МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____Юрій Кондратенко
«____»_____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ДЛЯ БПЛА НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Олександр САТІН

«_ __ _»___червня_____ 2025 р.

Керівник канд. пед. наук, доцент _____ Надія БОЛЮБАШ

«_ __ _»____червня_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем
_____ Юрій КОНДРАТЕНКО
«____» _____ 2025 р.

З А В Д А Н Н Я

на кваліфікаційну роботу здобувача

_____Сатіну Олександру Володимировичу_ __

(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи «Інформаційна система розпізнавання об'єктів для БПЛА на основі згорткових нейронних мереж».

Керівник роботи: Болюбаш Надія Миколаївна, доцент кафедри інтелектуальних інформаційних систем, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «18» листопада 2024 р. № _321_

2. Строк представлення кваліфікаційної роботи студентом «_18_» червня_ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: інформаційна система розпізнавання об'єктів на зображеннях, відео та стрімах, які надходять з БПЛА, із використанням згорткових нейронних мереж; предметна сфера застосування згорткових нейронних мереж для розпізнавання об'єктів, набори даних із розміченими обмежувальними рамками об'єктів та їх класами.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки): дослідження теоретичних засад розпізнавання об’єктів для БПЛА з використанням згорткових нейронних мереж, огляд сучасних досліджень і програмних рішень цій сфері; здійснення аналізу моделей YOLO платформи Ultralytics та обґрунтування вибору моделі YOLOv8 для обробки даних з БПЛА; навчання моделей, здатних розпізнавати об’єкти у зонах масових руйнувань після обстрілів на зображеннях, відео і стрімах, та оцінка їх якості; обґрунтування вибору стеку технологій та розробка інформаційної системи, яка використовує створені моделі для ідентифікації об’єктів при обробці даних з дронів.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

Керівник роботи _____ Надія БОЛЮБАШ
(Особистий підпис) *(Власне ім’я ПРІЗВИЩЕ)*

Здобувач _____ Олександр САТІН
(Особистий підпис) *(Власне ім’я ПРІЗВИЩЕ)*

Дата видачі завдання «_18_» ___грудня__ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інформаційна система розпізнавання об'єктів для БПЛА на основі згорткових нейронних мереж___

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми КРБ. Подання заяви на затвердження теми КРБ	25.09.2024	17.11.2024	Виконано
2	Отримання завдання на виконання КРБ	18.12.2024	20.12.2024	Виконано
3	Аналіз предметної сфери розпізнавання об'єктів із застосуванням CNN та постановка задачі	21.12.2024	30.01.2025	Виконано
4	Огляд існуючих програмних рішень та досліджень архітектур нейромережевих моделей і метрик оцінки їх якості в обробці даних БПЛА	31.01.2025	17.02.2025	Виконано
5	Вибір технологій та інструментальних засобів розробки системи	18.02.2025	28.02.2025	Виконано
6	Створення дизайну, проєктування та програмна реалізація, тестування	01.03.2025	15.04.2025	Виконано
7	Робота над розділами кваліфікаційної роботи	16.04.2025	15.05.2025	Виконано
8	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	Виконано
9	Корегування роботи за результатами попереднього захисту	17.05.2025	29.05.2025	Виконано
10	Другий попередній захист КР	30.05.2025	30.05.2025	Виконано
11	Остаточне оформлення КР та слайдів доповіді до захисту	31.06.2025	10.06.2025	Виконано
12	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	11.06.2025	Виконано

Керівник роботи _____ Надія БОЛЮБАШ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Олександр САТІН
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
« 30 » січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи

здобувача групи 401 ЧНУ ім. Петра Могили

Сатіна Олександра Володимировича

на тему: **«ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ДЛЯ БПЛА НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ»**

Дана кваліфікаційна робота присвячена розробці системи розпізнавання об'єктів на зображеннях, відео та стрімах, які надходять з БПЛА, із використанням згорткових нейронних мереж у режимі реального часу. Що є актуальним в умовах стрімкого зростання сфери застосування БПЛА у різних сферах сучасного суспільства, зокрема, у зонах стихійного лиха та масових руйнувань після обстрілів у зонах ведення бойових дій.

Об'єкт роботи – процес розпізнавання об'єктів при обробці даних, які надходять з БПЛА.

Предмет роботи – нейромережеві моделі та програмні засоби розпізнавання об'єктів у місцях масових руйнувань на зображеннях, відео та стрімах БПЛА.

Мета роботи – підвищення точності розпізнавання об'єктів у зонах масових руйнувань після обстрілів шляхом розробки інформаційної системи на основі згорткових нейронних мереж Ultralytics YOLO.

Структура кваліфікаційної роботи включає вступ, чотири розділи, висновки та додатки. У першому розділі розкрито теоретичні засади розпізнавання об'єктів для БПЛА з використанням згорткових нейронних мереж, досліджено можливості комп'ютерного зору та існуючі програмні рішення в обробці даних БПЛА. У другому розділі обґрунтовано вибір моделі Ultralytics YOLOv8 для розпізнавання об'єктів на зображеннях, відео та стрімах, які надходять із БПЛА. У третьому розділі описано навчання моделей YOLOv8 для розпізнавання даних з дронів та оцінено їх якість. У четвертому розділі обґрунтовано вибір стеку технологій та описано розробку інформаційної системи, яка використовує розроблені моделі для обробки даних з БПЛА.

Кваліфікаційна робота містить ___ сторінок (без додатків), ___ рисунків, ___ таблиць, ___ джерел та ___ додатків.

Ключові слова: згорткова нейронна мережа, розпізнавання, набір даних, ідентифікація, безпілотний літальний апарат.

ABSTRACT

for qualification work

of a student of 401 group at Petro Mohyla Black Sea National University

Satina Oleksandra Volodymyrovycha

Theme: «An object recognition information system for UAVs based on convolutional neural networks»

This bachelor's qualification work is devoted to the development of a system for recognizing objects in images, videos and streams coming from UAVs using convolutional neural networks in real time. Which is relevant in the conditions of the rapid growth of the scope of UAV applications in various areas of modern society, in particular, in areas of natural disaster and mass destruction after shelling in combat zones.

Object of work – the process of object recognition during the processing of data received from UAVs.

Subject of work – neural network models and software for recognizing objects in places of mass destruction from images, videos and UAV streams.

The purpose of this work is to increase the accuracy of object recognition in areas of mass destruction after shelling by developing an information system based on Ultralytics YOLO convolutional neural networks.

The structure of the bachelor's work includes an introduction, four sections, conclusions and appendices. The first section reveals the theoretical principles of object recognition for UAVs using convolutional neural networks, explores the capabilities of computer vision and existing software solutions in UAV data processing. The second section justifies the choice of the Ultralytics YOLOv8 model for object recognition in images, videos, and streams coming from UAVs. The third section describes the training of YOLOv8 models for recognizing data from drones and assesses their quality. The fourth section justifies the choice of a technology stack and describes the development of an information system that uses the developed models for processing data from UAVs.

The qualification work contains ___ pages (without appendices), ___ figures, ___ tables, ___ sources and ___ appendices.

Keywords: convolutional neural networks, recognition, dataset, identification, unmanned aerial vehicle.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 4
ВСТУП 5
1 ТЕОРЕТИЧНІ ЗАСАДИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ 8
1.1 Використання БПЛА у місцях руйнувань після бойових дій 8
1.2 Системи зв'язку безпілотних літальних апаратів 10
1.3 Існуючі програмні рішення для обробки даних, отриманих з БПЛА 16
1.4 Можливості комп'ютерного зору в розпізнаванні об'єктів із зображень та відео БПЛА 21
1.5 Постановка задачі 26
Висновки до розділу 1 27
2 МОДЕЛІ ULTRALYTICS YOLO 29
2.1 Еволюція моделей YOLO 29
2.2 Архітектура моделі YOLOv8 та метрики оцінки її якості 36
Висновки до розділу 2 46
3 НАВЧАННЯ ТА ОЦІНКА МОДЕЛЕЙ YOLOV8 48
3.1 Підготовка наборів даних та моделей до навчання 48
3.2 Навчання моделей YOLOv8n та YOLOv8s 55
Висновки до розділу 3 63
4 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОБРОБКИ ДАНИХ БПЛА 64
4.1 Моделювання інформаційної системи 64
4.2 Інструментальні засоби розробки системи 66
4.3 Програмна реалізація застосунку для розпізнавання даних з БПЛА 71
4.4 Інтерфейс користувача та тестування застосунку 80
Висновки до розділу 4 86
ВИСНОВКИ 88

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 91

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БПЛА – Безпілотний літальний апарат

CNN – Convolutional Neural Networks

COCO – Common Objects in Context

FPS – Frame Per Second

NMS – Non-Maximum Suppression

UAV – Unmanned Aerial Vehicle

YOLO – You Only Look Once

ВСТУП

Актуальність. У сучасних умовах безпекових викликів, пов’язаних із бойовими діями та іншими серйозними катастрофами, питання оперативного реагування на масові руйнування набуває особливої актуальності. В таких ситуаціях вкрай важливо забезпечити швидкий збір достовірної інформації про масштаби пошкоджень, кількість постраждалих та загальну ситуацію на місцевості. Одним із найбільш ефективних інструментів, який сьогодні активно використовується в цій сфері, є безпілотні літальні апарати (БПЛА).

Ефективне використання БПЛА у рятувальних операціях потребує надійного програмного супроводу. До найбільш відомих програмних рішень для обробки даних, які надходять з БПЛА, відносять Pix4Dmapper, Pix4Dfields, Pix4Dcloud, Pix4Dreact, Pix4Dsurvey, Pix4Dcatch, Pix4Dmatic, Pix4Dcapture та Pix4Dengine. Проте існуючі системи мають низку обмежень, які істотно знижують їх ефективність саме в контексті пошуково-рятувальних операцій у зонах масових руйнувань. Оскільки більшість із них є комерційними, зосереджені на картографуванні, а не на оперативному виявленні окремих об'єктів, потребують потужного серверного обладнання для обробки даних та мають обмежену адаптивність до особливих умов зйомки. Тому існує потреба у розробці спеціалізованих програмних засобів, оптимізованого для задач виявлення людей та автотранспорту на відео з БПЛА в умовах масових руйнувань.

Сучасне програмне забезпечення для виявлення та розпізнавання об’єктів у базується на методах та моделях машинне навчання, безумовним лідером серед яких є згорткові нейронні мережі (CNN). До відомих моделей CNN відносять LeNet, AlexNet, ZfNet, VGG, GoogleLeNet. Для розпізнавання даних, які відслідковує БПЛА, найбільш прийнятними є моделі YOLO, сучасні версії яких можуть здійснювати виявлення об'єктів, їх сегментацію і класифікацію на зображеннях та потокових відео. Моделі останніх версій Ultralytics YOLO мають оптимізовані архітектури, забезпечуючи баланс між точністю та швидкодією, що є

критичним під час проведення рятувальних операцій. Інші архітектури: RetinaNet, EfficientDet, CenterNet, мають переваги в точності, однак поступаються YOLO у швидкодії.

Це обумовило **мету роботи**, яка полягає у підвищенні точності розпізнавання об’єктів у зонах масових руйнувань після обстрілів шляхом розробки інформаційної системи на основі згорткових нейронних мереж Ultralytics YOLO.

Відповідно до поставленої мети було сформульовано **завдання**:

– дослідити теоретичні засади розпізнавання об’єктів для БПЛА з використанням згорткових нейронних мереж, здійснити огляд сучасних досліджень і програмних рішень цій сфері;

– провести аналіз моделей YOLO платформи Ultralytics та обґрунтувати вибір моделі YOLOv8 для обробки даних з БПЛА;

– здійснити навчання моделей, здатних розпізнавати об’єкти у зонах масових руйнувань після обстрілів на зображеннях, відео і стрімах, та оцінити їх якість;

– обґрунтування вибір стеку технологій та розробити інформаційну систему, яка використовує створені моделі для ідентифікації об’єктів при обробці даних з дронів.

Об’єкт роботи – процес розпізнавання об’єктів при обробці даних, які надходять з БПЛА.

Предмет роботи – нейромережеві моделі та програмні засоби розпізнавання об’єктів у місцях масових руйнувань на зображеннях, відео та стрімах БПЛА.

Методологічною основою дослідження є загальнонаукові методи та методи і нейромережеві моделі в обробці даних, які надходять з БПЛА, які дозволили комплексно вивчити предмет і об’єкт роботи, дослідити можливості розпізнавання об’єктів із використанням згорткових нейронних мереж.

Практичне значення отриманих результатів полягає в тому, що розроблена інформаційна системи може бути використана для підтримки прийняття рішень у

надзвичайних ситуаціях, що базуються на використанні БПЛА і нейромережевих моделей для об’єктів на зображеннях, відео та стрімах у режимі реального часу.

Структура кваліфікаційної роботи. Відповідно до мети, завдань і предмета роботи, кваліфікаційна робота складається із вступу, чотирьох розділів, висновку, списку використаних джерел та __ додатків. Загальний обсяг роботи – __ сторінок, із них основного тексту – __ сторінок. Кількість використаних джерел – __.

1 ТЕОРЕТИЧНІ ЗАСАДИ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ

1.1 Використання БПЛА у місцях руйнувань після бойових дій

БПЛА дозволяють здійснювати розвідку великих територій без залучення значних людських ресурсів та без наражання операторів на небезпеку. Завдяки можливості здійснювати огляд з висоти, дрони забезпечують панорамний огляд зон руйнувань, що є надзвичайно важливим у випадках, коли доступ до місця подій утруднений через завали, пожежі чи нестабільні конструкції будівель.

Досягнення в цій галузі дають можливість здійснювати політ в автоматичному режимі від зльоту до посадки, вирішувати цілу низку складних і специфічних завдань, які багато в чому практично важко здійснити, а часто й нездійсненну самою людиною.

БПЛА включає в себе власне БПЛА, пункт управління (пульт оператора, приймальна апаратура), систему зв’язку з БПЛА (це може бути прямий радіозв'язок або супутниковий зв'язок), програмне забезпечення для обробки отриманих результатів, додаткове обладнання, необхідне перевезення або обслуговування БПЛА.

Кожна країна в залежності від потреб та наявності ресурсів визначає кількість та номенклатуру необхідного БПЛА, їх тип, модифікацію, технічні параметри та комплектність. На сьогоднішній день найбільших успіхів у створенні комплексів з БПЛА досягли США, Ізраїль, Франція, Німеччина, Великобританія, Китай. Усього ж понад 30 країн світу розробляють близько 300 комплексів та понад 70 літакового, мультироторного, аеростатичного, конвертопланового типів та гібридних моделей БПЛА, що відрізняються конструкцією та принципом роботи, зльоту/посадки та призначення.

Технологічний розвиток БПЛА дозволяє оснащувати їх сучасними сенсорами – як камерами високої роздільної здатності у видимому спектрі, так і

тепловізійними камерами для роботи у нічний час або в умовах поганої видимості. Це розширює функціональність дронів і дає змогу ефективно працювати як вдень, так і вночі. Переваги застосування дронів для пошуку та порятунку:

– висока швидкість та точність надання результатів, значне зниження часу пошуку, при цьому БПЛА може виявити об'єкти в найскладніших і найдоступніших місцях;

– робота у межах міста, зокрема – на гранично малих висотах, маневреність і малий розмір дозволяють дрону працювати в умовах, недоступних для пілотованих вертольотів;

– виконує завдання в епіцентрі лиха без ризику для співробітників;

– виявлення поза межами видимого спектра завдяки використанню тепловізора;

– невисока вартість експлуатації, головна причина – відсутність у конструкції БПЛА великої кількості механічних деталей, схильних до зносу.

У контексті рятувальних робіт після обстрілів або серйозних руйнувань основними об'єктами інтересу є люди, які можуть потребувати допомоги, а також транспортні засоби, що можуть вказувати на місця концентрації постраждалих чи використовуватись для евакуації (рис. 1.1). Виявлення цих об'єктів на основі аналізу відеопотоку з БПЛА дозволяє оперативно приймати рішення та ефективніше координувати дії рятувальних служб.

Особливу складність становлять умови роботи в таких зонах: наявність великої кількості завалів, нестабільна структура об'єктів, значні візуальні перешкоди, зміни освітлення та погодних умов. У цих умовах ручний аналіз відеоматеріалів втрачає ефективність, а оперативність ухвалення рішень стає критично важливою.

Застосування інтелектуальних алгоритмів обробки зображень, здатних автоматично виявляти об’єкти на кадрах відеопотоку в реальному часі, є перспективним напрямком удосконалення систем БПЛА для використання у

надзвичайних ситуаціях. Це створює передумови для переходу від систем спостереження до систем підтримки прийняття рішень.

Рисунок 1.1 – Вигляд села під час обстрілу з БПЛА

Таким чином, використання БПЛА у поєднанні з технологіями штучного інтелекту відкриває нові можливості у сфері пошуку та порятунку людей у зонах руйнувань внаслідок бойових дій або інших надзвичайних подій.

1.2 Системи зв’язку безпілотних літальних апаратів

Системи зв’язку БПЛА має важливе значення для його функціональності, дозволяючи передавати критично важливі дані, такі як координати GPS, телеметрію та відеопотоки, які необхідні для ефективної роботи дрону [1]. Вони також гарантують, що оператори можуть надійно контролювати рухи БПЛА, регулювати його висоту та вносити зміни до його курсу в режимі реального часу.

Системи зв’язку дронів відповідають за виконання трьох основних функцій:

– передача даних: БПЛА оснащені датчиками або камерами, які збирають дані під час польоту та передають їх через систему зв’язку оператору чи іншим системам реального часу. Це особливо важливо для таких програм, як повітряне спостереження, картографування та інспекція, де необхідний аналіз даних у режимі реального часу;

– телеметрія та моніторинг: відправка даних назад на контролер або наземну станцію про свій поточний стан, наприклад – рівень заряду батареї, висоту та швидкість тощо;

– управління та командування: базова комунікація, яка дозволяє оператору керувати дроном.

У міру того, як БПЛА стають все більш складними, удосконалюються системи зв’язку з ними. Сучасні дрони часто підтримують кілька частот, передові протоколи шифрування та адаптивні методи модуляції для підтримки надійних та безпечних каналів зв’язку. Необхідність у безперебійному обміні даними стає ще більш важливою зі зростанням використання дронів у відповідальних ситуаціях, таких як пошуку та порятунок людей у зонах руйнувань внаслідок бойових дій або інших надзвичайних ситуацій.

До основних компонентів систем зв’язку дронів відносять контролер, приймач, антену, бортову система управління польотом, GPS і системи позиціонування, телеметричні системи та системи передачі даних (рис. 1.2). Основні компоненти функціонують як єдина система, яка дозволяє БПЛА виконувати складні завдання з високою точністю і надійністю. Для більш просунутих дронів GPS і системи передачі даних відіграють вирішальну роль, особливо при виконанні автономних місій або передачі відео в реальному часі.

GPS та системи позиціонування призначені для допомоги у навігації та стабілізації точного розташування, висоти та швидкості, що необхідно для точного управління польотом. Телеметричні системи безперервно передають дані з БПЛА на контролер, включаючи висоту, швидкість, стан батареї та координати GPS. Системи передачі даних включають канали передачі даних, необхідні для передачі

додаткових даних, таких як прямі відеопотоки чи дані датчиків, на наземну станцію управління чи оператору.

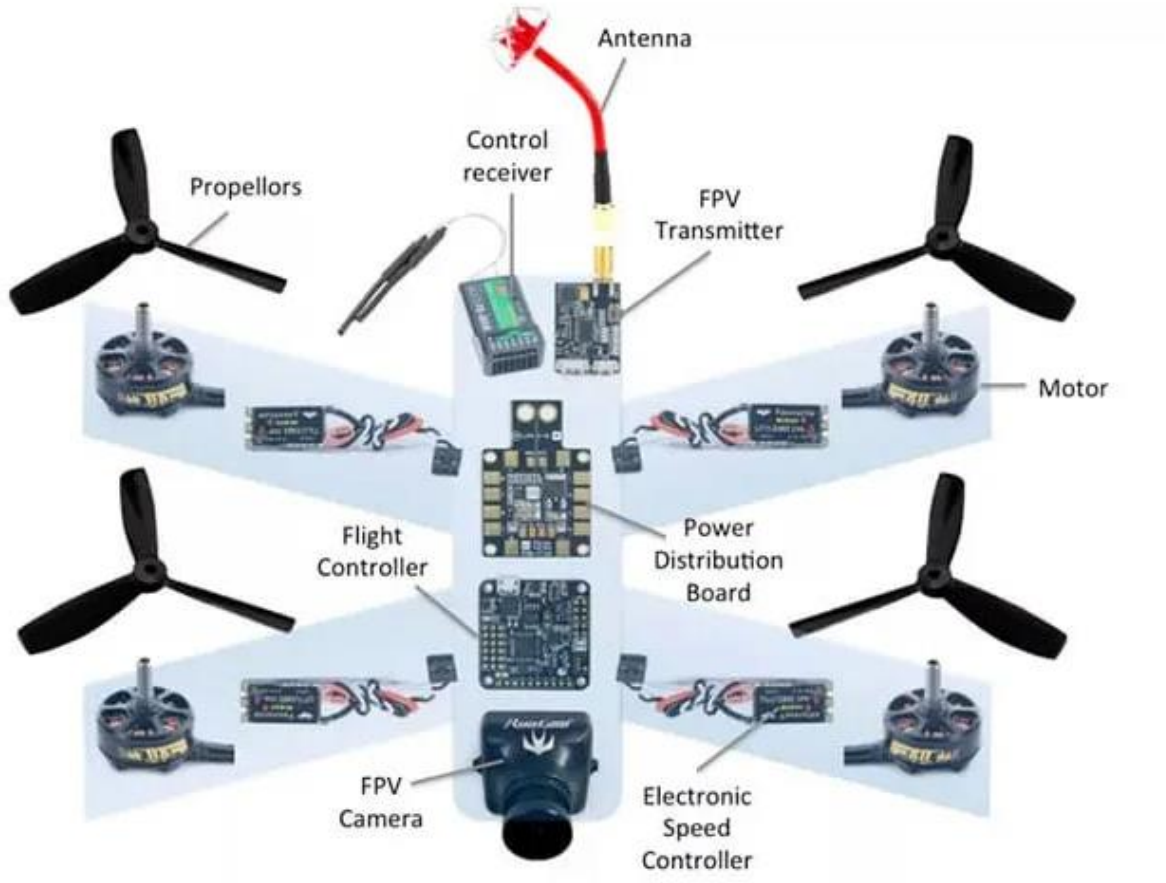

Рисунок 1.2 – Основні компоненти системи зв’язку БПЛА [1]

Ще одним важливим компонентом у роботі БПЛА є протоколи в системах зв’язку, які є набором правил і стандартів, що визначають, як дані передаються і приймаються між пристроями дрону (рис. 1.3). Зв’язок дронів ґрунтується на різних протоколах, які виконують різні функції в системі [1]. Вибір протоколу залежить від таких факторів, як тип БПЛА, його застосування та необхідний рівень керування або передачі даних. Вибір протоколу залежить від конкретних потреб БПЛА та його застосування. У міру того, як дрони стають все більш складними, часто виникає необхідність в інтеграції декількох протоколів для обробки різних аспектів комунікації. Зробимо огляд найчастіше використовуваних протоколів.

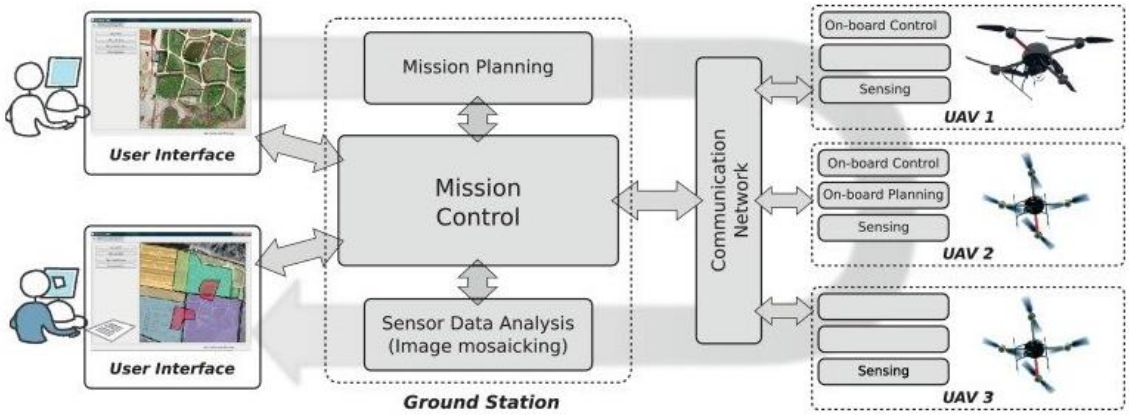

Рисунок 1.3 – Протоколи зв’язку, використовувані в БПЛА

MAVLink (Micro Air Vehicle Link) – один із найбільш широко використовуваних протоколів зв’язку в індустрії дронів з відкритим вихідним кодом, розроблений для полегшення зв'язку між дронами та наземними станціями управління або між самими дронами у роєвих операціях. MAVLink підтримує обмін даними телеметрії, керування польотами та потокову передачу відео. Підтримує двосторонню передачу даних, включає певні типи повідомлень для телеметрії, GPS, даних датчиків та команд управління, споживає мінімальну смугу пропускання, що робить його придатним для дронів, що працюють в умовах обмеженого підключення.

DSMX (цифрова спектральна модуляція) – протокол, розроблений Spektrum, працює на частоті 2,4 ГГц та використовує технологію стрибків частоти для зниження перешкод, забезпечуючи стабільний та надійний зв’язок. DSMX цінується за швидкий час відгуку та безпечне з’єднання, що робить його ідеальним для програм, що потребують керування в реальному часі.

S.Bus (послідовна шина): ще один протокол, розроблений Futaba, широко використовується в дронах, особливо в системах, що вимагають складних налаштувань управління. S.Bus дозволяє передавати кілька каналів по одній сигнальній лінії, що знижує складність проводки та покращує якість сигналу. Він часто використовується у поєднанні з іншими протоколами, такими як MAVLink, для обробки додаткових вимог до даних у сучасних дронах.

Lightbridge ma Ocusync (від DJI): це протоколи, розроблені DJI, спеціально оптимізовані для передачі відео високої чіткості та зв’язку на великі відстані. Lightbridge та Ocusync розроблені для забезпечення низької затримки, високої пропускної спроможності та можливостей великої дальності, які необхідні для професійних дронів та у застосунках для надання допомоги у зонах стихійного лиха.

Протоколи на основі *Zigbee ma Wi-Fi* менш поширені в БПЛА, але використовуються для спеціалізованих застосунків, де достатньо недорогої та малої дальності зв’язку. Ці протоколи зазвичай зустрічаються в побутових дронах або дронах, призначених для використання в приміщеннях, оскільки вони пропонують достатню пропускну здатність для керування та базової телеметрії на обмежених відстанях.

У БПЛА канал передачі даних використовує радіочастоти для передачі та отримання інформації, які є основою систем зв’язку дронів (рис. 1.4). У системі передачі даних використовуються різні частоти [2]. Використовувані частоти залежить від марки і навіть від функціональності БПЛА. Наприклад, системи DJI використовують 2,4 ГГц для управління БПЛА та 5 ГГц для передачі відео. Таке налаштування надасть користувачеві діапазон приблизно 4 милі. Однак, якщо використовувати 900 МГц для управління БПЛА та 1,3 ГГц для відео, можна досягти відстані 20+ миль.

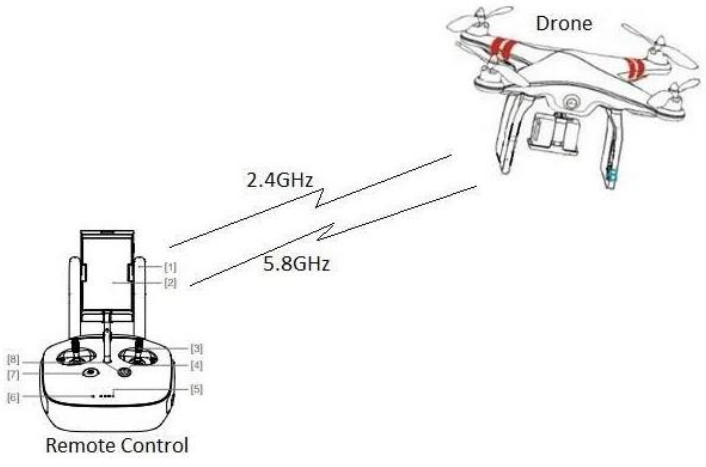

Рисунок 1.4 – Поширені частоти для БПЛА

Кожен з радіодіапазонів має своє застосування. Більш низькі частоти, такі як 433 МГц і 900 МГц, переважні для застосунків із більшим радіусом дії, в той час як більш високі частоти, такі як 5,8 ГГц, ідеально підходять для задач з близькою дистанцією та високою пропускною здатністю. Дрони, що працюють у густонаселених або міських районах, можуть виграти від використання частот із кращою проникною здатністю, таких як 2,4 ГГц або 900 МГц. Для застосунків із великим об'ємом відео або даних оптимальними є вищі частоти, такі як 5,8 ГГц, завдяки їхній пропускній здатності, незважаючи на обмежений діапазон.

Передача даних у реальному часі включає відправлення даних з дрону через радіочастоти, такі як 2,4 ГГц, 5,8 ГГц або, все частіше, мережі 4G і 5G. Для дронів з камерами система одночасно передає живі відеопотоки, які вимагають високої пропускної здатності та низької затримки для забезпечення плавних та точних візуальних ефектів. Для досягнення цього дрони покладаються на модулі передачі даних, які складаються з передавачів та приймачів, здатних відправляти та отримувати дані на високих швидкостях. Ці модулі часто включають протоколи виправлення помилок для забезпечення цілісності даних, компенсуючи потенційне погіршення сигналу чи перешкоди. Крім передачі даних назад на контролер, деякі просунуті дрони можуть передавати цю інформацію безпосередньо на хмарні сервери або системи керування даними, що дозволяє проводити аналіз даних у реальному часі та віддалений моніторинг групами, що знаходяться далеко від місця експлуатації БПЛА.

Для дронів, оснащених камерами, передача відео в реальному часі ґрунтується на складних технологіях, розроблених для обробки відео високої чіткості (HD) та надвисокої чіткості (UHD). Якість і швидкість відеопотоків залежать від декількох факторів, включаючи пропускну здатність каналу передачі, стандарт стандарту стиснення відео і наявну мережну інфраструктуру.

Для забезпечення передачі відеосигналів з мінімальною затримкою та максимальною чіткістю використовуються різні протоколи.

Протокол RTPS: є ключовим елементом у системах відеоспостереження, забезпечуючи контроль за потоковою передачею відео у реальному часі. RTPS використовують для управління сесіями передачі даних, він діє через Інтернет, забезпечуючи відділений доступ до потоків відеоданих. RTPS створений з розрахунком на сумісність із HTTP для спрощення його інтеграції та аналізу.

DJI Lightbridge і Ocusync: технології, розроблені DJI, широко відомі своєю здатністю передавати HD-відео на відстань у кілька кілометрів. Вони працюють на частотах 2,4 ГГц 5,8 ГГц і використовують адаптивну частотну перебудову для зменшення перешкод, забезпечуючи стабільне високоякісне відео, що підходить для професійних застосунків.

Протоколи на основі Wi-Fi: багато споживчих дронів використовують Wi-Fi для передачі відео, як правило, в діапазонах 2,4 ГГц або 5 ГГц. Хоча Wi-Fi економічно ефективний і забезпечує пристойний діапазон і якість для споживчого використання, він має здатність до перешкод, особливо в густонаселених районах, де співіснують кілька мереж Wi-Fi.

Мережі 4G/5G: останнім часом деякі дрони почали використовувати стільникові мережі для передачі відео [2]. Цей підхід особливо корисний для застосунків на великих відстанях, оскільки мережі 4G і 5G забезпечують широке покриття та високу пропускну здатність. Використання стільникових мереж також дозволяє отримувати живі відеопотоки практично з будь-якої точки з покриттям мережі, що дозволяє виконувати операції за межами прямої видимості (BVLOS).

1.3 Існуючі програмні рішення для обробки даних, отриманих з БПЛА

Ефективне використання БПЛА у рятувальних операціях вимагає не лише сучасного апаратного забезпечення, але й надійного програмного супроводу. Програмне забезпечення має забезпечувати автоматичне планування польотів, управління сенсорами, обробку даних у реальному часі та можливість інтеграції

результатів виявлення об'єктів у загальну систему управління рятувальними роботами.

Обробка даних з БПЛА має високу розрахункову складність, що висуває підвищені вимоги до потужності комп’ютерів, де буде здійснюватися обробка. Тому при використанні власної обчислювальної інфраструктури виникають складності з підбором техніки, оскільки швидкість обробки даних може бути повільною.

Одним із підходів вирішення цієї проблеми є здійснення обробки даних з використанням хмарних сервісів як обчислювальної інфраструктури. Наприклад, обчислювальні вузли Agisoft Cloud дозволяють обробити проект з 1 000 фотографій із роздільною здатністю 40 Мпікс за 3 години 40 хвилин. Водночас Agisoft Cloud має низку бонусів:

– переглядати результати обробки у браузері та надавати доступ до них колегам та клієнтам;

– виконувати вимірювання та векторизацію в браузері;

– вбудувати візуалізацію проекту на сайт.

Проте користуватися сервісом можуть лише власники ліцензій Agisoft Metashape Professional. Дисковий простір для зберігання проектів та фотографій доступний за щомісячною передплатою від $10 за 100 Гбайт до $500 за 5 Тбайт. Для виконання обробки купується час пакетами від $10 за 2 години, тарифікуються лише періоди обчислень без урахування завантаження та скачування даних.

Нові технології докорінно змінюють та знижують собівартість виробничих процесів або ж витрати на виконання різних робіт громадськими службами. Споживачі сьогодні не обмежені у виборі спеціальних програмних програм для обробки зображень і відео. Існує низка популярних комерційних програмних рішень для обробки зображень із БПЛА, таких як DJI Terra, Pix4Dreact, DroneDeploy, FlytBase. Серед лідерів виділяються Pix4D, DroneDeploy та DJI Terra. Кожна з цих програм має свої переваги і недоліки, які визначаються різними

факторами: вартість ПЗ, сумісність датчика і зображення, результат обробки і ряд інших моментів.

Pix4D була створена у 2011 році співробітниками лабораторії комп'ютерного зору EPFL у Швейцарії. На даний момент споживачам добре знайомі такі продукти, як: Pix4Dmapper, Pix4Dfields, Pix4Dcloud, Pix4Dreact, Pix4Dsurvey, Pix4Dcatch, Pix4Dmatic, Pix4Dcapture та Pix4Dengine. Програмні рішення створюються для роботи на різних платформах: комп'ютерах, мобільних та хмарних. Сьогодні очевидне лідерство Pix4D у галузі безпілотних технологій картографії та геодезії. Її програмні продукти добре відомі фермерам, будівельникам, геодезистам, рятувальникам та багатьом іншим фахівцям. Найбільш популярними сьогодні є програмні пакети Pix4Dreact і Pix4Dmapper.

Програмний пакет Pix4Dreact був спеціально розроблений для картографічних робіт із урахуванням потреб та специфічних завдань спеціалістів з надзвичайної допомоги [3]. Звідси одна з його очевидних переваг – швидкість обробки аерофотознімків за короткий час. При цьому сам пакет здатний добре функціонувати на комп’ютерах із середніми параметрами. Pix4Dreact позиціонується як швидке рішення для картографії у надзвичайних ситуаціях, з можливістю роботи без підключення до інтернету. Крім того, користувачам не потрібне підключення до Інтернету для обробки знімків у хмарі.

Області застосування Pix4Dreact для вирішення проблеми оперативного прийняття рішень на основі точної інформації, яка найчастіше виникає при ліквідації надзвичайних ситуацій, забезпеченні безпеки та порятунку людей. Ситуація під час різних надзвичайних ситуацій – пожеж, повеней, землетрусів – змінюється дуже швидко, причому нерідко це торкається ландшафту. В результаті, старі карти місцевості виявляються або взагалі непотрібними, або малопридатними. Саме тоді отримання точних, актуальних геофізичних даних може стати вирішальним фактором, особливо якщо доводиться вирішувати питання життя та смерті людей.

Одна з важливих переваг програмного пакету Pix4Dreact полягає в його сумісності з широким спектром обладнання, можливості обробляти зображення в різних форматах і сумісності з великою кількістю безпілотних платформ. Щоправда, є й один важливий недолік – можливість працювати лише з RGB-зображеннями. Проте цей пакет є комерційним, вартість пакету залежить від вибраного тарифу.

DroneDeploy надає інструменти для аналізу сільськогосподарських, промислових та будівельних об'єктів на основі даних із БПЛА [4]. DroneDeploy є хмарною платформою, або хмарним сервісом, який створений спеціально для картографії та аналізу даних, отриманих з дронів. Програмні рішення DroneDeploy на базі хмарної платформи дозволяють виконувати автоматичну перевірку безпеки польотів, контролювати робочі процеси та відображати їх у реальному часі, а також обробляти дані. Компанія давно та успішно співпрацює з провідними виробниками дронів, включаючи компанію DJI. Це дозволяє розробникам успішно просувати своє спеціалізоване програмне забезпечення кінцевим користувачам із різних сфер діяльності, включаючи сільське господарство, нерухомість, гірничодобувну промисловість, будівництво тощо. Програмні пакети DroneDeploy пропонують простий та зрозумілий для користувача функціонал. При цьому продукція має сумісність з різними форматами зображень.

З DroneDeploy легко отримати результати на рівні зйомки завдяки вбудованій підтримці наземних контрольних точок та технології RTK GPS. Також можна обробляти практично будь-які типи зображень дронів, включаючи теплові та мультиспектральні. До того ж є можливість зберегти завантажені зображення за допомогою DroneDeploy, щоб звернутися до них пізніше. Застосунки DroneDeploy активно використовують в гірничодобувних компаніях, будівельній індустрії, геодезисти, фермери, комунальники, енергетики, фахівці нафтогазової промисловості, страховики та інші професіонали. Однак програмні пакети також є комерційними.

FlytBase, своєю чергою, пропонує модулі з виявлення об’єктів на основі відеопотоку в реальному часі, однак більшість рішень передбачає наявність серверної обробки або потужного інтернет-з’єднання [5].

Серед найбільш відомих рішень слід відзначити також DJI Terra – програмний пакет для створення 2D/3D картографічних моделей місцевості на основі даних БПЛА [6]. DJI Terra є інструментом для картографії та геодезичних досліджень, розроблений компанією DJI. Однією з відмінних рис програми є те, що перед нами продукт типу “все в одному”. Тобто в ньому є всі ресурси, необхідні для планування польотів, створення потрібних зображень та їх обробки для генерації карт, моделей та аналізу даних для більш ефективного прийняття рішень.

Програма DJI Terra є рішенням, яке у плані картографії пропонує результати, аналогічні до DroneDeploy та Pix4Dmapper, включаючи 2D-картографування в реальному часі для оперативного отримання картографічних даних дистанційним чином або в тих випадках, коли оперативність прийняття рішень має вирішальне значення. Інші аналогічні продукти включають 2D-ортофотомозаїки та мультиспектральні реконструкції, а також 3D-моделі та реконструкції з хмари точок.

Проте DJI Terra має низку недоліків перед продукцією DroneDeploy або Pix4Dmapper. Ці недоліки полягають у загальній точності та високій роздільній здатності отриманих карт та моделей. Також відзначають деяку обмеженість щодо використання форматів файлів зображень в порівнянні з програмними пакетами від DroneDeploy і Pix4Dmapper. На практиці користувачі частіше використовують ПЗ від Pix4D, якщо йдеться про картографію в реальному режимі часу, оскільки вони вважають ПЗ від DJI або DD не так гнучким і сумісним порівняно з продуктами від Pix4D.

Здійснений аналіз програмних рішень показав, що існуючі системи мають низку обмежень, які істотно знижують їх ефективність саме в контексті пошуково-рятувальних операцій у зонах масових руйнувань. По-перше, більшість рішень зосереджені на картографуванні, а не на оперативному виявленні окремих об'єктів

на зображенні. По-друге, готові рішення часто потребують потужного серверного обладнання для обробки даних, що унеможливлює їх використання безпосередньо у польових умовах. По-третє, такі системи зазвичай мають обмежену адаптивність до особливих умов зйомки, як-от наявність пилу, диму, нестабільного освітлення. Вони не адаптовані до роботи в умовах часткових руйнувань, задимлення або нічного освітлення, що характерно для зон бойових дій або стихійного лиха. Крім того, комерційна ліцензія таких рішень створює обмеження для їх масштабного застосування в ДСНС.

Саме тому виникає необхідність у розробці власного спеціалізованого програмного модуля, оптимізованого для задач виявлення людей та автотранспорту на відео з БПЛА в умовах масових руйнувань. На відміну від універсальних рішень, розробка буде орієнтована на специфічні вимоги ДСНС: робота у реальному часі, обмеження обчислювальних ресурсів, адаптація до складних середовищ та можливість подальшого донавчання моделі на локальних даних.

Таким чином, запропонована розробка спрямована на вирішення практичних завдань оперативного аналізу ситуації у зонах бойових руйнувань із максимальною ефективністю та мінімальними витратами ресурсів.

1.4 Можливості комп’ютерного зору в розпізнаванні об’єктів із зображень та відео БПЛА

Протягом останніх десятиліть область комп’ютерного зору зробила помітний ривок у задачах виявлення об'єктів та їх семантичного аналізу. Комп’ютерне розпізнавання об'єктів є процесом автоматичного виявлення, класифікації та ідентифікації об’єктів на зображеннях або відео за допомогою методів машинного навчання та штучного інтелекту [7].

Комп’ютерний зір – це область штучного інтелекту, яка включає навчання інтерпретувати та розуміти візуальну інформацію з навколишнього світу.

Обробляючи зображення, отримані за допомогою камер, системи, що використовують комп’ютерний зір, можуть виконувати різні завдання, які раніше залежали від зорового сприйняття людини [8].

Упродовж останніх років тема використання штучного інтелекту в поєднанні з безпілотними літальними апаратами активно розвивається у контексті автоматизації рятувальних операцій. У сценаріях реагування на надзвичайні ситуації, таких як пошуково-рятувальні операції, можливості безпілотних літальних апаратів у поєднанні з комп’ютерним зором можуть урятувати життя. Безпілотники, оснащені тепловими датчиками, можуть швидко сканувати великі площі на предмет теплових сигнатур, ідентифікувати людей, які зазнають лиха або знаходити зниклих безвісти. Можливість швидкого охоплення великих територій у поєднанні з обробкою даних у реальному часі значно підвищує ситуаційну поінформованість рятувальників.

Основні цілі комп’ютерного зору включають виявлення та розпізнавання об'єктів, сегментацію зображень, відстеження руху та тривимірну реконструкцію. Для реалізації цих можливостей алгоритми комп'ютерного зору спираються на машинне навчання, зокрема, на моделі глибокого навчання, такі як нейронні згорткові мережі (англ. Convolutional Neural Networks, CNN). Ці мережі можна навчати на величезних наборах даних для розпізнавання шаблонів, класифікації зображень та складання прогнозів щодо візуального контенту.

Досягнення в галузі апаратних технологій та доступність великих маркованих наборів даних значно покращили продуктивність та застосовність цих алгоритмів. Зокрема, комп’ютерний зір на основі згорткових нейронних мереж дозволяє оперативно ідентифікувати об’єкти на зображеннях з дронів, зокрема людей і транспортні засоби, що є надзвичайно важливим у випадках масових руйнувань.

Згорткові нейронні мережі є на сьогодні одним із найбільш ефективних типів штучних нейронних мереж для обробки візуальної інформації, зокрема зображень та відео. Їх структура і принципи роботи забезпечують можливість автоматичного

виявлення релевантних ознак у даних без необхідності ручного створення ознакових векторів.

Основним елементом CNN є операція згортки. Згортка полягає у послідовному накладенні невеликого ядра на вхідне зображення та обчисленні скалярного добутку значень пікселів і ваг ядра. Результатом є карта ознак, яка відображає присутність певних локальних візуальних структур у різних ділянках зображення.

Одним із важливих етапів роботи згорткової мережі є також застосування операції пулінгу. Пулінг виконується після одного або кількох згорткових шарів для зменшення розмірності простору ознак та зниження обчислювальної складності. Найпоширенішим є max-pooling, при якому з кожного регіону зберігається лише максимальне значення. Це дозволяє зберігати найбільш характерні ознаки та підвищувати стійкість до невеликих зсувів на зображенні.

Архітектура типової згорткової нейронної мережі складається з чергування згорткових шарів та шарів пулінгу, які після кількох повторень завершуються повнозв’язними шарами (англ. Fully Connected Layers) (рис. 1.5). На перших рівнях мережа вчиться розпізнавати простіші патерни (лінії, кути), а на глибших – складніші об’єкти і контексти.

Важливою складовою роботи CNN є функція активації, яка надає мережі здатність до моделювання нелінійних залежностей. Під час навчання моделей нейронної мережі доступним є широкий спектр функцій активації. В основному використовуються функції активації sigmoid, tanh, ReLU та leaky ReLU.

Найчастіше у згорткових мережах використовується функція ReLU, яка реалізує простий пороговий перехід у нулі й обчислюється за формулою:

$$f(x) = max(0, x) \tag{1.1}$$

де x – вхідне значення, яке подається на функцію активації;

$f(x)$ – результат роботи функції активації ReLU.

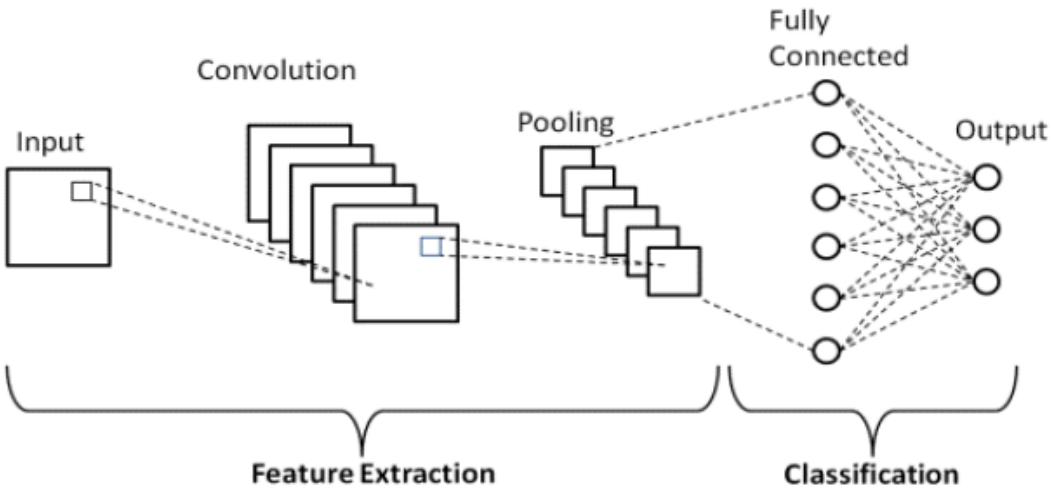

Рисунок 1.5 – Типова архітектура згорткової нейронної мережі [9]

У вихідному шарі зазвичай застосовують функцію активації Softmax для нормалізації вихідних даних мережі до розподілу ймовірностей по класам, які розпізнаються [10].

Процес навчання моделі CNN здійснюється на наборах даних великих обсягів. Ефективне навчання моделі базується на правильному налаштуванні процесу навчання, яке включає оптимальний вибір функції втрат та оптимізатора.

Для задачі багатокласової класифікації як функцію втрат часто використовують перехресну ентропію, яка кількісно визначає похибку моделі, вимірюючи різницю між передбаченими та справжніми результатами.

Оптимізатор відповідає за ітераційне оновлення параметрів моделі, щоб мінімізувати функцію втрат. Для навчання згорткової нейронної мережі найбільш розповсюдженим оптимізатором є стохастичний градієнтний спуск (англ. Stochastic Gradient Descent, SGD) та його модифікації: Adam, RMSprop, AdaGrad, AdaDelta [11].

Відомими моделями згорткових нейронних мереж є LeNet, AlexNet, ZfNet, VGG, GoogleLeNet, які, маючи різну архітектуру, перетворюють вхідні дані у вихідні, ієрархічно виділяючи та агрегуючи ознаки шляхом підвищення рівня абстракції даних у напрямку від входів до виходів мережі [12]. Для розпізнавання

даних, які відслідковує БПЛА, найбільш прийнятною моделлю є YOLO (англ. You Only Look Once) – нейронна мережа, призначена для роботи з об’єктами на зображеннях і може вирішувати такі завдання: детекція: виявлення об'єктів; сегментація: розділення зображення на області, що належать до кожного об'єкта; класифікація: визначення що знаходиться на зображенні; пошук ключових точок: для визначення пози об’єкту; трекінг об’єктів: потокова обробка, при якій для кожного об’єкта можливо зберігати та використовувати історію розташування [13].

Алгоритм YOLO, запропонований Redmon et al., став одним із базових досліджень у сфері детекції об’єктів, пропонуючи концепцію єдиного проходу згорткової мережі через зображення для одночасного виявлення й локалізації об’єктів. Його пізніші модифікації підвищили точність і швидкодію, що робить їх особливо актуальними для розгортання на БПЛА.

У науковій роботі «UAV-YOLOv8: A Small-Object-Detection Model Based on Improved YOLOv8 for UAV Aerial Photography Scenarios» було проаналізовано застосування YOLOv8 для обробки зображень з дронів і доведено ефективність моделі в завданнях з виявлення дрібних об’єктів [14]. Схожі дослідження присвячені покращенню точності YOLOv8-n у складних умовах освітлення та на фонових сценах з високою зашумленістю. Дослідники також активно розробляють інші архітектури: RetinaNet, EfficientDet, CenterNet, які мають свої переваги в точності, однак поступаються YOLOv8 у швидкодії. YOLO залишається найбільш збалансованим варіантом для вбудованих систем [15].

Таким чином, згорткові нейронні мережі завдяки своїй структурі є природним вибором для задач розпізнавання об’єктів на відео, особливо в умовах складної сцени, характерної для зон масових руйнувань. Оптимізовані архітектури, такі як YOLO, забезпечують ефективний баланс між точністю та швидкодією, що є критичним для оперативного використання результатів під час рятувальних операцій.

1.5 Постановка задачі

У сучасних умовах цифровізації всіх сфер суспільства розширюється сфера застосування БПЛА, що обумовлює розвиток та впровадження програмних засобів для автоматичного виявлення об’єктів при обробці даних, які надходять з БПЛА. Розробка даного проєкту спрямована на створення програмного засобу, здатного виявляти у зоні руйнувань після бойових дій людей і автотранспорт на зображеннях, відео та потокових відео – стрімах, у режимі реального часу при обмежених обчислювальних ресурсах та складних умовах зйомки (денний і нічний час, дим, пил, завали).

Об’єкт роботи – процес розпізнавання об’єктів при обробці даних, які надходять з БПЛА.

Предмет роботи – нейромережеві моделі та програмні засоби розпізнавання об’єктів у місцях масових руйнувань на зображеннях, відео та стрімах БПЛА.

Мета роботи полягає у підвищенні точності розпізнавання об’єктів у зонах масових руйнувань після обстрілів шляхом розробки інформаційної системи на основі згорткових нейронних мереж Ultralytics YOLO.

Для розробки інформаційної системи необхідно вирішити такі завдання:

– дослідити теоретичні засади розпізнавання об’єктів для БПЛА з використанням згорткових нейронних мереж, здійснити огляд сучасних досліджень і програмних рішень цій сфері;

– провести аналіз моделей YOLO платформи Ultralytics та обґрунтувати вибір моделі YOLOv8 для обробки даних з БПЛА;

– здійснити навчання моделей, здатних розпізнавати об’єкти у зонах масових руйнувань після обстрілів на зображеннях, відео і стрімах, та оцінити їх якість;

– обґрунтування вибір стеку технологій та розробити інформаційну систему, яка використовує створені моделі для ідентифікації об’єктів при обробці даних з дронів.

Обраний підхід передбачає використання моделей нейронної мережі YOLOv8 платформи Ultralytics, що дозволяє досягти балансу між точністю виявлення об’єктів та швидкістю обробки даних. Інформаційна система повинна бути реалізована у вигляді вебзастосунку з автоматизованим розпізнаванням об’єктів із використанням навчених моделей для оперативного інформування рятувальників у режимі реального часу.

Вебзастосунок для розпізнавання об’єктів повинен мати наступну функціональність:

– бути простим у використанні у реальних рятувальних операціях та швидким у ідентифікації об’єктів на зображеннях, відео та стрімах, які надходять від БПЛА;

– мати зрозумілий інтерфейс для оперативного перегляду результатів із виявленими об’єктами;

– забезпечувати легкість інтеграції різних моделей та незалежність системи від оптичного обладнання.

Розроблена інформаційна система буде функціонувати як у повністю автономному режимі (через протоколи Wi-Fi/5G), не потребуючи серверної архітектури, так і в умовах використання потокових протоколів (RTSP, HTTP), за яких можлива інтеграція з серверними рішеннями при необхідності віддаленої обробки або передачі даних через Інтернет.

Висновки до розділу 1

У першому розділі було здійснено дослідження предметної області, що стосується розпізнавання об’єктів у місцях масових руйнувань від обстрілів, з даних, які надходять від БПЛА. Виявлено, що ефективне використання БПЛА у рятувальних операціях потребує надійного програмного супроводу. Здійснений аналіз можливостей існуючих програмних рішень для обробки даних від БПЛА показав, що вини мають обмеження, які не забезпечують ефективність їх

застосування при виконанні пошуково-рятувальних операцій у зонах масових руйнувань. Більшість із них є комерційними і не безкоштовними. В основному вони зосереджені на картографуванні, а не на оперативному виявленні окремих об'єктів. До того ж потребують потужного серверного обладнання для обробки даних та мають обмежену адаптивність до особливих умов зйомки. Що обумовлює необхідність розробок, адаптованих до ситуацій, у яких дрони виконують пошуково-рятувальні операції.

Здійснений аналіз досліджень у сфері розпізнавання дозволив установити, що найбільш оптимальними моделями, які інтегрують у програмні засоби для розпізнавання об’єктів, є згорткові нейронні мережі. Виявлено, що найбільш прийнятними для розпізнавання об’єктів є моделі YOLO, сучасні версії яких можуть здійснювати виявлення об'єктів, їх сегментацію і класифікацію як на зображеннях так і на відео та потокових відео. Моделі останніх версій Ultralytics YOLO мають оптимізовані архітектури, забезпечуючи баланс між точністю та швидкодією, що є критичним під час проведення рятувальних операцій. Тому у розробці інформаційної системи було вирішено використовувати моделі YOLO платформи Ultralytics.

Спираючись на виявлені аспекти, було здійснено постановку задачі та сформульовано основні вимоги до розроблюваної системи розпізнавання об’єктів на основі обробки даних, які надходять від БПЛА.

2 МОДЕЛІ ULTRALYTICS YOLO

2.1 Еволюція моделей YOLO

Платформа Ultralytics, що спеціалізується на відкритих реалізаціях моделей YOLO [16], відіграє ключову роль у розвитку та поширенні цієї технології. YOLO – архітектура комп’ютерного зору, яка має велику кількістю версій: перша вийшла в 2016 році і вирішувала лише завдання детекції об'єктів на зображенні, а остання – одинадцята, з’явилася у вересні 2024 року і вже є фундаментальною моделлю, яку можна використовувати для класифікації, трекінгу у реальному часі [17, 18].

Детекція – це один із підвидів проблеми знаходження об’єктів на зображенні. Якщо потрібно визначити наявність або відсутність об’єкта певного домену на зображенні – це класифікація (англ. classification). Якщо потрібно виконати класифікацію, і до того ж визначити рамку, що обмежує розташування екземпляра одиночного об'єкта на зображенні, це класифікація з локалізацією (англ. classification and localization). Якщо потрібно для кожного пікселя на зображенні визначити його приналежність до певної категорії – це семантична сегментація (англ. semantic segmentation). Якщо потрібно виконати сегментацію, але при цьому диференціювати лише об'єкти певної сутності – це сегментація екземплярів (англ. instance segmentation) (рис. 2.1).

Детекцією (англ. object detection) ж традиційно називають завдання, у якій необхідно виділити кілька об'єктів на зображенні за допомогою знаходження координат їх рамок, що обмежують, і класифікації цих обмежувальних рамок з безлічі заздалегідь відомих класів. При цьому, на відміну завдання класифікації з локалізацією, кількість об'єктів на зображенні заздалегідь невідома.

Рамки, в які потрібно укладати об’єкти, називаються bounding boxes або просто b-boxes. Б-бокси традиційно мають прямокутну форму і розташовуються так, щоб сторони прямокутника були паралельні рамкам зображення. Також інтуїтивно зрозуміло, що рамка має бути "мінімальної", тобто захоплювати об'єкт

повністю так, щоб при цьому мати мінімальну площу. Те, як задаємо б-бокс – залежить вже від архітектури, але зазвичай вибирають один із двох варіантів: координати центру (x_0, y_0) + ширина (l) і висота (h), або координати верхнього лівого пікселя (x_1, y_1) + координати нижнього правого (x_2, y_2).

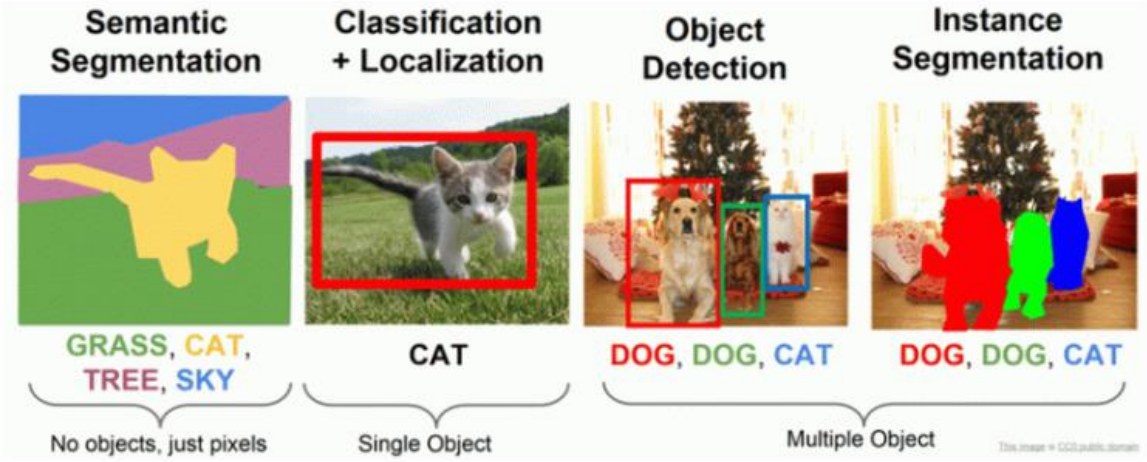

Рисунок 2.1 – Задачі, які вирішує YOLO

У задачі детекції модель повинна вміти вирішувати відразу дві проблеми: пошук оптимальних б-боксів та класифікацію. Тут найнаївніший підхід – перебрати всілякі б-бокси і прогнати кожен з них через класифікатор на основі згорток. Але всім відомо, що майже будь-який повний перебір – це неможливий довгий та неефективний процес. Тож у реальних архітектурах його трохи докручують. Наприклад, є група методів, званих двоетапними, які на першому кроці відбирають лише деякі б-бокси, які з високою ймовірністю містять об’єкт, а на другому такі обрані рамки згодовують класифікатору. До таких методів відноситься R-CNN та його модифікації Fast R-CNN та Faster R-CNN.

YOLO стала першим представником іншої групи методів – одноетапних алгоритмів, в яких окрема модель для відбору регіонів взагалі не використовується. Натомість YOLO являє собою єдину мережу, яка відразу передбачає координати деякої кількості б-боксів разом з їх характеристиками, такими, як ймовірність класу.

Головне нововведення YOLO: її творці змогли сформулювати та вирішувати задачу детекції як задачу регресії. Вихідний тензор мережі для кожної комірки зображення передбачає вектор, усередині якого ховається опис б-боксів та міток класів (рис. 2.2).

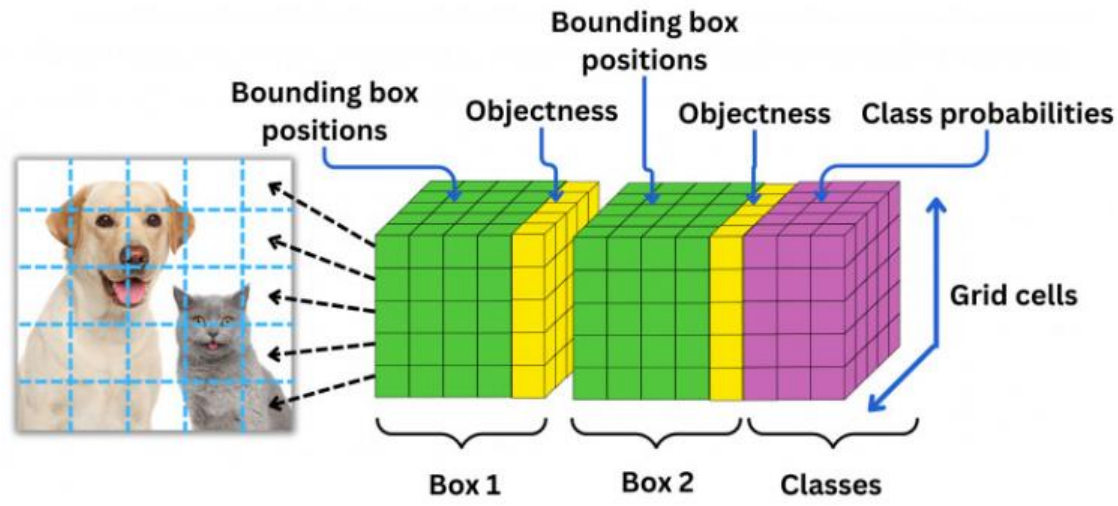

Рисунок 2.2 – Вихідний тензор YOLO [19]

Функція втрат YOLO являє собою складне склеювання класичних loss класифікації та детекції (рис. 2.3).

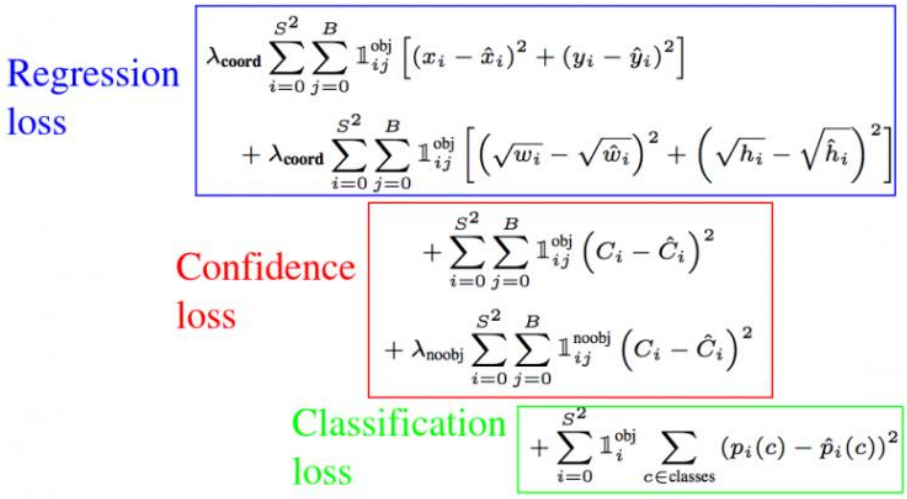

Рисунок 2.3 – Комбінація функцій loss YOLOv1 [20]

Помилка в передбаченні координат центру (x, y), висоти (h) і ширини (w) б-боксів – це є regression loss. Індекс і відповідає за об'єкти, індекс j – за бокси. це все

той же об'єкт і, але бокс інший, вираз обнулиться, щоб двічі не враховувати одну і ту ж помилку. Корені, які застосовуються до висоти та ширини б-боксів, це міра масштабування, необхідна для того, щоб більше штрафувати маленькі б-бокси за невідповідність реальній розмітці.

Остання частина – classification loss, це класичний квадратичний loss, в якому розраховують помилки на мітках класів. Індикаторна функція гарантує, що будуть враховані тільки ті комірки, у яких дійсно є об’єкт. Комірки без об’єктів враховують, однак коміркам, що містять щось, необхідно віддавати більшу вагу.

Нарешті, Confidence loss – це та частина функції втрат, яка відповідає за оцінки «впевненості» моделі в тому, що всередині б-боксу знаходиться центр об’єкта (вони позначені жовтим на попередній схемі). Це передбачене значення функції IoU (англ. Intersection over Union), яка є однією з ключових метрик комп’ютерного зору. У термінах б-боксів IoU – це перекриття між прогнозованою рамкою та справжнім прямокутником із трейну (рис. 2.4).

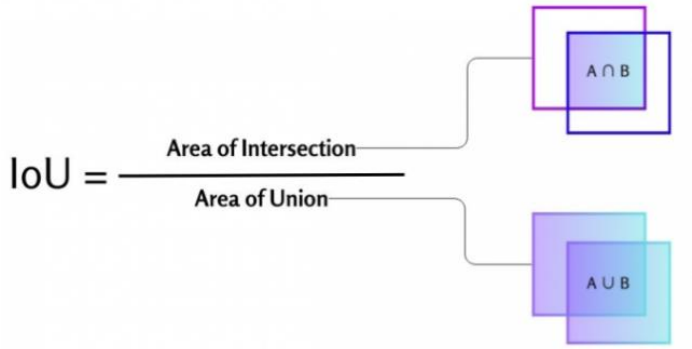

Рисунок 2.4 – Метрика IoU

Якщо модель передбачає високу IoU, то це свідчить про те, що перекриття буде більшим, модель сильніше впевнена в існуванні певного б-боксу. І навпаки, якщо передбачає низьке IoU, то модель впевнена, що в осередку об’єкта взагалі немає.

У наступних версіях було введено поняття якорів – anchor boxes, які містить вихідний тензор. Їх розраховують, кластеризуючи б-бокси за допомогою алгоритму

k-means, визначаючи, при якому значенні k точність виходить плато. При цьому уже задаються не координати центра, ширини та висоти б-боксу, а набір чисел, які відповідають за зміщення якоря вліво/вправо, вгору/вниз, зміну ширини та висоти відповідно.

YOLOv8 – найширше використовувана модель лінійки, яка була розроблена на початку 2023 року Ultralytics і відноситься до новітніх поколінь алгоритмів виявлення об’єктів у зображеннях, яке об’єднує в собі переваги глибокого навчання, ефективної архітектури комп’ютерного зору та практичного застосування в реальному часі. Розроблена у 2023 році компанією Ultralytics, ця модель стала логічним продовженням еволюції YOLO-сімейства та реалізувала низку принципово нових рішень, які забезпечують як зростання точності, так і спрощення процесу навчання та розгортання.

Модель YOLOv8 є популярною завдяки абсолютно новому репозитарію, побудованому як єдина платформа для навчання моделей детекції, сегментації та класифікації. Компанія випустила п’ять моделей, кожна з яких може працювати з усіма цими завданнями. Серед них YOLOv8 Nano і найбільша і найточніша YOLOv8 Extra Large (YOLOv8x). Усі моделі підтримують багато форматів експорту та можуть працювати і на CPU, і на GPU. А ще YOLOv8 – це найзручніше API, яке сумісне і з командним рядком, і з Python.

В архітектурі YOLOv8 відбулися суттєві зміни у порівнянні з попередніми версіями, зокрема YOLOv5 і YOLOv7: нова Backbone мережа, функція втрат та Anchor-Free head. Однією з ключових змін є відмова від anchor-based підходу, що використовувався у YOLOv5 та всіх попередніх версіях для визначення положення об’єктів за допомогою фіксованих шаблонів – якорів і вимагав ручного налаштування під конкретний dataset. У YOLOv8 реалізовано Anchor-Free підхід, при якому координати об’єктів передбачаються без прив’язки до фіксованих розмірів. Це спрощує процес адаптації моделі до нових умов і дозволяє краще працювати з нетиповими або складними сценами, такими як зйомка з дронів.

Для того щоб модель вміла виконувати різні види завдань, її навчали в кілька етапів на різних датасетах. Контрольні точки детекції навчені на основі COCO detection з роздільною здатністю 640 [21]. Чекпоінти сегментації – датасети COCO сегментації з тією ж роздільною здатністю. А для класифікації датасет не змінювався з першої версії, так і залишився ImageNet. У восьмій версії шари класифікації навчають на розмірі 224.

У YOLOv8 перетворення зображення відбувається через блоки типу C2f (Cross-Stage Partial connections with two convolution layers and feature fusion), що дозволяє зберігати більшу кількість важливих ознак на різних рівнях глибини мережі [22]. Архітектура поділяється на три ключові частини:

– Backbone: для вилучення ознак зі зображення;

– Neck: для об'єднання ознак з різних рівнів абстракції (SPPF, PANet):

– Head: для формування остаточних передбачень щодо координат рамок і класів об’єктів.

Особливістю YOLOv8 є використання розділеного головного блоку (Decoupled Head), де локалізація (визначення координат об’єкта) і класифікація (визначення типу об’єкта: людина, автомобіль) виконуються окремими гілками мережі. Це дозволяє покращити якість обох процесів і мінімізувати взаємний вплив задач локалізації та класифікації.

Починаючи з перших версій YOLO, розробники прагнули досягти обробки відеопотоку в реальному часі при збереженні достатньої точності (рис. 2.5). Згодом, із появою нових поколінь моделей, таких як YOLOv5, YOLOv7 та YOLOv8, значно покращилася якість передбачень без істотної втрати продуктивності. Серед останніх досягнень варто відзначити YOLOv8 — модель, реалізовану на фреймворку PyTorch [23], що відзначається гнучкістю, високою продуктивністю та можливістю глибокої адаптації під різноманітні завдання.

Платформа Ultralytics пропонує розширені можливості для розробників: автоматичне навчання моделей, візуалізацію результатів, підтримку аугментацій [24], а також експорт у формати ONNX, TensorRT, CoreML і TorchScript [25].

Завдяки цьому YOLOv8 може використовуватися не лише на потужних серверах, а й на edge-пристроях, таких як мобільні БПЛА або переносні польові станції рятувальників.

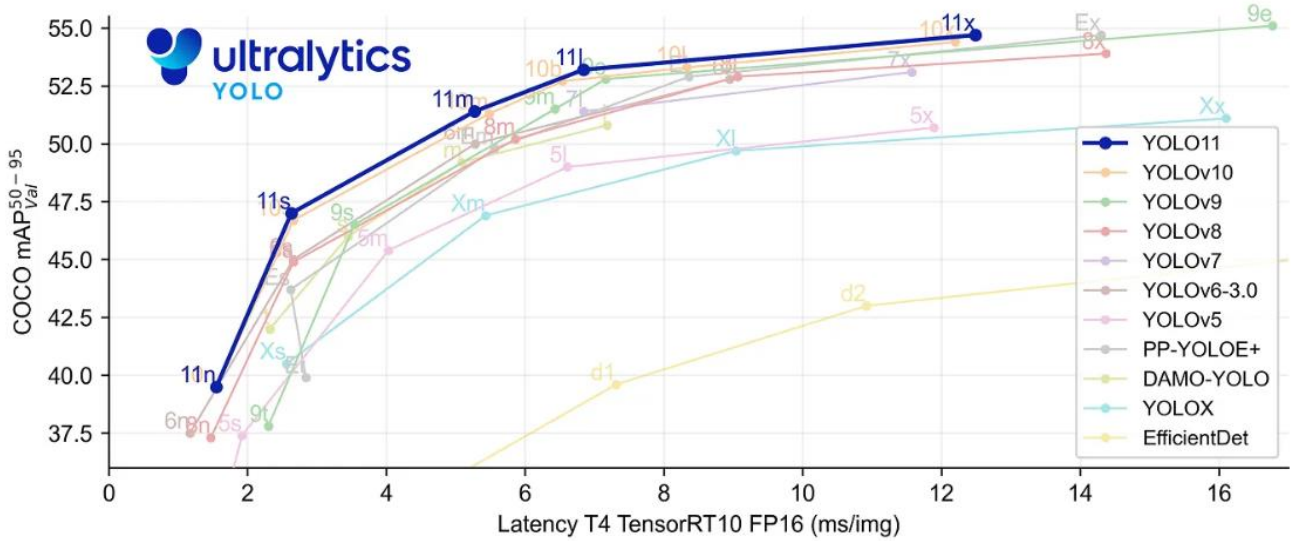

Рисунок 2.5 - Бенчмаркінг моделей YOLO на наборі даних COCO [17]

Особливо важливо, що Ultralytics підтримує різні задачі комп'ютерного зору: окрім детекції об'єктів, також класифікацію, сегментацію, оцінку пози (pose estimation) та трекінг. Така багатофункціональність робить платформу ідеальним вибором для комплексних систем, де необхідна інтеграція різних типів аналізу даних із зображень.

Отже, платформа Ultralytics із моделями YOLOv8 відкриває широкі можливості для побудови інтелектуальних систем розпізнавання об'єктів на основі відеоданих з БПЛА, що є надзвичайно актуальним для завдань оперативного реагування на наслідки бойових дій. Розробка власної системи виявлення людей і транспортних засобів на основі YOLOv8 дозволяє створити адаптивне, автономне та високошвидкісне рішення, орієнтоване саме на потреби рятувальних служб. Система не залежить від підключення до хмарних сервісів, дозволяє обробляти дані в реальному часі й здатна працювати в умовах обмежених ресурсів польового обладнання.

2.2 Архітектура моделі YOLOv8 та метрики оцінки її якості

Архітектура YOLOv8 є втіленням ідеї ефективного поєднання глибоких згорткових мереж із мінімізацією обчислювальної складності та високою точністю передбачень. Побудова моделі базується на чіткій структурі з трьох основних компонентів: *Backbone* – для вилучення ознак, *Neck* – для агрегації ознак і *Head* – для формування прогнозу (рис. 2.6) [26].

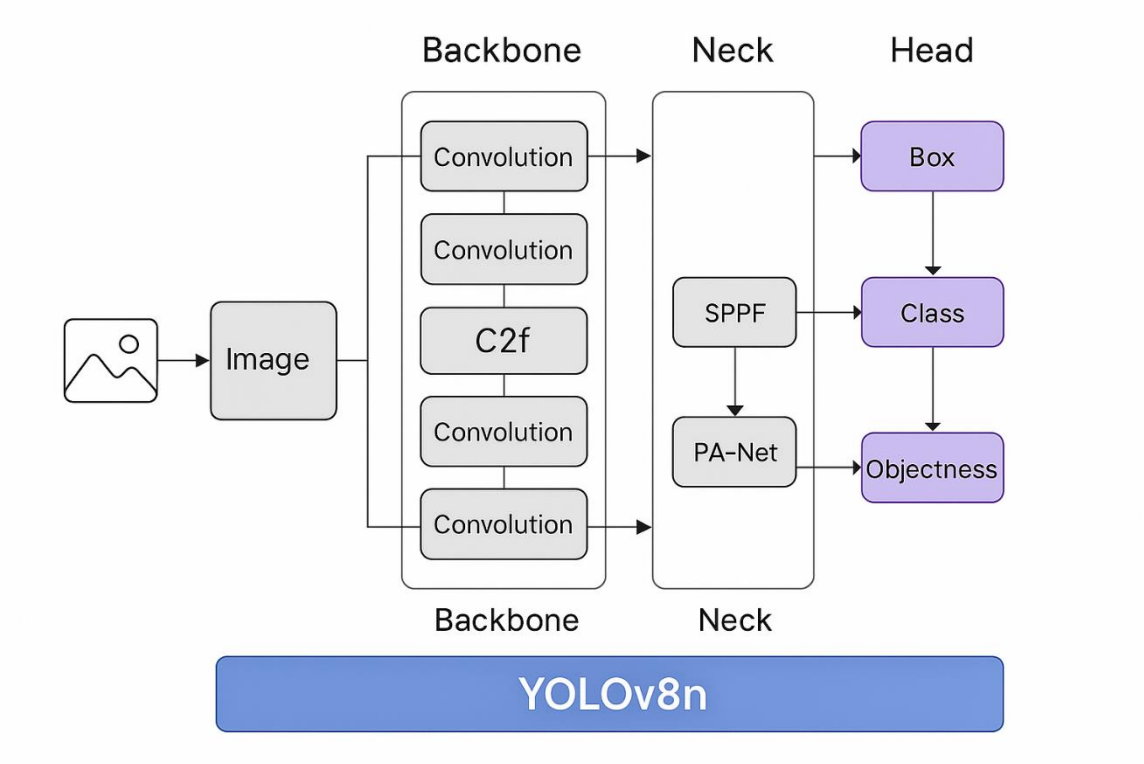

Рисунок 2.6 – Основні компоненти моделі YOLOv8 [19]

Backbone виконує функцію вилучення базових ознак із вхідного зображення. У YOLOv8 використовується удосконалена схема шарів типу C2f (C2f (Cross-Stage Partial Fusion), що сприяє ефективному збереженню ключових ознак та зменшенню дублювання обчислень. C2f прийшли на зміну попереднім CSP-блокам та забезпечують більш ефективне перетікання інформації між шарами, зменшують обчислювальну складність і покращують узагальнюючі властивості мережі.

Завдяки цьому мережа виділяє як локальні деталі (текстури, контури), так і глобальні об’єкти.

Блок *Neck* призначений для об'єднання ознак із різних рівнів глибини та інтегрування контекстної інформації. Для цього використовується комбінація методів та PAN (англ. Path Aggregation Network), який об’єднує ознаки з різних рівнів, та SPPF (англ. Spatial Pyramid Pooling Fast), який забезпечує більш широке поле зору. Таким чином у YOLOv8 реалізовано піраміду ознак, яка забезпечує високу точність детектування. Такий підхід дозволяє побудувати збагачену репрезентацію об’єкта перед остаточним передбаченням.

Head є вихідним модулем, який формує остаточні результати: генерує координати рамки об'єкта (x, y, ширина, висота), ступінь об’єктності (англ. objectness score) та вірогідності класів. В YOLOv8 реалізовано концепцію Decoupled Head — тобто розділення потоків для задач локалізації та класифікації, що дозволяє уникнути конфлікту між цими процесами та підвищити точність.

Модель YOLOv8 має модульну та ефективну архітектуру, яка дозволяє її адаптувати під різні задачі – зокрема, виявлення об’єктів на відео з дронів у реальному часі. У більш деталізовано вигляді архітектура моделі представлена на рисунку 2.7. Перші кілька шарів моделі (Conv2d, BatchNorm2d, SiLU) формують базову згорткову мережу, яка витягує ознаки з вхідного зображення. Блок C2f – відповідає за збереження частини ознак з пропуском через обхідні зв’язки, що покращує навчання глибоких мереж. Далі йде блок SPPF, який узагальнює ознаки з різних масштабів. Завершальна частина містить блоки для обробки ознак, вбудований DFL (англ. Distribution Focal Loss [15]), який дозволяє формувати більш точні координати об'єктів. Внутрішні послідовні модулі Sequential(...) поєднують згорткові шари в одну обробну лінію.

Кожен шар має свої параметри: кількість каналів, розмір фільтрів (kernel_size), крок згортки (stride), наявність padding тощо. Наприклад, шар Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2)) означає, що вхідні 3 канали (RGB)

будуть перетворені в 16 каналів за допомогою фільтра 3×3 зі зменшенням розміру зображення вдвічі.

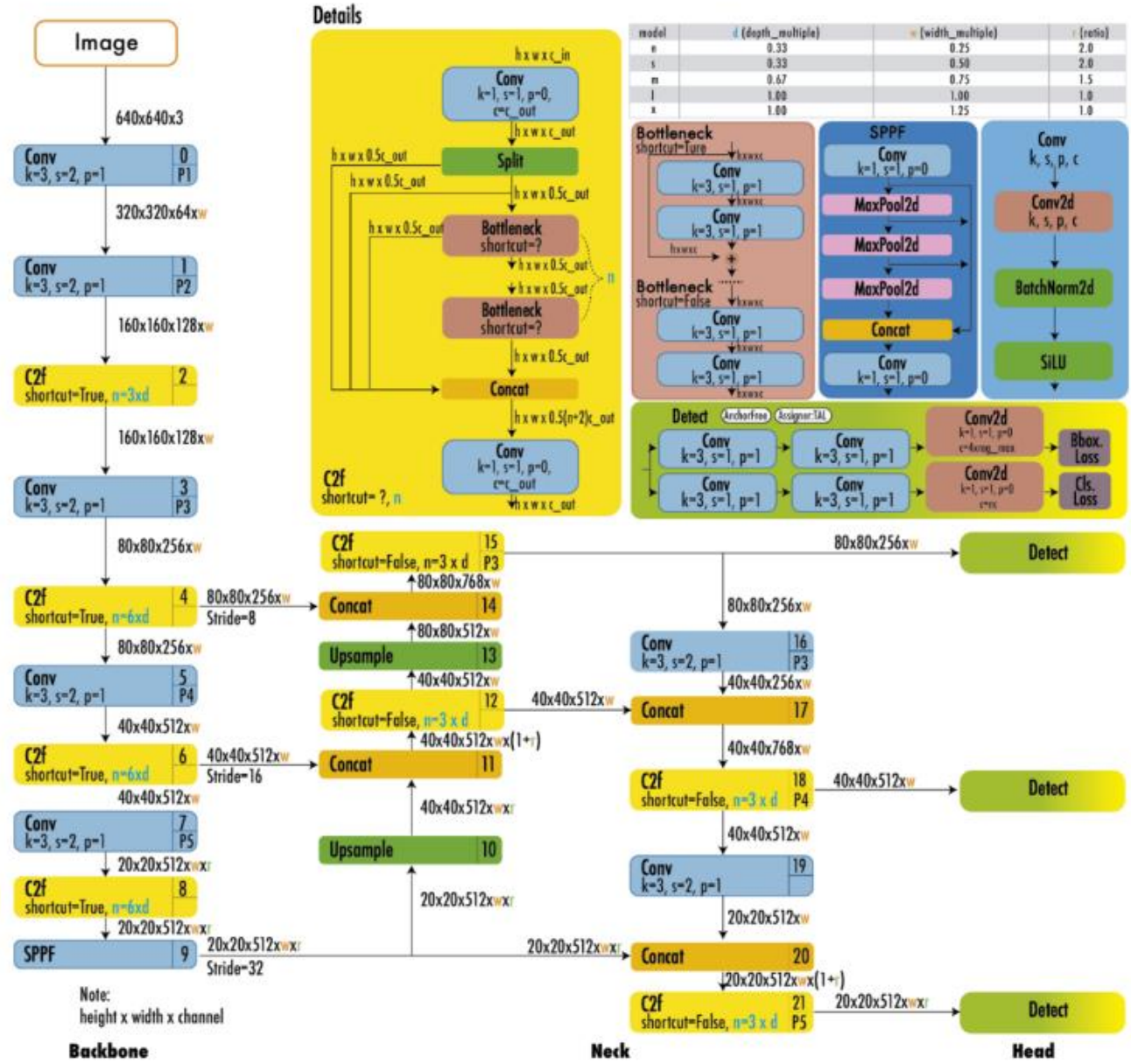

Рисунок 2.7 – Схема архітектури моделі YOLOv8 [27]

Блок C2f є вдосконаленою версією класичних CSP-блоків (англ. Cross-Stage Partial), які використовувались у YOLOv5. Основна його функція полягає у збереженні важливих ознак з глибших шарів без надмірного збільшення обчислювальної складності. Це досягається за рахунок часткового розгалуження ознак: частина ознак проходить через шари обробки, а частина — напряму

передається далі. Така структура дозволяє підвищити ефективність навчання та покращити узагальнення моделі, зберігаючи при цьому її компактність. Модель побудована як послідовність згорткових блоків з використанням C2f, SPPF та інших оптимізованих шарів. Це забезпечує компактність і високу швидкодію навіть при значній глибині мережі.

Графічна блок-схема моделі, яка представляє її роботу у спрощеному вигляді, представлена на рисунку 2.8. Concat – це операція об’єднання (конкатенації) кількох тензорів вздовж певної осі, зазвичай канальної. В архітектурі YOLOv8 вона дозволяє моделі поєднувати детальну локальну інформацію з високорівневим глобальним контекстом, що суттєво покращує розпізнавання об'єктів різного розміру на одному зображенні.

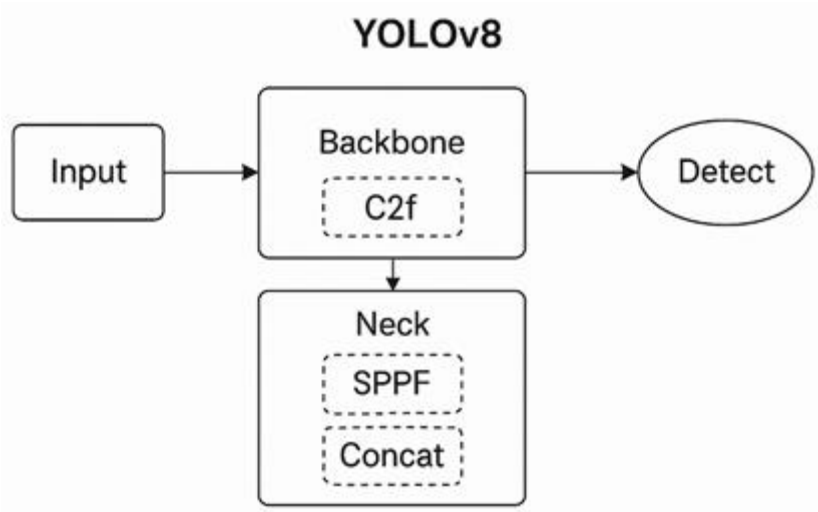

Рисунок 2.8 – Блок-схема моделі YOLOv8

Detect є фінальним блоком моделі YOLOv8, який генерує підсумкові передбачення. Він приймає злиті ознаки з Neck-блоків і обчислює координати об'єктів (bounding boxes), класову належність та рівень впевненості (confidence). Саме тут реалізується безякірна логіка передбачення: модель самостійно формує координати без попередньо заданих шаблонів. Вихід Detect-блоку подається до модуля Non-Maximum Suppression (NMS), який відкидає зайві передбачення та формує фінальний список розпізнаних об’єктів.

Особливої уваги заслуговує процес інференсу (англ. inference), тобто передбачення на нових зображеннях. При подачі зображення на вхід модель спочатку змінює його розмір до стандартного (640×640 пікселів), після чого обробляє його через згорткові шари для виділення просторових ознак. Потім здійснюється масштабна агрегація ознак, після чого модель видає набір bounding boxes із відповідними класами та ймовірностями. Застосування на фінальному етапі NMS дозволяє уникнути дублювання результатів шляхом фільтрації перекритих прямокутників.

Для оцінки якості роботи моделей виявлення об'єктів застосовуються ключові метрики [27, 28]:

– Intersection over Union (IoU) – оцінка перекриття між передбаченою та реальною рамкою;

– Precision і Recall – показники точності і повноти передбачень;

– F1 Score – баланс між Precision і Recall;

– Average Precision (AP) і Mean Average Precision (mAP) – інтегровані оцінки точності на різних порогах IoU.

IoU є основною метрикою для оцінки точності локалізації об'єктів. Вона вимірює ступінь перекриття між передбаченою рамкою та реальною рамкою об'єкта:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}, \tag{2.1}$$

де Area of Overlap – площа перетину передбаченої та реальної рамок;

Area of Union – площа об'єднання передбаченої та реальної рамок.

Значення IoU варіюється від 0 до 1, де 1 означає повне перекриття, а 0 — відсутність перекриття. У практиці, для визнання передбачення правильним, використовується порогове значення IoU, зазвичай 0.5.

У версії YOLOv8 застосовують вдосконалену метрику CIoU, яка враховує IoU (перекриття), відстань між центрами боксів та різницю у співвідношенні сторін:

$$\text{CIoU} = 1 - \text{IoU} + \frac{\rho^2(b,\hat{b})}{c^2} + \alpha v, \quad (2.2)$$

де $\rho(b, \hat{b})$ – евклідова відстань між центрами передбаченого та реального боксів;

c – діагональ мінімального прямокутника, який вміщує обидва бокси;

$v = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2$ – міра співвідношення сторін передбаченого та реального боків;

w і h – ширина та висота боксів; α – коефіцієнт балансу: $\alpha = \frac{v}{(1-IoU)+v}$.

Precision вимірює частку правильних передбачень серед усіх передбачених позитивних випадків. Вона показує, наскільки модель уникає хибнопозитивних передбачень:

$$\text{Precision} = \frac{TP}{TP+FP}, \quad (2.3)$$

де *TP* (англ. True Positives) – кількість правильних передбачень;

FP (англ. False Positives) – кількість хибнопозитивних передбачень.

Високе значення Precision свідчить про те, що модель рідко помилково ідентифікує об'єкти, яких немає на зображенні.

Recall вимірює частку правильно передбачених об'єктів серед усіх реальних об'єктів. Вона показує здатність моделі виявляти всі наявні об'єкти:

$$\text{Recall} = \frac{TP}{TP+FN}, \quad (2.4)$$

де *FN* (англ. False Negatives) — кількість хибнонегативних передбачень (реальні об'єкти, які модель не виявила).

Високе значення Recall свідчить про те, що модель рідко пропускає об'єкти на зображенні.

F1 Score є гармонічним середнім між Precision та Recall, забезпечуючи баланс між цими двома метриками:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \quad (2.5)$$

F1 Score особливо корисний у випадках, коли важливо одночасно мінімізувати як хибнопозитивні, так і хибнонегативні передбачення.

Average Precision (AP) об’єднує інформацію про Precision та Recall на різних порогах IoU, забезпечуючи загальну оцінку точності моделі для кожного класу об’єктів [27].

AP обчислюється як площа під кривою Precision-Recall для конкретного класу:

$$\mathrm{AP} = \int_0^1 \mathrm{Precision}(\mathrm{Recall})\, dRecall. \quad (2.6)$$

На практиці, ця інтеграція часто апроксимується дискретною сумою:

$$\mathrm{AP} = \sum_{\mathrm{n}} (R_n - R_{n-1}) \cdot P_n\,, \quad (2.7)$$

де P_n і R_n – значення Precision та Recall на n-му порозі CIoU.

Mean Average Precision є середнім значенням AP для всіх класів об'єктів, забезпечуючи загальну оцінку продуктивності моделі:

$$\mathrm{mAP} = \frac{1}{C} \cdot \sum_{i=1}^{C} APi, \quad (2.8)$$

де C — кількість класів;

AP_i — Average Precision для *i*-го класу.

У YOLOv8 використовуються два варіанти mAP:

– mAP@0.5: середнє значення AP при порозі $IoU \geq 0{,}5$;

– mAP@0.5:0.95: середнє значення AP при порогах IoU від 0,5 до 0,95 з кроком 0.05: $\text{mAP@0.5:0.95} = \frac{1}{10} \cdot \sum_{IoU=0{,}05}^{0{,}95} AP_{IoU}$.

Ці метрики дозволяють оцінити як точність локалізації об'єктів, так і точність класифікації – здатність моделі виявляти об'єкти різних розмірів та класів.

При застосуванні навчених моделей архітектури YOLOv8 виявлені об'єкти відображаються у рамці, поруч з якою виводиться показник Confidence Score – рівень впевненості моделі, що виявлений об'єкт належить до певного класу. Його розрахунок здійснюють за формулою:

$$\text{Confidence Score} = \text{Objectness Score} \cdot \text{Class Probability}, \quad (2.9)$$

де Objectness Score – ймовірність того, що в рамці є якийсь об'єкт, а не фон;

Class Probability – ймовірність того, що цей об'єкт належить до певного класу.

У процесі навчання YOLOv8 застосовується комбінована функція втрат *L*, яка складається з кількох компонент, які разом оцінюють якість передбачення моделі [20]:

$$L = \lambda_{box} \cdot L_{CIoU} + \lambda_{obj} \cdot L_{obj} + \lambda_{cls} \cdot L_{cls}, \quad (2.10)$$

де L_{CIoU} – bounding box regression loss: втрата локалізації;

L_{obj} – objectness loss: втрата об'єктності;

L_{cls} – classification loss: втрата класифікації;

λ_{box}, λ_{obj}, λ_{cls} – ваги кожної компоненти, зазвичай: $\lambda_{box} = 0{,}05$, $\lambda_{obj} = 1{,}0$, $\lambda_{cls} = 0{,}5$.

Втрата локалізації L_{CIoU} має відношення до передбачення обмежувальної рамки та показує, наскільки добре модель прогнозує розмір та розташування об’єктів:

$$L_{CIoU} = 1 - CIoU(b, \hat{b}). \quad (2.11)$$

Втрата об’єктності L_{obj} показує, наскільки модель впевнена у наявності об’єкта у рамці:

$$L_{obj} = BCE(y, \hat{p}) = -[y \log(\hat{p}) + (1 - y)\log(1 - \hat{p})], \quad (2.12)$$

де $\hat{p}$ – передбачена ймовірність об’єкта;

y – істинне значення (1 для об'єкта, 0 — фон).

Втрата класифікації L_{cls} для багатокласової класифікації використовує не функцію softmax, а Binary Cross-Entropy для кожного класу (multi-label):

$$L_{cls} = \sum_{c=1}^{C} BCE(y_c, \hat{p}_c), \quad (2.13)$$

де c – кількість класів;

$y_v \in \{0, 1\}$ – ground truth для кожного класу c;

$\hat{p}_c \in [0.1]$ – ймовірність передбачення для кожного класу c.

Класифікація здійснюється за кількома мітками. Модель навчається розпізнавати та призначати кілька міток або тегів зображенню. Замість призначення однієї категорії, як у випадку класифікації за кількома класами, класифікація за кількома мітками дозволяє одночасно мати на зображенні кілька об’єктів, атрибутів або концепцій. Цей тип класифікації особливо корисний при роботі зі складними сценами або зображеннями, де співіснує кілька об’єктів або атрибутів, забезпечуючи більш детальне та повне розуміння вмісту зображення.

Конфігурації моделей YOLOv8 включають версії n (nano), s (small), m (medium), l (large) та x (extra-large). Порівняльні характеристики та показники якості різних версій моделі наведено у таблиці 2.1. Слід зазначити, що наведені значення можуть змінюватися залежно від обладнання, середовища запуску та розміру вхідного зображення (тут орієнтовано на розмір 640x640 на NVIDIA T4 GPU). Різні модифікації дають змогу обрати оптимальний варіант під конкретні ресурси та вимоги задачі. Наприклад, модель yolov8n.pt із близько 3 млн параметрів є придатною для мобільних застосунків, тоді як yolov8s.pt забезпечує вищу точність, що може бути важливим для детального аналізу постраждалих зон.

Таблиця 2.1 – Характеристики версій моделі YOLOv8 [29]

Версія	**Параметри**	**Швидкість**	**mAp@0.5**	**mAp@0.5:0.95**	**Використання**
YOLO8n	~3,2 млн	~2,5 ms	~0,69	~0,44	CPU-пристрої, вбудовані системи
YOLO8s	~11,2 млн	~4,2 ms	~0,78	~0,55	Застосування в реальному часі
YOLO8m	~25,9 млн	~8,7 ms	~0,83	~0,61	Універсальні системи з GPU
YOLO 8l	~43,7 млн	~12,1 ms	~0,86	~0,65	Висока точність, для серверів
YOLO x	~68,2 млн	~16,1 ms	~0,88	~0,68	Розгортання у кластерних середовищах

Окрім архітектурних переваг, YOLOv8 має відмінну підтримку експорту моделей у формати, придатні для мобільних і вбудованих рішень. Наприклад, за

допомогою бібліотеки Ultralytics можна експортувати модель у формат ONNX для подальшої оптимізації через TensorRT, або у CoreML, який дозволяє запускати модель на пристроях Apple. Завдяки цьому YOLOv8 стає не лише потужною дослідницькою платформою, а й практичним інструментом для розгортання в реальних продуктах.

Підсумовуючи, YOLOv8 є універсальним, гнучким і високоточним інструментом для виявлення об’єктів, що поєднує у собі новітні дослідження в області глибокого навчання та практичні інженерні рішення. Саме тому вона була обрана як основа для реалізації системи виявлення об’єктів у даному дослідженні. YOLOv8 забезпечує високу швидкість обробки і точність виявлення об'єктів навіть у складних умовах, що робить її придатною для задач моніторингу зон бойових руйнувань із використанням БПЛА. Для розробки інформаційної системи розпізнавання об’єктів при обробці даних БПЛА доцільно обрати моделі YOLOv8n та YOLOv8s із врахуванням забезпечення оптимального балансу між вимогами до потужності комп’ютера та точності розпізнавання об’єктів.

Висновки до розділу 2

У другому розділі описано еволюцію нейромережевих моделей YOLO, які можна використовувати для розпізнавання об’єктів та трекінгу у реальному часі. Виявлено, що детекція об’єктів базується на пошуку оптимальних б-боксів та класифікації. Б-бокси мають прямокутну форму зі сторонами, паралельними рамкам зображення. YOLO реалізує одноетапний алгоритм, який передбачає координати рамок разом із ймовірностями класів, до яких віднесено об’єкти у рамках. Задача детекції вирішується як задача регресії, що обумовлює специфіку функції втрат, яка є комбінацією функцій втрат класифікації та детекції.

Для розробки інформаційної системи було обрано модель YOLOv8, у якій перетворення зображення відбувається через блоки C2f, зберігати більшу кількість важливих ознак на різних рівнях глибини мережі. В архітектурі моделі виділяють

складові частини: Backbone: для вилучення ознак зі зображення, Neck: для об'єднання ознак з різних рівнів абстракції, Head: для формування остаточних передбачень координат рамок і класів об'єктів. Особливістю YOLOv8 є використання розділеного головного блоку (Decoupled Head), де локалізація (визначення координат об’єкта) і класифікація (визначення типу об’єкта: людина, автомобіль) виконуються окремими гілками мережі. Це дозволяє покращити якість обох процесів і мінімізувати взаємний вплив задач локалізації та класифікації. Виявлено, що для розробки інформаційної системи розпізнавання об’єктів при обробці даних БПЛА доцільно обрати моделі YOLOv8n та YOLOv8s.

Виявлено ключові метрики, які застосовують для оцінки якості роботи моделі по розпізнаванню об’єктів: Intersection over Union (IoU) – оцінка перекриття між передбаченою та реальною рамкою; Precision і Recall – показники точності і повноти передбачень; F1 Score – баланс між Precision і Recall; Average Precision (AP) і Mean Average Precision (mAP) – інтегровані оцінки точності на різних порогах IoU. Ключовою метрикою оцінки точності є IoU, яка у термінах б-боксів є перекриттям між прогнозованою рамкою та справжнім прямокутником із трейну.

3 НАВЧАННЯ ТА ОЦІНКА МОДЕЛЕЙ YOLOV8

3.1 Підготовка наборів даних та моделей до навчання

Для навчання моделей було використано PyTorch – фреймворк з відкритим вихідним кодом, що базується на мові програмування Python і бібліотеці Torch [30]. Такий вибір було обумовлено тим, що PyTorch підтримує створення складних архітектур нейронних мереж, до яких відносяться моделі YOLOv8. До того ж YOLOv8 використовує Ultralytics API, яка вже обробляє дані на базі PyTorch [31]. У репозиторії платформи Ultralytics уже є готові моделі, навчені на великому датасеті COCO, що містить понад 80 категорій об’єктів, зокрема автомобілі, пішоходи, автобуси, мотоцикли та інші об'єкти повсякденного середовища [32].

На рисунку 3.1 наведено фрагмент коду для встановлення бібліотеки Ultralytics та завантаження попередньо навченої моделі YOLOv8n. Аналогічно здійснюють завантаження моделі YOLOv8s.

```
from ultralytics import YOLO

# Load a COCO-pretrained YOLOv8n model
model = YOLO("yolov8n.pt")

# Display model information (optional)
model.info()
```

Рисунок 3.1 – Налаштування середовища для навчання моделей

Підготовка набору даних для навчання моделей YOLOv8 включає кілька етапів. Опишемо їх детально.

1. Набір даних повинен бути в форматі, який підтримує YOLOv8, з анотаціями у форматі YOLO TXT. На рисунку 3.1 наведено приклад структури каталогу, у якому розміщують файли набору даних із зображеннями та анотаціями.

```
dataset/
├── images/
│   ├── img1.jpg
│   ├── img2.jpg
│   └── ...
├── labels/
│   ├── img1.txt
│   ├── img2.txt
│   └── ...
```

Рисунок 3.2 – Структура каталогу для набору даних із зображеннями

2. Формат анотацій (YOLO TXT). Файл .txt для кожного зображення містить анотацію в YOLO-форматі, де кожен рядок представляє окремий об'єкт і містить п’ять числових значень, розділених пробілами [33]:

<class_id> <x_center> <y_center> <width> <height>.

Анотації у форматі YOLO мають строгий формат. У файлі анотацій усі координати анотовані від 0 до 1 та задаються у відносних одиницях до ширини та висоти зображення (рис. 3.3). Наприклад, значення 0 0.5 0.5 0.2 0.3 означає, що на зображенні є об’єкт класу 0, центр якого розташований у середині зображення, а ширина та висота bounding box становлять 20% та 30% відповідно від загального розміру зображення.

```
0 0.45 0.50 0.20 0.30  # клас 0, об'єкт по центру (45%, 50%), розмір 20x30%
```

Рисунок 3.3 – Приклад анотації зображення у файлі .txt

3. YAML-файл конфігурації. Для опису набору даних YOLOv8 потребує .yaml-файл [34], у якому вказано шлях до уже анотованого в YOLO-форматі набору даних, кількість класів та їх назви.

4. Підготовка з відео. Файли з відео потребують розділення на кадри. Після чого кожен кадр анотується як окреме зображення. На рисунку 3.4 наведено приклад коду для виконання цієї процедури.

```
import cv2
import os

video_path = 'video.mp4'
output_dir = 'dataset/images/train'
os.makedirs(output_dir, exist_ok=True)

cap = cv2.VideoCapture(video_path)
frame = 0

while cap.isOpened():
    ret, img = cap.read()
    if not ret:
        break
    cv2.imwrite(f"{output_dir}/frame_{frame:05d}.jpg", img)
    frame += 1

cap.release()
```

Рисунок 3.4 – Розбиття відео на кадри

5. Анотування. Для анотування можна використовувати: LabelImg (ручне маркування bbox), Roboflow (зручна веб-платформа, дозволяє експортувати в YOLO формат), CVAT [35-37].

6. Підготовлені набори даних завантажуються в PyTorch через Ultralytics.

Під час підготовки наборів даних було дотримано наступних рекомендацій:

– кількість даних: мінімум ~200 зображень на клас;

– баланс: всі класи були рівномірно представлені – збалансовані;

– формат та розмір зображень: для сумісності розмір зображень повинен бути однаковим, а самі зображенні мають бути у форматах .jpg, .png або .jpeg;

– до набору даних бажано додати приблизно 1% не анотованих зображень, що дозволяє зменшити кількість хибнопозитивних виявлень.

У даному проєкті передбачено розробку інформаційної системи, яка буде розпізнавати у місцях масових руйнувань п’ять класів об’єктів – чотири класи автотранспортних засобів та клас людей. Можливими є два підходи до використання моделей YOLOv8 у розпізнаванні об’єктів:

– використання попередньо навчених моделей YOLOv8n і YOLOv8s на наборі даних COCO (англ. Common Objects in Context), із тестуванням моделей на новому наборі даних, що містить об’єкти із місць масових руйнувань після обстрілів [25];

– використання моделей YOLOv8n і YOLOv8s, навчених на новому наборі даних, що містить об’єкти із місць масових руйнувань після обстрілів.

При реалізації як першого, так і другого підходу потрібно підготувати новий набір даних, підготовлений та розмічений відповідно до формату YOLOv8. У першому підході цей набір даних буде використовуватися для тестування перенавчених для розпізнавання 5 об’єктів моделей YOLOv8n і YOLOv8s. У другому випадку новий набір даних використовують як для навчання, так і для тестування моделей.

При підготовці нового набору даних зображення та відеокадри з місць масових руйнувань були анотовані вручну з використанням інструментів Roboflow та Annotate, які забезпечують зручний інтерфейс для ручного маркування [38]. Хоча YOLOv8 підтримує автоматичну аугментацію (flip, scale, mosaic) [39], у даній роботі вона не використовувалась. Це дозволяє зберегти вихідні характеристики набору даних, хоч і потенційно обмежує узагальнюючу здатність моделі.

Створений набір даних із зображеннями людей та транспортних засобів у вигляді файлів зображень та розкадрованих файлів відео містить 2500 екземплярів розміром 640x640 пікселів, які належать п’яти класам: car, truck, bus, motorcycle, person. На рисунку 3.5 наведено приклади зображень нового набору даних, який є збалансованим з мінімальними варіаціями у кількості екземплярів класу.

Рисунок 3.5 – Приклади зображень набору даних із місць масових руйнувань

Для покращення точності розпізнавання до набору даних було додано файли фонових зображень без анотацій (рис. 3.6). Це допомагає моделі правильно розпізнавати зображення без об’єктів, які необхідно розпізнавати.

Рисунок 3.6 – Приклади зображень без анотацій

Підготовлений на анотований у форматі YOLO набір даних було використано для навчання моделей.

При реалізації *першого підходу* завантаженні моделі YOLOv8n і YOLOv8s попередньо навчені розпізнавати 80 класів, і не є адаптованими до розпізнавання тільки п’яти класів. Для того, щоб модель адекватно працювала з п’ятьма класами, її потрібно перенавчити для розпізнавання вказаних класів. Після перенавчання буде отримана модель, яку можна тестувати на новому наборі даних з місць масових руйнувань. При цьому файл конфігурації.yaml перенавченої моделі повинен містити кількість класів, їх мітки та шляхи до навчальної та валідаційної вибірок (рис. 3.7). Після перенавчання буде отримана модель, яку можна тестувати

на новому наборі даних, який містить зображення з місць масових руйнувань, оцінивши якість моделей та точність розпізнавання об’єктів.

```yaml
# Назви класів
names:
  0: car
  1: truck
  2: bus
  3: motorcycle
  4: person
# Шляхи до датасету
train: /path/train/images
val: /path/val/images
# Кількість класів
nc: 5
```

Рисунок 3.5 –YAML-файл конфігурації при перенавчанні моделі YOLO

Перенавчену та протестовану на новому наборі даних модель далі можна інтегрувати у застосунок та використовувати для розпізнавання об’єктів вказаних класів.

Для реалізації *другого підходу*, який передбачає навчання моделей YOLOv8n і YOLOv8s з нуля на власному новому наборі даних, уже сформований для реалізації першого підходу набір даних потребує розділення на три частини: навчальну та валідаційну – для навчання моделі, та тестову – для тестування та оцінки якості моделі. У цьому підході файли нового підготовленого набору даних було розділено на 3 папки (рис. 3.6): train (навчальна), val (валідаційна) та test (тестова).

Кожна папка містить ще 2 папки: images (зображення) та labels (мітки) - папка з текстовими файлами анотацій, що містять мітки об'єктів на цих зображеннях. Коренева папка містить три основні підкаталоги: train, val та test. Папка train містить дані, на яких безпосередньо виконується навчання нейронної мережі. Саме з цих зображень модель навчається розпізнавати ознаки об’єктів, визначати

координати меж та здійснювати класифікацію. Папка val використовується для валідації – оцінки ефективності моделі під час навчання, що дозволяє відстежувати точність і уникнути перенавчання. Папка test призначена для фінального тестування моделі на окремому наборі зображень, які вона ще не бачила після завершення всіх навчальних етапів.

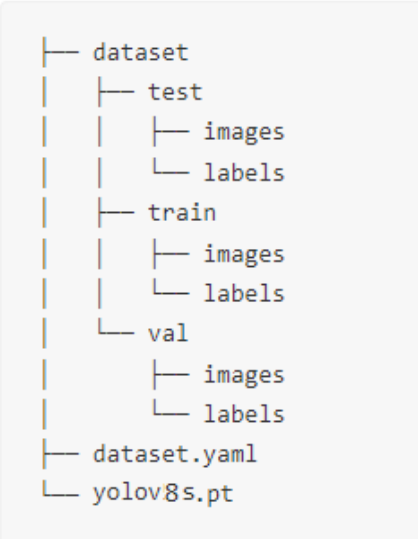

Рисунок 3.6 – Структура папок набору даних для навчання моделі на новому наборі даних

Кожна з папок train, val та test містить два обов'язкових підкаталоги – images та labels. Папка images містить графічні файли у форматах .jpg, .png або .jpeg. Ці файли є вхідними даними для моделі. У свою чергу, папка labels містить текстові файли з розширенням .txt, які мають однакові імена з відповідними зображеннями. Кожен текстовий файл містить анотований опис усіх об'єктів, які присутні на зображенні.

Завдяки такій структурі папок фреймворк Ultralytics може автоматично завантажувати дані, обробляти анотації, проводити навчання моделі, а також генерувати валідаційні метрики без потреби у додатковому кодуванні. Така уніфікація є важливою перевагою YOLO-платформи, яка робить її зручною для швидкої розробки і тестування систем машинного зору.

Попередньо оброблені та розмічені дані нового dataset було розбито на підвибірки train, val, test у відношенні: 70% даних використовувалося для навчання моделі, 20% зарезервовано для валідації а 10% для тестового набору, який використовується для оцінки якості моделі.

У файлі .yaml для другого підходу було прописано шляхи до тренувальної, валідаційної та тестової вибірок, а також вказано кількість класів та їх мітки.

3.2 Навчання моделей YOLOv8n та YOLOv8s

Процес навчання моделей відбувався у середовищі PyCharm із використанням бібліотеки Ultralytics.

На *першому етапі* було вирішено використати моделей YOLOv8n і YOLOv8s попередньо навчені на наборі даних COCO, здійснивши їх та перенавчання для розпізнавання 5 класів об'єктів та тестування на підготовленому новому наборі даних з зображеннями із зони стихійного лиха – масових руйнувань після обстрілів. Для цього моделі було перенавчено для розпізнавання об'єктів п'яти класів – car, truck, bus, motorcycle та person [40].

На рисунку 3.7 наведено фрагмент коду для перенавчання YOLOv8s.

```python
from ultralytics import YOLO

# === Завантаження базової моделі YOLOv8s ===
model = YOLO('G:/Course_four_2/КРБ/Project/YOLO/yolov8n.pt')

# === Навчання моделі ===
model.train(
    data='G:/Course_four_2/КРБ/Project/filtered_balanced_dataset/data.yaml',# Шлях до data.yaml
    epochs=25,                          # Кількість епох
    imgsz=640,                          # Розмір зображень
    batch=8,                            # Batch size для CPU
    name='yolov8n_trein_on_my_data_and_COCO1',     # Назва сесії (папка з результатами)
    project='runs/detect',              # Головна папка з результатами
    exist_ok=True,                      # Не видавати помилку, якщо така папка вже існує
    device='cpu',                       # Навчання на CPU
    patience=10                         # Early stopping – зупинка при відсутності покращення

)
```

Рисунок 3.7 – Перенавчання моделі YOLOv8s для розпізнавання п'яти класів

Детальна оцінка ефективності моделей здійснювалась за основними метриками якості: mAP@0.5, mAP@0.5:0.95, Precision, Recall, а також втратами Loss на етапах навчання та валідації [41]. Аналогічно здійснюють перенавчання моделі YOLOv8n. Після перенавчання моделі було протестовано на підготовленому новому наборі даних для розпізнавання 5 класів. Python-скрипт для тестування перенавченої моделі YOLOv8s наведено на рисунку 3.8. Перенавчання та тестування моделі YOLOv8n здійснено аналогічно.

```
# Завантажити перенавчену модель
model = YOLO('runs/detect/train/weights/best.pt')
# Тестування на test-піднаборі нового датасету
metrics = model.val(
    data='config.yaml',
    split='test',
    imgsz=640,
    conf=0.25,
    iou=0.5,
    save=True  # зберегти візуалізацію результатів
)
# Вивід основних метрик
print("✅ Результати тестування перенавченої YOLOv8s-моделі:")
print(f"mAP@0.5: {metrics.box.map50:.3f}")
print(f"mAP@0.5:0.95: {metrics.box.map:.3f}")
```

Рисунок 3.8 – Python-скрипт для тестування перенавченої моделі YOLOv8s

Для більш глибокого аналізу ефективності процесу навчання моделей були збудовані графіки зміни основних метрик та функцій втрат по епохам. На рисунку 3.9 зображено динаміку таких показників, як box_loss, cls_loss, dfl_loss на навчальній і валідаційній вибірках та метрик Precision, Recall, mAP@50, mAP@50:95 для моделі YOLOv8s. Аналіз показує, що функції втрат стабільно зменшуються, що свідчить про поступове навчання моделі без явного перенавчання. Паралельно з цим, метрики точності демонструють стабільне зростання.

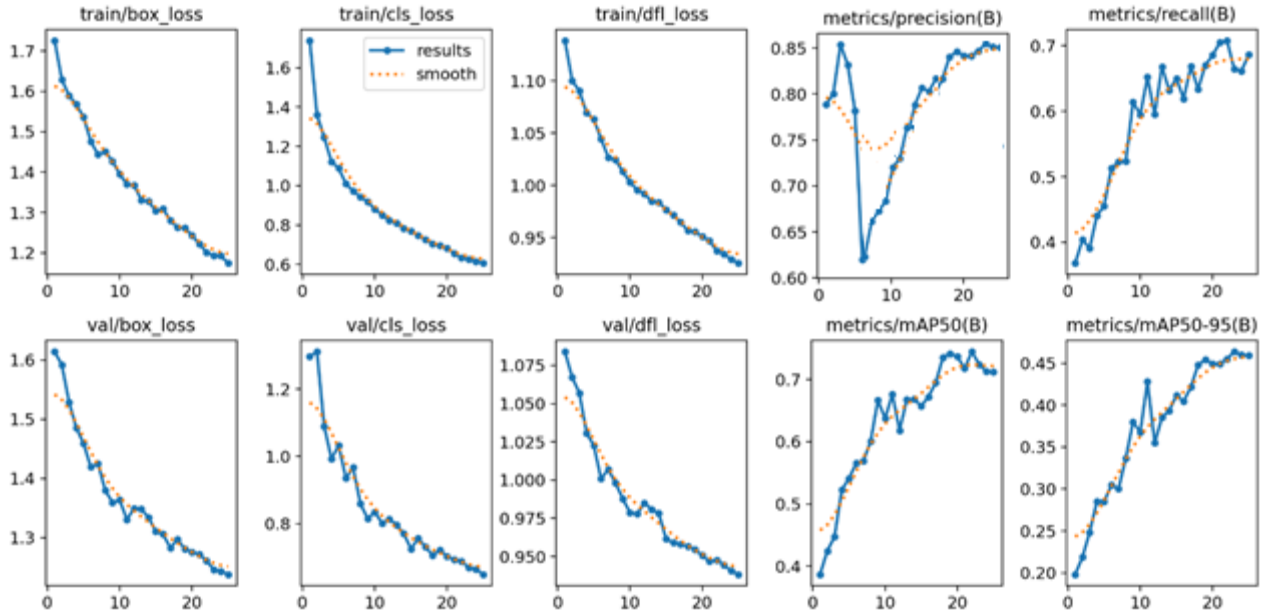

Рисунок 3.9 – Графіки метрик YOLOv8s по епохам

На рисунку 3.10 відображено динаміку показників функції втрат box_loss, cls_loss і dfl_loss на навчальній і валідаційній вибірках для моделі YOLOv8n та метрик Precision, Recall, mAP50, mAP50:95.

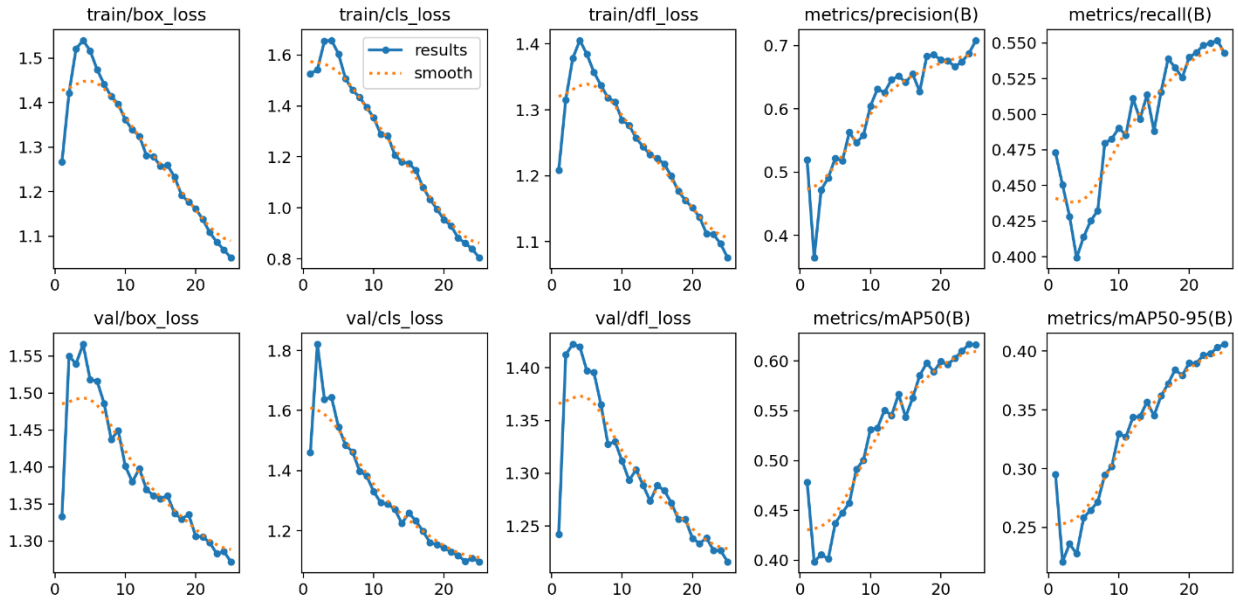

Рисунок 3.10 – Графіки метрик YOLOv8n по епохам

Порівняння моделей YOLOv8s та YOLOv8n свідчить про те, що модель YOLOv8s забезпечує вищу точність розпізнавання об'єктів за рахунок більшої кількості обчислень і тривалішого часу обробки. Модель YOLOv8s за mAP@50 має середню точність 0,77. Для моделі YOLOv8n точність за mAP@50 становила 0,65. Здатність локалізувати об'єкти на високих порогах IoU для обох моделей була нижчою: для YOLOv8s – mAP@50:95 становив лише 0,45, для YOLOv8n – 0,41. Значення Precision для YOLOv8s було рівним 0,85, тоді як для YOLOv8n – 0,70. Recall становила для YOLOv8s 0,71, для YOLOv8n – 0,55.

На *другому етапі* було реалізовано другий підхід, який використовує навчання моделей YOLOv8n і YOLOv8s з нуля на новому наборі даних, підготовленому та розміченому відповідно до формату YOLOv8 і розбитому на навчальний, валідаційний та тестовий набори даних. Навчання моделей YOLOv8n та YOLOv8s з нуля здійснюється з використанням гіперпараметрів уже навчених моделей.

Структура YAML-файлу для опису набору даних (config.yaml) наведена на рисунку 3.11.

```yaml
path: G:/Project/yolo_split_dataset

train: images/train
val: images/val
test: images/test

nc: 5
names: ['car', 'truck', 'bus', 'motorcycle', 'person']
```

Рисунок 3.11 – Структура YAML-файлу під час навчання моделей з нуля на новому dataset

На рисунку 3.12 наведено Python-скрипт для навчання з нуля моделі YOLOv8s. Буде створено каталог runs/detect/yolov8s_custom/, де буде: weights/best.pt – модель з найкращою mAP на валідації та логи, графіки, результати

валідації. Код для тестування найкращої моделі на тестовому наборі даних наведено на рисунку 3.13. Навчання з нуля моделі YOLOv8n та її тестування здійснюється аналогічно.

```python
from ultralytics import YOLO

# Створення нової моделі з нуля
model = YOLO('yolov8s.yaml')  # конфігурація моделі без ваг

# Навчання моделі
model.train(
    data='config.yaml',     # шлях до YAML із описом датасету
    epochs=50,              # кількість епох
    imgsz=640,              # розмір вхідного зображення
    batch=16,               # розмір батчу
    name='yolov8s_custom', # назва експерименту
    device=0                # GPU 0, або 'cpu' якщо без CUDA
)
```

Рисунок 3.12 – Python-скрипт для навчання з нуля моделі YOLOv8s

```python
# Завантажити найкращу модель
model = YOLO('runs/detect/yolov8s_custom/weights/best.pt')

# Оцінити модель на тестовому наборі
model.val(data='config.yaml', split='test', save=True)
```

Рисунок 3.13 – Тестування найкращої моделі YOLOv8s

Навчання здійснювалося протягом 25 епох та проводилося на CPU, що певною мірою обмежувало продуктивність. Застосовувався підхід продовження навчання з останнього збереженого вагового файлу (last.pt або best.pt).

Графіки зміни функцій втрат під час навчання моделей з нуля по епохах зображено на рисунках 3.14 та 3.15.

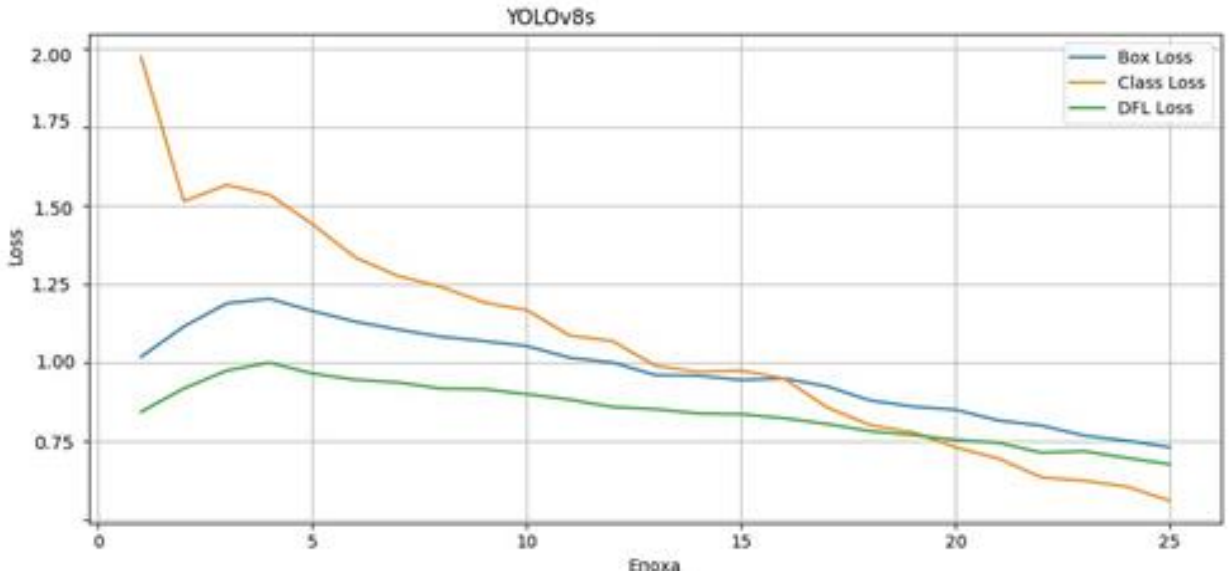

Рисунок 3.14 – Функція втрат YOLOv8s по епохам

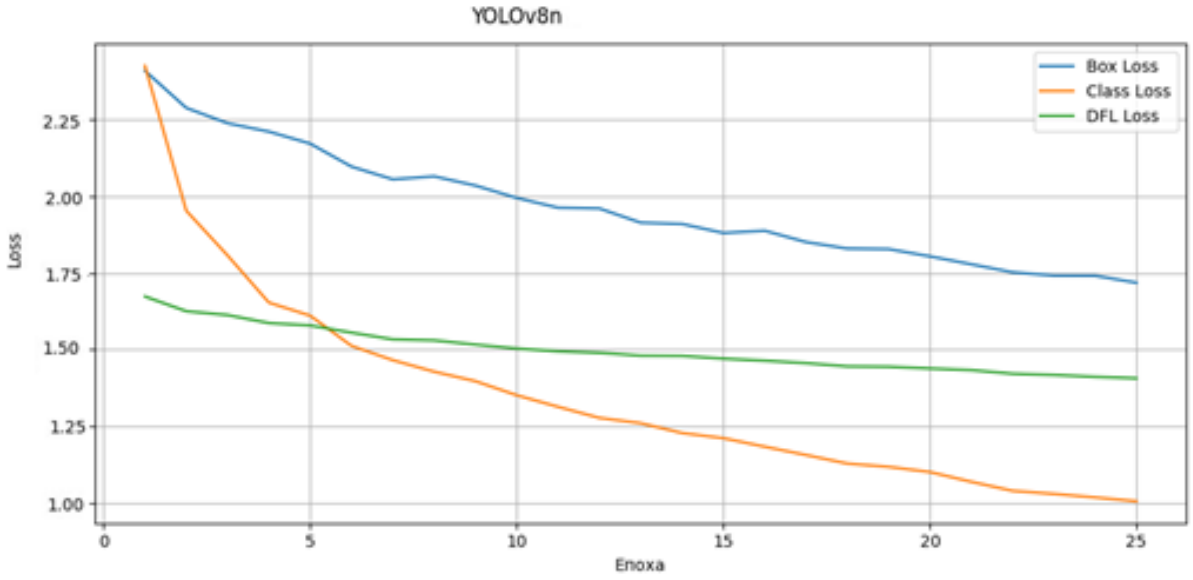

Рисунок 3.15 – Функція втрат YOLOv8n по епохам

Графіки зміни оцінки точності під час навчання моделей mAP@50:95 та mAP@50 по епохах зображено на рисунках 3.16 і 3.17. Отримані результати у разі навчання моделей YOLOv8n і YOLOv8s з нуля на новому підготовленому dataset

показали вищу точність розпізнавання об’єктів у порівнянні з точністю перенавчених на наборі даних COCO для розпізнавання 5-ти об’єктів моделей YOLOv8n і YOLOv8s, які було досліджено на першому етапі.

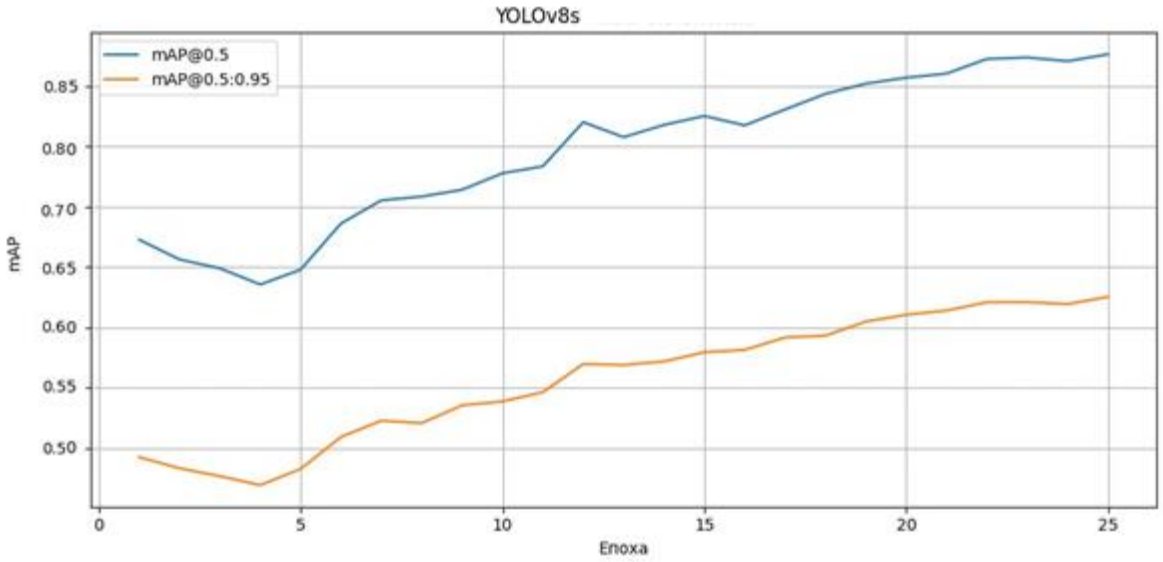

Рисунок 3.15 – Оцінка mAP YOLOv8s по епохам

Рисунок 3.15 – Оцінка mAP YOLOv8n по епохам

Порівняння навчених з нуля моделей YOLOv8s та YOLOv8n продемонструвало кращу точність розпізнавання об’єктів для моделі YOLOv8s.

Значення Precision для YOLOv8s було рівним 0,94, для YOLOv8n – 0,88. Recall становила для YOLOv8s 0,81, для YOLOv8n – 0,70. Середня точність за mAP@50 для моделі YOLOv8s становила 0,89, що є прийнятним результатом. Для моделі YOLOv8n точність за mAP@50 рівна 0,83. Навчання моделей з нуля на новому dataset супроводжувалося також зростанням здатності локалізувати об’єкти на високих порогах IoU. Для моделі YOLOv8s точність за mAP@50:95 становила 0,63, для YOLOv8n – 0,57.

Порівняння двох підходів дозволяє зробити низку важливих висновків:

– базові перенавчені моделі YOLOv8n та YOLOv8s демонструють прийнятну швидкодію, проте низьку точність у спеціалізованих умовах;

– навчання на новому наборі даних, що містить зображення із зони стихійного лиха – руйнувань після обстрілів значно покращує результати, особливо для об’єктів, які складно розпізнати в загальному випадку (наприклад, люди на знімках із дронів);

– YOLOv8s є більш ефективною для даного завдання за рахунок більшої кількості параметрів, однак потребує більше ресурсів;

– при правильній структурі dataset навіть YOLOv8n може бути застосована у практичних умовах із обмеженими ресурсами, як от ПК без GPU.

Перенавчені та навчені з нуля моделі YOLOv8n і YOLOv8s експортуються у файли з розширенням .pt та використовуються далі для інтеграції у застосунок і використання для розпізнавання об’єктів при отриманні даних з БПЛА. Під час розпізнавання об’єктів з даних, що надходять з БПЛА, у застосунку реалізовано можливість їх вибору. Однак отримані результати демонструють, що у проєкті доцільно використовувати навчені з нуля моделі YOLOv8n і YOLOv8s із врахуванням можливостей пристрою, на якому працюватиме застосунок. Це дозволяє значно підвищити ефективність системи обробки зображень без надмірних ресурсних витрат.

Висновки до розділу 3

У третьому розділі описано підготовку набору даних до навчання та навчання моделей YOLOv8n і YOLOv8s для розпізнавання об’єктів із зони масових руйнувань після обстрілів, які належать до п’яти класів: та 4 класи транспортних засобів: car, truck, bus, motorcycle, та клас людей – person.

Під час реалізації було використано два варіанти моделей – YOLOv8n (nano) та YOLOv8s (small), які відрізняються між собою складністю, кількістю параметрів, ресурсомісткістю та, відповідно, точністю. Детальна оцінка ефективності моделей здійснювалась за основними метриками якості: mAP@0.5, mAP@0.5:0.95, Precision, Recall, а також втратами Loss на етапах навчання та валідації.

При навчанні моделей було реалізовано два ключові підходи: 1) навчання моделей з нуля на новому наборі даних, підготовленому відповідно до формату YOLO; 2) перенавчання попередньо навчених на наборі даних COCO моделей для розпізнавання об’єктів 5 класів із тестуванням на новому наборі даних.

Проведене порівняння показує доцільність адаптації моделей до конкретного застосування. Попередньо натреновані моделі забезпечують швидкий старт та задовільні результати для загальних випадків, однак справжнє якості та точності розпізнавання продемонстровано при навчанні моделей з нуля на нових спеціалізованих наборах даних. У рамках проєкту було успішно реалізовано обидва підходи для моделей YOLOv8n і YOLOv8s, що дозволило знайти баланс між продуктивністю та точністю, а також зробити систему універсальною та гнучкою для реальних умов використання.

4 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОБРОБКИ ДАНИХ БПЛА

4.1 Моделювання інформаційної системи

На етапі моделювання інформаційної системи було розроблено діаграму сценаріїв використання системи яка відображає взаємодію користувача з системою та функціональні вимоги до неї (рис. 4.1).

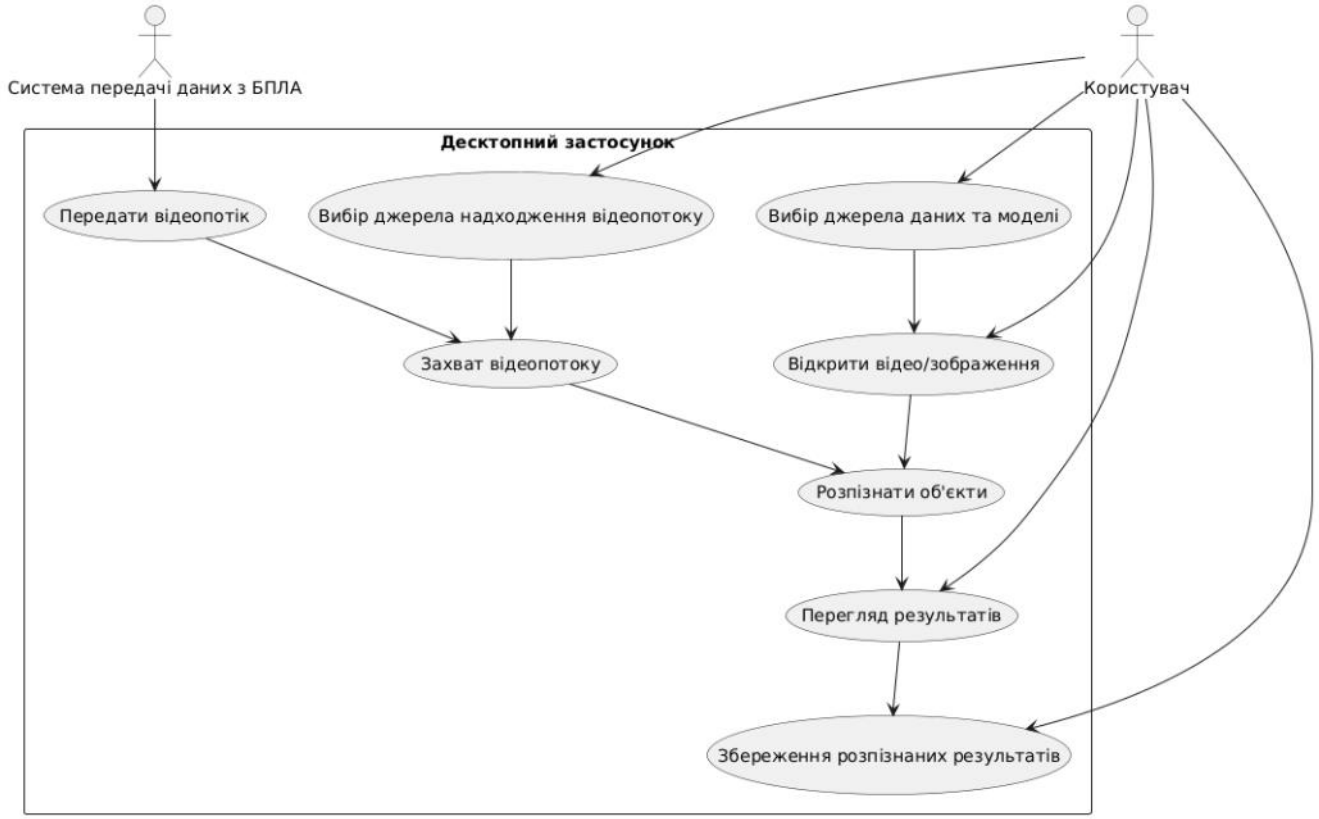

Рисунок 4.1 – Діаграма прецедентів системи

Діаграма прецедентів відображає основні сценарії взаємодії користувача з декстопним застосунком, що реалізує функціональність розпізнавання об'єктів у зображеннях, відео та потоковому відео за допомогою моделей YOLOv8. Взаємодія здійснюється через графічний інтерфейс застосунку, реалізований із використанням бібліотеки PyQt5.

Згідно з діаграмою, система підтримує такі сценарії використання:

– Use Case 1: вибір джерела даних та моделі – користувач має можливість обрати одну з доступних перенавчених або навчених на новому наборі даних моделей YOLOv8n і YOLOv8s;

– Use Case 2: відкрити відео/зображення – користувач завантажує файл із зображенням або відео;

– Use Case 3: передати відеопотік – система передачі даних з БПЛА передає дані через протоколи Wi-Fi/5G або потокові протоколи (RTSP, HTTP);

– Use Case 4: вибір джерела надходження відеопотоку – користувач з БПЛА має можливість обрати канал отримання стріму з БПЛА за IP-адресою або шляхом захоплення екрану;

– Use Case 5: захват відеопотоку – десктопний застосунок із використанням бібліотеки OpenCV, яка застосовується до IP та RTSP-потоків при роботі зі стрімами, відкриває пряме з’єднання з відеопотоком дрона;

– Use Case 6: розпізнати об’єкти – після вибору файлів відео чи зображення або захвату відеопотоку система автоматично з використанням обраної моделі ініціює процес розпізнавання об’єктів; цей сценарій є базовим, кожен раз, коли система отримує нове зображення чи кадр відео або стріму, автоматично виконується розпізнавання за обраною моделлю;

– Use Case 7: перегляд результатів – зображення та кожен кадр відео/стріму аналізується, відображаючи результати з розпізнаними об’єктами. Для стріму обробка кадрів виконується в реальному часі, а результати розпізнавання миттєво виводяться на екран. Результати обробки виводяться у графічному інтерфейсі у області перегляду застосунку та у текстовому форматі. Об’єкти, розпізнані на зображенні або у кадрі, візуалізуються рамками та підписами з іменем класу і значенням Confidence score – впевненості моделі у тому, що об’єкт знаходиться у рамці;

– Use Case 8: збереження результатів – після завершення обробки користувач має змогу зберегти: файли зображень та кадрів відео/стріму з розпізнаними

об’єктами у рамках, текстовий звіт із переліком виявлених об'єктів, їх кількістю та значенням Confidence score для кожного розпізнаного об’єкту.

Діаграма сценаріїв використання ілюструє логічну структуру взаємодії між користувачем і системою, де кожна дія є частиною єдиного робочого процесу: вибір моделі та джерела даних → отримання даних → обробка → вивід → збереження. Завдяки цьому застосунок забезпечує інтуїтивну і послідовну роботу з різними джерелами інформації.

В цілому на першому етапі передбачено, що буде здійснено збирання відео- або фотоданих за допомогою дрона, після чого отримана інформація передається на мобільний пристрій або пункт управління. Далі система проводить попередню обробку даних у локальному середовищі – зокрема, аналізує відео, зображення або інші мультимедійні дані. Після обробки виконується автоматичне розпізнавання об’єктів за допомогою моделей YOLOv8. Отримані результати візуалізуються у графічному інтерфейсі, де користувач має можливість перегляду, верифікації або аналізу даних. На завершальному етапі створюється звіт, що містить аналітичну інформацію, ідентифіковані об’єкти та їх характеристики. Ця інформація використовується у подальшому для ухвалення рішень стосовно проведення пошуку та порятунку людей у зонах руйнувань внаслідок бойових дій.

4.2 Інструментальні засоби розробки системи

Під час розробки програмного забезпечення для виявлення об'єктів на відео та зображеннях було обрано ряд сучасних і перевірених технологій, які дозволили забезпечити стабільну роботу застосунку, його інтеграцію з моделями штучного інтелекту та створити зручний для користувача інтерфейс. Вибір кожного компоненту обумовлювався його функціональністю, продуктивністю, документацією та сумісністю з іншими елементами.

Основною мовою програмування для реалізації застосунку було обрано Python 3.12.6. Ця мова є однією з найпопулярніших у сфері штучного інтелекту,

комп'ютерного зору та обробки зображень. Python має велику кількість бібліотек, які значно спрощують розробку, а також активну спільноту, що сприяє швидкому вирішенню технічних питань. Версія Python 3.12.6 забезпечує покращену продуктивність порівняно з попередніми версіями, що також стало перевагою для обробки відео в реальному часі. Розробка застосунку здійснювалася у середовищі PyCharm [42, 43].

Для аналізу ефективності моделей YOLOv8n та YOLOv8s після завершення навчання було використано низку Python-бібліотек, що дозволили вивести ключові метрики (mAP@0.5, mAP@0.5:0.95, loss, precision, recall) у графічному форматі. Зокрема, застосовувалися такі бібліотеки:

– pandas: для обробки CSV-файлу results.csv, який автоматично генерується бібліотекою Ultralytics під час навчання, цей файл містить епохи, значення втрат, метрик точності та навчальні коефіцієнти (lr) [44];

– matplotlib: для побудови лінійних графіків залежності метрик від епох – функцій втрат (box_loss, cls_loss, dfl_loss) та метрик (mAP, precision, recall), які дозволяють оцінити стабільність, точність та якість навчання;

– seaborn (опційно): для стилізації візуалізацій та побудови узагальнених діаграм, таких як порівняльні boxplot-графіки між моделями [45];

– os і glob: для автоматизованого пошуку та відкриття результатів навчання з відповідної папки (наприклад, runs/detect/yolov8s_custom/), що дає змогу масштабувати аналіз на декілька експериментів;

– Ultralytics: безпосередньо для запуску тренування та автоматичної генерації логів, графіків і результатів оцінки після кожної епохи.

Для реалізації графічного інтерфейсу користувача було використано бібліотеку PyQt5, яка є потужним інструментом для створення сучасних GUI-додатків. PyQt5 дозволяє створювати багатовіконні інтерфейси, підключати обробники подій та легко інтегрувати графічні елементи з логікою програми. Особливо цінною є її здатність працювати з QTimer і QImage, які використовуються для виводу відеопотоку на екран у реальному часі. QTimer та QImage є важливими

класами бібліотеки PyQt5 для роботи з таймерами та зображеннями відповідно. QTimer використовується для створення таймерів, які можуть періодично викликати функцію, передану в нього. QImage використовується для створення, обробки та відображення зображень у застосунку.

Платформа Ultralytics, як уже було зазначено, використовувалася для перенавчання та навчання моделей YOLOv8n та YOLOv8s. Детально це було описано у розділі 3. Моделі, які створюються на платформі Ultralytics мають ряд унікальних властивостей:

– продуктивність в реальному часі: високошвидкісні можливості виведення та навчання для чутливих ко часу застосунків;

– універсальність: підтримка задач виявлення, сегментації, класифікації та оцінки положень в єдиній системі;

– попередньо навчені моделі: доступ до високопродуктивним, попередньо навчених моделей, що скорочує час навчання;

– розширена підтримка спільноти: активне співтовариство та вичерпна документація для усунення неполадок і розробки;

– проста інтеграція: простий API для інтеграції з існуючими проектами та робочими процесами.

Ultralytics забезпечує підтримку різноманітних наборів даних для вирішення завдань комп’ютерного зору, таких як виявлення, сегментація об’єктів, оцінка позиції, класифікація та відстеження кількох об’єктів. При розробці системи у даній роботі було використано набір даних COCO, що включає 80 категорій об'єктів. COCO є набором даних по виявленню, сегментації об’єктів і створенню надписів.

Платформа Ultralytics використовує сегментацію екземплярів, яка є технікою комп’ютерного зору, що дозволяє ідентифікувати та локалізувати об'єкти на зображенні на рівні пікселів. На відміну від семантичної сегментації, яка класифікує тільки кожен піксель, сегментація екземплярів відрізняє різні

екземпляри одного класу. А COCO призначений для задач виявлення об'єктів, сегментації та створення надписів, містить понад 200 тисяч помічених зображень.

Платформа Ultralytics реалізує також метод оцінки пози, який використовується для визначення положення об’єкта відносно камери або світової системи координат. Для цього необхідно визначити ключові точки або суглоби на об’єктах, зокрема на людях або тваринах. Зокрема, COCO є набором даних з анотаціями по людині, призначений для оцінки завдань поз.

Платформа Ultralytics дозволяє реалізувати багато об’єктне спостереження, яке включає в себе виявлення і відстеження кількох об’єктів у відеопотоці. Ця задача розширює можливості виявлення об'єктів, зберігаючи їх ідентичність у різних кадрах.

Для обробки зображень і відео, а також для взаємодії з камерами, зокрема IP-камерами чи стрімами з дронів, у проєкті було задіяно бібліотеку OpenCV (англ. Open Source Computer Vision Library), яка активно використовується разом із моделями YOLO для розпізнавання об'єктів на зображеннях і відео [46]. OpenCV допомагає завантажити відео або зображення, попередньо обробити кадри чи зображення та відображати результати детекції в реальному часі. Під час попередньої обробки здійснюється зміна розміру зображення (кадру) та форматування. Також OpenCV відіграє важливу роль у конвертації кольорового простору, що є критичною частиною попередньої обробки зображень та відео перед подачею їх на модель YOLOv8. Це є необхідним, оскільки OpenCV читає зображення у форматі BGR (Blue, Green, Red), а моделі комп'ютерного зору YOLOv8, очікують вхід у RGB-форматі. Тому конвертація з BGR в RGB забезпечує правильну інтерпретацію кольорів під час розпізнавання об’єктів.

У даній роботі OpenCV забезпечує зчитування відеопотоку, конвертацію кольорового простору, а також детекцію об’єктів на зображеннях та кадрах [47]. Для відображення результатів детекції здійснюється малювання рамок та запис відео. Тобто, цей інструмент надає широкий спектр функцій для роботи з

візуальною інформацією: від зчитування кадрів до малювання фреймів і розпізнавання об’єктів.

Для збереження результатів обробки – як відео з нанесеними об'єктами, так і звітів з результатами розпізнавання, було обрано популярні формати. Фінальне відео з результатами детекції зберігається у форматі MP4, що забезпечує хорошу якість при відносно малій вазі та є підтримуваним більшістю плеєрів. Текстовий звіт зберігається у форматі .txt, де перераховані всі розпізнані об’єкти з ймовірностями. Такий формат зручний для подальшої аналітики, оскільки його легко обробляти програмно.

Однією з важливих частин розробки є підтримка потокових джерел зображення. У застосунку реалізовано можливість підключення до IP-камер або стрімів із дронів через URL або Wi-Fi/RTSP. Це дозволяє застосовувати програму в умовах реального середовища, наприклад, для моніторингу територій, охорони або зйомок з повітря. Така гнучкість дає змогу адаптувати систему до різних джерел відео без необхідності модифікації основного коду.

Уся обробка даних відбувається у локальному середовищі, що забезпечує швидкість та конфіденційність. Застосунок було протестовано на ноутбуці ASUS ROG Zephyrus G14 з операційною системою Windows 11, що забезпечило достатній рівень обчислювальної потужності для запуску моделей YOLOv8 у реальному часі. Системні вимоги для роботи застосунку залежать від обраної моделі: YOLOv8n працює швидше, але менш точно, YOLOv8s – повільніше, але з вищою точністю.

Кожен з компонентів стеку технологій було обрано з урахуванням його стабільності, гнучкості та зручності інтеграції. Таке поєднання дозволило створити застосунок, який не лише виконує детекцію об'єктів, а й має дружній до користувача інтерфейс, можливість збереження результатів, гнучке налаштування джерела даних та підтримку повноекранного режиму.

Описаний стек технологій став фундаментом для реалізації повноцінного інструменту виявлення об’єктів, який може бути застосований для реального

використання у відеоспостереженні, контролі доступу або у сфері безпілотного моніторингу територій у зонах руйнувань після обстрілів.

4.3 Програмна реалізація застосунку для розпізнавання даних з БПЛА

Застосунок інформаційної системи, до якого інтегрують моделі YOLO, складається з одного основного графічного вікна MainWindow з елементами управління відео, та кнопками для вибору того, які дані будуть оброблятися (зображення, відео чи стрім) та моделі YOLO й області для виведення детекцій та візуалізації зображень і кадрів.

Основні функціональні можливості застосунку є наступними:

- завантаження зображення або відеофайлу;
- підключення до стріму (вебкамера, IP-камера, дрон);
- вибір моделей YOLOv8n та YOLOv8s, перенавчених та навчених з нуля на новому наборі даних;
- виведення частоти кадрів за секунду FPS (англ. Frame Per Second) у режимі відео чи стріму;
- збереження отриманих результатів у форматах відео та текстовому у окремій папці;
- контроль відео: пауза, промотка вперед/назад;
- запуск застосунку у повноекранному режимі.

З використанням бібліотеки PyQt5 для створення графічного інтерфейсу GUI реалізовано клас MainWindow, який є нащадком базового класу QMainWindow (рис. 4.3). Метод _ _init_ _() ініціалізує головне вікно, setWindowTitle() задає назву застосунку, showMaximized() робить вікно повноекранним одразу при запуску [48].

Всі елементи інтерфейсу компонуються за допомогою вертикальних (QVBoxLayout) та горизонтальних (QHBoxLayout) блоків. Кнопки додаються у вертикальне компонування buttons. Для читабельності зверху розміщено надпис «Оберіть модель». На рисунках 4.4 та 4.5 наведено фрагменти коду для вибору

моделей, додавання кнопок у вікні застосунку та поля для виведення показників з результатами розпізнавання об’єктів. Кнопки та поле з виведенням результатів розміщуються у правій частині вікна застосунку.

```
45  class MainWindow(QMainWindow):  1 usage
46      def __init__(self):
47          super().__init__()
48          self.setWindowTitle("Object Detection App")
49          self.setWindowIcon(QIcon("icons/app.png"))
50          self.showFullScreen()
51
```

Рисунок 4.3 – Код для створення класу MainWindow

```
98
99          buttons = QVBoxLayout()
100         buttons.addWidget(QLabel("Оберіть модель:"))
101         buttons.addWidget(self.model_selector)
102         for btn in [self.open_image_btn, self.open_video_btn, self.stream_btn,
103                     self.save_btn, self.pause_btn, self.forward_btn,
104                     self.backward_btn, self.exit_btn]:
105             buttons.addWidget(btn)
106
```

Рисунок 4.4 – Обрання моделі для аналізу

```
106
107         right_layout = QVBoxLayout()
108         right_layout.addLayout(buttons)
109         right_layout.addWidget(QLabel("Результати"))
110         right_layout.addWidget(self.results_list)
111
112         main_layout = QHBoxLayout()
113         main_layout.addWidget(self.image_label, stretch=3)
114         main_layout.addLayout(right_layout, stretch=1)
115
```

Рисунок 4.5 – Створення поля з виведенням результатів детекції та розпізнавання

У лівій частині вікна передбачена область для виведення зображення або відео, на якому здійснюється розпізнавання об’єктів – віджет (рис. 4.6).

Встановлено пропорції розміру stretch=3 для віджету зображення та stretch=1 для панелі керування.

```
16  class ImageLabel(QLabel):  1 usage
17      def __init__(self):
18          super().__init__()
19          self.setAlignment(Qt.AlignCenter)
20          self._zoom = 1.0
21          self.setMinimumSize(400, 300)
22          self.setStyleSheet("border: 1px solid gray")
23
```

Рисунок 4.6 – Створення віджета для виведення обробленого кадру відео або зображення

Клас ImageLabel – це модифікований клас QLabel, який у бібліотеці PyQt5 є віджетом для відображення тексту або зображення. Він використовується для створення елементів інтерфейсу, які необхідно відображати на екрані, не припускаючи прямої користувальницької взаємодії. У застосунку його було використано для виведення обробленого кадру з відео або зображення: setAlignment(Qt.AlignCenter) вирівнює зображення по центру, setMinimumSize() задає мінімальний розмір вікна, setStyleSheet() додає рамку навколо зображення для візуального відділення.

У віджеті реалізовано масштабування зображення і відео за допомогою колеса мишки. Для цього використано перевизначений метод wheelEvent(), який дозволяє збільшувати та зменшувати зображення за допомогою прокрутки миші (рис. 4.7).

Масштаб оновлюється функцією update_image(). OpenCV-кадр масштабується та перетворюється у формат QImage, який відображається у QLabel (рис. 4.8) [49].

Кожна кнопка має піктограму та текстову підказку (рис. 4.9).

```
23
24      def wheelEvent(self, event: QWheelEvent):
25          if event.angleDelta().y() > 0:
26              self._zoom *= 1.1
27          else:
28              self._zoom /= 1.1
29          self.update_image()
30
```

Рисунок 4.7 – Метод wheelEvent

```
30
31      def update_image(self):  2 usages
32          if hasattr(self, 'image'):
33              resized = cv2.resize(self.image,  dsize: None, fx=self._zoom, fy=self._zoom)
34              rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
35              h, w, ch = rgb.shape
36              qimg = QImage(rgb.data, w, h, 3 * w, QImage.Format_RGB888)
37              self.setPixmap(QPixmap.fromImage(qimg))
38
```

Рисунок 4.8 – Масштабування та перетворення формату OpenCV-кадру у функції update_image()

```
66          self.open_image_btn = QPushButton("Відкрити зображення")
67          self.open_image_btn.setIcon(QIcon("icons/image.png"))
68          self.open_image_btn.clicked.connect(self.load_image)
69
70          self.open_video_btn = QPushButton("Відкрити відео")
71          self.open_video_btn.setIcon(QIcon("icons/video.png"))
72          self.open_video_btn.clicked.connect(self.load_video)
73
74          self.stream_btn = QPushButton("Стрім (IP / дрон)")
75          self.stream_btn.setIcon(QIcon("icons/stream.png"))
76          self.stream_btn.clicked.connect(self.start_stream)
77
78          self.save_btn = QPushButton("Зберегти результати")
79          self.save_btn.setIcon(QIcon("icons/save.png"))
80          self.save_btn.clicked.connect(self.save_results)
```

Рисунок 4.9 – Задання піктограм та текстових підказок для кнопок у функції update_image()

У застосунку було реалізовано кнопки з підтримкою базового керування відео: пауз та перемотки вперед і назад на 30 кадрів (рис. 4.10).

```
81
82          self.pause_btn = QPushButton("Пауза")
83          self.pause_btn.clicked.connect(self.toggle_pause)
84
85          self.forward_btn = QPushButton(">>")
86          self.forward_btn.clicked.connect(self.skip_forward)
87
88          self.backward_btn = QPushButton("<<")
89          self.backward_btn.clicked.connect(self.skip_backward)
90
```

Рисунок 4.10 – Задання кнопок для підтримки базового керування відео у функції update_image()

Користувач має змогу зупинити відео, переглянути важливий момент повторно або перемотати вперед. Функції для управління відео наведені на рисунку 4.11.

```
217     def toggle_pause(self):  1 usage
218         self.paused = not self.paused
219
220     def skip_forward(self):  1 usage
221         if self.cap and self.video_path:
222             frame_no = int(self.cap.get(cv2.CAP_PROP_POS_FRAMES))
223             self.cap.set(cv2.CAP_PROP_POS_FRAMES, frame_no + 30)
224
225     def skip_backward(self):  1 usage
226         if self.cap and self.video_path:
227             frame_no = int(self.cap.get(cv2.CAP_PROP_POS_FRAMES))
228             self.cap.set(cv2.CAP_PROP_POS_FRAMES, max(0, frame_no - 30))
229
```

Рисунок 4.11 – Функції для управління відео

Однією з ключових переваг застосунку є підтримка кількох моделей детекції об’єктів. Для цього у коді передбачено словник model_paths, де за ключами зберігаються шляхи до різних моделей YOLOv8 (рис. 4.12). За замовчуванням завантажується модель YOLOv8n, що є найлегшою і найшвидшою. За потреби користувач може змінити модель через випадаючий список (рис. 4.13). Це дозволяє динамічно перемикатися між моделями під час роботи програми без необхідності її перезапуску.

```
51
52          self.model_paths = {
53              "YOLOv8n": "YOLO/yolov8n.pt",
54              "YOLOv8s": "YOLO/yolov8s.pt",
55              "Custom": "runs/detect/train4/weights/best.pt"
56          }
57          self.model = YOLO(self.model_paths["YOLOv8n"])
```

Рисунок 4.12 – Словник зі шляхами до моделей YOLOv8

```
self.model_selector.currentIndexChanged.connect(self.change_model)

def change_model(self):
    model_name = self.model_selector.currentText()
    self.model = YOLO(self.model_paths[model_name])
```

Рисунок 4.13 – Список для вибору моделей YOLOv8

Обробка відео виконується з частотою ~30 мс, що відповідає 30 кадрам на секунду реалізована за допомогою функції update_frame(), яка зчитує кадр, виконує та оновлює інтерфейс (рис. 4.14). Тут: self.cap.read() – читає наступний кадр з відеопотоку або відеофайлу і повертає ret = True та frame (кадр) у разі успішного зчитування, якщо кадр не прочитано (ret = False), таймер зупиняється, і відеопотік звільняється (release()); self.model(frame) – передає кадр у модель YOLOv8 для інференсу (детекції об’єктів), повертає список результатів (в даному випадку – лише перший елемент [0]).

```
169     def update_frame(self):  1 usage
170         if self.paused or self.cap is None:
171             return
172         ret, frame = self.cap.read()
173         if not ret:
174             self.timer.stop()
175             if self.cap:
176                 self.cap.release()
177             if self.output_writer:
178                 self.output_writer.release()
179             return
```

Рисунок 4.14 – Функція update_frame()

Функція self.display_results(frame, results) обробляє результати та відображає їх у GUI (рис. 4.15). Якщо кадр зчитано успішно, модель робить передбачення, а функція display_results виводить їх.

```
191
192     def display_results(self, frame, results):  2 usages
193         self.results_list.clear()
194         for r in results.boxes:
195             cls_id = int(r.cls[0].item())
196             label = self.model.names[cls_id]
197             conf = r.conf[0].item()
198             x1, y1, x2, y2 = map(int, r.xyxy[0])
199             color = (0, 255, 0) if conf >= 0.6 else (0, 255, 255)
200             cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
201             cv2.putText(frame,  text: f"{label}: {conf:.2f}",  org: (x1, y1 - 10),
202                         cv2.FONT_HERSHEY_SIMPLEX,  fontScale: 0.6, color,  thickness: 2)
203             self.results_list.addItem(f"{label}: {conf:.2f}")
204             self.detected_objects.append(f"{label}: {conf:.2f}")
205         return frame
```

Рисунок 4.15 – Функція self.display_results()

Код на рисунку 4.14 додає прямокутники на об'єкти, підписує їх мітками та виводить ймовірність. Дані зберігаються в список detected_objects для подальшого формування звіту. Пояснення:

– self.results_list.clear() – очищає список детекцій в інтерфейсі перед додаванням нових;

– results.boxes – це виявлені об'єкти (bounding boxes);

– r.cls[0].item() – ідентифікатор класу (наприклад, 0 = person, 1 = car);

– self.model.names[cls_id] – назва класу за ідентифікатором;

– r.conf[0].item() – ймовірність (confidence) для передбачення;

– r.xyxy[0] – координати прямокутника (ліва верхня і права нижня точка);

– cv2.rectangle – малює прямокутник навколо об'єкта;

– cv2.putText – додає підпис з назвою класу та ймовірністю;

– self.results_list.addItem(...) – додає результат у віджет списку;

– self.detected_objects.append(...) – зберігає всі розпізнані об’єкти у список для подальшого звіту;

– self.image_label.set_image(frame) – оновлює відображення кадру в інтерфейсі (використовується власний метод для обробки QImage/Pixmap().

На рисунку 4.16 наведено код функції start_strim(), яка реалізує надходження даних зі стріму.

```
def start_stream(self):  1 usage
    choice, ok = QInputDialog.getItem(
        self, "Тип стріму",
        "Оберіть джерело:",
        ["Ввести IP-адресу", "Захоплення екрана"],
        editable=False
    )

    if ok:
        if choice == "Ввести IP-адресу":
            ip, ok = QInputDialog.getText(self, "Введення IP", "Введіть посилання на стрім:")
            if ok and ip:
                self.cap = cv2.VideoCapture(ip)
                self.prepare_output()
                self.timer.start(30)
        elif choice == "Захоплення екрана":
            self.start_screen_capture()
```

Рисунок 4.16 – Функція start_strim()

Функція cv2.VideoCapture(ip) з бібліотеки OpenCV використовується для відкриття та прямого з’єднання з відеопотоком дрона і застосовується до IP та

RTSP-потоків при роботі зі стрімами. Тут передбачено отримання стріму з БПЛА через два канали: за IP-адресою та шляхом захоплення екрану. За IP-адресою дрон передає мережевий RTSP-відеопотік, який можна підключити через URL. Варіант захоплення з екрану передбачає, що відео з БПЛА вже транслюється або відображається на екрані комп'ютера у вікні програми. Тоді застосунок просто захоплює частину екрана, де транслюється відео.

Після завершення обробки та у разі потреби користувача (він має натиснути кнопку) створюється нова папка, де зберігаються результати у відео, графічному та текстовому форматі. Для цього створена функція prepare_output() (рис. 4.17).

```
191
192     def display_results(self, frame, results):  2 usages
193         self.results_list.clear()
194         for r in results.boxes:
195             cls_id = int(r.cls[0].item())
196             label = self.model.names[cls_id]
197             conf = r.conf[0].item()
198             x1, y1, x2, y2 = map(int, r.xyxy[0])
199             color = (0, 255, 0) if conf >= 0.6 else (0, 255, 255)
200             cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
201             cv2.putText(frame, text: f"{label}: {conf:.2f}", org: (x1, y1 - 10),
202                         cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.6, color, thickness: 2)
203             self.results_list.addItem(f"{label}: {conf:.2f}")
204             self.detected_objects.append(f"{label}: {conf:.2f}")
205         return frame
```

Рисунок 4.17 – Функція prepare_output()

Пояснення до функції prepare_output():

– datetime.now().strftime(...) – створює унікальний мітку часу у форматі: рік-місяць-день_година-хвилина-секунда., це потрібно, щоб уникнути перезапису файлів попередніх запусків;

– os.path.join(...) – формує шлях до папки results/result_<дата_час>;

– os.makedirs(..., exist_ok=True) – створює цю папку, якщо її ще немає, якщо вже є – помилка не виникне [50].

Отже ця функція створює нову папку щоб зберегти результати у вказаній папці. Після цього відбувається підготовка до запису і зчитування результатів обробки даних, для відео – щоб кадри обробленого відео збереглося у правильній роздільності та FPS. Далі здійснюється очищення області вікна з результатами – щоб уникнути помилок з попередніх запусків. Таким чином, результати кожного запуску зберігаються окремо, що дозволяє вести історію обробок.

Для контролю продуктивності застосунку щосекунди обчислюється FPS (рис. 4.18). Це дозволяє в реальному часі бачити, наскільки швидко працює застосунок при обробці відео – особливо важливо при використанні IP-камер або дронів.

```
fps_text = f"FPS: {1 / (time.time() - self.start_time):.2f}"
self.start_time = time.time()
cv2.putText(frame, fps_text, org: (10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 1.0, color: (0, 0, 255), thickness: 2)
self.image_label.set_image(frame)
```

Рисунок 4.18 – Лічильник FPS

4.4 Інтерфейс користувача та тестування застосунку

Інтерфейс застосунку, вікно якого наведено на рисунку 4.18, є інтуїтивно зрозумілим. У правій частині вікна зверху знаходиться панель керування із кнопками: *Відкрити зображення*, *Відкрити відео*, *Стрім (IP/дрон)*, *Вийти* та кнопками з іконками >> і << для керування відтворенням.

Над кнопками розміщено список для вибору моделі. Для початку роботи з застосунком користувач може обрати модель, яка буде використовуватися для розпізнавання об'єктів при обробці даних з БПЛА. Для цього необхідно відкрити список, що розкривається під надписом *Оберіть модель* (рис. 4.19). По замовчуванню буде використовуватися модель YOLOv8n.

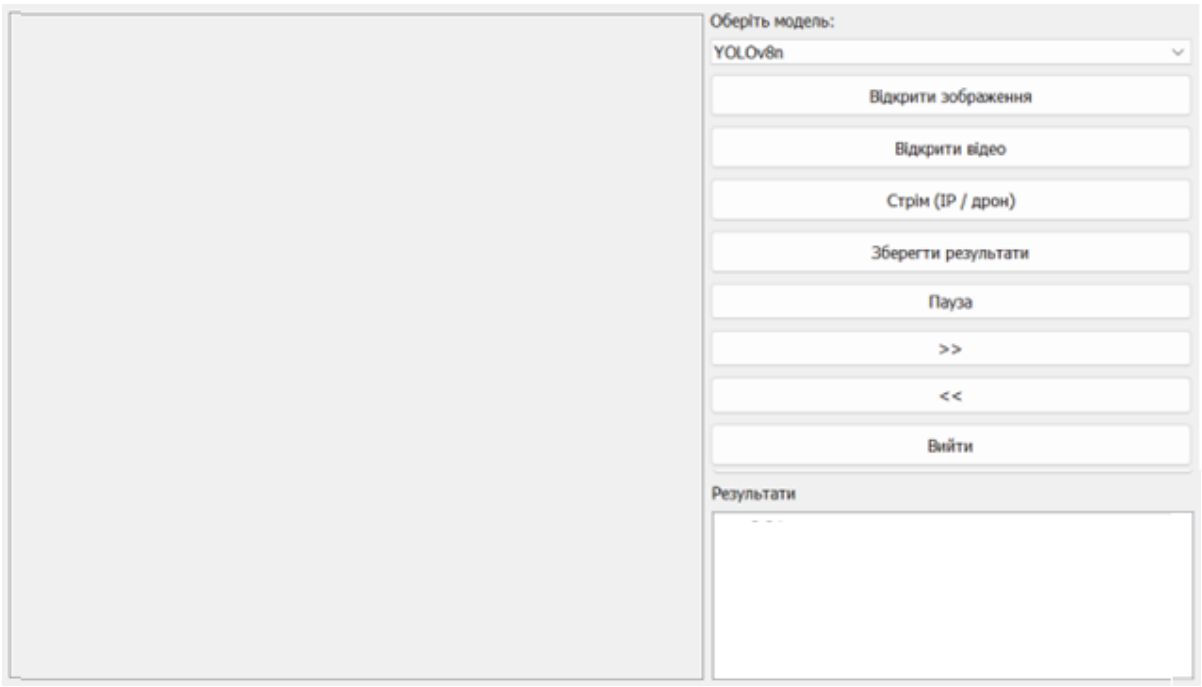

Рисунок 4.18 – Вікно застосунку

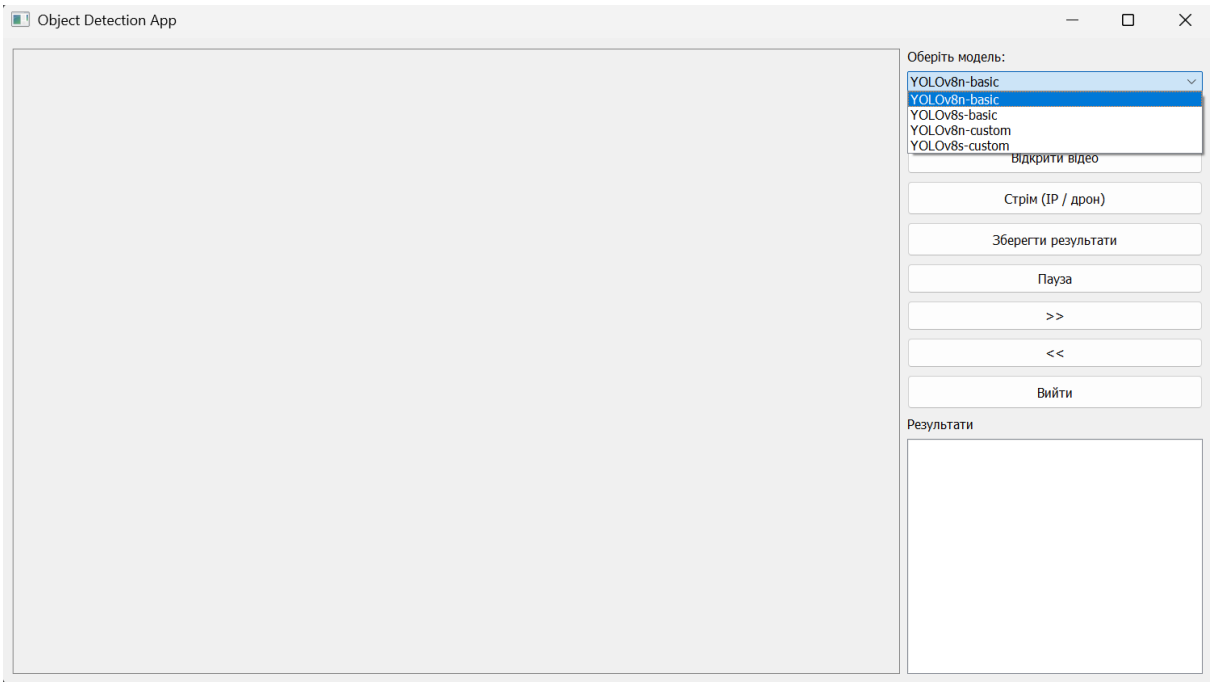

Рисунок 4.19 – Вибір моделі

Далі для розпізнавання об’єктів на зображенні необхідно натиснути кнопку *Відкрити зображення*. Відкриється вікно для вибору шляху до зображення

(рис. 4.20). Обране зображення з’явиться у віджеті у основній частині вікна зліва (рис. 4.21). Його можна масштабувати у цій області за допомогою миші.

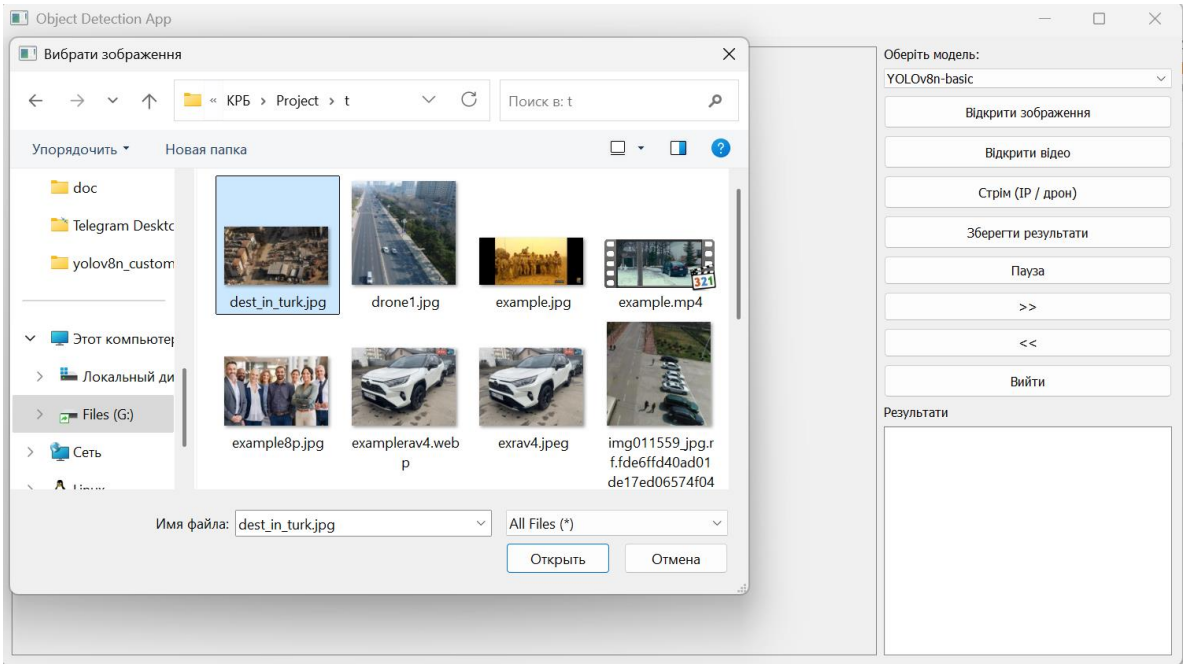

Рисунок 4.20 – Вибір зображення для розпізнавання

Рисунок 4.21 – Вибір зображення для розпізнавання

На зображенні будуть відображені розпізнані об’єкти в рамках із вказівкою класу та *Confidence score* – рівень впевненості, з якою об’єкт до цього класу відноситься. Ці показники виводяться біля об’єктів на віджеті з зображенням та у полі для виведення результатів у правій частині вікна.

У прикладі, наведеному на рисунку 4.21 ми бачимо, що було розпізнано 8 транспортних засобів класу car засобів та 5 людей із класу person.

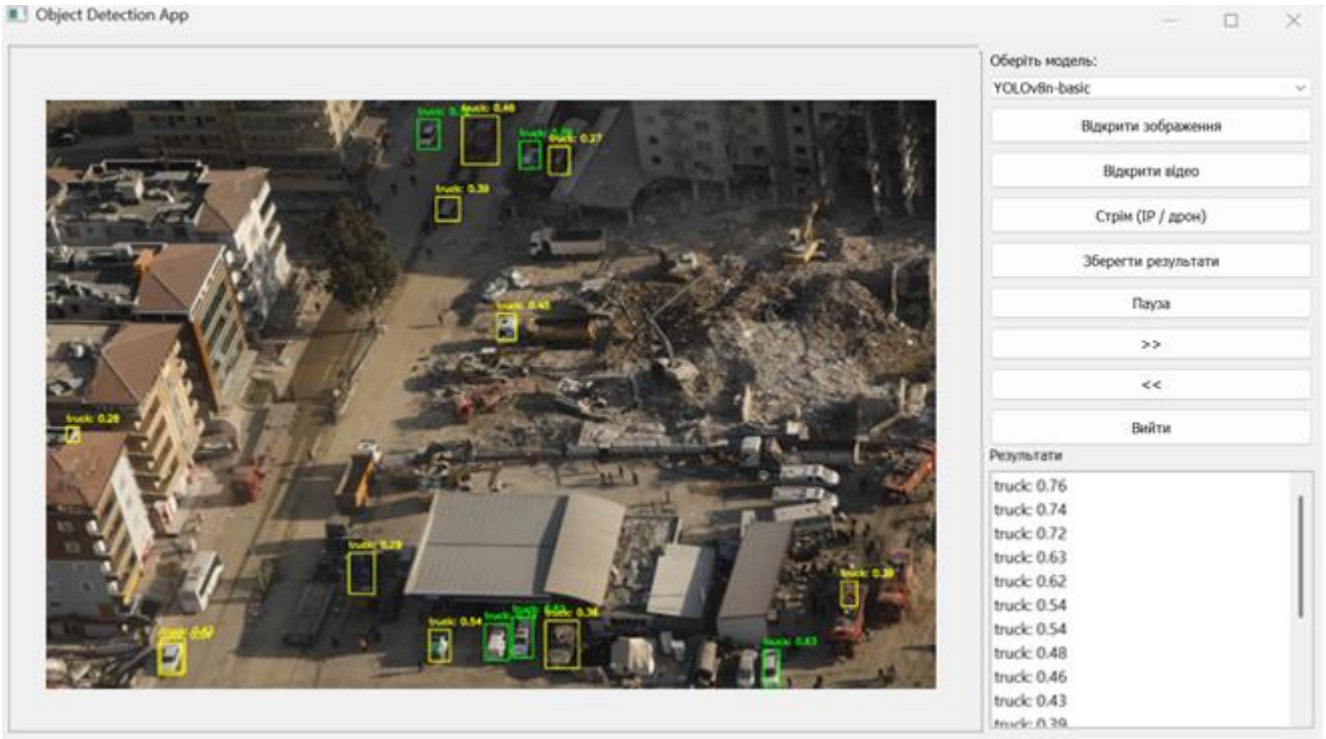

Рисунок 4.22 – Вибір зображення для розпізнавання

Далі для розпізнавання відео на зображенні необхідно натиснути кнопку *Відкрити відео*. Відкриється вікно для вибору шляху до нього, після вибору файлу відео з’явиться у віджеті у основній частині вікна зліва (рис. 4.22). Розпізнані об’єкти для кожного кадра будуть відображені в рамках із вказівкою класу та Confidence score. При зміні кадрів марковані об’єкти будуть зміщуватися, а показник Confidence score буде також змінювати своє значення у залежності від положення об’єкту.

Далі для розпізнавання стріму з БПЛА необхідно натиснути кнопку *Стрім (IP/дрон)*. З'явиться вікно, у якому необхідно вказати джерело й обрати: *Ввести IP-адресу* або *Захоплення екрану*, в залежності від того, яким чином будуть надходити дані від БПЛА (рис. 4.23). Далі кадри стріму будуть відображатися у основній часині вікна із розпізнаними об'єктами в рамках та показниками Confidence score, які будуть змінюватися при зміні кадрів.

Рисунок 4.23 – Вибір джерела надходження стріму

Було проведено тестування розробленого застосунку. В результаті було виявлено як суттєві так і не значні недоліки застосунку. Так, одним з суттєвих недоліків було виявлено що при натисканні кнопки *Відкрити зображення* і спробі завантажити відео, застосунок не міг обробити дію і вимикався (рис. 4.24).

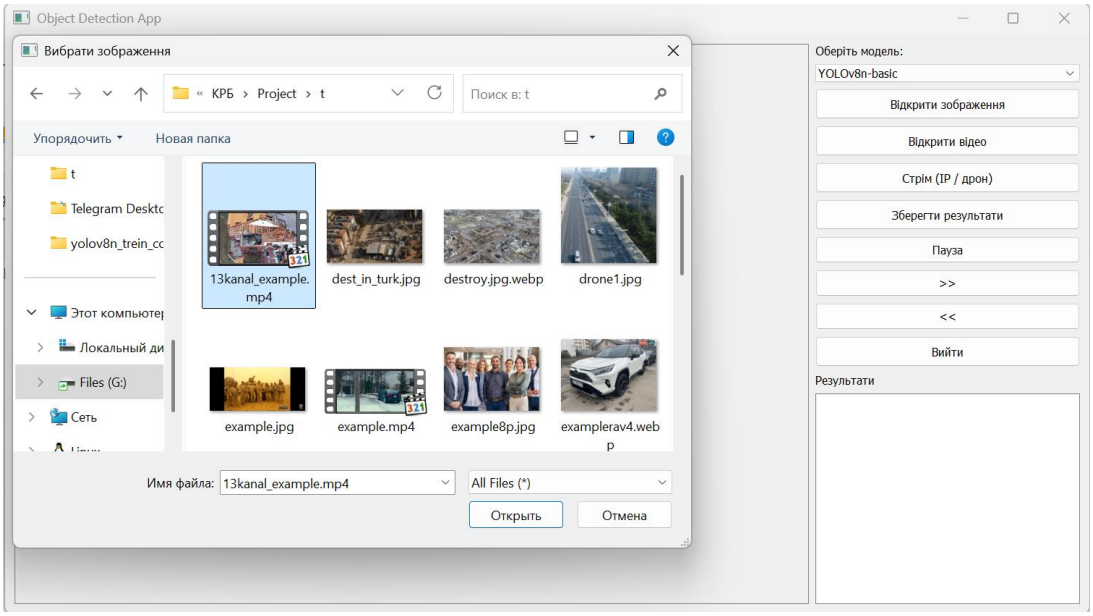

Рисунок 4.24 – Помилки при натисканні кнопки під час тестування

Виявлено, що помилка була обумовлена обранням невірного формату. Цю помилку було оброблено та додано до коду фрагменти, які будуть виводити відповідні повідомлення у застосунку у цьому випадку. Для вирішення цього аспекту було внесено зміни у код методів load_image та load_video, які передбачають перевірку розширення файлу та виведення вікна з повідомленням у випадку невідповідності обраного формату (рис. 4.28, рис. 4.29).

```
135     def load_image(self):  1 usage
136         path, _ = QFileDialog.getOpenFileName(self, "Вибрати зображення")
137         if path:
138             # Перевірка розширення
139             if not any(path.lower().endswith(ext) for ext in [".jpg", ".jpeg", ".png", ".bmp", ".tiff"]):
140                 QMessageBox.warning(self, "Помилка формату", "Файл не є зображенням.")
141                 return
142             image = cv2.imread(path)
143             if image is None:
144                 QMessageBox.critical(self, "Помилка", "Не вдалося завантажити зображення.")
145                 return
146             results = self.model(image)[0]
147             image = self.display_results(image, results)
148             self.image_label.set_image(image)
```

Рисунок 4.28 – Метод load_image із внесеними змінами

```
150     def load_video(self):  1 usage
151         self.video_path, _ = QFileDialog.getOpenFileName(self, "Вибрати відео")
152         if self.video_path:
153             # Перевірка розширення
154             if not any(self.video_path.lower().endswith(ext) for ext in [".mp4", ".avi", ".mov", ".mkv"]):
155                 QMessageBox.warning(self, "Помилка формату", "Файл не є відео.")
156                 return
157             self.cap = cv2.VideoCapture(self.video_path)
158             if not self.cap.isOpened():
159                 QMessageBox.critical(self, "Помилка", "Не вдалося відкрити відеофайл.")
160                 return
161             self.prepare_output()
162             self.timer.start(30)
163             self.start_time = time.time()
```

Рисунок 4.28 – Метод load_video із внесеними змінами

Тепер при обранні не вірного формату користувач побачить повідомлення, приклади якого наведені на рисунках 4.26 та 4.27, а робота застосунку не завершалася.

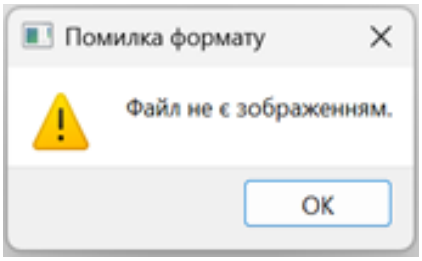

Рисунок 4.26 – Повідомлення про не вірно обраний формат файлу з зображенням

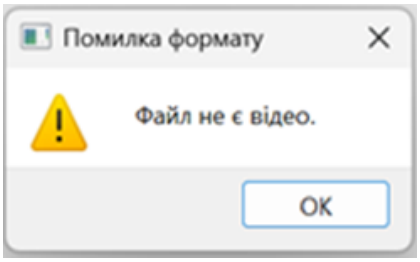

Рисунок 4.27 – Повідомлення про не вірно обраний формат файлу з відео

Також у ході тестування було виявлено незручність реалізації методу захоплення екрану, так як під час цього користувачу необхідно дивитися на зображення що транслюється, а не на те що в застосунку і на якому відбувається розпізнавання. Це може бути покращено в майбутніх ітерація проекту і дозволить полегшити користування застосунком.

Висновки до розділу 4

У четвертому описано процес розробки та програмної реалізації інформаційної системи розпізнавання об’єктів БПЛА на основі згорткових нейронних мереж. Здійснено моделювання інформаційної системи, яке визначає логічну структуру взаємодії між користувачем і системою та її функціональність.

Обґрунтовано вибір інструментальних засобів розробки. Розробка здійснювалася на мові програмування Python 3.12.6. Навчання моделей YOLOv8n і YOLOv8s відбувався у середовищі PyCharm із використанням бібліотеки Ultralytics. Для аналізу та оцінки якості моделей і візуалізації результатів було

використано Python-бібліотеки: pandas, matplotlib, seaborn os і glob. Для реалізації графічного інтерфейсу користувача використано бібліотеку PyQt5. Для обробки зображень і відео та взаємодії з відеопотоками з дронів у проєкті задіяно бібліотеку OpenCV.

У розробленому застосунку для розпізнавання даних, які надходять з БПЛА, реалізовано інтуїтивно зрозумілий інтерфейс, який підтримує підключення до IP-камер або стрімів із дронів через URL або Wi-Fi/RTSP. Це дозволяє застосовувати програму в умовах реального середовища для моніторингу територій, які були зруйновані під час обстрілів для виявлення транспортних засобів та людей. У застосунку реалізована можливість розпізнавання об’єктів на зображеннях, відео та стрімах із збереженням отриманих результатів у вигляді файлів відео чи зображень та у текстовому форматі.

Застосунок інформаційної системи дозволяє ефективно працювати з потоковими джерелами зображення, використовує сучасні моделі комп’ютерного зору, забезпечує збереження результатів у структурованому вигляді та є зручним для подальшого розширення функціоналу.

ВИСНОВКИ

Проведений аналіз предметної області розпізнавання об’єктів у місцях масових руйнувань від обстрілів, з даних, які надходять від БПЛА дозволив установити, що ефективне використання БПЛА у рятувальних операціях потребує надійного програмного супроводу. Здійснений аналіз можливостей існуючих програмних рішень для обробки даних від БПЛА показав, що вини мають обмеження, які не забезпечують ефективність їх застосування при виконанні пошуково-рятувальних операцій у зонах масових руйнувань. В основному вони зосереджені на картографуванні, а не на оперативному виявленні окремих об’єктів, більшість із них є комерційними і не безкоштовними й потребує потужного серверного обладнання для обробки даних та мають обмежену адаптивність до особливих умов зйомки. Існує необхідність розробок, адаптованих до ситуацій, у яких дрони виконують пошуково-рятувальні операції.

Огляд досліджень у сфері розпізнавання показав, що найбільш оптимальними моделями, які інтегрують у програмні засоби для розпізнавання об’єктів, є згорткові нейронні мережі. Виявлено, що найбільш прийнятними для розпізнавання об’єктів є моделі YOLO, сучасні версії яких можуть здійснювати виявлення об'єктів, їх сегментацію і класифікацію як на зображеннях так і на відео та потокових відео. Моделі останніх версій Ultralytics YOLO мають оптимізовані архітектури, забезпечуючи баланс між точністю та швидкодією, що є критичним під час проведення рятувальних операцій.

У розробці інформаційної системи було вирішено використовувати моделі YOLO платформи Ultralytics. Виявлено, що детекція об’єктів базується на пошуку оптимальних б-боксів та класифікації. YOLO реалізує одноетапний алгоритм, який передбачає координати рамок разом із ймовірностями класів, до яких віднесено об’єкти у рамках. Задача детекції вирішується як задача регресії, що обумовлює специфіку функції втрат, яка є комбінацією функцій втрат класифікації та детекції.

Для розробки інформаційної системи було обрано версії моделі YOLOv8 – YOLOv8n та YOLOv8s. У моделі YOLOv8 перетворення зображення відбувається через блоки C2f, які зберігають більшу кількість важливих ознак на різних рівнях глибини мережі. В архітектурі моделі виділяють: Backbone: для вилучення ознак зі зображення, Neck: для об’єднання ознак з різних рівнів абстракції, Head: для формування остаточних передбачень координат рамок і класів об’єктів. Особливістю YOLOv8 є використання розділеного головного блоку (Decoupled Head), де локалізація (визначення координат об’єкта) і класифікація (визначення типу об’єкта: людина, автомобіль) виконуються окремими гілками мережі. Це дозволяє покращити якість обох процесів і мінімізувати взаємний вплив задач локалізації та класифікації.

Для системи розпізнавання об’єктів БПЛА було проведено підготовку спеціалізованого набору даних, що містить зображення із зони масових руйнувань після обстрілів, які належать до п’яти класів: та 4 класи транспортних засобів: car, truck, bus, motorcycle, та клас людей – person. Дані було анотовано та відформатовано у форматі YOLO. При навчанні моделей YOLOv8n і YOLOv8s було реалізовано два ключові підходи: 1) навчання моделей з нуля на новому наборі даних; 2) перенавчання попередньо навчених на наборі даних COCO моделей для розпізнавання об’єктів 5 класів із тестуванням на новому наборі даних.

Детальна оцінка ефективності моделей здійснювалась за основними метриками якості: mAP@0.5, mAP@0.5:0.95, Precision, Recall, а також втратами Loss на етапах навчання та валідації. Отримані результати показали, що навчання з нуля на наборі даних, адаптованому до предметної області, з якої будуть надходити дані, забезпечують кращі показники якості та точності розпізнавання об’єктів. У рамках проєкту було реалізовано обидва підходи для моделей YOLOv8n і YOLOv8s, що дозволило знайти баланс між продуктивністю та точністю, оскільки архітектури версій моделей відрізняються між собою складністю, кількістю параметрів, ресурсомісткістю та точністю.

Обґрунтовано вибір інструментальних засобів розробки. Розробка здійснювалася на мові програмування Python 3.12.6. Навчання моделей YOLOv8n і YOLOv8s відбувався у середовищі PyCharm із використанням бібліотеки Ultralytics. Для аналізу та оцінки якості моделей і візуалізації результатів було використано Python-бібліотеки: pandas, matplotlib, seaborn os і glob. Для реалізації графічного інтерфейсу користувача використано бібліотеку PyQt5. Для обробки зображень і відео та взаємодії з відеопотоками з дронів у проєкті задіяно бібліотеку OpenCV.

У розробленому застосунку для розпізнавання даних, які надходять з БПЛА, реалізовано інтуїтивно зрозумілий інтерфейс, який підтримує підключення до IP-камер або стрімів із дронів через URL або Wi-Fi/RTSP. Реалізована можливість розпізнавання об’єктів на зображеннях, відео та стрімах із збереженням отриманих результатів у вигляді файлів відео чи зображень та у текстовому форматі. Це дозволяє застосовувати програму в умовах реального середовища для моніторингу територій, які були зруйновані під час обстрілів для виявлення транспортних засобів та людей.

Поставлені завдання виконано, однак у подальшому функціонал системи може бути розширено за рахунок додавання каналів надходження даних з БПЛА, інтеграції вдосконалених моделей з більшою точністю розпізнавання та розширення кількості об’єктів, які система може розпізнавати.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кучеренко О. І., Вакалюк Т. А. Огляд технічних та програмних засобів керування БпЛА. *Вісник Херсонського національного технічного університету*. 2024. № 2(89). С. 170–176. URL: https://doi.org/10.35546/ kntu2078-4481.2024.2.24 (дата звернення: 08.05.2025).

2. Особливості розгортання мереж мобільного зв’язку за допомогою БПЛА / А.В. Антоненко та ін. *Таврійський науковий вісник. Серія: технічні науки*. 2024. № 3. С. 3-12. URL: https://doi.org/10.32782/tnvtech.2024.3.1 (дата звернення: 18.04.2025).

3. Rapid Mapping for Emergency Response. *Pix4Dreact*: вебсайт. URL: https://www.pix4d.com/product/pix4dreact (дата звернення: 27.04.2025).

4. Cloud Mapping for UAVs. *DroneDeploy*: вебсайт. URL: https://www.dronedeploy.com (дата звернення: 27.02.2025).

5. AI-Powered Drone Software Platform. *FlytBase*: вебсайт. URL: https://flytbase.com/ (дата звернення: 17.04.2025).

6. DJI Terra – Mapping Software for Aerial Surveying and 3D Reconstruction. *DJI*: вебсайт. URL: https://www.dji.com/dji-terra (дата звернення: 27.04.2025).

7. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed, Springer, 2022. 925 p. DOI: https://doi.org/10.1007/978-3-030-34372-9.

8. OpenCV Documentation. *OpenCV*: вебсайт. URL: https://docs.opencv.org (дата звернення: 12.03.2025).

9. Nair V., Hinton G. Rectified Linear Units Improve Restricted Boltzmann Machines. *In Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010. URL: https://proceedings.mlr.press/v13/nair10a.html (дата звернення: 27.03.2025).

10. Bishop C. Deep Learning: Foundations and Concepts. Springer, 2024. 669 p. DOI: https://doi.org/10.1007/978-3-031-45468-4.

11. Ruder S. An overview of gradient descent optimization algorithms. *ArXiv preprint.* 2016. ArXiv:1609.04747. URL: https://arxiv.org/abs/1609.04747 (дата звернення: 4.04.2025).

12. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. *In Proceedings of the 25th International Conference on Neural Information Processing Systems (NeurIPS)*, 2012. URL: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c4b-Abstract.html (дата звернення: 25.03.2025).

13. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv preprint.* 2016. URL: https://arxiv.org/abs/1506.02640 (дата звернення: 15.04.2025).

14. Wang G., Chen Y., An P., Hong H., Hu J., Huang T. UAV-YOLOv8: A Small-Object-Detection Model Based on Improved YOLOv8 for UAV Aerial Photography Scenarios. *Sensors*. 2023. Vol. 23(16). P. 7190. URL: https://doi.org/10.3390/s23167190 (дата звернення: 10.03.2025).

15. Lin T., Goyal P., Girshick R., He K., Dollár P. Focal Loss for Dense Object Detection. *ArXiv preprint.* 2017. URL: https://arxiv.org/abs/1708.02002 (дата звернення: 12.03.2025).

16. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *ArXiv preprint.* 2016. URL: https://arxiv.org/abs/1506.02640 (дата звернення: 29.03.2025).

17. Terven J., Cordova-Esparta D., Romero-Gonsales J. Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*. 2023. Vol 5(4). P. 1680-1716. DOI: https://doi.org/10.3390/make5040083.

18. YOLOv9 Release Notes. *GitHub: вебсайт*. URL: https://github.com/ultralytics/ultralytics/releases (дата звернення: 07.05.2025).

19. Hussain M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature Toward Digital Manufacturing and Industrial Defect Detection. *Machines*. 2023. Vol. 11(7). P. 677. DOI: https://doi.org/10.3390/machines11070677.

20. Zhou G., Liu M., Wang H., Zheng Y. Power equipment image enhancement processing based on YOLO-v8 target detection model under MSRCR algorithm. *International Journal of Low-Carbon Technologies*. 2024. Volume 19. P. 1717–1724. DOI: https://doi.org/10.1093/ijlct/ctae122.

21. Train YOLOv8 on Custom Data. *Ultralytics: вебсайт*. URL: https://docs.ultralytics.com/modes/train/ (дата звернення: 07.04.2025).

22. YOLOv8 Architecture. *Ultralytics: вебсайт*. URL: https://docs.ultralytics.com/models/yolov8/#architecture (дата звернення: 17.04.2025).

23. Use YOLOv8 with PyTorch. *Ultralytics: вебсайт*. URL: https://docs.ultralytics.com/modes/predict/ (дата звернення: 07.04.2025).

24. Data Augmentation Techniques. *Ultralytics: вебсайт*. URL: https://docs.ultralytics.com/modes/train/#data-augmentation (дата звернення: 07.04.2025).

25. Ultralytics – Export Formats: ONNX, TensorRT, CoreML, TorchScript. *Ultralytics*: вебсайт. URL: https://docs.ultralytics.com/modes/export/ (дата звернення: 13.05.2025).

26. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *ArXiv preprint*. 2016. arXiv:1506.02640. 1 с. URL: https://arxiv.org/abs/1506.02640 (дата звернення: 29.05.2025).

27. Li N., Ye T., Zhou Z., Gao C., Zhang P. Enhanced YOLOv8 with BiFPN-SimAM for Precise Defect Detection in Miniature Capacitors. *Applied Science*. Vol. 14(1). DOI: https://doi.org/10.3390/app14010429.

28. Metrics in Object Detection. *Ultralytics*: вебсайт. URL: https://docs.ultralytics.com/guides/metrics/ (дата звернення: 07.06.2025).

29. Solawetz J. What's is YOLOv8? A Complete Guide. *Roboflow*: вебсайт. URL: https://blog.roboflow.com/what-is-yolov8/.

30. PyTorch: Tensors and Autograd. *PyTorch*: вебсайт. URL: https://pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html (дата звернення: 04.06.2025).

31. Train Custom Model with YOLOv8. *Ultralytics*: вебсайт. URL: https://docs.ultralytics.com/tasks/train-custom-model/ (дата звернення: 04.06.2025).

32. COCO Dataset. *COCO*: вебсайт. URL: https://cocodataset.org (дата звернення: 12.03.2025).

33. YOLO Annotation Format Explained. Roboflow: вебсайт. URL: https://blog.roboflow.com/yolo-object-detection-format/ (дата звернення: 04.06.2025).

34. Dataset YAML Format for YOLOv8. Ultralytics: вебсайт. URL: https://docs.ultralytics.com/datasets/detect/ (дата звернення: 04.06.2025).

35. LabelImg Tool. *GitHub*: вебсайт. URL: https://github.com/tzutalin/labelImg (дата звернення: 10.03.2025).

36. Free Image Labeling Tool. *Annotate.tech*: вебсайт. URL: https://annotate.tech/ (дата звернення: 10.03.2025) (дата звернення: 29.05.2025).

37. Computer Vision Annotation Tool. *CVAT*: вебсайт. URL: https://cvat.org (дата звернення: 10.03.2025).

38. Data Augmentation with YOLOv8. *Ultralytics*: вебсайт. URL: https://docs.ultralytics.com/guides/augmentation/ (дата звернення: 14.04.2025).

39. Annotating & Exporting for YOLO. *Roboflow*: вебсайт. URL: https://roboflow.com (дата звернення: 10.03.2025).

40. Bai H., Shen F., Wang M., Lu J., Zhang Z. Improving Detection Capabilities of YOLOv8-n for Small Objects in Remote Sensing Imagery: Towards Better Precision with Simplified Model Complexity. *Research Square*. 2023. DOI: https://doi.org/10.21203/rs.3.rs-3085871/v1.

41. Matplotlib: Visualization with Python. *Matplotlib*: вебсайт. URL: https://matplotlib.org (дата звернення: 10.03.2025).

42. PyCharm. *JetBrains*: вебсайт. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 12.03.2025).

43. Python 3.12 Documentation. *Python Software Foundation*: вебсайт. URL: https://docs.python.org/3.12/ (дата звернення: 12.03.2025).

44. Pandas: Python Data Analysis Library. *Pandas*: вебсайт. URL: https://pandas.pydata.org (дата звернення: 10.03.2025).

45. Seaborn: Statistical Data Visualization. *Seaborn*: вебсайт. URL: https://seaborn.pydata.org (дата звернення: 10.03.2025).

46. YOLOv8 GitHub Repository. *Ultralytics*: вебсайт. URL: https://github.com/ultralytics/ultralytics (дата звернення: 10.03.2025).

47. Glob – Unix Style pathname pattern expansion. *Python*: вебсайт. URL: https://docs.python.org/3/library/glob.html (дата звернення: 25.03.2025).

48. Qt for Python Documentation. *Qt*: вебсайт. URL: https://doc.qt.io/qtforpython/ (дата звернення: 10.03.2025).

49. OpenCV Documentation. *OpenCV*: вебсайт. URL: https://docs.opencv.org (дата звернення: 10.03.2025).

50. Os — Miscellaneous Operating System Interfaces. *Python*: вебсайт. URL: https://docs.python.org/3/library/os.html (дата звернення: 10.03.2025).