

1.2 Аналіз існуючих програмних систем

На даний час різні організації створюють застосунки і сервіси, що надають можливість повідомляти про знаходження небезпечних об'єктів та отримувати інформацію про такі об’єкти поруч з собою.

Конфлікт в Україні став стимулом для інновацій у сфері безпеки та допомоги громадянам. Багато українських компаній та організацій, включаючи державні структури, доклали всіх зусиль для створення застосунків, спрямованих на безпеку громадян. Серед цих розробок можна виділити застосунки, які попереджають про можливі небезпеки у найближчому оточенні. Ці інноваційні інструменти надають інформацію про потенційні загрози та допомагають людям приймати обґрунтовані рішення для збереження власної безпеки.

Завдяки таким застосункам люди отримують актуальні дані про ситуацію в реальному часі, що дозволяє їм уникати небезпечних зон і вживати запобіжних заходів. Українські компанії продовжують впроваджувати нові ідеї, використовуючи технології для забезпечення безпеки та підтримки своїх громадян за умов нестабільності. Ці програми стають невід'ємною частиною допомоги та захисту громадян у періоди непередбачуваних подій.

Застосунок MineFree від Free Ukraine - це безкоштовний мобільний застосунок, який допомагає користувачам безпечно переміщатися по території України, де все ще залишається багато боєприпасів, що не розірвалися [2]. Програма використовує дані з відкритих джерел, щоб ідентифікувати потенційні мінні поля та інші небезпечні зони. Користувачі можуть повідомляти про підозрілі об'єкти у застосунку, і ці повідомлення перевірятимуться фахівцями (рис. 1.1).

Деякі основні функції програми MineFree:

– ідентифікація потенційних мінних полів та інших небезпечних зон за допомогою даних із відкритих джерел;

– можливість повідомляти про підозрілі об'єкти;

– перевірка повідомлень про підозрілі об'єкти фахівцями;

– доступ до інформації про безпеку та першу допомогу.

Для того, щоб повідомити про загрозу, необхідно зареєструватись у застосунку - вказати прізвище, ім’я, по-батькові та номер телефону. Після цього необхідно вказати категорію та тип знахідки, обравши зі списку, а потім опис та адресу об’єкта. Після цього можна вказати точну геолокацію знайденого об’єкта (рис. 1.2).

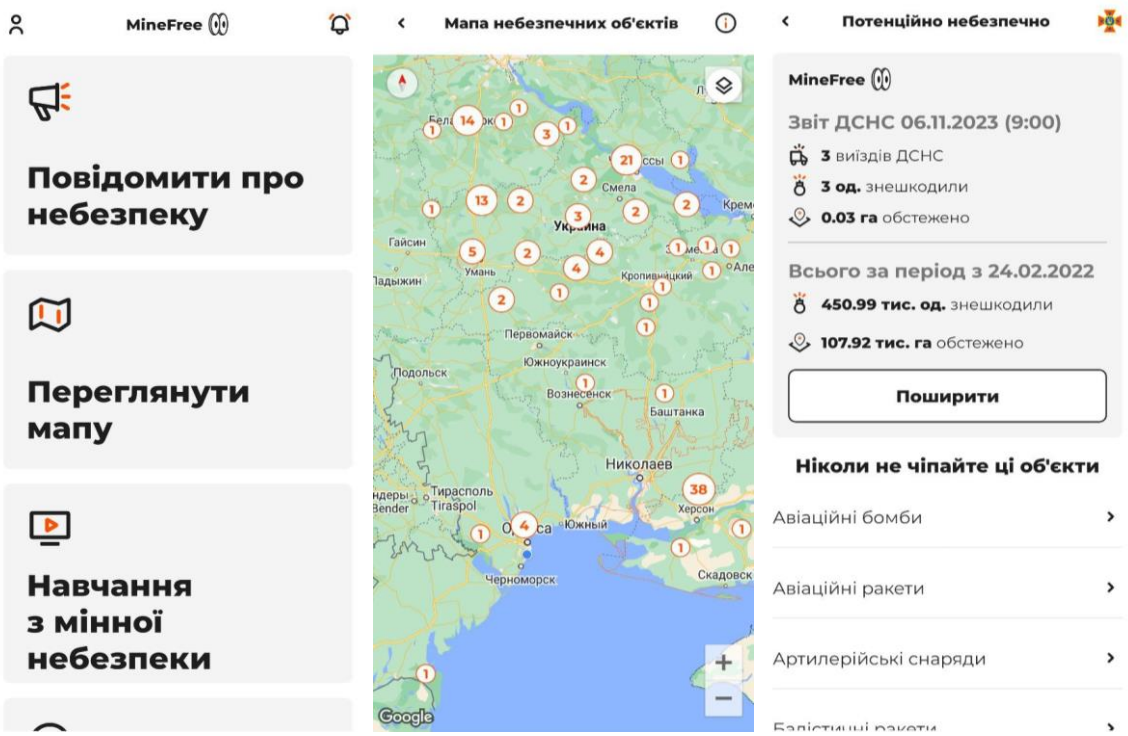

Рисунок 1.1 – Основні екрани застосунку MineFree від Free Ukraine

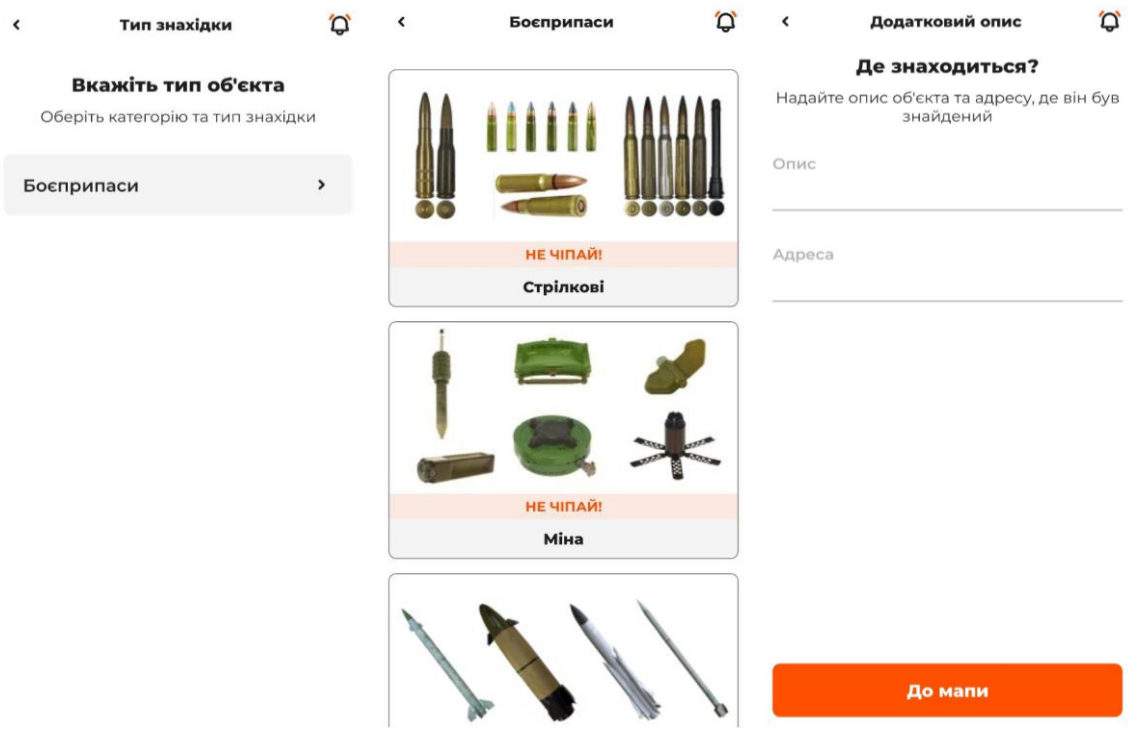

Рисунок 1.2 – Основні етапи повідомлення про небезпечну знахідку в застосунку MineFree

Розмінування України від ДСНС - це офіційний застосунок ДСНС для інформування про виявлені підозрілі предмети та небезпечних зон [3].

Додаток використовує дані від Державної служби надзвичайних ситуацій України. Користувачі можуть повідомляти про підозрілі об'єкти у додатку, і ці повідомлення перевірятимуться фахівцями ДСНС.

Для того, щоб повідомити про небезпечний об’єкт, необхідно авторизуватися через id.gov.ua або Дія.Підпис, а потім додати контактні дані (номер телефону та електронну пошту). Після цього необхідно обрати точну локацію, додати текстовий опис знахідки та додати фото або відео об’єкту (рис. 1.3).

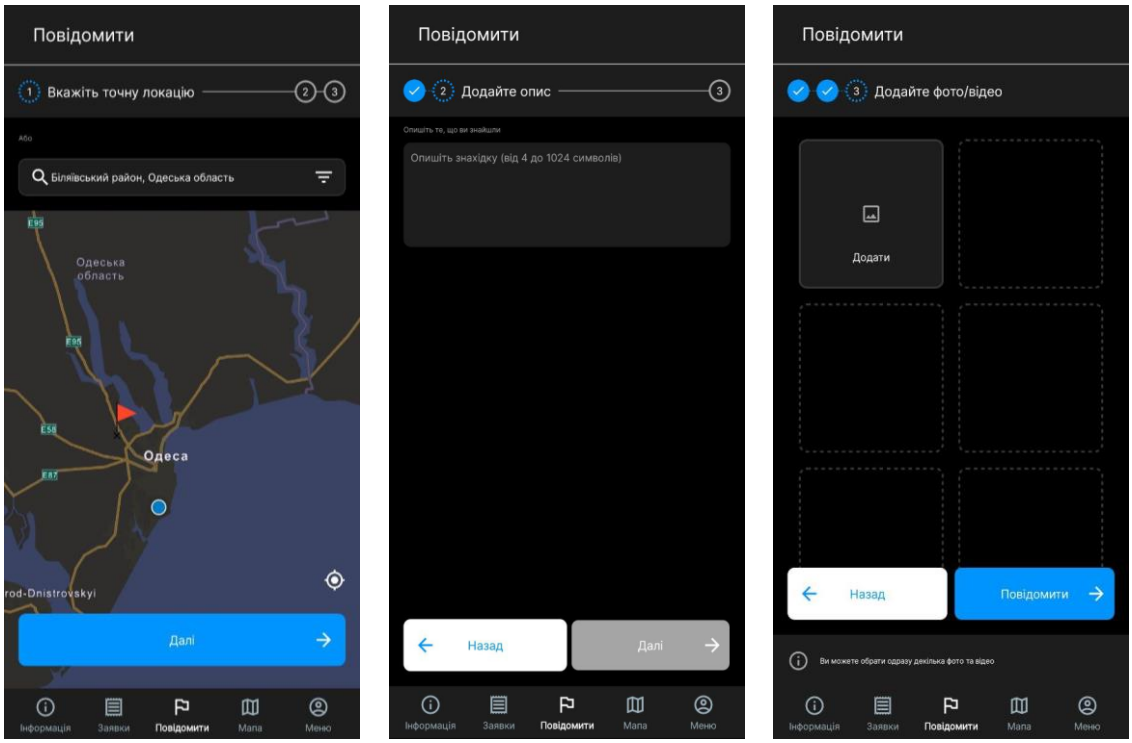

Рисунок 1.3 – Основні кроки для повідомлення про небезпечний об’єкт в застосунку Розмінування України від ДСНС

В результаті аналізу існуючих застосунків для повідомлень про небезпечні знахідки було виявлено, що в них відсутня функція автоматичного розпізнавання підозрілих об’єктів, що робить дану розробку вкрай актуальною.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Функціональні та нефункціональні вимоги

Перед початком розробки застосунку були визначені функціональні та нефункціональні вимоги до нього.

Функціональні вимоги визначають функції, які має виконувати програмне забезпечення або застосунок. Це конкретні можливості, операції або послуги, які система повинна забезпечувати для користувачів. Наприклад, це може бути можливість реєстрації користувача, обмін повідомленнями, функція пошуку тощо.

Нефункціональні вимоги описують якості чи характеристики системи, які не є конкретними функціями, але впливають на її ефективність, безпеку, зручність використання тощо. Це можуть бути рівень безпеки, продуктивність, швидкість реакції, надійність системи, доступність для користувачів з обмеженими можливостями, масштабованість та інші параметри, які визначають якість роботи системи.

До функціональних вимог належать:

- відображення небезпечних об’єктів на карті;
- сповіщення про небезпечні об’єкти на пристрої користувача;
- відображення інформації про тип небезпечного об’єкту;
- можливість встановити радіус, в межах якого надходять повідомлення про небезпечні об’єкти;
- відображення детальної інформації про кожний об'єкт;
- можливість додати інформацію про знайдений небезпечний об’єкт;
- автоматичне розпізнавання типу об’єкта при наведенні камери.

До нефункціональних вимог належать:

- простий і зрозумілий інтерфейс;
- швидкість та ефективність у використанні;
- необхідність підключення до Інтернету.

2.2 Обґрунтування вибору операційної системи

При виборі операційної системи для розробки застосунку необхідно враховувати ряд факторів. Основні операційні системи: Android, iOS та Windows. В даній роботі була обрана система Android. Одним з основних факторів, який вплинув на вибір операційної системи, є те, що вже розроблений застосунок, який оптимізується та розширюється в даній роботі, розрахований на платформу Android.

До того, операційна система Android має численні переваги, які роблять її однією з найпопулярніших і широко використовуваних платформ для мобільних пристроїв у світі. Ось кілька ключових переваг Android:

Відкритий вихідний код: Android є операційною системою з відкритим вихідним кодом, що означає, що розробники можуть вільно користуватися і змінювати її за необхідності. Це сприяє інноваціям та розвитку нових функцій і додатків.

Широкий асортимент пристроїв: Android працює на різних мобільних пристроях від різних виробників, від економ-класу до високоякісних смартфонів і планшетів. Це надає користувачам великий вибір і можливість підібрати пристрій за своїми потребами та можливостями.

Гнучкість і кастомізація: Android дозволяє користувачам кастомізувати свої пристрої, змінювати вигляд і відчуття інтерфейсу, встановлювати сторонні додатки та робити різні системні модифікації. Це надає більше свободи в користуванні пристроєм.

Широка підтримка додатків: Google Play Store пропонує великий вибір застосунків та ігор, які можна легко встановити на Android-пристрої. Розробники мають можливість легко публікувати свої додатки на цій платформі.

Інтеграція з Google: Android має глибоку інтеграцію з послугами Google, такими як Gmail, Google Drive, Google Maps, інші сервіси хмарного сховища та застосунки офісних програм.

Низькі вартості та доступність: Через широкий спектр пристроїв, від доступних до більш високих за ціною, Android-пристрої є більш доступними для широкого кола користувачів.

Постійний розвиток: Google постійно вдосконалює Android, випускаючи оновлення, які включають нові функції та покращення безпеки, що робить платформу сучасною і конкурентоспроможною.

2.3 Вибір програмного забезпечення глибокого навчання

Програмне забезпечення глибокого навчання (deep learning software) - це набір інструментів, бібліотек, фреймворків і програм, розроблених для створення, навчання та використання глибоких нейронних мереж. Глибоке навчання - це розділ машинного навчання, який орієнтований на створення та навчання глибоких нейронних мереж, моделей, які мають безліч шарів для отримання ієрархічних ознак даних. На даний момент існує безліч програмного забезпечення для глибокого навчання. Основними та найпоширенішими є PyTorch, Caffe 2, TensorFlow.

PyTorch – це фреймворк для глибокого навчання з відкритим кодом, розроблений Facebook (рис. 2.1).

Рисунок 2.1 – Логотип фреймворку PyTorch

Він надає гнучкі інструменти для створення та навчання нейронних мереж [4]. Ось кілька ключових особливостей PyTorch, що є його перевагами:

Динамічний граф обчислень: Однією з ключових переваг PyTorch є динамічний граф обчислень, який робить код більш гнучким та інтуїтивно зрозумілим для розробників. Це спрощує процес налагодження та експериментів.

Інтуїтивний інтерфейс: PyTorch надає простий та інтуїтивно зрозумілий інтерфейс, особливо близький до стандартного синтаксису Python. Це полегшує навчання новачків та прискорює процес розробки моделей.

Підтримка передачі навчання (transfer learning): Фреймворк забезпечує зручні засоби для використання попередньо вивчених моделей, що спрощує розробку нових завдань з обмеженими ресурсами.

Великі інструменти для досліджень: PyTorch надає багатий набір інструментів для візуалізації, налагодження та профілювання, що робить його зручним інструментом для досліджень та експериментів.

Активна спільнота та велика документація: Існує активна спільнота PyTorch, і фреймворк має велику документацію, навчальні матеріали та ресурси для обміну досвідом.

Інтеграція з іншими бібліотеками: PyTorch легко інтегрується з іншими бібліотеками машинного навчання та науковими бібліотеками в екосистемі Python.

Серед недоліків PyTorch можна виділити:

Продуктивність: Граф обчислень будується динамічно, що може утруднити його оптимізацію і розподілене навчання у деяких випадках, особливо при роботі з великими моделями та даними.

Великий обсяг коду: Написання динамічного графа обчислень потребує більшого обсягу коду, порівняно зі статичним графом, що може бути незручним у деяких випадках.

Обмежені інструменти для розгортання: PyTorch має порівняно небагато інструментів для розгортання моделей у продукцію, що може бути важливим фактором у промислових сценаріях.

Caffe2 - це бібліотека глибокого навчання з відкритим вихідним кодом, призначена для розробки та навчання глибоких нейронних мереж. Вона є наступником оригінального фреймворку Caffe, розробленого в Berkeley AI Research (рис. 2.2).

Рисунок 2.2 – Логотип бібліотеки Caffe2

Перевагами Caffe2 є [5]:

Продуктивність: Caffe2 спроектований з акцентом на ефективність та продуктивність. Це особливо важливо при роботі з великими моделями та об'ємними даними.

Модульність: Фреймворк побудований з урахуванням модульності, що робить його гнучким для інтеграції в різні програми та сценарії. Ця модульність забезпечує легкість у використанні та масштабованість.

Безліч оптимізацій: Caffe2 включає безліч оптимізацій для поліпшення продуктивності навчання та передбачень, включаючи підтримку багатопоточності та можливості використання апаратного прискорення, такого як GPU.

Підтримка переносимості: Caffe2 створено з урахуванням переносимості, що означає можливість використання моделей різних пристроях і платформах.

Інтеграція з іншими фреймворками: Caffe2 може використовуватися спільно з іншими фреймворками, такими як PyTorch. Інтеграція з PyTorch дозволяє комбінувати переваги обох фреймворків.

Підтримка розгортання: Caffe2 надає інструменти та інтерфейси для розгортання моделей у продакшені, що є важливим для промислових застосувань.

Активна спільнота: Caffe2 підтримується активною спільнотою, що сприяє обміну досвідом та розвитку фреймворку.

Недоліки Caffe2 включають:

Більш обмежена база знань та ресурсів у порівнянні з іншими фреймворками.

Складність початку використання: Для деяких користувачів може знадобитися час на освоєння Caffe2, особливо якщо вони вже знайомі з іншими фреймворками.

Менша кількість навчальних матеріалів: У порівнянні з більш популярними фреймворками, є менше навчальних матеріалів та посібників для Caffe2.

TensorFlow – це відкрита бібліотека глибокого навчання, розроблена Google. Вона надає інструменти для побудови та навчання різних типів нейронних мереж, включаючи згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та інші [6].

Рисунок 2.3 – Логотип бібліотеки TensorFlow

Ось кілька ключових характеристик TensorFlow:

Граф обчислень: TensorFlow використовує статичний граф обчислень, що означає, що структура обчислень визначається заздалегідь. Це забезпечує оптимізацію та ефективність під час виконання.

Широкий спектр застосувань: TensorFlow застосовується в різних галузях, включаючи машинне навчання, штучний інтелект, комп'ютерний зір, природну мову, посилення навчання та багато іншого.

Гнучкість та масштабованість: TensorFlow надає гнучкі засоби для створення та навчання моделей на різних рівнях складності – від простих моделей до глибоких нейронних мереж. Він також забезпечує підтримку розподіленого навчання, що дозволяє масштабувати процес навчання на кілька пристроїв.

Широке співтовариство: TensorFlow має велику і активну спільноту, що покращує доступність навчальних матеріалів, посібників та підтримки.

Інструменти візуалізації: TensorFlow надає інструменти для візуалізації графів обчислень, процесу навчання та результатів передбачень. TensorBoard - один з таких інструментів, що надає безліч можливостей для аналізу та налагодження моделей.

Підтримка передачі навчання (transfer learning): TensorFlow забезпечує зручні засоби для використання попередньо навчених моделей, що дозволяє вирішувати нові завдання з обмеженими ресурсами.

Екосистема TensorFlow: Разом з основною бібліотекою TensorFlow має різні розширення та проекти, такі як TensorFlow Lite для мобільних пристроїв, TensorFlow.js для виконання моделей у браузері, і TensorFlow Extended (TFX) для побудови повних пайплайнів машинного навчання.

Інтеграція з Keras: Keras, високорівневий API для побудови нейронних мереж, став вбудованою частиною TensorFlow. Це робить TensorFlow більш доступним та інтуїтивно зрозумілим для новачків.

Недоліки TensorFlow:

Складність для початківців: Завдяки своїй потужності та гнучкості TensorFlow може бути складним для новачків. Однак інтеграція Keras та покращення в документації зробили бібліотеку доступнішою.

Великі обсяги коду: Побудова моделей TensorFlow іноді вимагає написання більшого обсягу коду в порівнянні з деякими іншими фреймворками.

Великий обсяг навчальних даних: Деякі моделі TensorFlow, особливо глибокі нейронні мережі, можуть вимагати великих обсягів навчальних даних для досягнення хороших результатів.

Менша гнучкість динамічного графа: У порівнянні з фреймворками, що використовують динамічний граф, таким як PyTorch, TensorFlow може бути менш гнучким у певних сценаріях, особливо там, де потрібне динамічне визначення графа.

В даній кваліфікаційній роботі буде використовуватись TensorFlow, що обґрунтоване його потужністю, гнучкістю та підтримкою широкого спектру моделей глибокого навчання. TensorFlow Lite забезпечує розгортання на мобільних пристроях, інтеграція з Keras спрощує створення моделей, а підтримка GPU та TPU забезпечує ефективне навчання.

3 ЗАДАЧА КЛАСИФІКАЦІЇ ДАНИХ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Основні методи штучного інтелекту

Штучний інтелект (ШІ) – це область комп'ютерних наук, яка займається створенням програм та систем, здатних виконувати завдання, які потребують інтелектуальних здібностей людини. В останні десятиліття дослідження в галузі штучного інтелекту призвели до розвитку різних методів. Штучний інтелект базується на різних методах і техніках, які допомагають системам та комп'ютерам вирішувати завдання, які традиційно пов'язані з людським інтелектом. Основні методи ШІ включають машинне навчання, нейронні мережі, логічне програмування, обробка природної мови, системи експертів, генетичні алгоритми та еволюційне програмування [7].

Машинне навчання - це підрозділ штучного інтелекту, в якому комп'ютерні системи навчаються виконувати завдання на основі досвіду та даних. Існує кілька основних методів машинного навчання:

Навчання з учителем (Supervised Learning): Цей метод включає навчання моделі на основі розмічених даних, де кожен вхідний приклад зіставляється з відповідним виходом. Приклади включають алгоритми регресії та класифікації.

Під час навчання з учителем моделі навчаються за допомогою позначеного набору даних, де модель дізнається про кожен тип даних. Після завершення процесу навчання модель перевіряється на основі тестових даних (підмножина навчального набору), а потім прогнозується результат.

Схематично принцип навчання з учителем наведено на рис. 3.1.

Якщо є набір даних різних типів фігур, який включає квадрат, прямокутник, трикутник і багатокутник, перший крок полягає в навчанні моделі для кожної форми.

Наприклад, якщо дана фігура має чотири сторони, і всі сторони рівні, то вона буде позначена як квадрат. Якщо дана форма має три сторони, то вона

буде позначена як трикутник. Якщо задана фігура має шість рівних сторін, вона буде позначена як шестикутник. Тепер, після навчання, необхідно протестувати модель за допомогою тестового набору, і завдання моделі - визначити форму.

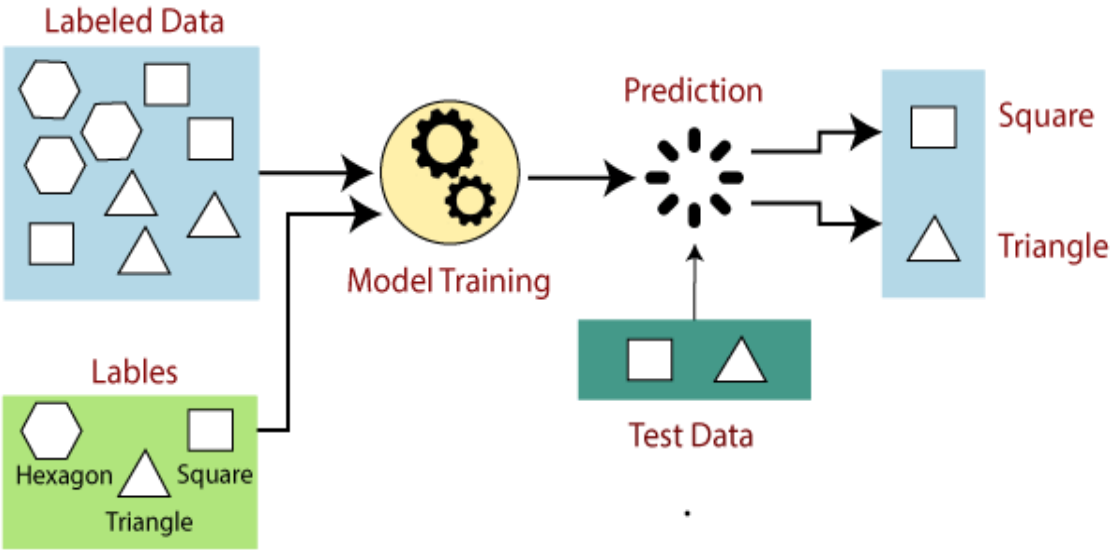

Рисунок 3.1 - Принцип машинного навчання з учителем

Машина вже навчена на всіх типах фігур, і коли вона знаходить нову форму, вона класифікує форму на основі кількох сторін і прогнозує результат.

Навчання без вчителя (Unsupervised Learning): Модель навчається на нерозмічених даних, намагаючись виявити закономірності та шаблони даних. Кластеризація та методи зниження розмірності, такі як метод головних компонентів (PCA), є прикладами методів навчання без вчителя.

Навчання без вчителя — це техніка машинного навчання, у якій моделі не контролюються за допомогою навчального набору даних. Натомість моделі самі знаходять приховані закономірності та ідеї з наданих даних. Це можна порівняти з навчанням, яке відбувається в людському мозку під час вивчення нового. Навчання без вчителя – це тип машинного навчання, у якому моделі

навчаються за допомогою немаркованого набору даних і їм дозволяється діяти з цими даними без будь-якого контролю.

Навчання без вчителя не можна безпосередньо застосувати до проблеми регресії чи класифікації, оскільки, на відміну від контрольованого навчання, є вхідні дані, але немає відповідних вихідних даних. Мета навчання без вчителя полягає в тому, щоб знайти базову структуру набору даних, згрупувати ці дані за подібністю та представити цей набір даних у стиснутому форматі [8].

Приклад: алгоритму навчання без вчителя надано вхідний набір даних, що містить зображення різних типів котів і собак. Алгоритм ніколи не навчається на заданому наборі даних, що означає, що він не має жодного уявлення про особливості набору даних. Завдання алгоритму полягає в тому, щоб самостійно визначити особливості зображення. Алгоритм навчання без вчителя виконає це завдання шляхом кластеризації набору даних зображень у групи відповідно до подібності між зображеннями. Схематично цей алгоритм наведено на рис. 3.2.

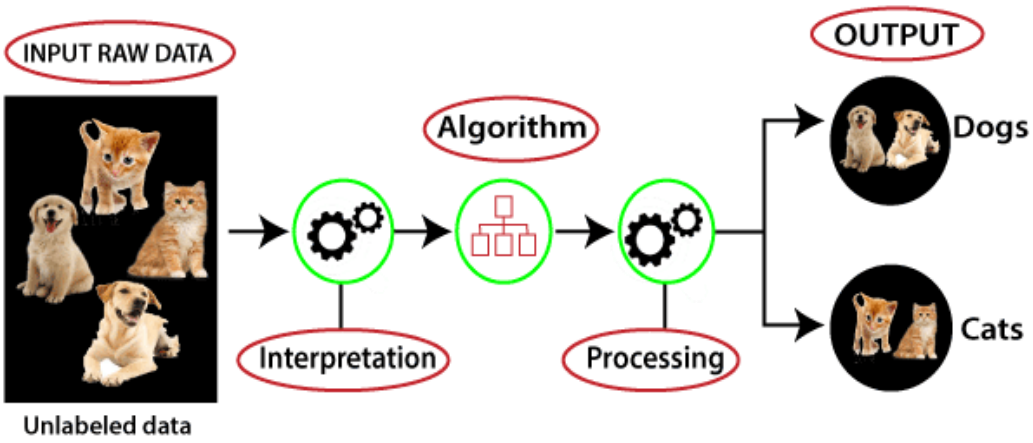

Рисунок 3.2 - Принцип машинного навчання без учителя

Навчання з підкріпленням (Reinforcement Learning): У цьому методі агент навчається приймати рішення, максимізуючи деяку нагороду в заданому

середовищі. Агент приймає рішення на основі своїх дій та нагород, що спостерігаються.

Навчання з підкріпленням — це техніка машинного навчання на основі зворотного зв’язку, за якої агент вчиться поводитися в середовищі, виконуючи дії та бачачи результати дій. За кожну хорошу дію агент отримує позитивний відгук, а за кожну погану дію агент отримує негативний відгук або штраф.

У Reinforcement Learning агент навчається автоматично, використовуючи зворотний зв’язок без будь-яких позначених даних, на відміну від навчання під наглядом.

Агент взаємодіє з середовищем і досліджує його сам. Основна мета агента в навчанні з підкріпленням полягає в тому, щоб покращити продуктивність шляхом отримання максимальної позитивної винагороди.

Агент на основі досвіду вчиться виконувати завдання кращим чином. Це основна частина штучного інтелекту, і всі агенти ШІ працюють на концепції навчання з підкріпленням. Не потрібно попередньо програмувати агента, оскільки він навчається на власному досвіді без втручання людини [9].

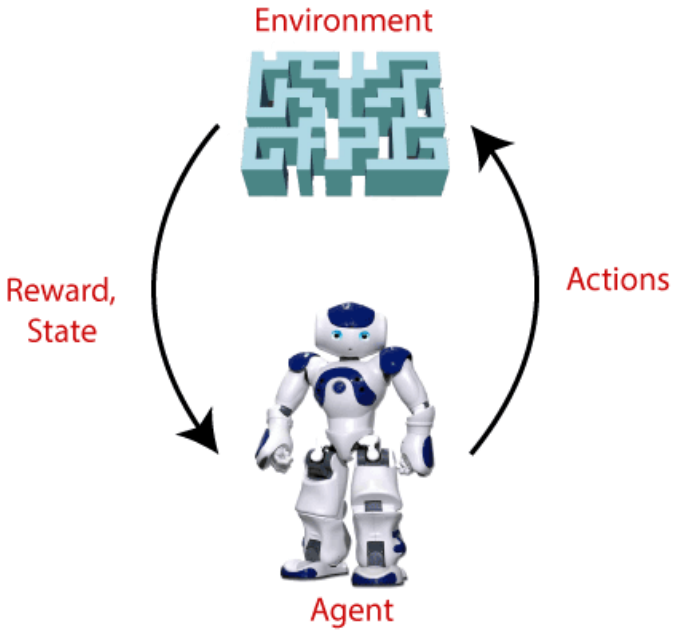

Рисунок 3.3 - Принцип машинного навчання з підкріпленням

Приклад: в середовищі лабіринту є агент ШІ, і його мета — знайти алмаз. Агент взаємодіє з середовищем, виконуючи певні дії, і на основі цих дій змінюється стан агента, а також він отримує винагороду або штраф як зворотний зв’язок. Агент продовжує виконувати ці три речі (виконувати дії, змінювати стан/залишатися в тому самому стані та отримувати зворотний зв’язок), і, виконуючи ці дії, він вивчає та досліджує середовище. Агент дізнається, які дії призводять до позитивного відгуку або винагороди, а які – до покарання за негативний відгук. В якості позитивної винагороди агент отримує позитивний бал, а в якості штрафу – негативний бал. Схематично це зображено на рис. 3.3.

Логічне програмування – це підхід до розробки програм, що базується на логіці та правилах. Головною ідеєю логічного програмування є використання логічних розсудків для вирішення завдань [10]. Основні елементи логічного програмування включають:

Пропозиційні системи використовують математичну логіку та правила для виведення нової інформації з вже відомої. Вони складаються з бази даних фактів та множини правил, які використовуються для виведення нових фактів. Прикладами таких систем є Prolog, Datalog, CLIPS.

Мови представлення знань дозволяють описати знання та правила у вигляді, зрозумілому для комп'ютера. Основними елементами є факти, правила та запити. Факти – це ствердження про об'єкти або стани речей. Правила – це умови, за якими можна зробити певний висновок. Запити – це запити до системи про певну інформацію.

Логічне програмування використовує дедуктивний підхід до вирішення задач. Перш за все, воно встановлює базу фактів, а потім використовує правила логіки для здійснення висновків. Системи, створені за принципами логічного програмування, здатні вирішувати завдання у галузях, де потрібно логічне мислення, таких як штучний інтелект, експертні системи, обробка природної мови тощо.

Однією з основних переваг логічного програмування є його декларативний характер. Ви описуєте, що потрібно зробити, а не як це робити. Програми у логічному програмуванні можуть бути більш зрозумілими та елегантними, оскільки вони працюють на основі правил та логіки.

Однак, у складних задачах, логічне програмування може мати обмеження в ефективності та швидкості роботи, оскільки деякі завдання можуть вимагати більше обчислювальних ресурсів та складних умов для формулювання правил.

Обробка природної мови (Natural Language Processing, NLP) - це галузь штучного інтелекту, що займається розумінням та обробкою мови, що використовується людьми. Вона охоплює широкий спектр технік, що дозволяють комп'ютерам розуміти, аналізувати та генерувати людську мову. деякі ключові аспекти обробки природної мови:

1. Токенізація та сегментація: токенізація - це процес розбиття тексту на окремі слова або токени. Це важливий етап у роботі з текстом, оскільки він дозволяє розділити текст на менші одиниці для подальшої обробки.

2. Синтаксичний та семантичний аналіз: синтаксичний аналіз визначає структуру речень, встановлює зв'язки між словами. Семантичний аналіз спрямований на розуміння значень слів та їх контексту у реченні для визначення смислового змісту.

3. Машинне навчання та глибоке навчання в NLP: машинне навчання та глибоке навчання використовуються для навчання моделей розуміти текст, виконувати переклади, визначати настрій тексту, відповідати на питання та інше.

Системи експертів (СЕ) є типом штучного інтелекту, призначеним для моделювання та використання знань конкретних експертів у певній галузі для розв'язання складних проблем. Вони базуються на правилах та даних, зібраних від фахівців у галузі, і використовують цю інформацію для прийняття рішень. Основні аспекти систем експертів включають бази знань, інференційні

механізми, експертні системи для діагностики, системи експертів для планування та системи підтримки прийняття рішень.

Центральним елементом СЕ є бази знань, що містять інформацію, правила та факти про певну предметну область. Ця інформація збирається від експертів та перетворюється на логічні структури даних для подальшого використання системою. Системи експертів використовують інференційні механізми для виведення нової інформації на основі правил та фактів, що містяться в базі знань. Вони дозволяють системі генерувати висновки та рекомендації на основі введених даних.

Експертні системи для діагностики використовується для визначення причин проблеми або стану системи, заснований на вхідних даних. Вони можуть використовуватися в медицині, техніці, програмуванні та інших областях. Системи планування розроблені для генерування стратегій та рекомендацій з планування в різних областях, включаючи виробництво, логістику та управління проектами. Системи підтримки прийняття рішень надають користувачам рекомендації та варіанти вибору для прийняття оптимальних рішень в різних ситуаціях.

Системи експертів глибоко інтегровані в різні галузі, забезпечуючи ефективне використання знань та досвіду експертів. Вони допомагають у вирішенні проблем, які вимагають експертного аналізу та великої кількості даних, і грають ключову роль в прийнятті рішень у складних сценаріях.

Генетичні алгоритми (ГА) та еволюційне програмування (ЕП) - це методи оптимізації та пошуку рішень, що моделюють процес еволюції у природі для вирішення складних проблем. Вони використовуються в різних областях, де потрібно знайти оптимальні рішення та велику кількість варіантів. Основними принципами генетичних алгоритмів та еволюційного програмування є популяція, генетичні операції, функція пристосованості та відбір. Генетичні алгоритми використовують популяцію індивідів (часто представлених у вигляді хромосом), кожен з яких представляє потенційне рішення. Ці індивіди піддаються генетичним операціям, таким як

схрещування (комбінування хромосом батьків для створення нових індивідів) та мутація (випадкове змінення генетичного матеріалу). У ГА кожен індивід оцінюється за допомогою функції пристосованості, яка визначає, наскільки добре він вирішує проблему. Індивіди з більшою пристосованістю мають більші шанси бути відібраними для наступного покоління. Вибір індивідів для наступного покоління здійснюється на підставі їхньої пристосованості. Індивіди з вищою пристосованістю мають більші шанси бути відібраними.

Генетичні алгоритми застосовуються для знаходження оптимальних параметрів у складних системах, таких як налаштування параметрів моделей або оптимізація процесів. Вони також можуть бути використані для створення оптимальних конструкцій та дизайну, включаючи структури та архітектури різних машин та пристроїв [11].

3.2 Задача класифікації в штучному інтелекті

Класифікація в штучному інтелекті (ШІ) - це завдання, яке полягає в призначенні вхідного об'єкта одній з певної кількості категорій або класів на підставі його характеристик. Це може бути визначення типу об'єкта на зображенні (наприклад, чи це собака чи кіт), визначення категорії тексту (наприклад, тема новини) або класифікація даних у сфері медицини (наприклад, визначення хвороби за симптомами пацієнта).

Для реалізації задачі класифікації в ШІ зазвичай використовуються алгоритми машинного навчання, зокрема, нейронні мережі, дерева прийняття рішень, метод опорних векторів та багато інших [12].

Нейронні мережі широко використовуються для реалізації задач класифікації в штучному інтелекті. Особливо ефективними виявляються глибокі нейронні мережі, такі як згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та їхні варіації, що здатні працювати з різними типами даних, такими як зображення, текст, аудіо або послідовності даних.

Згорткові нейронні мережі (CNN) чудово підходять для обробки зображень. Вони вміють виявляти різні фільтри або ознаки на зображеннях, такі як краї, текстури, форми тощо. Кожен шар мережі впізнає все складніші або абстрактніші ознаки об'єктів на зображенні. Після тренування модель може класифікувати зображення за цими ознаками.

Рекурентні нейронні мережі (RNN) використовуються для обробки послідовних даних, таких як текст або часові послідовності. Вони мають здатність "пам'ятати" попередні стани та використовувати цю інформацію для аналізу нових даних. Наприклад, для класифікації тексту RNN може аналізувати послідовні слова та контекст, щоб визначити тему або емоційний тон тексту.

Глибокі нейронні мережі - це мережі з великою кількістю шарів, які дозволяють моделі вчитися більш складним шаблонам та характеристикам даних. Вони можуть автоматично визначати корисні ознаки для класифікації, що робить їх ефективними для завдань зі складними структурами даних [13].

Дерева прийняття рішень - це графічна модель прийняття рішень, яка використовується для реалізації задач класифікації в штучному інтелекті. Вони представляють собою структуру у вигляді дерева, де кожен вузол представляє певну властивість чи атрибут даних, розділяючи набір даних на підгрупи.

Основна ідея дерев прийняття рішень полягає у послідовному поділі набору даних на менші підгрупи шляхом поставлення питань про їхні атрибути. На кожному вузлі дерева приймається рішення про те, який атрибут найкраще розділить дані. Цей процес триває до тих пір, поки всі дані в кожній гілці дерева не належать до одного класу або досягне певної зупинки (наприклад, максимальної глибини дерева).

Дерева прийняття рішень мають кілька переваг, серед яких можна визначити легкість інтерпретації, бо рішення, прийняті деревом, легко інтерпретувати та візуалізувати, що робить їх зрозумілими для фахівців і нефахівців. Дерева можуть ефективно працювати з різними типами даних,

включаючи категоріальні та числові дані, а також вони здатні автоматично обробляти відсутні дані. До того, дерева прийняття рішень не вимагають передпопередньої нормалізації даних, що є однією з їхніх переваг.

Проте дерева прийняття рішень можуть мати деякі обмеження, наприклад тенденцію до перенавчання - великі та складні дерева можуть перенавчитися на навчальних даних і погіршити свою універсальність на нових даних, та недостатня точність у порівнянні з деякими складнішими моделями.

Для використання дерев прийняття рішень у завданнях класифікації важливо враховувати оптимальний вибір параметрів моделі, таких як глибина дерева, критерії розділення тощо, для досягнення оптимальної продуктивності моделі.

Метод опорних векторів (SVM) - це потужний алгоритм машинного навчання, який використовується для задач класифікації та регресії. В основі його роботи лежить пошук оптимального гіперплощинного розділення між класами даних, що найкращим чином відокремлює їх [14].

Основні принципи використання SVM для класифікації включають:

Визначення оптимальної гіперплощини: SVM шукає оптимальну гіперплощину, яка найкраще розділяє дані на класи. У двовимірному просторі це може бути пряма лінія, а у вищих вимірах - гіперплощина.

Максимізація ширини розділення: SVM прагне максимізувати ширину "коридору" (маржі) між класами, що допомагає забезпечити кращу універсальність моделі на нових даних та уникнути перенавчання.

Ядерні функції: SVM використовує ядерні функції для відображення даних у вищорозмірний простір, де вони можуть бути краще розділені лінійними гіперплощинами.

Основні переваги використання SVM для класифікації:

Ефективність у високорозмірних просторах: SVM працює добре навіть у високорозмірних просторах даних, що робить його ефективним для задач з багатьма ознаками.

Ефективність у випадку обмеженого набору даних: Завдяки використанню маржинальних класифікаторів, SVM може працювати добре навіть при обмеженому обсязі тренувальних даних.

Гнучкість вибору ядра: Використання різних ядерних функцій дозволяє адаптувати SVM до різних типів даних та задач.

Недоліки методу опорних векторів - чутливість до шуму та перекривання класів у деяких випадках та складність вибору оптимальних параметрів, таких як тип ядра, параметри ядра, регуляризація.

Процес класифікації включає кілька етапів:

1. Підготовка даних. Цей етап включає збір, очищення та підготовку даних для подальшого використання. Це може включати в себе нормалізацію даних, обрізання, видалення шуму, вирівнювання даних тощо.
2. Вибір моделі. Вибір алгоритму або моделі, яка найкраще підходить для конкретної задачі. Наприклад, для класифікації тексту може бути використана рекурентна нейронна мережа, а для класифікації зображень - згорткова нейронна мережа.
3. Тренування моделі. Цей етап включає в себе подачу вхідних даних в модель та оптимізацію параметрів моделі, щоб вона могла вчитися розпізнавати певні патерни та залежності у даних.
4. Оцінка та налаштування. Після тренування моделі проводиться оцінка її продуктивності за допомогою тестових даних. Якщо результат не задовільний, можливе налаштування параметрів моделі або вибір іншої моделі.
5. Застосування. Коли модель готова, її можна використовувати для класифікації нових даних.

Задача класифікації є ключовою у штучному інтелекті, оскільки вона застосовується в різних галузях та завданнях, допомагаючи робити важливі висновки на основі вхідних даних.

3.3 Нейронні мережі класифікації зображень в TensorFlow

У процесі навчання моделі для класифікації зображень в TensorFlow зазвичай використовуються згорткові нейронні мережі (Convolutional Neural Networks - CNN).

Загальний алгоритм процесу навчання моделі TensorFlow:

Підготовка даних: зображення розбиваються на тренувальний, валідаційний та тестовий набори, перетворюються до необхідного формату та розміру.

Створення моделі CNN: створюється архітектура CNN, що складається зі шарів згортки, пулінгу, повністю з'єднаних шарів та функцій активації. Зазвичай, кожен згортковий шар виявляє різні функції або ознаки на зображенні, а шари пулінгу допомагають зменшити розмірність.

Компіляція моделі: обираються функція втрат (наприклад, категоріальна крос-ентропія), оптимізатор (наприклад, Adam, SGD) та метрики для оцінки моделі під час тренування (наприклад, точність).

Тренування моделі: модель навчається на тренувальних даних за допомогою методу зворотнього розповсюдження помилки (backpropagation). Для кожного зображення обчислюються передбачення, порівнюються з правильними мітками та оновлюються ваги мережі так, щоб зменшити втрати.

Оцінка моделі: після тренування модель оцінюється на валідаційному наборі для оцінки її продуктивності. Проводиться аналіз метрик (точність, втрати тощо) для визначення ефективності моделі та можливого підбору гіперпараметрів.

Тестування моделі: коли модель виявиться задовільною на валідаційному наборі, вона тестується на тестовому наборі для оцінки її загальної продуктивності.

У TensorFlow для реалізації цих кроків використовуються вбудовані функції та модулі, такі як tf.keras для побудови та навчання моделі, функції компіляції compile, методи тренування fit та методи оцінки evaluate.

tf.keras є високорівневим API для швидкої та зручної розробки нейронних мереж на базі TensorFlow. tf.keras надає легкий спосіб побудови нейронних мереж за допомогою високорівневих абстракцій, що дозволяє швидко створювати моделі. Цей інтерфейс може використовуватися для створення різних типів нейронних мереж, включаючи звичайні шарові мережі, згорткові нейронні мережі, рекурентні нейронні мережі та їх комбінації. tf.keras є частиною TensorFlow, тому вона безперервно оновлюється та підтримується в межах цієї бібліотеки, що дозволяє користуватися всіма перевагами TensorFlow. Вона підтримує різні оптимізатори для навчання моделей та великий набір функцій втрат для різних видів завдань машинного навчання. Можливість створення моделей шляхом додавання шарів та об'єднання їх у функційні структури дозволяє легко експериментувати з архітектурами нейронних мереж.

Загалом, tf.keras надає простий та потужний інструментарій для створення, навчання та застосування нейронних мереж з використанням TensorFlow, допомагаючи спростити розробку складних моделей глибокого навчання.

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Проектування застосунку перед реалізацією є критичним етапом, оскільки воно надає основу для ефективної розробки програмного забезпечення. Створення UML-діаграм допомагає визначити структуру та функціонал програми, що полегшує розуміння, спілкування та управління проектом.

UML-діаграми дозволяють аналізувати потреби користувачів та визначати функціональні вимоги програми перед її реалізацією. Вони дозволяють визначити структуру програми, включаючи класи, компоненти, модулі та їх взаємодію. Це сприяє організації коду та підготовці до реалізації. UML дозволяє вивчити систему перед її створенням. Це допомагає уникнути перенавантаження та зменшити складність проектування шляхом виокремлення окремих компонентів. Моделювання за допомогою UML може виявити потенційні проблеми архітектури або дизайну ще до початку програмування, що дозволяє їх виправити на ранній стадії. У результаті проектування з UML створюється детальна картинка проекту, що слугує базою для подальшої реалізації, полегшуючи процес програмування та тестування [15].

4.1 Проектування діаграми прецедентів

Діаграма прецедентів (Use Case Diagram) - це тип діаграми в Unified Modeling Language (UML), який дозволяє моделювати функціональні вимоги системи з погляду того, як користувачі взаємодіятимуть із системою. Вона фокусується на функціях системи, ілюструючи різні способи, якими різні типи користувачів можуть взаємодіяти із системою та її компонентами.

Основні елементи діаграми прецедентів в UML:

1. Актори (Actors) представляють ролі, які взаємодіють із системою. Це можуть бути реальні користувачі, інші системи або зовнішні компоненти, які взаємодіють із системою.
2. Прецеденти (Use Cases) описують конкретні функціональні можливості системи. Кожен прецедент представляє певну дію чи серію дій, які система виконує користувачів чи інших систем.
3. Відношення між акторами та прецедентами показують, які прецеденти доступні для кожного актора.

Іноді прецеденти можуть бути пов'язані, наприклад, через включення або розширення, щоб показати зв'язки між ними.

Переваги використання діаграми прецедентів:

Ясне розуміння функціональних вимог. Діаграма дозволяє легко зрозуміти, як система використовуватиметься різними користувачами чи акторами.

Поліпшене планування та проектування. Допомагає виявити основні функції системи та їх взаємозв'язки.

Комунікація із заінтересованими сторонами. Надає простий спосіб обговорення функціональних можливостей системи із заінтересованими сторонами.

Діаграми прецедентів часто використовуються на початкових етапах проектування системи для покращення розуміння її функцій та вимог.

В даному випадку акторами є користувач та адміністратор. Варіантами використання є, з боку користувача: перегляд мапи небезпечних об’єктів, перегляд інформації про знайдені об’єкти, додавання інформації про знайдений небезпечний об’єкт, налаштування сповіщень, розпізнавання типу об’єкта; адміністратора - перевірка інформації про додані об’єкти та видалення з бази після знешкодження.

Розроблена діаграма варіантів використання зображена на рис. 4.1.

Рисунок 4.1 – Діаграма прецедентів

4.2 Створення діаграми активностей

Діаграма активності (або діаграма діяльності) - це тип діаграми Unified Modeling Language (UML), який використовується для візуалізації потоків роботи або діяльності в системі або процесі. Вона відображає послідовність кроків, дій, процесів або операцій, які виконуються у системі, а також умови переходу між ними.

Основні елементи діаграми активності в UML - це вузли та переходи. Вузли (Nodes) поділяються на:

– Дії (Actions): Визначені операції або кроки, які виконуються в рамках діаграми. Наприклад, надсилання повідомлення, обчислення, очікування введення тощо.

– Рішення (Decisions): Вузли, де відбувається вибір між альтернативними шляхами, залежно та умовами.
– Початок (Start) та Кінець (End): Вказують початок та кінець потоку виконання.

Переходи (Transitions) - зв'язки між вузлами, що позначають потік виконання від дії до іншого. Вони можуть бути умовними, які залежать від результатів дій.

Переваги використання діаграм активності:

Візуалізація процесів та діяльності. Дозволяють зрозуміти послідовність та логіку виконання операцій у системі чи процесі.

Поліпшене розуміння послідовності дій. Сприяють кращому розумінню та аналізу діяльності системи.

Планування та оптимізація процесів. Допомагають виявити вузькі місця у процесах та оптимізувати їх.

Діаграми активності часто застосовуються при аналізі бізнес-процесів, проектуванні програмного забезпечення, моделюванні систем, щоб візуалізувати та описати потоки дій та сценарії виконання.

Для застосунку була створена діаграма діяльності, яка відображає основний процес проектованої системи – додавання користувачем знайденого об’єкту (рис. 4.2).

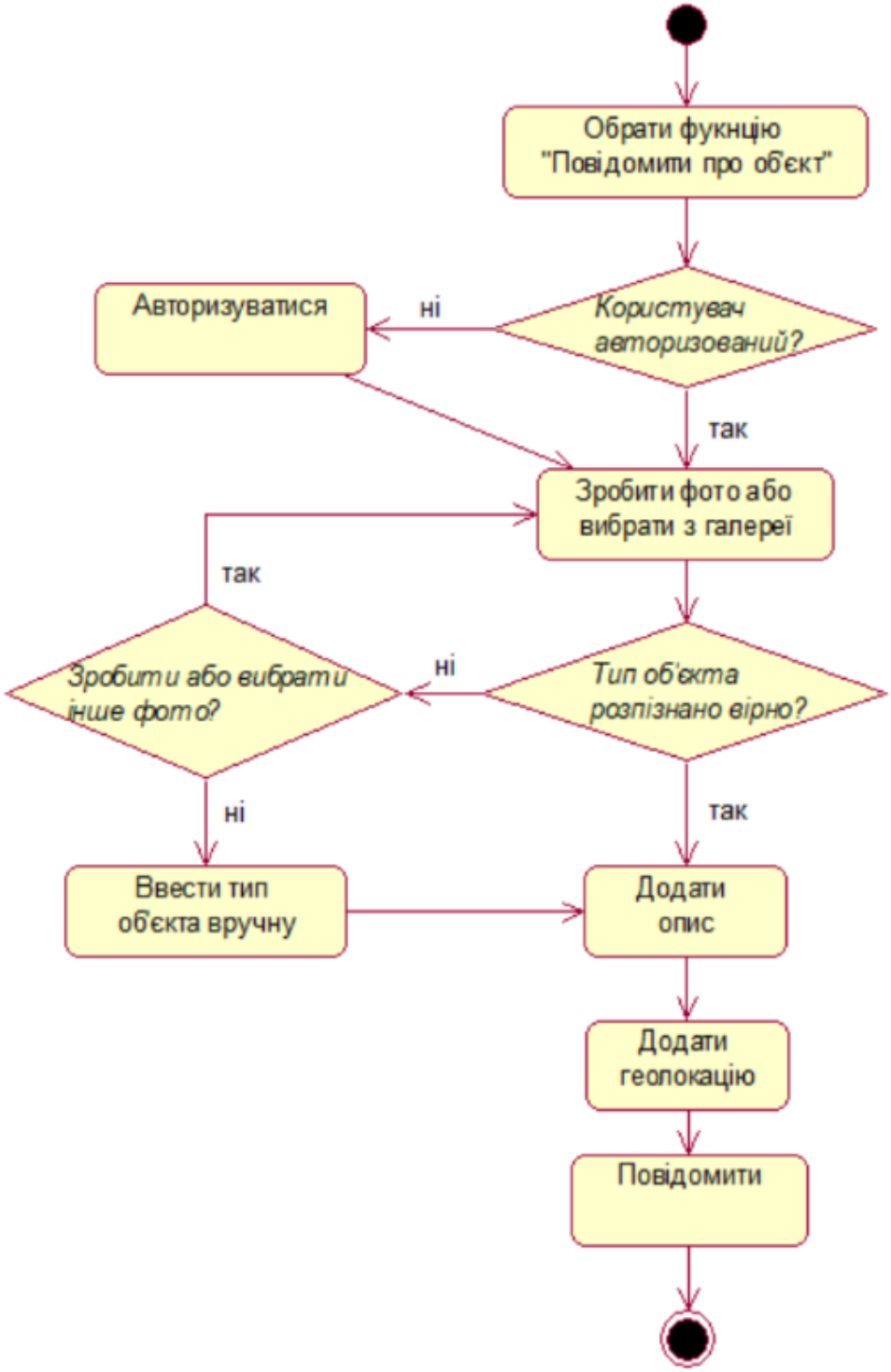

Рисунок 4.2 – Діаграма діяльності

Користувач, який хоче додати повідомлення про знайдений небезпечний об’єкт, має авторизуватися у застосунку. Це необхідно для захисту від спаму. Після авторизації користувач може зробити фото або вибрати фотографію з галереї. Після завантаження зображення у застосунок буде автоматично

розпізнано тип об’єкта. Якщо користувач впевнений, що тип розпізнано неправильно, він може змінити його вручну або завантажити інше фото та спробувати ще раз. Після цього потрібно додати текстовий опис та вказати геолокацію.

Таким чином, на етапі проектування системи були створені UML-діаграми прецедентів та активності, які допомогли зрозуміти основні варіанти використання системи та логіку роботи застосунку.

5 РЕАЛІЗАЦІЯ ЗАСТОСУНКУ «DANGER ALERTS»

5.1 Створення моделі TensorFlow для класифікації зображень

Для роботи з TensorFlow використовується середовище Google Colab. Google Colab (або Colaboratory) – це безкоштовне хмарне середовище для виконання коду мовою програмування Python. Воно надає доступ до обчислювальної потужності графічних процесорів (GPU) та тензорних процесорів (TPU) Google, що робить його особливо корисним для навчання моделей глибокого навчання [16].

Перевагами Google Colab є:

Безкоштовність: Використання Google Colab абсолютно безкоштовно. Це включає доступ до GPU і TPU, що забезпечує високу обчислювальну потужність для навчання моделей.

Хмарне середовище: Colab базується на хмарних обчислювальних ресурсах Google, що дозволяє користувачам виконувати код у браузері без необхідності встановлення та налаштування середовища на локальному комп'ютері.

Інтеграція з Google Drive: Можна легко зберігати та завантажувати дані та моделі, взаємодіючи з Google Drive. Це спрощує збереження результатів та обмін даними між Colab та іншими сервісами Google.

Підтримка Jupyter Notebooks: Colab підтримує Jupyter Notebooks, що робить його зручним для аналізу даних, візуалізації та навчання моделей в інтерактивному середовищі.

Доступ до бібліотек та фреймворків: Colab попередньо встановлений і надає доступ до безлічі бібліотек і фреймворків Python, включаючи TensorFlow та багато інших.

Загальний доступ та спільна робота: Можна легко ділитися блокнотами Colab з іншими користувачами і навіть працювати над проектами в режимі реального часу, що робить його корисним для командної роботи.

Інтеграція з BigQuery: Colab може бути використаний для роботи з BigQuery, що забезпечує зручний доступ до великих наборів даних та можливості виконання SQL-запитів.

Підтримка багатьох мов: Крім Python, Colab підтримує кілька інших мов програмування, таких як R і Scala.

Google Colab є популярним вибором для навчання та експериментів у галузі машинного навчання та глибокого навчання завдяки своїй доступності, обчислювальній потужності та зручності у використанні.

Для побудови та навчання моделі TensorFlow для класифікації зображень необхідно правильно підготувати дані. Потрібен наступний набір даних: тренувальні дані, дані для оцінки та дані для тестування. Ці дані повинні бути набором зображень у форматі, що підтримується TensorFlow (наприклад, масиви NumPy, зображення у форматі JPEG або PNG). Зображення у кожній групі даних повинні бути згруповані по класам, до яких належать зображення.

Тренувальні дані (Training Data) - це набір зображень, які представляють різні класи, які потрібно класифікувати, для кожного зображення вказано клас або категорію, до якої воно належить.

Дані для оцінки (Validation Data) - це додатковий набір зображень для оцінки продуктивності моделі під час навчання. Ці дані допомагають відстежувати узагальнюючу здатність моделі та запобігають перенавчанню. Аналогічно тренувальних даних тут вказуються класи для зображень.

Дані для тестування (Test Data) - це окремий набір зображень, який не використовувався під час навчання або валідації. Використовується для остаточної оцінки продуктивності моделі після завершення навчання.

Пропорції між тренувальними, валідаційними та тестовими даними можуть змінюватись в залежності від доступного обсягу даних та характеру завдання. Однак загальноприйнята практика включає використання близько 60-80% даних для тренування, 10-15% для валідації та 15-20% для тестування. Ці пропорції забезпечують достатню кількість даних для навчання, оцінки та

тестування, що допомагає отримати стійку та узагальнюючу модель. Важливо також, щоб дані в кожному наборі були репрезентативними та добре відображали різноманітність, яку модель повинна розпізнавати у реальних умовах використання.

Для моделі класифікації небезпечних предметів було підготовлено 475 зображень, поділених на 5 основних класів: артилерійські снаряди, авіабомби, ракети, ручні гранати та міни. Вони були згруповані наступним чином: тренувальні дані - 325 зображень (68%), дані для оцінки - 55 зображень (12%), дані для тестування - 95 зображень (20%).

Після створення блокноту в Google Collab, архів з цими даними був завантажений та розархівований у проекті (рис. 5.1).

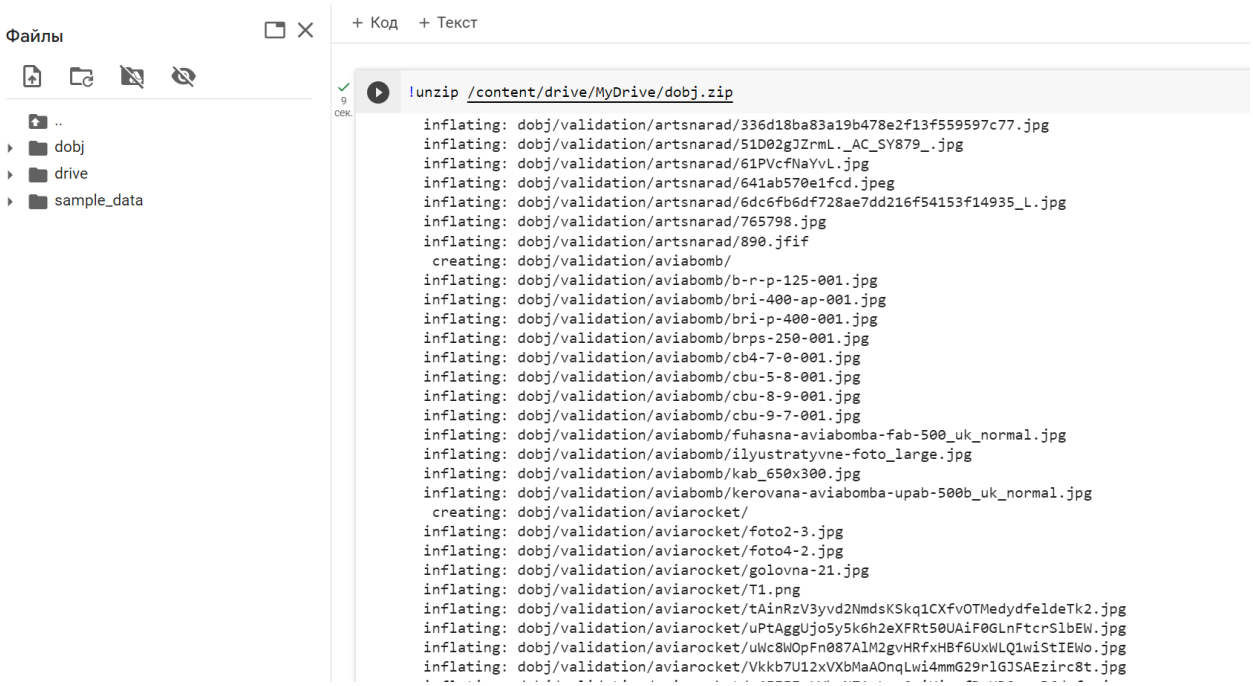

Рисунок 5.1 - Підготовка даних для моделі

За допомогою функції image_dataset_from_directory в бібліотеці TensorFlow можна створити датасет із зображень, організованих у підкаталогах основного каталогу (рис. 5.2). Це зручний спосіб підготувати дані для навчання моделі класифікації зображень.

```
img_h, img_w = 64, 64
batch_size = 20

train_ds = tf.keras.utils.image_dataset_from_directory(
    "dobj/train",
    image_size = (img_h, img_w),
    batch_size = batch_size
)

val_ds = tf.keras.utils.image_dataset_from_directory(
    "dobj/validation",
    image_size = (img_h, img_w),
    batch_size = batch_size
)

test_ds = tf.keras.utils.image_dataset_from_directory(
    "dobj/test",
    image_size = (img_h, img_w),
    batch_size = batch_size
)
```

```
Found 325 files belonging to 5 classes.
Found 55 files belonging to 5 classes.
Found 95 files belonging to 5 classes.
```

Рисунок 5.2 - Створення датасетів з даними для тренування, валідації та тестування

TensorFlow має розвинену документацію з різними прикладами та підходами до створення моделей. Шаблон дій для глибокого навчання в TensorFlow наведено на рис. 5.3.

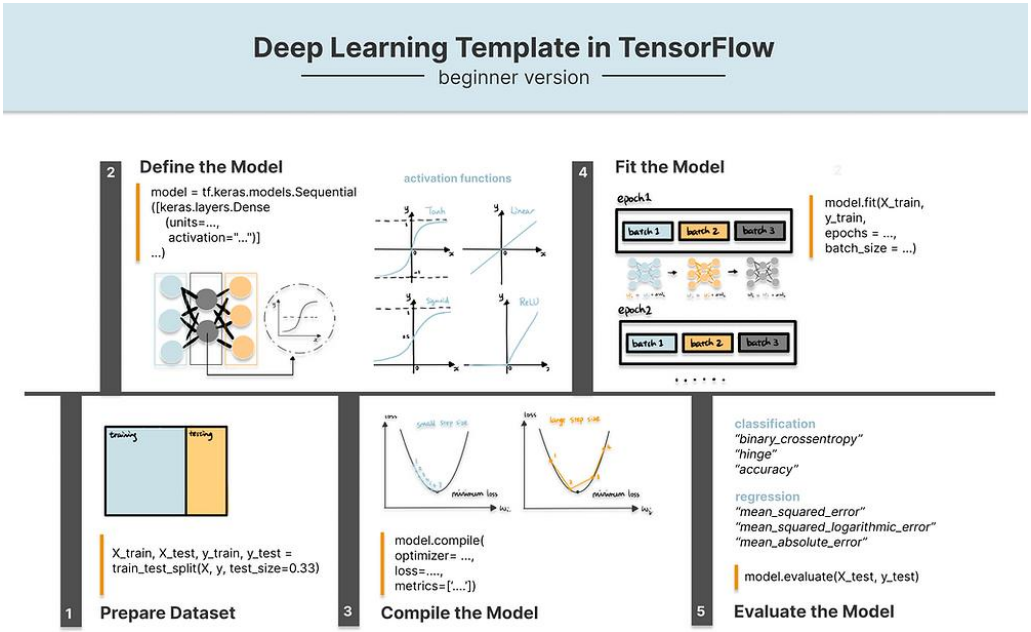

Рисунок 5.3 – Шаблон дій для глибокого навчання в TensorFlow

Розповсюдженим підходом у бібліотеці TensorFlow є створення моделі для класифікації за допомогою Sequential (рис. 5.4). Sequential є простим стековим контейнером для створення моделей з одним входом і одним виходом.

```
model = tf.keras.Sequential(
    [
     tf.keras.layers.Rescaling(1./255),
     tf.keras.layers.Conv2D(64,3,activation="relu"),
     tf.keras.layers.Rescaling(1./255),
     tf.keras.layers.Conv2D(64,3,activation="relu"),
     tf.keras.layers.Rescaling(1./255),
     tf.keras.layers.Conv2D(64,3,activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Flatten(),
     tf.keras.layers.Dense(128,activation="relu"),
     tf.keras.layers.Dense(5)
    ]
)
```

Рисунок 5.4 - Створення моделі TensorFlow

Шар Rescaling виконує масштабування вхідних значень пікселів з діапазону [0, 255] до діапазону [0, 1]. Це корисно для покращення стабільності тренування. Перший шар згорткових шарів Conv2D Використовує 64 фільтри розміром 3x3, використовуючи функцію активації ReLU. ReLU (Rectified Linear Unit) - це функція активації, яка широко використовується в нейронних мережах для введення нелінійності. Вона визначається як max(0, x), де x - вхідний сигнал. Цей шар відповідає за виявлення важливих ознак у зображенні. Далі повторюється патерн для другого та третього згорткових шарів.

Шар MaxPooling2D використовується для підвищення роздільної здатності, використовуючи максимальне значення з певної області. У даному випадку, після трьох згорткових шарів, використовується пулінг для зменшення розміру зображення.

Шар Flatten() вирівнює результуючий об'єкт у вектор, готовий для введення в повністю з'єднаний (Dense) шар.

Шар Dense(128, activation="relu") - повністю з'єднаний шар з 128 нейронами та функцією активації ReLU.

Останній шар, Dense(5), - вихідний повністю з'єднаний шар з 5 нейронами. Оскільки немає функції активації, цей шар видає лінійний вихід, і його вихід інтерпретується як оцінки для класів (у цьому випадку, існує 5 класів).

Наступний крок - компіляція моделі (рис. 5.5). Використовується оптимізатор "adam". Adam (Adaptive Moment Estimation) є ефективним алгоритмом оптимізації градієнта, який адаптивно налаштовує швидкість навчання для кожного параметра моделі.

```
[21] model.compile(
         optimizer="adam",
         loss = tf.losses.SparseCategoricalCrossentropy(from_logits=True),
         metrics=['accuracy']
     )
```

Рисунок 5.5 - Компіляція моделі

Функція втрати SparseCategoricalCrossentropy є часто використовуваною функцією для задач класифікації з кількома класами. Параметр from_logits=True вказує, що модель генерує "логіти", тобто невідмасштабовані перед передачею через функцію активації softmax, і функція втрати самостійно виконує перетворення логітів в ймовірності.

Для визначення якості моделі розраховується метрика точності (accuracy). Метрика "accuracy" служить для вимірювання точності класифікації моделі на основі прогнозованих істинних класів.

Наступний крок - навчання нейронної мережі (рис. 5.6). Викликається метод fit() моделі TensorFlow для навчання нейронної мережі з використанням

даних, представлених у вигляді train_ds і val_ds (навчальний і валідаційний датасети).

Саме метод fit() навчає модель на навчальному датасеті (train_ds) і оцінює її продуктивність на валідаційному датасеті (val_ds). Параметр epochs = 10 вказує на кількість епох навчання, тобто скільки разів модель пройде через всі навчальні дані. Кожна епоха є повним циклом навчання, де модель коригує свої ваги на основі помилки передбачення на навчальних даних.

У процесі навчання метод fit() використовує передані датасети для оновлення ваги моделі відповідно до обраного оптимізатора та функції втрат, які були визначені при компіляції моделі.

Валідаційні дані використовуються для оцінки продуктивності моделі після кожної епохи навчання. Це допомагає відстежувати, як модель узагальнюється на даних, які вона бачила у процесі навчання, і може допомогти у виявленні перенавчання (надто хорошого запам'ятовування навчальних даних).

Такий виклик fit() є базовим способом навчання моделі TensorFlow, де дані подаються у вигляді датасетів, що спрощує роботу з великими обсягами даних і дозволяє ефективно використовувати ресурси навчання.

```
model.fit(
    train_ds,
    validation_data = val_ds,
    epochs = 10
)
```

```
Epoch 1/10
17/17 [==============================] - 29s 1s/step - loss: 0.2540 - accuracy: 0.9200 - val_loss: 1.2062 - val_accuracy: 0.6545
Epoch 2/10
17/17 [==============================] - 22s 1s/step - loss: 0.2478 - accuracy: 0.9231 - val_loss: 1.2790 - val_accuracy: 0.6545
Epoch 3/10
17/17 [==============================] - 23s 1s/step - loss: 0.2237 - accuracy: 0.9323 - val_loss: 1.3701 - val_accuracy: 0.6182
Epoch 4/10
17/17 [==============================] - 21s 1s/step - loss: 0.1800 - accuracy: 0.9446 - val_loss: 1.4413 - val_accuracy: 0.7091
Epoch 5/10
17/17 [==============================] - 24s 1s/step - loss: 0.1743 - accuracy: 0.9354 - val_loss: 1.6007 - val_accuracy: 0.6364
Epoch 6/10
17/17 [==============================] - 22s 1s/step - loss: 0.1843 - accuracy: 0.9385 - val_loss: 1.4997 - val_accuracy: 0.6727
Epoch 7/10
17/17 [==============================] - 21s 1s/step - loss: 0.1491 - accuracy: 0.9538 - val_loss: 1.6317 - val_accuracy: 0.6909
Epoch 8/10
17/17 [==============================] - 23s 1s/step - loss: 0.1641 - accuracy: 0.9477 - val_loss: 1.7380 - val_accuracy: 0.6727
Epoch 9/10
17/17 [==============================] - 22s 1s/step - loss: 0.1486 - accuracy: 0.9631 - val_loss: 1.5216 - val_accuracy: 0.7091
Epoch 10/10
17/17 [==============================] - 23s 1s/step - loss: 0.1346 - accuracy: 0.9569 - val_loss: 1.5911 - val_accuracy: 0.6909
<keras.src.callbacks.History at 0x7eade6ebee60>
```

Рисунок 5.6 - Навчання моделі

Як видно з виводу методу fit(), значення функції втрат на навчальному наборі знижується з кожною епохою, що свідчить про те, що модель навчається і зменшує помилку в прогнозах. Точність на навчальному наборі збільшується з кожною епохою, що також говорить про те, що модель покращує свої передбачувальні здібності на навчальних даних. Однак на валідаційному наборі результати не такі однозначні. Втрати (loss) на валідації збільшуються після перших епох, а точність (accuracy) не покращується стабільно. Це може свідчити про перенавчання моделі - вона може дуже добре підлаштуватися під навчальні дані, не узагальнюючи нові дані. Але різниця між точністю на навчанні та валідації не настільки велика, що може вказувати на те, що модель не перенавчена сильно, але є потенційний ризик.

Далі використовується метод evaluate() для оцінки продуктивності навченої моделі на наборі тестових даних (рис. 5.7).

```
model.evaluate(test_ds)
```

```
5/5 [==============================] - 3s 351ms/step - loss: 1.2346 - accuracy: 0.6947
[1.2346365451812744, 0.6947368383407593]
```

Рисунок 5.7 - Оцінка навченої моделі

Була проведена візуалізація результатів передбачень моделі на тестовому наборі даних (рис. 5.8). Цей код створює графіки, кожному з яких показано зображення з тестового набору даних із зазначенням передбаченого класу моделлю та її реального класу.

```
import numpy
plt.figure(figsize=(10,10))
for images, labels in test_ds.take(1):
  classification = model(images)
  #print(classification)
  for i in range(9):
    ax=plt.subplot(3,3,i+1)
    plt.imshow(images[i].numpy().astype("uint8"))
    index = numpy.argmax(classification[i])
    plt.title(class_names[index]+" (real: "+ class_names[labels[i]]+")")
```

Рисунок 5.8 - Візуалізація результатів передбачень моделі

5.2 Інтеграція моделі Tensorflow в Android

Щоб використовувати створену модель Tensorflow у застосунку Android, необхідно конвертувати модель з формату TensorFlow/Keras у формат TensorFlow Lite (TFLite), який є компактним форматом для запуску моделей на мобільних пристроях та вбудованих системах, забезпечуючи швидке виконання та мале споживання ресурсів (рис. 5.9).

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

with open("model.tflite",'wb') as f:
  f.write(tflite_model)
```

Рисунок 5.9 - Конвертація моделі у формат TensorFlow Lite

Створений файл model.tflite необхідно зберегти для подальшого використання (рис. 5.10).

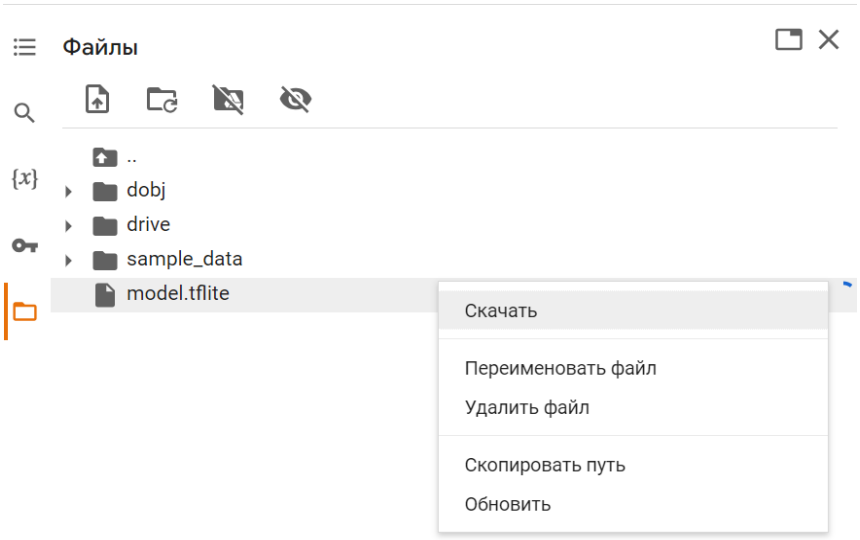

Рисунок 5.10 - Завантаження файлу моделі TensorFlow Lite

В проекті Android необхідно додати файл моделі до проекту. Це можна зробити через меню New > Other > TensorFlow Lite Model (рис. 5.11).

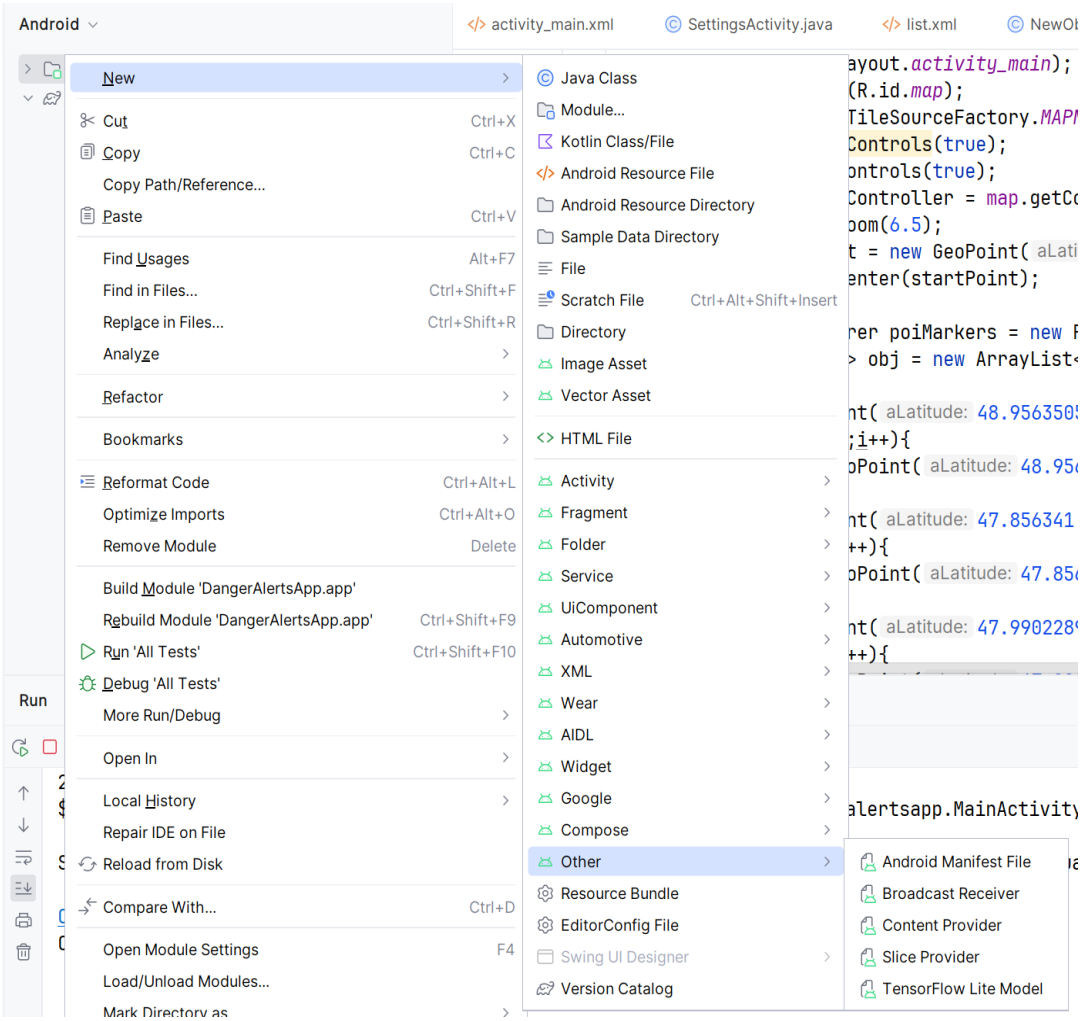

Рисунок 5.11 - Додавання моделі в Android Studio

При додаванні моделі буде створена директорія ml, куди буде завантажений обраний файл моделі TensorFlow Lite. Також можна одразу додати залежності у файл build.gradle (рис. 5.12).

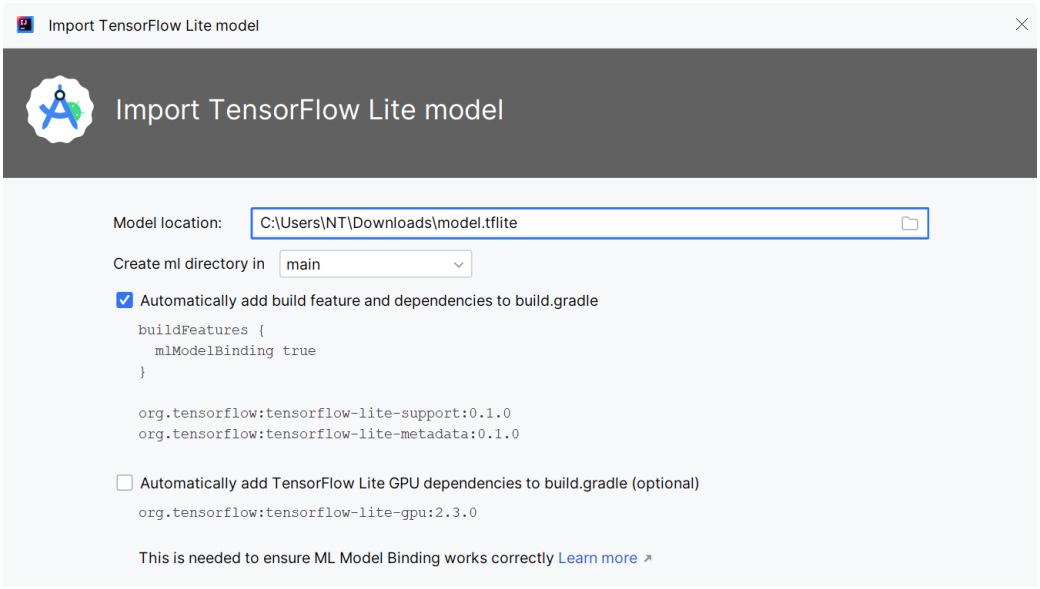

Рисунок 5.12 - Налаштування при додаванні моделі

Android Studio надає код для початку роботи з завантаженою моделлю, який можна використовувати для класифікування зображень у застосунку (рис. 5.13).

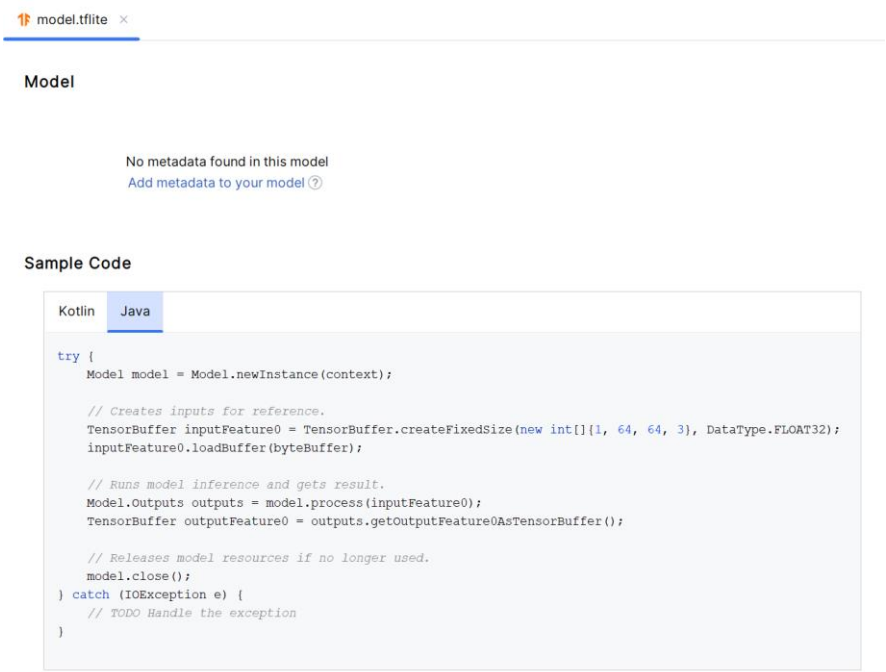

Рисунок 5.13 - Приклад коду для роботи з моделлю TensorFlow Lite

В розмітці сторінки додавання нового небезпечного об’єкта на мапу є дві кнопки - для додавання зображення з камери та з галереї: newImgBtn та fromGalleryImgBtn (рис. 5.14).

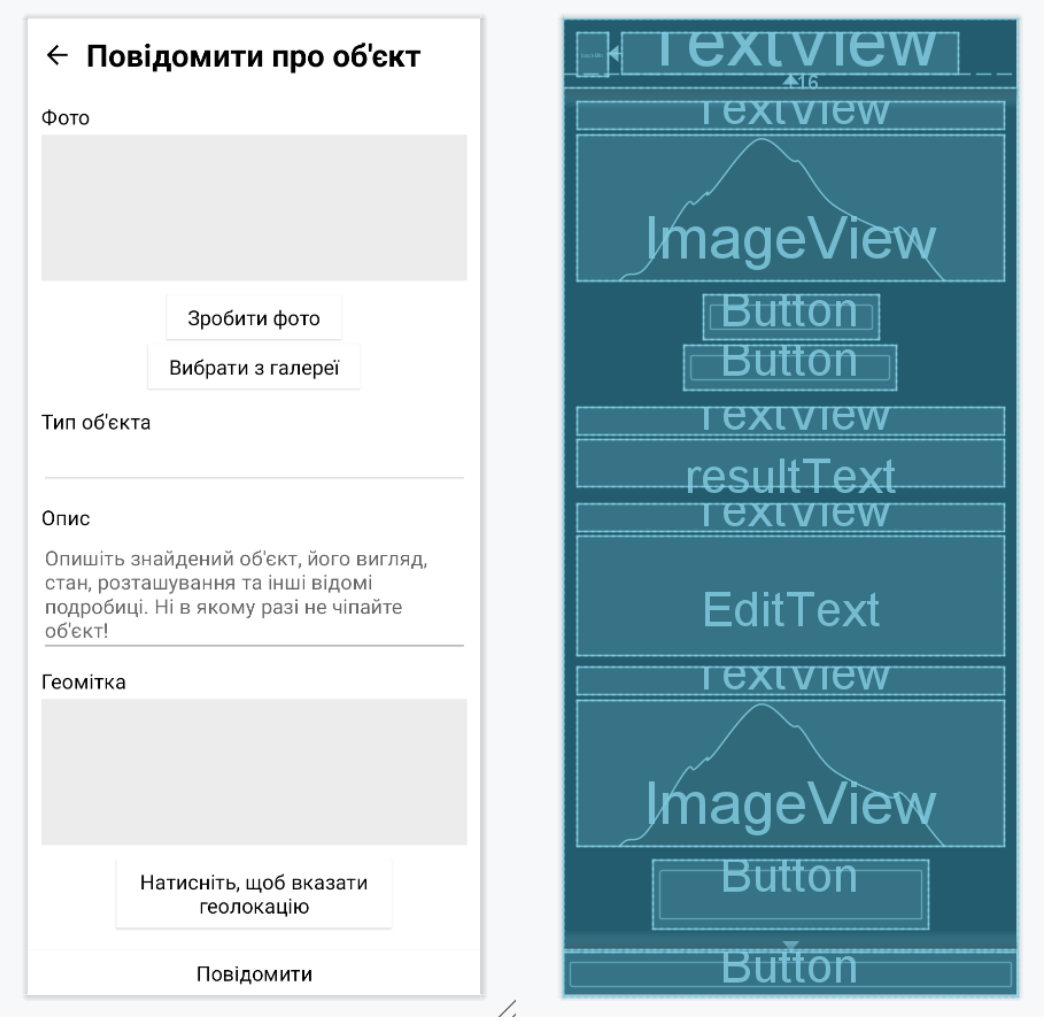

Рисунок 5.14 - Розмітка сторінки додавання нового об’єкта

У відповідному класі прописані обробники onClickListener, які запускають камеру або галерею:

```
newImgBtn.setOnClickListener(view -> {
            if(checkSelfPermission(CAMERA) ==
PackageManager.PERMISSION_GRANTED){
```

```
                Intent cameraIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(cameraIntent, 3);
            } else {
                requestPermissions(new String[]{CAMERA},100);
            }
        });

        fromGalleryImgBtn.setOnClickListener(view -> {
                Intent galleryIntent = new
Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(galleryIntent, 1);
        });
```

При запуску активності за допомогою startActivityForResult(), система Android очікує отримати результат - в даному випадку зроблене або обране зображення. Коли завершена активність закривається та повертається на попередній екран, результат цієї операції буде оброблений методом onActivityResult():

```
 @Override
    protected void onActivityResult(int requestCode, int
resultCode, @Nullable @org.jetbrains.annotations.Nullable Intent
data) {
        if (resultCode == RESULT_OK){
            if(requestCode == 3){
                Bitmap image = (Bitmap)
data.getExtras().get("data");
                int dimension = Math.min(image.getWidth(),
image.getHeight());
                image = ThumbnailUtils.extractThumbnail(image,
dimension, dimension);
                imageView.setImageBitmap(image);
```

```
                image = Bitmap.createScaledBitmap(image,
imageSize,imageSize,false);
                classifyImage(image);
            } else {
                Uri dat = data.getData();
                Bitmap image = null;
                try {
                    image =
MediaStore.Images.Media.getBitmap(this.getContentResolver(),
dat);
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
                imageView.setImageBitmap(image);

                image = Bitmap.createScaledBitmap(image,
imageSize,imageSize,false);
                classifyImage(image);
            }
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
```

В цьому методі перевіряється resultCode, щоб переконатися, що операція завершилася успішно (RESULT_OK). Далі перевіряється requestCode визначення, який запит було виконано. Якщо дані прийшли від камери, витягується зображення з Intent'a, зменшується до квадратного розміру і відображається на imageView. Потім зображення масштабується до заданого розміру і передається класифікацію методом classifyImage(image). Якщо дані прийшли з галереї, отриманий Uri вилучається та використовується для отримання зображення за допомогою MediaStore.Images.Media.getBitmap().

Потім зображення також відображається на imageView, масштабується до заданого розміру і передається класифікацію методом classifyImage(image).

Метод classifyImage відповідає за класифікацію зображень за допомогою завантаженої моделі TensorFlow Lite.

```
private void classifyImage(Bitmap image) {
        try {
            Model model =
Model.newInstance(getApplicationContext());

            TensorBuffer inputFeature0 =
TensorBuffer.createFixedSize(new int[]{1, 64, 64, 3},
DataType.FLOAT32);

            ByteBuffer byteBuffer =
ByteBuffer.allocateDirect(4*imageSize*imageSize*3);
            byteBuffer.order(ByteOrder.nativeOrder());
            int[] intValues = new int[imageSize*imageSize];
            image.getPixels(intValues,0,image.getWidth(), 0, 0,
image.getWidth(),image.getHeight());

            int pixel = 0;
            for (int i=0; i<imageSize;i++){
                for(int j=0; j<imageSize;j++){
                    int val = intValues[pixel++];
                    byteBuffer.putFloat(((val >> 16) &
0xFF)*(1.f/1));
                    byteBuffer.putFloat(((val >> 8) &
0xFF)*(1.f/1));
                    byteBuffer.putFloat((val & 0xFF)*(1.f/1));
                }
            }
            inputFeature0.loadBuffer(byteBuffer);
```

```
            Model.Outputs outputs =
model.process(inputFeature0);
            TensorBuffer outputFeature0 =
outputs.getOutputFeature0AsTensorBuffer();
            float[] confidences =
outputFeature0.getFloatArray();

            int maxPos = 0;
            float maxConfidence = 0;
            for(int i=0;i<confidences.length;i++){
                if(confidences[i]>maxConfidence){
                    maxConfidence = confidences[i];
                    maxPos = i;
                }
            }

            String[] classes = {"артилерійський
снаряд","авіабомба","ракета","ручна граната", "міна"};
            resultText.setText(classes[maxPos]+" (визначено
автоматично)");
            resultText.setEnabled(true);
            model.close();
        } catch (IOException e) {
            Log.e("Error", e.getMessage());
        }
    }
```

Класифікація зображень за допомогою моделі машинного навчання починається з ініціалізації моделі машинного навчання. Потім створюється буфер, в який записуються пікселі зображення для подальшого використання у моделі. Пікселі зображення зберігаються у форматі ARGB в масиві intValues. Цикл перебирає кожен піксель, розбиває його на компоненти (червоний, зелений, синій) та записує у відповідний байтовий буфер у форматі Float.

Вхідні дані (зображення у відповідному форматі) передаються моделі для обробки через model.process(inputFeature0). Результати отримуються у вигляді TensorBuffer. Отримані дані про ймовірності різних класів зберігаються у масиві confidences. Далі визначається клас з найбільшою ймовірністю та відповідний текст виводиться на екран. Після використання модель закривається для оптимізації використання ресурсів.

Цей код працює із зображеннями розміром 64x64 пікселів та класифікує їх на один із п'яти класів: "артилерійський снаряд", "авіабомба", "ракета", "ручна граната" або "міна". Результат класифікації виводиться на екран у вигляді тексту.

5.3 Використання Firebase Cloud

Firebase Cloud – це комплексний набір хмарних інструментів і служб, які надає Google. Він пропонує розробникам широкий спектр функцій і можливостей, які допоможуть швидко й ефективно створювати та масштабувати їхні мобільні та веб-додатки. Одним із основних компонентів Firebase Cloud є Firebase Realtime Database, яка спеціально розроблена для додатків Android.

Firebase Realtime Database — це хмарна база даних NoSQL, яка дозволяє розробникам зберігати та синхронізувати дані в режимі реального часу між кількома клієнтами. Вона надає гнучке та масштабоване рішення для керування даними програми та дозволяє безперебійну співпрацю між користувачами [17].

Використання Firebase як бази даних у додатку для Android має кілька переваг:

Синхронізація в реальному часі: Firebase забезпечує синхронізацію даних у реальному часі, що означає, що будь-які зміни, внесені в базу даних, миттєво відображаються на всіх підключених пристроях. Ця функція особливо корисна для додатків, які потребують оновлення в режимі реального часу, таких як програми для чату або інструменти для спільної роботи.

Підтримка в автономному режимі: Firebase має вбудовану підтримку в автономному режимі, що дозволяє користувачам отримувати доступ до даних і змінювати їх навіть у режимі офлайн. Коли пристрій повторно підключається до Інтернету, Firebase автоматично синхронізує локальні зміни з сервером, забезпечуючи узгодженість даних.

Масштабованість: Firebase — це хмарна база даних, що означає, що вона може легко обробляти великі обсяги даних і розраховувати на зростаючу базу користувачів. Він автоматично масштабується відповідно до вимог вашої програми, забезпечуючи плавну роботу та мінімальний час простою.

Легка інтеграція: Firebase надає повний набір API і SDK, які спрощують процес інтеграції з програмами Android. Він пропонує рідні бібліотеки для Android, що спрощує реалізацію функціональності Firebase і використання її функцій без тривалого програмування чи складних конфігурацій.

Автентифікація та безпека: Firebase пропонує вбудовані служби автентифікації, що дозволяє автентифікувати користувачів за допомогою різних методів, таких як електронна пошта/пароль, вхід із соціальних мереж (наприклад, Google, Facebook) тощо. Він також забезпечує надійні правила безпеки та механізми для контролю доступу до даних і захисту конфіденційної інформації.

Безсерверна архітектура: Firebase піклується про серверну інфраструктуру, обслуговування та масштабування, усуваючи потребу розробникам керувати серверами. Ця безсерверна архітектура дозволяє розробникам більше зосереджуватися на розробці додатків і зменшує операційні витрати.

Аналітика та моніторинг продуктивності: Firebase надає потужні інструменти аналітики для відстеження поведінки користувачів, вимірювання продуктивності програми та отримання цінної інформації. Він пропонує докладні звіти про залучення користувачів, утримання, звіти про збої та моніторинг продуктивності, що дозволяє розробникам оптимізувати свої програми на основі рішень, керованих даними.

Firebase використовує NoSQL базу даних. NoSQL (Not Only SQL) база даних - це тип бази даних, який відрізняється від традиційних реляційних баз даних тим, що не використовує структуровану мову запитів SQL і не використовує схему фіксованих таблиць зі зв'язками між ними.

Основні риси NoSQL баз даних:

Гнучкість у роботі з даними: NoSQL бази даних дозволяють зберігати і обробляти різноманітні типи даних, включаючи структуровані, напівструктуровані і невструктуровані дані. Це означає, що ви можете зберігати дані різної структури без потреби заздалегідь визначати схему таблиць.

Горизонтальне масштабування: NoSQL бази даних розроблені з орієнтацією на горизонтальне масштабування, що означає здатність розподіляти дані на кілька серверів для забезпечення високої доступності та швидкодії. Це робить їх популярними в розподілених системах з великим обсягом даних.

Простота швидкого збереження та отримання даних: NoSQL бази даних зазвичай пропонують простий API для збереження та отримання даних, що спрощує розробку програм, особливо тих, що працюють з великим обсягом даних.

Гнучкість схеми: У NoSQL базах даних немає фіксованої схеми, що дозволяє додавати, змінювати та видаляти поля без необхідності проводити масштабні зміни схеми бази даних. Це робить їх більш гнучкими для розробки додатків, які вимагають частих змін у структурі даних.

Підтримка розподіленості: NoSQL бази даних добре підходять для розподілених середовищ, оскільки вони зазвичай підтримують реплікацію та шарування даних. Це дозволяє забезпечити високу доступність та надійність системи, а також підвищити продуктивність шляхом розподілу навантаження на кілька серверів.

Щоб використовувати Firebase Realtime Database у програмі Android, необхідно виконати такі дії:

Створити проект Firebase: перейти до консолі Firebase (console.firebase.google.com) і створити новий проект. Дати йому назву та вказати ім’я пакета програми.

Налаштувати застосунок: зареєструвати застосунок у Firebase, надавши назву пакета та інші відомості. Завантажити файл google-services.json і помістити його в папку модуля програми.

Додати Firebase SDK: у файл build.gradle застосунка додати залежності Firebase SDK для Realtime Database та синхронізувати проект після додавання залежностей.

Якщо програма вимагає автентифікації користувача, Firebase надає різні методи автентифікації, такі як електронна пошта/пароль, вхід у Google тощо.

Ініціалізувати Firebase у точці входу програми (наприклад, MainActivity) за допомогою методу FirebaseApp.initializeApp().

Використовувати Firebase Realtime Database API для взаємодії з базою даних. Читати та записувати дані можна за допомогою класу DatabaseReference. Наприклад, щоб записати дані, можна викликати setValue() для об’єкта DatabaseReference, а щоб прочитати дані, можна використовувати метод addValueEventListener().

Обробляти оновлення в реальному часі: Firebase Realtime Database забезпечує синхронізацію в реальному часі, дозволяючи отримувати оновлення щоразу, коли змінюються дані. Можна відстежувати зміни даних за допомогою різних прослуховувачів, таких як ValueEventListener або ChildEventListener, і відповідно оновлювати інтерфейс користувача.

Firebase дозволяє визначати правила безпеки, щоб обмежити доступ до бази даних. Можна визначити правила на основі автентифікації користувача, перевірки даних та інших умов для забезпечення безпеки та цілісності даних.

Firebase Realtime Database спрощує процес керування та синхронізації даних у програмах Android, надаючи потужне та масштабоване серверне рішення. Крім того, Firebase Cloud пропонує кілька інших служб, як-от

Firebase Cloud Messaging, Firebase Authentication, Firebase Storage тощо, які можна інтегрувати для покращення функціональності вашої програми.

5.4 Опис застосунку та тестування автоматичної класифікації зображень

При запуску застосунку відкривається головна сторінка з мапою небезпечних об’єктів (рис. 5.15).

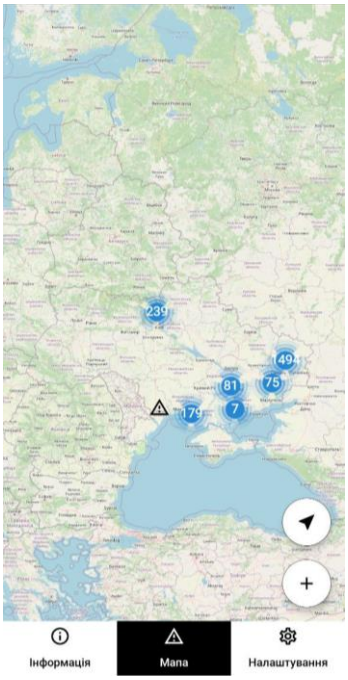

Рисунок 5.15 - Головна сторінка застосунку Danger Alerts

Головна сторінка має меню внизу з трьох пунктів: Інформація, Мапа та Налаштування. Кнопки на мапі дозволяють швидко перейти до поточного місцезнаходження користувача та змінити масштаб мапи. Підтверджені небезпечні об’єкти відображаються символом «!». При збільшенні масштабу об’єкти, що розташовані поруч, об’єднуються в кластери для того, щоб мапа не мала перевантажений деталями вигляд.

Щоб повідомити про знайдений об’єкт, необхідно натиснути “+”. Відкриється нова сторінка застосунку (рис. 5.16).

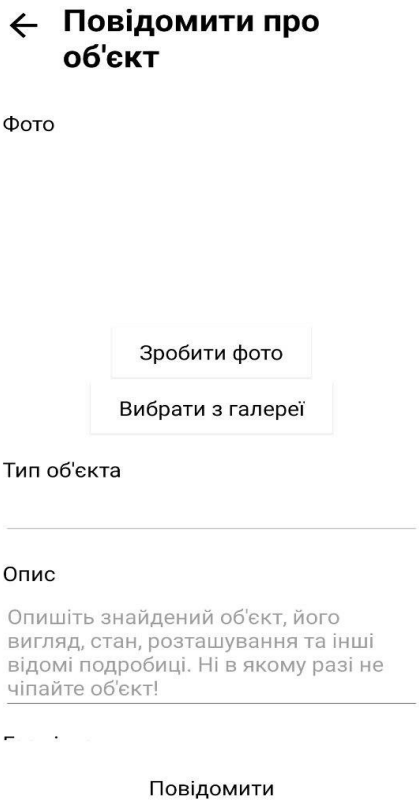

Рисунок 5.16 - Сторінка повідомлення про знайдений об’єкт

Якщо натиснути кнопку Зробити фото, буде відкрито камеру (рис. 5.17).

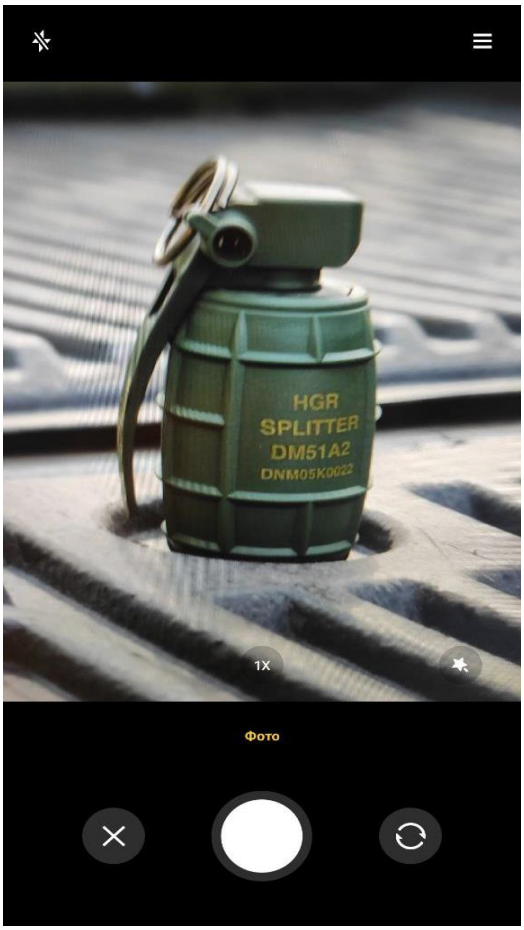

Рисунок 5.17 - Завантаження фото з камери

Після підтвердження, зображення з камери завантажується на сторінку повідомлення про новий об’єкт та у полі «Тип об’єкта» автоматично додається визначене моделлю значення (рис. 5.18). В дужках додається пояснення-попередження про те, що тип визначено автоматично. За необхідності його можна змінити. Адміністраторам застосунку слід уважно розглядати повідомлення з автоматично розпізнанним типом об’єкта.

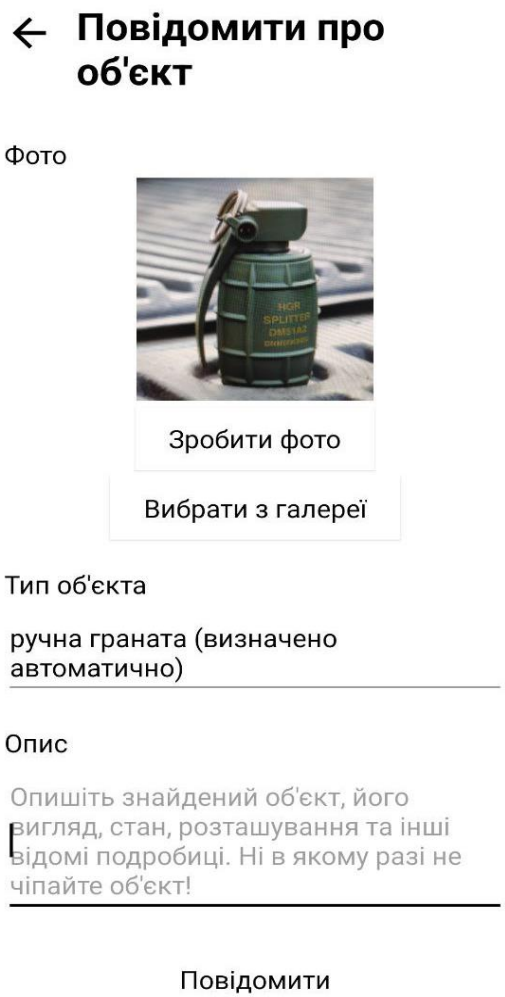

Рисунок 5.18 - Автоматична класифікація завантаженого зображення

Також була додана частина адміністрування. Якщо дані при авторизації користувача співпадають з даними одного з адміністраторів застосунку, на сторінці налаштувань можна побачити нові повідомлення про знайдені об’єкти, які можна переглянути, та редагувати їх статус. Також можна перейти до всіх повідомлень, шукати за назвою, фільтрувати та редагувати записи (рис. 5.19).

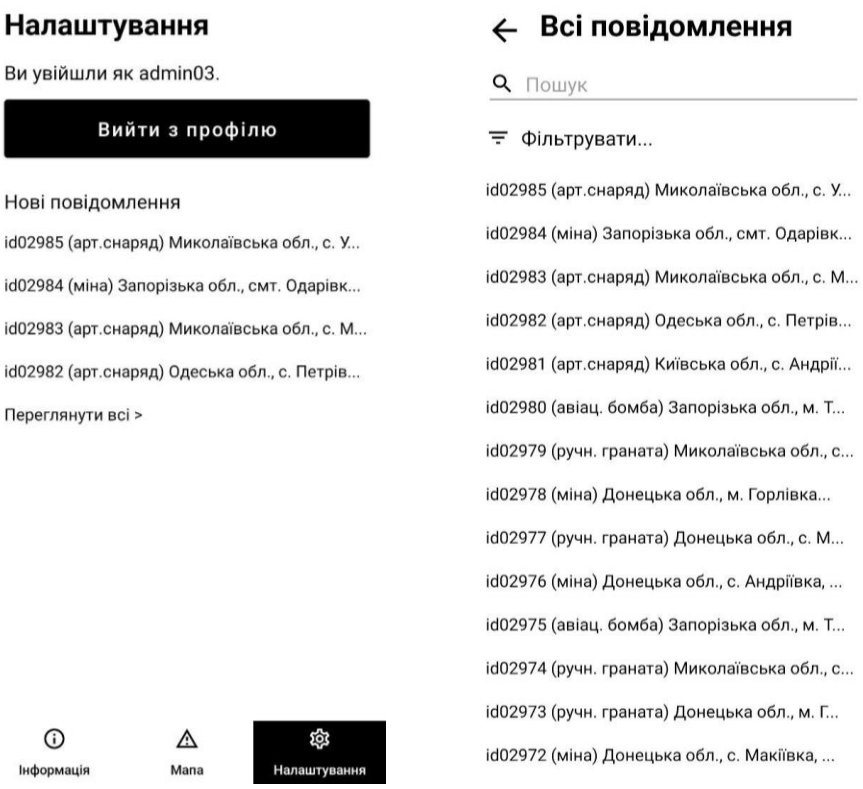

Рисунок 5.19 - Сторінки налаштувань та список повідомлень про знайдені об’єкти (доступ адміністратора)

Після проведення тестування застосунку «Danger Alerts», він виявився зручним у використанні та мав зрозумілий інтерфейс для користувачів.

Загальний результат тестування підтвердив, що застосунок «Danger Alerts» є надійним, зручним та ефективним джерелом інформації про небезпечні об'єкти під час та після воєнних дій. Він допомагає користувачам бути обізнаними та усвідомленими щодо потенційних небезпек і надає необхідні поради щодо безпечного поводження з цими об'єктами. А функція автоматичного розпізнавання полегшить та убезпечить користувачів при повідомленні про знайдені об’єкти.

ВИСНОВКИ

У результаті даної кваліфікаційної роботи була додана функція автоматичного розпізнавання типу об’єкта з використанням технологій штучного інтелекту для застосунку на платформі Android, який дозволяє користувачам отримувати сповіщення про знаходження небезпечних об'єктів на території України та додавати інформацію про знайдені об’єкти.

Після аналізу ситуації у предметній області стало очевидним, що у зв'язку зі збільшенням кількості небезпечних об'єктів внаслідок воєнного конфлікту, необхідний інструмент для сповіщення громадськості про потенційні загрози в їхньому оточенні. З огляду на неосвіченість звичайних громадян про вигляд різних типів небезпечних предметів було додано функцію автоматичного розпізнавання типів об'єктів.

Обґрунтування вибору програмних засобів розробки та технологій було зроблено на основі їхньої популярності, широких можливостей та підтримки спільнотою розробників. Була обрана бібліотека TensorFlow, що обґрунтовано її потужністю, гнучкістю та підтримкою широкого спектру моделей глибокого навчання. TensorFlow Lite забезпечує розгортання на мобільних пристроях, що й було потрібно в цьому проекті для вже існуючого застосунку.

Проектування системи було проведено з урахуванням функціональних та нефункціональних вимог. Були визначені основні компоненти застосунку, створені UML-діаграми прецедентів та активностей. Реалізація застосунку була проведена з використанням зазначених програмних засобів.

В результаті тестування було виявлено, що застосунок успішно виконує свої завдання і може бути використаний для сповіщення користувачів про потенційні небезпеки та автоматичного розпізнавання типів знайдених об’єктів, допомагаючи покращити безпеку та свідомість українців щодо проблеми забруднення країни небезпечними об'єктами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Алакберлі, Н.Я. (2023) Розробка мобільного додатку для сповіщення про знаходження небезпечних об'єктів, Одеський державний екологічний університет. [Електронний ресурс]. – Режим доступу: http://eprints.library.odeku.edu.ua/id/eprint/11999. – Загол. з екрана. – Дата звернення 27.10.2023

2. MineFree 3.0 [Електронний ресурс]. – Режим доступу: https://www.minefree.info – Загол. з екрана. – Дата звернення 27.10.2023

3. Сервіс протимінної діяльності ДСНС [Електронний ресурс]. – Режим доступу: https://mine.dsns.gov.ua – Загол. з екрана. – Дата звернення 27.10.2023

4. PyTorch [Електронний ресурс]. – Режим доступу: https://pytorch.org – Загол. з екрана. – Дата звернення 28.10.2023

5. Caffe2 [Електронний ресурс]. – Режим доступу: https://caffe2.ai – Загол. з екрана. – Дата звернення 28.10.2023

6. Create production-grade machine learning models with TensorFlow [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org – Загол. з екрана. – Дата звернення 28.10.2023

7. Гнатієнко Г.М., Снитюк В.Є. Експертні технології прийняття рішень. – К.: Маклаут, 2008. – 444 с.

8. Кузьменко Б.В., Чайковська О.А. Системи штучного інтелекту: Навч.посібник.-К.:Альтерпрес, 2006.-140 с.

9. Глибовець М. М., Олецький О.В. Штучний інтелект. — Київ : «Києво-Могилянська академія», 2002. – 364 с. – ISBN 966518153X

10. Засоби штучного інтелекту: навч. посіб. / Р. О. Ткаченко, Н. О. Кустра, О. М. Павлюк, У. В. Поліщук ; М-во освіти і науки України, Нац. ун-т «Львів. політехніка». — Львів: Вид-во Львів. політехніки, 2014. – 204 с. – ISBN 978-617-607-692-6

11. Stuart J. Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. – 3. – Pearson, 2015. – ISBN 978-9332543515

12. Претт У. Цифрова обробка зображень/ Претт У – М.: Мир, 2001. – 780с

13. В.В'югін. Математичні основи машинного навчання і прогнозування. – МЦМНО, 2014. – 304 с. – ISBN 978-5-457-71889-0

14. Theodoridis, Sergios; and Koutroumbas, Konstantinos; "Pattern Recognition", 4th Edition. – Academic Press. – 2009. – ISBN 978-1-59749-272-0

15. Фаулер М., Скотт К. UML. Основи. — Пер. с англ. — Київ: Символ-Плюс, 2002. — 192 с., іл. ISBN 5-93286-032-4

16. Welcome To Colaboratory [Електронний ресурс]. – Режим доступу: https://colab.research.google.com – Загол. з екрана. – Дата звернення 05.11.2023

17. Firebase Realtime Database [Електронний ресурс]. – Режим доступу: https://firebase.google.com – Загол. з екрана. – Дата звернення 05.11.2023