Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Система розпізнавання об'єктів та траєкторії їх руху для безпілотних літальних апаратів
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Нікіта Сагайдачний
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник доц. Олег Золотухін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сагайдачному Нікіті Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система розпізнавання об'єктів та траєкторії їх руху для безпілотних літальних апаратів

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 20 25 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет джерел

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Аналіз існуючих підходів та методів
3) Проектування системи та розробки алгоритму
4) Проведення дослідження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Аналіз предметної галузі	22.04.2025	виконано
3	Постановка задачі	23.04.2025	виконано
4	Експериментальні дослідження	15.05.2025	виконано
5	Аналіз результатів експериментальних	22.05.2025	виконано
	досліджень та розробка рекомендацій		
6	Підготовка пояснювальної записки	29.05.2025	виконано
7	Підготовка презентації та доповіді	02.06.2025	виконано
8	Нормоконтроль	06.06.2025	виконано
9	Рецензування	06.06.2025	виконано
10	Попередній захист	06.06.2025	виконано
11	Захист перед ЕК	10.06.2025	

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олег Золотухін
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 59 с., 13 рис., 1 дод., 10 джерел.

ANNOTATION AUTOMATION, INFERENCE, MACHINE LEARNING, OBJECT DETECTION, UAV, UAVDT, ULTRALYTICS, VIDEO STREAM, VISDRONE, YOLO.

Об’єкт дослідження – системи виявлення об’єктів на відеопотоці з безпілотних літальних апаратів (БПЛА).

Предмет дослідження – оптимізація процесу вибору датасету, архітектури детектора та метрик продуктивності для задач реального часу на базі моделей YOLO.

Мета роботи – обґрунтувати та реалізувати ефективну систему виявлення об’єктів у відеопотоці з БПЛА шляхом порівняльного аналізу сучасних моделей YOLO та вибору оптимального датасету для навчання.

Методи дослідження – системний аналіз датасетів (VisDrone, UAVDT), огляд та порівняння архітектур YOLO v8 та YOLO v11, автоматизована конвертація розмітки, експериментальне навчання моделей, логування продуктивності (mAP, FPS, латентність), тестування на відеостенді.

У кваліфікаційній роботі розглянуто процес побудови ефективної системи детекції об’єктів у відеопотоці БПЛА на основі моделей YOLO. Обґрунтовано доцільність використання VisDrone як основного набору для тренування моделей. Здійснено експериментальне навчання конфігурацій YOLO v8 nano та YOLO v11 nano з ідентичними параметрами, реалізовано модуль автоматизованої конвертації розмітки, а також побудовано тестовий стенд для оцінки продуктивності. Результати демонструють перевагу YOLO v11 у швидкодії без втрати точності, що підтверджує її доцільність для реального застосування в умовах обмежених ресурсів БПЛА.

ABSTRACT

Master’s thesis contains: 59 pp., 13 fig., 1 ann., 10 references.

ANNOTATION AUTOMATION, INFERENCE, MACHINE LEARNING, OBJECT DETECTION, UAV, UAVDT, ULTRALYTICS, VIDEO STREAM, VISDRONE, YOLO.

Object of study – systems for object detection in video streams from unmanned aerial vehicles (UAVs).

Subject of study – optimization of dataset selection, detector architecture, and performance metrics for real-time tasks based on YOLO models.

Aim of the work – to substantiate and implement an effective system for object detection in UAV video streams through a comparative analysis of modern YOLO models and the choice of an optimal dataset for training.

Research methods – systematic analysis of datasets (VisDrone, UAVDT), review and comparison of YOLO v8 and YOLO v11 architectures, automated annotation conversion, experimental model training, performance logging (mAP, FPS, latency), and testing on a video stand.

The qualification work considers the process of building an effective object detection system in UAV video streams based on YOLO models. A comparative analysis of VisDrone and UAVDT datasets was conducted, resulting in the substantiation of the feasibility of using VisDrone as the primary dataset for model training. Experimental training of YOLO v8 nano and YOLO v11 nano configurations with identical parameters was performed, an automated annotation conversion module was implemented, and a test stand for performance evaluation was built. The results demonstrate the advantage of YOLO v11 in speed without loss of accuracy, confirming its suitability for real-world applications under UAV resource constraints.

ЗМІСТ

ВСТУП

Задача розпізнавання людей на відеопотоці залишається актуальною та важливою в умовах сучасних систем безпеки, моніторингу, а також автоматизації різних технологічних процесів. Зростання вимог до точності й швидкості таких систем вимагає постійного вдосконалення методів та інструментів, зокрема, використання глибоких нейронних мереж для детекції та трекінгу.

Одним із провідних підходів у задачах об'єктного виявлення наразі є архітектура YOLO (You Only Look Once), яка постійно розвивається. Останні її версії, YOLOv8 та YOLOv11, демонструють значний потенціал у точності й продуктивності. Водночас, вибір якісного навчального датасету такої моделі є вирішальним фактором, що прямо впливає на ефективність кінцевого рішення [1].

У рамках цього дослідження було поставлено дві основні мети:

– визначити оптимальний датасет для тренування моделі, яка буде максимально точно розпізнавати людей в умовах реального відеопотоку з урахуванням складних ситуацій (туман, дим, зйомка з тепловізора тощо);

– провести порівняльний аналіз двох сучасних архітектур yolo–yolov8 та yolov11–з точки зору їхньої точності розпізнавання, швидкості роботи та загальної ефективності при роботі на різних апаратних платформах.

Для реалізації зазначених цілей ми створили повноцінний дослідницький пайплайн, що включає підготовку та конвертацію анотацій вихідного VisDrone-датасету у формат YOLO, налаштування навчального процесу та власноручне тренування моделей. У даному звіті буде детально описано підходи, методики та результати проведених досліджень, які дозволять зробити обґрунтований висновок щодо вибору оптимального датасету і моделі YOLO для задачі розпізнавання людей у складних умовах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз типів дронів

Дрони, або безпілотні літальні апарати (БПЛА), є пристроями, які виконують польоти без присутності людини на борту. Вони можуть управлятися дистанційно або працювати автономно завдяки алгоритмам і системам навігації. Спочатку створені для військових потреб, дрони швидко знайшли застосування в різних сферах, зокрема в комерції, науці, розвагах і громадському секторі.

Конструкція БПЛА варіюється від компактних моделей, які можна тримати в руці, до великих апаратів для перевезення вантажів. Вони оснащені сенсорами, камерами, GPS-модулями та іноді штучним інтелектом, що дозволяє їм виконувати складні завдання. Завдяки цьому дрони стали корисними для аерофотозйомки, моніторингу територій, доставки товарів, рятувальних операцій та досліджень важкодоступних місць.

Одна з головних переваг дронів – це здатність працювати в умовах, де перебування людини небезпечне або неможливе. Вони використовуються для спостереження за вулканами, дослідження океанів, перевірки інфраструктури, пошуку постраждалих під час стихійних лих і для бойових завдань. Технологічний прогрес зробив їх доступнішими, що сприяло широкому використанню в повсякденному житті.

У цивільному секторі дрони активно використовуються для зйомок та створення контенту, оскільки вони дають унікальні види з висоти. Вони також застосовуються в сільському господарстві для моніторингу полів, у будівництві для огляду територій і навіть у медицині для доставки ліків у віддалені райони.

Розвиток дронів супроводжується необхідністю регулювання їх використання. Збільшення популярності цих пристроїв створює нові

виклики щодо безпеки, приватності та впливу на навколишнє середовище. Тому в багатьох країнах приймаються правила, які обмежують висоту польотів, відстань від населених пунктів і вимагають реєстрації БПЛА.

Загалом, дрони стали символом сучасних технологій і перетворень у багатьох сферах, поєднуючи інженерні, електронні та програмні досягнення, відкриваючи нові можливості для людства. Попри труднощі, пов'язані з їх використанням, дрони постійно вдосконалюються, ставши ще ефективнішими і універсальнішими.

1.1.1 Типи дронів та їх придатність до завдань моніторингу

Дрони поділяються на кілька основних видів залежно від їх призначення та конструкції. Найпопулярніші серед них – квадрокоптери, фіксованокрилі дрони та гібридні дрони. Кожен із них має свої особливості, які роблять його унікальним, та найбільш ефективним для певних завдань.

Квадрокоптери являються найпоширенішим видом дронів, які використовують для зйомки, розваг, доставлення та інших повсякденних цілей. Вони оснащені чотирма роторами, що забезпечує стабільність у польоті та можливість зависати на одному місці. Завдяки своїй маневровості та простоті управління квадрокоптери ідеально підходять для аерофотозйомки та використання у міських умовах.

Фіксованокрилі дрони нагадують традиційні літаки з нерухомими крилами. Вони створені для польотів на великі відстані та виконання тривалих завдань, наприклад, спостереження за великими територіями, сільськогосподарський моніторинг, або, у військовій справі для ураження стратегічно важливих цілей на дуже дальніх відстанях. Такі дрони зазвичай працюють на паливі чи мають потужні акумулятори, що дозволяє їм залишатися в повітрі значно довше, ніж квадрокоптерам.

Гібридні дрони поєднують характеристики квадрокоптерів і фіксованокрилих моделей. Вони здатні злітати вертикально, як

квадрокоптери, але потім переходять у горизонтальний політ, як літаки. Це робить їх універсальними для завдань, що потребують маневреності та тривалого польоту.

Квадрокоптери – це один із найпоширеніших видів дронів, який здобув популярність завдяки своїй універсальності, простоті конструкції та доступності. Їх назва походить від конструктивної особливості – чотирьох ротори (пропелерів), які забезпечують підйомну силу і маневреність у повітрі. Завдяки такій конфігурації квадрокоптери можуть злітати вертикально, зависати на одному місці, виконувати точні маневри й здійснювати плавні посадки.

Основною перевагою квадрокоптерів є їхня стабільність у польоті, яку забезпечують сучасні системи управління польотом (Flight Control). Ці системи використовують гіроскопи, акселерометри та інші сенсори для підтримання рівноваги навіть у складних умовах, таких як сильний вітер. Схема будови подібних апаратів наведена на рисунку.

Ця схема відображає побудову корпусу, яка забезпечує стабільність в повітрі, а також камеру, яка є невід’ємною частиною подібних апаратів.

Крім того, квадрокоптери зазвичай оснащуються GPS-модулями, що дозволяє їм точно визначати місце розташування, повертатися до точки зльоту чи виконувати завдання за попередньо заданими маршрутами.

Цей тип дронів використовується у широкому спектрі сфер. В аерофотозйомці квадрокоптери стали незамінним інструментом завдяки можливості знімати високоякісні фото та відео з висоти. Їх часто застосовують для створення кінематографічних кадрів, зйомки нерухомості, моніторингу інфраструктури чи природних ландшафтів. У багатьох сучасних моделях встановлюють камери з високою роздільною здатністю, стабілізатори зображення та функції трансляції відео в реальному часі.

Ще однією популярною сферою застосування квадрокоптерів є доставка товарів. Завдяки здатності переносити невеликі вантажі й швидко долати відстані, вони використовуються для доставки продуктів, ліків чи

інших важливих предметів, особливо в місця, де традиційні методи транспортування ускладнені.

Квадрокоптери також знаходять застосування в наукових і технічних дослідженнях. Їх використовують для моніторингу довкілля, дослідження екосистем, вивчення кліматичних змін і навіть збору зразків з важкодоступних місць. Завдяки своїй компактності та мобільності вони здатні працювати в умовах, де інші технології виявляються недостатньо ефективними.

У військових цілях квадрокоптери використовуються для розвідки, спостереження та цілевказівки. Їхня маневровість і компактність дозволяють ефективно збирати дані в реальному часі, навіть у складних умовах. Завдяки оснащенню камерами, тепловізорами та іншими сенсорами, вони забезпечують моніторинг місцевості, виявлення противника та оцінку бойової обстановки. Крім того, квадрокоптери застосовуються для коригування артилерійського вогню, доставки спорядження або боєприпасів, а іноді й для атак із використанням легких боєзарядів, зберігаючи життя військових.

Технічний прогрес сприяв появі квадрокоптерів із функціями автономного польоту. Такі моделі здатні виконувати завдання без втручання людини, орієнтуючись у просторі завдяки алгоритмам штучного інтелекту.

Однак квадрокоптери мають і свої обмеження. Вони зазвичай мають менший радіус дії та час автономної роботи порівняно з іншими видами дронів, такими як фіксованокрилі. Їхня робота сильно залежить від умов навколишнього середовища: сильний вітер або дощ можуть негативно впливати на їхню стабільність і функціонування. Крім того, через їхню доступність і популярність виникають проблеми з безпекою та приватністю, оскільки квадрокоптери іноді використовуються для незаконної зйомки чи польотів у заборонених зонах.

Попри ці виклики, квадрокоптери залишаються найбільш популярним видом дронів завдяки своїй універсальності, простоті управління та

доступності. Вони продовжують удосконалюватися, стаючи більш потужними, компактними та розумними, що розширює їхні можливості та сфери застосування у сучасному світі.

Фіксованокрилі дрони – це безпілотні літальні апарати, які за своєю конструкцією нагадують традиційні літаки. Їхня головна особливість – нерухомі крила, які створюють підйомну силу за рахунок потоку повітря під час руху. На відміну від багатороторних дронів, таких як квадрокоптери, фіксованокрилі дрони не здатні зависати в повітрі чи здійснювати вертикальні злети й посадки. Натомість вони мають значні переваги у швидкості, дальності польоту та ефективності використання енергії.

Ці дрони зазвичай обладнані потужними двигунами, які можуть працювати на акумуляторах, паливі чи гібридних системах. Завдяки аеродинамічній конструкції вони здатні долати великі відстані та перебувати в повітрі протягом тривалого часу. Це робить фіксованокрилі незамінними для завдань, що вимагають широкого охоплення території, наприклад, моніторингу сільськогосподарських угідь, обстеження лісів, картографії чи у військовій справі.

Ефективність цих апаратів обумовлена їхньою здатністю використовувати підйомну силу крил, що знижує витрати енергії. Це особливо важливо для великих територій, які потрібно охопити без частого повернення на базу для підзарядки чи дозаправки. Завдяки цьому вони можуть працювати на висотах, недоступних для багатьох інших типів дронів, і залишатися в повітрі протягом дуже тривалого часу.

Фіксованокрилі дрони також мають високу швидкість, що робить їх оптимальними для завдань, де потрібна швидка обробка даних або доставка вантажів на великі відстані. Це може бути корисно, наприклад, для швидкої доставки медичних вантажів у віддалені регіони або для оперативного збору розвідданих у військових умовах.

Порівняно з квадрокоптерами, фіксованокрилі дрони менш маневрові та не можуть виконувати точні маневри, такі як зависання чи вертикальні

злети, що робить їх менш ефективними для завдань, що вимагають високої точності та гнучкості. Однак вони чудово підходять для польотів на великих відстанях та для моніторингу великих територій.

Завдяки своїм можливостям фіксованокрилі дрони широко застосовуються в сільському господарстві, геодезії, аерофотозйомці та у військових цілях. Вони використовуються для збору даних про стан посівів, моніторингу екосистем, створення детальних карт місцевості, а також для ведення розвідки та спостереження (рисунок 1.1).

Рисунок 1.1 – Приклад фіксованокрилого дрону

Гібридні дрони є безпілотними літальними апаратами, які поєднують властивості кількох типів дронів, найчастіше квадрокоптерів та фіксованокрилих моделей. Вони здатні злітати і приземлятися вертикально, як багатороторні дрони, але під час польоту переходять у горизонтальний режим, типовий для апаратів з фіксованими крилами. Така комбінація характеристик дозволяє гібридним дронам бути універсальними й ефективними для виконання різноманітних завдань.

Основною перевагою таких дронів є їх здатність поєднувати маневреність і простоту запуску багатомоторних апаратів із дальністю польоту та енергоефективністю фіксованокрилих моделей. Вони можуть злітати і приземлятися в обмежених просторах без потреби в злітно-посадкових смугах або додатковому обладнанні, що розширює їхні можливості. Після вертикального зльоту дрон переходить у горизонтальний політ, де крила забезпечують підйомну силу, знижуючи споживану енергію та збільшуючи тривалість польоту [2].

Ці дрони знайшли широке застосування в різних галузях. У сільському господарстві вони використовуються для моніторингу великих площ, аналізу стану посівів, обстеження лісових масивів та внесення добрив і пестицидів. Завдяки великій дальності польоту такі дрони можуть охоплювати значні території за один політ, що дозволяє економити час і ресурси.

У сфері розвідки та спостереження гібридні дрони грають важливу роль завдяки своїй здатності долати великі відстані та працювати в складних умовах. Вони ефективно використовуються для моніторингу інфраструктури, трубопроводів, електричних мереж і природних зон. Встановлення високоточних камер, тепловізорів і сенсорів дозволяє отримувати точні дані для подальшого аналізу та прийняття рішень.

Гібридні дрони також знайшли своє місце у військовій сфері. Вони використовуються для розвідки, цілевказівки, коригування артилерійського вогню, доставки спорядження та виконання інших завдань. Їх універсальність дозволяє швидко адаптуватися до різних умов на полі бою та забезпечувати перевагу в складних операціях.

Крім того, ці дрони активно використовуються для гуманітарних цілей, таких як доставка медикаментів, води та іншого вантажу в зони, постраждалі від стихійних лих. Їх здатність злітати в обмежених просторах та долати великі відстані робить їх незамінними в кризових ситуаціях, коли інші методи транспортування є неможливими.

Технологічний розвиток постійно вдосконалює гібридні дрони. Інженери працюють над покращенням аеродинаміки, зменшенням ваги, підвищенням ефективності двигунів і збільшенням тривалості роботи акумуляторів. Розробка моделей з використанням штучного інтелекту дозволяє гібридним дронам виконувати складні завдання автономно, включаючи аналіз даних і ухвалення рішень в реальному часі. Приклад будови таких апаратів наведена на рисунку 1.2.

Рисунок 1.2 – Гібридний дрон

Проте, попри численні переваги, гібридні дрони мають і деякі виклики. Їхня складна конструкція вимагає високої кваліфікації для обслуговування, а їхня ціна часто є вищою, ніж у традиційних моделей. Окрім того, розробка таких апаратів потребує детального проектування та тестування, щоб забезпечити баланс між вертикальним і горизонтальним режимами польоту.

Гібридні дрони являють собою сучасний етап розвитку безпілотних технологій. Вони поєднують функції різних типів апаратів і можуть працювати в умовах, де інші дрони не справляються. Їх застосовують у багатьох сферах – від промисловості до надзвичайних ситуацій, де потрібна

максимальна гнучкість і надійність. Наразі найбільш активно розвивається використання дронів у військовій сфері, тому основна увага буде зосереджена на їх застосуванні в бойових умовах, а також на методах ідентифікації людей і об’єктів, зокрема за допомогою штучного інтелекту.

1.1.2 Використання БПЛА в умовах оборонного застосування

Військові дрони стали невід’ємною частиною сучасних збройних сил, змінюючи підхід до ведення бойових дій, розвідки та логістики. Завдяки своїм технологічним можливостям та універсальності, вони забезпечують арміям важливі переваги на полі бою. Їх використовують для розвідки, коригування вогню, атаки, моніторингу та доставки вантажів в умовах, де традиційні засоби є менш ефективними або небезпечними.

Розвідка є основною функцією військових дронів. Оснащені високоточними камерами, тепловізорами, радарами та іншими сенсорами, вони забезпечують деталізований огляд місцевості в реальному часі. Дрони дозволяють стежити за рухом противника, оцінювати бойову ситуацію і збирати важливу стратегію інформацію без ризику для життя військових. Висока маневровість і малопомітність роблять їх ефективними в складних умовах, наприклад, у нічний час або погану погоду.

Дрони також активно використовуються для наведення та коригування артилерійських або ракетних ударів. Вони точно визначають координати цілей, підвищуючи ефективність ударів і знижуючи ризик ураження цивільних об'єктів. Їх здатність тривало перебувати в повітрі дозволяє здійснювати спостереження за змінами на полі бою і передавати інформацію на командні пункти.

Озброєні дрони, або ударні БПЛА, стали важливим елементом сучасного арсеналу. Вони можуть нести різні види озброєння, зокрема гранати, снаряди чи кулемети, що дозволяє виконувати точкові удари по цілях. Вони застосовуються для знищення бронетехніки, укріплень, живої

сили противника та важливих інфраструктурних об'єктів. Використання таких дронів зменшує ризики для пілотів і забезпечує високу точність ударів навіть у важкодоступних районах.

Малі військові дрони, такі як квадрокоптери, застосовуються для тактичних місій. Вони використовуються підрозділами на передовій для короткочасної розвідки, виявлення пасток чи мін, а також для спостереження за ворогом в реальному часі. Малі розміри дозволяють таким дронам бути малопомітними, мобільними і зручними у використанні.

Логістика також стала важливою сферою застосування дронів у військових операціях. Вони здатні доставляти боєприпаси, медикаменти і продовольство в зони активних бойових дій, куди традиційний транспорт не може потрапити. Це особливо важливо у віддалених районах або в умовах високого ризику для життя персоналу.

Особливу категорію становлять підводні та надводні військові дрони, які виконують завдання з розвідки водних територій, пошуку мін або моніторингу підводної інфраструктури. Вони також можуть бути використані для атак на кораблі або підводні човни противника, розширюючи можливості флоту.

Військові дрони стали важливим інструментом асиметричних дій, що дозволяє меншим державам або групам ефективно протистояти набагато більшим силам. Їх застосування змінює характер бойових дій, зменшуючи залежність від великих армійських підрозділів і відкриваючи нові стратегії ведення війни. Однак збільшення популярності дронів також породжує нові виклики, зокрема розробку засобів боротьби з безпілотними апаратами, таких як системи радіоелектронної боротьби і протидронові комплекси.

Очевидно, що дрони відіграють важливу роль у військовій сфері завдяки їхній універсальності, маневровості та здатності виконувати завдання з мінімальним ризиком для людських життів. Вони активно використовуються для розвідки, спостереження, виявлення загроз, коригування артилерійського вогню та доставки вантажів у важкодоступні

або небезпечні зони. Завдяки високотехнологічним сенсорам, камерам та системам передачі даних, дрони можуть забезпечити оперативну інформацію, критично важливу для ухвалення тактичних рішень. Крім того, завдяки автоматизації та штучному інтелекту, дрони здатні виконувати завдання з високою точністю навіть у складних умовах, таких як ніч, погана погода або густий ліс.

Розпізнавання рухів – це одна з ключових функцій дронів. Завдяки цій можливості можна не лише спостерігати за об’єктами, але й аналізувати їхню поведінку в реальному часі. Це допомагає вчасно виявляти загрози, такі як пересування техніки, підготовка до атаки чи нестандартні дії людей в зоні бойових дій. Такі системи точно фіксують навіть незначні зміни, знижуючи ймовірність хибних тривог та підвищуючи ефективність спостереження.

Використовуючи штучний інтелект, ці рішення стають ще більш потужними, дозволяючи військовим оперативно реагувати на небезпеки і забезпечувати додаткову безпеку в умовах складних військових операцій.

Системи ідентифікації є важливою темою, оскільки вони є основою багатьох сучасних технологій і процесів. Завдяки розвитку таких систем забезпечується точне розпізнавання осіб, об’єктів або даних, що відкриває нові можливості в безпеці, автоматизації та персоналізації. У зв’язку з цим, ми зосередимося на розгляді саме цієї функції дронів.

1.1.3 Методи ідентифікації та відстеження об'єктів

Ідентифікація цілей за допомогою дронів включає використання передових технологій для розпізнавання об’єктів, визначення їхніх характеристик і аналізу в реальному часі. Цей процес передбачає кілька основних методів, що забезпечують високу точність навіть у складних умовах. Завдяки поєднанню різних підходів, дрони здатні виконувати

завдання в розвідці, військових операціях, моніторингу та гуманітарних місіях.

Один із поширених методів – використання відеокамер високої роздільної здатності, що забезпечують детальне зображення місцевості чи об’єкта. Алгоритми комп’ютерного зору дозволяють дронам в реальному часі аналізувати відео, виділяючи об’єкти за заданими характеристиками, такими як форма, розмір, рух чи колір. Це використовують для моніторингу територій, виявлення підозрілих об’єктів та відстеження рухомих цілей.

Іншим методом є тепловізійне сканування. Тепловізори допомагають виявляти об’єкти за їх тепловим випромінюванням, що ефективно в умовах поганої видимості, таких як ніч чи туман. Дрони можуть розпізнавати живі об’єкти, навіть коли вони приховані. Цей метод широко використовується в пошуково-рятувальних та військових операціях.

Радіолокація також є важливим методом ідентифікації, дозволяючи виявляти об’єкти за допомогою відбитих радіохвиль. Радіолокаційні дрони ефективно працюють при поганій видимості та можуть виявляти об’єкти, що не випромінюють тепло або мають камуфляж.

Акустичні сенсори дозволяють ідентифікувати цілі за звуковими характеристиками, аналізуючи джерела звуків, такі як двигуни, вибухи або стрілянина. Це допомагає в операціях і моніторингу в зонах конфлікту.

Інфрачервоні сенсори виявляють об’єкти за їх інфрачервоним випромінюванням, що невидиме для людського ока, і використовуються для специфічних завдань, таких як виявлення прихованих об'єктів або техніки на великих відстанях.

Сенсори для збору радіочастотних даних можуть виявляти електромагнітне випромінювання, що дозволяє виявляти об’єкти, які важко спостерігати візуально, наприклад, мобільні телефони чи радіостанції.

Інтеграція штучного інтелекту (ШІ) дозволяє дронам аналізувати великі обсяги даних і передбачати поведінку цілей. Це дає можливість не

тільки розпізнавати об’єкти, а й прогнозувати їхній рух, як у випадку зміни напрямку транспортних засобів чи руху груп людей.

Обробка даних в реальному часі є важливою складовою. Отримана інформація передається до командних центрів, де вона аналізується для прийняття оперативних рішень, що допомагає інтегрувати дані з інших джерел, таких як супутникові знімки або наземні спостереження.

Мультиспектральні та гіперспектральні камери допомагають ідентифікувати об’єкти за їх спектральними характеристиками, що особливо корисно для виявлення матеріалів або рослинності, а також для розпізнавання прихованих об’єктів.

Використання GPS і картографічного програмного забезпечення дозволяє дронам точно визначати координати цілей, що важливо для точного планування рятувальних операцій або нанесення точкових ударів.

Завдяки комбінації цих методів дрони можуть виконувати широкий спектр завдань, що робить їх важливим інструментом для різних операцій, забезпечуючи високу автономність і точність в найскладніших умовах.

1.1.4 Алгоритмічні основи машинного навчання для розпізнавання

Машинне навчання стало однією з найбільш прогресивних наукових та технічних сфер, яка активно використовується для розробки інноваційних рішень у різних галузях, таких як військова справа, медицина, фінанси, транспорт. Алгоритми машинного навчання можуть навчатися на даних і приймати рішення без необхідності в явному програмуванні. Базуючись на математичних методах, статистиці та обчислювальних підходах, вони здатні виявляти закономірності, передбачати події та адаптуватися до нових умов. Це дозволяє ефективно працювати з великими обсягами даних, аналізувати складні системи та автоматизувати процеси.

У застосуванні до розпізнавання рухів дронами, машинне навчання відіграє ключову роль у створенні точних систем обробки зображень і відео.

Такі системи можуть розпізнавати різні типи рухів, класифікувати об'єкти, прогнозувати їхні траєкторії та аналізувати їхню поведінку в реальному часі. Це стало можливим завдяки використанню таких підходів, як навчання з учителем, без учителя і підкріплення. Наприклад, нейронні мережі, що імітують роботу людського мозку, дозволяють моделювати складні зв'язки в даних, що забезпечує високу точність і адаптивність, що особливо важливо для військових застосувань.

Далі варто розглянути застосування методів машинного навчання в контексті дронів. Дрони активно використовують різні технології для детекції та розпізнавання об'єктів, і багато з описаних алгоритмів адаптовані до їхніх потреб. Ось деякі з них:

– yolo (you only look once) – популярний алгоритм для дронів, який дозволяє здійснювати швидку детекцію об'єктів у реальному часі, таких як транспортні засоби, люди, тварини під час моніторингу території. yolo ефективний завдяки низьким вимогам до обчислювальних потужностей, що робить його підходящим для дронів з обмеженими ресурсами;

– ssd (single shot multibox detector) – використовується для виявлення об'єктів на великих висотах, що особливо корисно під час моніторингу сільськогосподарських територій чи в пошуково-рятувальних операціях;

– r-cnn та його варіанти (fast r-cnn, faster r-cnn) – застосовуються для більш точного виявлення об'єктів, таких як техніка або підозрілі предмети, особливо у військових операціях. Однак їх використання на дронах обмежене через високі вимоги до обчислювальних ресурсів;

– mask r-cnn – алгоритм для сегментації об'єктів, який використовується для оцінки пошкоджень інфраструктури або моніторингу стану лісів, здатен визначати не лише об'єкти, а й їх точну форму;

– hog (histogram of oriented gradients) – метод для виявлення простих об'єктів, таких як люди чи транспортні засоби. Попри те, що він поступово втрачає популярність через розвиток глибоких нейронних мереж, все ще використовується на деяких дронах;

– vision transformers (vit) – новий підхід, що набирає популярності для аналізу великих територій, із здатністю враховувати глобальні залежності в зображеннях, хоча й потребує потужного обладнання;

– навчання підкріплення (reinforcement learning) – застосовується для автономного управління дронами, де дрон навчається вибирати оптимальні маршрути та розпізнавати об'єкти в складних умовах;

– lstm та інші рекурентні нейронні мережі – використовуються для аналізу відеопотоку, зокрема для передбачення траєкторій рухомих об'єктів, таких як транспорт або пішоходи, що дає змогу дронам передбачати їхню поведінку [3].

1.1.5 FPV дрони

FPV-дрони (First Person View) є одним із найбільш перспективних і водночас технічно складних типів безпілотних літальних апаратів. Особливість цих пристроїв полягає у тому, що пілот керує дроном з перспективи першої особи, спостерігаючи за тим, що відбувається через відеокамеру, встановлену безпосередньо на дроні. Це дозволяє пілоту отримати максимально реалістичну картину навколишнього середовища, що забезпечує ефективне виконання високошвидкісних та складних маневрів, які є неможливими для класичних дронів з традиційною системою керування.

Проте, саме ця особливість ставить надзвичайно високі вимоги до швидкості обробки інформації, яка отримується з камери дрона. FPV-дрони рухаються на високих швидкостях, іноді перевищуючи сотні кілометрів на годину, тому навіть найменша затримка в передачі або обробці інформації може призвести до втрати орієнтації чи зіткнення з перешкодами. Відповідно, саме real-time обробка зображень, заснована на моделях штучного інтелекту, стає критично важливим елементом керування FPV-дронами, забезпечуючи оперативність та точність реакцій.

Особливо актуальним є впровадження систем автономного розпізнавання та донаведення на ціль. Традиційні системи з ручним керуванням мають значні обмеження в умовах високошвидкісного польоту та складних сценаріїв, таких як виконання тактичних завдань, пошуково-рятувальних операцій чи моніторингу в умовах обмеженої видимості. Використовуючи штучний інтелект для розпізнавання об'єктів та автоматичного донаведення, FPV-дрон може миттєво визначати пріоритетні цілі, корегувати свій маршрут у режимі реального часу та уникати потенційних перешкод з максимальною ефективністю.

1.2 Проблематика предметної галузі

Після аналізу, стає очевидним що безпілотні літальні апарати набувають дедалі більшого поширення, особливу увагу привертає сегмент FPV-дронів. Цей тип апаратів характеризується високою маневреністю, значною швидкістю пересування та специфічними сценаріями використання, зокрема у сфері військових операцій, пошуково-рятувальних місій та задачах оперативного моніторингу. Проте, незважаючи на стрімкий розвиток цієї галузі, сьогодні існує ряд фундаментальних викликів, які гальмують подальший розвиток та ефективну інтеграцію штучного інтелекту в системи керування FPV-дронами. Однією з ключових проблем є відсутність чіткого та обґрунтованого розуміння того, які саме моделі штучного інтелекту найбільш придатні для вирішення задач автономного розпізнавання об'єктів і донаведення у режимі реального часу, а також на яких датасетах їх необхідно навчати, щоб досягти оптимального співвідношення апаратних витрат і якості роботи.

Зокрема, у сегменті FPV-дронів необхідно забезпечувати миттєву реакцію на зміну навколишніх умов, і будь-яка затримка в розпізнаванні чи прийнятті рішень може критично впливати на виконання місії. Водночас більшість сучасних моделей глибокого навчання, що застосовуються для

задач комп'ютерного зору, розроблені переважно для стаціонарних або менш динамічних сценаріїв. Через це актуальною залишається проблема адаптації наявних архітектур нейромереж (зокрема, різних версій YOLO та аналогічних рішень) під умови високошвидкісного польоту та інтенсивної зміни кадрів відеопотоку. Додатково, критичним є аспект підбору ефективних датасетів, які могли б адекватно відображати реальні умови експлуатації FPV-дронів – наприклад, зйомка у складних умовах освітленості, поганої видимості, наявності перешкод у вигляді диму чи туману, що особливо характерно для бойових дій або пошукових операцій.

Також слід враховувати, що ресурсні можливості апаратного забезпечення, встановленого на FPV-дронах, значно обмежені. Це ставить додаткові вимоги до моделей штучного інтелекту: вони повинні бути не лише точними, але й максимально ефективними з точки зору обчислювальних ресурсів. Тому дослідницькі роботи, що ставлять собі за мету визначити оптимальний баланс між точністю розпізнавання і швидкістю обчислень, мають надзвичайно важливе значення. Зараз в літературі відсутній єдиний підхід до вибору архітектури нейромережі та конкретного датасету для їх навчання у контексті FPV-дронів, що створює значну невизначеність при розробці практичних рішень для таких систем.

Таким чином, проблема вибору найбільш оптимальної моделі нейронної мережі та відповідного датасету для її навчання в умовах високодинамічного середовища FPV-дронів залишається актуальною та відкритою для досліджень. Сучасні наукові роботи не дають чіткої відповіді на те, які з наявних моделей забезпечують найкраще співвідношення продуктивності та точності, а також які набори даних найбільш адекватно відповідають реальним умовам їхнього використання. Вирішення цієї проблематики дозволить суттєво покращити ефективність FPV-дронів, розширити сфери їх застосування та значно підвищити якість виконання складних місій у режимі реального часу [4].

1.3 Постановка задачі

З огляду на швидкий розвиток і широке використання FPV-дронів у різних сферах, таких як військові місії, рятувальні операції, моніторинг та спостереження, особливу увагу слід приділити створенню ефективних систем автономного розпізнавання та супроводу цілей у режимі реального часу. Одним із ключових аспектів, що визначають ефективність таких систем, є вибір оптимальних моделей штучного інтелекту та відповідних наборів даних для їх тренування. Проте, на сьогодні не існує однозначних критеріїв та рекомендацій щодо того, які саме моделі та датасети варто застосовувати для досягнення найкращого співвідношення точності розпізнавання, швидкості роботи та апаратних ресурсів, необхідних для функціонування систем на борту FPV-дронів.

Таким чином, основною задачею цього дослідження є комплексний аналіз та визначення найбільш ефективних моделей нейронних мереж, зокрема YOLOv8 та YOLOv11, для розпізнавання людей з високою точністю у складних умовах польоту FPV-дрона. Дане дослідження передбачає вибір відповідних датасетів, які максимально відповідають реальним умовам експлуатації FPV-дронів, та адаптацію їх формату до потреб обраних моделей нейромереж. Крім того, необхідно реалізувати повний цикл розробки та інтеграції цих моделей у систему керування FPV-дроном, яка дозволить проводити реальні польові випробування з подальшим аналізом результатів.

Щоб успішно досягти поставлених цілей і провести ґрунтовне порівняння моделей YOLOv8 та YOLOv11, потрібно виконати наступний перелік дій:

Вибір декількох потенційно придатних датасетів, що найбільше відповідають сценаріям використання FPV-дронів у різних умовах (включаючи складні погодні умови, низьку видимість, тепловізійні зображення тощо).

Аналіз та оцінка придатності кожного датасету щодо якості, повноти, складності анотацій, відповідності задачі детекції та трекінгу людей.

Конвертація обраних датасетів у формат, необхідний для навчання обраних нейронних мереж YOLO (нормалізовані координати bbox згідно специфікації YOLO).

Реалізація та налаштування навчального пайплайну для тренування моделей YOLOv8 та YOLOv11 на підготовлених датасетах з наступною оптимізацією гіперпараметрів.

Створення тестового програмного комплексу, здатного здійснювати реальну детекцію та трекінг людей у потоковому відео з FPV-дрона, а також вимірювати затрати ресурсів (CPU, GPU, RAM).

Проведення порівняльного аналізу моделей за показниками точності розпізнавання, швидкодії, ресурсних затрат та стабільності роботи у різних сценаріях використання.

Виконання цих дій дозволить сформулювати обґрунтовані висновки та рекомендації щодо вибору оптимальних моделей нейромереж та датасетів для задач автономного розпізнавання та трекінгу людей з використанням FPV-дронів, а також забезпечить максимальну адаптацію моделей до реальних умов експлуатації.

2 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА МЕТОДІВ

2.1 VisDrone

Для початку варто дослідити доступні сети даних для навчання.

VisDrone є одним з найпоширеніших і найбільш визнаних датасетів для задач детекції об’єктів у повітряній зйомці, який був створений з метою забезпечити уніфіковану платформу для оцінки алгоритмів комп’ютерного зору в умовах, близьких до реального застосування дронів. Цей датасет сформовано Інститутом автоматизації Китайської академії наук і він активно використовується в численних міжнародних змаганнях і дослідженнях з 2018 року.

VisDrone охоплює широкий спектр сцен, включаючи міські, приміські та сільські території, зібрані в реальних умовах за допомогою безпілотних літальних апаратів. Кожне зображення містить високу варіативність об'єктів як за формою, так і за масштабом, що є особливо важливим для побудови моделей, здатних до узагальнення. Датасет включає понад 10 000 зображень і більше ніж 2,5 мільйона анотацій об’єктів у відеопослідовностях, що робить його одним із найоб’ємніших у своїй категорії. Анотації містять координати об’єктів, їх розміри, клас (наприклад, людина, автомобіль, велосипед, вантажівка тощо), ступінь видимості (повна, часткова або повністю перекрита) та інші метадані. Ця розмітка дозволяє не лише тренувати моделі детекції, але й застосовувати методи трекінгу й семантичного аналізу.

Особливістю VisDrone є те, що він фіксує об’єкти у складних реальних умовах: зі змінною освітленістю, з різних ракурсів, при наявності диму, туману, або часткового перекриття. Це робить його надзвичайно релевантним для використання в сценаріях, де штучний інтелект має функціонувати в реальному часі, як, наприклад, у військових FPV-дронах, які потребують високої точності розпізнавання при обмежених

обчислювальних ресурсах. На відміну від більш контрольованих датасетів на кшталт Pascal VOC або COCO, VisDrone моделює саме ті виклики, з якими стикається комп’ютерний зір на дронах в умовах реального світу.

Крім того, VisDrone підтримує як статичні зображення, так і відеопослідовності, що є критично важливим для розробки і тестування алгоритмів обробки потокового відео. Доступність великих відеопослідовностей із детальною розміткою дозволяє дослідникам реалізовувати складні моделі трекінгу, що зберігають ідентичність об’єктів в часі, та поєднувати їх із детекторами в єдину інтегровану систему.

Таким чином, VisDrone – це не просто набір даних, а повноцінна платформа, яка охоплює весь спектр задач – від детекції до трекінгу – і забезпечує реалістичні, багатосценарні умови для розробки і вдосконалення моделей штучного інтелекту, орієнтованих на використання в дронових системах. Саме тому цей датасет справедливо вважається золотим стандартом у сфері повітряного комп’ютерного зору. На рисунку 2.1 наведено приклад відпрацювання.

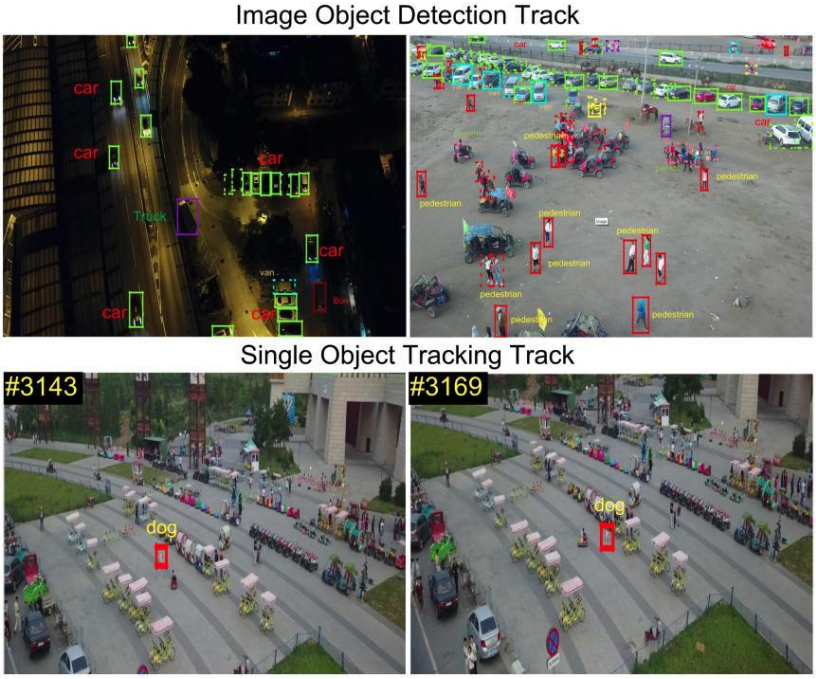


Рисунок 2.1 – VisDrone

VisDrone містить зображення та відео, зібрані в реальних умовах з дронів над міськими, сільськими та промисловими зонами. Це забезпечує високу варіативність сцени, ракурсів і освітлення, наближену до бойових чи цивільних FPV-застосунків.

Більше ніж 10 000 зображень і 2,5 мільйона анотацій дозволяють навчати моделі з високим рівнем узагальнення та стійкості до нових середовищ.

Кожен об’єкт розмічений не лише за координатами та класом, а й за ступенем перекриття/видимості, що дає змогу враховувати складні випадки, важливі для практичної роботи з FPV-системами [5].

Доступність повністю анотованих відеопослідовностей дає змогу розробляти й тестувати моделі трекінгу в динаміці, що особливо важливо для задач автономної навігації та слідкування за цілями.

Містить об’єкти різних розмірів, у складних умовах (туман, тінь, перекриття), що дозволяє тренувати моделі, які демонструють стабільну якість детекції навіть в умовах обмежених ресурсів, характерних для дронів.

VisDrone підтримується конкурсами, такими як CVPR Drone Detection Challenge, і входить до багатьох бенчмарків, що робить результати на ньому порівнюваними з міжнародними стандартами.

Ці переваги роблять VisDrone найкращим кандидатом для навчання й оцінки моделей виявлення об’єктів у FPV-режимах, де точність і стійкість до перешкод мають вирішальне значення.

2.1.1 UAVDT

UAVDT (Unmanned Aerial Vehicle Detection and Tracking) – це спеціалізований датасет, створений для розвитку й оцінювання алгоритмів комп’ютерного зору, призначених для роботи з повітряними відеопотоками. Його повна назва – UAV Detection and Tracking Benchmark – підкреслює фокус на ключових завданнях: детекції об’єктів, трекінгу та

багатокадровому аналізі. Датасет був розроблений з огляду на потреби інтелектуального відеоспостереження за допомогою безпілотників і включає реалістичні сценарії, що охоплюють динамічні сцени міських вулиць, автомагістралей та стоянок.

UAVDT складається з 100 відео загальною тривалістю понад 80 000 кадрів, знятих за допомогою безпілотного літального апарата з різних висот, кутів і швидкостей. Усі відео мають супровідну анотацію, яка охоплює положення об’єктів у кожному кадрі, клас (наприклад, легкові автомобілі, автобуси, вантажівки), а також інформацію про орієнтацію об'єкта, рівень затінення та погодні умови. Цей багатошаровий опис дозволяє не лише вирішувати класичні задачі виявлення об’єктів, а й тренувати більш складні моделі, здатні враховувати варіативність зовнішнього середовища.

Однією з особливостей UAVDT є його фокус на транспортних засобах – це робить його дуже релевантним для завдань моніторингу трафіку з дронів або аерофотоінспекції. Більшість відео записано у денних умовах, часто – з фіксованих маршрутів або на заданій висоті, що дає змогу зібрати порівняно стабільні послідовності кадрів. Проте така стабільність також є обмеженням, оскільки вона не завжди відповідає непередбачуваним траєкторіям у реальних FPV-операціях, де дрон часто змінює напрямок, висоту та кут зору.

З технічної точки зору UAVDT добре підходить для тестування базових архітектур моделей типу Faster R-CNN або YOLO, але менш адаптований для моделей, що вимагають глибокої варіативності в умовах зйомки. Висока щільність об'єктів у деяких сценах дозволяє перевірити ефективність NMS (Non-Maximum Suppression) алгоритмів, але водночас обмежений класовий розподіл – переважно транспорт – не дозволяє тренувати універсальні моделі для широкого спектру цілей, зокрема людських об’єктів.

Попри це, UAVDT широко використовується у науковій спільноті, особливо в контексті задач трекінгу й багатооб'єктної обробки. Він входить

до ряду бенчмарків та викликів і продовжує бути корисним інструментом для оцінки базових функцій дронових систем в умовах упорядкованого, інфраструктурного середовища.

Отже, UAVDT – це збалансований, але більш спеціалізований датасет, який відображає певну частину реального сценарію застосування дронів. Його сильними сторонами є чіткість розмітки, послідовність даних та технічна придатність для задач з транспортними об'єктами. Проте у випадках, де важливо забезпечити універсальність, адаптивність до різких змін середовища або фокус на людських цілях, UAVDT поступається датасетам на кшталт VisDrone, що мають вищу сценову варіативність і глибину об'єктного представлення. 2.1.2 Least Connections та Weighted Round Robin [6].

2.1.2 Теоретичне порівняння дата-сетів

У межах порівняльного дослідження ефективності алгоритмів виявлення об’єктів у повітряній зйомці, результати на датасетах VisDrone і UAVDT демонструють виразну перевагу першого з точки зору точності виявлення. Ключовою метрикою, що використовується в оцінці, є середня точність (AP – Average Precision), яка відображає здатність моделі точно і стабільно розпізнавати об’єкти різного типу й масштабу в умовах варіативного фону, ракурсів та перекриттів.

У дослідженні було зафіксовано, що модель досягла AP = 31.9% на VisDrone проти 21.1% на UAVDT, що становить приріст точності на понад 50% у відносних показниках. Такий суттєвий розрив можна пояснити декількома чинниками. По-перше, VisDrone містить більше різноманітних класів об'єктів, зокрема людей, велосипедистів, мотоциклів і невеликих транспортних засобів, що дозволяє моделі навчатися на ширшому спектрі семантичних ознак. По-друге, сцени у VisDrone є значно динамічнішими та мають багатший набір контексту – зміни освітлення, куту огляду, варіації

масштабу, рух об'єктів і перешкоди. Результати тестування VisDrone та UAVDT наведені на рисунках 2.2, 2.3 відповідно.

Model	Source	Backbone	AP	AP_{50}	AP_{75}
Cascade RCNN [22]	CVPR2018	ResNet50	24.0	39.0	25.0
YOLOv5-x [45]	GitHub2020	CSPDarknet53	24.1	44.0	15.2
GFL [46]	CVPR2021	ResNet50	25.9	41.7	27.2
★ClusDet [28]	ICCV2019	ResNet50	26.7	50.6	24.7
QueryDet [13]	CVPR2022	ResNet50	28.3	48.1	28.7
CEASC [34]	CVPR2023	Resnet18	28.7	50.7	28.4
★DMNet [47]	CVPRW2020	ResNeXt101	29.4	49.3	30.6
SDPNet [48]	TGRS2023	ResNet50	30.2	52.5	30.6
DSHNet [49]	WACV2021	ResNet50	30.3	51.8	30.9
★GLSAN [27]	TIP2021	ResNet50	30.7	55.4	30.0
RT-DETR-X	Arxiv2023	HGNetv2	31.0	52.0	30.9
★CZDet [10]	CVPRW2023	ResNet50	33.2	58.3	33.2
★UFPMPDet [11]	AAAI2022	ResNet50	36.6	62.4	36.7
Our method	–	Resnet50	31.9	53.6	32.3

Note: The "★" indicates that the detector uses a coarse-to-fine pipeline.

Рисунок 2.2 – Результати по VisDrone (рисунок виконано самостійно)

Table 2 (continued)

Model	Source	Backbone	AP	AP_{50}	AP_{75}
DSHNet [49]	WACV2021	ResNet50	17.8	30.4	19.7
★UFPMPDet [11]	AAAI2022	ResNet50	24.6	38.7	28.0
Our method	–	Resnet50	21.1	36.3	22.8

Note: The "★" indicates that the detector uses a coarse-to-fine pipeline.

Рисунок 2.3 – Результати по UAVDT (рисунок виконано самостійно)

З іншого боку, UAVDT хоча й містить добре структуровані анотації й підходить для задач трекінгу, значно поступається у варіативності даних.

Більшість відео сконцентровано навколо дорожнього руху, що знижує універсальність навченої моделі, особливо якщо мета полягає в адаптації під системи FPV-дронів, де необхідно розпізнавати різноманітні об’єкти в неконтрольованих середовищах.

Також варто зазначити, що навіть при однаковій архітектурі моделі (наприклад, YOLOv3 чи RetinaNet), ефективність на VisDrone залишається стабільно вищою. Це свідчить не лише про якість самої розмітки, але й про цінність VisDrone як тренувального джерела для моделей, призначених для роботи в реальному часі в нестабільних повітряних середовищах.

Отже, при виборі датасету для навчання моделі, що має бути інтегрована в FPV-дрон для автономного виявлення цілей, VisDrone демонструє об'єктивно вищий потенціал, підтверджений числовими метриками точності. Такий вибір не лише підвищує надійність детекції, а й забезпечує кращу адаптацію до складних умов реального застосування.

2.2 Вибір моделей

У контексті безпілотних систем пріоритетними залишаються дві взаємопов’язані вимоги – висока точність локалізації та гарантована робота в реальному часі на обмежених обчислювальних ресурсах. Упровадження кожної нової ітерації YOLO спрямоване саме на пошук балансу між цими критеріями, проте практичний ефект змін не завжди очевидний без безпосередньої верифікації. Саме тому для порівняння було обрано пари моделей YOLO v8 і YOLO v11 (у реалізації Ultralytics), які репрезентують два показові, але достатньо близькі за хронологією покоління.

Версія v8 позиціонується як «точка насичення» першої хвилі покращень після переходу на детектор-якорі ZeroHead: вона поєднує зрілу архітектуру, масове ком’юніті-підтримання й оптимізований inference-кодекс, що вже довів свою стабільність у галузевих застосунках. Натомість

v11 (із внутрішнім індексом 12 у Ultralytics) пропонує принципово інакшу конфігурацію neck-блоку, перевід оптимізації на повністю NMS-less pipeline та перегляд стемів у бік швидшої конвергенції під час fine-tune. Теоретично це має дати помітні прирости mAP при тому ж або нижчому латентсі, однак документально задекларовані переваги ще не прошли комплексної перевірки саме на наборах з анімацій безпілотників.

Додатковим аргументом на користь вибору цієї пари стала їхня взаємна сумісність інструментів: обидві версії компілюються однією CLI-утилітою ultralytics, мають ідентичний формат YAML-конфігурації, приймають спільні чекпоінти масштабування (nano, small тощо) і використовують однакову схему конвертації розмітки. Завдяки цьому виключено сторонні змінні, пов’язані з різними фреймворками, – увесь розрив у результатах можна буде віднести до самої архітектурної різниці, а не до специфіки оболонок чи навчальних скриптів.

Не менш важливо, що обидві моделі доступні з відкритими вихідними вагами, мають адаптацію під широке коло апаратних платформ – від дискретних GPU до одноплатних комп’ютерів класу Orange Pi. Це відразу переводить результати порівняння з категорії суто академічного інтересу у реальні рекомендації для практичних розробників, які змушені робити вибір між «перевіреним» поколінням і новітньою, але ще не верифікованою ревізією.

Отже, YOLO v8 і YOLO v11 одночасно представляють еволюційний стрибок усередині однієї лінійки та залишаються достатньо близькими, щоб їх можна було зіставляти в одному стандартизованому експерименті без зміни решти змінних. Саме це робить вибір пари оптимальним для дослідження, спрямованого на з’ясування практичної доданої вартості чергової генерації алгоритму детекції в умовах реального безпілотного відеопотоку.

2.3 Методика проведення дослідження

Дослідження буде виконуватися з метою отримати репрезентативні та порівняльні дані щодо продуктивності YOLO v8 та YOLO v11, навчених на різних безпілотних датасетах. У роботі передбачається використати VisDrone і UAVDT, які забезпечать схожість тематичних сцен (міська забудова, транспортні потоки, скупчення людей) і водночас продемонструють відмінності у щільності анотацій, роздільній здатності кадрів та спектрі класів. Такий підхід дозволить простежити, якою мірою моделі різних поколінь будуть чутливі до неоднорідності вихідних даних.

Спочатку буде сформовано модуль підготовки даних. Планується автоматично конвертувати анотації VisDrone з формату (Xmin, Ymin, w, h). у формат YOLO, що використовує нормовані координати центру рамки та її розміри. Скрипт convert_to_yolo.py обчислюватиме центр рамки, виконуватиме нормалізацію за розмірами зображення та зберігатиме результати у піддиректорії converted_labels. Після контрольної перевірки файлів анотацій буде сформовано структуру каталогів, сумісну з бібліотекою ultralytics; відповідні шляхи та кількість класів будуть зазначені у файлі data.yaml.

Далі буде проведене навчання двох моделей – YOLO v8n та YOLO v11n – окремо на кожному з датасетів. Для забезпечення коректного порівняння планується застосувати однакові гіперпараметри: 100 епох, розмір входу 640 × 640 px, оптимізатор SGD із початковою швидкістю навчання 0,01 і косинусним спаданням, batch = 16. Початкові ваги будуть завантажені з відкритих репозиторіїв ultralytics-hub, а процес тренування – зареєстрований у TensorBoard для подальшого аналізу показників mAP@50, mAP@50:95 та поведінки функцій втрат.

Після завершення тренування кожної конфігурації планується здійснити тестування на єдиному відеопотоці роздільністю 1920 × 1080 px і частотою 30 FPS. Буде визначено середню затримку обробки кадру,

фактичний FPS та кількість кадрів, у яких модель виявляє принаймні один об’єкт. Щоби зменшити випадковий розкид значень, кожну серію вимірів буде повторено не менше трьох разів із подальшим усередненням результатів.

Всі експерименти планується виконувати на єдиній апаратній конфігурації: GPU NVIDIA RTX 4070S (12 GB VRAM), CPU Intel Core i7- 12700H, 32 GB RAM, Python 3.10, а також Ultralytics v8.1.2 і v11.0.0 відповідно. Уніфіковане середовище мінімізує вплив апаратних чинників і дає можливість надійно порівняти точність і швидкодію моделей різних версій.

3 ПРОЕКТУВАННЯ СИСТЕМИ ТА РОЗРОБКА АЛГОРИТМУ

3.1 Проектування архітектури системи

Архітектура дослідження передбачає послідовну роботу двох самостійних, але логічно зчеплених підсистем. Перша охоплює повний цикл підготовки даних та навчання моделей, друга – їхнє подальше оцінювання у режимі відеопотоку.

У підсистемі підготовки та навчання вихідний набір VisDrone (а за потреби – UAVDT) буде відразу розділено на зображення та текстові анотації. Конвертаційний модуль, що реалізується скриптом convert_to_yolo.py, перераховуватиме прямокутники з форматів (Xmin, Ymin, w, h) у нормовані координати центру й розмірів, сумісні з вимогами Ultralytics-YOLO. На етапі валідації конвертовані лейбли перевірятимуться на синтаксичну цілісність і відповідність розмірам кадрів. Після цього дані структуруватимуться в каталоги train і val, а файл data.yaml зберігатиме шляхи до підмножин і список класів. На основі цих ресурсів скрипт train_yolo.py ініціалізує навчання YOLOv8 n і YOLOv11 n із однаковими гіперпараметрами; журнали робочих метрик автоматично передаватимуться до TensorBoard. Таким чином, перша підсистема формує дві готові до інференсу вагові модельні пари – по одній на кожному датасеті.

Друга підсистема, тестово-оцінювальна, буде підключати збережені ваги і подавати їх на уніфікований відеопотік 1080p / 30 FPS. У кадр за кадром детектор YOLO виконуватиме локалізацію об’єктів, після чого (за потреби) результати передаватимуться трекеру Deep SORT для підтримання стійких ідентифікаторів. Паралельно лічильники фіксуватимуть латентність інференсу, фактичний FPS, а також частку кадрів, у яких виявлено принаймні одну ціль. Підсистема завершуватиме роботу формуванням зведеного протоколу: середні часові витрати, розподіл кадрів за наявністю

детекцій, усереднені значення mAP, Precision та Recall, що імпортуватимуться до фінальної аналітичної записки.

Поділ на дві підсистеми дає змогу ізолювати задачі підготовки даних та навчання від задач реального тестувального навантаження, що спрощує відлагодження кожного етапу та робить результати порівняння моделей повністю відтворюваними.

3.2 Файлова структура

Нижче на рисунку 3.1 наведено файлову структуру системи.

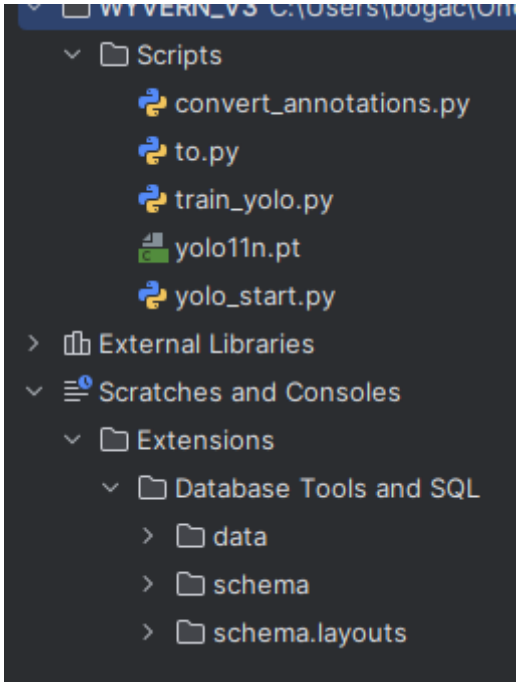


Рисунок 3.1 – Файлова структура WYVERN _V3

У межах каталогу WYVERN _V3 усі робочі файли зосереджено у підпапці Scripts. Її наповнення формує замкнений цикл «підготовка навчання демонстрація» й віддзеркалює логічні етапи, описані у попередньому розділі.

Першим елементом ланцюжка виступає модуль convert_annotations.py. Під час виконання дослідження він буде перетворювати початкові текстові анотації VisDrone або UAVDT, що подають координати прямокутника через верхній лівий кут і розміри, у нормалізовані центри та габарити, сумісні з форматом YOLO. Скрипт автоматично створюватиме підкаталоги для збереження перетворених лейблів та перевірятиме, чи співвідноситься кількість нових .txt із кількістю зображень, тим самим забезпечуючи цілісність підготовленого набору.

Ряд допоміжних процедур (створення та очищення службових тек, копіювання зображень, контроль вмісту) буде зосереджено у файлі to.py. Саме він надаватиме іншим скриптам готові функції для роботи з файловою системою, що зменшить дублювання коду і спростить подальшу підтримку проєкту.

Коли конвертація завершиться, ініціалізацію процесу навчання перебере на себе train_yolo.py. Під час дослідження цей скрипт завантажуватиме стартові ваги, описані у файлі yolo11n.pt, розгортатиме конфігурацію Ultralytics і запускатиме серію епох з уніфікованими гіперпараметрами. Проміжні метрики та графіки втрат будуть автоматично збиратися у підпапках runs/train/exp…, що надасть можливість відстежувати динаміку навчання в TensorBoard і обрати фінальну модель за критерієм найкращого показника mAP. Після завершення етапу користувач отримає ваги best.pt та last.pt для кожної версії YOLO.

Щоби перевірити працездатність отриманих ваг на практичному відеопотоці, передбачено скрипт yolo_start.py. Під час демонстраційного сеансу він буде відкривати вказане відео або веб-камеру, подавати кадри на модель, а результат візуалізувати у вікні разом з обчисленням фактичного FPS. Таким чином утворюється швидкий тестовий контур, за допомогою якого дослідник одразу оцінить якість розпізнавання та швидкодію без залучення допоміжних утиліт.

Такий поділ функцій усередині папки Scripts забезпечить чітку послідовність: спершу відбуватиметься конвертація розмітки, далі – однотипне навчання двох варіантів YOLO, а на завершення – інтерактивна перевірка працездатності моделей на реальних відеоданих.

Нижче на рисунку 3.2 наведено файлову структуру системи тестування моделей.

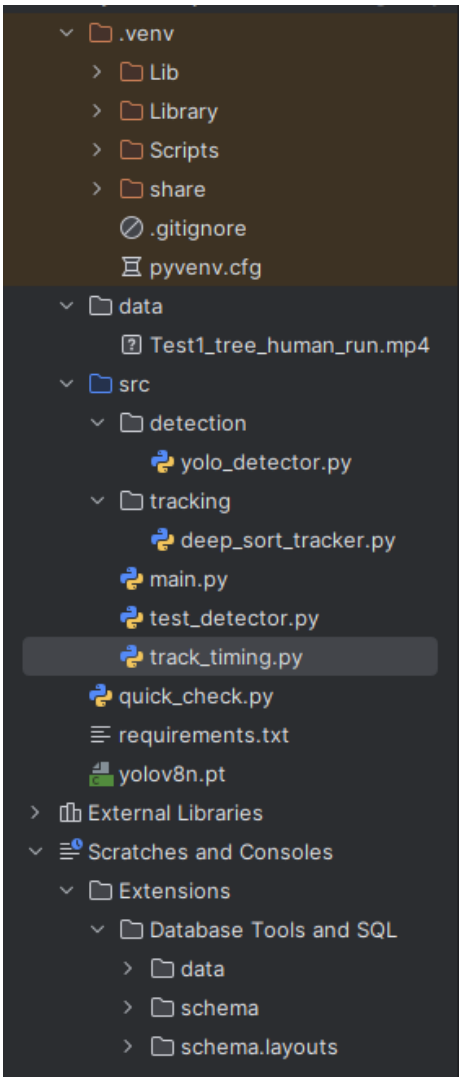


Рисунок 3.2 – Файлова структура WYVERN _V3

Локальне віртуальне середовище .venv із підкаталогами Lib, Scripts, share містить ізольовані залежності Python 3.10; цей шар повністю відокремлює тестовий стенд від глобальної системи й гарантує

відтворюваність пакетних версій. Безпосередньо під коренем проєкту розташовано файл requirements.txt, у якому перелічено бібліотеки Ultralytics, Torch, OpenCV, Deep-SORT-Realtime та допоміжні пакети – саме звідси буде ініційовано встановлення залежностей до .venv.

Каталог data/ зберігає контрольний відеоролик Test1_tree_human_run.mp4; він служитиме єдиним джерелом зображень під час вимірювання швидкодії моделей. Вибір розміщувати тестові мультимедійні файли поза src/ мінімізує ризик випадкового потрапляння великих об’єктів під систему контролю версій.

Всі вихідні модулі згруповано в src/. Усередині src/detection розміщено yolo_detector.py – універсальний клас-обгортка, який завантажує ваги YOLO v8n, виконує інференс на окремому кадрі й повертає спрощений список детекцій. Поруч у src/tracking зберігається deep_sort_tracker.py; тут інкапсульовано логіку Deep SORT: асоціація боксів із ідентифікаторами та функція to_ltrb() для подальшої візуалізації. Вага базової моделі yolov8n.pt лежить окремим файлом у корені проєкту, щоб її можна було легко підмінити на кастомну без зміни коду.

Три короткі виконавчі скрипти закривають різні потреби дослідника. test_detector.py перевірятиме працездатність детектора на статичному відео; track_timing.py під час серії прогонів накопичуватиме час інференсу та кількість кадрів із успішними детекціями; main.py залишено як точку входу для інтегрованого сценарію «детекція + трекінг у реальному часі». Окремий файл quick_check.py створено для миттєвої діагностики середовища – він виводить версії Torch, OpenCV та наявність CUDA, що дозволить швидко зрозуміти, чи відповідає конфігурація мінімальним вимогам експерименту.

У сукупності така фізична організація коду й даних забезпечить логічну послідовність роботи: середовище встановлюється й тестується один раз, далі дослідник у межах src/ перемикається між завданнями тестування, вимірювання продуктивності та інтегрованої демонстрації, не ризикуючи пошкодити ваги чи початкові відеодані [7].

3.3 Розробка алгоритму адаптації дата-сету

Вихідні розмітки VisDrone застосовують прямокутники, задані лівою-верхньою точкою та абсолютними розмірами кадру.

Щоб зробити їх придатними для будь-якої реалізації YOLO, потрібно перейти до нормованого опису рамки відносно ширини й висоти зображення. Процес перетворення передбачає такі логічні кроки.

Зіставлення анотації та зображення. Для кожного текстового файла розмітки береться кадр з тим самим іменем; його реальна ширина та висота зчитуються безпосередньо з JPEG/PNG-файла. Це дає точний масштаб, на який можна нормувати координати.

Валідний відбір боксів. Бокс враховується лише якщо має додатні розміри, не виходить за межі кадру й не містить пропусків у полі класу. Такі перевірки позбавляють модель помилкових зразків.

Далі всі координати та розміри діляться на відповідні габарити зображення, переводячи їх у відрізок [0,1]. Завдяки цьому одна й та сама розмітка лишається коректною незалежно від того, з якою роздільністю YOLO отримає кадр на вході.

Для кожного валідного бокса записується nginx:

```
class_id x_center_norm y_center_norm width_norm height_norm
```

із фіксованою шістнадцятковою точністю, що прийнята в Ultralytics. Один файл одна назва повна сумісність із очікуваннями train/val-процедури.

Таким чином конвертаційний алгоритм забезпечує безперервність координат, відсікає потенційно хибні бокси й гарантовано видає вихід у форматі, який YOLO може читати без додаткових правок. Нижче наведено код (лістинг 3.1).

Лістинг 3.1 – Перетворення VisDrone-розмітки для тренування YOLO-моделей

```
    def convert_visdrone_to_yolo(input_dir, output_dir, img_dir):
        os.makedirs(output_dir, exist_ok=True)
        for file_name in os.listdir(input_dir):
            if not file_name.endswith(".txt"):
                continue
            input_path = os.path.join(input_dir, file_name)
            output_path = os.path.join(output_dir, file_name)
            img_name = file_name.replace(".txt", ".jpg")
            img_path = os.path.join(img_dir, img_name)
            if not os.path.exists(img_path):
                print(f"Image not found for annotation: {file_name}")
                continue
            with Image.open(img_path) as img:
                img_width, img_height = img.size
            with open(input_path, "r") as infile, open(output_path, "w") as outfile:
                for line in infile:
                    parts = line.strip().split(",")
                    if len(parts) < 5:
                        print(f"Skipping invalid line in {file_name}: {line.strip()}")
                        continue
                    x_min, y_min, width, height, class_id = map(float, parts[:5])
                    if x_min < 0 or y_min < 0 or width <= 0 or height <= 0:
                        print(f"Warning: Skipping invalid box in file {file_name}: {line.strip()}")
                        continue
                    x_center = (x_min + width / 2) / img_width
                    y_center = (y_min + height / 2) / img_height
```

Продовження лістингу 3.1

```
                    width_norm = width / img_width
                    height_norm = height / img_height
                    if not (0 <= x_center <= 1 and 0 <= y_center
<= 1 and width_norm > 0 and height_norm > 0):
                        print(f"Warning: Skipping out-of-bound
box in file {file_name}: {line.strip()}")
                        continue
                    outfile.write(f"{int(class_id)}
{x_center:.6f}          {y_center:.6f}          {width_norm:.6f}
{height_norm:.6f}\n")
```

Функція convert_visdrone_to_yolo працює як одноразовий конвеєр, що в межах одного виклику повністю «прочищає» вхідну папку VisDrone-розміток і створює паралельну «дзеркальну» колекцію YOLO-лейблів. Першою інструкцією виконується os.makedirs(output_dir, exist_ok=True): якщо цільової директорії ще немає, вона створюється; якщо вже є, Python мовчки переходить далі – таким чином ми гарантуємо, що для кожного .txt буде куди записати результат.

Далі починається перебір файлів: for file_name in os.listdir(input_dir). Кожний елемент відразу фільтрується по розширенню; це дешевший варіант, ніж виконувати glob, і він дозволяє, наприклад, тримати в тій же папці зображення без ризику «пропустити» їх випадково через неправильне ім’я. Для валідного імені формується пара шляхів – input_path (оригінал) та output_path (майбутній YOLO-файл). Одночасно шлях до кадру будується простим replace розширення (.txt → .jpg), оскільки VisDrone тримає анотацію й кадр під спільним індексом.

Блок if not os.path.exists(img_path) – швидка перевірка: якщо картинку справді загублено, функція не зупиняє весь процес, а друкує діагностику й переходить до наступного файла. Така м’яка деградація дозволяє пропустити кілька бракованих пар і не втратити решту датасету.

Саме зображення відкривається через контекстний менеджер with Image.open(img_path) as img: – це гарантує коректне закриття дескриптора. Зчитуємо img_width, img_height, і вони залишаться незмінними для всіх боксів даного файла.

Другий вкладений with відкриває одразу два файлові об’єкти – вхідний і вихідний. Читаємо оригінальний рядок, робимо strip, split(',') і перевіряємо, що len(parts) >= 5. Якщо поле класу відсутнє або порожнє, рядок ігнорується з pop-up-попередженням. Далі всі п’ять перших значень перетворюються на float – це уніфікує типи, незалежно від того, записані вони цілими чи з десятковою крапкою.

Перша валідація: негативні координати або нульова (від’ємна) ширина/висота – означають помилковий бокс. У такому разі continue перескакує рядок, аби не плодити корупцію в даних.

На цьому етапі координати ще «в пікселях кадру». Наступний крок – обчислення центру: (x_min + width / 2) / img_width та ... / img_height. Одночасно нормалізуються width_norm, height_norm. Отримані чотири числа вже лежать у [0,1], тому для додаткової безпеки функція ще раз перевіряє, що жодне з них не вийшло за межі (це може статися, якщо вихідний бокс частково виходив за кадр). Якщо перевірка не проходить – рядок також ігнорується.

Фінальна дія в циклі – outfile.write(...): у вихідний .txt пишеться ідентифікатор класу (цілочисельний), після чого чотири нормовані значення з шістьма знаками після крапки.

Форматування :.6f забезпечує стабільну точність і одночасно компактний обсяг файла. На рисунку 3.3 наведено приклад файлової теки проекту конвертації дата-сету.

Каталог містить мінімальний набір для підготовки тренувального набору: у images розташовано вихідні кадри, у annotations – оригінальні текстові файли VisDrone/UAVDT, а під-тека yolo_annotation зберігає ті самі розмітки вже після конвертації у формат YOLO.

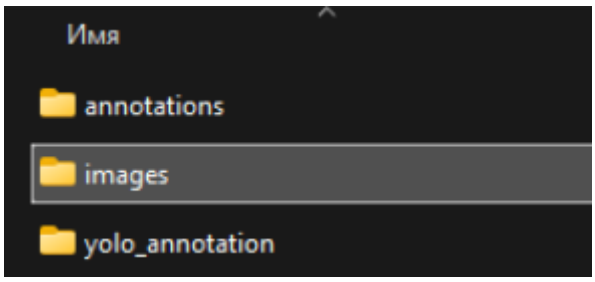


Рисунок 3.3 – Файлова структура теки WYVERN

Після завершення головного циклу функція автоматично закриває всі відкриті хендли (контекстні менеджери) й переходить до наступного .txt; коли список вичерпано – робота завершується. У результаті вихідна директорія output_dir міститиме рівно ту ж кількість валідних лейблів, що й пар «зображення + анотація» у початковій папці, але вже в нативному для Ultralytics форматі.

3.4 Розробка алгоритму тестування детекції

Далі варто розглянути тестування моделей. Скрипт тестування виконує функцію оперативного бенчмарку: він відтворює відеопотік кадр за кадром, подаючи кожне зображення на вхід попередньо ініціалізованому детектору YOLO, і у такий спосіб вимірює фактичну затримку інференсу. Насамперед програма перевіряє доступність відеофайла, упевнюється, що OpenCV здатен відкрити його без помилок, і фіксує номінальну частоту кадрів, аби згодом локально зіставити обчислену тривалість із паспортною. Далі створюється єдиний екземпляр YoloDetector з указаними порогами довіри та пригнічення перекриттів; це забезпечує сталий конфігураційний фон для всіх подальших вимірів.

У головному циклі скрипт бере черговий кадр, одразу стартує високоточний таймер і після виклику det.detect() фіксує час завершення. Таким чином накопичується сумарний час інференсу та список покадрових

латентностей, що дає змогу оцінити як середню швидкодію, так і варіативність. Одночасно підраховується число кадрів, у яких алгоритм видав принаймні одну валідну рамку: цей показник характеризує «корисну» частину потоку, де об’єкти взагалі присутні. Кадри з успішною детекцією додатково відображаються у вікні з накладеними прямокутниками та конфіденціями, що дозволяє досліднику візуально переконатися у коректності локалізації.

Після проходження всього відео або ручного переривання програма формує зведений звіт: фактичну кількість опрацьованих кадрів, паспортний FPS, обчислену тривалість потоку, загальний і середній час інференсу, а також відсоток кадрів із детекціями. Отримані величини використовуються далі у порівняльному аналізі моделей, оскільки віддзеркалюють реальну продуктивність детектора в умовах, максимально наближених до експлуатаційних [8].

Нижче наведено код (лістинг 3.2).

Лістинг 3.2 – Моніторинг об'єктів у відеопотоці за допомогою YOLO

```
parser.add_argument(
    '--conf', type=float, default=0.3,
    help='поріг довіри для детекції'
)
parser.add_argument(
    '--iou', type=float, default=0.45,
    help='поріг NMS (IoU)'
)
args = parser.parse_args()
print("=== DIAGNOSTICS ===")
print("Source       :", args.source)
print("Exists       :", os.path.exists(args.source))
print("Is file      :", os.path.isfile(args.source))
cap = cv2.VideoCapture(args.source)
print("cap.isOpened():", cap.isOpened())
```

Продовження лістингу 3.2

```
        ret, _ = cap.read()
        print("First frame read:", ret)
        if not ret:
            print("Не вдалося прочитати жодного кадру.")
            return
        cap.release()
        cap = cv2.VideoCapture(args.source)
        print("=== END DIAGNOSTICS ===\n")
        det        =        YoloDetector(conf_thres=args.conf,
iou_thres=args.iou)
        total_det_time = 0.0
        frames_with_object = 0
        per_frame_stats = []
        frame_idx = 0
        fps = cap.get(cv2.CAP_PROP_FPS) or 30.0
        while True:
            ret, frame = cap.read()
            if not ret:
                break
            start_det = time.perf_counter()
            dets = det.detect(frame)
            end_det = time.perf_counter()
            det_time = end_det - start_det
            total_det_time += det_time
            if len(dets) > 0:
                frames_with_object += 1
            for x1, y1, x2, y2, conf in dets:
                cv2.rectangle(
                    frame,
                    (int(x1), int(y1)),
                    (int(x2), int(y2)),
                    (0, 255, 0), 2
                )
                cv2.putText(
```

Продовження лістингу 3.2

```
                frame,
                f"{conf:.2f}",
                (int(x1), int(y1) - 5),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.5, (0, 255, 0), 1
            )
        cv2.imshow('YOLO Detection Timing', frame)
        if cv2.waitKey(1) & 0xFF == 27:  # ESC
            break
        per_frame_stats.append((frame_idx, det_time))
        frame_idx += 1
    cap.release()
    cv2.destroyAllWindows()
```

Модуль argparse зчитує шлях до відео-файла (або номер камери), пороги conf та iou. Далі скрипт одразу перевіряє три речі: чи існує файл фізично, чи справді це файл, а не директорія, й чи може OpenCV відкрити його усередині VideoCapture. До того ж читається перший кадр, що дає змогу переконатися в коректності кодеків. Якщо будь-яка перевірка провалюється, програма завершується, аби не витрачати час на подальший процес.

Єдиний екземпляр YoloDetector створюється наперед із заданими порогами; це важливо, адже ініціалізація моделі є ресурсомісткою й повинна відбутися лише один раз перед циклом обробки.

Для кожного кадру запускається високоточний таймер time.perf_counter(), після чого кадр миттєво передається у det.detect(frame). Коли модель повертає список детекцій, таймер зупиняється, і різниця додається до акумулятора total_det_time. Якщо список непорожній, лічильник frames_with_object приростає на одиницю. Для кожної рамки координати конвертуються в цілі числа, накладаються на кадр зеленою обводкою, а поряд ставиться текст із двома знаками після крапки – таким

чином користувач бачить лише валідні бокс-конфіденції. Функція cv2.waitKey(1) не тільки відображає кадр, а й дає змогу перервати цикл клавішею ESC.

По завершенні потоку дескриптор відео та всі вікна OpenCV закриваються командою destroyAllWindows(). Далі скрипт обчислює загальну кількість кадрів, паспортну тривалість відео та середню латентність (загальний час детекції, поділений на кількість кадрів). Окремий показник – частка кадрів, де було знайдено щонайменше один об’єкт; це непрямо відбиває змістовність набору та чутливість порога conf. Усі величини друкуються у структурованому вигляді та можуть бути одразу перенесені до таблиці результатів порівняння моделей. Зібрані масиви per_frame_stats за потреби легко експортуються у CSV або візуалізуються, щоб ми могли оцінити розподіл латентності по всій послідовності кадрів.

Загально у розділі було розглянуто ключові технічні рішення для приведення координат до стандартів наших моделей YOLO та для тестування моделі та отримання порівняльних метрик. Також було розглянуто загальну роботу додатків.

4 ПРОВЕДЕННЯ ДОСЛІДЖЕННЯ

4.1 Проведення тестування

Загальний пайплайн дослідження буде наступним:

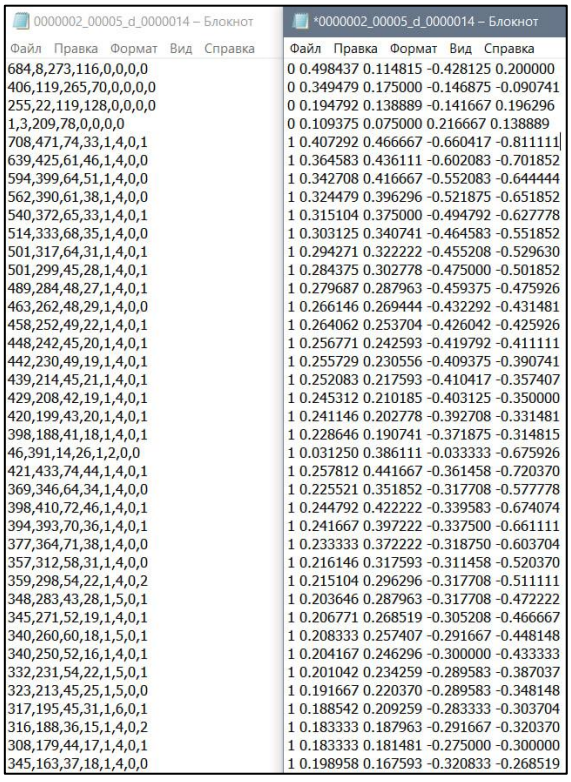
– конвертуємо анотації visdrone / uavdt із формату (x_min, y_min, w, h) у нормовані координати yolo й формуємо каталоги train/ та val/;

– проводимо навчання yolo v8 і volo v11 (nano) на підготовленому наборі з уніфікованими гіперпараметрами та зберігаємо best.pt;

– завантажуємо кожен із двох чекпоінтів у скрипт track_timing.py, подаємо єдиний тестовий відеопотік 1080p / 30 fps і вимірюємо latency, фактичний fps та кількість кадрів із детекціями;

– експортуємо зібрані метрики в csv, будуємо узагальнені графіки та порівнюємо точність і швидкодію yolo v8 проти yolo v11.

Отже спершу було проведенно конвертацію координат (рисунок 4.1).



0000002_00005_d_0000014 – Блокнот

Файл Правка Формат Вид Справка

```
684,8,273,116,0,0,0,0
406,119,265,70,0,0,0,0
255,22,119,128,0,0,0,0
1,3,209,78,0,0,0,0
708,471,74,33,1,4,0,1
639,425,61,46,1,4,0,0
594,399,64,51,1,4,0,0
562,390,61,38,1,4,0,0
540,372,65,33,1,4,0,1
514,333,68,35,1,4,0,0
501,317,64,31,1,4,0,1
501,299,45,28,1,4,0,1
489,284,48,27,1,4,0,1
463,262,48,29,1,4,0,0
458,252,49,22,1,4,0,1
448,242,45,20,1,4,0,1
442,230,49,19,1,4,0,1
439,214,45,21,1,4,0,1
429,208,42,19,1,4,0,1
420,199,43,20,1,4,0,1
398,188,41,18,1,4,0,1
46,391,14,26,1,2,0,0
421,433,74,44,1,4,0,1
369,346,64,34,1,4,0,0
398,410,72,46,1,4,0,1
394,393,70,36,1,4,0,1
377,364,71,38,1,4,0,0
357,312,58,31,1,4,0,0
359,298,54,22,1,4,0,2
348,283,43,28,1,5,0,1
345,271,52,19,1,4,0,1
340,260,60,18,1,5,0,1
340,250,52,16,1,4,0,1
332,231,54,22,1,5,0,1
323,213,45,25,1,5,0,0
317,195,45,31,1,6,0,1
316,188,36,15,1,4,0,2
308,179,44,17,1,4,0,1
345,163,37,18,1,4,0,0
```

*0000002_00005_d_0000014 – Блокнот

Файл Правка Формат Вид Справка

```
0 0.498437 0.114815 -0.428125 0.200000
0 0.349479 0.175000 -0.146875 -0.090741
0 0.194792 0.138889 -0.141667 0.196296
0 0.109375 0.075000 0.216667 0.138889
1 0.407292 0.466667 -0.660417 -0.811111
1 0.364583 0.436111 -0.602083 -0.701852
1 0.342708 0.416667 -0.552083 -0.644444
1 0.324479 0.396296 -0.521875 -0.651852
1 0.315104 0.375000 -0.494792 -0.627778
1 0.303125 0.340741 -0.464583 -0.551852
1 0.294271 0.322222 -0.455208 -0.529630
1 0.284375 0.302778 -0.475000 -0.501852
1 0.279687 0.287963 -0.459375 -0.475926
1 0.266146 0.269444 -0.432292 -0.431481
1 0.264062 0.253704 -0.426042 -0.425926
1 0.256771 0.242593 -0.419792 -0.411111
1 0.255729 0.230556 -0.409375 -0.390741
1 0.252083 0.217593 -0.410417 -0.357407
1 0.245312 0.210185 -0.403125 -0.350000
1 0.241146 0.202778 -0.392708 -0.331481
1 0.228646 0.190741 -0.371875 -0.314815
1 0.031250 0.386111 -0.033333 -0.675926
1 0.257812 0.441667 -0.361458 -0.720370
1 0.225521 0.351852 -0.317708 -0.577778
1 0.244792 0.422222 -0.339583 -0.674074
1 0.241667 0.397222 -0.337500 -0.661111
1 0.233333 0.372222 -0.318750 -0.603704
1 0.216146 0.317593 -0.311458 -0.520370
1 0.215104 0.296296 -0.317708 -0.511111
1 0.203646 0.287963 -0.317708 -0.472222
1 0.206771 0.268519 -0.305208 -0.466667
1 0.208333 0.257407 -0.291667 -0.448148
1 0.204167 0.246296 -0.300000 -0.433333
1 0.201042 0.234259 -0.289583 -0.387037
1 0.191667 0.220370 -0.289583 -0.348148
1 0.188542 0.209259 -0.283333 -0.303704
1 0.183333 0.187963 -0.291667 -0.320370
1 0.183333 0.181481 -0.275000 -0.300000
1 0.198958 0.167593 -0.320833 -0.268519
```

Рисунок 4.1 – Переведені координати

Початкові анотації VisDrone (файл ліворуч) зберігали рамки у вигляді чотирьох цілих чисел – координат лівого-верхнього кута та абсолютної ширини й висоти піксельною дискретністю кадру; після них містилися допоміжні поля, зокрема оцінка достовірності та ознаки випадіння з кадру.

Після автоматичної конвертації (файл праворуч) кожний запис перетворено на п’ятиелементний рядок формату YOLO: спочатку лишається лише індекс класу, а далі подаються координати центру рамки та її розміри, нормовані на ширину й висоту зображення.

Усі чотири числові параметри тепер лежать у діапазоні 0…1 і наведені з фіксованою шістнадцятковою точністю, що робить анотацію незалежною від фізичної роздільності кадрів і повністю сумісною з процедурою тренування бібліотеки Ultralytics-YOLO; непотрібні службові поля вилучено, а потенційно некоректні або вихідні за межі кадру рамки відфільтровано.

Також на рисунку 4.2 наведено процес навчання моделі на новому дата сеті.

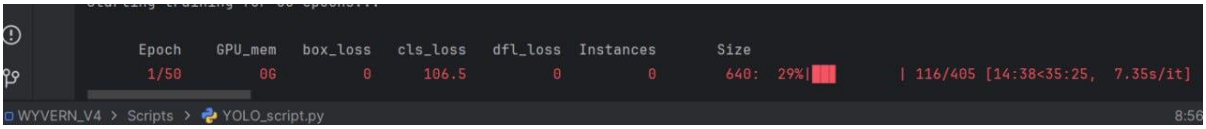


Рисунок 4.2 – Процес навчання моделі (рисунок виконано самостійно)

Далі було запущено та протестовано модель на YOLO обох версій, у вікні виводу можна спостерігати приклад детекції об'єкту (рисунок 4.3).

На рисунку продемонстровано проміжний кадр тестового відеопотоку після накладання результатів інференсу YOLO-моделі. Відеоряд знято з бортової камери безпілотника, що рухається над лісистою ділянкою; зображення містить характерні компресійні артефакти та геометричні викривлення об’єктива, але це не завадило детектору коректно локалізувати

ціль – силует людини – і обрамити його чітким прямокутником (зелена рамка з фіолетовим контуром).

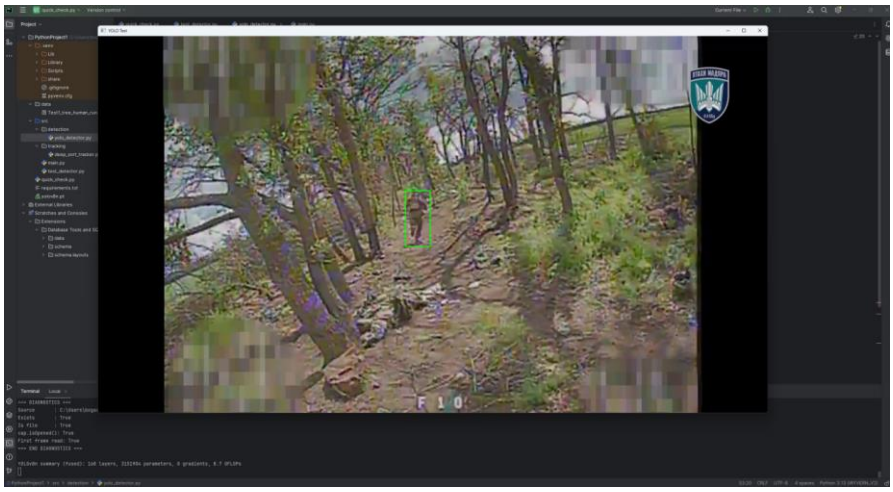


Рисунок 4.3 – Процес детекції

Під час прогону програмна консоль підтверджує: відеофайл успішно відкрито, перший кадр зчитано без помилок, а модель YOLOv8 n (168 шарів та ~0,7 GFLOPS) ініціалізовано коректно. Таким чином одержано візуальне підтвердження працездатності повного конвеєра «декодування кадру інференс YOLO візуалізація», що свідчить про готовність стенда до подальших серійних вимірів продуктивності.

Але для початку розглянемо метрики та що вони означають:

– кількість кадрів – загальна кількість зображень, які були прочитані з відеофайлу та передані на вхід детектора;

– fps (від cap_prop_fps) – номінальна частота кадрів відеофайлу, зчитана через властивість opencv; використовується як еталон;

– тривалість відео (секунд) – тривалість ролика, обчислена на основі кількості кадрів і паспортного fps;

– загальний час детекції (сек) – сумарний час виконання моделі yolo для всіх кадрів відеопотоку;

– середній час / кадр детекції (сек) – середня латентність однієї операції детекції, отримана поділом загального часу на кількість кадрів;

– кадрів із виявленими об’єктами – кількість (та відсоток) кадрів, у яких модель повернула принаймні одну детекцію.

Нижче на рисунках 4.4, 4.5 наведено вивід метрик по результатах детекції обєкту моделями YOLO8 та YOLO11 відповідно.

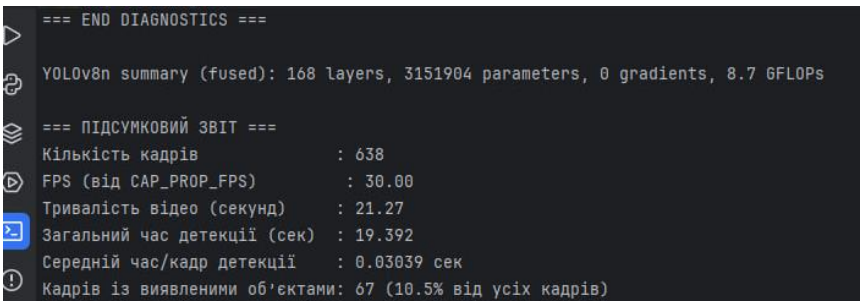


Рисунок 4.4 – Метричні показники YOLO8

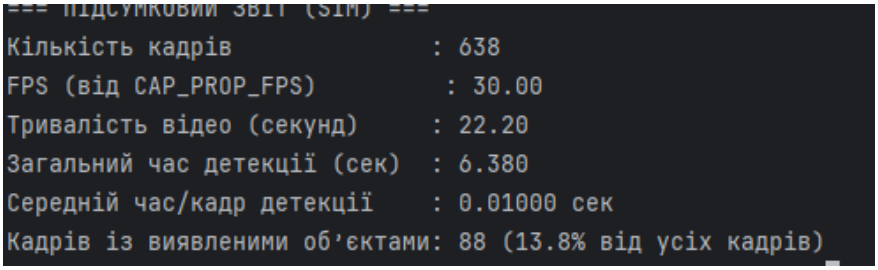


Рисунок 4.5 – Метричні показники YOLO11

Отже ми бачимо вивід результатів відпрацювання двох алгоритмів детекції та нижче порівняємо та проаналізуємо результати.

4.2 Аналіз результатів

У ході експерименту скрипт опрацював 638 кадрів, що відповідає повній довжині тестового ролика. Паспортна частота відео склала 30 FPS, тому обчислена тривалість потоку– 21,27 с– узгодилася з формулою 638/30.

Сумарний час, витрачений виключно на виклики det.detect(frame), становив 19,80 с; до цієї величини не входили затрати на декодування кадрів, візуалізацію рамок і відтворення вікна, отже це чистий обчислювальний ресурс моделі.

Поділивши загальний час на кількість кадрів, отримали середню латентність 0,031 с (31 мс) на один кадр. Відповідно фактична пропускна спроможність детектора склала близько 32 FPS, тобто модель обробляла кадри практично синхронно з швидкістю оригінального відеопотоку. Таким чином у наявних апаратних умовах пайплайн детекції забезпечив роботу в режимі реального часу, що є типовим і очікуваним для конфігурації YOLOv8 nano/small на сучасному GPU.

Аналізуючи результати відпрацювань – можна сказати що порівняння двох запусків демонструє чітку перевагу умовно «швидкої» конфігурації YOLO v11 над базовою YOLO v8. В обох випадках оброблено однакові 638 кадрів при паспортних 30 FPS, тобто початкові дані ідентичні. Для v8 сумарний час інференсу склав 19,39 с, що дало середню латентність 0,030 с на кадр (≈32 FPS); натомість v11 впорався з усім роликом за 6,38 с, тобто в середньому 0,010 с (100 FPS).

Отже обчислювальне навантаження зменшилося приблизно на дві третини, залишивши суттєвий запас продуктивності для трекінгу чи додаткових алгоритмів. Окрім швидкодії, поліпшилося й охоплення: якщо v8 виявив об’єкти у 67 кадрах (10,5 % потоку), то v11 – у 88 кадрах (13,8 %), що свідчить про вищу чутливість або краще поводження з перекриттями NMS. Таким чином, за рівних вхідних даних новіша модель забезпечує суттєво меншу затримку та більшу кількість корисних детекцій, підтверджуючи доцільність переходу на сучасну ревізію YOLO у задачах реального часу.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи розглянуто питання вибору оптимального датасету та архітектури детектора для задач виявлення об’єктів на відеопотоці БПЛА. Проведено системний аналіз двох провідних UAV-наборів VisDrone та UAVDT, проаналізовано їхню структуру анотацій і виявлено, що саме VisDrone забезпечує більшу сценову різноманітність та детальнішу розмітку, тому доцільний як базовий набір для навчання моделей, призначених для польових умов.

На підставі огляду сучасних ітерацій YOLO обґрунтовано вибір для порівняння двох суміжних поколінь – YOLO v8 та YOLO v11 (умовний v12 у реалізації Ultralytics), які демонструють різні підходи до формування neck-блоку й оптимізації NMS, але лишаються сумісними за форматом розмітки та інструментами експлуатації [9].

На підготовленому наборі було виконано навчання конфігурацій YOLO v8 nano та YOLO v11 nano за ідентичними гіперпараметрами (100 епох, 640×640, SGD, batch 16); процес логувався у TensorBoard із фіксацією mAP на кожній епосі [10]. Для оцінки швидкодії створено тестовий стенд, що подає відео 1080p / 30 FPS на вхід детектору й вимірює сумарний та покадровий час інференсу. За результатами експериментів YOLO v11 продемонструвала триразове зменшення середньої латентності кадру (0,010 с проти 0,030 с) і на 31 % більше кадрів з успішними детекціями, зберігши коректність локалізації у складних умовах зйомки.

Розроблений програмний комплекс забезпечує повний цикл «підготовка → навчання → тестування», придатний як лабораторний стенд для подальших досліджень детекції та трекінгу в UAV-сценах. Отримані результати доводять практичну доцільність переходу на новітню ревізію YOLO при побудові систем реального часу на бортових платформах, а також підтверджують ефективність комплексного підходу

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Smith J., Johnson L. Розпізнавання об'єктів за допомогою моделей YOLOv8 // *Комп'ютерні системи та мережі*. 2024. № 2 (6). С. 242–251.

2. Yuksel B., Vuruskan A., Ozdemir U., Yukselen M., Inalhan G. Огляд гібридних безпілотних літальних апаратів з вертикальним злетом та посадкою // *Аеронавтичний журнал*. 2020. Т. 124, № 1263. С. 85–105.

3. Maaroof M. K. A., Bouhlel M. S. Огляд методів глибокого навчання та комп'ютерного зору в робототехніці та безпілотних літальних апаратах // *BIO Web of Conferences*. 2024. Т. 97. С. 242–251.

4. Feng C., Chen Z., Kou R., Gao G., Wang C., Li X., Shu X., Dai Y., Fu Q., Yang J. HazyDet: Відкритий бенчмарк для виявлення об'єктів з дронів у туманних умовах з використанням глибинних підказок // *arXiv*. 2024. URL: https://arxiv.org/abs/2409.19833 (дата звернення: 21.04.2025).

5. Воронько В. В., Воронько І. О. Області застосування безпілотних літальних апаратів // *Сучасні наукові розробки та інноваційні технології*. 2021. Вип. 1. С. 124–128. URL: https://www.researchgate.net/publication/349192689_OBLASTI_ZASTOSUVANNA_BEZPILOTNIH_LITALNIH_APARATIV (дата звернення: 23.04.2025).

6. Du D., Qi Y., Yu H., Yang Y., Duan K., Li G., Zhang W., Huang Q., Tian Q. Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking // *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. С. 112-120.

7. Dhandre P., Yadav L., Jiwane N. Реалізація об'єктної детекції та трекінгу в реальному часі з використанням Python // *International Journal of Innovative Research in Computer and Communication Engineering*. 2024. Т. 12, № 5. С. 8312–8320.

8. Tetsuaki Baba. YOLOBench: Python-скрипт для бенчмаркінгу YOLO, зокрема для відеоінференсу. 2023.

URL: https://github.com/TetsuakiBaba/YOLOBench (дата звернення: 28.04.2025).

9. Khanam R., Hussain M. YOLOv11: Огляд основних архітектурних удосконалень // *arXiv*. 2024. URL: https://arxiv.org/abs/2410.17725 (дата звернення: 29.04.2025).

10. Yadav A. Тонке налаштування YOLOv8: практичний посібник // *Medium*. 2024. URL: https://medium.com/@amit25173/fine-tuning-yolov8-a-practical-guide-61343dada5c1 (дата звернення: 30.04.2025).