Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

До захисту допущено
Кафедрою комп'ютерних систем та робототехніки
протокол № __ від __ червня 2026 р.

завідувач кафедри _____ Максим ХРУСЛОВ
(підпис)
«___»_____ 2026 р.

Кваліфікаційна робота

здобувача першого (бакалаврського) рівня вищої освіти

«МОДЕЛЬ ІДЕНТИФІКАЦІЇ РУХОМИХ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ»

Спеціальність ***151 – Автоматизація та комп'ютерно-інтегровані технології***
Освітня програма ***Автоматизація та комп'ютерно інтегровані технології***

Виконавець _____ **Дар'я ХАРЧЕНКО**
(підпис)

Науковий керівник _____ **Тетяна БИКОВА**
(підпис)

Харків – 2026

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та чотирьох додатків. Загальний обсяг роботи становить 64 сторінки, з яких 37 сторінок основної частини, що містить 6 рисунків, 3 таблиці, 13 найменувань у списку використаних джерел.

Метою кваліфікаційної роботи є забезпечення ефективного виявлення та ідентифікації рухомих об'єктів на відеоданих, отриманих з безпілотних літальних апаратів, із використанням архітектури YOLOv8.

Об'єкт дослідження – процеси автоматизованого виявлення та ідентифікації об'єктів у відеоданих, отриманих з безпілотних літальних апаратів.

Предмет дослідження – методи та програмні засоби глибокого навчання для задач детекції об'єктів на основі згорткових нейронних мереж, зокрема архітектури YOLOv8.

Проблема полягає у складності забезпечення точного та швидкого виявлення об'єктів у відеоданих з БПЛА в умовах обмежених обчислювальних ресурсів, високої щільності сцен та малих розмірів об'єктів.

Область застосування – системи моніторингу та безпеки, безпілотні літальні апарати, автономні системи спостереження, робототехнічні комплекси, а також задачі аналізу відеопотоків у реальному часі.

Ключові слова: автоматизація та комп'ютерно-інтегровані технології, комп'ютерний зір, YOLOv8, нейронні мережі, детекція об'єктів, безпілотні літальні апарати, відеопотік, глибоке навчання, PyTorch, OpenCV.

ABSTRACT

The explanatory note for the bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and four appendices. The total volume of the work is 64 pages, including 37 pages of main content, 6 figures, 3 tables, and 13 references.

Objective: The objective of the qualification thesis is to ensure effective detection and identification of moving objects in video data obtained from unmanned aerial vehicles using the YOLOv8 architecture.

Object of research: The processes of automated detection and identification of objects in UAV-acquired video data.

Subject of research: Methods and software tools of deep learning for object detection based on convolutional neural networks, in particular the YOLOv8 architecture.

Problem statement: The research addresses the challenge of achieving accurate and real-time object detection in UAV video data under conditions of limited computational resources, high scene density, and small object dimensions.

Scope of application: Monitoring and security systems, unmanned aerial vehicles, autonomous surveillance systems, robotic systems, and real-time video stream analysis applications.

Keywords: automation and computer-integrated technologies, computer vision, YOLOv8, neural networks, object detection, unmanned aerial vehicles, video stream, deep learning, PyTorch, OpenCV.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ……5
ВСТУП ……7
РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ ТА ІДЕНТИФІКАЦІЇ ОБ'ЄКТІВ……9
1.1 Основні поняття комп'ютерного зору……9
1.2 Сучасні методи виявлення та ідентифікації об'єктів……11
1.3 Особливості обробки відеоданих з БПЛА……14
1.4 Вибір моделі для детекції об'єктів……16
Висновки за розділом 1……18
РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ ІДЕНТИФІКАЦІЇ ОБ'ЄКТІВ……19
2.1 Архітектура програмної системи……19
2.2 Алгоритм підготовки та конвертації даних датасету VisDrone……21
2.3 Конфігурування навчання та проектування модуля інференсу системи……23
2.4 Реалізація навчання моделі та алгоритму ідентифікації об’єктів……26
Висновки за розділом 2……28
РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МОДЕЛІ……29
3.1 Порівняльний аналіз варіантів YOLOv8 за показниками продуктивності……29
3.2 Аналіз результатів навчання та продуктивності моделі……31
3.3 Оцінка точності та швидкодії моделі……35
Висновки за розділом 3……36
ВИСНОВКИ……38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……40
ДОДАТКИ……42

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

БПЛА – безпілотний літальний апарат;

AMP – режим автоматичної змішаної точності обчислень (Automatic Mixed Precision);

API – інтерфейс програмування застосунків (Application Programming Interface);

BBox – обмежувальна рамка об’єкта (Bounding Box);

CNN – згорткова нейронна мережа (Convolutional Neural Network);

COCO – набір даних Common Objects in Context;

CPU – центральний процесор (Central Processing Unit);

CUDA – платформа паралельних обчислень NVIDIA Compute Unified Device Architecture;

FPS – кількість кадрів за секунду (Frames Per Second);

GPU – графічний процесор (Graphics Processing Unit);

IoU – коефіцієнт перекриття обмежувальних рамок (Intersection over Union);

mAP – середня точність детекції (Mean Average Precision);

mAP50 – середня точність при порозі IoU = 0,5;

mAP50-95 – середня точність при порогах IoU від 0,5 до 0,95;

MixUp – метод аугментації даних шляхом змішування двох зображень та їхніх анотацій;

Mosaic – метод аугментації даних шляхом об’єднання чотирьох зображень в одне;

NMS – алгоритм придушення дубльованих детекцій (Non-Maximum Suppression);

OpenCV – бібліотека алгоритмів комп'ютерного зору Open Source Computer Vision Library;

PyTorch – фреймворк для розробки та навчання нейронних мереж;

Recall – повнота виявлення об'єктів;

SGD – алгоритм стохастичного градієнтного спуску (Stochastic Gradient Descent);

VisDrone – набір даних для детекції об'єктів з безпілотних літальних апаратів;

YOLO – модель детекції об'єктів You Only Look Once;

YOLOv8 – восьма версія архітектури YOLO;

YOLOv8n – компактна модифікація моделі YOLOv8 Nano.

ВСТУП

Безпілотні літальні апарати з кожним роком стають невід'ємною частиною сучасних технологічних рішень у цивільній та оборонній галузях. Їхнє застосування охоплює галузь критичної інфраструктури, логістичне забезпечення, екологічний контроль, ліквідацію наслідків надзвичайних ситуацій та забезпечення правопорядку. Вбудовані відеокамери надвисокої роздільної здатності генерують потужні потоки зображень, тому необхідність автоматизованої ідентифікації об'єктів на борту БПЛА набуває стратегічного значення для своєчасного реагування в мінливому оточенні.

Актуальність роботи. Аналіз відеоматеріалів, отриманих з повітряних носіїв, має свою специфіку, тобто об'єкти маленькі, фон складний, камера рухається. Ключовою складністю є незначний розмір більшості цільових об'єктів, які при типових висотах польоту займають малу частку пікселів кадру. Класичні алгоритми обробки зображень погано справляються з такими умовами через недостатню збереженість просторових ознак. Натомість нейромережеві детектори покоління YOLO, і зокрема архітектура YOLOv8, завдяки вдосконаленій структурі та сучасним процедурам оптимізації дозволяють одночасно досягати високої точності та продуктивності, бо затримка навіть у кілька секунд робить систему непрактичною[4].

Метою кваліфікаційної роботи є забезпечення ефективного виявлення та ідентифікації рухомих об’єктів на відеоданих, отриманих з безпілотних літальних апаратів, із використанням архітектури YOLOv8.

Об’єкт дослідження – процеси автоматизованого виявлення та ідентифікації об’єктів у відеопотоках, отриманих з безпілотних літальних апаратів.

Предмет дослідження – нейромережеві моделі детекції об’єктів на основі архітектури YOLOv8 та методи їх навчання для аналізу аерофотознімків і відеоданих.

Завдання дослідження:

1. Проаналізувати сучасні методи виявлення та ідентифікації об’єктів у системах комп’ютерного зору.
2. Дослідити особливості обробки відеоданих з безпілотних літальних апаратів.
3. Обрати та обґрунтувати модель для детекції об’єктів.
4. Розробити програмну модель ідентифікації об’єктів.
5. Провести тестування моделі та оцінити її ефективність за показниками точності та швидкодії.

Методи дослідження. У роботі використано методи глибокого навчання та штучних нейронних мереж для розпізнавання об’єктів, методи цифрової обробки зображень для підготовки та аугментації даних, а також методи математичної статистики для оцінювання ефективності моделі.

Практична значущість полягає у створенні програмної системи для автоматизованої ідентифікації наземних об’єктів (людей і транспортних засобів) на відеоданих або статичних зображеннях з БПЛА. Розроблене програмне забезпечення може бути використане для задач моніторингу дорожнього руху, охорони територій, пошуково-рятувальних операцій та інших сфер, де необхідний оперативний аналіз даних аерозйомки.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ ТА ІДЕНТИФІКАЦІЇ ОБ'ЄКТІВ

1.1 Основні поняття комп'ютерного зору

Комп'ютерний зір сьогодні є одним із найбільш швидко зростаючих напрямів у галузі штучного інтелекту[12]. Головна ідея, що лежить в його основі, полягає у наданні обчислювальним системам здатності сприймати, аналізувати та інтерпретувати візуальну інформацію так, як це притаманно людині. Поширенню таких систем сприяв прогрес у сфері обчислювальних потужностей та алгоритмів машинного навчання, що відкрило можливості для їх практичного застосування в промисловості, медицині, транспорті, робототехніці та безпілотних платформах.

У загальному розумінні комп'ютерний зір охоплює сукупність алгоритмічних підходів, що забезпечують автоматичне вилучення корисної інформації з цифрових зображень або відеопотоків[12]. Центральним завданням таких систем є пошук закономірностей, структур та об'єктів у візуальних даних без участі людини.

Однією з ключових переваг сучасних систем комп'ютерного зору є здатність опрацьовувати значні обсяги візуальних даних у темпі, наближеному до реального часу[2]. Ця властивість є особливо важливою для автономних транспортних засобів, інтелектуальних систем спостереження та безпілотних літальних апаратів, де затримка у прийнятті рішень безпосередньо позначається на результативності всієї системи.

Залежно від характеру поставленої задачі комп'ютерний зір вирішує кілька принципово різних типів проблем.

Найпростішим рівнем є класифікація зображень — система встановлює, до якого з наперед визначених класів належить вхідне зображення, наприклад

визначає, чи містить кадр автомобіль або пішохода. Результатом є одна або кілька категоріальних міток без інформації про розташування об'єкта.

Складнішою задачею є локалізація об'єктів, яка передбачає не лише встановлення класу, але й визначення просторового положення об'єкта на зображенні. Для цього, як правило, використовуються прямокутні обмежувальні рамки, що описують координати об'єкта у площині зображення.

Найбільш поширеною в практичних застосуваннях є задача детекції об'єктів, яка поєднує класифікацію та локалізацію і дозволяє одночасно знаходити на одному кадрі множину різнорідних об'єктів. Результатом є набір обмежувальних рамок із зазначенням класу кожного об'єкта та рівня впевненості моделі. Саме ця задача є центральною в даній роботі[12].

Окремим напрямом є сегментація зображень, при якій межі об'єктів визначаються не прямокутником, а точною піксельною маскою. Такий підхід застосовується, зокрема, в медичній діагностиці, дистанційному зондуванні Землі та системах автономного водіння.

Основу сучасних систем комп'ютерного зору складають глибокі нейронні мережі, серед яких особливе місце займають згорткові архітектури (CNN)[6]. На відміну від класичних підходів, що потребували ручного формування ознак, такі мережі здатні самостійно навчатися виділяти інформативні характеристики із зображень, що суттєво підвищує їх ефективність на різноманітних задачах.

У контексті безпілотних літальних апаратів технології комп'ютерного зору набувають особливого значення. Поєднання камер високої роздільної здатності з сучасними алгоритмами детекції дозволяє автоматично виявляти транспортні засоби, людей та елементи інфраструктури на місцевості, що відкриває широкі можливості для контролю дорожнього руху, охорони об'єктів, пошуково-рятувальних операцій та розвідувальних місій.

1.2 Сучасні методи виявлення та ідентифікації об'єктів

Задача виявлення та розпізнавання об'єктів належить до ключових напрямів комп'ютерного зору. Її метою є не лише визначення наявності об'єкта на зображенні або у відеопотоці, а й встановлення його просторового розташування та класової належності. Результативність розв'язання цієї задачі безпосередньо впливає на роботу систем відеоспостереження, робототехнічних комплексів, безпілотних літальних апаратів і транспортних засобів з елементами автономного керування.

У розвитку методів детекції можна виділити два основні напрями: класичні алгоритми, що використовують попередньо створені ознаки, та сучасні підходи, засновані на глибоких нейронних мережах.

Класичні методи

До широкого поширення глибокого навчання розпізнавання об'єктів здійснювалося за рахунок аналізу спеціально сформованих дескрипторів, які характеризували форму, текстуру або зміни яскравості на зображенні. Отримані ознаки надалі використовувалися класифікаторами для прийняття рішення щодо належності об'єкта до певного класу.

Одним із перших успішних рішень став алгоритм Віоли–Джонса, який використовує ознаки Хаара[10]. Для прискорення обробки застосовується інтегральне подання зображення, а відбір потенційних областей здійснюється за допомогою каскадної структури класифікаторів. Завдяки цьому метод забезпечує високу швидкість роботи навіть на малопотужних пристроях. Разом із тим його ефективність помітно знижується при складному освітленні, зміні орієнтації об'єкта або наявності насиченого фону.

Ще одним відомим підходом є комбінація дескриптора HOG та класифікатора SVM[1]. У цьому випадку аналізуються напрямки локальних градієнтів, які формують вектор ознак для подальшої класифікації. Такий метод характеризується кращою стійкістю до змін освітлення та довгий час активно

використовувався для виявлення пішоходів. Проте необхідність аналізу великої кількості областей робить його недостатньо швидким для багатьох систем реального часу.

Головним недоліком класичних алгоритмів є залежність від вручну розроблених ознак. У складних сценах або за умов, що відрізняються від навчальних, це часто призводить до зниження якості детекції.

Методи глибокого навчання

Поява згорткових нейронних мереж суттєво змінила підходи до аналізу зображень. На відміну від традиційних алгоритмів, такі моделі самостійно формують інформативні ознаки без необхідності їх попереднього проєктування людиною.

Початком нового етапу розвитку стала архітектура R-CNN[6]. Її робота передбачала пошук потенційних областей за допомогою Selective Search, подальше вилучення ознак згортковою мережею та остаточну класифікацію знайдених фрагментів. Незважаючи на високу якість результатів, значна кількість окремих проходів через мережу призводила до великих витрат часу на обробку.

З метою підвищення продуктивності було створено Fast R-CNN. На відміну від попередньої реалізації, згорткові операції виконувалися лише один раз для всього зображення, після чого необхідні ознаки вилучалися зі спільної карти активацій. Такий підхід дозволив значно скоротити час роботи системи[6].

Наступним кроком стала поява Faster R-CNN, у якій генерація областей-кандидатів була реалізована за допомогою спеціальної мережі RPN. Завдяки інтеграції цього механізму процес детекції став повністю нейромережевим, що позитивно вплинуло як на швидкодію, так і на точність[6]. Проте двоетапна структура архітектури все ще потребує значних обчислювальних ресурсів.

Інший напрям розвитку представлений одноетапними детекторами. Одним із перших рішень цього типу став SSD, де локалізація та класифікація виконуються одночасно під час одного проходу через мережу. Для роботи з об'єктами різних розмірів використовуються карти ознак декількох масштабів. Це забезпечує високу швидкість роботи, хоча ефективність виявлення дуже дрібних об'єктів може бути нижчою порівняно з двоетапними моделями.

Особливе місце серед одноетапних алгоритмів займає сімейство YOLO. На відміну від попередніх підходів, тут процес пошуку об'єктів розглядається як задача прямого прогнозування координат і класів без проміжних етапів генерації областей. Завдяки цьому досягається висока швидкість обробки кадрів, що є критично важливим для систем реального часу.

Упродовж розвитку архітектури було запропоновано декілька поколінь YOLO, кожне з яких покращувало баланс між точністю та продуктивністю. Однією з найсучасніших реалізацій є YOLOv8, яка використовує вдосконалені механізми вилучення ознак і сучасні стратегії навчання. Це дозволяє ефективно виявляти об'єкти різних масштабів та підтримувати стабільну роботу навіть за високих вимог до швидкодії[3,4].

Таблиця 1.1.

Порівняльний аналіз розглянутих методів

Метод	**Точність**	**Швидкодія**	**Робота в режимі реального часу**
Haar Cascade	Низька	Висока	Так
HOG + SVM	Середня	Середня	Частково
R-CNN	Висока	Низька	Ні
Fast R-CNN	Висока	Середня	Частково
Faster R-CNN	Дуже висока	Середня	Частково
SSD	Висока	Висока	Так
YOLOv8	Висока	Дуже висока	Так

Узагальнена порівняльна таблиця 1.1, сформована на основі аналізу наукових публікацій і оглядових робіт у сфері комп'ютерного зору[1,3,4,6,10]. Порівняння розглянутих підходів свідчить, що класичні алгоритми поступаються сучасним нейромережевим рішенням за точністю та адаптивністю. Серед сучасних архітектур найбільш збалансованим рішенням для задач реального часу є сімейство YOLO, яке поєднує високу швидкість роботи та достатню точність детекції.

1.3 Особливості обробки відеоданих з безпілотних літальних апаратів

Широке впровадження безпілотних літальних апаратів у практику моніторингу територій, інспекції інфраструктури та проведення рятувальних операцій зумовило потребу в ефективних засобах автоматизованої обробки аерофотоданих[13]. Відеоматеріали, отримані з борту БПЛА, мають низку характерних особливостей, що принципово відрізняють їх від знімків, зроблених наземними камерами зі стаціонарних позицій, і суттєво ускладнюють задачу автоматичного розпізнавання об'єктів.

Першою і найбільш очевидною особливістю є постійна рухомість платформи. Під час польоту позиція камери безперервно змінюється відносно сцени, що спричиняє варіації кута огляду, масштабу та взаємного розташування об'єктів у кадрі. Навіть незначні коливання висоти або кута нахилу носія можуть суттєво змінювати зовнішній вигляд об'єктів і ускладнювати їх автоматичне розпізнавання.

Другою характерною рисою є широкий діапазон масштабів об'єктів. Один і той самий об'єкт, залежно від висоти зйомки, може займати від кількох до сотень пікселів у кадрі. Наприклад, автомобіль, знятий з малої висоти, займає помітну частку кадру, тоді як при підйомі на більшу висоту він перетворюється на майже непомітну точку. Це створює серйозні труднощі для алгоритмів, які не мають механізмів роботи з багатомасштабними ознаками.

Ще однією суттєвою проблемою є висока щільність об'єктів у сцені. При аерозйомці міських районів на одному кадрі можуть одночасно перебувати сотні транспортних засобів і пішоходів, що вимагає від алгоритму здатності точно розмежовувати близько розташовані та частково перекриті об'єкти[2,11].

Додаткову складність вносить різноманітність підстилаючої поверхні. Аерознімки охоплюють дороги, будівлі, рослинність, промислові зони та відкриті простори, кожен з яких формує власний складний фон, що збільшує ризик хибних спрацьовувань алгоритму.

Серйозний вплив на якість матеріалу чинять умови зйомки, які можуть суттєво варіюватися залежно від часу доби, стану атмосфери та наявності тіней. Туман, дощ, пил або нерівномірне освітлення знижують контрастність зображення та погіршують видимість дрібних деталей.

Окремою проблемою залишається виявлення малорозмірних об'єктів. Пішоходи, велосипедисти та мотоциклісти нерідко займають лише декілька десятків пікселів, що є критичним обмеженням для традиційних детекторів[12]. Розв'язання цієї задачі вимагає використання нейромережевих архітектур з явним механізмом обробки багатомасштабних ознак.

Нарешті, для переважної більшості практичних застосувань принципово важливою є швидкодія. Системи безпеки, дорожнього моніторингу та автоматизованого управління платформою вимагають формування результатів аналізу в темпі, що не поступається частоті кадрів вхідного відеопотоку. Тому алгоритм повинен забезпечувати не лише достатню точність, але й прийнятний час обробки кожного кадру.

Саме сукупність перелічених вимог визначила вибір архітектури YOLOv8 як основи для розробленої системи: вона поєднує здатність до багатомасштабного аналізу, стійкість до варіацій умов зйомки та достатню швидкодію для роботи в реальному часі.

1.4 Вибір моделі для детекції об'єктів

Ефективність системи автоматичного виявлення об'єктів значною мірою залежить від правильно обраної архітектури нейронної мережі. Під час обробки відеопотоку, отриманого з безпілотного літального апарата, необхідно враховувати одночасно дві вимоги: достатню точність розпізнавання та можливість швидкої обробки великої кількості кадрів.

Сучасні алгоритми детекції об'єктів умовно поділяють на дві категорії. До першої належать двоетапні моделі. Робота таких систем складається з формування потенційних областей розташування об'єктів та їх подальшої класифікації. Подібна схема забезпечує високі показники точності, однак потребує значних обчислювальних ресурсів.

Інший підхід реалізовано в одноетапних детекторах, до яких належать SSD, RetinaNet та різні версії YOLO. У цих моделях пошук об'єктів і визначення їх класу виконуються одночасно в межах одного проходу даних через мережу. Завдяки цьому вдається досягти значно вищої швидкості роботи, що є важливим для практичних систем моніторингу та відеоаналізу.

Таблиця 1.2

Порівняльний аналіз найбільш поширених моделей

Модель	**Тип**	**Особливості**	**Сфера застосування**
Faster R-CNN	Двоетапна (Two-stage)	Висока якість локалізації, значні обчислювальні витрати	Аналіз статичних зображень
SSD	Одноетапна (One-stage)	Оптимальне співвідношення швидкості та точності	Системи реального часу
RetinaNet	Одноетапна	Використання функції втрат Focal Loss	Складні сцени з дисбалансом класів
YOLOv5	Одноетапна	Компактна архітектура та швидке виконання	Вбудовані та edge-пристрої
YOLOv8	Одноетапна	Покращена структура мережі та висока продуктивність	БПЛА та системи відеоаналізу

Аналіз було проведено на основі узагальнення даних наукових джерел [3,4,6,12].

Порівняння з таблиці 1.2 розглянутих моделей показало, що для поставленої задачі найбільш доцільним є застосування архітектури YOLOv8. Дана модель належить до останніх поколінь сімейства YOLO та характеризується вдосконаленими механізмами обробки ознак, що позитивно впливають як на швидкість роботи, так і на якість детекції.

Особливе значення для аналізу аерофотознімків має здатність моделі працювати з об'єктами різного масштабу[4]. У відеоматеріалах з БПЛА розміри цілей можуть суттєво змінюватися залежно від висоти польоту, тому підтримка багатомасштабного аналізу є важливою вимогою до детектора.

Додатковою перевагою YOLOv8 є ефективне використання графічних процесорів із підтримкою технології CUDA[3]. Це дозволяє скоротити час виконання обчислень і забезпечити обробку відеопотоку без помітних затримок.

У межах роботи використовується модифікація YOLOv8n (Nano), яка є найменшою за розміром серед моделей даного сімейства. Незважаючи на компактність та невелику кількість параметрів, вона демонструє достатню якість розпізнавання для задач моніторингу територій і водночас потребує мінімальних апаратних ресурсів.

Вибір саме YOLOv8n пояснюється такими перевагами:

- висока швидкість опрацювання відеокадрів;
- можливість функціонування в режимі реального часу;
- підтримка сучасних графічних прискорювачів;
- ефективне виявлення об'єктів невеликого розміру;
- зручність інтеграції в програмні комплекси комп'ютерного зору;
- відкритість програмного коду та наявність детальної документації[3,9].

Висновки за розділом 1

У межах першого розділу було досліджено сучасні підходи до виявлення та розпізнавання об'єктів у системах комп'ютерного зору. Проведений огляд показав тенденцію переходу від традиційних методів, що базуються на використанні попередньо визначених ознак, до нейромережевих моделей, здатних автоматично формувати інформативні представлення даних під час навчання.

Порівняння сучасних архітектур детекторів дозволило визначити YOLOv8 як найбільш придатне рішення для поставленої задачі. Серед розглянутих варіантів саме ця модель демонструє вдале поєднання швидкодії та якості виявлення об'єктів, що є важливим для систем, орієнтованих на роботу в режимі реального часу.

Результати проведеного дослідження слугують теоретичною основою для розробки та практичної реалізації системи детекції об'єктів, опис якої наведено в наступному розділі.

РОЗДІЛ 2

РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ ІДЕНТИФІКАЦІЇ ОБ'ЄКТІВ

2.1 Архітектура програмної системи

Під час розробки програмного забезпечення для аналізу відеоданих, отриманих із безпілотного літального апарата, особливу увагу необхідно приділити вибору програмних засобів. Обране середовище повинно забезпечувати швидке виконання обчислень, коректну роботу з потоковими даними та можливість використання апаратного прискорення. Це особливо актуально для систем, у яких обробка кадрів здійснюється безпосередньо під час надходження відеопотоку.

Мова програмування Python

Базою програмної реалізації виступає мова Python. Її популярність у сфері штучного інтелекту та комп'ютерного зору пояснюється широким набором спеціалізованих інструментів і готових програмних модулів. Крім того, Python дозволяє швидко інтегрувати бібліотеки машинного навчання та скорочує час розробки програмних рішень[8,12].

Фреймворк PyTorch

Для виконання операцій, пов'язаних із нейронними мережами, використовується PyTorch. Даний фреймворк надає засоби для створення, навчання та запуску моделей глибокого навчання. Однією з його переваг є можливість перенесення обчислень на графічний процесор за допомогою технології CUDA, що суттєво скорочує час обробки даних та підвищує продуктивність системи[8].

Бібліотека Ultralytics

Робота з моделлю YOLOv8 реалізується через програмний пакет Ultralytics. Цей інструмент містить готові механізми для підготовки вхідних даних, запуску інференсу та формування кінцевих результатів детекції. Значна частина службових операцій виконується автоматично, що дозволяє зосередитися безпосередньо на логіці програмного застосунку та зменшує кількість допоміжного коду[3].

Однією з причин вибору модифікації YOLOv8n є її орієнтація на використання в системах із обмеженими ресурсами. Компактна структура моделі дає можливість застосовувати її не лише на потужних робочих станціях, а й на бортових обчислювальних платформах безпілотних комплексів. Додатковою перевагою є сумісність із графічними адаптерами NVIDIA, що підтримують CUDA[6].

Для прискорення роботи системи можуть використовуватися методи оптимізації обчислень. Одним із таких підходів є застосування змішаної точності (mixed precision), коли частина операцій виконується з меншою розрядністю. Це сприяє зменшенню використання відеопам'яті та дозволяє збільшити швидкість інференсу без суттєвого погіршення результатів розпізнавання[8].

Окрему роль у програмному комплексі відіграє бібліотека OpenCV[7]. Вона забезпечує взаємодію з відеоджерелом, підготовку кадрів до аналізу та відображення отриманих результатів. Саме OpenCV поєднує підсистему отримання зображень із модулем нейромережевої обробки.

У розробленому програмному рішенні бібліотека OpenCV використовується для:

• отримання кадрів із камери або іншого джерела відеосигналу за допомогою механізму VideoCapture;

• перетворення отриманих даних у формат, придатний для подальшої обробки нейронною мережею;

• зміни розміру та попередньої підготовки зображень;

• виведення відеопотоку з результатами аналізу на екран користувача;

• побудови рамок локалізації об'єктів, відображення назв класів та значень достовірності прогнозу.

Спільне використання OpenCV, PyTorch і бібліотеки Ultralytics формує єдиний цикл обробки відеоданих: отримання кадру, його підготовка, виконання детекції та відображення результату. Така організація роботи дозволяє реалізувати систему комп'ютерного зору, здатну функціонувати в режимі реального часу[3,7,8].

2.2 Алгоритм підготовки та конвертації даних датасету VisDrone

Ефективність навчання нейронних мереж значною мірою залежить від якості та коректності підготовки вхідних даних, а також їх відповідності вимогам архітектури детектора. У даній роботі використано спеціалізований набір даних для аерозйомки VisDrone-DET2019[11], який містить 6471 зображення для навчання, 548 для валідації та 1610 для тестування.

Набір даних VisDrone був обраний через його наближеність до реальних умов роботи безпілотних літальних апаратів. На відміну від загальних датасетів, таких як COCO[6], зображення VisDrone характеризуються високою щільністю об’єктів та значною кількістю дрібних цілей, що ускладнює задачу детекції.

Більшість об’єктів у кадрах мають малий розмір, що вимагає підвищеної точності моделі під час локалізації та класифікації. Крім того, сцени містять перекриття об’єктів, різні рівні видимості та складні умови зйомки[11,13].

Оригінальні анотації набору даних VisDrone представлені у текстових файлах, де кожен рядок містить інформацію про об’єкт у форматі:

[left, top, width, height, score, object_category, truncation, occlusion]

Архітектура YOLOv8 використовує інший формат представлення даних — нормалізовані координати центру bounding box. Для використання у моделі YOLOv8 необхідно привести дані до стандартного формату, що включає нормалізовані координати центру bounding box та його розміри:

[class_id, x_center, y_center, width, height]

Для приведення даних до необхідного формату було розроблено алгоритм конвертації, який виконує наступні перетворення:

1. Обчислення центру bounding box:

$$x_{center} = left + \frac{width}{2}$$

$$y_{center} = top + \frac{height}{2}$$

2. Нормалізація координат відносно розміру зображення:

$$x_{norm_center} = \frac{x_{center}}{W}$$

$$y_{norm_center} = \frac{y_{center}}{H}$$

$$wight_{norm} = \frac{width}{W}$$

$$height_{norm} = \frac{height}{H}$$

,де W та H — ширина і висота зображення відповідно.

3. Відбір класів:

У наборі даних VisDrone присутні додаткові категорії, такі як “ignored regions” та “others”, які не використовуються для навчання[11]. У процесі підготовки даних вони були виключені. Залишено 10 цільових класів, які відповідають основним об’єктам міського середовища (пішоходи, автомобілі, автобуси, вантажівки, велосипеди та інші транспортні засоби). Оскільки у VisDrone класи починаються з 1, а YOLO використовує індексацію з 0, виконується зсув:

$$class_{YOLO} = class_{VisDrone} - 1$$

Для забезпечення сумісності з бібліотекою Ultralytics було створено конфігураційний файл visdrone.yaml, який містить:

- шляхи до навчальної, валідаційної та тестової вибірок;
- кількість класів (nc = 10);
- перелік назв класів.

2.3 Конфігурування навчання та проектування модуля інференсу системи

Для адаптації базової архітектури YOLOv8n до задачі детекції об’єктів на відеоданих з безпілотних літальних апаратів було розроблено програмний модуль навчання, що забезпечує контроль процесу тренування, налаштування гіперпараметрів та подальше тестування моделі.

Процес навчання базується на використанні попередньо навчених ваг (pre-trained weights), отриманих на наборі даних COCO[6]. Такий підхід дозволяє виконати transfer learning, що значно прискорює збіжність моделі та підвищує якість детекції на спеціалізованому авіаційному домені.

У рамках експериментальної реалізації було використано оптимізований набір гіперпараметрів, підібраний з урахуванням специфіки малих об’єктів на аерозображеннях та обмежених обчислювальних ресурсів GPU[8]:

- Оптимізатор: AdamW
- Початкова швидкість навчання (lr0): 0.001
- Weight decay: 0.0005
- Batch augmentation (MixUp): 0.05
- Mosaic augmentation: 0.8
- Flipping (fliplr): 0.5
- Колірні трансформації:
  - hsv_h = 0.015;
  - hsv_s = 0.7;
  - hsv_v = 0.4;
- Early stopping (patience): 20 епох
- AMP (Automatic Mixed Precision): увімкнено
- Workers (паралельна обробка даних): 2

Застосування оптимізатора AdamW дозволяє стабілізувати навчання при роботі з шумними градієнтами, що є характерним для щільних сцен із великою кількістю дрібних об’єктів. Низьке значення learning rate (0.001) забезпечує більш плавну адаптацію ваг моделі, зменшуючи ризик перенавчання[8].

Аугментаційні техніки відіграють ключову роль у підвищенні узагальнюючої здатності моделі. Зокрема, Mosaic (0.8) дозволяє моделі навчатися на композиційно складних сценах, які імітують реальні умови зйомки з БПЛА, де об’єкти можуть з’являтися у різних масштабах та положеннях. MixUp (0.05) використовується як додатковий регуляризатор, що покращує стійкість до перекриття об’єктів[3,8].

Колірні аугментації (HSV-перетворення) моделюють зміни освітлення, які часто виникають під час польоту через зміну висоти, кута зйомки та погодних умов[3,8].

Для реалізації режиму реального часу було розроблено програмний модуль інференсу, який забезпечує безперервну обробку відеопотоку та виконання детекції об’єктів у режимі наближеному до реального часу.

Модуль підтримує три основні сценарії роботи:

- обробка відеопотоку з веб-камери або камери БПЛА;
- аналіз попередньо записаних відеофайлів (MP4, AVI);
- пакетна обробка наборів зображень.

Архітектура модуля побудована за принципом розділення відповідальностей і включає:

1. Модуль ініціалізації моделі

Завантажує навчену модель YOLOv8n (best.pt) та виконує підготовку до інференсу з використанням GPU (CUDA), що забезпечує мінімальну затримку при обробці кадрів[3,8].

2. Модуль обробки кадру (preprocess + inference + postprocess)

Кожен кадр проходить стандартний конвеєр:

- нормалізація та зміна розміру до 640×640;
- подача на вхід нейромережі;
- застосування non-maximum suppression (NMS) для усунення дублюючих bounding box’ів.

3. Модуль візуалізації результатів

На вихідне зображення накладаються:

- bounding boxes;
- назви класів (pedestrian, car, bus тощо);
- значення confidence score.

4. Модуль оцінки продуктивності

Додатково фіксується час інференсу на кадр та частота обробки (FPS), що дозволяє оцінити придатність системи для роботи в реальному часі.

2.4 Реалізація навчання моделі та алгоритму ідентифікації об'єктів

Реалізація програмної системи виявлення об'єктів на відеоданих з безпілотних літальних апаратів включає два взаємопов'язані етапи: навчання нейронної моделі та її подальше використання в режимі інференсу. У межах даної роботи ці етапи реалізовані як єдиний програмний конвеєр із можливістю повторного навчання та тестування моделі.

Навчання моделі YOLOv8n здійснюється із використанням бібліотеки Ultralytics, яка надає високорівневий API для організації повного циклу тренування[3,8].

Запуск процесу навчання виконується за допомогою командного рядка: «yolo detect train data=configs/visdrone.yaml model=yolov8n.pt imgsz=640 batch=8 device=0 workers=2»

У процесі навчання система автоматично виконує:

- завантаження попередньо навчених ваг (transfer learning на COCO);
- формування батчів із тренувального набору даних VisDrone;
- застосування аугментацій (mosaic, mixup, HSV-перетворення, flipping);
- пряме та зворотне поширення похибки (forward/backpropagation);
- оновлення параметрів моделі за допомогою оптимізатора AdamW;
- валідацію моделі на контрольній вибірці після кожної епохи.

Результати навчання автоматично зберігаються у директорії runs/detect/train-*, де основними файлами є:

- best.pt — модель із найкращими метриками mAP;
- last.pt — модель після останньої епохи навчання;
- графіки втрат та метрик (precision, recall, mAP).

Конфігурація датасету задається у файлі visdrone.yaml, який визначає шляхи до тренувальної, валідаційної та тестової вибірок, а також перелік класів об'єктів.

Після завершення навчання модель використовується для детекції об'єктів у режимі реального часу. Основою реалізації є модуль інференсу, який базується на бібліотеці PyTorch + OpenCV + Ultralytics YOLO[3,7,8].

Система підтримує три режими роботи:

- обробка відеопотоку з веб-камери або камери БПЛА;
- аналіз відеофайлів (MP4, AVI);
- пакетна обробка зображень.

Алгоритм роботи системи в режимі інференсу включає такі етапи:

1. Захоплення кадру

   Відеопотік зчитується за допомогою OpenCV (cv2.VideoCapture).

2. Попередня обробка

   Кадр приводиться до необхідного розміру (640×640) та нормалізується для подачі в нейронну мережу.

3. Інференс моделі

   Обробка кадру нейронною мережею YOLOv8n з використанням GPU (CUDA).

4. Постобробка результатів

   Застосовується алгоритм Non-Maximum Suppression (NMS) для усунення дублюючих bounding box'ів та фільтрація за порогом впевненості (confidence threshold)[3].

5. Візуалізація результатів

   На кадр накладаються: bounding boxes, назви класів та значення confidence score.

6. Вивід результату

   Оброблений кадр відображається в реальному часі або зберігається у відеофайл.

Програмна реалізація системи побудована модульно та включає наступні компоненти:

- train.py — модуль навчання моделі;
- detector.py — клас реалізації детекції (UAVObjectDetector);

- run_detection.py — модуль запуску інференсу;
- evaluate.py — модуль оцінки метрик якості;
- convert.py — конвертація анотацій датасету;
- visdrone.yaml — конфігурація датасету;

Висновки за розділом 2

У другому розділі виконано практичну реалізацію системи автоматичного виявлення об'єктів на відеоматеріалах, отриманих із безпілотних літальних апаратів. Увагу було зосереджено на побудові програмного рішення, здатного працювати з аерознімками та відеопотоками в умовах реального часу.

Для забезпечення коректної роботи нейронної мережі підготовлено набір даних на основі датасету VisDrone. Зокрема, реалізовано програмний модуль перетворення вихідних анотацій у формат, придатний для використання моделями сімейства YOLO. Крім того, сформовано структуру вибірок та налаштовано конфігураційні параметри, необхідні для процесу навчання і тестування.

Під час побудови моделі використано архітектуру YOLOv8n. Її налаштування виконувалося із застосуванням перенесення знань із попередньо навчених ваг, що дозволило скоротити час навчання та покращити якість розпізнавання. Для підвищення стійкості моделі до різноманітних умов зйомки використовувалися методи штучного розширення набору даних, серед яких комбінування зображень, геометричні перетворення та модифікація колірних характеристик.

Результатом виконаної роботи стала функціональна система детекції об'єктів, придатна для роботи з відеоданими, зображеннями та в режимі реального часу. Запропонований підхід може бути використаний як основа для створення програмних комплексів моніторингу, спостереження та аналізу обстановки із застосуванням безпілотних літальних апаратів.

РОЗДІЛ 3

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МОДЕЛІ

3.1 Порівняльний аналіз варіантів YOLOv8 за показниками продуктивності

Експериментальне дослідження проводилося з метою комплексного порівняльного аналізу різних варіантів архітектури YOLOv8 для визначення оптимальної конфігурації, придатної до розгортання в умовах бортових обчислювальних систем безпілотних літальних апаратів. Оцінювання ефективності моделей здійснювалося за трьома основними критеріями: точність детекції, швидкодія інференсу та апаратні витрати, зокрема обсяг пам'яті, необхідний для зберігання ваг моделі[3,8].

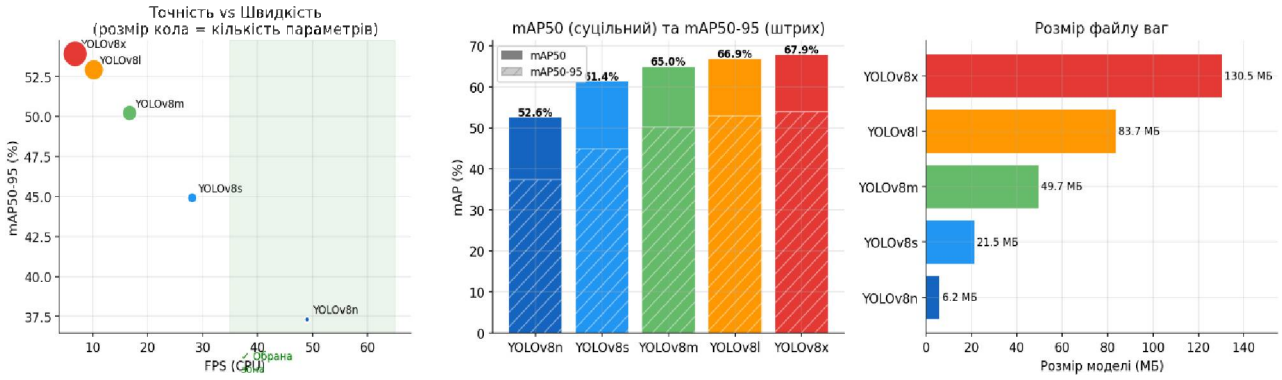


Рисунок 3.1 – Порівняльний аналіз варіантів моделі YOLOv8.

На рисунку 3.1 представлено результати порівняльного аналізу варіантів архітектури YOLOv8 (n, s, m, l, x), отриманих у процесі навчання та валідації на еталонному наборі даних COCO val2017[6]. Отримані результати використовуються як базова орієнтирна оцінка продуктивності перед подальшою адаптацією моделі до специфічного аерофотознімального набору VisDrone[11].

Аналіз точності та швидкодії моделей. Результати експериментів демонструють чітко виражений компроміс між складністю архітектури нейронної мережі та її обчислювальною ефективністю.

YOLOv8n (Nano) характеризується найвищою швидкодією серед усіх розглянутих варіантів і забезпечує інференс до ~50 FPS на центральному процесорі та ще вищі значення на GPU. При цьому середній показник точності mAP становить 52.6%, що є найнижчим серед моделей сімейства, однак достатнім для задач реального часу, де критичним параметром є швидкість реагування системи.

YOLOv8s (Small) забезпечує помітне підвищення точності до 61.4% mAP, однак супроводжується зниженням швидкодії до приблизно 28 FPS. Це обмежує її використання у сценаріях, де необхідна стабільна обробка відеопотоку в реальному часі на легких обчислювальних платформах.

Важчі моделі (YOLOv8m, YOLOv8l, YOLOv8x) демонструють подальше зростання точності (до 67.9% mAP для YOLOv8x), однак їхня продуктивність знижується до діапазону 7–17 FPS, що є недостатнім для задач динамічного відеоспостереження з БПЛА, де затримка обробки кадрів безпосередньо впливає на якість детекції рухомих об’єктів.

Апаратні витрати та обсяг моделі

Окремим критерієм оцінки є розмір ваг моделі, який визначає придатність для використання на бортових системах з обмеженим обсягом оперативної пам’яті.

1. YOLOv8n має мінімальний розмір — близько 6.2 МБ, що забезпечує швидке завантаження моделі в пам’ять та низьке енергоспоживання.
2. YOLOv8s збільшує обсяг приблизно утричі — до 21.5 МБ, що вже частково обмежує використання на ультракомпактних пристроях.
3. YOLOv8x досягає 130.5 МБ, що у поєднанні з низькою швидкодією робить її непридатною для використання на легких безпілотних платформах.

Таблиця 3.1

Порівняння продуктивності моделей сімейства YOLOv8

Варіант моделі	mAP 50, %	mAP 50-95, %	Розмір ваг,МБ	Швидкість (FPS)
YOLOv8n	52,6	37,3	6,2	50
YOLOv8s	61,4	44,9	21,5	28
YOLOv8m	65,0	50,2	49,7	17
YOLOv8l	66,9	52,9	83,7	11
YOLOv8x	67,9	53,9	130,5	8

3.2 Аналіз результатів навчання та продуктивності моделі

На графіках кривих навчання представлено зміну основних показників протягом тренувального процесу.

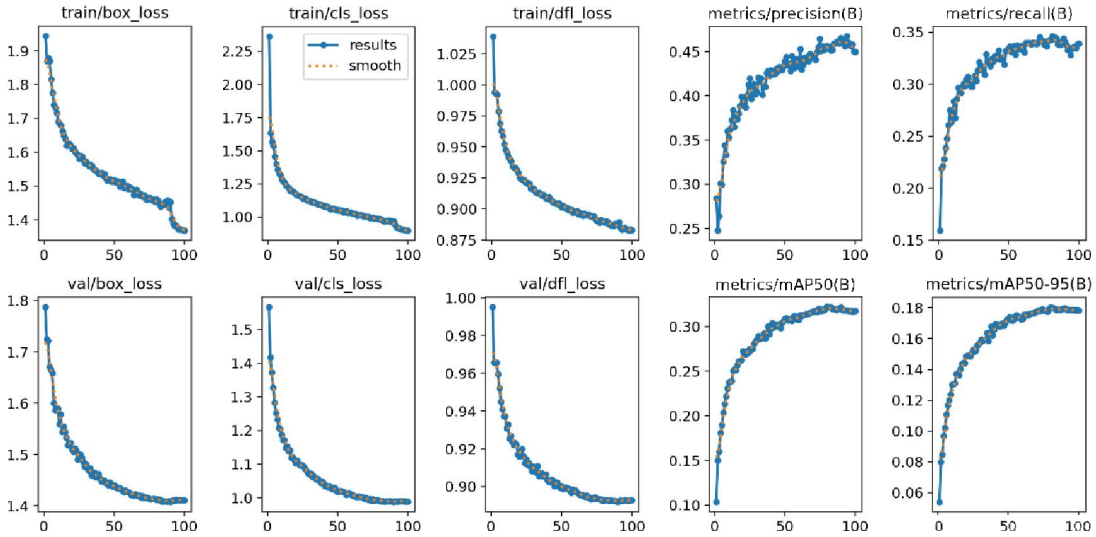


Рисунок 3.2 – Аналіз динаміки навчання та збіжності моделі.

Аналіз функцій втрат. Для моніторингу процесу навчання та оцінки здатності моделі до узагальнення було проаналізовано динаміку трьох основних компонентів втрат:

1. Box Loss (похибка локалізації): Графіки train/box_loss та val/box_loss демонструють стабільне зниження (від 1.94 до 1.36 на тренувальній вибірці), що свідчить про успішне навчання мережі точно фіксувати координати обмежувальних рамок навіть для дрібних об’єктів.

2. Class Loss (похибка класифікації): Показник train/cls_loss плавно спадає протягом усіх епох, підтверджуючи ефективність розрізнення 10 цільових категорій VisDrone.Відсутність розбіжності між тренувальною та валідаційною кривими класифікації вказує на коректний добір параметрів оптимізації.
3. DFL Loss (Distribution Focal Loss): Постійне зниження цього показника свідчить про успішне уточнення імовірнісного розподілу меж об'єктів, що є критично важливим для ідентифікації цілей в умовах нечітких контурів або оклюзій, характерних для аерозйомки.

Характер кривих на валідаційних даних (val/loss) повністю повторює тенденцію тренувальних графіків без висхідних трендів («uptick») на фінальних етапах. Це є математичним підтвердженням відсутності явища перенавчання (overfitting) та високої надійності розробленої моделі при роботі з новими даними.Успішна конвергенція всіх функцій втрат дозволила моделі вилучити стійкі семантичні ознаки об'єктів, мінімізуючи вплив динамічного фону та его-руху БПЛА.

Метрики точності. Аналіз динаміки навчання показав, що показник mAP@0.5 (середня точність) демонструє стрімке зростання протягом перших 20 епох навчання, після чого швидкість конвергенції поступово знижується і метрика стабілізується ближче до фінальної стадії процесу.

Згідно з результатами, зафіксованими у звітному файлі results.csv, пікове значення точності для ваг моделі best.pt (80-та епоха) становить 32,2% (0.3224). Матриця плутанини дозволяє детально проаналізувати якість ідентифікації для кожного окремого класу.

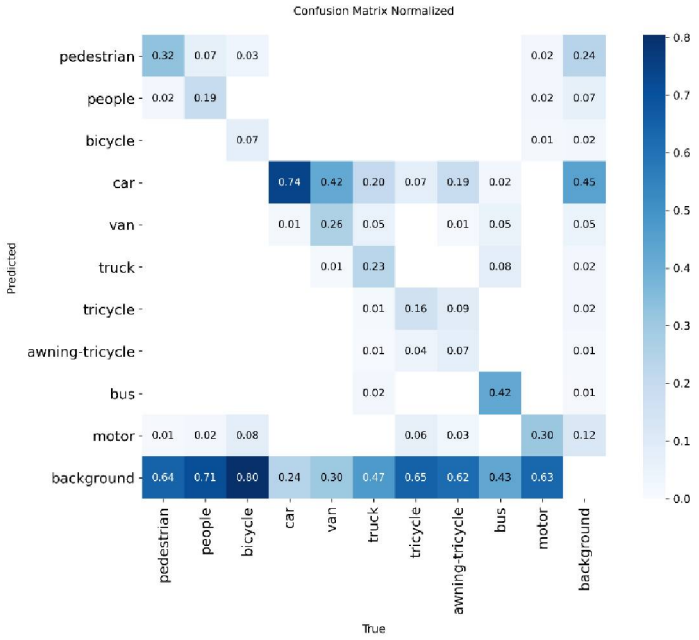


Рисунок 3.3 – Матриця плутанини для класів.

Найвища точність: Модель демонструє найкращі результати для класу «Car» (Автомобіль) із показником впевненості 0.74.

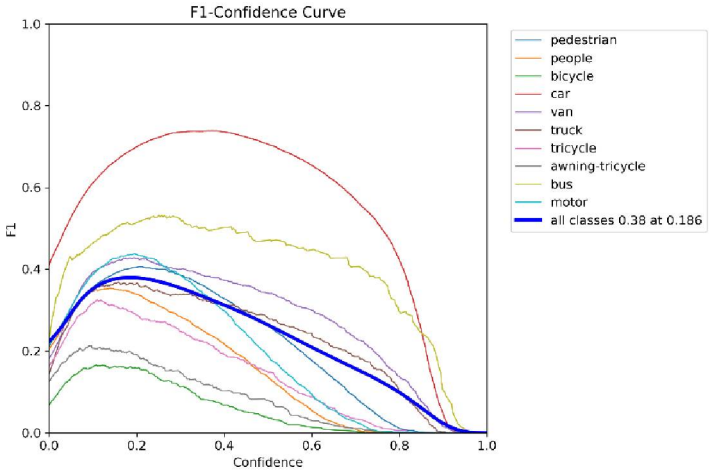


Рисунок 3.4 – Крива залежності F1-метрики від порогу впевненості.

F1-крива відображає оптимальний баланс між точністю (precision) та повнотою (recall), що дозволяє визначити найефективніше значення порогу confidence для роботи моделі в реальному часі. Згідно з результатами експериментів, оптимальне значення порогу становить приблизно 0.4.

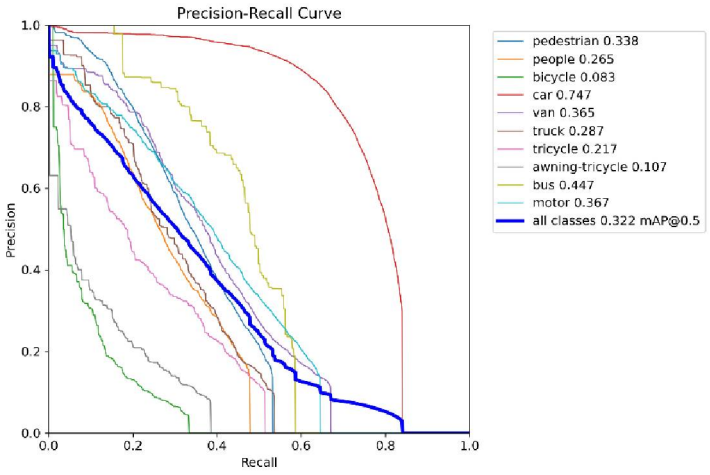


Рисунок 3.5 – Криві залежності прецизії від повноти детектування.

PR-крива (Precision–Recall) характеризує здатність моделі правильно розпізнавати об’єкти при зміні рівня впевненості. Аналіз показує, що найбільшу площу під кривою (AP) мають класи «Car», «Bus» та «Truck», що пов’язано з їхніми чіткими візуальними ознаками та достатньою кількістю навчальних прикладів. Водночас класи «Bicycle» та «Awning-tricycle» є більш складними для детекції через їхній малий розмір та часткові перекриття в сцені.

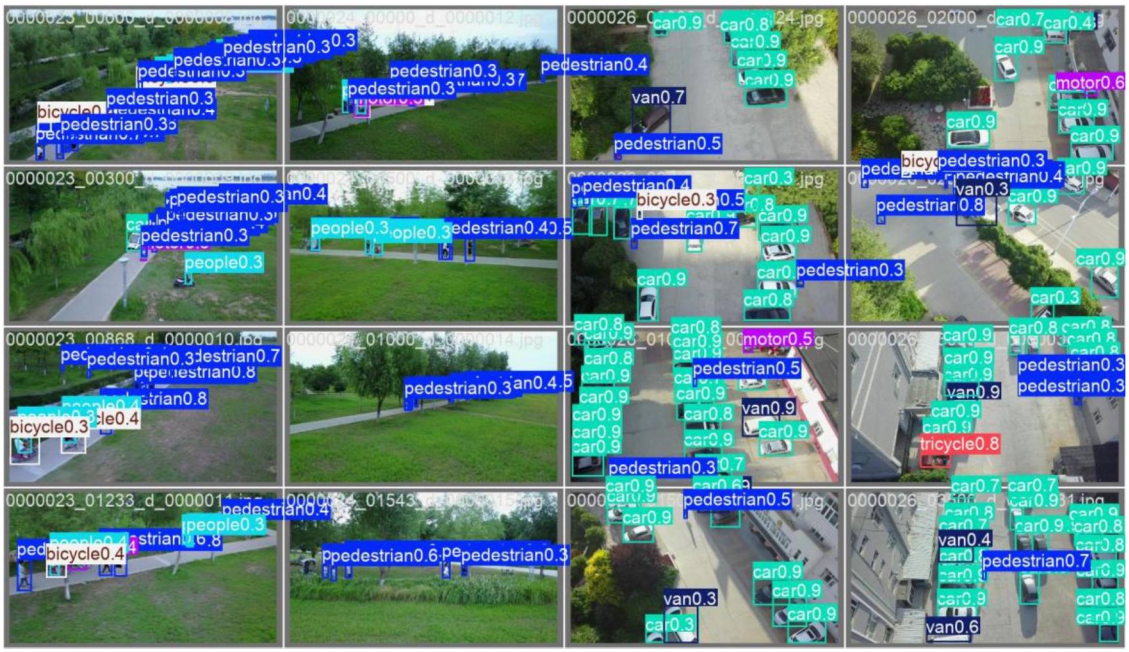


Рисунок 3.6 – Візуалізація результатів детектування.

Результат: Модель успішно ідентифікує та маркує об'єкти обмежувальними рамками (bounding boxes) з ідентифікаторами класів та показниками впевненості.

Специфіка детектування в складних умовах: Аналіз візуальних результатів показує, що модель успішно локалізує об'єкти навіть у зонах із надвисокою щільністю цілей, таких як міські перехрестя або стоянки транспорту.

Система демонструє високу стійкість до взаємних оклюзій (часткового перекриття об'єктів один одним) та здатність стабільно утримувати рамки на дрібних цілях (пішоходах та велосипедах), які на аерознімках часто зливаються з текстурою дорожнього покриття.

Це підтверджує ефективність обраної без'якірної архітектури YOLOv8n та застосованої стратегії аугментації Mosaic, які дозволяють моделі точно ідентифікувати межі об'єктів незалежно від їхнього просторового розташування та ступеня скупченості у кадрі.

3.3 Оцінка точності та швидкодії моделі

У даному підрозділі наведено оцінку ефективності розробленої моделі на основі архітектури YOLOv8 з точки зору точності детекції об’єктів та швидкодії обробки відеопотоку[3,4]. Дослідження виконувалося на основі набору даних VisDrone-DET2019, що містить складні аерозображення з великою кількістю дрібних та частково перекритих об’єктів[11].

Під час навчання моделі використовувалися стандартні метрики: precision, recall, mAP@0.5 та mAP@0.5:0.95.

Отримані результати для best.pt становлять:

- Precision: 0.4566
- Recall: 0.3426
- mAP@0.5: 0.3224
- mAP@0.5:0.95: 0.1804

Аналіз результатів показує, що модель краще виявляє об’єкти з вираженими візуальними ознаками та достатньою кількістю прикладів у

навчальній вибірці. Зниження точності спостерігається для дрібних та частково перекритих об’єктів, що є типовою проблемою для аерозйомних даних VisDrone.

Отримані значення mAP свідчать про середній рівень якості детекції, що є типовим результатом для легкої версії моделі YOLOv8n при роботі зі складними сценами.

Аналіз функцій втрат

На 80-й епосі спостерігається стабілізація функцій втрат:

- Box Loss (train/val): 1.4619 / 1.4124
- Class Loss (train/val): 0.9847 / 0.9899
- DFL Loss: ~0.89

Отримані значення свідчать про збіжність моделі та стабільний процес навчання без суттєвих ознак перенавчання.

Швидкодія моделі оцінювалася експериментально під час обробки відеопотоку. Середні результати становлять:

- Середній FPS: 50.98
- Середній час інференсу: 29.38 ms
- Максимальний FPS: 70.30
- Мінімальний FPS: 0.61

Отримані експериментальні результати свідчать про ефективність розробленої моделі на основі архітектури YOLOv8 для задач виявлення та ідентифікації рухомих об’єктів у відеоданих з безпілотних літальних апаратів. Значення метрик точності у поєднанні зі стабільною швидкодією підтверджують можливість її використання в режимі реального часу.

Висновки за розділом 3

У межах третього розділу основну увагу було зосереджено на практичній перевірці роботи розробленого рішення для детекції об'єктів у відеопотоці з

БПЛА. Дослідження виконувалося для кількох конфігурацій YOLOv8, що дало змогу оцінити вплив складності моделі на її поведінку в реальних умовах.

Отримані результати показали, що підвищення обчислювальної складності моделі не завжди є виправданим, оскільки приріст точності супроводжується помітним зниженням швидкодії.

Під час навчання модель демонструвала стабільну поведінку, без ознак деградації якості на валідаційній вибірці. Значення метрик precision, recall та mAP підтвердили, що система здатна коректно працювати навіть у складних сценах із великою кількістю дрібних та частково перекритих об'єктів.

Окремо була оцінена швидкодія запропонованого рішення. Встановлено, що система забезпечує близько 50 кадрів за секунду при середньому часі обробки одного зображення на рівні ~29 мс, що дозволяє використовувати її у сценаріях, наближених до реального часу.

Узагальнюючи результати всіх етапів дослідження, можна відзначити, що поставлена мета — створення підходу до ефективного виявлення та ідентифікації рухомих об'єктів на відеоданих з використанням архітектури YOLOv8 — вважається досягнутою.

ВИСНОВКИ

У даній кваліфікаційній роботі було виконано комплексне дослідження та розробку програмної моделі для автоматичної ідентифікації об'єктів на відеоданих з безпілотних літальних апаратів із використанням сучасних методів комп'ютерного зору та глибокого навчання.

1. Проаналізовано сучасні методи виявлення та ідентифікації об'єктів у системах комп'ютерного зору. Розглянуто основні підходи двоетапної та одноетапної детекції, їхні основні характеристики, особливості архітектури та сфери застосування.. Встановлено, що одноетапні моделі, зокрема сімейство YOLO, є найбільш придатними для задач реального часу завдяки високій швидкодії та достатньому рівню точності.
2. Досліджено особливості обробки відеоданих з безпілотних літальних апаратів. Визначено, що такі дані характеризуються малими розмірами об'єктів, високою щільністю сцен, наявністю часткових перекриттів та зміною ракурсу зйомки. Це вимагає використання ефективних моделей, здатних працювати в умовах реального часу та обмежених обчислювальних ресурсів.
3. Обрано та обґрунтовано модель для детекції об'єктів. Показано, що YOLOv8 є оптимальним вибором для поставленої задачі завдяки поєднанню швидкодії та точності. Для практичної реалізації використано легку версію YOLOv8n, яка забезпечує мінімальні апаратні вимоги та стабільну роботу в режимі реального часу.
4. Розроблено програмну модель ідентифікації об'єктів. Реалізовано повний програмний конвеєр, який включає підготовку та конвертацію даних у формат YOLO, налаштування процесу навчання, застосування аугментацій, а також модуль інференсу для обробки відеопотоку в режимі реального часу. Забезпечено підтримку різних джерел

відеоданих, включаючи веб-камеру та відеофайли, а також візуалізацію результатів детекції.

5. Проведено тестування моделі та оцінено її ефективність. Результати експериментів підтвердили стабільну роботу системи в складних умовах аерозйомки. Отримані значення метрик Precision, Recall та mAP свідчать про достатній рівень точності моделі. Додатково встановлено, що система забезпечує необхідну швидкодію для застосування в режимі реального часу.

Апробацію результатів дослідження було здійснено шляхом участі у XI Міжнародній науково-технічній конференції «Інтелектуальні технології у міждисциплінарних дослідженнях (ІТМД-2025)», яка проходила 12–14 листопада 2025 року в м. Харків. Представлення доповіді на науковому заході підтвердило їх актуальність і практичну значущість у сфері комп’ютерного зору та систем обробки відеоданих.

Отже, всі поставлені завдання виконано повністю. Розроблена система може бути використана для автоматичної ідентифікації об’єктів у відеоданих з безпілотних літальних апаратів та є основою для подальшого розвитку інтелектуальних систем моніторингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dalal N., Triggs B. Histograms of Oriented Gradients for Human Detection // Proceedings of CVPR. 2005. P. 886–893.
2. Du D., Qi Y., Yu H. et al. The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking // European Conference on Computer Vision (ECCV). 2018. P. 370–386.
3. Jocher G., Chaurasia A., Qiu J. Ultralytics YOLOv8 [Електронний ресурс]. – Режим доступу: https://github.com/ultralytics/ultralytics (дата звернення: 27.03.2026).
4. Liu C.-M., Juang J.-C. Systematic Evaluation of YOLOv8 Variants for UAV-Based Object Detection. Applied Sciences. 2026. Vol. 16, No. 7. P. 3559.
5. Liu S., Li X., Lu H., He Y. Multi-Object Tracking Meets Moving UAV // Proceedings of IEEE/CVF CVPR. 2022. P. 8876–8885.
6. Lin T.-Y., Maire M., Belongie S. et al. Microsoft COCO: Common Objects in Context // ECCV. 2014.
7. OpenCV documentation [Електронний ресурс]. – Режим доступу: https://docs.opencv.org (дата звернення: 27.03.2026).
8. Paszke A., Gross S., Massa F. et al. PyTorch: an imperative style, high-performance deep learning library // Advances in Neural Information Processing Systems (NeurIPS). 2019. – Режим доступу: https://pytorch.org (дата звернення: 27.03.2026).
9. Ultralytics YOLOv8 documentation [Електронний ресурс]. – Режим доступу: https://docs.ultralytics.com (дата звернення: 27.03.2026).
10. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). 2001. P. 511–518.

11. VisDrone-DET2019 dataset. AISKYEYE Team, Tianjin University [Електронний ресурс]. – Режим доступу: https://github.com/VisDrone/VisDrone-Dataset (дата звернення: 27.03.2026).
12. Wu X., Li W., Hong D., Tao R., Du Q. Deep Learning for UAV-based Object Detection and Tracking: A Survey. IEEE Geoscience and Remote Sensing Magazine. 2021.
13. Zhu P., Wen L., Du D. et al. Detection and Tracking Meet Drones Challenge // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021. Vol. 44, No. 11. P. 7380–7399.

ДОДАТКИ

Додаток А

Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних
систем та робототехніки
Максим ХРУСЛОВ
_____
(підпис)
«13» жовтня 2025року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувача першого (бакалаврського) рівня вищої освіти
(першого (бакалаврського) / другого (магістерського))

ХАРЧЕНКО ДАР'Ї В'ЯЧЕСЛАВІВНИ
(прізвище, ім'я, по батькові здобувача)

Спеціальність: 151 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма «Комп'ютеризовані системи управління та автоматика»
1. Тема роботи **МОДЕЛЬ ІДЕНТИФІКАЦІЇ РУХОМИХ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ**
керівник роботи **Бикова Тетяна Володимирівна, к.т.н., доцент зво кафедри,**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від ***№ 4101-5/1245 від 28.05.2026 року.***
2. Строк подання студентом роботи ***30 травня 2026 року***
3. Вихідні дані до роботи:
   1) методи виявлення та ідентифікації об'єктів у відеоданих з безпілотних літальних апаратів;
   2) програмні засоби для розробки, моделювання та тестування алгоритмів комп'ютерного зору;
   3) моделі глибокого навчання для детекції об'єктів (зокрема YOLO);
   4) критерії оцінювання ефективності моделі;
   5) обчислювальна система для розробки та експериментального дослідження програмного забезпечення.
4. Перелік питань, які потрібно розробити:
   1) Проаналізувати сучасні методи виявлення та ідентифікації об'єктів у системах комп'ютерного зору.
   2) Дослідити особливості обробки відеоданих з безпілотних літальних апаратів..
   3) Обрати та обґрунтувати модель для детекції об'єктів.
   4) Розробити програмну модель (алгоритм) ідентифікації об'єктів.
   5) Провести тестування моделі та оцінити її ефективність (точність, швидкодія).

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1.	Затвердження теми кваліфікаційної роботи та узгодження індивідуального плану	02.10.2025 – 13.10.2025
2.	Літературний огляд з проблематики комп'ютерного зору та виявлення об'єктів у відеоданих	13.10.2025 – 20.10.2025
3.	Огляд сучасних методів виявлення та розпізнавання об'єктів у системах штучного інтелекту (зокрема моделей типу YOLO)	21.10.2025 – 30.10.2025
4.	Аналіз особливостей обробки відеоданих, отриманих з безпілотних літальних апаратів	30.10.2025 – 10.11.2025
5.	Аналіз та вибір методів виявлення рухомих об'єктів	11.11.2025 – 20.11.2025
6.	Розробка концепції програмної моделі виявлення об'єктів для БПЛА	20.11.2025 – 24.12.2025
7.	Проєктування структурної схеми програмної системи та алгоритму роботи моделі	10.01.2026 – 01.02.2026
8.	Програмна реалізація моделі виявлення об'єктів	02.02.2026 – 30.04.2026
9.	Тестування розробленої моделі на відеоданих та збір експериментальних результатів	01.04.2026 – 30.04.2026
10.	Аналіз результатів тестування та оцінка ефективності моделі (точність, швидкодія)	01.05.2026 – 13.05.2026
11.	Підготовка та оформлення пояснювальної записки кваліфікаційної роботи відповідно до вимог	01.05.2026 – 30.05.2026
12.	Оформлення звіту про переддипломну практику	01.05.2026 – 30.05.2026
13.	Представлення кваліфікаційної роботи керівнику та рецензенту	30.05.2026

5. Дата видачі завдання ***13 жовтня 2025 року.***

Здобувач вищої освіти _____ Дар'я ХАРЧЕНКО
(підпис)

Керівник роботи _____ Тетяна БИКОВА
(підпис)

Додаток Б

Затверджую

_____

«_____»_____2026 р.

Технічне завдання
на розробку програмного виробу «МОДЕЛЬ ІДЕНТИФІКАЦІЇ РУХОМИХ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ**»**

1.	Введення	1.1. Назва: Модель ідентифікації рухомих об'єктів для безпілотних літальних апаратів. 1.2. Галузь застосування: комп'ютерні технології, системи комп'ютерного зору та обробки відеоданих.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології 2.2. Завдання на кваліфікаційну роботу бакалавра № 4101-5/225 від «30» травня 2026 р. (Додаток А до пояснювальної записки).
3.	Призначення розробки	3.1. Мета розробки: дослідження програмної моделі ідентифікації рухомих об'єктів для безпілотних літальних апаратів на основі методів глибокого навчання. 3.2. Призначення розробки полягає у забезпеченні автоматичного виявлення та класифікації рухомих об'єктів у відеопотоці з БПЛА з метою підвищення ефективності систем моніторингу, спостереження та автономної навігації, а також зменшення навантаження на оператора. 3.3. Вихідні дані розробки: відеодані та зображення, отримані з безпілотних літальних апаратів, а також навчальні набори даних для задач детекції об'єктів (зокрема VisDrone).
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: відсутні. 4.2. Вимоги до надійності: забезпечення стабільної роботи моделі та точності ідентифікації об'єктів не нижче сучасних аналогів у задачах комп'ютерного зору. 4.3.Вимоги до умов експлуатації: відсутні. 4.4. Вимоги до складу і параметрів технічних засобів: стандартний персональний комп'ютер або обчислювальна система з GPU (за можливості).

<table>
<tr><td></td><td></td><td>4.5. Вимоги до інформаційної та програмної сумісності: відсутні.
4.6. Вимоги до маркування та упаковки: відсутні.
4.7. Вимоги до транспортування і зберігання: на стандартних цифрових носіях інформації.
4.8. Спеціальні вимоги: відсутні.</td></tr>
<tr><td>5.</td><td>Вимоги до програмної документації</td><td>Програмною документацією до виробу «Модель ідентифікації рухомих об’єктів для безпілотних літальних апаратів» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику підготовки даних та навчання моделі (розділ 2 пояснювальної записки).
3) Опис програмної реалізації та алгоритму інференсу (основна частина роботи).</td></tr>
<tr><td>6.</td><td>Вимоги до техніко-економічних показників</td><td>Техніко-економічні показники розроблюваної моделі ідентифікації рухомих об’єктів для безпілотних літальних апаратів повинні забезпечувати її придатність до використання в умовах реального часу при обмежених обчислювальних ресурсах.
Економічна ефективність розробки полягає у зменшенні витрат часу на аналіз відеоданих та зниженні потреби у ручному моніторингу, що дозволяє підвищити автономність системи та зменшити експлуатаційні витрати.</td></tr>
<tr><td>7.</td><td>Стадії і етапи розробки</td><td>
<table>
<tr><th>Дата</th><th>Назва етапу</th></tr>
<tr><td>02.10.2025 – 13.10.2025</td><td>Затвердження теми кваліфікаційної роботи та узгодження індивідуального плану</td></tr>
<tr><td>13.10.2025 – 20.10.2025</td><td>Літературний огляд з проблематики комп’ютерного зору та виявлення об’єктів у відеоданих</td></tr>
<tr><td>21.10.2025 – 30.10.2025</td><td>Огляд сучасних методів виявлення та розпізнавання об’єктів у системах штучного інтелекту (зокрема моделей типу YOLO)</td></tr>
<tr><td>30.10.2025 – 10.11.2025</td><td>Аналіз особливостей обробки відеоданих, отриманих з безпілотних літальних апаратів</td></tr>
<tr><td>11.11.2025 – 20.11.2025</td><td>Аналіз та вибір методів виявлення рухомих об’єктів</td></tr>
<tr><td>20.11.2025 – 24.12.2025</td><td>Розробка концепції програмної моделі виявлення об’єктів для БПЛА</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table>
<tr><td>10.01.2026 – 01.02.2026</td><td>Проєктування структурної схеми програмної системи та алгоритму роботи моделі</td></tr>
<tr><td>02.02.2026 – 30.04.2026</td><td>Програмна реалізація моделі виявлення об’єктів</td></tr>
<tr><td>01.04.2026 – 30.04.2026</td><td>Тестування розробленої моделі на відеоданих та збір експериментальних результатів</td></tr>
<tr><td>01.05.2026 – 13.05.2026</td><td>Аналіз результатів тестування та оцінка ефективності моделі (точність, швидкодія)</td></tr>
<tr><td>01.05.2026 – 30.05.2026</td><td>Підготовка та оформлення пояснювальної записки кваліфікаційної роботи відповідно до вимог</td></tr>
<tr><td>01.05.2026 – 30.05.2026</td><td>Оформлення звіту про переддипломну практику</td></tr>
<tr><td>30.05.2026</td><td>Представлення кваліфікаційної роботи керівнику та рецензенту</td></tr>
</table></td></tr>
<tr><td>8.</td><td>Порядок контролю і приймання програмного продукту</td><td>1. Контроль виконання згідно календарного плану;
2. Захист на засіданні Екзаменаційної комісії;
3. Подання пояснювальної записки у паперовому та електронному вигляді.</td></tr>
</table>

Виконавець
студент групи КУ- 41
ХАРЧЕНКО Д. В.

Замовник
к.т.н., доцент зво
БИКОВА Т. В.

Додаток В

Програма і методика випробувань програмного виробу

«МОДЕЛЬ ІДЕНТИФІКАЦІЇ РУХОМИХ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ»

1 Об'єкт випробувань

1. Назва програмного виробу : «Модель ідентифікації рухомих об'єктів для безпілотних літальних апаратів»
2. Галузь застосування : автоматизація та комп'ютерно-інтегровані технології, комп'ютерний зір, безпілотні літальні апарати, системи відеоспостереження та аналізу аероданих.
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональних можливостей програмної реалізації моделі ідентифікації рухомих об'єктів вимогам технічного завдання (Додаток Б до пояснювальної записки до кваліфікаційної роботи), а також оцінка точності детекції та швидкодії інференсу в режимі реального часу.

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна забезпечувати:

1. роботу на операційних системах Windows, Linux, MacOS;

2. коректну обробку відеопотоків у режимі реального часу;
3. стабільну детекцію рухомих об'єктів на зображеннях і відео;
4. використання GPU (CUDA) для прискорення обчислень;
5. захист від некоректного вводу відеоданих або файлів;
6. сумісність із бібліотеками PyTorch, OpenCV та Ultralytics YOLO;
7. можливість роботи з відео, веб-камерою та датасетом VisDrone;
8. вимоги до складу і параметрів технічних засобів: звичайний ПК або ноутбук з GPU (рекомендовано NVIDIA);
9. вимоги до маркування та упаковки: не висуваються;
10. вимоги до транспортування і зберігання: на стандартних носіях інформації.

Спеціальні вимоги (не пред'являються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Модель ідентифікації рухомих об'єктів для безпілотних літальних апаратів» вважати:

1. Справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
2. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);
3. Опис програмної системи та алгоритму інференсу (Розділ 2 і 3 пояснювальної записки)
4. Текст програмної реалізації (додаток Г).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідні:

- персональний комп'ютер або ноутбук;
- Python 3.8+;
- бібліотеки PyTorch, OpenCV, Ultralytics;
- навчені ваги моделі YOLOv8 (best.pt);
- тестові відео або зображення (включно з VisDrone).

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:
-ознайомчий (1-й етап);
-випробування програмного виробу (2-й етап).
Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.
2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.
3. Перевірку комплектності складу технічних і програмних засобів.

4. Методику проведення перевірок на 1 етапі випробувань.
5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. перевірку відповідності технічних характеристик програми вимогам технічного завдання;
2. перевірку ступеня виконання функціональних вимог до програми;
3. методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

1. Програма працює відповідно до умов експлуатації операційних систем MS Windows, Linux та MacOS.
2. Для роботи необхідний компілятор мови програмування python, версії не нижчої ніж 3.8
3. Порядок проведення випробувань:
   3.1. Відкрити термінал у каталозі проєкту.
   3.2. Запустити програму командою: python src/run_detection.py --source 0(для тестування на веб-камері в реальному часі); python src/run_detection.py --source image.jpg(для тестування на зображені);python src/run_detection.py --source video.mp4(для тестування на відео);
   3.3. Програма завантажує навчену модель YOLOv8 та виконує інференс для заданого джерела даних (зображення, відеофайл або відеопотік з камери).
   3.4. Результати детекції зберігаються у вихідні файли та/або відображаються засобами OpenCV залежно від режиму роботи програми.

Для проведення випробувань пропонується тест 1, тест 2 та тест 3.

Тест 1

- перевірка запуску програмного модуля та ініціалізації системи детекції об’єктів на базі YOLOv8;
- запуск інференсу з використанням вебкамери як джерела відеопотоку (source = 0);
- автоматичне підключення до камери та перевірка коректності отримання кадрів у реальному часі;
- завантаження навченої моделі best.pt та ініціалізація нейронної мережі для виконання детекції;

– відображення відеопотоку у вікні OpenCV з накладенням результатів розпізнавання (bounding boxes, класи об’єктів та рівень впевненості);
– перевірка стабільності роботи системи під час безперервного потоку кадрів;
– оцінка продуктивності системи в реальному часі шляхом візуального контролю плавності відеопотоку та відсутності затримок;
– завершення роботи системи коректним закриттям вікна відображення після натискання клавіші «q» або сполуки клавіш «Ctrl+C».



Рисунок В.1 — Тестування на веб-камері.

Тест 2

– перевірка роботи програмного модуля детекції об’єктів на одиночному статичному зображенні;
– запуск інференсу з використанням зображення у форматі JPG/PNG як вхідних даних;
– ініціалізація та завантаження навченої моделі YOLOv8 (best.pt);

– виконання одноразового аналізу зображення з визначенням присутніх об’єктів;
– формування результату у вигляді обмежувальних рамок (bounding boxes) із зазначенням класів та рівня впевненості;
– відображення результатів обробки у вікні OpenCV;
– утримання результату на екрані до натискання будь-якої клавіші користувачем та завершення роботи програми.

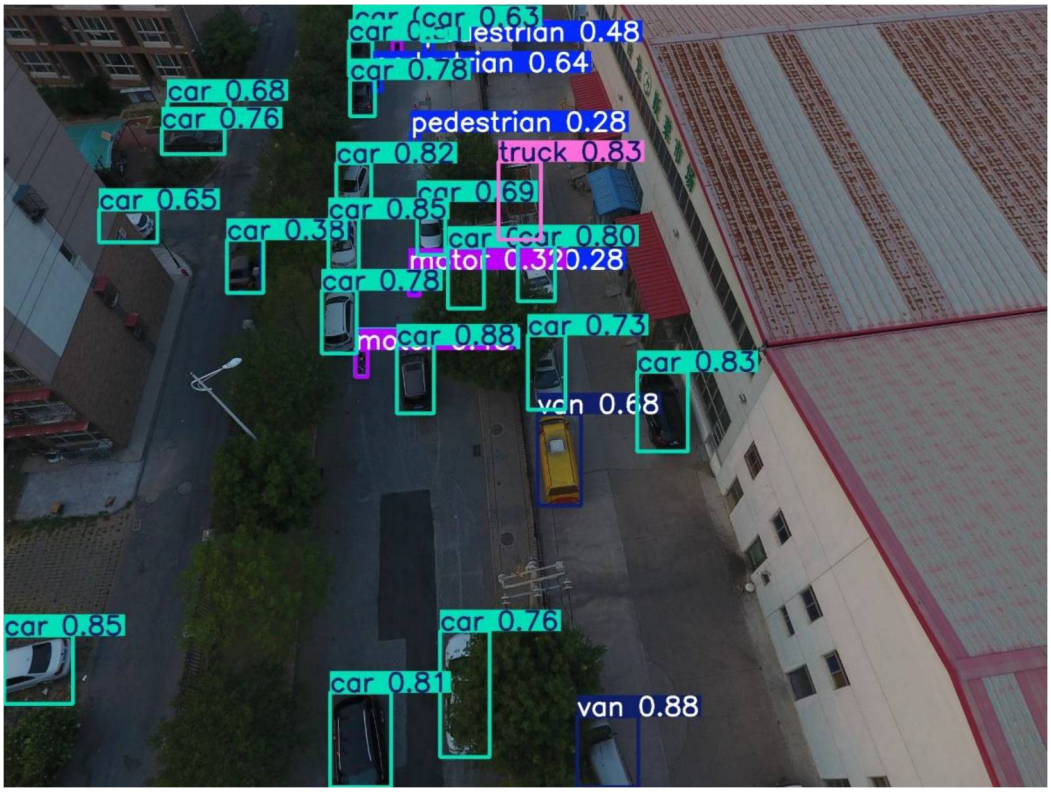


Рисунок В.2 — Тестування на зображенні.

Тест 3

– перевірка роботи програмного модуля детекції об’єктів при обробці відеофайлу у форматі MP4/AVI;
– запуск системи інференсу з використанням попередньо записаного відеопотоку як вхідного джерела даних;
– завантаження навченої моделі YOLOv8 (best.pt) та ініціалізація нейронної мережі для аналізу відео;

- послідовна обробка кожного кадру відеофайлу з виконанням детекції рухомих об’єктів;
- відображення результатів аналізу у вікні OpenCV із накладенням bounding boxes, класів об’єктів та рівня впевненості моделі;
- перевірка коректності обробки безперервного відеопотоку та відсутності пропусків кадрів;
- завершення роботи системи після повного відтворення відеофайлу або примусового зупинення користувачем.

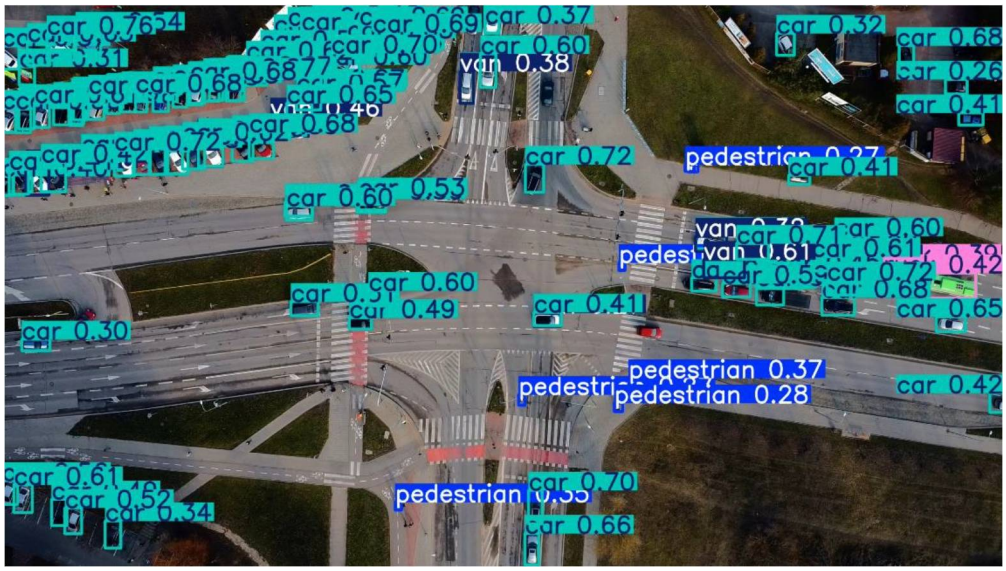


Рисунок В.3 — Тестування на відео.

Висновки: тест 1 успішно пройшов випробування, тест 2 успішно пройшов випробування і тест 3 успішно пройшов випробування. Система підтвердила коректну роботу програмного модуля ідентифікації рухомих об’єктів на основі YOLOv8 у режимі роботи з вебкамерою, відеофайлом та статичним зображенням. У межах пояснювальної записки демонстрація роботи у режимі відеопотоку не наводиться, оскільки у друкованому документі можна представити лише статичні зображеня.

Додаток Г

Вихідні коди програми

```
"""
detector.py — Ядро системи виявлення об'єктів для БПЛА.
"""
from ultralytics import YOLO
import cv2


class UAVObjectDetector:


    def __init__(self, model_path="best.pt", conf=0.25, iou=0.45, imgsz=640,
device="cuda"):
        self.model = YOLO(model_path)
        self.conf = conf
        self.iou = iou
        self.imgsz = imgsz
        self.device = device


    def detect(self, frame):


        res = self.model.predict(
            frame,
            conf=self.conf,
            iou=self.iou,
            imgsz=self.imgsz,
            device=self.device,
            verbose=False
        )[0]
```

```
        detections = []

        if res.boxes is None:
            return []
        for b in res.boxes:
            x1, y1, x2, y2 = map(int, b.xyxy[0])
            cls = int(b.cls[0])
            conf = float(b.conf[0])

            detections.append({
                "class_id": cls,
                "confidence": conf,
                "bbox": (x1, y1, x2, y2)
            })
        return detections
"""
run_detection.py — модуль запуску інференсу
"""
import cv2
import time
from ultralytics import YOLO
import argparse
from pathlib import Path
MODEL_PATH = Path(__file__).parent / "weights" / "best.pt"

def run(source):
    model = YOLO(MODEL_PATH)

    is_image = isinstance(source, str) and source.lower().endswith((".jpg",
".png", ".jpeg"))
```

```
is_camera = source == 0

cap = None

if not is_image:
    cap = cv2.VideoCapture(source)

prev_time = time.time()

while True:

    if is_image:
        frame = cv2.imread(source)
    else:
        ret, frame = cap.read()
        if not ret:
            break

    start = time.time()

    results = model.predict(
        frame,
        conf=0.25,
        iou=0.45,
        imgsz=640,
        verbose=False
    )[0]

    frame = results.plot()

    end = time.time()

    fps = 1.0 / (end - prev_time + 1e-6)
    prev_time = end

    latency = (end - start) * 1000

    if is_camera:
        cv2.putText(frame, f"FPS: {fps:.2f}", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

```
            cv2.putText(frame, f"Latency: {latency:.2f} ms", (10, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)

        cv2.imshow("YOLO Drone Detection", frame)

        if is_image:
            cv2.waitKey(0)
            break
        else:
            if cv2.waitKey(1) & 0xFF == ord("q"):
                break

    if cap:
        cap.release()

    cv2.destroyAllWindows()

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--source", default="0")
    args = parser.parse_args()

    source = 0 if args.source == "0" else args.source

    run(source)
"""
train.py —  навчання YOLOv8 на датасеті з БПЛА.
"""
import argparse
from ultralytics import YOLO


def train(
    model="yolov8n.pt",
    data="configs/visdrone.yaml",
    epochs=100,
    batch=8,
    imgsz=640,
    device=0,
):
```

```
    yolo = YOLO(model)

    yolo.train(
        data=data,
        epochs=epochs,
        batch=batch,
        imgsz=imgsz,
        device=device,

        optimizer="AdamW",
        lr0=0.001,
        weight_decay=0.0005,

        mosaic=0.8,
        mixup=0.05,

        fliplr=0.5,
        hsv_h=0.015,
        hsv_s=0.7,
        hsv_v=0.4,

        patience=20,
        amp=True,
        workers=2,
    )

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--model", default="yolov8n.pt")
    parser.add_argument("--data", default="configs/visdrone.yaml")
    parser.add_argument("--epochs", type=int, default=100)
    parser.add_argument("--batch", type=int, default=8)
    parser.add_argument("--imgsz", type=int, default=640)
    parser.add_argument("--device", default=0)

    args = parser.parse_args()

    train(args.model, args.data, args.epochs, args.batch, args.imgsz,
args.device)
"""
```

```
convert.py — Конвертація анотацій VisDrone-DET у формат YOLO.
"""
from pathlib import Path
from PIL import Image
def convert_split(split_dir: Path):

    images_dir = split_dir / "images"
    labels_in = split_dir / "annotations"
    labels_out = split_dir / "labels"
    labels_out.mkdir(parents=True, exist_ok=True)

    if not labels_in.exists():
        print("Missing annotations:", labels_in)
        return

    for ann in labels_in.glob("*.txt"):

        img_path = images_dir / (ann.stem + ".jpg")
        if not img_path.exists():
            continue

        w, h = Image.open(img_path).size

        lines_out = []

        for line in open(ann):

            parts = line.strip().split(",")
```

```
            if len(parts) < 6:
                continue

            x, y, bw, bh = map(float, parts[:4])
            score = int(parts[4])
            cls = int(parts[5])

            # ignore invisible objects
            if score != 1:
                continue

            if cls < 1 or cls > 10:
                continue

            cx = (x + bw / 2) / w
            cy = (y + bh / 2) / h
            nw = bw / w
            nh = bh / h

            lines_out.append(f"{cls-1} {cx} {cy} {nw} {nh}")

        (labels_out / ann.name).write_text("\n".join(lines_out))

def process(root):
    root = Path(root)

    for split in [
```

```
        "VisDrone2019-DET-train",
        "VisDrone2019-DET-val",
        "VisDrone2019-DET-test-dev",
    ]:
        convert_split(root / split)

if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser()
    parser.add_argument("--root", default="datasets")
    args = parser.parse_args()

    process(Path(args.root))
"""
evaluate.py —оцінка швидкодії.
"""

import cv2
import time
import numpy as np
from ultralytics import YOLO
from pathlib import Path

class ModelEvaluation:

    def __init__(self, model_path):
        self.model = YOLO(model_path)

        self.fps_list = []
        self.times = []
        self.total_frames = 0
        self.total_detections = 0
```

```
def evaluate_video(self, video_path, show=False):

    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        raise FileNotFoundError("Video not found")

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        start = time.perf_counter()

        results = self.model.predict(
            frame,
            conf=0.25,
            verbose=False
        )[0]

        end = time.perf_counter()

        dt = end - start
        fps = 1 / dt if dt > 0 else 0

        self.fps_list.append(fps)
        self.times.append(dt)

        if results.boxes is not None:
            self.total_detections += len(results.boxes)

        self.total_frames += 1

        if show:
            img = results.plot()
            cv2.imshow("Evaluation", img)

            if cv2.waitKey(1) & 0xFF == ord("q"):
                break
```

```
    cap.release()
    cv2.destroyAllWindows()

    return self._summary("video")

def evaluate_images(self, folder_path):

    images = list(Path(folder_path).glob("*.jpg")) + \
             list(Path(folder_path).glob("*.png"))

    for img_path in images:
        img = cv2.imread(str(img_path))

        start = time.perf_counter()

        results = self.model.predict(img, verbose=False)[0]

        end = time.perf_counter()

        dt = end - start
        fps = 1 / dt if dt > 0 else 0

        self.fps_list.append(fps)
        self.times.append(dt)

        if results.boxes is not None:
            self.total_detections += len(results.boxes)

        self.total_frames += 1

    return self._summary("images")

def _summary(self, mode):

    avg_fps = np.mean(self.fps_list)
    min_fps = np.min(self.fps_list)
    max_fps = np.max(self.fps_list)

    avg_time = np.mean(self.times) * 1000  # ms

    print("\n==============================")
```

```
        print(" MODEL EVALUATION")
        print("============================")
        print("Mode:", mode)
        print("Frames:", self.total_frames)
        print("Detections:", self.total_detections)
        print(f"FPS avg: {avg_fps:.2f}")
        print(f"FPS min: {min_fps:.2f}")
        print(f"FPS max: {max_fps:.2f}")
        print(f"Inference time: {avg_time:.2f} ms")
        print("============================\n")

        return {
            "mode": mode,
            "frames": self.total_frames,
            "detections": self.total_detections,
            "fps_avg": float(avg_fps),
            "fps_min": float(min_fps),
            "fps_max": float(max_fps),
            "inference_ms": float(avg_time)
        }


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--model", default="weights/best.pt")
    parser.add_argument("--source", default="video.mp4")
    args = parser.parse_args()

    evaluator = ModelEvaluation(args.model)
    result = evaluator.evaluate_video(args.source, show=True)
    print(result)
```

Файл YAML – конфігурація

```
path: ../datasets

train: VisDrone2019-DET-train/images
```

```
val: VisDrone2019-DET-val/images
test: VisDrone2019-DET-test-dev/images

nc: 10

names:
  0: pedestrian
  1: people
  2: bicycle
  3: car
  4: van
  5: truck
  6: tricycle
  7: awning-tricycle
  8: bus
  9: motor
```

Шляхи до файлів задаються відносно кореневого каталогу проєкту або передаються як аргументи командного рядка.