Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

Кваліфікаційна робота освітній ступінь – магістр

на тему: **“Автоматизована система підтримки прийняття рішень для супроводу та ідентифікації об'єктів у просторі на основі комп'ютерного зору та аналітики подій**”

Виконав: студент 2-го року навчання
освітньої програми «Комп’ютерні науки»,
спеціальності 122 Комп’ютерні науки
Мороз Дмитро Валентинович
Керівник: Волинець Є. А.,
старший викладач
Рецензент: (TODO).,

Кваліфікаційна робота захищена
з оцінкою _____
Секретар ЕК _____
«____» _____ 20____ р.

Київ 2026

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
доцент, к.ф-м.н.,
С. С. Гороховський

_____
(підпис)

„____”_____2026 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на дипломну роботу

студенту 2-го курсу, факультету інформатики
Морозу Дмитру Валентиновичу
Розробити Автоматизовану систему для ідентифікації та супроводу об'єктів у просторі на основі комп'ютерного зору
Зміст ТЧ до магістерської роботи:

Зміст
Анотація
Вступ
1 Аналіз предметної області
2 Архітектура та технології розробки
3 Створення датасету
4 Розробка програмних компонентів для навчання та валідації системи відстеження об’єктів
5 Оцінка ефективності та можливості розвитку

Висновки

Список літератури

Додатки

Дата видачі „___” _____ 2025 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Графік підготовки кваліфікаційної роботи до захисту

Графік узгоджено «_____» _____ 2025 р.

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	01.10.2025	
2.	Огляд технічної літератури за темою роботи	15.10.2025	
3.	Аналіз існуючих рішень та формулювання вимог до платформи	01.11.2025	
4.	Збір та попередній аналіз даних для навчання	15.12.2025	
5.	Реалізація MVP-версії програмного коду для тренування і валідації системи відстеження	15.01.2026	
6.	Розширення варіативності даних, експерименти з оптимізації гіперпараметрів навчання детектора об'єктів	10.02.2026	
7.	Створення середовища для збору і аналізу метрик з відстеження об'єктів	25.02.2026	
8.	Розробка модифікації реалізації трекера ByteTrack для трекінгу об’єктів за допомогою ключових точок на фрагменті картинки	15.03.2026	

9.	Проведення тестування, підбір гіперпараметрів трекінгу, доопрацювання датасету	25.03.2026	
10.	Написання пояснювальної записки	15.04.2026	
11.	Створення слайдів для доповіді та чорнової версії доповіді	25.04.2026	
12.	Попередній захист, отримання рецензій, аналіз зауважень	29.04.2026	
13.	Остаточне оформлення роботи та презентації	15.05.2026	
14.	Захист магістерської роботи (проекту)		

ЗМІСТ

Стор.

АНОТАЦІЯ 7
ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10
1.1 Проблематика автоматизації роботи БПЛА за умов втрати зв'язку.... 10
1.2 Огляд існуючих рішень для відстеження об’єктів у реальному часі 11
1.3 Способи оцінки ефективності роботи системи відстеження об'єктів 13
Висновки до розділу 1 14
РОЗДІЛ 2. СТВОРЕННЯ ДАТАСЕТУ..... 15
2.1 Збір та розмітка відеоматеріалів..... 15
2.2 Конвеєр очищення відео 19
2.3 Формування картинкового датасету на основі відео..... 22
Висновки до розділу 2 26
РОЗДІЛ 3. АРХІТЕКТУРА ТА ТЕХНОЛОГІЇ РОЗРОБКИ 27
3.1 Загальна архітектура системи 27
3.2 Вибір технологій 30
Висновки до розділу 3 32
РОЗДІЛ 4. НАВЧАННЯ ДЕТЕКТОРА YOLO ……..... 33
4.1 Підбір гіперпараметрів та навчання 33
4.2 Валідація та аналіз метрик детектора 35
Висновки до розділу 4 43
РОЗДІЛ 5. ОЦІНКА ЕФЕКТИВНОСТІ ВІДСТЕЖЕННЯ ОБ'ЄКТІВ ТА ПОДАЛЬШИЙ РОЗВИТОК 44
5.1 Принцип збору метрик з відстеження об'єктів 44
5.2 Порівняння класичного ByteTrack та розробленої модифікації 50
5.3 Перспективи розвитку та шляхи оптимізації 54
Висновки до розділу 5 56
ВИСНОВКИ 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60

АНОТАЦІЯ

У магістерській роботі представлено процес розробки автоматизованої системи підтримки прийняття рішень для ідентифікації та відстеження наземних об'єктів у реальному часі на основі комп'ютерного зору. Систему спроєктовано для розгортання на борту безпілотного літального апарату з орієнтацією на автономну роботу в умовах радіоелектронної боротьби та відсутності зв'язку з оператором.

У ході роботи було сформовано власний датасет на основі відеоматеріалів з відкритих джерел, розроблено конвеєр очищення відеоданих від небажаних артефактів та реалізовано процес ручної розмітки у середовищі CVAT. Навчання моделі виявлення об'єктів виконано на базі згорткової нейронної мережі YOLO [1] з використанням аугментації даних. Для відстеження об'єктів між кадрами та присвоєння їм унікальних ідентифікаторів інтегровано алгоритм ByteTrack [2] та розроблена його модифікація. Особливу увагу приділено оптимізації системи з метою подальшого портування на мікрокомп'ютери з обмеженими обчислювальними ресурсами. Результати тестування підтверджують здатність системи ефективно ідентифікувати наземну техніку в різних погодних умовах.

ВСТУП

Сучасні збройні конфлікти дедалі більше визначаються не лише чисельністю особового складу чи кількістю техніки, а й здатністю сторін ефективно збирати, обробляти та використовувати розвідувальні дані в режимі реального часу. Безпілотні літальні апарати стали одним із ключових інструментів тактичної розвідки та ураження цілей завдяки своїй

маневреності, доступності та здатності діяти в небезпечних умовах без ризику для оператора.

Проте ефективність застосування БПЛА суттєво обмежується в умовах радіоелектронної боротьби. Глушіння каналів керування та навігаційних сигналів, зокрема GPS, позбавляє оператора можливості дистанційно керувати апаратом і отримувати розвідувальні дані. Окремою проблемою є втрата зв'язку на малих висотах польоту через радіогоризонт, що особливо критично в умовах міської або горбистої місцевості. Це актуалізує потребу в бортових системах, здатних діяти автономно — без постійного зв'язку з оператором.

Зважаючи на ці обмеження, перспективним напрямком є розробка бортових систем автоматичного виявлення та відстеження об'єктів на основі комп'ютерного зору. У науковій літературі та інженерній практиці поширеними є підходи, що базуються на згорткових нейронних мережах для детекції об'єктів [1] та алгоритмах багатооб'єктного відстеження для підтримки їх ідентифікації між кадрами [2]. Інтеграція систем машинного зору безпосередньо на борту літальних апаратів є активною зоною наукових досліджень і інженерних розробок, що відображає зростаючий попит на автономні рішення в умовах сучасних збройних конфліктів.

У зв'язку з цим, доцільною є розробка системи, яка поєднуватиме детекцію наземної техніки за допомогою нейронної мережі YOLO з алгоритмом відстеження ByteTrack для присвоєння унікальних ідентифікаторів об'єктам між кадрами. Така система має працювати в режимі реального часу, функціонувати автономно без зовнішнього зв'язку та бути придатною для портування на мікрокомп'ютери з обмеженими обчислювальними ресурсами.

У межах роботи запропоновано та реалізовано систему виявлення і відстеження наземних об'єктів, яка включає власний датасет військової техніки, навчену модель детекції та інтегрований модуль відстеження з підтримкою унікальної ідентифікації об'єктів.

Робота складається з п'яти розділів. У першому подано аналіз проблематики застосування БПЛА в умовах РЕБ та огляд існуючих рішень з відстеження об'єктів. У другому — описано архітектуру системи та обґрунтовано вибір технологій. У третьому — розглянуто процес формування власного датасету. Четвертий розділ присвячено розробці програмних компонентів для навчання та валідації системи. П'ятий — оцінці ефективності та перспективам розвитку.

Об’єкт дослідження: процес автоматичної ідентифікації та відстеження наземних об'єктів у відеопотоці з борту БПЛА.

Предмет дослідження: архітектура, алгоритми і технології виявлення та відстеження наземної техніки в режимі реального часу на основі комп'ютерного зору в умовах обмежених обчислювальних ресурсів.

Мета роботи: розробити автоматизовану бортову систему підтримки прийняття рішень для ідентифікації та відстеження наземних об'єктів з використанням орієнтовану на автономну роботу без зовнішнього зв'язку. Задача сводиться до здатності системи стабільно локалізувати цілі в 2D просторі пікселів зображення з відео, продукуючи параметри обмежувальної рамки (Bounding Box) об'єктів та забезпеченні їх безперервного відстеження з присвоєнням унікального ідентифікатора (ID) для збереження цілісності спостереження між кадрами.

Постановка задачі:

1. Провести аналіз проблематики застосування БПЛА в умовах радіоелектронної боротьби, огляд існуючих рішень з виявлення і відстеження об'єктів та вибір підходів по оцінці ефективності роботи системи.
2. Сформулювати технічні вимоги до бортової системи та обґрунтувати вибір архітектури і технологій розробки.
3. Сформувати власний датасет наземної військової техніки, включаючи збір, очищення та розмітку відеоматеріалів.

4. Розробити та навчити модель детекції об'єктів на основі YOLO з інтеграцією алгоритму відстеження ByteTrack для підтримки унікальної ідентифікації між кадрами.
5. Здійснити оцінку точності та продуктивності системи, виявити потенційні недоліки.
6. Запропонувати напрями подальшого розвитку системи та її адаптації до розгортання на мікрокомп'ютерах.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика автоматизації роботи БПЛА за умов втрати зв'язку

У Сучасне поле бою дедалі більше визначається не лише фізичною присутністю техніки та особового складу, а й здатністю сторін контролювати електромагнітний простір. Безпілотні літальні апарати стали невід'ємним елементом тактичної розвідки, коригування вогню та ураження цілей завдяки своїй маневреності й відносній доступності. Проте саме їхня залежність від радіоканалів керування та навігаційних сигналів робить їх вразливими до засобів радіоелектронної боротьби.

Радіоелектронна боротьба (РЕБ) — це сукупність заходів, спрямованих на порушення, пригнічення або перехоплення електромагнітних сигналів противника. У контексті застосування БПЛА засоби РЕБ діють за кількома напрямками:

– Глушіння каналу керування. Постановники завад генерують шум у діапазонах частот, які використовуються для передачі команд від оператора до апарату, що унеможливлює дистанційне керування; – Придушення сигналів GPS/GNSS. Глушіння навігаційних сигналів позбавляє БПЛА можливості визначати своє місцеположення, що критично для автоматичного утримання позиції та прокладання маршруту; – Спуфінг навігаційного сигналу. Більш складний вид впливу, за якого замість придушення сигналу здійснюється його підміна хибними координатами, що

може призвести до відведення апарату у заздалегідь визначену зону або його втрати.

Окремою проблемою, не пов'язаною безпосередньо з РЕБ, є втрата зв'язку через радіогоризонт. На малих висотах польоту — характерних для розвідувальних місій у міській або горбистій місцевості — радіохвилі не можуть поширюватись за межі прямої видимості між антеною оператора та апаратом. Це фізичне обмеження унеможливлює передачу відеопотоку та команд керування навіть за відсутності активної протидії з боку противника. Особливо критичним цей ефект є для ударних БПЛА: під час кінцевого зниження для ураження цілі апарат неминуче опускається нижче лінії радіогоризонту, що призводить до втрати зв'язку з оператором саме на найвідповідальнішому етапі місії. Неможливість вносити корективи в траєкторію в режимі реального часу безпосередньо знижує відсоток успішних влучань і зменшує бойову ефективність застосування таких апаратів.

Із огляду на вищезазначене, покладатись виключно на дистанційне керування в умовах сучасного збройного конфлікту є недостатнім. Втрата зв'язку — незалежно від її причини — фактично перетворює БПЛА на некерований об'єкт, нездатний виконувати завдання. Це актуалізує потребу в бортових системах автономної обробки інформації, які дозволяють апарату продовжувати виконання завдань з ідентифікації та відстеження цілей без участі оператора і без залежності від зовнішніх каналів зв'язку чи навігації.

1.2 Огляд існуючих рішень

Задача автоматичного виявлення та відстеження об'єктів у відеопотоці є однією з центральних у галузі комп'ютерного зору. За останнє десятиліття підходи до її вирішення зазнали суттєвої еволюції — від класичних методів виділення ознак до глибоких нейронних мереж, здатних працювати в режимі реального часу.

Ранні підходи до детекції об'єктів базувались на двоетапних архітектурах — зокрема сімействі R-CNN [3] — де спочатку генерувались регіони-кандидати, а потім класифікувались окремою мережею. Такі підходи демонструють високу точність, однак через значні обчислювальні витрати є непридатними для застосувань реального часу на обладнанні з обмеженими ресурсами.

Одноетапні детектори, такі як SSD [4] та RetinaNet [5], запропонували компроміс між швидкістю та точністю, виконуючи детекцію за один прохід через мережу. Найбільшого поширення серед одноетапних підходів набуло сімейство YOLO [1], яке з кожною версією послідовно покращувало баланс між продуктивністю та точністю. Сучасна версія YOLOv11 є актуальним представником цього сімейства, що підтримує ефективну детекцію об'єктів у реальному часі.

Задача відстеження об'єктів між кадрами (Multi-Object Tracking, MOT) традиційно вирішується за парадигмою tracking-by-detection, де детектор і трекер функціонують як окремі модулі. Алгоритм SORT [6] запропонував поєднання фільтра Калмана для передбачення положення об'єкта та угорського алгоритму для зіставлення детекцій між кадрами. Незважаючи на простоту та високу швидкодію, SORT втрачає треки при тимчасовому зникненні об'єктів або перекритті. DeepSORT [7] розширив цей підхід, додавши модуль повторної ідентифікації об'єктів за їхнім зовнішнім виглядом, що підвищило стійкість відстеження, однак збільшило обчислювальне навантаження через необхідність додаткової нейромережі.

ByteTrack [2] запропонував інший підхід до вирішення проблеми втрати треків: замість відкидання низькодостовірних детекцій алгоритм використовує їх для підтримки існуючих треків на етапі повторного зіставлення. Це дозволяє зберігати стійкість відстеження навіть коли детектор тимчасово знижує впевненість у присутності об'єкта. Разом з тим, підходи класу tracking-by-detection загалом залишаються залежними від якості детектора — при систематично низькій впевненості детекцій, наприклад через обмежений обсяг навчальних даних, стійкість трекера

суттєво погіршується. Легковагові методи зіставлення ознак, зокрема на основі дескрипторів AKAZE та ORB у поєднанні з геометричною верифікацією через RANSAC, є перспективним напрямком для компенсації цього обмеження без залучення додаткових нейромережевих компонентів.

1.3 Способи оцінки ефективності системи відстеження об'єктів

Оцінка якості систем багатооб'єктного відстеження є нетривіальною задачею, оскільки такі системи поєднують два взаємопов'язані процеси — детекцію об'єктів та їх асоціацію між кадрами. Помилки в кожному з них впливають на кінцевий результат по-різному, що зумовлює необхідність використання набору спеціалізованих метрик, кожна з яких висвітлює окремий аспект роботи системи. Стандартизовані підходи до оцінки трекінгу сформульовано в рамках MOT Challenge [8] — відкритого бенчмарку, який визначив основний набір метрик і протоколів порівняння алгоритмів відстеження.

Метрика MOTA (Multiple Object Tracking Accuracy) є однією з найпоширеніших для загальної оцінки якості трекера.

Вона агрегує три типи помилок: хибнопозитивні детекції (FP), пропущені детекції (FN) та переключення ідентифікаторів між треками (IDSW). Висока значення MOTA свідчить про низький рівень помилок усіх трьох типів одночасно, однак метрика чутлива до дисбалансу між ними і не розрізняє внесок детектора та трекера окремо.

Метрика MOTP (Multiple Object Tracking Precision) оцінює просторову точність локалізації об'єктів — тобто наскільки точно обмежувальні рамки збігаються з реальним положенням об'єктів. На відміну від MOTA, MOTP не враховує кількість помилок асоціації і відображає виключно якість позиціонування.

IDF1 (Identity F1 Score) оцінює здатність системи зберігати стійку ідентифікацію об'єктів протягом усієї послідовності кадрів. Метрика базується на співвідношенні правильно ідентифікованих детекцій до

загальної кількості детекцій з урахуванням як точності так і повноти. Висока значення IDF1 свідчить про те, що система не лише виявляє об'єкти, а й послідовно підтримує їхні ідентифікатори між кадрами без зайвих переключень.

HOTA (Higher Order Tracking Accuracy) [9] є більш сучасною метрикою, що усуває обмеження MOTA шляхом явного розділення внесків детекції та асоціації. Вона обчислюється як геометричне середнє двох складових: DetA (Detection Accuracy) — що відображає якість виявлення об'єктів, та AssA (Association Accuracy) — що оцінює якість їх зіставлення між кадрами. Такий підхід дозволяє точніше діагностувати де саме система припускається помилок — на рівні детектора чи трекера.

Окрім агрегованих метрик, для детального аналізу використовуються також кількісні показники: FP (хибнопозитивні детекції), FN (пропущені детекції), IDSW (кількість переключень ідентифікаторів), MT (mostly tracked — об'єкти що відстежуються більш ніж у 80% кадрів своєї присутності) та ML (mostly lost — об'єкти що відстежуються менш ніж у 20% кадрів). Ці показники дозволяють інтерпретувати причини відхилень в агрегованих метриках.

Для практичного підрахунку описаних метрик у даній роботі використовується бібліотека TrackEval [10], яка реалізує стандартизовані протоколи оцінки та підтримує всі розглянуті метрики в єдиному інструменті.

Висновки до розділу 1

У першому розділі було проаналізовано предметну область автоматизованої ідентифікації та відстеження наземних об'єктів в контексті бортових систем безпілотних літальних апаратів. Встановлено, що ефективне виконання розвідувальних та ударних місій в умовах сучасного збройного конфлікту суттєво ускладнюється через вплив засобів

радіоелектронної боротьби та фізичні обмеження радіогоризонту, що актуалізує потребу в автономних бортових системах обробки візуальної інформації без залежності від зовнішніх каналів зв'язку.

Проведено огляд існуючих підходів до детекції та відстеження об'єктів у відеопотоці. Показано еволюцію від двоетапних архітектур до одноетапних детекторів сімейства YOLO як домінуючого підходу для задач реального часу. Серед алгоритмів відстеження розглянуто SORT, DeepSORT та ByteTrack, визначено їхні переваги та обмеження. Встановлено, що підходи класу tracking-by-detection залишаються залежними від якості детектора, що є критичним в умовах обмеженого обсягу навчальних даних. Визначено перспективність легковагових методів зіставлення ознак на основі дескрипторів AKAZE та ORB у поєднанні з геометричною верифікацією через RANSAC як засобу підвищення стійкості трекінгу без залучення додаткових нейромережевих компонентів.

Розглянуто основні метрики оцінки якості систем багатооб'єктного відстеження: MOTA, MOTP, IDF1, HOTA, DetA, AssA, а також кількісні показники FP, FN, IDSW, MT та ML. Обґрунтовано доцільність використання бібліотеки TrackEval для стандартизованого підрахунку цих метрик у рамках даної роботи.

Таким чином, на основі проведеного аналізу сформульовано технічні передумови для розробки бортової системи виявлення та відстеження наземної техніки, що поєднує детектор YOLOv11 з модифікованим алгоритмом ByteTrack, орієнтованої на автономну роботу в умовах відсутності зовнішнього зв'язку.

РОЗДІЛ 2. СТВОРЕННЯ ДАТАСЕТУ

2.1 Збір та відбір відеоматеріалів

Процес Формування датасету розпочалось зі збору відеоматеріалів з відкритих джерел. Основним інструментом пошуку слугувала платформа Reddit, яка завдяки системі тематичних спільнот та пошуку за ключовими

словами дозволяє ефективно знаходити відеозаписи з конкретними типами техніки. Такий підхід дав змогу цілеспрямовано збалансувати датасет за класами об'єктів, уникаючи надмірного домінування окремих категорій на етапі збору.

При відборі матеріалів свідомо дотримувались різноманітності умов зйомки. Відеофрагменти охоплюють різні погодні умови — ясну погоду, хмарність, дощ, туман та зимові умови з сніговим покривом — з метою рівномірного покриття кожного з них. Географічно представлені три типи місцевості: відкрита місцевість, сільська забудова та міське середовище, хоча урбаністичні фрагменти становлять лише близько 5–7% від загального обсягу через обмежену доступність відповідних матеріалів у відкритому доступі.

Переважну більшість зібраних матеріалів — близько 90% — становлять записи реальних бойових дій. Це зумовлює ряд специфічних характеристик датасету: нерідко присутній дим, транспортні засоби інколи зафіксовані в стані горіння або пошкодження, зображення має різний ступінь розмитості через динамічний рух камери. Зйомка ведеться переважно з двох типів платформ: ударних FPV-дронів із динамічною камерою та дронів-розвідників зі стабілізованою статичною зйомкою. Це безпосередньо впливає на характер треків у датасеті — більшість з них є короткими, що пояснюється природою FPV-зйомки: апарат швидко зближується з ціллю і уражає її, після чого об'єкт зникає з кадру.

Загалом було зібрано 855 відеофрагментів різної тривалості із загальним хронометражем 88.1 хвилини при частоті кадрів 30 fps та роздільній здатності 1280×720 пікселів. Нерідко одні й ті самі локації фігурують у різних відеофрагментах, що є природним наслідком використання відкритих джерел і враховувалось при подальшій розмітці. Весь процес збору та відбору матеріалів виконувався вручну, що вимагало значних часових витрат але забезпечило контроль якості на кожному етапі формування датасету.

У контексті відеорозмітки під терміном **трек** розуміється послідовність анотованих обмежувальних рамок одного й того самого об'єкта впродовж кількох послідовних кадрів відеофрагменту. Кожному треку присвоюється унікальний ідентифікатор у межах відео, що дозволяє відрізняти різні екземпляри одного класу та відстежувати переміщення конкретного об'єкта в часі. Таким чином, один відеофрагмент може містити кілька треків різних об'єктів, а загальна кількість анотованих кадрів відображає сумарну кількість поява усіх об'єктів у всіх відео.

Таблиця 2.1

Загальна статистика відео-датасету

Параметр	**Значення**
Загальна кількість відеофрагментів	855
Загальна тривалість відеоматеріалів	88.1 хв
Частота кадрів	30 fps
Роздільна здатність	1280×720 пікселів
Загальна кількість унікальних треків	1 337
Загальна кількість анотованих кадрів	158 512
Кількість перекритих анотацій (occluded)	17 309 (10.9%)
Кількість анотацій поза кадром (outside)	883 (0.6%)
Середня тривалість треку	118.6 кадрів
Медіанна тривалість треку	75 кадрів
Мінімальна тривалість треку	1 кадр
Максимальна тривалість треку	2 423 кадри

Таблиця 2.2

Розподіл треків за тривалістю

Кластер	**Діапазон(кадри)**	**Середня тривалість**	**Кількість треків**	**Частка**
Short	1 – 130	56.1	951	71.1%
Med-Short	131 – 369	205.2	329	24.6%

Med-Long	375 – 996	541.6	51	3.8%
Long	1 246 – 2 423	1 679.2	6	0.4%

Розподіл анотацій за розміром обмежувальної рамки наведено в таблиці 2.3. Розмір визначається як нормалізована площа рамки відносно площі кадру.

Таблиця 2.3

Розподіл анотацій за розміром об’єкта

Кластер	**Нормалізована площа**	**Середня тривалість**	**Частка**
Tiny	0.00014 – 0.08299	129 342	81.6%
Small	0.08301 – 0.26031	18 938	11.9%
Medium	0.26034 – 0.52682	7 691	4.9%
Large	0.52710 – 1.00000	2 541	1.6%

Розмітка зібраних відеоматеріалів виконувалась вручну у середовищі CVAT (Computer Vision Annotation Tool) [8] — інструменті з відкритим вихідним кодом, що розгортається локально у Docker-контейнері. Сторінку перегляду матеріалів і анотацій зображено на рисунку 2.1. Вибір CVAT обумовлений рядом практичних переваг: інструмент є безкоштовним, має просту процедуру розгортання, підтримує розмітку відеопослідовностей з присвоєнням унікальних ідентифікаторів трекам та забезпечує експорт анотацій у сучасних форматах, зокрема COCO, YOLO та MOT.

Ключовою функцією, що суттєво прискорила процес розмітки, стала інтерполяція обмежувальних рамок між кадрами. Замість того щоб вручну позначати положення об'єкта на кожному кадрі, достатньо було розмістити рамку на початковому та кінцевому кадрі треку — CVAT автоматично інтерполює проміжні положення. Це дозволило значно скоротити час розмітки при збереженні прийнятної точності анотацій, особливо для об'єктів з плавним рухом.

Розмітка виконувалась на рівні відеопослідовностей, а не окремих кадрів, що дозволило зберегти інформацію про треки — послідовності появ

одного об'єкта впродовж відео. Кожному об'єкту присвоювався клас із заздалегідь визначеного набору та унікальний ідентифікатор треку в межах відеофрагменту. Статистика сформованого датасету наведена в таблицях 2.1–2.3.

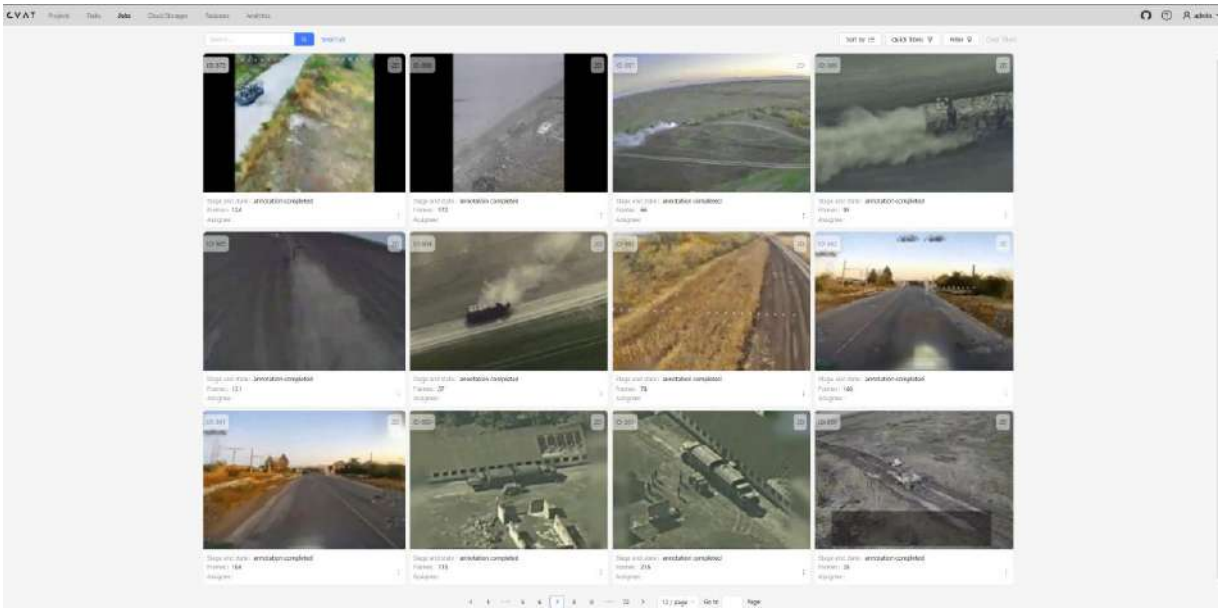

Рисунок. 2.1 – Сторінка перегляду відеоматеріалів в CVAT

2.2 Конвеєр очищення відео

Значна частина зібраних відеоматеріалів містила небажані візуальні артефакти — логотипи каналів, вотермарки, елементи інтерфейсу дрона (приціли, статус батареї, тощо) та інші накладені графічні елементи. Присутність таких артефактів у навчальних даних є небажаною, оскільки нейронна мережа може асоціювати їх з певними класами об'єктів або використовувати як хибні ознаки при навчанні. Для вирішення цієї проблеми було розроблено власний автоматизований конвеєр очищення відеоданих.

Конвеєр керується єдиним оркестратором та складається з трьох послідовних етапів. Для кожного відео вручну створювалась бінарна маска — статичне зображення, на якому білим кольором позначались області кадру що підлягають видаленню. Одна маска автоматично застосовується

до всіх сегментів відповідного відео, що спрощує процес обробки великої кількості файлів.

На першому етапі вихідне відео розбивається на рівні сегменти тривалістю 4 секунди за допомогою FFmpeg. Необхідність сегментації зумовлена апаратними обмеженнями: модель інпейнтингу завантажує кожен кадр у відеопам'ять GPU, тому обробка тривалого відео як єдиного цілого призводить до переповнення VRAM. При частоті 30 fps чотирисекундний сегмент містить 120 кадрів — обсяг сумісний із наявними апаратними ресурсами.

Рисунок. 2.2 – Кадр з відео до очистки конвеєром

На другому етапі кожен сегмент передається до моделі ProPainter [9] — інструменту відеоінпейнтингу, що відновлює вміст замаскованих областей шляхом аналізу сусідніх кадрів. Процес відновлення складається з чотирьох послідовних кроків. Спочатку модель RAFT обчислює оптичний потік між кадрами — попіксельні карти руху вперед і назад у часі. Потім рекурентна мережа Flow Completion Net відновлює напрямок руху фону всередині замаскованої зони на основі оточуючого потоку. Далі модуль просторово-часового перенесення пікселів заповнює порожні ділянки вмістом із незамаскованих кадрів. На фінальному кроці трансформерна

модель синтезує остаточне зображення, поєднуючи перенесені пікселі з нейромережевою генерацією відсутнього вмісту для кожного вікна з 10 сусідніх кадрів.

На третьому етапі оброблені сегменти об'єднуються в єдиний відеофайл за допомогою операції FFmpeg concat без перекодування потоку, що дозволяє зберегти якість зображення.

Розроблений конвеєр підтримує пакетну обробку файлів у черзі: оркестратор автоматично зіставляє відеофайли з відповідними масками за назвою файлу, що дозволяє завантажити довільну кількість відео і масок у відповідні папки та запустити обробку без подальшого ручного втручання. Завдяки цьому весь масив із 855 відеофрагментів було оброблено в автоматичному режимі. Через обчислювальну складність моделі ProPainter сумарна обробка 88.1 хвилини відеоматеріалів зайняла приблизно 20 годин безперервної роботи комп'ютера. Результат роботи конвеєру можна оцінити порівнявши кадр до та після обробки на рисунках 2.2 та 2.3 відповідно.

Рисунок. 2.3 – Кадр з відео після очистки конвеєром

2.3 Формування картинкового датасету на основі відео

Безпосереднє використання відеоанотацій для навчання детектора є неефективним через надлишковість відеоданих — сусідні кадри відео часто є майже ідентичними, що призводить до перенавчання моделі на повторюваних прикладах. На рисунку 2.4 зображено загальну кількість кадрів, на яких видно певні класи датасету. Для вирішення проблеми надлишкової дублікації було розроблено власний скрипт генерації картинкового датасету з відеокадрів, що реалізує чотирифазний алгоритм інтелектуальної вибірки.

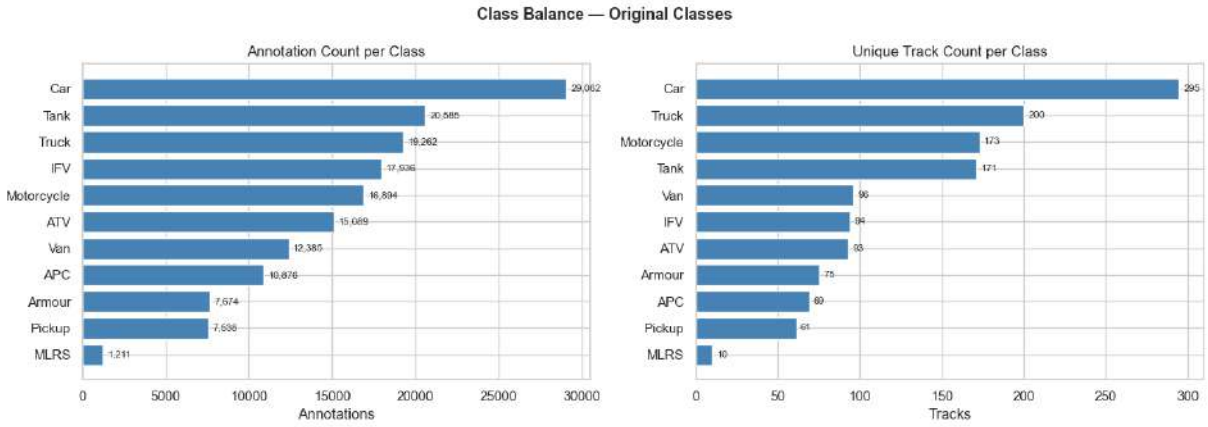

Рисунок. 2.4 – Оригінальний розподіл класів на кадрах відео, по обмежувальним рамкам та по трекам

Розбивка на train/val/test виконувалась вручну, а не випадковим чином. Val-сет містить 92 відео, test-сет — 29 відео, train-сет — 706 відео. Ручний відбір зумовлений специфікою зібраних матеріалів: різні відео можуть містити одні й ті самі локації або об'єкти, тому випадкова розбивка могла б помістити схожі відео одночасно в train і val, що суттєво спотворило б об'єктивність оцінки моделі.

Оригінальні 11 класів CVAT були свідомо об'єднані у 4 вихідних класи через їх візуальну схожість. (Рисунок 2.5) Це спростить задачу класифікатору, поєднуючи візуально-схожі класи в один супер-клас:

- Armor (Tank, IFV, APC, Armour)

- Truck (Truck, Van, MLRS)
- Car (Car, Pickup)
- LightVehicle (Motorcycle, ATV)

Таке групування зумовлене значним дисбалансом між класами у вихідному датасеті — окремі класи, зокрема MLRS, мають недостатньо прикладів для надійного навчання окремого класифікатора (Рисунок 2.4). Паралельно розглядався також підхід із єдиним класом "ціль (Target)", де всі об'єкти об'єднуються в один клас — це дозволяє зосередити детектор виключно на задачі локалізації без класифікації (Рисунок 2.6).

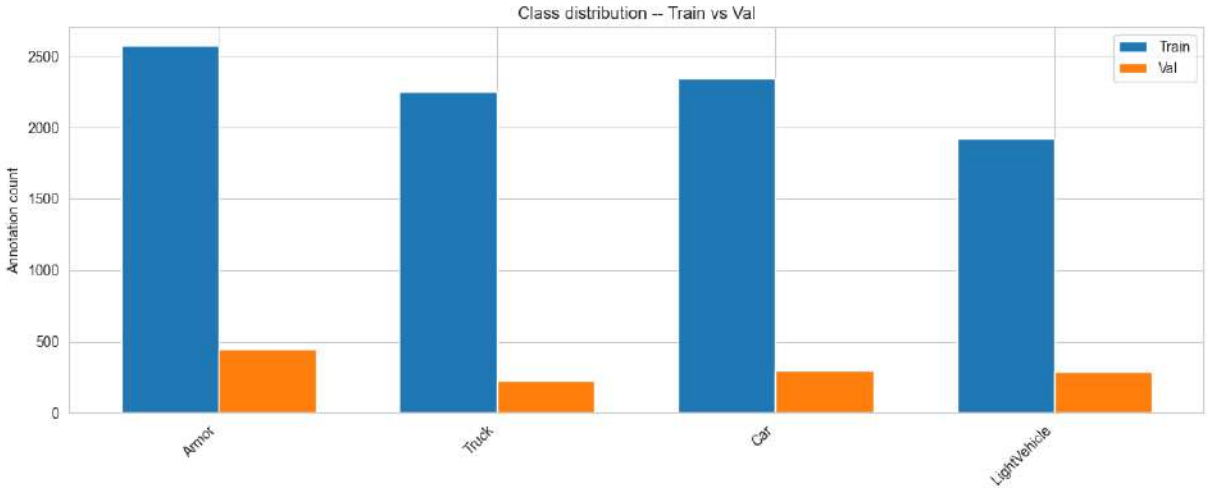

Рисунок. 2.5 – Розподіл класів при групуванні на 4 супер-класи об’єктів

Алгоритм вибірки кадрів складається з чотирьох послідовних фаз. На першій фазі виконується event-driven відбір — безумовно включаються кадри де відбувається значима подія по кожному треку: перша та остання поява об'єкта у кадрі, зміна стану перекриття (occluded) та перехід між scale-бакетами. Такі кадри несуть максимум інформації про стан треку і включаються до вибірки незалежно від результатів наступних фаз.

На другій фазі виконується scale-стратифікована дедуплікація. Простір розмірів об'єктів поділяється на чотири бакети за нормалізованою площею обмежувальної рамки: tiny, small, medium та large (див. Таблиця 2.4). Для кожного бакету незалежно відбираються кадри на основі порогу схожості між сусідніми кадрами — кадр пропускається якщо об'єкти майже

не змістились відносно попереднього збереженого кадру. Незалежна обробка бакетів є принциповою: без неї дрібні об'єкти були б недопредставлені, оскільки їхній рух не домінує над статикою великих об'єктів у кадрі.

Третя фаза виконує глобальну дедуплікацію об'єднаного набору кадрів з вказаним порогом схожості, усуваючи майже-дублікати що могли потрапити до вибірки через кілька бакетів одночасно. Порівняння виконуються базуючись на значенні IoU обмежувальних рамок одного і того ж треку між кадрами.

Таблиця 2.4

Бакет	**Діапазон площі (нормалізована площа)**	**Опис**
Tiny	[0.0, 0.003)	Дуже дрібні об'єкти
Small	[0.003, 0.012)	Малі
Medium	[0.012, 0.05)	Середні
Large	[0.05, 1.0)	Великі

Четверта фаза застосовує перцептивне хешування (pHash) як додатковий фільтр візуальної схожості. Для кожного бакету підтримується окремий пул збережених хешів, і новий кадр зберігається лише якщо його відстань Хеммінга від усіх раніше збережених хешів перевищує встановлений поріг. Event-кадри з першої фази обходять цей фільтр безумовно, гарантуючи що жодна значима подія не буде відфільтрована через візуальну схожість.

Для навчання моделі придушення хибних спрацювань до тренувального набору додано 262 фонових зображення без об'єктів. Їх отримано двома способами: автоматичним відбором кадрів без анотацій та вирізанням об'єктів із кадрів з подальшим відновленням фону сусіднім фрагментом зображення.

Скрипт запускався з такими параметрами:

- similarity 0.3 – IoU має бути менше цього значення, для кадрів в одному бакеті, щоб кадр потрапив в першу добірку
- max-fps 6 – мінімальний корк – 6 кадрів
- min-bbox-size 0.0008 – мінімальна площа обмежувальної рамки. Тобто це буде квадрат із стороною sqrt(1280 * 720 * 0.0008) = 27 пікселів
- phash-hamming 18,18,12,12 – додаткова дедублікація кадрів по значенню відстані Хемінгу між хешами кадрів для кожного бакета (tiny, small, medium, large)

З 117 108 доступних кадрів алгоритм відібрав 7 115 (6.02%), з яких сформовано фінальний датасет: 6055 зображень у train, 752 у val та 308 у test. З огляду на відносно невеликий обсяг зібраних даних, у подальшому було прийнято рішення зосередитись виключно на задачі локалізації цілей без їх класифікації — тобто використовувати єдиний клас "ціль" замість чотирьох. Такий підхід дозволяє сконцентрувати навчальний потенціал моделі на якісному виявленні об'єктів у кадрі, не розпорошуючи його на розрізнення між класами.

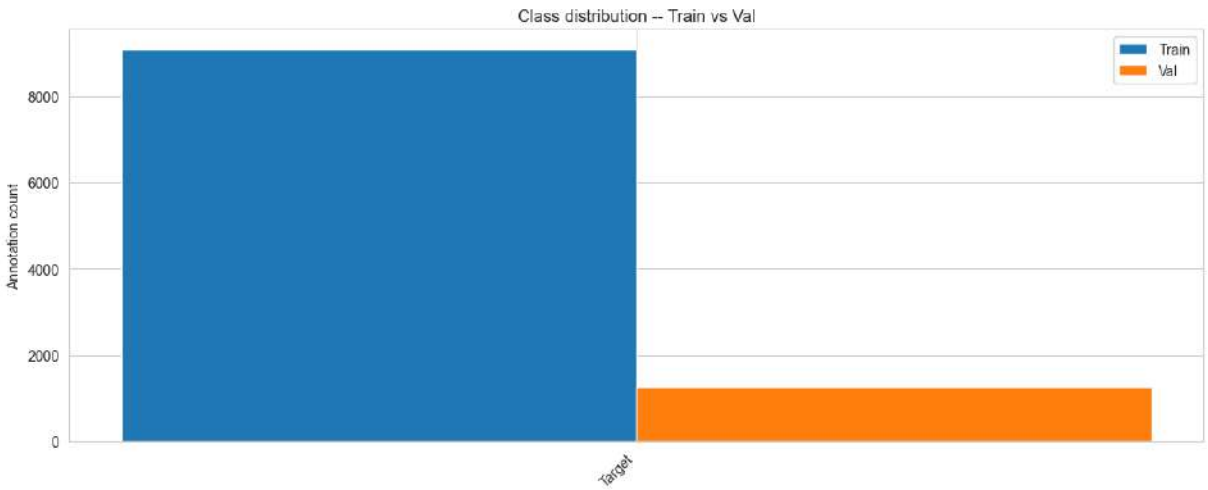

Рисунок. 2.6 – Співвідношення анотацій у train/val для одно-класового картинкового датасету (Target).

Висновки до розділу 2

У другому розділі описано повний процес формування власного датасету для навчання системи виявлення наземних об'єктів з борту БПЛА. Весь цикл — від збору відеоматеріалів до формування фінального картинкового датасету — виконувався вручну без використання готових публічних датасетів, що зумовило значні часові витрати але забезпечило повний контроль над якістю та складом даних.

Зібрано 855 відеофрагментів загальною тривалістю 88.1 хвилини, що охоплюють різні погодні умови, типи місцевості та класи наземної техніки. Розмітка виконувалась вручну у середовищі CVAT з використанням функції інтерполяції обмежувальних рамок між кадрами, що дозволило сформувати 1 337 унікальних треків із загальною кількістю 158 512 анотованих кадрів. Характерною особливістю датасету є переважання дрібних об'єктів — 81.6% анотацій належать до категорії Tiny — що відображає реальні умови зйомки з БПЛА на значній висоті.

Для усунення небажаних візуальних артефактів розроблено власний конвеєр очищення відео на основі моделі ProPainter з підтримкою пакетної обробки. Сумарна обробка всього масиву відеоматеріалів зайняла близько 20 годин безперервної роботи комп'ютера.

Для формування картинкового датасету розроблено власний алгоритм вибірки кадрів, що поєднує event-driven відбір, scale-стратифіковану дедуплікацію, глобальну дедуплікацію та перцептивне хешування. З 117 108 доступних кадрів відібрано 7 115 (6.02%), розподілених між train, val та test наборами. З огляду на обмежений обсяг даних прийнято рішення використовувати єдиний клас "ціль" для оптимізації детектора на задачі локалізації без класифікації.

РОЗДІЛ 3. АРХІТЕКТУРА ТА ТЕХНОЛОГІЇ РОЗРОБКИ

3.1 Загальна архітектура системи

Розроблена система побудована за парадигмою tracking-by-detection і складається з двох основних модулів: детектора об'єктів та трекера. На кожному кадрі відеопотоку детектор виявляє об'єкти і повертає набір обмежувальних рамок з відповідними оцінками впевненості. Трекер отримує ці детекції як вхідні дані, зіставляє їх з існуючими треками та оновлює їхні стани, забезпечуючи стійку ідентифікацію кожного об'єкта між кадрами.

Як детектор використовується модель YOLOv11n [1] — найлегша версія сімейства YOLO11, орієнтована на максимальну швидкодію при мінімальному споживанні обчислювальних ресурсів. Суфікс "n" (nano) відповідає найменшій конфігурації архітектури, що робить її придатною для розгортання на обладнанні з обмеженою обчислювальною потужністю. Модель навчена на власному датасеті з єдиним класом "ціль" для задачі локалізації наземної техніки.

За основу трекера взято імплементацію ByteTrack з відкритої бібліотеки BoxMOT [10], яка надає уніфікований інтерфейс для різних алгоритмів відстеження. Поверх цієї імплементації розроблено власне розширення — KPByteTrack (Keypoint ByteTrack) — що додає модуль підтримки треків на основі ключових точок. У моменти коли детектор демонструє високу впевненість у виявленні об'єкта, система витягує ключові точки з області обмежувальної рамки за допомогою одного з підтримуваних дескрипторів — AKAZE або ORB. На наступному кадрі виконується пошук відповідностей між збереженими та новими ключовими точками, після чого за допомогою RANSAC та естімації аффінної трансформації обчислюється нове положення обмежувальної рамки. Синтезована таким чином рамка передається до ByteTrack як додаткова детекція, що дозволяє підтримувати треки навіть у моменти зниження

впевненості детектора або повній відсутності детекцій. Конфігурація системи визначається YAML-файлом, параметри якого наведено в таблиці 3.1.

Таблиця 3.1

Конфігурація трекра та межі значень його параметрів

Параметр	**Тип**	**Можливі значення**	**Опис**
track_high_thresh	float	0.0 – 1.0	Поріг впевненості для першого етапу зіставлення — детекції вище цього порогу беруть участь в основній асоціації
track_low_thresh	float	0.0 – 1.0	Поріг для другого етапу — низькодостовірні детекції використовуються для відновлення втрачених треків
new_track_thresh	float	0.0 – 1.0	Мінімальна впевненість для ініціалізації нового треку
track_buffer	int	≥ 1 (кадри)	Кількість кадрів протягом яких втрачений трек залишається активним

match_thresh	float	0.0 – 1.0	Поріг схожості при зіставленні детекцій з треками
fuse_score	bool	True/False	Об'єднання оцінки детектора з IoU при зіставленні
kp_min_confirmed_age	int	≥ 1 (кадри)	Мінімальна кількість кадрів активності треку перед збереженням ключових точок
kp_min_update_score	float	0.0 – 1.0	Мінімальна впевненість детектора для оновлення банку ключових точок
kp_rolling_window_k	int	≥ 1	Кількість збережених наборів ключових точок на трек
kp_max_bank_age	int	≥ 1 (кадри)	Максимальна кількість кадрів без оновлення до видалення запису з банку
kp_min_inliers	int	≥ 1	Мінімальна кількість відповідних точок для синтезу рамки

kp_lowe_ratio	float	0.0 – 1.0	Поріг тесту Лоу для фільтрації хибних відповідностей
kp_n_features	int	≥ 1	Максимальна кількість ключових точок що витягуються з рамки
kp_bbox_reconstruct_method	string	median_fixed / ransac_affine / kalman_pred	Метод реконструкції рамки з ключових точок

2.2 Вибір технологій

Для реалізації системи виявлення та відстеження об'єктів у реальному часі було обрано сучасний набір інструментів, що поєднує перевірені бібліотеки для досліджень у галузі машинного навчання та комп'ютерного зору. Розробка велась мовою програмування Python 3.12.0, що є стандартним вибором для цієї галузі завдяки широкій екосистемі спеціалізованих пакетів та активній науковій спільноті.

Навчання нейронних мереж виконувалось на базі фреймворку PyTorch. Його застосування забезпечує гнучку побудову та навчання моделей глибокого навчання з підтримкою апаратного прискорення на GPU. Навчання проводилось на відеокарті NVIDIA RTX 5090, що забезпечило достатню обчислювальну потужність для ефективного тренування моделі.

Для роботи з детектором використовувалась бібліотека Ultralytics [11], яка надає повноцінний інтерфейс для навчання та використання моделей сімейства YOLO. Ultralytics визначає функцію втрат, реалізує широкий

спектр технік аугментації зображень, підтримує різні режими та способи навчання, а також надає зручні інструменти для валідації та експорту навченої моделі у різні формати.

Окремою перевагою використання бібліотеки Ultralytics є вбудована підтримка експорту моделей у формат TensorRT, що дозволяє виконати квантизацію нейронної мережі для розгортання на вбудованих пристроях, зокрема на платформах серії NVIDIA Jetson. Такий експорт доступний за замовчуванням і не потребує додаткових модифікацій моделі, що відкриває прямий шлях до портування розробленої системи на бортове обладнання з обмеженими обчислювальними ресурсами.

Для інтеграції трекера використовувалась бібліотека BoxMOT [10] — уніфікована платформа для підключення різних алгоритмів відстеження до довільного детектора. Саме на основі імплементації ByteTrack з цієї бібліотеки було розроблено власне розширення KPByteTrack, описане в підрозділі 3.1.

Задачі обробки зображень та відео вирішувались за допомогою бібліотеки OpenCV-python, яка надає широкий набір інструментів комп'ютерного зору — зокрема реалізації дескрипторів ключових точок AKAZE та ORB, що використовуються в модулі KPByteTrack.

Для автоматизованого підбору гіперпараметрів навчання детектора використовувалась бібліотека Optuna — фреймворк для оптимізації гіперпараметрів на основі байєсівського пошуку. В рамках пошуку виконувалось 30 випробовувань по 40 епох кожен з використанням TPE-семплера (Tree-structured Parzen Estimator). Цільовою метрикою оптимізації слугував показник mAP50-95 на валідаційній вибірці. В межах кожного випробовування підбирались параметри динаміки навчання (learning rate, batch size, кількість епох прогріву), ваги функції втрат (box, cls) та параметри аугментації зображень (HSV-перетворення, геометричні трансформації, mosaic, mixup, copy-paste). Результати всіх випробовувань зберігались у SQLite базі даних, що забезпечувало можливість відновлення

пошуку після переривання. Найкращі знайдені параметри використовувались для фінального навчання моделі.

Основою для роботи з числовими даними слугує бібліотека NumPy, а для роботи з табличними даними при аналізі статистики датасету та результатів оцінки — бібліотека pandas. Для візуалізації результатів, побудови графіків та відображення статистики датасету і метрик використовувались бібліотека matplotlib у поєднанні з Jupyter Notebook, що слугував інтерактивним середовищем для дослідницьких задач що потребували безпосереднього перегляду проміжних результатів.

Обраний технологічний стек є мінімалістичним та орієнтованим на дослідницькі задачі, що дає змогу системі ефективно виконувати виявлення та відстеження наземних об'єктів у реальному часі з можливістю подальшого розширення та оптимізації.

Висновки до розділу 3

У третьому розділі описано архітектуру розробленої системи виявлення та відстеження наземних об'єктів, а також обґрунтовано вибір технологій для її реалізації.

Система побудована за парадигмою tracking-by-detection і складається з двох основних модулів: детектора YOLOv11n та трекера KPByteTrack. Детектор виконує локалізацію об'єктів на кожному кадрі відеопотоку, після чого трекер зіставляє отримані детекції з існуючими треками та підтримує стійку ідентифікацію об'єктів між кадрами. Розроблене власне розширення KPByteTrack доповнює стандартний ByteTrack модулем підтримки треків на основі ключових точок з використанням дескрипторів AKAZE або ORB та геометричної верифікації через RANSAC, що дозволяє підвищити стійкість відстеження в умовах зниження впевненості детектора. Система є гнучко конфігурованою через YAML-файл, параметри якого охоплюють як базові

налаштування ByteTrack, так і специфічні параметри модуля ключових точок.

Розробка велась мовою Python 3.12.0 з використанням перевірених інструментів екосистеми машинного навчання та комп'ютерного зору. Навчання детектора виконувалось засобами бібліотеки Ultralytics з апаратним прискоренням на відеокарті NVIDIA RTX 5090. Підбір гіперпараметрів навчання здійснювався автоматизовано за допомогою бібліотеки Optuna з використанням TPE-семплера. Вбудована підтримка експорту в формат TensorRT відкриває можливість квантизації моделі для подальшого розгортання на вбудованих пристроях серії NVIDIA Jetson.

РОЗДІЛ 4. НАВЧАННЯ ДЕТЕКТОРА YOLO

4.1 Підбір гіперпараметрів та навчання детектора

Навчання детектора проводилось у два етапи: спочатку автоматизований підбір гіперпараметрів за допомогою Optuna, після чого фінальне навчання з найкращими знайденими параметрами. Обидва підходи до класифікації — з чотирма класами та з єдиним класом "ціль" — проходили через повний цикл оптимізації незалежно один від одного.

Перед навчанням було виконано маппінг оригінальних 11 класів CVAT у скорочений набір. На першому кроці класи об'єднувались у чотири групи: Armor (Tank, IFV, APC, Armour), Truck (Van, MLRS), Car (Car, Pickup) та LightVehicle (Motorcycle, ATV). Таке групування зумовлене значним дисбалансом між класами у вихідному датасеті — окремі класи, зокрема MLRS з лише 10 унікальними треками, мають недостатньо прикладів для надійного навчання окремого класифікатора. На другому кроці всі чотири групи було об'єднано в єдиний клас "ціль". Це рішення обґрунтовується тим, що для задачі автономного наведення та відстеження пріоритетом є надійна локалізація об'єкта у кадрі, а не його точна класифікація. Зосередження

моделі на єдиному класі дозволяє повністю сконцентрувати навчальний потенціал на якості детекції.

Підбір гіперпараметрів виконувався за допомогою Optuna з використанням TPE-семплера. В рамках кожного дослідження виконувалось 30 триалів по 40 епох з patience=7. Цільовою метрикою оптимізації слугував показник mAP50-95 на валідаційній вибірці. В межах кожного триалу підбирались параметри динаміки навчання (lr0, lrf, batch, warmup_epochs), ваги функції втрат (box, cls) та параметри аугментації (HSV-перетворення, геометричні трансформації, mosaic, mixup, copy-paste). Результати всіх триалів зберігались у SQLite базі даних з можливістю відновлення після переривання.

Аугментація зображень під час тренування відігравала особливо важливу роль у даній роботі через відносно невеликий обсяг датасету та обмежену варіативність прикладів — з одного відео зазвичай витягується до 5-10 кадрів, що означає певну кореляцію між прикладами в межах одного відео. Широкий діапазон аугментацій дозволяв штучно збільшити різноманітність навчальних прикладів і знизити ризик перенавчання.

Для покращення виявлення малих об'єктів, які становлять 81.6% анотацій датасету, розмір вхідного зображення було встановлено на рівні 800px замість стандартного 640px. Більший розмір вхідного зображення дозволяє зберегти більше деталей дрібних об'єктів при проходженні через згорткові шари мережі, що безпосередньо впливає на здатність детектора виявляти цілі на значній відстані.

Фінальні гіперпараметри навчання для обох варіантів наведено в таблиці 4.1.

Таблиця 4.1

Параметр	**4 класи**	**1 клас**
epochs	48	45
patience	20	20
batch	32	32

imgsz	800	800
lr0	0.000240	0.000174
lrf	0.03949	0.07384
warmup_epochs	10	14
box	22.705	17.859
cls	1.427	0.232
hsv_h	0.087	0.053
hsv_s	0.536	0.471
hsv_v	0.614	0.483
degrees	9.031	0.035
translate	0.361	0.232
scale	0.799	0.532
mosaic	0.743	0.898
mixup	0.196	0.093
copy_paste	0.029	0.081

4.2 Валідація та аналіз метрик детектора

Після завершення навчання обох варіантів детектора було проведено їх порівняльну валідацію на тестовому наборі даних. Аналіз охоплює криві навчання, загальні метрики детекції, PR-криві, окрему оцінку локалізаційних можливостей та якості виявлення малих об'єктів.

Криві навчання

Криві навчання для обох варіантів наведено на рисунках 4.1 та 4.2 відповідно. Для варіанту з єдиним класом усі три компоненти функції втрат — box loss, classification loss та DFL loss — демонструють стабільне і плавне зниження як на тренувальній, так і на валідаційній вибірці. Метрики mAP50 та mAP50-95 зростають монотонно протягом усього навчання без суттєвих коливань, що свідчить про стабільну оптимізацію.

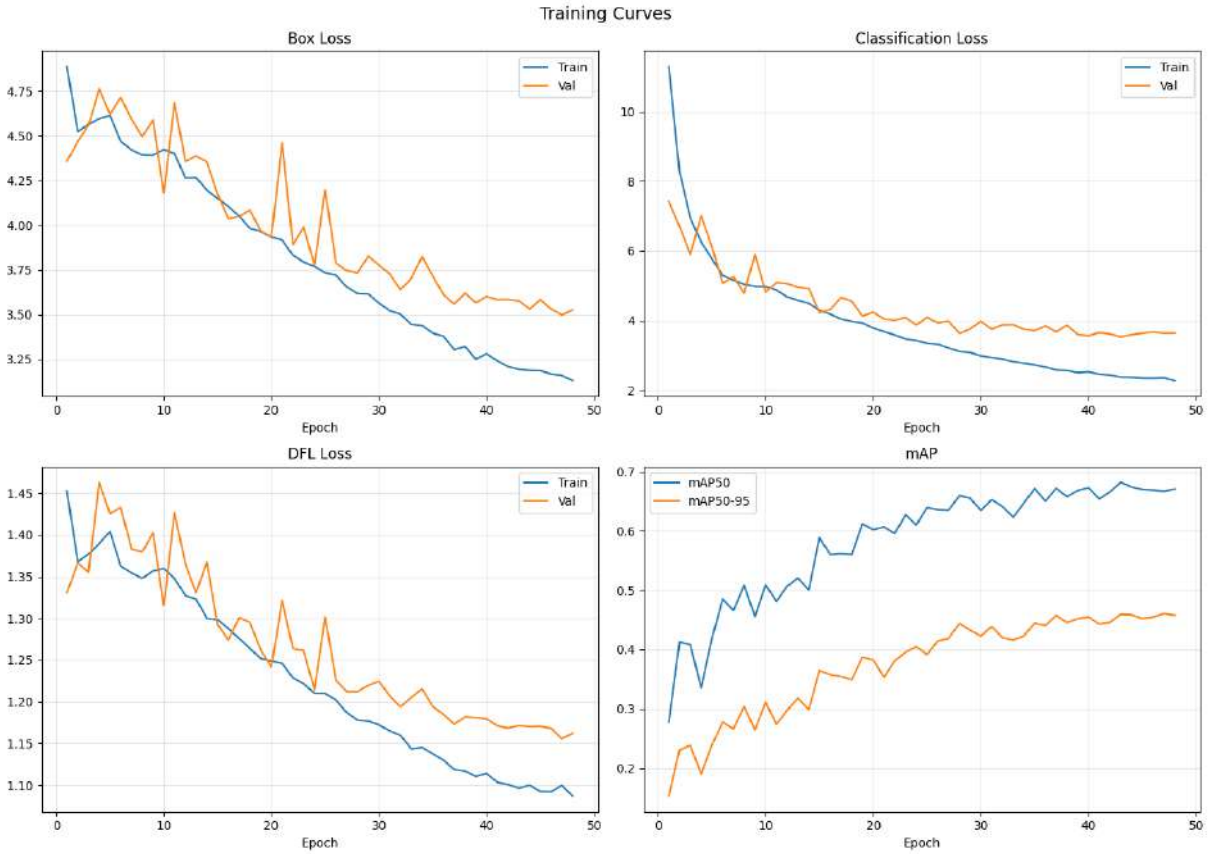

Рисунок 4.1 – Криві навчання для 4-класового детектора

Для варіанту з чотирма класами картина є помітно відмінною: classification loss починає з значно вищого початкового значення (~11 проти ~4 для єдиного класу), а криві mAP демонструють виражені коливання протягом навчання. Це пояснюється тим, що чотири класи з різною кількістю прикладів та високою візуальною варіативністю створюють нестабільний градієнтний сигнал — покращення для одного класу може тимчасово погіршувати результати для іншого. Ознак класичного перенавчання в жодному з варіантів не виявлено: val loss хоч і вище за train loss, але обидва продовжують знижуватись у тому самому напрямку до кінця навчання без розходження кривих.

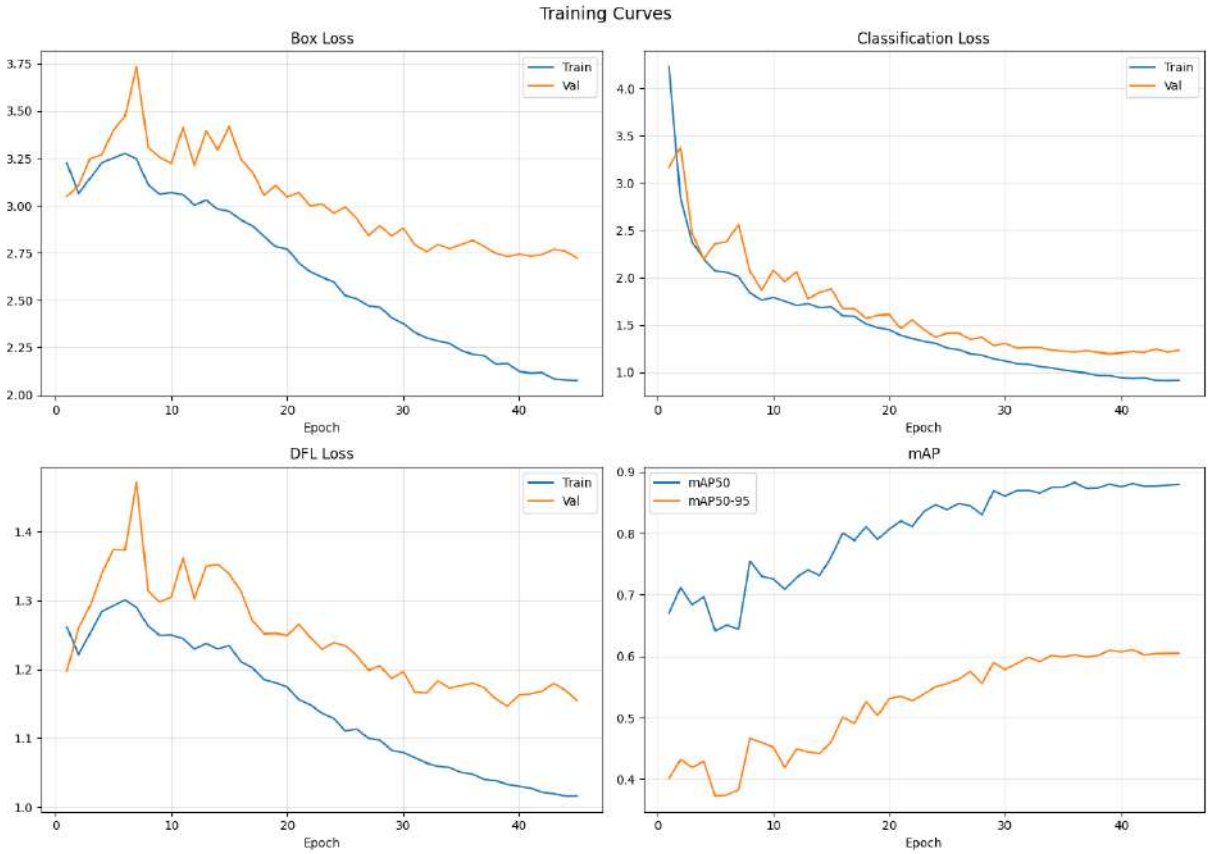

Рисунок 4.2 – Криві навчання для 1-класового детектора

Загальні метрики валідації

Варіант з єдиним класом демонструє суттєво вищі показники за всіма метриками. Перевага у mAP50 складає 0.214, а у mAP50-95 — 0.151. Проте ці відмінності значною мірою зумовлені саме помилками класифікації у чотирикласовому варіанті, а не якістю самої локалізації об'єктів. Подальші експерименти з class-agnostic оцінкою, наведені нижче, дозволять відокремити внесок класифікаційної помилки від якості локалізації — яка і є першочерговою метрикою для подальшого застосування у задачі відстеження.

Аналіз результатів по класах для чотирикласового варіанту виявляє значний розкид між класами. Клас Armor демонструє найкращі показники (mAP50: 0.902, Precision: 0.856, Recall: 0.860) завдяки відносно однорідному зовнішньому вигляду броньованої техніки. Натомість клас Truck показує найгірші результати (mAP50: 0.492, Precision: 0.434, Recall: 0.516).

```
======================================================
Overall Validation Metrics
======================================================
  Precision       0.6547
  Recall          0.6525
  F1              0.6536
  mAP@50          0.6671
  mAP@50-95       0.4603
======================================================

Class                P       R    mAP50   mAP50-95
------------------------------------------------
Armor              0.856   0.860    0.902      0.637
Truck              0.434   0.516    0.492      0.350
Car                0.550   0.604    0.570      0.404
LightVehicle       0.779   0.630    0.704      0.449
------------------------------------------------
```

Рисунок 4.3 – Загальні метрики валідації 4-класового детектора

```
======================================================
Overall Validation Metrics
======================================================
  Precision       0.8557
  Recall          0.8021
  F1              0.8280
  mAP@50          0.8813
  mAP@50-95       0.6109
======================================================

Class                P       R    mAP50   mAP50-95
------------------------------------------------
Target             0.856   0.802    0.881      0.611
------------------------------------------------
```

Рисунок 4.4 – Загальні метрики валідації 1-класового детектора

Це пояснюється надзвичайно високою візуальною варіативністю цього класу: він об'єднує цивільні вантажівки, мінівени та військові автомобілі, які нерідко мають додаткові надбудови для антидронового захисту. Висока різноманітність зовнішнього вигляду в поєднанні з недостатньою кількістю навчальних прикладів не дозволяє моделі

сформувати стійкі ознаки для цього класу. Класи Car та LightVehicle займають проміжне положення з mAP50 0.570 та 0.704 відповідно.

PR-криві

PR-криві для обох варіантів наведено на рисунках 4.5 та 4.6. Для варіанту з єдиним класом крива має характерну форму з високою точністю при низькому значенні recall та плавним спаданням — площа під кривою відповідає mAP50 = 0.881, що є сильним результатом для задачі детекції малих об'єктів у складних умовах зйомки.

Для чотирикласового варіанту PR-криві наочно демонструють розрив між класами: Armor утримує високу точність на широкому діапазоні recall, тоді як крива Truck опускається значно швидше, підтверджуючи проблему з цим класом. Середня крива (all mAP50=0.667) відображає вплив слабких класів на загальний результат.

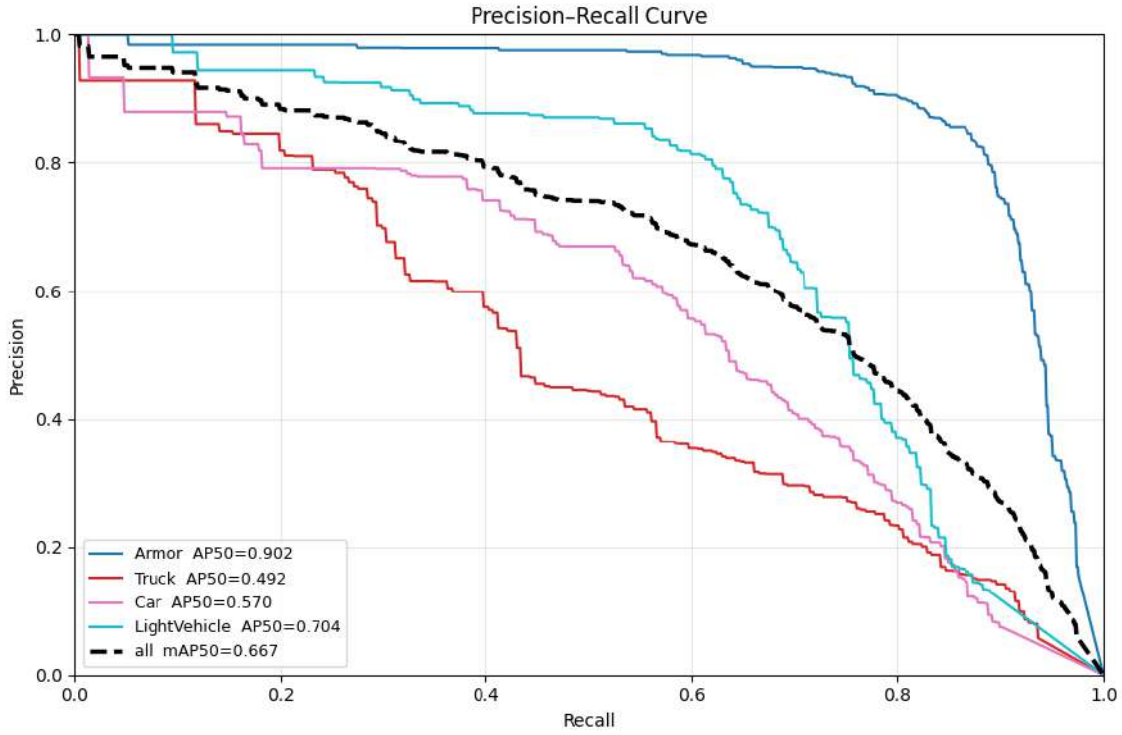

Рисунок 4.5 – Precision-Recall криві для 4-класового детектора

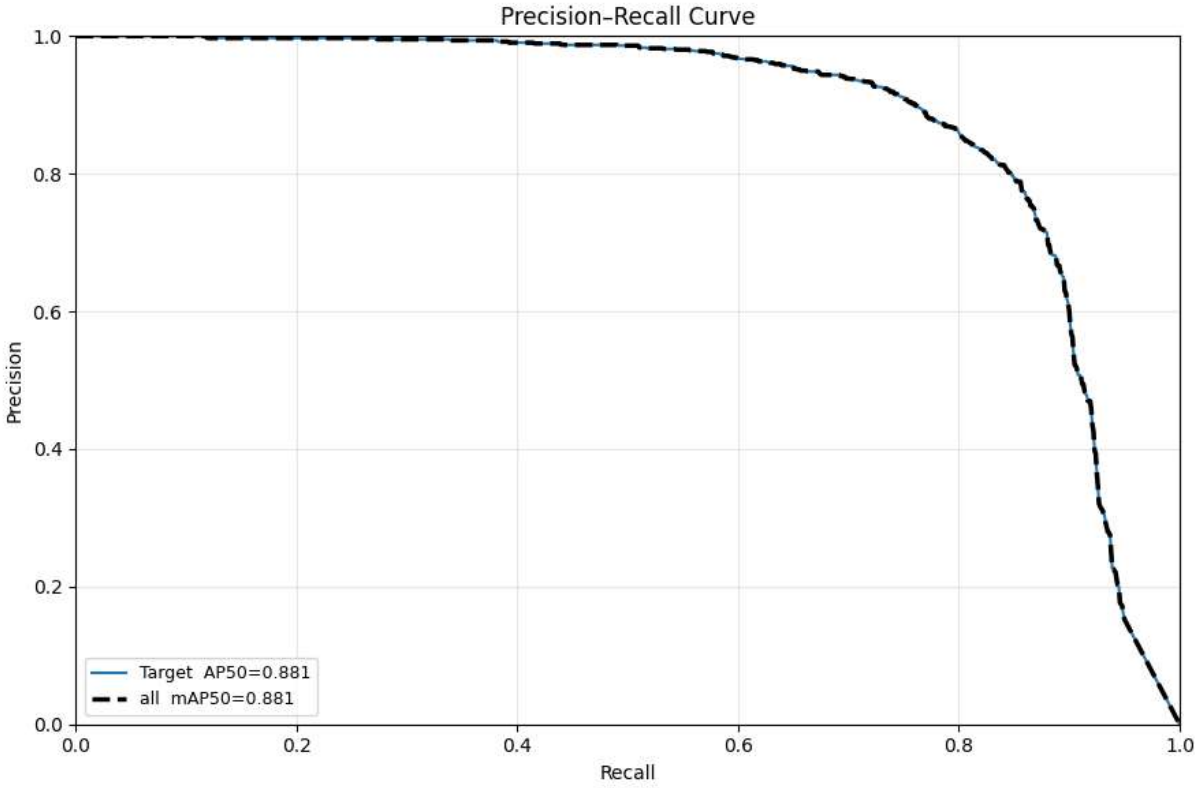

Рисунок 4.6 – Precision-Recall крива для 1-класового детектора

Порівняння локалізаційних можливостей

Зважаючи на те, що чотирикласовий варіант значно поступається одно класовому за стандартними метриками, виникає природне питання: чи зумовлено це лише складністю задачі класифікації, чи 4-класовий детектор також гірше локалізує об'єкти як такі. Для відповіді на це питання було проведено окрему class-agnostic оцінку — обидва детектори оцінювались виключно за здатністю виявляти об'єкти будь-якого класу, без врахування правильності їх класифікації. Результати наведено в таблиці 4.2.

Результати порівняння виявились несподіваними: обидва детектори локалізують об'єкти практично рівноцінно. AP50 відрізняється лише на 0.004 (0.863 vs 0.859), кількість пропущених об'єктів (FN) майже ідентична (226 vs 224). Більше того, чотирикласовий детектор демонструє навіть дещо нижчу кількість хибнопозитивних спрацювань (169 vs 194) та трохи вищу Precision (0.8593 vs 0.8420). Ці відмінності знаходяться в межах статистичного шуму для датасету такого розміру і не дозволяють стверджувати про перевагу одного з варіантів у задачі локалізації.

Таблиця 4.2

Class-agnostic порівняння локалізації (IoU ≥ 0.5, conf ≥ 0.25)

Метрика	Однокласовий детектор	Чотирикласовий детектор
Total GT boxes	1258	1258
True Positives (TP)	1034	1032
False Negatives (FN)	224	226
False Positives (FP)	194	169
Precision	0.8420	0.8593
Recall	0.8219	0.8203
F1	0.8319	0.8394
AP50	0.863	0.859

Таке спостереження має кілька можливих пояснень. По-перше, навчаючись розрізняти чотири класи, модель змушена формувати більш специфічні та дискримінативні ознаки для кожного з них, що побічно покращує розуміння меж об'єктів. По-друге, чотирикласовий детектор є "обережнішим" при спрацюванні — щоб зробити детекцію, він повинен не лише знайти об'єкт, а й віднести його до конкретного класу, що додатково фільтрує хибні спрацювання на фоні.

Головний висновок з цього порівняння полягає в тому, що проблема чотирикласового варіанту — виключно в задачі класифікації, а не в здатності виявляти об'єкти. Це підтверджує що обидві моделі однаково придатні як основа для трекера, який працює лише з обмежувальними рамками.

Оцінка детекції малих об'єктів

Окрему увагу приділено аналізу якості виявлення малих об'єктів — тих що займають площу менше 2500 px² при роздільній здатності 1280×720 пікселів. Ця оцінка є особливо важливою в контексті застосування на БПЛА, адже чим раніше система виявить ціль — тим більше часу є для прийняття

рішення. Результати оцінки наведено на рисунках 4.9 та 4.10, а зведені показники — в таблиці 4.4.

Оцінка детекції малих об'єктів

Окрему увагу приділено аналізу якості виявлення малих об'єктів — тих що займають площу менше 2500 px² при роздільній здатності 1280×720 пікселів. Ця оцінка є особливо важливою в контексті застосування на БПЛА, адже чим раніше система виявить ціль — тим більше часу є для прийняття рішення. Результати оцінки наведено на рисунках 4.9 та 4.10, а зведені показники — в таблиці 4.3.

Таблиця 4.3

Оцінка детекції малих об'єктів (area < 2500 px², IoU ≥ 0.5)

Метрика	Одноклассовий детектор	Чотирикласовий детектор
Зображень з малими об'єктами	118	118
Всього малих об'єктів (GT)	190	190
True Positives (TP)	123	112
False Negatives (FN)	67	78
False Positives (FP)	210	217
Precision	0.3694	0.3404
Recall	0.6474	0.5895
F1	0.4704	0.4316

Варіант з єдиним класом демонструє кращі показники виявлення малих об'єктів за всіма метриками. Recall є вищим на 0.058 (0.6474 проти 0.5895), що означає що модель виявляє більше малих цілей — 123 проти 112 зі 190 можливих. Precision для обох варіантів є відносно низькою через велику кількість хибнопозитивних спрацювань, що є характерною проблемою для детекції дрібних об'єктів: при малому розмірі об'єкта навіть незначні текстурні артефакти фону можуть спровокувати хибну детекцію. В контексті задачі відстеження та ураження цілей вищий Recall є пріоритетнішою характеристикою — пропустити реальну ціль є

критичнішою помилкою ніж отримати зайву детекцію, яку трекер може відфільтрувати на наступних кадрах.

Висновки до розділу 4

У четвертому розділі описано процес навчання та валідації детектора об'єктів на базі моделі YOLOv11n. Розглянуто два варіанти підходу до класифікації — з чотирма агрегованими класами (Armor, Truck, Car, LightVehicle) та з єдиним класом "ціль" — обидва з повноцінним циклом автоматизованого підбору гіперпараметрів через Optuna з використанням TPE-семплера.

Для покращення детекції малих об'єктів, які становлять переважну частину анотацій датасету, розмір вхідного зображення було збільшено до 800px замість стандартного 640px. Аугментація відіграла особливо важливу роль через обмежений обсяг та кореляцію прикладів у межах одного відео — параметри HSV-перетворень, геометричних трансформацій, mosaic, mixup та copy-paste підбирались Optuna незалежно для кожного варіанту.

Порівняльна валідація показала, що варіант з єдиним класом перевершує чотирикласовий за стандартними метриками: mAP50 0.881 проти 0.667, mAP50-95 0.611 проти 0.460. Аналіз результатів по класах виявив значний розкид у чотирикласовому варіанті — найкращим є клас Armor (mAP50: 0.902), найгіршим — клас Truck (mAP50: 0.492), що пояснюється високою візуальною варіативністю останнього в поєднанні з обмеженою кількістю прикладів.

Окрема class-agnostic оцінка локалізаційних можливостей виявила несподіване: обидва варіанти локалізують об'єкти практично рівноцінно (AP50: 0.863 та 0.859 відповідно), причому чотирикласовий навіть має дещо нижчу кількість хибнопозитивних спрацювань. Це довело що проблема чотирикласового варіанту полягає виключно в задачі класифікації, а не у здатності виявляти об'єкти. Натомість у виявленні малих об'єктів одно

класовий варіант показав помітну перевагу за всіма метриками, що є критично важливим для бортового застосування.

На основі проведеного аналізу для інтеграції з модулем відстеження обрано варіант з єдиним класом "ціль". Хоча локалізаційні можливості обох варіантів виявились практично рівноцінними, одно класовий варіант демонструє кращі результати у виявленні малих об'єктів. Окрім метричних показників, цей вибір обґрунтовується і архітектурними міркуваннями: алгоритм ByteTrack не використовує інформацію про клас об'єкта при зіставленні детекцій з треками — його цікавить виключно положення та розмір обмежувальної рамки. Таким чином, навіть якби чотирикласовий детектор використовувався як основа, клас об'єкта міг би змінюватись між кадрами попри збереження того самого ідентифікатора треку, що вносило б непослідовність у вихідні дані системи. В розрізі прикладної задачі — автономного наведення та відстеження наземних цілей — локалізація об'єкта є першочерговим пріоритетом, тоді як його класифікація є другорядною задачею що може бути вирішена окремо при наявності достатнього обсягу даних.

РОЗДІЛ 5. ОЦІНКА ЕФЕКТИВНОСТІ ТА МОЖЛИВОСТІ РОЗВИТКУ

5.1 Принцип збору мерик для оцінки ефективності відстеження

Оцінка якості розробленої системи відстеження виконувалась за стандартизованими метриками багатооб'єктного трекінгу з використанням бібліотеки TrackEval, яка реалізує канонічні протоколи підрахунку метрик відповідно до бенчмарку MOT Challenge. Власні скрипти автоматизували процес генерації передбачень, підготовки даних у форматі що очікує TrackEval, та агрегації результатів по всьому набору тестових відео.

Загальна схема оцінки

Процес оцінки складається з трьох послідовних етапів. На першому етапі для кожного відеофайлу запускається повний цикл інференсу: детектор обробляє кадри послідовно, після чого детекції передаються трекеру для зіставлення з існуючими треками. Результати інференсу — обмежувальні рамки з присвоєними ідентифікаторами треків — зберігаються у форматі CSV для кожного відео окремо.

На другому етапі для кожного відео обчислюються метрики шляхом порівняння передбачень трекера з ground truth анотаціями. Метрики по кожному відео відображаються в таблиці, яка рендериться в .ipynb ноутбуці. Приклад зображено на риснку 5.1 На третьому етапі результати по окремих відео агрегуються у загальні показники системи.

video	MOTA	MOTP	IDF1	HOTA	DetA	AssA	IDSW	FP	FN	MT	ML
2	1.000000	0.894176	1.000000	0.897410	0.897410	0.897410	0	0	0	1	0
4	1.000000	0.844307	1.000000	0.846131	0.846131	0.846131	0	0	0	1	0
54	0.760870	0.852363	0.791209	0.650392	0.705944	0.599357	4	8	10	1	0
74	-0.564706	0.812902	0.247826	0.235721	0.233284	0.239679	2	192	72	0	0
79	0.528302	0.792909	0.691358	0.536755	0.418997	0.687775	0	0	50	1	1
81	0.769231	0.823863	0.782178	0.655427	0.649896	0.664289	1	20	0	1	0
93	0.974066	0.868834	0.987179	0.847401	0.840778	0.854231	0	32	2	8	0
100	0.538336	0.812063	0.240137	0.283311	0.553646	0.147227	27	98	158	3	0
101	1.000000	0.825496	1.000000	0.843389	0.843389	0.843389	0	0	0	1	0
117	-0.250000	0.913697	0.409938	0.323203	0.282112	0.381605	0	52	43	0	0
118	0.767442	0.766420	0.895833	0.673851	0.608861	0.745833	0	10	0	1	0
119	0.937500	0.814928	0.967742	0.743920	0.743920	0.743920	0	0	4	1	0
125	0.747525	0.796664	0.848485	0.603018	0.599794	0.606666	2	4	45	1	0
142	0.802632	0.805593	0.525000	0.494003	0.664424	0.367851	1	11	3	1	0
157	1.000000	0.851215	1.000000	0.830125	0.830125	0.830125	0	0	0	1	0
170	0.976190	0.831173	0.987952	0.813541	0.813541	0.813541	0	0	3	1	0

Рисунок 5.1 – Відображення метрик по кожному окремому відео.

Підготовка даних для оцінки

Ключовим аспектом коректної оцінки є правильне формування набору кадрів для аналізу. Множина кадрів формується як об'єднання кадрів з ground truth та кадрів з передбаченнями трекера. Це принципово важливо: якщо трекер видає детекції на кадрах, де об'єкт вже зник зі сцени, такі детекції коректно враховуються як хибнопозитивні (FP). Аналогічно, якщо ground truth містить об'єкт на кадрі, але трекер його не виявив — це фіксується як хибнонегативна помилка (FN).

Для кожного кадру будується матриця подібності між усіма парами ground truth боксів та передбачених боксів, де подібність визначається як значення IoU (Intersection over Union):

$$IoU(A, B) = \frac{| A \cap B |}{| A \cup B |}$$

де A — обмежувальна рамка з ground truth, B — обмежувальна рамка з передбачень трекера. Алгоритми метрик самостійно визначають оптимальне зіставлення між боксами на основі цієї матриці з використанням порогу IoU ≥ 0.5 для більшості метрик.

Базові поняття та лічильники

В основі підрахунку метрик лежать чотири фундаментальних лічильники, що формуються при зіставленні передбачень з ground truth на кожному кадрі:

– **TP** (True Positives) — передбачена рамка коректно відповідає ground truth рамці (IoU ≥ 0.5);

– **FP** (False Positives) — передбачена рамка не має відповідної ground truth рамки;

– **FN** (False Negatives) — ground truth рамка не має відповідної передбаченої рамки;

– **IDSW** (Identity Switches) — кількість випадків, коли той самий ground truth об'єкт отримує новий ідентифікатор треку.

MOTA — Multiple Object Tracking Accuracy

MOTA [13] є однією з найпоширеніших метрик для агрегованої оцінки якості трекера. Вона обчислюється за формулою:

$$MOTA = 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t}$$

де сумування виконується по всіх кадрах t, а GT — загальна кількість ground truth об'єктів. Метрика приймає значення в діапазоні $(-\infty, 1]$, де 1 означає ідеальний трекінг без жодної помилки. Від'ємні значення можливі коли сумарна кількість помилок перевищує кількість ground truth об'єктів. MOTA штрафує за всі три типи помилок одночасно з однаковою вагою, що є водночас її перевагою (комплексність оцінки) та недоліком (неможливість окремо діагностувати джерело проблеми).

MOTP — Multiple Object Tracking Precision

MOTP [13] вимірює просторову точність локалізації для коректно зіставлених пар:

$$MOTP = \frac{\sum_{t,i} IoU(GT_{t,i}, Pred_{t,i})}{\sum_t TP_t}$$

де сумування виконується по всіх успішно зіставлених парах і на всіх кадрах t. Метрика характеризує виключно якість обмежувальних рамок без врахування кількості помилок асоціації, тому слабко залежить від поведінки самого трекера.

IDF1 — Identity F1 Score

IDF1 [14] оцінює здатність системи зберігати стійку ідентифікацію об'єктів. Замість підрахунку помилок на рівні окремих кадрів метрика будує оптимальне глобальне зіставлення між ground truth треками та передбаченими треками на основі максимального перекриття за всю послідовність:

$$IDF1 = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN}$$

де **IDTP** — кількість кадрів, на яких ground truth трек та зіставлений з ним передбачений трек коректно збігаються; **IDFP** — кадри де передбачений трек є, але не зіставлений з жодним ground truth; **IDFN** — кадри де ground truth трек є, але не покритий зіставленим передбаченим треком.

Висока IDF1 свідчить про те, що система не лише виявляє об'єкти, а й послідовно підтримує їхні ідентифікатори впродовж усього часу присутності у відеопотоці. Саме ця метрика є **першочерговою для розробленої системи**, оскільки безпосередньо вимірює ключову функцію трекера — стійкий супровід цілі.

HOTA — Higher Order Tracking Accuracy

HOTA [14] є сучасною метрикою, що усуває ключове обмеження MOTA шляхом явного розділення внесків детекції та асоціації. Вона обчислюється як геометричне середнє двох незалежних складових:

$$HOTA_\alpha = \sqrt{DetA_\alpha \cdot AssA_\alpha}$$

де **DetA** (Detection Accuracy) оцінює якість виявлення об'єктів:

$$DetA_\alpha = \frac{TP_\alpha}{TP_\alpha + FP_\alpha + FN_\alpha}$$

а **AssA** (Association Accuracy) — якість зіставлення треків між кадрами:

$$AssA_\alpha = \frac{1}{|\ TP_\alpha\ |} \sum_{c \in TP_\alpha} \frac{TPA(c)}{TPA(c) + FPA(c) + FNA(c)}$$

де TPA, FPA та FNA — асоціативні лічильники для кожного коректно зіставленого матчу c. Індекс α означає поріг IoU. Підсумкова метрика HOTA усереднюється по 19 значеннях α від 0.05 до 0.95 з кроком 0.05:

$$HOTA = \frac{1}{19} \sum_{\alpha=0.05}^{0.95} HOT\,A_\alpha$$

Завдяки усередненню по широкому діапазону порогів HOTA є менш чутливою до вибору конкретного значення IoU порівняно з MOTA та IDF1.

Додаткові показники

Окрім агрегованих метрик для детального аналізу використовуються:

– **MT** (Mostly Tracked) — кількість ground truth треків, що були покриті передбаченнями більш ніж у 80% кадрів свого життєвого циклу; – **ML** (Mostly Lost) — кількість треків, покритих менш ніж у 20% кадрів; – кількісні лічильники **FP**, **FN** та **IDSW** — дозволяють інтерпретувати причини відхилень у агрегованих метриках: переважання FP свідчить про "галюцинації" трекера, переважання FN — про пропуск реальних об'єктів, високий IDSW — про нестабільність ідентифікаторів.

Пріоритетність метрик у контексті задачі

В контексті розробленої системи метрики мають різну пріоритетність. IDF1 є першочерговою метрикою, оскільки безпосередньо вимірює здатність системи зберігати стійку ідентифікацію об'єкта між кадрами — це ключова функція трекера для задачі автономного супроводу цілі. HOTA та її складові DetA і AssA надають збалансовану оцінку якості детекції та асоціації окремо, що дозволяє діагностувати на якому саме рівні виникають проблеми. MOTA є корисною як агрегований показник загальної якості, проте її чутливість до балансу між типами помилок робить її менш інтерпретованою при порівнянні різних варіантів системи. MOTP характеризує виключно якість локалізації і слабко залежить від поведінки трекера.

Агрегація метрик по всьому набору відео

Просте усереднення метрик за окремими відео призводить до спотворення результатів: коротке відео з однією легкою ціллю отримує таку саму вагу як довге відео з кількома складними цілями. Для уникнення цього використовується підхід combine_sequences, реалізований у TrackEval. Замість усереднення похідних метрик він підсумовує сирі лічильники (TP, FP, FN, IDSW, IDTP, IDFP, IDFN) по всіх відео, після чого перераховує підсумкові метрики на основі цих сум. Таким чином довші та складніші

відео природним чином отримують більшу вагу у фінальному результаті, що відповідає реальній складності датасету.

5.2 Порівняння класичного ByteTrack та розробленої модифікації

Для оцінки впливу розробленої модифікації на якість відстеження було проведено серію порівняльних експериментів. У кожному експерименті порівнювались два варіанти трекера: класичний ByteTrack та розроблений KPByteTrack з підтримкою ключових точок на основі дескрипторів ORB. Обидва варіанти використовували ідентичну конфігурацію базового алгоритму ByteTrack — відмінність полягала лише в порозі впевненості детектора для генерації ключових точок (kp_min_update_score). Для емуляції класичного ByteTrack цей поріг встановлювався на нереалістично високе значення (0.9999), за якого ключові точки фактично ніколи не генеруються і трекер поводиться як стандартна реалізація. Для KPByteTrack поріг знижувався до 0.3, що активує механізм підтримки треків ключовими точками.

Експерименти проводились у двох конфігураціях, що відрізнялись якістю детектора: з повноцінно навченим детектором та з навмисно недотренованим детектором. Це дозволило перевірити ключову гіпотезу роботи — що модифікація на основі ключових точок є особливо корисною за умов недосконалого детектора.

Експеримент 1 — повноцінно навчений детектор

У першому експерименті використовувався найкращий одно класовий детектор (mAP50: 0.881). Результати збору метрик наведено на рисунках 5.2 та 5.3.

video	MOTA	MOTP	IDF1	HOTA	DetA	AssA	IDSW	FP	FN	MT	ML
Overall	0.640336	0.830915	0.715726	0.611248	0.575838	0.654911	108	1797	4178	85	10

Рисунок 5.2 – Метрики трекінгу класичного ByteTrack

video	MOTA	MOTP	IDF1	HOTA	DetA	AssA	IDSW	FP	FN	MT	ML
Overall	0.610773	0.829671	0.717565	0.607670	0.559862	0.665173	82	2435	4066	87	11

Рисунок 5.3 – Метрики трекінгу KPByteTrack + дескриптори ORB

За умов якісного детектора різниця між трекерами є мінімальною. KPByteTrack демонструє незначне покращення першочергової метрики IDF1 (+0.0015), зниження кількості переключень ідентифікаторів IDSW (з 108 до 82) та хибнонегативних помилок FN (з 4178 до 4066), а також вищу якість асоціації AssA (+0.0102). Водночас спостерігається зростання хибнопозитивних спрацювань FP (з 1797 до 2435), що знижує MOTA та DetA. Це пояснюється тим, що синтезовані за ключовими точками рамки інколи продовжують існувати на кадрах, де реальний об'єкт уже зник, породжуючи додаткові FP.

Експеримент 2 — недотренований детектор

У другому експерименті використовувався навмисно недотренований детектор (mAP50: 0.762, mAP50-95: 0.488) з помітно нижчою якістю виявлення об'єктів. Результати збору метрик наведено на рисунках 5.4 та 5.5.

За умов слабкого детектора перевага KPByteTrack стає значно виразнішою. Покращення IDF1 складає +0.0175, що майже в 12 разів більше ніж для повноцінного детектора. Метрика HOTA зросла на +0.0116, причому покращились обидві її складові — і DetA (+0.0085), і AssA (+0.0159). Кількість пропущених об'єктів FN скоротилась суттєво — з 6878 до 6287, а IDSW знизилось з 73 до 62. Як і в першому експерименті, ціною цих покращень є зростання FP (з 1956 до 2621).

video	MOTA	MOTP	IDF1	HOTA	DetA	AssA	IDSW	FP	FN	MT	ML
Overall	0.473364	0.820258	0.618530	0.516636	0.447873	0.600777	73	1956	6878	52	33

Рисунок 5.4 – Метрики трекінгу класичного ByteTrack з недотренованим детектором

video	MOTA	MOTP	IDF1	HOTA	DetA	AssA	IDSW	FP	FN	MT	ML
Overall	0.470171	0.816417	0.636141	0.528182	0.455661	0.616054	53	2621	6287	62	33

Рисунок 5.5 – Метрики трекінгу KPByteTrack + дескриптори ORB з недотренованим детектором

Аналіз результатів

Зіставлення двох експериментів виявляє чітку та послідовну закономірність: позитивний вплив модифікації на якість відстеження зростає в міру погіршення якості детектора. Це безпосередньо підтверджує закладену в роботі гіпотезу. Механізм роботи модифікації пояснює цей результат: коли детектор через низьку впевненість пропускає об'єкт на певних кадрах, модуль ключових точок синтезує обмежувальну рамку на основі збережених ознак з попередніх кадрів і "дотягує" трек. За якісного детектора такі пропуски рідкісні, тому ефект мінімальний. За слабкого детектора пропуски стають частими — і саме тут механізм компенсації проявляє себе найкраще, що відображається у зниженні FN та покращенні стійкості ідентифікації.

Оскільки детектор є критично важливим компонентом у парадигмі tracking-by-detection, здатність часткової компенсації його недоліків на рівні трекера є цінною властивістю — особливо в умовах обмеженого датасету, коли досягнення ідеальної якості детектора є проблематичним.

Водночас слід чесно зазначити обмеження проведеного дослідження. Експерименти виконані на одному датасеті, з одним типом дескриптора (ORB) та двома рівнями якості детектора. Тому отримані результати коректно інтерпретувати як стійку тенденцію, що підтверджує гіпотезу, а не як остаточно доведену універсальну закономірність. Для більш переконливих висновків необхідне розширене дослідження з різними типами дескрипторів, кількома рівнями недотренованості детектора та повторними прогонами для оцінки статистичної варіативності результатів.

Швидкодія алгоритму

Окремим і критично важливим аспектом порівняння є швидкодія. Результати замірів продуктивності наведено в таблиці 5.1.

Таблиця 5.1

Порівняння швидкодії трекерів

Показнпик	**ByteTrack**	**KPByteTrack(ORB)**
Оброблено відео	95	95
Всього кадрів	13 149	13 149
Тривалість відео	7 хв 18 с	7 хв 18 с
Час обробки	2 хв 21 с	18 хв 02 с
Пропускна здатність	92.6 FPS	12.1 FPS
Коефіцієнт реального часу	3.09x	0.40x

Швидкодія є головним недоліком розробленої модифікації у її поточному вигляді. Класичний ByteTrack обробляє відеопотік зі швидкістю 92.6 FPS, що відповідає 3.09-кратному запасу відносно реального часу (при джерелі 30 FPS). KPByteTrack натомість досягає лише 12.1 FPS, що становить 0.40x реального часу — тобто система не встигає обробляти відеопотік у режимі реального часу. Падіння швидкодії майже у 8 разів зумовлене обчислювальною вартістю екстракції та зіставлення ключових точок з геометричною верифікацією через RANSAC, що виконується для кожного активного треку на кожному кадрі.

Цей результат означає, що в поточній реалізації KPByteTrack забезпечує вищу якість відстеження ціною втрати здатності працювати в реальному часі. Для практичного бортового застосування ця проблема є визначальною і потребує суттєвої оптимізації, напрями якої розглянуто в наступному підрозділі.

5.3 Перспективи розвитку та шляхи оптимізації

Розроблена система демонструє життєздатність обраного підходу, проте має значний простір для подальшого вдосконалення. Нижче окреслено ключові напрями розвитку, що дозволять підвищити практичну придатність системи до бортового застосування.

Оптимізація швидкодії

Найкритичнішою проблемою поточної реалізації є недостатня швидкодія KPByteTrack, виявлена в підрозділі 5.2. Першочерговим напрямом оптимізації є квантування вагів детектора з подальшим експортом у формат TensorRT, що є штатною можливістю бібліотеки Ultralytics. Це дозволить суттєво прискорити інференс детектора на цільових платформах серії NVIDIA Jetson за рахунок зниження точності обчислень з мінімальною втратою якості.

Другим суттєвим напрямом є переписання обчислювально критичних компонентів алгоритму ключових точок мовою C++. Поточна реалізація екстракції та зіставлення ключових точок з геометричною верифікацією через RANSAC виконується засобами Python, що накладає значні накладні витрати. Перенесення цих операцій на нативний рівень дозволить отримати багаторазове прискорення без зміни логіки алгоритму. Додатково можливе розпаралелювання обробки ключових точок між активними треками, оскільки ці обчислення є незалежними одне від одного.

Як крайній варіант оптимізації може розглядатись адаптивне вмикання модуля ключових точок лише в моменти невпевненості детектора. Проте такий підхід вносив би нестабільність у швидкодію системи — час обробки кадру став би залежати від поточної сцени — що є небажаним для бортового застосування з жорсткими вимогами до передбачуваності. Тому цей напрям доцільно розглядати лише після вичерпання попередніх можливостей оптимізації.

Покращення детектора

Оскільки детектор є критично важливим компонентом у парадигмі tracking-by-detection, підвищення його якості безпосередньо покращує роботу всієї системи. У ході роботи було емпірично встановлено, що з кожною ітерацією розширення датасету метрики детектора стабільно зростали, що свідчить про відсутність виходу на плато можливостей архітектури YOLOv11n. Це вказує на те, що розширення обсягу та підвищення варіативності навчальних даних є одним з найперспективніших і найдоступніших шляхів покращення.

Окрему увагу варто приділити балансуванню рідкісних класів, зокрема MLRS та APC, які наразі суттєво недопредставлені в датасеті. Також доцільним є збільшення частки урбаністичних сцен, що наразі становлять лише 5–7% матеріалів — це підвищить узагальнювальну здатність моделі в умовах міської забудови.

Розвиток алгоритму відстеження

Архітектура KPByteTrack передбачає можливість вибору між дескрипторами ключових точок AKAZE та ORB, проте в межах даної роботи детальне порівняння їхньої ефективності не проводилось — усі експерименти виконувались з дескриптором ORB. Систематичне порівняння різних дескрипторів за співвідношенням якість/швидкодія є логічним наступним кроком дослідження.

Крім того, для отримання статистично переконливих висновків щодо ефективності модифікації необхідне розширене дослідження: тестування на кількох рівнях якості детектора, з різними типами дескрипторів та з повторними прогонами для оцінки варіативності результатів. Це дозволить перейти від виявленої тенденції до обґрунтованих кількісних закономірностей.

Прикладні напрями

У перспективі систему доцільно інтегрувати з реальним бортовим обладнанням та провести польові випробування в умовах, наближених до реальних. Також можливим є розширення функціоналу модулями оцінки

відстані до цілі та підтримки наведення, що перетворило б систему відстеження на повноцінний компонент контуру автоматичного наведення. За умови накопичення достатнього обсягу даних окремим напрямом може стати повернення до задачі класифікації об'єктів — як додаткового інформаційного шару поверх надійної локалізації.

Висновки до розділу 5

У п'ятому розділі представлено методологію оцінки якості розробленої системи відстеження, проведено порівняльний аналіз класичного ByteTrack та розробленої модифікації KPByteTrack, а також окреслено напрями подальшого розвитку.

Описано принципи збору та підрахунку метрик багатооб'єктного відстеження з використанням бібліотеки TrackEval. Розглянуто математичні основи ключових метрик — MOTA, MOTP, IDF1, HOTA з її складовими DetA та AssA — а також обґрунтовано їхню пріоритетність у контексті задачі. Першочерговою визначено метрику IDF1, що безпосередньо вимірює стійкість ідентифікації об'єктів — ключову функцію трекера для автономного супроводу цілі. Окрему увагу приділено коректній агрегації метрик по всьому набору відео через підсумовування сирих лічильників, що забезпечує об'єктивне врахування складності кожного відеофрагменту.

Порівняльні експерименти проводились у двох конфігураціях — з повноцінно навченим та навмисно недотренованим детектором. Результати виявили чітку та послідовну закономірність: позитивний вплив модифікації KPByteTrack на якість відстеження зростає в міру погіршення якості детектора. За якісного детектора покращення першочергової метрики IDF1 було мінімальним (+0.0015), тоді як за слабкого детектора різниця зросла майже у 12 разів (+0.0175), супроводжуючись покращенням HOTA та суттєвим зниженням кількості пропущених об'єктів. Це підтверджує закладену в роботі гіпотезу про те, що компенсація недоліків детектора на

основі ключових точок є особливо цінною за умов недосконалого детектора — типової ситуації при обмеженому обсязі навчальних даних.

Водночас чесно зазначено обмеження дослідження: результати отримані на одному датасеті з одним типом дескриптора та двома рівнями якості детектора, тому коректно інтерпретуються як стійка тенденція, а не остаточно доведена універсальна закономірність.

Головним недоліком розробленої модифікації у її поточному вигляді є швидкодія: на відміну від класичного ByteTrack, що працює з 3.09-кратним запасом відносно реального часу, KPByteTrack досягає лише 0.40x реального часу через обчислювальну вартість обробки ключових точок. Це визначає необхідність оптимізації як першочергове завдання для практичного бортового застосування.

На основі проведеного аналізу окреслено напрями розвитку системи: квантування детектора через TensorRT та переписання критичних компонентів мовою C++ для підвищення швидкодії, розширення та балансування датасету для покращення детектора, систематичне порівняння дескрипторів ключових точок, а також інтеграцію з реальним бортовим обладнанням і польові випробування.

ВИСНОВКИ

У межах виконання магістерської роботи було спроєктовано та реалізовано автоматизовану систему підтримки прийняття рішень для ідентифікації та відстеження наземних об'єктів у реальному часі на основі комп'ютерного зору, орієнтовану на бортове застосування в умовах радіоелектронної боротьби та відсутності зв'язку з оператором.

Проведено аналіз проблематики застосування БПЛА в умовах сучасного збройного конфлікту. Встановлено, що вплив засобів радіоелектронної боротьби та фізичні обмеження радіогоризонту суттєво ускладнюють дистанційне керування апаратами — особливо для ударних дронів, які втрачають зв'язок під час кінцевого зниження на ціль. Це обґрунтовує потребу в автономних бортових системах обробки візуальної інформації. Здійснено огляд підходів до детекції та відстеження об'єктів, розглянуто еволюцію детекторів і алгоритмів трекінгу, а також основні метрики оцінки якості систем багатооб'єктного відстеження.

Сформовано власний датасет наземної техніки на основі 855 відеофрагментів з відкритих джерел загальною тривалістю 88.1 хвилини. Весь цикл — від збору матеріалів до розмітки у середовищі CVAT — виконано вручну без використання готових публічних датасетів. Розроблено власний конвеєр очищення відеоданих від небажаних артефактів на основі моделі ProPainter, а також власний чотирифазний алгоритм формування картинкового датасету з відеокадрів, що поєднує event-driven відбір, scale-стратифіковану дедуплікацію, глобальну дедуплікацію та перцептивне хешування.

Спроєктовано архітектуру системи за парадигмою tracking-by-detection, що поєднує детектор YOLOv11n з трекером ByteTrack. Розроблено власне розширення KPByteTrack, яке доповнює стандартний алгоритм модулем підтримки треків на основі ключових точок з дескрипторами AKAZE або ORB та геометричною верифікацією через RANSAC,

спрямованим на підвищення стійкості відстеження за умов недосконалого детектора.

Навчено та провалідовано детектор у двох варіантах — з чотирма агрегованими класами та з єдиним класом "ціль". Підбір гіперпараметрів виконано автоматизовано за допомогою Optuna. Порівняльний аналіз показав, що варіант з єдиним класом перевершує чотирикласовий за стандартними метриками, проте окрема class-agnostic оцінка довела, що локалізаційні можливості обох варіантів практично рівноцінні, а різниця зумовлена виключно складністю задачі класифікації. Для інтеграції з трекером обрано одно класовий варіант як такий, що демонструє кращі результати у виявленні малих об'єктів — критичної характеристики для бортового застосування.

Проведено оцінку ефективності системи відстеження за стандартизованими метриками. Експерименти підтвердили закладену гіпотезу: позитивний вплив модифікації KPByteTrack зростає в міру погіршення якості детектора, що проявляється у покращенні стійкості ідентифікації та зниженні кількості пропущених об'єктів. Водночас виявлено головне обмеження поточної реалізації — недостатню швидкодію через обчислювальну вартість обробки ключових точок, що наразі унеможливлює роботу в реальному часі.

Запропоновано напрями подальшого розвитку системи: квантування детектора через TensorRT та переписання критичних компонентів мовою C++ для досягнення реального часу, розширення та балансування датасету для покращення якості детектора, систематичне порівняння дескрипторів ключових точок, а також інтеграцію з реальним бортовим обладнанням і проведення польових випробувань.

Таким чином, поставлені у роботі мета та задачі повністю виконані. Розроблена система демонструє життєздатність обраного підходу до автономного відстеження наземних об'єктів з борту БПЛА, а виявлені обмеження та окреслені напрями розвитку формують чітку основу для подальшого вдосконалення системи до рівня практичного застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Redmon J. et al. You Only Look Once: Unified, Real-Time Object Detection // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1506.02640
2. A Zhang Y. et al. ByteTrack: Multi-Object Tracking by Associating Every Detection Box // European Conference on Computer Vision (ECCV). – 2022. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2110.06864.
3. Girshick R. et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2014. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1311.2524
4. Liu W. et al. SSD: Single Shot MultiBox Detector // European Conference on Computer Vision (ECCV). – 2016. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1512.02325
5. Lin T.-Y. et al. Focal Loss for Dense Object Detection (RetinaNet) // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2017. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1708.02002
6. Bewley A. et al. Simple Online and Realtime Tracking (SORT) // IEEE International Conference on Image Processing (ICIP). – 2016. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1602.00763
7. Wojke N. et al. Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT) // IEEE International Conference on Image Processing (ICIP). – 2017. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1703.07402

8. CVAT: Sekachev B. et al. Computer Vision Annotation Tool (CVAT) // GitHub. – Режим доступу: https://github.com/opencv/cvat
9. ProPainter: Zhou et al. ProPainter: Improving Propagation and Transformer for Video Inpainting // ICCV. – 2023. – Режим доступу: https://arxiv.org/abs/2309.03897
10. BoxMOT: Brostrom M. BoxMOT: pluggable SOTA tracking modules for segmentation, object detection and pose estimation models // GitHub. – Режим доступу: https://github.com/mikel-brostrom/boxmot
11. Ultralytics: Jocher G. et al. Ultralytics YOLO // GitHub. – Режим доступу: https://github.com/ultralytics/ultralytics
12. Bernardin K., Stiefelhagen R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics // EURASIP Journal on Image and Video Processing. – 2008. – [Електронний ресурс]. – Режим доступу: https://doi.org/10.1155/2008/246309
13. Ristani E., Solera F., Zou R., Cucchiara R., Tomasi C. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking // European Conference on Computer Vision (ECCV) Workshops. – 2016. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1609.01775
14. Luiten J., Osep A., Dendorfer P., Torr P., Geiger A., Leal-Taixé L., Leibe B. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking // International Journal of Computer Vision. – 2021. – Vol. 129. – [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2009.07736