НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО» НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 303.732.4

До захисту допущено
Завідувач кафедри ММСА
_____Оксана ТИМОЩУК
«___» _____ 2024р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку» зі спеціальності 124 «Системний аналіз»
на тему: «Виявлення та захоплення швидкісних рухомих цілей глибокими нейронними мережами»

Виконав:
Студент 2 курсу, групи КА-32мп
Бєздєтний Данііл Дмитрович _____

Науковий керівник:
Професор кафедри ШІ,
Доктор технічних наук, професор
Данилов Валерій Якович _____

Рецензент:
Професор, д.т.н. Новіков О.М. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань
Студент: _____

Київ-2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Бєздєтний Даніїл Дмитрович
(прізвище ім'я по батькові)

1. Тема дисертації: «Виявлення та захоплення швидкісних рухомих цілей глибокими нейронними мережами», науковий керівник дисертації професор кафедри ШІ, доктор технічних наук, професор Данилов Валерій Якович, затверджені наказом по університету від 07 листопада 2024 року № 5001-с.

2. Строк подання студентом дисертації: 05 грудня 2024 року

3. Об'єкт дослідження: швидкісні рухомі об'єкти, відстежувати та захоплювати за допомогою методів комп'ютерного зору та глибоких нейронних мереж

4. Предмет дослідження: алгоритми виявлення, трекінгу та вимірювання відстані до швидкісних об'єктів, які базуються на використанні глибоких нейронних мереж у поєднанні з алгоритмами трекінгу та стереоскопічними методами.

5. Перелік завдань, які потрібно розробити:

1) огляд предметної області;
2) відбір інструментів для практичної реалізації задачі;
3) аналіз та візуалізація дослідницьких даних;
4) розробка моделей для виявлення та захоплення об'єктів;
5) тестування та огляд отриманих моделей, аналіз та оцінка результатів.

6. Перелік графічного (ілюстративного) матеріалу:

1) рисунки;
2) таблиці;
3) презентація.

7. Орієнтовний перелік публікацій.

Бєздєтний Д. Д., Данилов В. Я. Прогнозування економічних процесів на основі новин з відкритих джерел за допомогою нейронних мереж// Системні науки та інформатика: збірник доповідей III Всеукраїнської науково-практичної конференції «Системні науки та інформатика», 25–29 листопада 2024 року, Київ. – К., НН ІПСА КПІ ім. Ігоря Сікорського, 2024., 7 стор.

8. Консультанти розділів дисертації.

Розділ	Прізвище, ініціали та	Підпис, дата	
		завдання видав	завдання прийняв
-	-	-	-

9. Дата видачі завдання: 02 вересня 2024 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації (МД)	Строк виконання етапів магістерської дисертації	Примітка
1	Затвердження теми МД. Ознайомлення зі структурою МД згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	01.09.2024-10.09.2024	виконано
2	Ознайомлення з ДСТУ 3008-2015.	11.09.2024-17.09.2024	виконано
3	Перший розділ. Історія розвитку технологій виявлення та захоплення об’єктів.	18.09.2024-01.10.2024	виконано
4	Другий розділ. Сучасні технології виявлення та відстеження швидкісних об'єктів.	02.10.2024-15.10.2024	виконано
5	Третій розділ. Програмна реалізація.	16.10.2024-29.10.2024	виконано
6	Третій розділ. Робота над практичним розділом магістерської дисертації.	30.10.2024-12.11.2024	виконано
7	Четвертий розділ. Стартап-проект.	13.11.2024-19.11.2024	виконано

Студент _____ Даніїл БЄЗДЄТНИЙ

Науковий керівник дисертації _____ Валерій ДАНИЛОВ

РЕФЕРАТ

Дипломна робота: 103 с., 23 рис., 22 табл., 28 джерел, 2 додатка.

ВИЯВЛЕННЯ ОБ'ЄКТІВ, ТРЕКІНГ, ГЛИБОКІ НЕЙРОННІ МЕРЕЖІ, YOLO, RCNN, CSRT

Об’єктом дослідження є швидкісні рухомі об’єкти.

Предметом дослідження є алгоритми виявлення, трекінгу та вимірювання відстані до швидкісних об’єктів, які базуються на використанні глибоких нейронних мереж у поєднанні з алгоритмами трекінгу та стереоскопічними методами.

Метою роботи є розробка та експериментальна перевірка інтегрованого підходу до виявлення, трекінгу та вимірювання відстані до швидкісних рухомих об’єктів у реальному часі на основі сучасних моделей глибокого навчання та алгоритмів тривимірного аналізу простору.

У роботі проаналізовано історичний розвиток технологій комп’ютерного зору, еволюцію методів детекції й трекінгу, а також роль апаратного забезпечення, такого як GPU та FPGA, у забезпеченні продуктивності. Досліджено сучасні алгоритми виявлення та відстеження, включно з Zero-Shot Detectors, Two-Shot Detectors, DeepSORT і ByteTrack. Практична частина роботи включала розробку моделі детекції, базованої на архітектурі YOLO, інтеграцію трекера CSRT і визначення метрик точності, таких як mAP, IOU, Precision, Recall.

Програмний продукт розроблено з використанням мови програмування Python.

ABSTRACT

Thesis: 103 pages, 23 figures, 22 tables, 28 sources, 2 appendices.

OBJECT DETECTION, TRACKING, DEEP NEURAL NETWORKS, YOLO, RCNN, CSRT

The object of research is high-speed moving objects.

The subject of research is algorithms for detection, tracking and distance measurement of high-speed objects, which are based on the use of deep neural networks in combination with tracking algorithms and stereoscopic methods.

The purpose of the work is the development and experimental verification of an integrated approach to detecting, tracking and measuring the distance to high-speed moving objects in real time based on modern deep learning models and three-dimensional spatial analysis algorithms.

The paper analyzes the historical development of computer vision technologies, the evolution of detection and tracking methods, as well as the role of hardware, such as GPUs and FPGAs, in providing performance. State-of-the-art detection and tracking algorithms are explored, including Zero-Shot Detectors, Two-Shot Detectors, DeepSORT, and ByteTrack. The practical part of the work included the development of a detection model based on the YOLO architecture, the integration of the CSRT tracker and the definition of accuracy metrics such as mAP, IOU, Precision, Recall.

The software product is developed using the Python programming language.

ЗМІСТ

ВСТУП

Розвиток технологій виявлення та захоплення швидкісних рухомих цілей має велике значення в сучасних системах безпеки, оборони та автономних транспортних засобах. Сучасні методи комп'ютерного зору, зокрема, використання глибоких нейронних мереж, дозволяють значно підвищити ефективність виявлення об'єктів, що швидко рухаються, а також забезпечити їх відстеження в реальному часі. Це особливо актуально в умовах, де важливими є висока точність, надійність та здатність працювати в складних умовах, таких як змінне освітлення, шум чи змішування об'єктів.

Метою даної роботи є дослідження та розробка методів виявлення і трекінгу швидкісних об'єктів за допомогою глибоких нейронних мереж, що дозволяють ефективно вирішувати ці задачі в реальному часі. Об’єктом дослідження є цифрові зображення та відео, а також використання алгоритмів штучного інтелекту для їх аналізу, виявлення та трекінгу швидко рухомих об'єктів. Робота охоплює як теоретичні аспекти застосування таких технологій, так і практичну реалізацію з використанням сучасних методів глибокого навчання.

Важливим аспектом роботи є вивчення алгоритмів детекції об'єктів, таких як Zero-Shot Detectors та Two-Shot Detectors, а також методів трекінгу, серед яких виділяються фільтри Калмана, CSRT, KCF, DeepSORT та ByteTrack. У роботі розглядається також використання стереозору та монокулярних методів для вимірювання відстані до об’єктів, що сприяє підвищенню точності та ефективності виявлення та трекінгу.

Застосування глибоких нейронних мереж у таких системах дозволяє досягати значних успіхів у вирішенні задач, пов'язаних з виявленням та відстеженням швидкісних об'єктів, що відкриває нові можливості для розвитку вищезгаданих технологій у різних сферах діяльності. Робота націлена на

вивчення цих перспектив та розробку більш ефективних методів для виявлення та трекінгу рухомих цілей.

1 ІСТОРІЯ РОЗВИТКУ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ ТА ЗАХОПЛЕННЯ ОБ’ЄКТІВ

1.1 Виникнення та розвиток комп’ютерного зору

Комп’ютерний зір — це технологія, яка дозволяє машинам «бачити» і розпізнавати об’єкти на зображеннях та у відео. Її розвиток почався ще в 1960-х роках, коли вчені намагалися навчити комп’ютери аналізувати прості візуальні дані. Хоча перші кроки здавалися перспективними, виявилося, що ця задача набагато складніша, ніж передбачалося.

Однією з перших ініціатив у цій сфері був Summer Vision Project в MIT [1]. Дослідники мали намір створити алгоритми для розпізнавання об'єктів протягом літа. Однак навіть прості завдання, такі як сегментування меж об’єктів на фотографії, вимагали більше часу та зусиль, ніж очікувалося. На початку комп’ютерне бачення використовувало прості підходи для аналізу зображень. Наприклад, поштові служби почали спроби розпізнавання рукописних цифр, щоб автоматично сортувати листи. У цих пристроях обробка зображень базувалася на простих логічних правилах і геометричних фігурах.

Протягом 70-х років були розроблені обчислювальні методи, які використовували математичні теорії для вивчення зображень. Одним з найпопулярніших перетворень є перетворення Хафа, яке дозволяє ідентифікувати лінії на зображеннях, фільтр Сейбл полегшує виділення меж об'єктів. Це полегшувало визначення меж об’єктів, що мало вирішальне значення для розпізнавання складніших сцен. У 1990-х роках комп’ютерне бачення почало використовувати статистичні методи та машинне навчання. Саме тоді алгоритми стали здатні «навчатися» на прикладах. Значним прогресом став метод каскадів Хаара, який сприяв швидкому розпізнаванню

облич на зображеннях. Цей підхід досі використовується в багатьох системах розпізнавання.

Також у цей час почали розроблятися алгоритми аналізу послідовності кадрів у відео, які є основою для відстеження об’єктів. У 2010-х роках розвиток технологій сприяв використанню глибоких нейронних мереж. Зокрема, архітектура AlexNet, яка була випущена в 2012 році, стала популярною завдяки своїй ролі в комп'ютерному зорі.

Завдяки використанню графічних процесорів (GPU) і великих баз даних ці моделі стали успішно розпізнавати об’єкти. Сьогодні комп’ютерний зір використовується в багатьох галузях, включаючи автономні автомобілі, які керують самостійно, а також медичну діагностику. Алгоритми стали настільки точними, що можуть не тільки розпізнавати об’єкти, але й випереджати їхню поведінку в реальному часі.

Сфера комп’ютерного зору значно розвинулась, від простих інструкцій до складних нейронних мереж. Це технологія, яка постійно розвивається і використовується в різних галузях.

1.2 Перші алгоритми детекції об'єктів

Виявлення об’єктів є одним із основних завдань комп’ютерного зору, яке передбачає ідентифікацію та розміщення конкретних об’єктів на зображеннях або відео. На початку розвитку комп’ютерного зору ця технологія була менш складною та більш обмеженою у своїх ресурсах, але ці основи все ще були необхідні для подальшого розвитку технології. Давайте обговоримо основні етапи еволюції технології виявлення об'єктів.

1.2.1 Детекція за допомогою шаблонів

Одним із найперших методів виявлення об’єктів є виявлення шаблонів. Метод заснований на пошуку певних зображень або форм, які повинні бути схожі на заданий візерунок. Алгоритм працює шляхом порівняння частини зображення з шаблоном, який, як вважають, відображає важливі характеристики об’єкта, який шукають. Якщо шаблон розташований десь на зображенні, система вважає це наявністю об'єкта. Найпростішим методом є кореляція — вимірювання подібності між шаблоном і підозрілими ділянками зображення. Однак цей підхід має значні обмеження. Він не стійкий до змін масштабу, орієнтації чи освітлення об’єкта, тому підходить лише для дуже обмежених випадків, коли об’єкт має чітко визначену та постійну форму.

1.2.2 Детекція за допомогою гістограм орієнтованих градієнтів

Наступним важливим етапом у розвитку алгоритмів виявлення об'єктів стало використання орієнтованих градієнтних гістограм, запропонованих у 2005 році Нібалі Бернштейном і Крістофером Лаугерою. Цей метод став популярним, оскільки він успішно виявляє об’єкти навіть у складних умовах, таких як різне освітлення чи фоновий шум.

Основна ідея HOG [2] полягає в тому, що замість пошуку всього об’єкта система намагається знайти його окремі локальні особливості, зокрема градієнти інтенсивності пікселів. Вони допомагають визначити контури і структуру об'єктів, тим самим підвищуючи точність виявлення.

Для реалізації алгоритмів виявлення на основі HOG використовуються траєкторії ознак, які записують напрямок та інтенсивність градієнтів у невеликих підобластях (локальних ділянках) зображення. Потім ці функції

використовуються для класифікації, що дозволяє виявляти певні об’єкти, наприклад людей або транспортні засоби, навіть за різних фонових умов.

Метод HOG став основою для створення багато чисельних систем детекції, зокрема, системи детекції пішоходів, яка широко використовувалася в мобільних додатках та автономних транспортних системах.

1.2.3 Детекція за допомогою каскадних класифікаторів (Haar-like features)

У середині 2000-х років був запропонований один з найвідоміших методів виявлення об'єктів - каскадний класифікатор на основі ознак Хаара. Цей метод був запропонований Полом Віолою та Майклом Джонсоном у 2001 році та популярний завдяки своїй швидкості та ефективності в реальному часі.

Цей метод заснований на функціях Хаара, які є прямокутними візерунками, які дозволяють вимірювати різницю між інтенсивністю пікселів у різних частинах зображення. Оскільки ці функції мають просту структуру, їх можна швидко обчислити, що дозволяє алгоритму працювати в реальному часі.

Одним із ключових моментів цього методу є використання каскадних класифікаторів. Алгоритм працює в кілька етапів, кожен з яких поступово відкидає непотрібні частини зображення, зосереджуючи ресурси на найбільш перспективних напрямках. Це дозволяє значно скоротити час обробки, оскільки система спочатку відкидає більшість непотрібних об’єктів і лише на завершальному етапі сортує найскладніші випадки.

Цей метод був широко застосовуваний для детекції облич, а також у багатьох комерційних програмах, таких як системи відеоспостереження або автономні автомобілі.

1.2.4 Розпізнавання об'єктів за допомогою методів машинного навчання

Методи, засновані на методах машинного навчання, стали популярними в кінці 2000-х з використанням простих функцій і алгоритмів. Зокрема, для виявлення об’єктів активно використовується метод опорних векторів (SVM) [4], який дозволяє класифікувати об’єкти на основі наявних навчальних даних.

Методи опорних векторів використовують принцип пошуку найкращої гіперплощини, яка розділяє різні класи даних. У контексті виявлення об’єктів SVM використовувався для ідентифікації об’єктів на зображеннях після попередньої обробки (наприклад, шляхом вилучення функцій HOG). За допомогою цієї технології можна досягти високої точності в задачі ідентифікації людей, транспортних засобів та інших об'єктів.

Підхід SVM також стає важливим кроком до більш гнучких і точних моделей, оскільки дозволяє обробляти великі обсяги даних і вдосконалювати моделі складних варіацій об’єктів.

1.2.5 Перехід до глибоких нейронних мереж

Наприкінці 2000-х і на початку 2010-х виявлення об’єктів стало значно ефективнішим завдяки впровадженню глибоких нейронних мереж. Одним із найбільших проривів стало впровадження згорткових нейронних мереж (CNN), які дозволили автоматично витягувати ознаки із зображень, не вказуючи шаблони чи характеристики вручну.

Нейронні мережі, такі як YOLO, Fast R-CNN і SSD, стали основою сучасних систем виявлення об’єктів. Вони здатні працювати в режимі

реального часу і ідентифікувати кілька об'єктів на одному зображенні, забезпечуючи високу точність і швидкість роботи.

Ці алгоритми стали основою для багатьох додатків від систем безпеки до самокерованих транспортних засобів, змінивши підхід до проблеми виявлення об’єктів на цифрових зображеннях. Перехід від простих методів виявлення об’єктів до більш складних методів, заснованих на машинному навчанні та глибоких нейронних мережах, став вирішальним етапом у розвитку комп’ютерного зору. Перше використання простих алгоритмів заклало основу для більш складних моделей, які зараз використовуються в широкому діапазоні додатків від дронів до автоматизованих систем спостереження.

1.3 Еволюція трекінгу: від класичних методів до нейромереж

Відстеження об’єктів є ключовим завданням комп’ютерного зору та передбачає відстеження рухомих об’єктів у відео або послідовності зображень. Завдання відстеження мають широке застосування в багатьох сферах, таких як відеоспостереження, автономні транспортні засоби, аналіз руху та робототехніка. З розвитком комп’ютерної техніки та алгоритмів методи стеження зазнали значних змін – від класичних методів до сучасних методів з використанням нейронних мереж.

1.3.1 Класичні підходи до трекінгу

Ранні методи відстеження базувалися на математичних моделях і використовували прості принципи відстеження об’єктів. Один із основних методів заснований на пошуку змін між послідовними кадрами, наприклад,

оптичний потік, який намагається оцінити рух пікселів на основі змін інтенсивності пікселів. Також використовується модель фільтрації, яка використовує попередні дані для прогнозування траєкторії об’єкта з урахуванням шуму та невизначеності.

Ці класичні методи добре працюють в умовах простих сцен, але мають обмеження в більш складних завданнях, таких як швидко рухомі об'єкти або їх часткове приховування. Крім того, їхні можливості обмежені, коли потрібно відстежувати декілька об’єктів одночасно.

1.3.2 Перехід до кореляційних фільтрів

З розвитком комп’ютерних технологій і вдосконаленням алгоритмів з’явилися нові методи підвищення ефективності та точності відстеження. У цій категорії важливі методи, засновані на кореляційних фільтрах. Вони відстежують об’єкти за допомогою порівняння шаблонів між кадрами. Ці методи можуть зменшити вплив шуму та покращити відстеження в складних умовах, наприклад, частково затриманих або швидко рухомих об’єктів. Пов’язані методи значно підвищують швидкість і точність відстеження, дозволяючи одночасно обробляти кілька об’єктів у сцені. Однак вони все ще мають деякі обмеження в обробці великих обсягів даних у реальному часі та потребують подальшого вдосконалення.

1.3.3 Перехід до глибоких нейронних мереж

Одним із найбільших досягнень у розвитку трекінгу стало використання глибоких нейронних мереж. Цей метод докорінно змінює метод відстеження,

оскільки нейронна мережа здатна автоматично виділяти важливі особливості об’єкта та його рух, таким чином значно підвищуючи точність і стабільність відстеження в складних умовах. Завдяки здатності нейронних мереж адаптуватися до мінливих умов, таких як різне освітлення, об’єкти, що швидко рухаються, і складний фон, ці методи стали новим стандартом відстеження об’єктів. Використання нейронних мереж не тільки покращує якість відстеження, але й робить його доступним у режимі реального часу, що вкрай важливо для додатків в автономних системах і відеоспостереження. Крім того, такі мережі, як Siamese Networks та інші подібні архітектури, здатні відстежувати декілька об’єктів одночасно з високою точністю.

1.3.4 Сучасні підходи та комбіновані методи

Сьогодні більшість сучасних алгоритмів відстеження поєднують різні методи для отримання кращих результатів. Наприклад, нейронні мережі використовуються для первинного виявлення об’єктів, тоді як кореляційні фільтри або методи фільтрації допомагають покращити відстеження, особливо коли потрібна висока швидкість і обробка в реальному часі. Такий комбінований підхід забезпечує високу точність відстеження навіть у складних умовах, коли на сцені з’являються нові перешкоди або об’єкти змінюють своє положення чи вигляд.

1.3.5 Перспективи розвитку трекінгу

У майбутньому, з розвитком технологій і появою нових нейронних мереж, відстеження цілей буде продовжувати вдосконалюватися. Очікується,

що новий метод стане більш стійким до складних умов, таких як швидке переміщення, часткове приховування об'єктів, зміни зовнішнього вигляду і збільшення кількості одночасно відстежуваних об'єктів. Важливим напрямком стане також інтеграція трекінгу з іншими технологіями, такими як стереоскопічний і змішана реальність, що дозволить створювати більш потужні та точні системи трекінгу в реальному часі.

Таким чином, еволюція трекінгу об'єктів від класичних методів до сучасних підходів на основі глибоких нейронних мереж стала значним кроком вперед, що дозволило реалізувати ефективніші і більш точні системи для відстеження об'єктів в складних умовах. Подальші розробки в цій галузі обіцяють ще більші досягнення в таких сферах, як автономні транспортні засоби, відеоспостереження, робототехніка та інші.

1.4 Вплив апаратного забезпечення на розвиток технологій (GPU, FPGA)

Апаратне забезпечення відіграє ключову роль у розвитку технологій комп'ютерного зору, зокрема в задачах детекції та трекінгу об'єктів. Різноманітні апаратні рішення, зокрема графічні процесори (GPU) та програмовані логічні інтегральні схеми (FPGA) [5], значно впливають на ефективність і швидкість виконання складних обчислювальних операцій, що є критичними для сучасних систем на основі штучного інтелекту та машинного навчання. Розглянемо, як ці технології змінили картину обробки зображень та розвитку алгоритмів.

1.4.1 Роль графічних процесорів (GPU)

Графічні процесори (GPU) стали основним компонентом для прискорення обчислювальних задач, пов'язаних з обробкою великих обсягів даних, зокрема в задачах машинного навчання та глибокого навчання. GPU спочатку розроблялися для прискорення графічних процесів, таких як рендеринг зображень, однак з часом їх можливості були адаптовані для виконання паралельних обчислень, що стало революцією для обчислювальних завдань в галузі штучного інтелекту. Однією з основних переваг GPU є їх здатність виконувати тисячі паралельних обчислень одночасно. Це дозволяє значно пришвидшити навчання нейронних мереж, оскільки в таких мережах на кожному етапі обробляється велика кількість даних, що потребує значних обчислювальних ресурсів. У контексті задач детекції та трекінгу об'єктів це означає можливість обробляти відео в реальному часі, що є критично важливим для застосувань в автономних транспортних системах, відеоспостереженні та дроновому моніторингу. Завдяки використанню GPU, алгоритми, що раніше вимагали багато часу для тренування, стали здатні досягати високої точності за коротший період. Наприклад, сучасні моделі детекції об'єктів, такі як YOLO та Faster R-CNN [7], значно прискорились завдяки використанню графічних процесорів, що дозволяє здійснювати обробку відео та зображень у реальному часі. Це також стало можливим завдяки інтеграції GPU з популярними бібліотеками для глибокого навчання, такими як TensorFlow, PyTorch і Keras.

1.4.2 Вплив FPGA на технології

Програмовані логічні інтегральні схеми (FPGA) представляють собою ще один важливий компонент апаратного забезпечення, який знайшов своє

застосування в розвитку технологій комп'ютерного зору та штучного інтелекту. FPGA дозволяють розробляти спеціалізовані апаратні рішення, що можуть бути налаштовані для виконання конкретних обчислювальних задач, забезпечуючи високу ефективність та низьке енергоспоживання.

Основною перевагою FPGA є їх гнучкість. Вони можуть бути програмовані для виконання специфічних операцій, що дозволяє створювати апаратні рішення, оптимізовані для конкретних задач. Наприклад, FPGA можуть бути використані для прискорення специфічних етапів обробки зображень, таких як фільтрація або обчислення характеристик, що є необхідними для детекції об'єктів. У порівнянні з традиційними процесорами, FPGA забезпечують значно більшу пропускну здатність і менший час затримки при виконанні операцій, що є критично важливим для задач реального часу, таких як трекінг об'єктів у відео.

Використання FPGA в задачах трекінгу об'єктів дозволяє створювати високопродуктивні системи, здатні ефективно відстежувати рухомі об'єкти в умовах з обмеженими ресурсами, наприклад, у вбудованих системах або робототехніці. FPGA також використовуються для створення спеціалізованих акселераторів для нейронних мереж, що дозволяє досягати значних переваг у швидкості обробки порівняно з традиційними процесорами, зберігаючи при цьому низьке енергоспоживання.

1.4.3 Спільне використання GPU та FPGA

В останні роки з'явилася тенденція до комбінування можливостей GPU та FPGA для створення гібридних систем, що поєднують переваги обох технологій. Наприклад, в таких системах GPU може використовуватися для виконання інтенсивних обчислень, таких як тренування нейронних мереж, а FPGA — для оптимізації специфічних операцій, пов'язаних з обробкою

зображень або прискоренням окремих етапів роботи нейронних мереж. Такий підхід дозволяє досягати високої ефективності та оптимізувати використання апаратних ресурсів.

Комбінація GPU та FPGA знайшла своє застосування в багатьох реальних проектах, таких як автономні транспортні засоби, де потрібно одночасно обробляти величезний потік даних у реальному часі, а також в робототехніці, де важлива максимальна швидкість обробки та точність трекінгу.

1.4.4 Перспективи розвитку

У майбутньому очікується, що розвиток апаратного забезпечення, зокрема GPU та FPGA, продовжуватиме прискорювати технології обробки зображень і штучного інтелекту. З новими інноваціями в обробці зображень, такими як використання більш ефективних архітектур нейронних мереж та удосконалення апаратних платформ, можна очікувати ще більші досягнення в області реального часу трекінгу об'єктів і детекції.

Розвиток спеціалізованих чіпів, таких як нейронні процесори (NPU) і інші апаратні рішення для машинного навчання, відкриває нові можливості для високопродуктивних систем. Технології на основі FPGA та GPU продовжать зростати і розвиватися, забезпечуючи ще більші досягнення в забезпеченні швидкості, точності і ефективності обробки даних для задач комп'ютерного зору та штучного інтелекту.

Таким чином, апаратне забезпечення, зокрема графічні процесори та програмовані логічні інтегральні схеми, відіграють вирішальну роль у розвитку технологій комп'ютерного зору, забезпечуючи швидку і ефективну обробку великих обсягів даних, що є критичним для задач детекції, трекінгу об'єктів і роботи з нейронними мережами.

1.5 Використання глибокого навчання в детекції та трекінгу

Глибоке навчання (deep learning) стало основним інструментом у розвитку сучасних технологій детекції та трекінгу об'єктів. Використання нейронних мереж з багатьма шарами дозволяє автоматично навчатися витягувати важливі ознаки з великих обсягів даних, що є надзвичайно важливим для задач, пов'язаних з обробкою зображень і відео. В останні роки глибоке навчання продемонструвало значні досягнення в цих областях, зокрема завдяки розвитку таких архітектур, як згорткові нейронні мережі (CNN) [8], рекурентні нейронні мережі (RNN) та трансформери. Розглянемо, як глибоке навчання впливає на детекцію та трекінг об'єктів.

1.5.1 Глибоке навчання для детекції об'єктів

Детекція об'єктів є однією з основних задач комп'ютерного зору, що полягає у виявленні і локалізації об'єктів на зображеннях або відео. Раніше для розв'язання цієї задачі використовувались традиційні методи, які включали використання ручних ознак, таких як контури або текстури. Однак з появою глибокого навчання ці методи були значно покращені, оскільки нейронні мережі здатні автоматично витягувати релевантні ознаки з сирих даних, що дозволяє досягати високої точності та ефективності.

Один з найважливіших проривів у детекції об'єктів стався з появою згорткових нейронних мереж (CNN), які стали стандартом для багатьох задач обробки зображень. CNN здатні автоматично вивчати ознаки зображень на різних рівнях абстракції — від простих текстур до складних структур, таких як об'єкти або сцени. Це дозволяє не тільки точно виявляти об'єкти, а й робити це за менший час.

Однією з основних архітектур, яка змінила підхід до детекції, є YOLO (You Only Look Once) [9]. YOLO — це нейронна мережа, яка дозволяє виконувати детекцію об'єктів у реальному часі. Завдяки тому, що мережа прогнозує координати об'єктів і їх категорії за один прохід через мережу, вона значно прискорює процес детекції, що є важливим для задач реального часу. Іншою важливою архітектурою є Faster R-CNN, яка використовує регіональні пропозиції для локалізації об'єктів. Вона також дозволяє досягати високої точності у визначенні місць, де знаходяться об'єкти на зображенні.

Завдяки глибокому навчанню стало можливим виявлення широкого спектра об'єктів, від звичайних предметів до складних та малопомітних об'єктів, таких як пішоходи чи транспортні засоби на великих відстанях або в умовах поганої видимості.

1.5.2 Глибоке навчання для трекінгу об'єктів

Трекінг об'єктів — це процес відстеження руху об'єктів на серіях зображень або відео. Це завдання є складним, оскільки воно вимагає не тільки правильної ідентифікації об'єктів, але й здатності їх відстежувати через різні кадри, враховуючи зміни в позиції, масштабі, орієнтації та інші варіації, що виникають під час руху.

Традиційні методи трекінгу, такі як алгоритм Калмана або кореляційні фільтри, здебільшого спиралися на геометричні властивості та апаратуру. Однак ці методи не завжди справляються з складними ситуаціями, коли об'єкти перекриваються, змінюють свій вигляд або перебувають у складних умовах освітлення. Глибоке навчання дозволяє подолати ці обмеження завдяки здатності автоматично розпізнавати й відстежувати об'єкти на основі ознак, що генеруються в глибоких шарах нейронної мережі.

Одним з основних підходів до трекінгу на основі глибокого навчання є використання так званих методів трекінгу, заснованих на детекціях (tracking-by-detection). У цьому підході спочатку виконується детекція об'єктів за допомогою нейронної мережі, а потім ці детекції відстежуються через різні кадри за допомогою алгоритмів, таких як DeepSORT (Deep Learning-based SORT) або ByteTrack [10]. DeepSORT використовує глибоку нейронну мережу для створення унікальних векторів для кожного об'єкта, що дозволяє ефективно відстежувати об'єкти через кадри, навіть якщо вони зникають і знову з'являються в кадрі.

Інші підходи до трекінгу використовують рецидивні нейронні мережі (RNN) або трансформери, які дозволяють обробляти послідовності кадрів і враховувати часову залежність у русі об'єктів. Використання таких архітектур дозволяє ще точніше відстежувати рухи об'єктів, оскільки ці моделі здатні зберігати інформацію про попередні кадри і передбачати позиції об'єктів у майбутньому.

1.5.3 Переваги використання глибокого навчання в реальному часі

Однією з головних переваг використання глибокого навчання в задачах детекції та трекінгу є здатність працювати в реальному часі. Це стало можливим завдяки значному прискоренню обчислень завдяки використанню сучасного апаратного забезпечення, зокрема графічних процесорів (GPU), які можуть одночасно виконувати тисячі паралельних операцій.

Завдяки здатності швидко обробляти відео в реальному часі, системи на основі глибокого навчання можуть використовуватись у таких практичних застосунках, як автономні транспортні засоби, відеоспостереження, безпілотні літальні апарати (дрони), а також в робототехніці. Наприклад, дрони, оснащені системами на базі глибокого навчання, можуть автономно відстежувати об'єкти,

знаходити їх і коригувати свою траєкторію, що дозволяє використовувати їх у різних галузях, від моніторингу навколишнього середовища до доставки вантажів.

1.5.4 Перспективи розвитку

Незважаючи на значний прогрес у використанні глибокого навчання для детекції та трекінгу об'єктів, існують ще багато викликів, які потрібно подолати. Одним з основних напрямків розвитку є поліпшення точності алгоритмів при роботі в складних умовах, таких як низька освітленість, часткове перекриття об'єктів, або складні фонові умови.

Перспективними є також розробки нових архітектур нейронних мереж, які дозволяють ефективно використовувати менші обсяги даних або працювати в умовах обмежених ресурсів, таких як на мобільних пристроях чи вбудованих системах. Зокрема, розвиток трансформерних моделей, таких як DETR [12] (Detection Transformer), який використовує механізм самовнимання, відкриває нові можливості для точного й ефективного трекінгу об'єктів.

Таким чином, глибоке навчання значно покращило можливості детекції та трекінгу об'єктів, дозволяючи створювати точні й ефективні системи, здатні працювати в реальному часі. У майбутньому можна очікувати ще більші досягнення завдяки подальшому розвитку нейронних мереж, покращенню апаратного забезпечення і новим інноваціям у галузі штучного інтелекту.

1.6 Висновки до розділу 1

У цьому розділі було розглянуто історію розвитку технологій виявлення та захоплення швидкісних об’єктів, починаючи від зародження комп’ютерного зору та перших алгоритмів детекції, до сучасних підходів, що базуються на глибокому навчанні.

Аналіз еволюції трекінгу показав поступовий перехід від класичних методів, таких як кореляційні фільтри, до використання нейронних мереж, що дозволяють підвищити точність і швидкість обробки даних. Важливу роль у цьому процесі відіграло вдосконалення апаратного забезпечення, зокрема розвиток графічних процесорів (GPU) та програмованих логічних інтегральних схем (FPGA), які суттєво розширили можливості комп’ютерного зору в реальному часі.

Глибоке навчання стало ключовою технологією для виявлення та захоплення об’єктів завдяки здатності працювати з великими обсягами даних і забезпечувати високу точність. Переваги таких підходів дозволяють ефективно застосовувати їх у різних практичних завданнях, включаючи захоплення швидкісних об’єктів у реальному часі.

Таким чином, розвиток алгоритмів детекції та трекінгу тісно пов’язаний із прогресом в апаратному забезпеченні та появою методів глибокого навчання, що відкриває нові перспективи для створення більш точних та ефективних систем.

2 СУЧАСНІ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ ШВИДКІСНИХ ОБ'ЄКТІВ

2.1 Алгоритми виявлення об'єктів

У сучасних системах комп'ютерного зору виявлення об'єктів є важливою задачею, яка включає різноманітні підходи. Серед нових тенденцій у цій області виділяються нульові (zero-shot) та дво-шотні (two-shot) детектори. Обидва ці підходи мають свої особливості, переваги та недоліки, які варто розглянути.

2.1.1 Zero-Shot Detectors

Zero-shot детектори представляють собою алгоритми, здатні виявляти об'єкти, для яких у них не було раніше навчальних зразків. Цей підхід ґрунтується на здатності моделі узагальнювати знання про об'єкти, використовуючи семантичні описи чи атрибути, щоб зробити висновки про нові класи об'єктів. Наприклад, замість того щоб навчатися на конкретних зразках, zero-shot детектори можуть використовувати векторні представлення класів у семантичному просторі, які описують характерні риси об'єкта, такі як форма, колір або текстура.

Цей підхід особливо корисний у сценаріях, де дані для певних класів об'єктів обмежені або взагалі відсутні. Zero-shot детектори можуть зменшити витрати на збори даних та аннотації, що є важливим у багатьох практичних застосуваннях. Проте, точність таких детекторів часто нижча в порівнянні з традиційними методами, оскільки їхня продуктивність залежить від якості семантичних описів та здатності моделі узагальнювати.

2.1.2 Two-Shot Detectors

Two-shot детектори [13], або детектори з двома зразками, вимагають наявності щонайменше двох зразків для кожного класу об'єктів під час навчання. Цей підхід забезпечує більше контексту для моделі, що дозволяє їй навчатися на конкретних прикладах та поліпшувати точність виявлення. Two-shot детектори зазвичай застосовують у сценаріях, де є можливість отримати невелику кількість аннотованих даних для нових класів, що дозволяє моделі швидко адаптуватися до нових умов.

Цей метод забезпечує більш високу точність виявлення порівняно з zero-shot, оскільки модель має можливість навчатися на реальних прикладах. Однак він вимагає більше витрат на збори та аннотації даних, що може бути проблемою у випадках, коли нові класи з’являються рідко або їх важко зібрати.

2.1.3 Порівняння Zero-Shot та Two-Shot Detectors.

Ключові відмінності між zero-shot і two-shot детекторами полягають у їхній здатності до навчання та залежності від даних. Zero-shot детектори можуть працювати без аннотованих даних, але їхня точність часто страждає через обмежені можливості узагальнення. З іншого боку, two-shot детектори забезпечують вищу точність, але вимагають збору даних та аннотацій для кожного нового класу.

При виборі між цими підходами важливо враховувати специфіку задачі. Якщо у вас є можливість отримати обмежену кількість аннотованих даних для нових класів, two-shot підхід може бути оптимальним. У протилежному випадку, якщо зразки недоступні, zero-shot детектори можуть стати ефективним рішенням для виявлення об'єктів у нових, не бачених раніше класах.

Загалом, обидва підходи мають свої переваги та недоліки, які можуть бути використані в різних контекстах виявлення та відстеження дронів. У зв'язку з цим важливо провести додаткове тестування та оцінку ефективності кожного з методів у конкретних умовах застосування.

2.2 Алгоритми відстеження

Відстеження об'єктів є важливою складовою комп'ютерного зору, що має на меті стежити за рухомими об'єктами в реальному часі. Це може включати в себе дрони, автомобілі, пішоходів та інші об'єкти. Відстеження є складним завданням через різноманітність можливих перешкод, таких як зміна масштабу, повороти, освітлення, а також взаємодію з іншими об'єктами. У цій секції ми розглянемо різні алгоритми відстеження, включаючи математичні фільтри, такі як фільтр Калмана, CSRT [14], KCF [15], а також сучасні алгоритми, такі як DeepSORT та ByteTrack. Математичні фільтри використовуються для обробки сигналів і даних з метою зменшення шуму та покращення точності відстеження. Один із найвідоміших методів — це фільтр Калмана, який є рекурсивним фільтром, що оцінює стан динамічної системи з урахуванням шуму в даних.

2.2.1 Фільтр Калмана

Фільтр Калмана застосовується для оцінки стану рухомого об'єкта на основі попередніх даних та нових вимірювань, працюючи у два послідовні етапи: прогнозування та корекції.

На етапі прогнозування фільтр оцінює майбутній стан об'єкта, використовуючи модель руху та попередні стани. Після отримання нових вимірювань на етапі корекції фільтр оновлює свою оцінку, інтегруючи ці дані для уточнення прогнозу. Цей метод є потужним інструментом для роботи в динамічних середовищах, однак його ефективність залежить від точності обраних моделей руху та вимірювань.

2.2.2 CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability)

CSRT — це алгоритм, що базується на використанні кореляційних фільтрів для відстеження об'єктів, з акцентом на просторову та каналізовану інформацію для забезпечення високої точності.

Просторова надійність дозволяє алгоритму враховувати зміни в зображенні, зменшуючи вплив шуму та покращуючи стійкість до змін середовища. Використання даних із різних кольорових каналів підвищує точність визначення об'єкта, особливо в умовах низького контрасту.

Завдяки цьому CSRT демонструє ефективність у відстеженні об'єктів із різними формами, розмірами та в ситуаціях, коли вони можуть перекриватися або змінювати форму.

2.2.3 KCF (Kernelized Correlation Filters)

CSRT — це сучасний алгоритм, що використовує кореляційні фільтри для відстеження об'єктів і виділяється своєю здатністю враховувати як просторову надійність, так і каналізовану інформацію. Просторова надійність

дозволяє алгоритму ефективно оцінювати зміни в зображенні, зберігаючи стійкість до шуму, що робить його адаптивним до різних умов середовища. Завдяки цьому CSRT здатний зберігати стабільність навіть при значних змінах у фоні або часткових перекриттях об'єктів.

Інформація, яку CSRT отримує з різних кольорових каналів, забезпечує більш точне визначення об'єкта, особливо в умовах низької контрастності, коли традиційні алгоритми можуть втрачати точність. Алгоритм добре підходить для роботи з об'єктами, які мають змінну форму або розмір, і демонструє високу продуктивність у складних сценаріях, де можливі перекриття об'єктів. Завдяки своїм особливостям CSRT є універсальним інструментом для трекінгу в умовах, що вимагають точності та адаптивності.

2.2.4 DeepSORT

DeepSORT є розширенням методу SORT (Simple Online and Realtime Tracking), що базується на використанні фільтра Калмана для прогнозування траєкторій об'єктів. Основною інновацією DeepSORT є інтеграція методів глибокого навчання, які дозволяють значно підвищити точність асоціації між виявленими об'єктами в різних кадрах. Замість використання лише просторових даних, як у SORT, DeepSORT аналізує комплексні ознаки об'єктів, отримані за допомогою нейронних мереж.

Нейронна мережа в DeepSORT витягує високорівневі ознаки, такі як колір, текстура та форма, що дозволяє ідентифікувати об'єкти навіть за умов складного середовища [16]. Використання цих ознак дозволяє алгоритму ефективніше проводити асоціацію між об'єктами в послідовних кадрах, враховуючи як їх просторові характеристики, так і візуальні особливості. Завдяки цьому DeepSORT може точно слідкувати за об'єктами навіть у разі

часткових перекриттів, раптового зникнення з кадру або з'явлення в нових місцях.

DeepSORT демонструє високу продуктивність у складних умовах трекінгу, забезпечуючи стійкість і надійність роботи для об'єктів із різними траєкторіями руху. Це робить алгоритм популярним вибором для систем моніторингу та відеоаналітики в реальному часі.

2.2.5 ByteTrack

ByteTrack — це сучасний алгоритм трекінгу, який покращує точність відстеження в складних умовах, таких як швидкий рух об'єктів або їхнє з'явлення і зникнення в кадрі. Його ключова перевага полягає у здатності враховувати всі виявлені об'єкти, навіть ті, що мають низьку ймовірність асоціації, що допомагає зменшити втрати треків. Однак цей підхід може бути менш ефективним у випадках з дуже швидкими об'єктами, такими як дрони, де динаміка руху може виходити за межі передбачуваних моделей.

Алгоритм використовує всі доступні об'єкти незалежно від їхнього статусу, що покращує точність у середовищах із середньою щільністю об'єктів, але може генерувати помилкові асоціації в умовах високої швидкості руху та частих змін траєкторії. ByteTrack оптимізований для роботи в реальному часі, що робить його придатним для задач моніторингу, проте у сценаріях із дронами він може стикатися зі складнощами, такими як часткове перекриття або швидкі зміни положення.

Хоча ByteTrack показує хороші результати у більшості стандартних сценаріїв, для трекінгу дронів можуть знадобитися додаткові оптимізації або поєднання з іншими методами, що враховують специфіку аеродинамічних траєкторій і велику варіативність швидкостей.

2.3 Використання стереозору та монокулярних методів вимірювання

Вимірювання дистанції до об'єктів у реальному часі є важливим завданням у комп'ютерному зорі, особливо при відстеженні швидко рухомих цілей, таких як дрони. Для цього існує кілька підходів, серед яких стереозір і монокулярні методи. Стереозір ґрунтується на використанні двох камер, розташованих на певній відстані одна від одної. Це дозволяє захоплювати зображення одного й того ж об'єкта з різних ракурсів, що відкриває можливості для побудови тривимірної моделі сцени. Основними етапами цього процесу є калібрування камер, захоплення зображень, стерео-розпізнавання і обчислення відстані.

Калібрування камер є критично важливим етапом, адже правильне налаштування забезпечує точність вимірювань, визначаючи внутрішні та зовнішні параметри. Після калібрування обидві камери одночасно фіксують зображення, що дозволяє отримати дві проекції об'єкта. Далі виконується стерео-розпізнавання, яке реалізується через алгоритми кореляції або глибокого навчання. Цей етап включає в себе відшукання відповідностей між точками на обох зображеннях, що є ключовим для побудови карти глибини. Використовуючи геометричні принципи, такі як тригонометрія, стає можливим обчислення відстані до об'єкта на основі різниці в положеннях точок у двох зображеннях. Стереозір має безсумнівні переваги, зокрема можливість точно оцінювати глибину та формувати 3D-моделі об'єктів, однак існують і недоліки. Зокрема, висока вартість апаратури та складність обробки даних можуть обмежити його застосування в певних випадках.

На відміну від стереозору, монокулярні методи використовують лише одну камеру для вимірювання дистанції, що спрощує реалізацію та зменшує вартість системи. Хоча такі методи зазвичай є менш точними, вони пропонують альтернативні рішення, які можуть бути ефективними в певних умовах. Основними підходами в монокулярних методах є методи на основі руху, які

використовують інформацію про зміни положення камери або об'єкта для оцінки глибини. Спостерігаючи за об'єктами на фоні, системи можуть вимірювати відстань, що часто реалізується через аналіз відео. Окрім того, сучасні нейронні мережі можуть бути навчені для оцінки глибини з однокадрових зображень. Це дозволяє виконувати вимірювання без необхідності в додаткових камерах. Інший підхід полягає в використанні контексту, коли монокулярні системи беруть до уваги відомі розміри певних елементів сцени для оцінки дистанції.

Попри свої переваги, монокулярні методи мають й обмеження, адже вони не надають прямої інформації про глибину, що може призводити до неточностей. Однак їх легкість у реалізації робить їх привабливими для використання в умовах обмеженого простору. Додатково до цих методів, можна використовувати технології, такі як LIDAR. LIDAR використовує лазерні імпульси для вимірювання відстані до об'єктів, забезпечуючи точність і надійність. У процесі роботи LIDAR пропускає лазерний пучок, який відбивається від об'єкта та повертається до сенсора, дозволяючи точно обчислити відстань на основі часу, що минув [17].

Основні переваги LIDAR полягають у високій точності вимірювань та можливості створення детальних 3D-карт навколишнього середовища. Ця технологія є особливо корисною для картографування, навігації та контролю за дронами. Вона також може комбінуватися з камерами для покращення точності та надійності вимірювань, використовуючи сильні сторони кожного методу. Системи, що поєднують LIDAR і камери, можуть досягати відмінних результатів, забезпечуючи більш детальне та точне відстеження об'єктів у складних умовах. Додаткові типи датчиків, такі як ультразвукові, радіолокаційні та інфрачервоні, також можуть бути використані для вимірювання відстані до об'єктів. Наприклад, ультразвукові датчики вимірюють відстань за допомогою ультразвукових імпульсів, хоча вони зазвичай менш точні.

Таким чином, використання стереозору, монокулярних методів, LIDAR та інших сенсорів представляє різноманітні підходи до вимірювання дистанції до об'єктів, таких як дрони. Кожен з цих методів має свої переваги та недоліки, тому вибір конкретного підходу залежить від вимог до точності, швидкості обробки та умов експлуатації. Поєднання різних технологій може забезпечити оптимальні результати в реальних сценаріях, де точність вимірювань є критично важливою. Цей комплексний підхід до вимірювання дистанції дозволяє значно покращити ефективність системи відстеження та підвищити її надійність.

2.4 Функції втрат для задачі виявлення об’єктів

Функції втрат є важливою складовою частиною навчання нейронних мереж для задач детекції об'єктів. Вони використовуються для вимірювання відмінності між прогнозованими значеннями та істинними мітками, а також для коригування ваг мережі з метою покращення точності прогнозів.

Однією з основних функцій втрат є Box Loss, яка відповідає за оцінку точності локалізації об'єктів у зображенні. Вона визначається на основі метрики IoU, яка вимірює перетин між передбаченими та істинними обмежувальними прямокутниками. Чим вищий IoU, тим точніша локалізація. Крім класичного IoU, для покращення точності локалізації використовуються інші метрики, такі як GIoU (Generalized IoU), DIoU (Distance IoU) та CIoU (Complete IoU), які враховують не лише перетин, але й відстань між центрами об'єктів або їхню форму [18]. Ці метрики дозволяють моделі точніше визначати координати об'єктів у кадрі.

Classification Loss — це функція втрат, яка відповідає за класифікацію об'єктів у межах виявлених прямокутників. У задачах класифікації зазвичай

використовується перехресна ентропія, яка вимірює відстань між передбаченими ймовірностями для кожного класу та істинними мітками.

$$Classification\ loss = -\sum_{i} y_i * \log(p_i),$$

де $y_i -$ істинна мітка;

$p_i -$ прогнозована ймовірність для кожного класу.

У випадках, коли в даних спостерігається значний дисбаланс класів, для покращення точності класифікації використовується Focal Loss [19]. Ця функція дозволяє знизити вплив легко класифікованих прикладів і зосереджує увагу на складніших випадках, що значно покращує ефективність класифікації, особливо в умовах, де класів багато і їх розподіл нерівномірний.

$$Focal\ Loss\ = -\alpha(1 - p_t)^{\gamma} \log(p_t),$$

де $\alpha, \gamma -$ параметри інтенсивності фокусування.

Таким чином, classification loss забезпечує правильне визначення типів об'єктів, що критично важливо для моделей, які мають справу з великою кількістю різних класів об'єктів, таких як різні види дронів або інших швидко рухомих цілей.

DFL Loss (Distribution Focal Loss) являє собою функцію втрат, що використовується для вдосконалення регресії координат об'єктів. Замість того щоб прогнозувати одне значення для координат, DFL [20] дозволяє моделі виводити розподіл ймовірностей для кожної координати. Такий підхід допомагає моделі краще враховувати невизначеність у прогнозах і покращує точність при локалізації об'єктів, особливо в складних ситуаціях, де об'єкти можуть мати різні форми, швидко рухатися або бути частково перекритими іншими об'єктами. Використання ймовірнісної регресії дозволяє моделі більш точно відображати позиції об'єктів, оскільки вона працює з повною картиною

можливих варіантів, а не з єдиним передбаченням. Це робить DFL особливо корисною для детекції об'єктів, таких як дрони, які часто швидко змінюють своє положення та можуть мати складні форми.

$$DFL\ Loss = -\sum_{i} p_i * \log(q_i),$$

де $p_i -$ істинні значення;

$q_i -$ прогнозовані значення координат.

У загальному, функції втрат Box Loss, Cls Loss і DFL Loss є основою для побудови точних моделей детекції та трекінгу об'єктів. Вони працюють у тісній взаємодії для покращення локалізації, класифікації та регресії координат об'єктів, що дозволяє нейронним мережам ефективно виконувати складні завдання, такі як відстеження швидко рухомих об'єктів, включаючи дронів [21]. Кожна з цих функцій має свою специфіку і важливість, що робить їх незамінними для створення потужних і точних систем для реального використання.

2.5 Метрики оцінювання якості моделі детекції

Оцінювання продуктивності моделей комп’ютерного зору є важливим для визначення їхньої точності, ефективності та надійності у реальних застосуваннях. Різні метрики надають уявлення про окремі аспекти роботи моделі залежно від того, чи є пріоритетом точне визначення місцезнаходження об’єктів, повне їх виявлення або мінімізація помилок. Основними метриками є середня точність на різних рівнях, перетин на об'єднання, точність, повнота та F1-міра [22], кожна з яких призначена для оцінювання моделей у різних умовах.

Середня точність (mAP) є комплексною метрикою, яка розраховує середню точність для різних класів об’єктів на різних рівнях IoU. Поєднуючи показники точності та повноти, mAP забезпечує загальну оцінку того, наскільки добре модель виявляє об’єкти та розрізняє категорії. Ця метрика дозволяє надійно порівнювати продуктивність моделей у складних умовах. Оцінювання продуктивності на різних порогах робить mAP цінним інструментом для аналізу якості детекції та її стабільності.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Перетин на об'єднання (IoU) вимірює ступінь перекриття між прогнозованими та реальними рамками об’єктів. IoU [23] є критично важливою для завдань, які потребують точного визначення місцезнаходження об’єктів, таких як автономні транспортні засоби чи дрони, де навіть незначна помилка у позиціонуванні може призвести до серйозних наслідків. Високий показник IoU свідчить про кращу відповідність прогнозованих і реальних об’єктів, що відображає просторову точність моделі. Зазвичай порогом успішного визначення вважається IoU ≥ 0,5, хоча для більш суворих застосувань можуть вимагатися вищі значення.

$$IoU = \frac{|Ground\ truth\ bounding\ box \cap Predicted\ bounding\ box|}{|Ground\ truth\ bounding\ box \cup Predicted\ bounding\ box|}$$

Точність зосереджена на мінімізації помилкових спрацювань, оцінюючи частку правильно виявлених об’єктів серед усіх прогнозів. Ця метрика є надзвичайно важливою у сферах, де помилкові тривоги можуть мати серйозні наслідки, наприклад, у системах безпеки або медичній діагностиці. Висока точність забезпечує, що модель виявляє об’єкти лише за високої впевненості у прогнозі, підвищуючи надійність у критично важливих середовищах.

$$Precision = \frac{TP}{TP + FP}$$

Натомість повнота оцінює здатність моделі виявляти всі відповідні об’єкти, наголошуючи на їхньому повному охопленні. Ця метрика є особливо важливою у завданнях, де пропуск хоча б одного об’єкта може призвести до значних наслідків, наприклад, у пошуково-рятувальних операціях або медичному скринінгу [24]. Висока повнота гарантує, що модель виявляє всі наявні об’єкти, навіть якщо це іноді призводить до зайвих спрацювань.

$$Precision = \frac{TP}{TP + FN}$$

F1-міра балансує між точністю та повнотою, розраховуючи їх середнє гармонічне. Ця метрика є корисною, коли потрібно враховувати як помилкові позитивні, так і негативні спрацювання. F1-міра забезпечує єдине значення, що відображає компроміс між цими двома аспектами, забезпечуючи збалансовану оцінку у сценаріях, де важлива як точність, так і повнота, наприклад, у відеоспостереженні або автономному водінні.

$$F_1 = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Кожна метрика має своє значення для оцінювання моделей комп’ютерного зору. mAP забезпечує загальну оцінку продуктивності, IoU фокусується на точності локалізації, точність мінімізує помилкові визначення, повнота забезпечує повне виявлення об’єктів, а F1-міра надає збалансовану оцінку [25]. Правильний вибір метрики відповідно до вимог завдання дозволяє ефективно оцінити та оптимізувати моделі для їхнього практичного застосування.

2.6 Висновки до розділу 2

Цей розділ було присвячено огляду сучасних технологій виявлення та відстеження швидкісних об'єктів, узагальнює результати аналізу ключових алгоритмів і методів, що застосовуються у цій сфері. Спочатку було детально розглянуто алгоритми виявлення об'єктів, зокрема Zero-Shot та Two-Shot детектори, що характеризуються різними підходами до попередньої обробки даних та здатністю визначати невідомі об'єкти. Порівняння Zero-Shot та Two-Shot детекторів дало змогу оцінити їхні переваги та недоліки, що є корисним при виборі методів для специфічних завдань виявлення швидкісних об’єктів.

Далі були розглянуті основні алгоритми відстеження, такі як фільтр Калмана, CSRT, KCF, DeepSORT, та ByteTrack. Кожен із цих алгоритмів має унікальні особливості та обмеження, що дозволяє ефективно застосовувати їх у різних умовах, наприклад, у сценах з високою щільністю об'єктів або при потребі високої точності. Особливий акцент був зроблений на алгоритмах DeepSORT та ByteTrack, які застосовують методи глибокого навчання для трекінгу й демонструють стабільність та точність при роботі зі швидкісними об'єктами.

Окремо були розглянуті методи вимірювання відстані та швидкості за допомогою стереозору та монокулярних методів. Застосування стереозору дозволяє забезпечити точніші дані для виявлення та відстеження, а монокулярні методи є економічнішими, хоча й мають певні обмеження в точності.

Таким чином, проведений аналіз сучасних технологій та алгоритмів показав, що комбінування різних підходів і методів може значно підвищити ефективність виявлення та відстеження швидкісних об'єктів. Отримані результати можуть бути корисними для подальшого вдосконалення методів і вибору оптимальних алгоритмів для специфічних завдань виявлення та відстеження швидкісних об’єктів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Збір даних

Одними з актуальних швидкісних рухомих цілей на даний момент є дрони. Для цього слід виконати пошук даних у відкритому доступі. Пошук даних слід поділити на дві частини, спочатку треба знайти релевантні для нашої задачі фотографії дрона для того, щоб розрізняти їх сутності на відеосигналі. Другий етап складається з пошуку наповнення даних, для того, щоб інші схожі на дрон сутності не сприймалися як ціль. Для датасету з сутностями дрона було взято відкритий датасет з платформи Kaggle (рис.3.1) та додані мануально зібрані дані з польотів дрона (рис.3.2). У якості додавання різних сутностей також було вирішено взяти певні дані з Kaggle та також власноруч додано інші зображення для покращення якості навчання (рис.3.3-3.4). Обрана ціль на дальній дистанції може бути сплутана з іншими речами, тож слід їх додати самостійно.

Рисунок 3.1 — Приклад даних дронів з набору Kaggle

Рисунок 3.2 — Приклад даних дронів зібраних власноруч

Рисунок 3.3 — Приклад додаткових даних зібраних власноруч

Рисунок 3.4 — Приклад додаткових даних зібраних власноруч

3.2 Форматування вхідних даних

Після збору даних слід їх правильно згрупувати для навчання моделі. Модель YOLO, що ми використовуємо потребує спеціального форматування розташування тренувальних даних. Для початку слід для усіх нерозмічених додаткових даних створити для них правильні мітки (рис.3.5-рис.3.6). Структура файлів-міток для додаткових даних досить прості, вони мають бути просто пустими.

Після створення міток для додаткових даних, наступним етапом є розмітка додатково зібраних зображень. Для цієї задачі використовуємо LabelImg та підключаємо її за допомогою conda оточення (рис.3.7).

```
Created /content/added_labels/no-drone_20.txt
Created /content/added_labels/no-drone_12.txt
Created /content/added_labels/no-drone_19.txt
Created /content/added_labels/no-drone_63.txt
Created /content/added_labels/no-drone_92.txt
Created /content/added_labels/no-drone_27.txt
Created /content/added_labels/no-drone_75.txt
Created /content/added_labels/no-drone_64.txt
Created /content/added_labels/no-drone_91.txt
Created /content/added_labels/no-drone_79.txt
Created /content/added_labels/no-drone_7.txt
Created /content/added_labels/no-drone_41.txt
Created /content/added_labels/no-drone_18.txt
Created /content/added_labels/no-drone_33.txt
Created /content/added_labels/no-drone_78.txt
Created /content/added_labels/no-drone_5.txt
Created /content/added_labels/no-drone_45.txt
Created /content/added_labels/no-drone_73.txt
Created /content/added_labels/no-drone_35.txt
```

Рисунок 3.5 — Фрагмент додавання міток для додаткових даних зібраних власноруч

```
Created /content/added_labels/22101.txt
Created /content/added_labels/23972.txt
Created /content/added_labels/20687.txt
Created /content/added_labels/23487.txt
Created /content/added_labels/23608.txt
Created /content/added_labels/21036.txt
Created /content/added_labels/20266.txt
Created /content/added_labels/22079.txt
Created /content/added_labels/20759.txt
Created /content/added_labels/21736.txt
Created /content/added_labels/20405.txt
Created /content/added_labels/22730.txt
Created /content/added_labels/21497.txt
Created /content/added_labels/20882.txt
Created /content/added_labels/21816.txt
Created /content/added_labels/24313.txt
Created /content/added_labels/20317.txt
Created /content/added_labels/22974.txt
Created /content/added_labels/21415.txt
```

Рисунок 3.6 — Фрагмент створення міток для додаткових даних з датасету

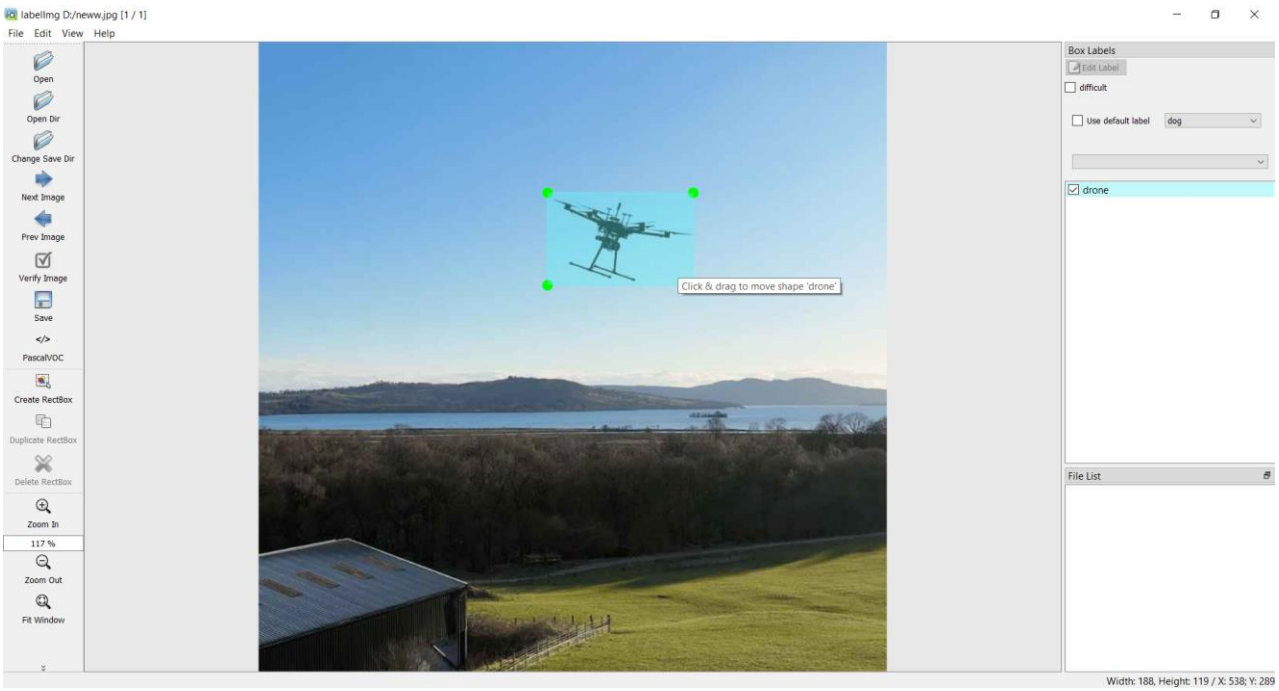


Рисунок 3.7 — Приклад розмітки даних зображень дронів зібраних власноруч

Далі виконуємо розподілення зібраних файлів з мітками та зображеннями тренування моделі (рис.3.8).

```
Copying files to Data/train/images
100% |████████████████████| 3574/3574 [00:01<00:00, 1982.88it/s]
Copying files to Data/train/labels
100% |████████████████████| 3574/3574 [00:00<00:00, 7260.04it/s]
Copying files to Data/test/images
100% |████████████████████| 179/179 [00:00<00:00, 2382.66it/s]
Copying files to Data/test/labels
100% |████████████████████| 179/179 [00:00<00:00, 3184.59it/s]
Copying files to Data/val/images
100% |████████████████████| 715/715 [00:00<00:00, 3269.60it/s]
Copying files to Data/val/labels
100% |████████████████████| 715/715 [00:00<00:00, 6128.50it/s]
```

Рисунок 3.8 — Розподілення файлів по директоріям

Також для правильного тренування YOLO моделі створюємо yaml-файл з визначеною структурою (рис. 3.9). Слід вказати відповідні шляхи відповідно до директорій з тренувальними даними, валідаційними даними та тестового

набору. Після цього зазначаємо розмічені класи, в нашому випадку це один клас, який ми і плануємо відслідковувати.

```
train: /content/Data/train/images
test: /content/Data/test/images
val: /content/Data/val/images

names:
  0: drone
```

Рисунок 3.9 — Структура yaml-файлу

3.3 Тренування моделі

Після того, як все сформовано ми можемо розпочати тренування моделі. Для тренування візьмемо уже готові ваги моделі версії YoloV8n. Також слід правильно підібрати параметри для тренування. Кількість епох має бути значною, також слід встановити параметр single_cls на True, оскільки навчання відбувається лише для розпізнавання одного класу. Через значні об’єми даних та велику складність архітектури моделі для навчання слід підключити графічний обчислювальний юніт. Наводимо фрагмент тренування на рисунку 3.10.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
10/30      11.4G      1.628       1.66      1.714          2        640: 100%|██████████| 224/224 [02:57<00:00,  1.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:13<00:00,  1.68it/s]

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
11/30      11.3G      1.587      1.637      1.679          4        640: 100%|██████████| 224/224 [02:57<00:00,  1.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:12<00:00,  1.81it/s]

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
12/30      11.4G      1.592      1.581      1.703          4        640: 100%|██████████| 224/224 [02:58<00:00,  1.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:12<00:00,  1.85it/s]

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
13/30      11.4G      1.579      1.522      1.669          3        640: 100%|██████████| 224/224 [02:58<00:00,  1.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:12<00:00,  1.81it/s]

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
14/30      11.3G      1.515      1.467      1.624          6        640: 100%|██████████| 224/224 [02:57<00:00,  1.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:12<00:00,  1.85it/s]
```

Рисунок 3.10 — Фрагмент тренування моделі

Перевіримо результат тренування на декількох зображеннях (рис. 3.11-3.13).

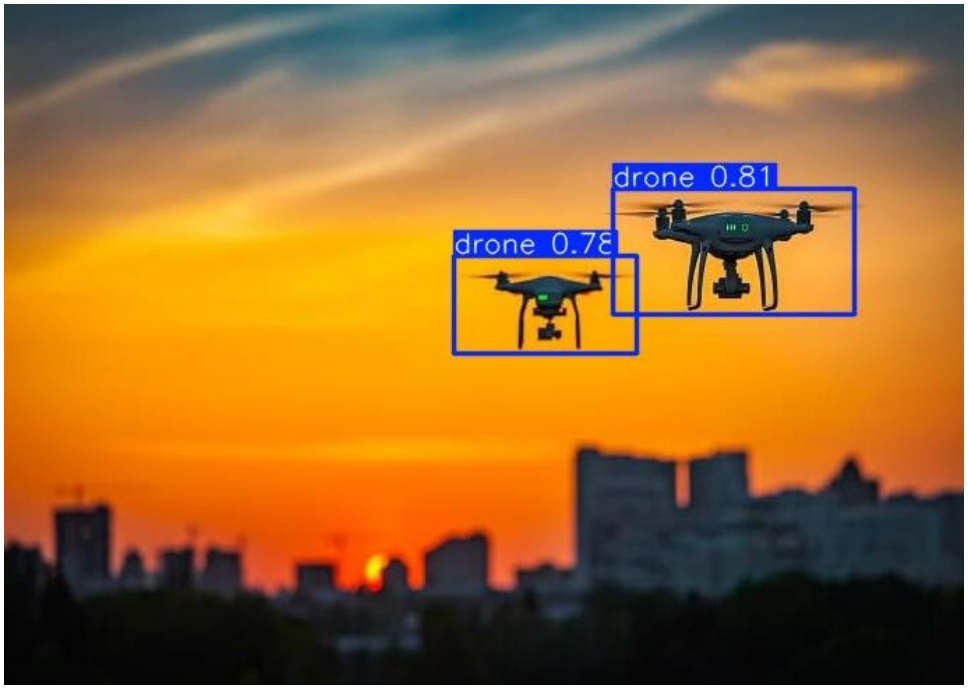


Рисунок 3.11 — Результат роботи моделі на першому зображенні

Рисунок 3.12 — Результат роботи моделі на другому зображенні

Рисунок 3.13 — Результат роботи моделі на третьому зображенні

Виведемо порівняльну характеристику роботи моделі на валідаційних даних на рисунках 3.14-3.17.

Рисунок 3.14 — Розмічені дані №1

Рисунок 3.15 — Передбачені дані №1

Модель досить непогано впоралася із класифікацією дальніх зображень, однак помилилася і не знайшла ціль, коли вона занадто близько.

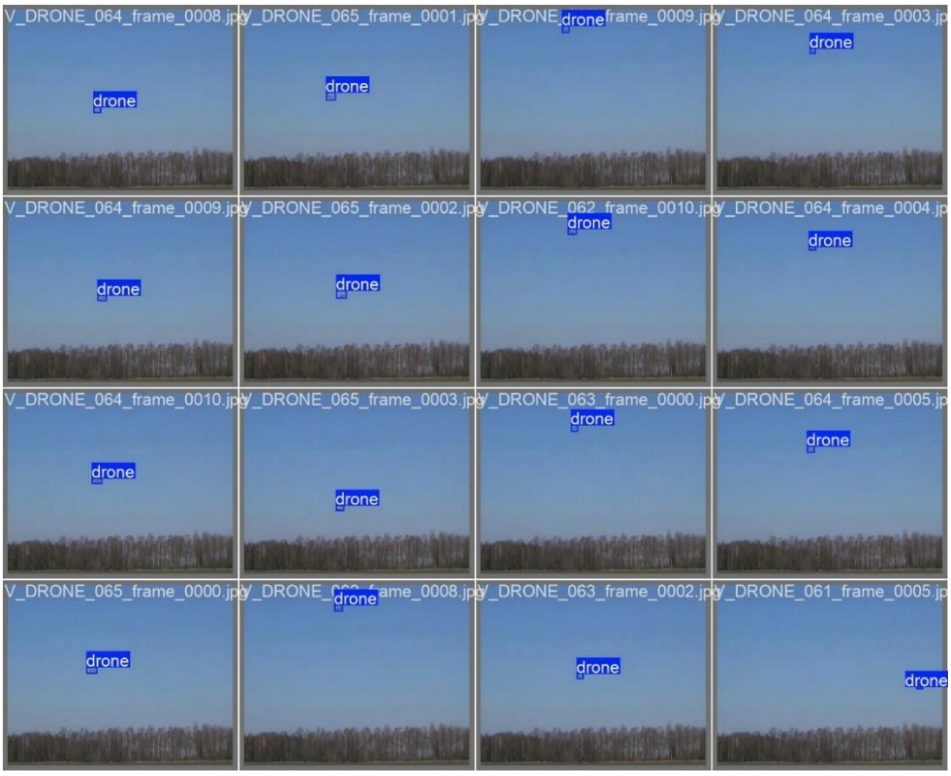


Рисунок 3.16 — Розмічені дані №2

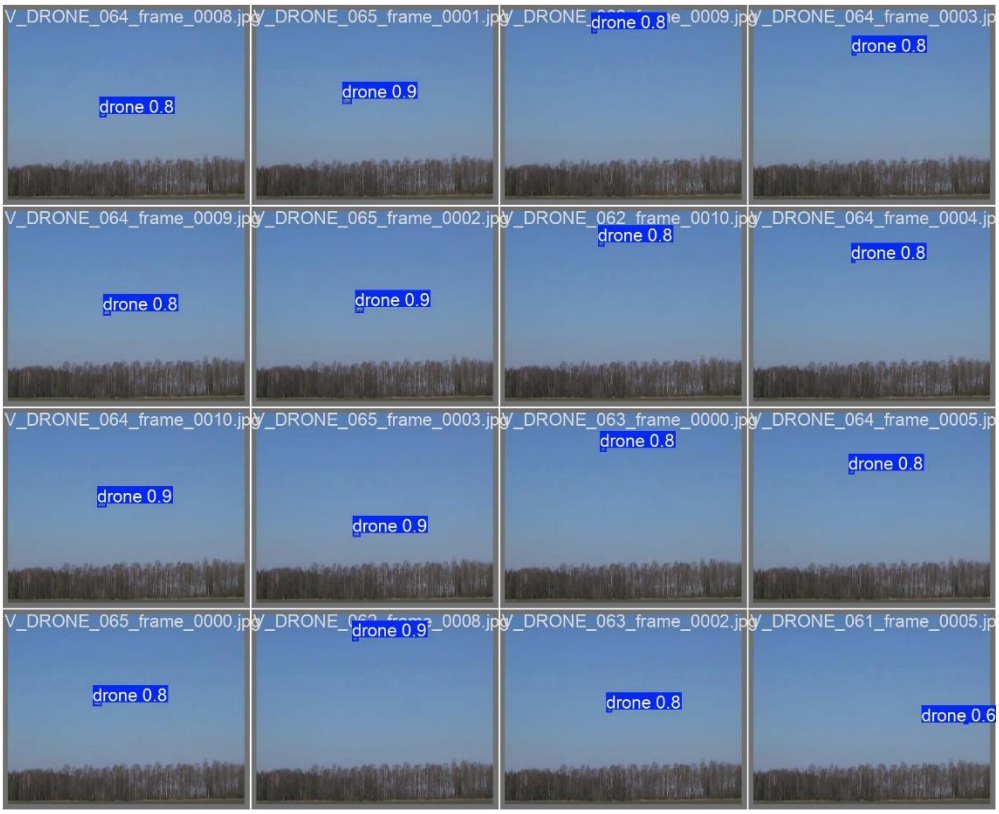


Рисунок 3.17 — Передбачені дані №2

На другому наборі для перевірки ми можемо побачити, що модель досить добре знаходить ціль, яка знаходиться від нас досить далеко та показує високу впевненість у цих розпізнаваннях.

Далі наведемо функції втрат моделі, отримані з тренування (рис. 3.18).

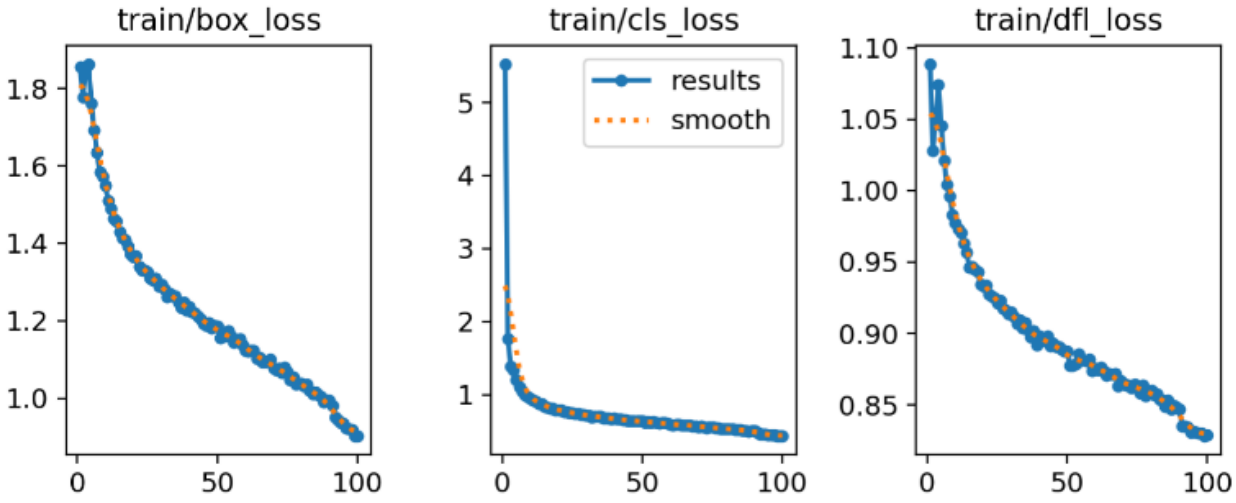


Рисунок 3.18 — Функції втрат

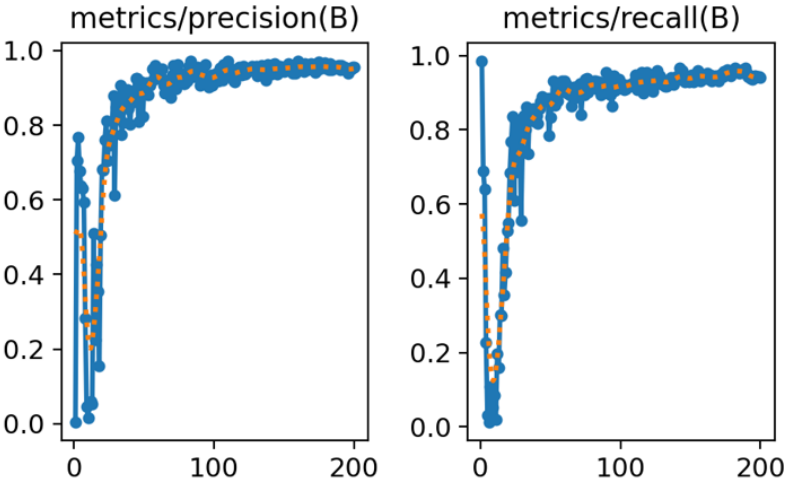


Рисунок 3.19 — Метрики оцінки якості моделі №1

З отриманих графіків на рисунках 3.19-3.20 ми бачимо, що модель демонструє стійке зростання метрик Precision і Recall у процесі тренування. На графіку Precision спостерігається поступове підвищення до рівня 0.96, що свідчить про здатність моделі мінімізувати кількість хибних позитивних спрацьовувань, тобто неправильно передбачених об’єктів. Графік Recall показує, що модель досягає рівня 0.92, що вказує на успішне виявлення близько 92% об’єктів, які фактично є в даних. Помітні коливання значень на ранніх ітераціях пов’язані зі стадією адаптації моделі до даних, проте ближче до завершення тренування метрики стабілізуються, що свідчить про хорошу узгодженість результатів. Такий прогрес підтверджує ефективність тренування на заданому наборі даних.

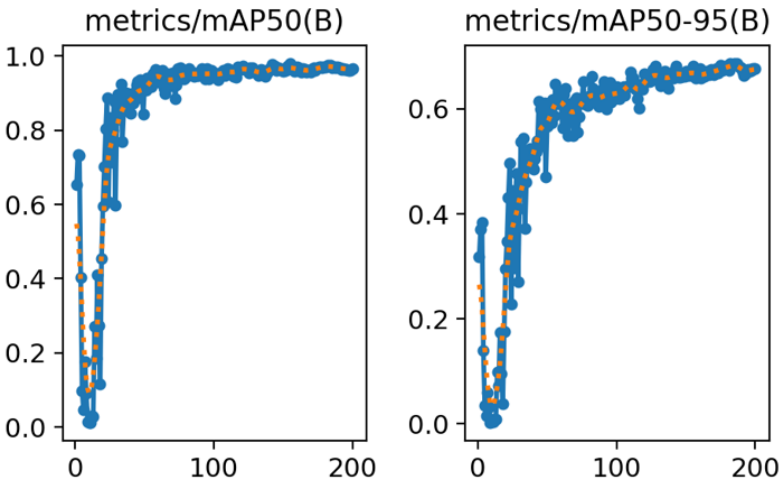


Рисунок 3.20 — Метрики оцінки якості моделі №2

З отриманих графіків видно, що модель демонструє стабільне покращення якості роботи за метриками середньої точності (протягом тренування. Графік mAP50 відображає зростання з початкових значень близько до 0.97 наприкінці навчання. Ця метрика свідчить про те, що модель ефективно ідентифікує об'єкти за умов помірного перекриття між передбаченими й реальними об'єктами (IoU = 0.5). Коливання на ранніх етапах пояснюються адаптацією моделі до навчальних даних, але зі збільшенням кількості епох тренування метрика стабілізується, демонструючи узгодженість передбачень.

Графік mAP50:95, який є більш суворою оцінкою (враховує діапазон перекриття 0.5 < IoU < 0.95), показує зростання до рівня 0.65. Хоча значення цієї метрики нижчі, її стабільний ріст свідчить про те, що модель поступово стає здатною точно ідентифікувати об'єкти навіть у складних умовах. Такий прогрес демонструє потенціал моделі для вирішення задач із високими вимогами до точності, а її стабільність наприкінці тренування вказує на добре виконане навчання. Наведемо PR-криву на рисунку 3.21.

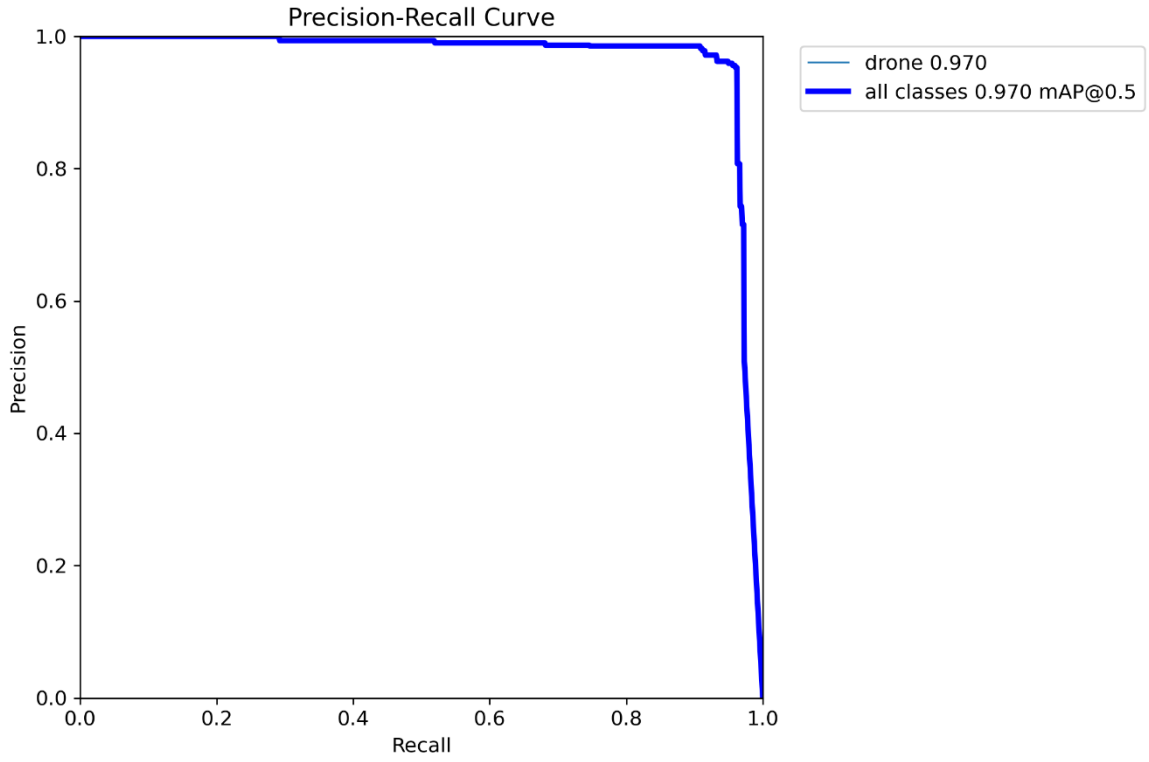


Рисунок 3.21 — Крива Precision-Recall для класу, який досліджуємо

Також для задачі досить важливо розрахувати швидкість обробки, щоб визначити показник кількості оброблених кадрів в секунду. Наведемо отримані дані про швидкість обробки одного кадру в таблиці 3.1.

Таблиця 3.1 — Швидкість обробки кадру

Процес	Час(мс)
‘preprocess’	2.56
‘inference’	12.18
‘postprocess’	23.39

Розрахуємо кількість кадрів в секунду:

$$FPS = \frac{1000}{2.56 + 12.18 + 23.39} = 26.22 \frac{\text{кадр}}{\text{с}}$$

3.4 Визначення дистанції до об’єкту

Для визначення відстані до об’єкту, що відстежуємо застосуємо алгоритм стереоскопічного зору(рис.3.13). Сутність полягає у тому, щоб за рахунок відомої відстані та відстані виявлених об’єктів на зображеннях знайти реальну відстань від об’єкту до камер.

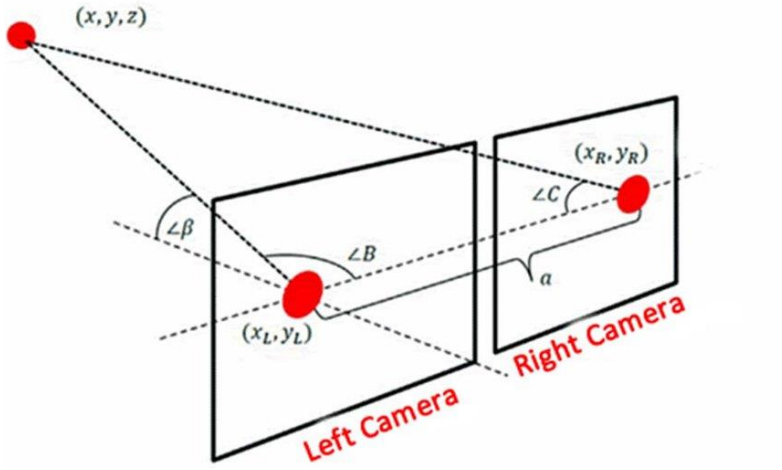


Рисунок 3.22 — Схема стереоскопічного зору

Для розрахунку слід спочатку визначити модуль різниці координат зображення, виміряти відстань між камерами та підібрати правильний коефіцієнт, щоб замінити фокусну відстань. Результуюча формула матиме такий вигляд:

$$Z = \frac{B * f}{d},$$

де $B -$ відстань між камерами;

$f -$ фокусна відстань;

$d -$ різниця координат об′єкту на зображенні.

3.5 Додавання трекеру

Для оптимізації швидкості роботи моделі, що є дуже важливим для слідкування в реальному часі, виконаємо інтегрування трекеру. Ідея полягає у тому, щоб звертатися до моделі с певною частотою, а не кожного кадру. Тому для слідкування за захопленим об’єктом дуже доречно було б застосувати алгоритм трекінгу.

Рисунок 3.23 — Результат роботи CSRT

На відео вище наведено результат роботи алгоритму CSRT (Channel and Spatial Reliability Tracker) на тренувальному відео. Як ми бачимо, при досить різкій зміні фону алгоритм легко може втратити об’єкт при різкій зміні фону, отже звертатися до моделі слід не досить часто, однак і не досить рідко, оскільки самостійно трекер не є досить надійним.

4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

Зараз у сучасному світі існує велика потреба в системах здатних збільшувати роздільну здатність зображеннь. Взагалі є багато систем які використовують класичні методи такі як інтерполяція пікселів зображення, проте практично немає небагато систем, якість яких би задовольняла домінуючу більшість користувачів та дослідників. Це пов’язано з тим, що ця задача збільшення роздільної здатності з одного зображення вимагає складних алгоритмів, великих обчислювальних ресурсів, а також багато даних для навчання та тестування. Тим не менш, питання супер роздільної здатності одного зображення цікавить багато технічних компаній, що виготовляють цифрові фото і відеокамери та компанії що працюють з цифровими зображеннями у сфері інформаційних технологій, мобільної фотографії, безпеки, медицини тощо.

Все це означає, що проблема, яку вирішує стартап, являється актуальною і може стати успішним на ринку та стати основним вибором для переважної кількості цільової аудиторії.

4.1 План розробки стартапу та масштабування його на ринок

Наведемо план розробки стартапу та виведення його на ринок.

Спочатку треба провести маркетинговий аналіз, який включає в себе:

- конкурентний аналіз, щоб зрозуміти, якими методами вирішення проблем вже користуються люди;
- формування ідеї самого проекту та виділення цільової аудиторії;
- розробити стратегію виведення товару на ринок, базуючись на аналізі ринкового середовища.

Наступним кроком являється організація самого стартапу. На цьому етапі мають бути:

- складений весь план та побудований таймлайн розробки та запуску продукту;
- запланований обсяг виробництва та оцінений потенційний обсяг ресурсу, який буде потрібен для виконання плану;
- розраховані витрати, необхідні для реалізації проекту, та витрати на запуск проекту.

Далі необхідно виконати фінансово-економічний аналіз та оцінити ризики стартап-проекту, в межах якого:

- визначити обсяг інвестиційних втрат;
- розрахувати основні фінансово-економічні показники проекту (собівартість, ціну продукту/послуги, податковий збір та чистий прибуток) та визначити показники інвестиційної привабливості проекту (рентабельність продажів, період окупності проекту);
- визначити основні ризики проекту та способи для їх запобігання.

Фінальним кроком являється розробка заходів з комерціалізації продукту. Цей крок являється важливим для масштабування та збільшення розмірів продукту.

Для того, щоб залучити інвесторів та знайти різні способи фінансування проекту, необхідно:

- провести дослідження на предмет інтересів потенційних інвесторів та бізнесів;
- скласти інвестиційну пропозицію, яка включає в себе як опис самого продукту та його теперішні розміри, так і можливі шляхи розширення та розвитку;
- обрати канали комунікації із потенційно зацікавленими персонами.

Далі наведемо результати виконання кожного з описаних кроків.

4.2 Опис ідеї стартап-проекту

Стартап-проект спрямований на вирішення проблеми виявлення та відстеження швидкісних рухомих об'єктів, зокрема дронів. Основна суть продукту полягає у створенні системи, яка в реальному часі зможе автоматично виявляти та відстежувати дрони, а також вимірювати відстань до них. Ця система може застосовуватися в сфері безпеки, управління повітряним простором та рятувальних операціях.

У таблиці 4.1 наведена інформаційна карта стартап-проекту.

Таблиця 4.1 — Інформаційна карта стартап-проекту

Назва проекту	ViCap
Автори проекту	Бєздєтний Даніїл Дмитрович
Коротка анотація	Система буде автоматично виявляти та відстежувати дрони, вимірювати до них дистанцію, забезпечуючи ефективний моніторинг рухомих об'єктів у реальному часі.
Термін реалізації проекту	12 місяців
Необхідні ресурси	Приміщення з обладнанням (комп'ютери з GPU, доступ до Інтернету, датчики або камери для тестування), програмне забезпечення для розробки (TensorFlow, OpenCV), хмарне середовище для зберігання даних.

Продовження таблиці 4.1

Опис проблеми, яку вирішує проект	Проблема автоматичного виявлення, відстеження та вимірювання відстані до швидкісних рухомих об'єктів, таких як дрони.
Головні цілі та завдання проекту	Створення системи для автоматичного виявлення та трекінгу дронів із можливістю вимірювання дистанції, яка працює в реальному часі і може бути використана в різних сферах (безпека, моніторинг).
Очікувані результати	Привернення уваги технологічних компаній, що працюють з дронами, створення автономної системи, яка стане базовим рішенням для відстеження швидкісних об'єктів у реальних умовах.

4.3. Технологічний аудит ідеї проекту

Тепер можна розібрати ідею стартапу та провести конкурентний аналіз. У таблиці 4.2 наведений опис ідеї стартапу.

Таблиця 4.2 – Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основна ідея полягає у створенні комплексної системи для виявлення та трекінгу швидкісних рухомих об'єктів, таких як дрони, з можливістю вимірювання дистанції до них. Система збору даних та зворотного зв'язку для подальшого покращення точності трекінгу та адаптації алгоритмів.	Безпека та моніторинг повітряного простору, контроль дронів у логістиці та рятувальних операціях.	Автоматизоване відстеження дронів у реальному часі з високою точністю і можливістю вимірювання відстані.
	Постійне вдосконалення системи на основі зібраних даних. Персоналізація під потреби різних користувачів.	Покращення продуктивності та адаптивність під конкретні умови використання.

Далі проведемо порівняльний аналіз конкурентів проекту та наведемо результати у таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз конкурентів проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		DJI AirSense	**Amazon Rekogniti on**	Skydio	Свій власний алгоритм			
1	Точність виявлення та трекінгу дронів	Середня, залежить від ADS-B сигналу	Середня, не оптимізова но для дронів	Висока, завдяки AI	Висока, адаптивна			+
2	Доступність по ціні	Висока ціна	Передплат а, Pixel потрібен	Дорого	Гнучко, залежить від реалізації		+	
3	Можливість персоналізац ії	Обмежена екосистем ою DJI	Мінімальн а через API	Середня	Повна адаптація			+

Далі аналізуємо реальність технічно здійснити ідею проекту (таблиця 4.4).

Таблиця 4.4 – Технологічна здійсненність продукту

№ п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
1	Створення комплексної системи для виявлення та відстеження швидкісних рухомих об'єктів (дронів) з вимірюванням відстані до них	Використання мови програмування Python з бібліотеками для комп'ютерного зору (OpenCV, YOLO)	Наявні	Доступні
2		Використання технології DeepSORT для трекінгу об'єктів	Наявні	Доступні
3		Використання датчиків або камер для точного вимірювання відстані	Наявні, необхідні налаштування	Доступні
Обрана технологія реалізації проекту: Python із використанням бібліотек для комп'ютерного зору (Ultralytics, OpenCV), DeepSORT для трекінгу та сенсорів для вимірювання відстані.				

4.4. Аналіз ринкових можливостей запуску стартап-проекту

Далі проведемо попередній аналіз ринку для запуску стартап-проекту (таблиця 4.5).

Таблиця 4.5 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	6000
3	Динаміка ринку (якісна оцінка)	Позитивна, зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Мінімальні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	15%

Тепер проведемо характеристику потенційних клієнтів, які можуть бути зацікавлені в проекті (таблиця 4.6).

Тепер проведемо характеристику потенційних клієнтів, які можуть бути зацікавлені у використанні системи трекінгу та моніторингу дронів (таблиця 4.6).

Таблиця 4.6 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреби, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Відстеження та моніторинг швидкісних об'єктів (дронів)	Особисте використання користувачем	Цікавить загальна точність та простота використання	Простота налаштування та використання
2	Захист повітряного простору	Державні служби, правоохоронні органи, армія	Особлива потреба в точності трекінгу та швидкій реакції	Висока точність трекінгу та стабільність роботи
3	Інспекція та контроль дронів у логістиці	Компанії з доставки дронів	Необхідність в автоматизації процесу та надійності системи	Надійність і підтримка великих обсягів даних

Обрахуємо фактори загроз (таблиця 4.7) та можливостей (таблиця 4.8). Проаналізуємо загрози, щоб зрозуміти можливі перешкоди при запуску продукт на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 4.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Ринок систем для відстеження швидкісних об'єктів, зокрема дронів, вже має кілька великих гравців, які займають ключові позиції та мають власні цільові групи	Виявити нові можливості та сфери застосування, щоб створити унікальну цінність для користувачів
2	Ціна збуту	Конкуренти можуть пропонувати продукти за нижчою ціною, оскільки їхні рішення можуть бути менш точними або націлені на масовий ринок	Зосередитися на високій якості трекінгу та вимірювання відстані, а також розробити ефективну маркетингову стратегію
3	Технологічні виклики	Через складність задачі відстеження швидкісних об'єктів, система може мати труднощі з точністю в різних умовах, наприклад під час поганої видимості або швидких маневрів дронів	Оптимізувати модель для різних умов, розробити адаптивні алгоритми та постійно вдосконалювати технологію

Після аналізу загроз, слід також розглянути фактори можливостей (таблиця 4.8), щоб скористатися сприятливими ринковими умовами.

Таблиця 4.8 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Універсальність	Продукт для відстеження дронів не залежить від конкретної апаратної платформи, як це буває у більшості конкурентів	Підкреслити це в маркетинговій стратегії, просуваючи продукт як універсальне рішення для різних платформ
2	Простота використання	Від користувача вимагається лише надати відео чи інші дані для аналізу та трекінгу	Створити зручний інтерфейс для легкого завантаження даних і налаштувань системи
3	Якість і надійність	Надання високоякісних послуг з точного відстеження та вимірювання дистанції до дронів	Пропонувати системи з найкращими алгоритмами для трекінгу і надавати технічну підтримку користувачам
4	Безкоштовний сервіс на етапі MVP	Швидко залучити клієнтів та заявити про себе на ринку, надаючи безкоштовний доступ на етапі MVP	Провести активну маркетингову кампанію, залучати клієнтів конкурентів, використовуючи безкоштовну пропозицію

Далі розглянемо питання конкуренції, а саме визначимо її тип та рівень (таблиця 4.9).

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: недосконала конкуренція	На ринку представлено обмежену кількість продуктів та експертів з трекінгу дронів	Зосередитися на швидкому просуванні продукту, залучити максимальну кількість клієнтів на початкових етапах
2. Рівень конкурентної боротьби: міжнародний	Існують проекти, доступні на глобальному ринку, розроблені компаніями з різних країн	Розширити цільову аудиторію, створити інтерфейси на різних мовах для залучення користувачів з різних регіонів
3. Галузева ознака: внутрішньогалузева	Технології можуть застосовуватися у різних галузях, наприклад, безпека або логістика	Покращувати персоналізацію продукту для різних ринкових сегментів, зосередитися на спеціалізованих рішеннях

Продовження таблиці 4.9

4. Конкуренція за видами товарів: товарно-родова	Конкуренція з іншими системами трекінгу та аналітики дронів	Підтримувати та вдосконалювати наявні функції, забезпечуючи їх відповідність вимогам користувачів
5. Характер конкурентних переваг: нецінова	Різні компанії пропонують продукти з різною якістю та функціоналом	Інвестувати в розробку високоякісних алгоритмів для точного виявлення дронів та вимірювання відстані
6. Інтенсивність: брендингова	На ринку вже є компанії зі значними брендовими перевагами	Створити чітку комунікаційну стратегію для розвитку власного бренду і диференціації від конкурентів

Далі необхідно виконаємо аналіз конкуренції за моделлю 5 сил конкуренції Майкла Портера (таблиця 4.10).

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаровиробники
	Інші існуючі системи трекінгу та вимірювання	Інші існуючі системи трекінгу та вимірювання	Відсутність залежності від постачальників	Контроль якості, порівняння цін	Обмежені або відсутні на ринку

Продовження таблиці 4.10

Висновки	Конкуренція має середню інтенсивність	Можливість входження нових учасників	Постачальники не мають вагомого впливу	Клієнти не диктують умови	Відсутність прямих товарозамінників

Маючи результати аналізу конкуренції (таблиця 4.10), характеристики ідеї стартап-проекту (таблиця 4.5), характеристики потенційних клієнтів і їх вимоги до продукту (таблиця 4.6) та фактори ринкового середовища (таблиці 4.7 і 4.8) було сформульовано та обґрунтовано перелік факторів конкурентоспроможності (таблиця 4.11).

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Універсальність	Продукт не залежить від апаратної платформи, що дає йому перевагу перед більшістю конкурентів.
2	Простота у використанні	Від користувача вимагається лише завантажити файл зображення, що робить продукт зручним та доступним.
3	Якість та гарантії	Пропонуються найбільш якісні послуги та сервіси, що створює конкурентну перевагу за рахунок надійності.

Продовження таблиці 4.11

4	Безкоштовний сервіс при MVP	Стартовий безкоштовний сервіс дозволяє швидко залучити клієнтів та заявити про себе на ринку.

Тепер можна провести аналіз сильних та слабких сторін продукту (таблиця 4.12).

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Універсальність	20	+						
2	Простота у використанні	16		+					
3	Якість та гарантії	10		+					
4	Безкоштовний сервіс при MVP	17			+				

Далі проведемо SWOT-аналіз продукту (таблиця 4.13).

Таблиця 4.13 – SWOT-аналіз стартап-проекту

Сильні сторони Універсальність Простота у використанні Якість та гарантії Безкоштовний сервіс при MVP	Слабкі сторони Відсутність сильного бренду Невелика база клієнтів Не підключені альтернативні канали маркетингу

Продовження таблиці 4.13

Можливості Покращення системи Персоналізація	Проблеми зі збутом Нові системи та конкуренти

Завдяки проведеному SWOT-аналізу, ми змогли визначити сильні та слабкі сторони, можливості та загрози, пов'язані з ринковою конкуренцією та стратегією запуску стартап-проекту. Наступним етапом буде розробка альтернативної ринкової стратегії інтеграції продукту та визначення орієнтовних строків реалізації проекту з урахуванням потенційних конкурентів (таблиця 4.14).

Таблиця 4.14 – Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок з продуктом нижчої якості для початкового тестування	70%	4 місяці
2	Запуск продукту одразу з платним доступом	50%	6 місяців
3	Представлення користувачам системи для відстеження дронів без повноцінного інтерфейсу	60%	5 місяці

У цьому розділі проведено детальний аналіз ринку та продукту для впровадження системи "Захоплення та виявлення швидкісних рухомих цілей". Відповідно до результатів конкурентного аналізу, визначених ринкових факторів та сприятливих умов, можна зробити висновок, що існують

перспективи для успішного виходу продукту на ринок, зокрема для застосування в відстеженні дронів і вимірюванні дистанції до них.

4.5. Розроблення ринкової стратегії стартап-проекту

Для розробки ринкової стратегії продукту "Захоплення та виявлення швидкісних рухомих цілей", спершу необхідно проаналізувати цільову аудиторію проекту (таблиця 4.15).

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Персональні користувачі	Висока	10%	Висока	Середня
2	Великі бізнеси	Середня	40%	Середня	Середня
3	Малі та середні бізнеси	Середня	10%	Середня	Середня
4	Держава	Низька	40%	Низька	Висока
Які цільові групи обрано: 1, 3					

Маючи аналіз цільових груп, далі визначимо базову стратегію розвитку продукту (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	1 та 3	Диференційованого маркетингу	Масштабування та максимізація	Оптимальних витрат

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 4.17, 4.18).

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Так	Ні	Виклику лідера

Таблиця 4.18 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
Універсальність Простота у використанні Якість результатів	Оптимальних витрат	Універсальність Простота у використанні Якість та гарантії Безкоштовне використання при MVP	Система, яка краще всіх покращує якість користувацьких зображень Система з простим інтерфейсом

4.6 Розроблення маркетингової програми стартап-проекту

Після проведеного всебічного аналізу можна повноцінно описати ключові переваги концепції запропонованого продукту (таблиця 4.19) та сформувати концепцію маркетингових комунікацій (таблиця 4.20).

Таблиця 4.19 – Ключові переваги концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Точне виявлення	Висока точність і швидкість ідентифікації дронів	Висока точність і швидкість ідентифікації дронів
2	Універсальність	Незалежність від конкретних апаратних платформ	Можливість використання системи будь-яким користувачем
3	Простий інтерфейс	Легке налаштування та управління	Інтуїтивний інтерфейс, що потребує мінімальних дій користувача

Таблиця 4.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пошук спеціалізованих систем	B2B-продажі, теплі контакти, таргетована реклама, спеціалізовані видання	Точність, Якість, Універсальність	Інформувати про високу якість та універсальність	Таргетована реклама на професійну аудиторію

Продовження таблиці 4.20

2	Пошук доступного продукту	Рекламні банери, форуми, реклама від інфлюенсерів	Простота, Безкоштовне тестування MVP	Підвищення довіри до бренду і продукту	Реклама у лідерів думок Вивіски в публічних місцях

4.7 Висновки до розділу 4

У цьому розділі було проведено всебічний аналіз ринку для впровадження стартап-проекту "Захоплення та виявлення швидкісних рухомих цілей" з акцентом на відстеження дронів і вимірювання до них дистанції. Ринкові умови є досить сприятливими, оскільки поки на ринку відсутня значна кількість конкурентних продуктів з подібними функціями. Це створює можливість для виходу стартапу з новітньою системою, що об’єднує виявлення, відстеження та оцінку відстані до швидкісних об'єктів, зокрема дронів, що є актуальним для зростаючого ринку комерційного та приватного моніторингу повітряного простору.

Вибір цільової аудиторії показав, що головними сегментами є персональні користувачі, а також малі та середні бізнеси, що мають попит на системи для моніторингу безпілотних апаратів. Персональні користувачі зацікавлені у доступному та простому рішенні для забезпечення безпеки і контролю в повітрі, а малі та середні підприємства потребують надійних інструментів для моніторингу у комерційних цілях. Під час аналізу цих сегментів було також враховано рівень конкуренції і простоту входу в кожен з

них, що дозволило вибудувати пріоритети для подальшої маркетингової діяльності стартапу.

Завдяки чіткому визначенню конкурентних переваг продукту, серед яких універсальність, простота використання та висока точність, сформовано міцну основу для маркетингової стратегії. У рамках маркетингової програми пропонується використовувати таргетовану рекламу, співпрацю з інфлюенсерами та B2B-продажі, щоб донести інформацію про якість та надійність продукту до цільових сегментів. Також важливо виділити переваги системи над конкурентами, позиціонуючи її як універсальне рішення, що легко інтегрується в різні сфери застосування. Загалом, результати дослідження показали, що стартап має значні шанси на успішний вихід на ринок і закріплення своїх позицій у ключових сегментах.

ВИСНОВКИ

У ході виконання дослідження на тему виявлення та захоплення швидкісних рухомих об'єктів з використанням глибоких нейронних мереж було розроблено ефективну методику для розв'язання складних задач комп'ютерного зору. На основі аналізу сучасних алгоритмів виявлення та трекінгу об'єктів обґрунтовано вибір YOLO моделей та трекерів, таких як DeepSORT і ByteTrack, для досягнення високої точності і швидкодії в умовах реального часу.

Важливим етапом дослідження стало порівняння різних підходів до виявлення об'єктів, включно з Zero-Shot та Two-Shot детекторами, що дозволило вибрати оптимальні алгоритми залежно від цілей і вимог до продуктивності. Також, було детально вивчено методи трекінгу, де особлива увага приділялась фільтру Калмана та алгоритмам на основі кореляційних фільтрів, що забезпечують надійну стабільність у відстеженні об'єктів за умов динамічного середовища.

Реалізована модель, що поєднує глибоке навчання з класичними методами вимірювання відстані до об'єкта за допомогою стереозору та монокулярних підходів, підтвердила ефективність запропонованих рішень. Результати тестувань засвідчили, що така інтеграція дозволяє значно підвищити точність і швидкість виявлення та відстеження об'єктів, що критично важливо для задач моніторингу безпілотних літальних апаратів та інших швидкісних рухомих об'єктів.

Таким чином, робота зробила внесок у розвиток методів комп'ютерного зору для трекінгу швидкісних об'єктів і створила основу для подальших досліджень у цій сфері. Отримані результати можуть бути використані для підвищення безпеки та автоматизації в різних галузях, таких як транспорт, безпека і оборона.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Huadong H., Binyu W., Jiannan X., Tianyu Z. Improved small-object detection using YOLOv8: A comparative study. International Conference on Machine Learning and Automation. 2023. p. 80-88.

2. SOD-YOLO: Small-Object-Detection Algorithm Based on Improved YOLOv8 for UAV Images / Yangang L. et. al. Remote Sensing. 2024. p. 10-14.

3. Discriminative Correlation Filter with Channel and Spatial Reliability. International Journal of Computer Vision / A. Lukežič et. al. 2019. p. 24-27.

4. Xurshedjon F., Suk-Hwan L., Ki-Ryong K. Object Tracking using CSRT Tracker and RCNN. 7th International Conference on Bioimaging. 2020. p. 210-213.

5. Sümeyya I., Fidan G., Merve A., Suhap S. Embedded Visual Object Tracking System Based on CSRT Tracker. International Conference on Electronics, Information, and Communication. February 2022. p. 27-41.

6. Aishwarya R., Maik V., Chithravathi B., Robust object tracking using kernalized correlation filters (KCF) and Kalman predictive estimates. IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology. 2017. p. 10-15.

7. Ming T., Bin Y., Fan Z., Jinqiao W. High-speed Tracking with Multi-kernel Correlation Filters. IEEE Transactions on Instrumentation and Measurement. 2018.

8. Zhigang F., Peng W. A model adaptive updating kernel correlation filter tracker with deep CNN features. Engineering Applications of Artificial Intelligence. 2023. p. 78-91.

9. Dillon R., Jordan K., Jacqueline H., Ahmad D. Real-Time Flying Object Detection with YOLOv8. 2023. 10 p. URL: https://arxiv.org/abs/2305.09972.

10. Muhammad Y. What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector. 2024. p. 12. URL: https://arxiv.org/abs/2408.15857.

11. Ross G., Jeff D., Trevor D., Jitendra M., Rich feature hierarchies for accurate object detection and semantic segmentation. 2014. 21 p. URL: https://arxiv.org/abs/1311.2524v5.

12. Бєздєтний Д. Д., Данилов В. Я. Прогнозування економічних процесів на основі новин з відкритих джерел за допомогою нейронних мереж// Системні науки та інформатика: збірник доповідей III Всеукраїнської науково-практичної конференції «Системні науки та інформатика», 25–29 листопада 2024 року, Київ. – К., НН ІПСА КПІ ім. Ігоря Сікорського, 2024., 7 стор.

13. Shrishti B., Sunita J., Urvi R., Comprehensive Review of R-CNN and its Variant Architectures. 2024. 37 p. URL: https://arxiv.org/html/2402.15490v2.

14. Shaoqing R., Kaiming H., Ross G., Jian S. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2015. 14 p. URL: https://arxiv.org/abs/1506.01497.

15. Yifu Z., Peize S., Yi J., Dongdong Y. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. 2022. URL: https://arxiv.org/abs/2110.06864.

16. Maximilian P., Nahomkifle S., Object Tracking Evaluation: BoT-SORT & ByteTrack with YOLOv8. A Comparison of Accuracy and Computational Efficiency. 2024. p.20-28.

17. Nicolai W., Bewley A., Dietrich P. Simple Online and Realtime Tracking with a Deep Association Metric. 2017. 5 p. URL: https://arxiv.org/abs/1703.07402.

18. Xinyu H., Yi W., Lap-Pui C. Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering. Advanced Video and Signal Based Surveillance. 2019. p. 45-51.

19. Ribeiro M, Ribeiro I. Kalman and Extended Kalman Filters: Concept, Derivation and Properties. CSCI 545: Introduction to Robotics. 2004. 34 p.

20. Yan P., Swarnendu B., Donald F., Keshav P. An Elementary Introduction to Kalman Filtering. 2019. 14 p.

21.Cermakova I., Komarkova J. Modelling a process of UAV data collection and processing. 2016. p. 72-81.

22. UAV-VisLoc: A Large-scale Dataset for UAV Visual Localization / X. Wenjia et. al. 2024. p. 6.

23. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified Real-Time Object Detection. 2016. p. 10. URL: https://arxiv.org/abs/1506.02640.

24. Pande S., Lavanya G. Enhancing Real-time Object Detection with YOLO Algorithm. EAI Endorsed Transactions on Internet of Things. 2023. p. 9.

25. Terven J., Cordova-Esparza D. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Mach. Learn. Knowl. Extr. 2023 v5. p. 1680-1716.

26. Hussain M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. 16th IEEE International Conference on Advanced Video and Signal Based Surveillance. 2023. p. 77-85.

27. The YOLOv8 Edge: Harnessing Custom Datasets for Superior Real-Time Detection / A. Tafreed et. al. 18th International Conference on Emerging Technologies (ICET), 2023. p.20.

28. Aryan Garg, How to Use Yolo v5 Object Detection Algorithm for Custom Object Detection. Computer Modeling in Engineering & Sciences. 2024. p.12.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Train.py

```
!pip install opendatasets --upgrade --quiet

import opendatasets as od

dataset = 'https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-
uav'
od.download(dataset)

dataset = 'https://www.kaggle.com/datasets/pankajkumar2002/random-
image-sample-dataset'
od.download(dataset)

import os

image_folder = '/content/custom_photos'

text_folder = '/content/added_labels'

os.makedirs(text_folder, exist_ok=True)

for filename in os.listdir(image_folder):
    if filename.startswith('no-drone_') and filename.endswith('.jpg'):
#
        identifier = filename.split('_')[1].split('.')[0]

        text_filename = f'no-drone_{identifier}.txt'
        text_filepath = os.path.join(text_folder, text_filename)

        with open(text_filepath, 'w') as file:
            file.write('')
```

```
        print(f'Created {text_filepath}')

image_folder = '/content/random-image-sample-dataset/data'

#image folder
text_folder = '/content/added_labels'

os.makedirs(text_folder, exist_ok=True)

for filename in os.listdir(image_folder):
    if filename.endswith('.jpg') and filename[:-4].isdigit():
        identifier = int(filename[:-4])
        if 20000 <= identifier <= 25000:
            text_filename = f'{identifier}.txt'
            text_filepath = os.path.join(text_folder, text_filename)

            with open(text_filepath, 'w') as file:
                file.write('')

            print(f'Created {text_filepath}')

import locale
locale.getpreferredencoding = lambda: "UTF-8"
!cp -a /content/added_labels/. /content/drone-dataset-
uav/drone_dataset_yolo/dataset_txt
!cp -a /content/custom_photos/. /content/drone-dataset-
uav/drone_dataset_yolo/dataset_txt
!cp -a /content/random-image-sample-dataset/data/. /content/drone-
dataset-uav/drone_dataset_yolo/dataset_txt

"""***Preprocess dataset to apropriate YOLO format***"""

from pathlib import Path

DATA_DIR = Path("/content/drone-dataset-
uav/drone_dataset_yolo/dataset_txt")
print(len(os.listdir(DATA_DIR)))
```

```
images = sorted(list(DATA_DIR.glob("*.jpg")))
print("Total images: ", len(images))
images[:5]

labels = sorted(list(DATA_DIR.glob("*.txt")))
print("Total labels: ", len(labels))
labels[:5]

"""There is difference b/w count of labels and images, finding the
redundant label
***italicized text***
"""

set([label_path.stem for label_path in labels]) - set([image_path.stem
for image_path in images])

"""#### There is a redundant file **"classes.txt"**"""

with open(DATA_DIR/"classes.txt", 'r') as f:
    print(f.read())

FILE_TO_REMOVE = DATA_DIR / "classes.txt"
labels.remove(FILE_TO_REMOVE)

print("Images: ", len(images), " Labels: ", len(labels))

"""## Generating dataset in YOLO format

```
Data
|-- data.yaml
|-- train
| |-- images
| `-- labels
|-- test
| |-- images
| `-- labels
`-- val
```
```

```
    |-- images
    `-- labels
```

```
### Creating directories and copying files
"""

DATA_DIR = Path("Data")
TRAIN_DIR = DATA_DIR / "train"
TEST_DIR = DATA_DIR / "test"
VALIDATION_DIR = DATA_DIR / "val"
os.makedirs(TRAIN_DIR/ "images")
os.makedirs(TRAIN_DIR/ "labels")
os.makedirs(TEST_DIR/ "images")
os.makedirs(TEST_DIR/ "labels")
os.makedirs(VALIDATION_DIR/ "images")
os.makedirs(VALIDATION_DIR/ "labels")

from sklearn.model_selection import train_test_split

X_train,X_temp,y_train,y_temp = train_test_split(images, labels,
test_size = 0.2)
X_val,X_test,y_val,y_test = train_test_split(X_temp, y_temp, test_size
= 0.2)

print("Train image count: ", len(X_train))
print("Validation image count: ", len(X_val))
print("Test image count: ", len(X_test))

X_train[:5], y_train[:5]

import shutil
from tqdm.notebook import tqdm

def copyfiles(files, dest):
    print(f"Copying files to {dest}")
    for file in tqdm(files):
        shutil.copy(file, dest)
```

```
copyfiles(X_train, TRAIN_DIR/ "images")
copyfiles(y_train, TRAIN_DIR/ "labels")

copyfiles(X_test, TEST_DIR/ "images")
copyfiles(y_test, TEST_DIR/ "labels")

copyfiles(X_val, VALIDATION_DIR/ "images")
copyfiles(y_val, VALIDATION_DIR/ "labels")

"""## Creating configuration file `data.yaml`"""

data_yaml = """
train: /content/Data/train/images
test: /content/Data/test/images
val: /content/Data/val/images

names:
  0: drone
"""

with open(DATA_DIR/"data.yaml", "w") as f:
    f.write(data_yaml)

os.listdir(DATA_DIR)

with open(DATA_DIR/"data.yaml", "r") as f:
    print(f.read())

"""## Installing ultralytics and training the Yolov9c model

Removing `wandb` to avoid the unnecessary API Key request while
training the model
"""

!pip uninstall wandb -y
!pip -q install ultralytics
```

```
from ultralytics import YOLO

model = YOLO('yolov9c')

results = model.train(
    data = DATA_DIR/"data.yaml",
    epochs = 120,
    imgsz = 640,
    single_cls = True,
)

model.info()

results = model(source = '/content/image_2024-06-28_213613731.png',
save = True, conf = 0.3)

for result in results:
    boxes = result.boxes.xyxy
    confs = result.boxes.conf
    classes = result.boxes.cls

    for box, conf, cls in zip(boxes, confs, classes):
        label = result.names[int(cls)]
        print(f"Label: {label}, Box: {box.tolist()}, Confidence:
{conf.item()}")

"""Plotting validation metrics"""

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

source = model('/content/drone_1.png', save = True, conf = 0.45)

print(source[0].boxes.xyxy)

vectors = [source[0].boxes.xyxy[i] for i in
range(source[0].boxes.xyxy.shape[0])]
```

```
print(vectors)

tensor_data = source[0].boxes.xyxy

print("Tensor Data:\n", tensor_data)
first_vector = tensor_data[0]
print("\nFirst Vector:\n", first_vector)

vectors = [tensor_data[i] for i in range(tensor_data.shape[0])]
for i, vector in enumerate(vectors):
    print(f"\nVector {i+1}:\n", vector)

numpy_vectors = tensor_data.cpu().numpy()
for i, vector in enumerate(numpy_vectors):
    print(f"\nNumPy Vector {i+1}:\n", vector)

print(numpy_vectors)

image_path

source[0].save("detected.jpg")
Image.open("detected.jpg")
```

Resulting.py

```
import cv2
import torch
import numpy as np
from ultralytics import YOLO

# Load the custom YOLO model using Ultralytics YOLO API
# Replace 'path_to_your_best.pt' with the path to your trained YOLOv5
weights file (e.g., best.pt)
model = YOLO('path_to_your_best.pt')  # Load the custom model
```

```
# Initialize stereo cameras (left and right)
cap_left = cv2.VideoCapture(0)  # Replace with your left camera index or
path
cap_right = cv2.VideoCapture(1)  # Replace with your right camera index
or path

# Ensure the cameras are opened
if not cap_left.isOpened() or not cap_right.isOpened():
    print("Error: Could not open video streams.")
    exit()

# Load stereo calibration data (intrinsics and extrinsics)
# Replace with your actual camera calibration parameters
focal_length = 800  # Example focal length in pixels
baseline = 0.1   # Example baseline in meters (distance between the
cameras)

# Initialize CSRT tracker
trackers = []  # List to store trackers for each detected object
frame_count = 0

while True:
    # Read frames from the stereo cameras
    ret_left, frame_left = cap_left.read()
    ret_right, frame_right = cap_right.read()

    if not ret_left or not ret_right:
        print("Error: Could not read frames.")
        break

    frame_count += 1

    # Perform object detection on the left frame using the custom YOLO
model
    results = model(frame_left)  # Perform detection on the left frame

    # Convert the frame to a format that CSRT tracker can use
    if frame_count % 5 == 0:  # Update trackers every 5th frame
```

```
        # Get bounding boxes of detected objects
        boxes = results.xyxy[0].cpu().numpy()  # YOLO returns (x1, y1,
x2, y2, conf, cls)
        for box in boxes:
            x1, y1, x2, y2, conf, cls = box
            # Initialize the CSRT tracker for each detected object
            tracker = cv2.TrackerCSRT_create()
            tracker.init(frame_left, (x1, y1, x2 - x1, y2 - y1))  #
Initialize tracker with bounding box
            trackers.append(tracker)

    # Update the trackers on each frame
    for tracker in trackers:
        success, bbox = tracker.update(frame_left)
        if success:
            x, y, w, h = [int(v) for v in bbox]
            cv2.rectangle(frame_left, (x, y), (x + w, y + h), (0, 255,
0), 2)

            # Calculate distance for the object using disparity
            # Assume that the object is at the center of the bounding
box
            center_x = int(x + w / 2)
            center_y = int(y + h / 2)

            # Find the corresponding point in the right frame (same y-
coordinate, different x-coordinate)
            # This requires finding the best match for the x-coordinate
of the object in the right frame
            min_disp = None
            best_match_x_right = -1
            for offset in range(-10, 10):  # Search for corresponding x
in a small window around center_x
                x_right = center_x + offset
                if 0 <= x_right < frame_right.shape[1]:  # Ensure it's
within the right frame
                    diff = np.abs(frame_left[center_y, center_x] -
frame_right[center_y, x_right])
```

```
                    if min_disp is None or diff < min_disp:
                        min_disp = diff
                        best_match_x_right = x_right

            if best_match_x_right != -1:
                # Calculate the disparity as the horizontal difference
between the left and right frames
                disparity = np.abs(center_x - best_match_x_right)
                if disparity > 0:  # Avoid division by zero
                    # Calculate the distance to the object using the
disparity
                    distance = (focal_length * baseline) / disparity
                    cv2.putText(frame_left, f'Distance: {distance:.2f}
m', (x, y - 10),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255,
0), 2)

    # Display the stereo frames
    cv2.imshow('Left Frame', frame_left)
    cv2.imshow('Right Frame', frame_right)

    # Exit on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video streams
cap_left.release()
cap_right.release()
cv2.destroyAllWindows()
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

ВИЯВЛЕННЯ ТА ЗАХОПЛЕННЯ ШВИДКІСНИХ РУХОМИХ ЦІЛЕЙ ГЛИБОКИМИ НЕЙРОННИМИ МЕРЕЖАМИ

Студент КА-32мп Бєздєтний Д.Д.

Науковий керівник професор д.т.н. Данилов В.Я.

1

Актуальність дослідження

- Завдання виявлення та захоплення швидкісних рухомих об'єктів є ключовим для сучасних систем моніторингу, безпеки та управління. Висока швидкість і складні умови створюють виклики для традиційних методів, тоді як використання глибоких нейронних мереж, таких як YOLO, у поєднанні з алгоритмами трекінгу та тривимірного аналізу, забезпечує точність і стабільність роботи навіть у реальному часі. Ці підходи мають важливе значення для оборони, транспорту та автоматизованих систем спостереження.

2

Визначення області роботи

- **Об’єкт дослідження:** швидкісні рухомі об’єкти, які необхідно виявляти, відстежувати та захоплювати за допомогою методів комп’ютерного зору та глибоких нейронних мереж.
- **Предмет дослідження:** алгоритми виявлення, трекінгу та вимірювання відстані до швидкісних об’єктів, які базуються на використанні глибоких нейронних мереж у поєднанні з алгоритмами трекінгу та стереоскопічними методами.
- **Мета дослідження:** розробка та експериментальна перевірка інтегрованого підходу до виявлення, трекінгу та вимірювання відстані до швидкісних рухомих об’єктів у реальному часі на основі сучасних моделей глибокого навчання та алгоритмів тривимірного аналізу простору.

3

Постановка дослідження

1. Аналіз та визначення основного підходу для обробки ознак зображень.
2. Збір та підготовка даних.
3. Тренування моделі на зібраних даних.
4. Визначення підходу для захоплення виявлених цілей.
5. Залучення алгоритму для визначення відстані.
6. Проаналізувати майбутні напрямки досліджень.

4

Сіймество YOLO

YOLO (You Only Look Once)

Одноетапна модель детекції об'єктів.

Виконує одночасно виявлення та класифікацію об'єктів на зображенні.

Забезпечує високу швидкість роботи в реальному часі.

Підходить для сценаріїв з обмеженими обчислювальними ресурсами.

5

Сімейство RCNN

RCNN (Region-based Convolutional Neural Network)

Двоетапна модель детекції об'єктів.

Спочатку виділяє регіони-кандидати, а потім класифікує їх.

Забезпечує високу точність, але працює повільніше.

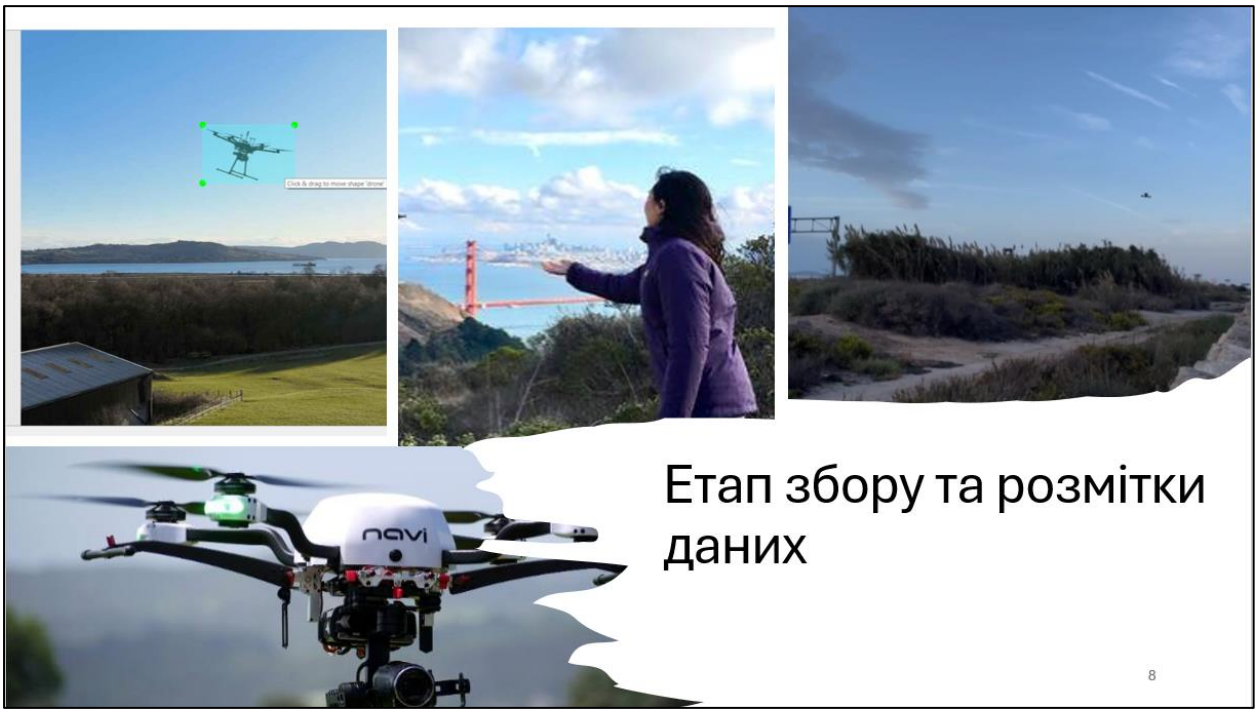
Підходить для задач, де важливіша якість, ніж швидкість.

6

Визначення основного підходу для обробки графічних даних

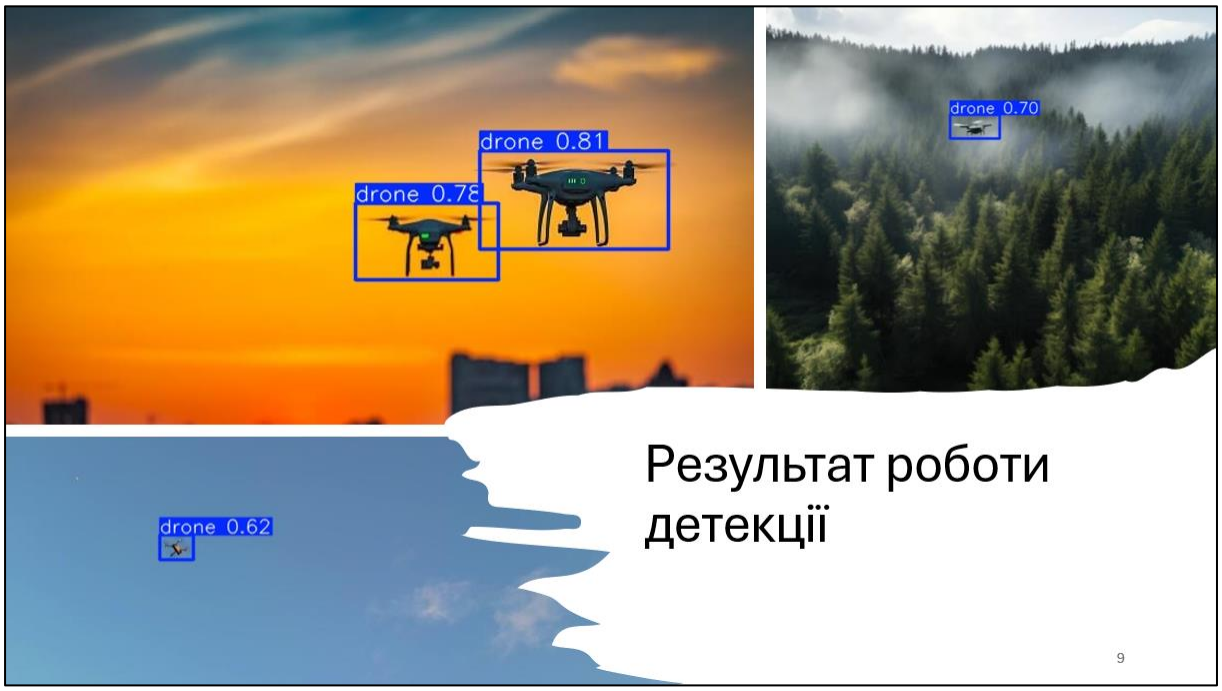
- Оскільки виявлення та захоплення об'єктів є важливою задачею для обробки в реальному часі, то швидкодія системи є основним фактором у прийнятті рішень, отже було вирішено зупинитися на моделі сімейства YOLO, а саме на версії YOLOv11n, оскільки вона є акумулює у собі точність попередніх версій, однак має меншу кількість параметрів, що покращує її швидкодію.

7

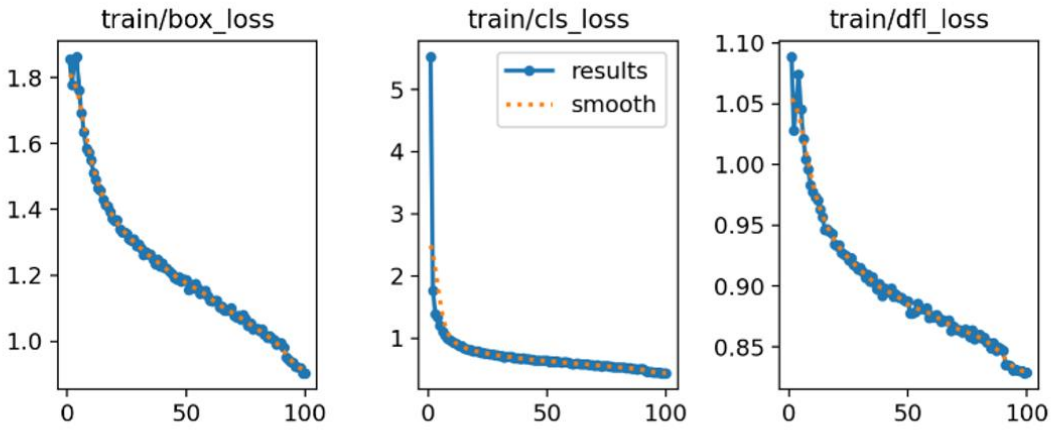


Етап збору та розмітки даних

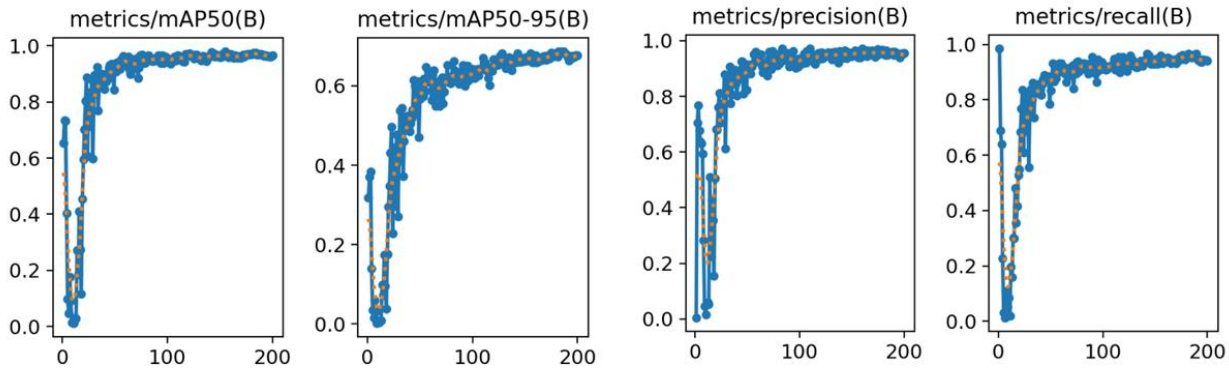
8



Функції втрат

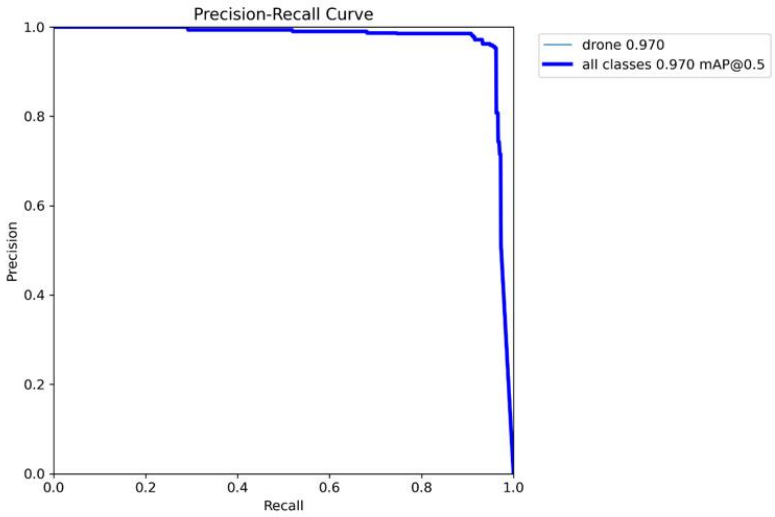


Метрики

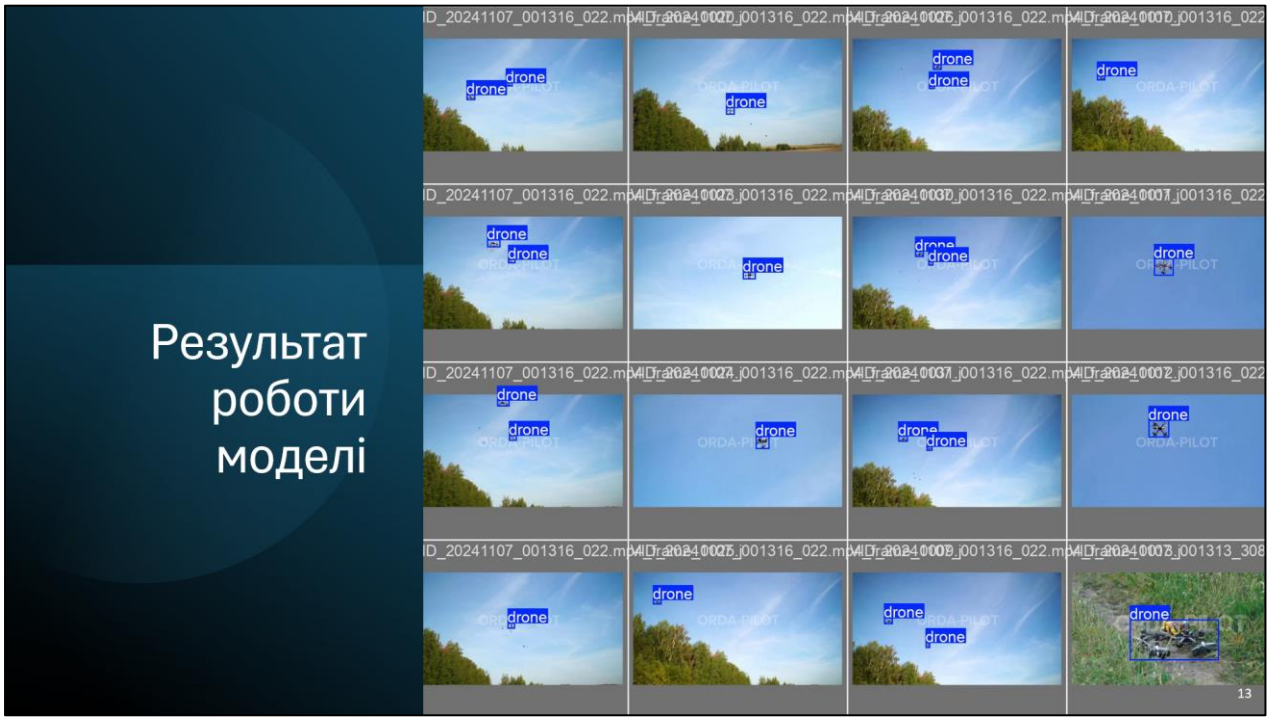


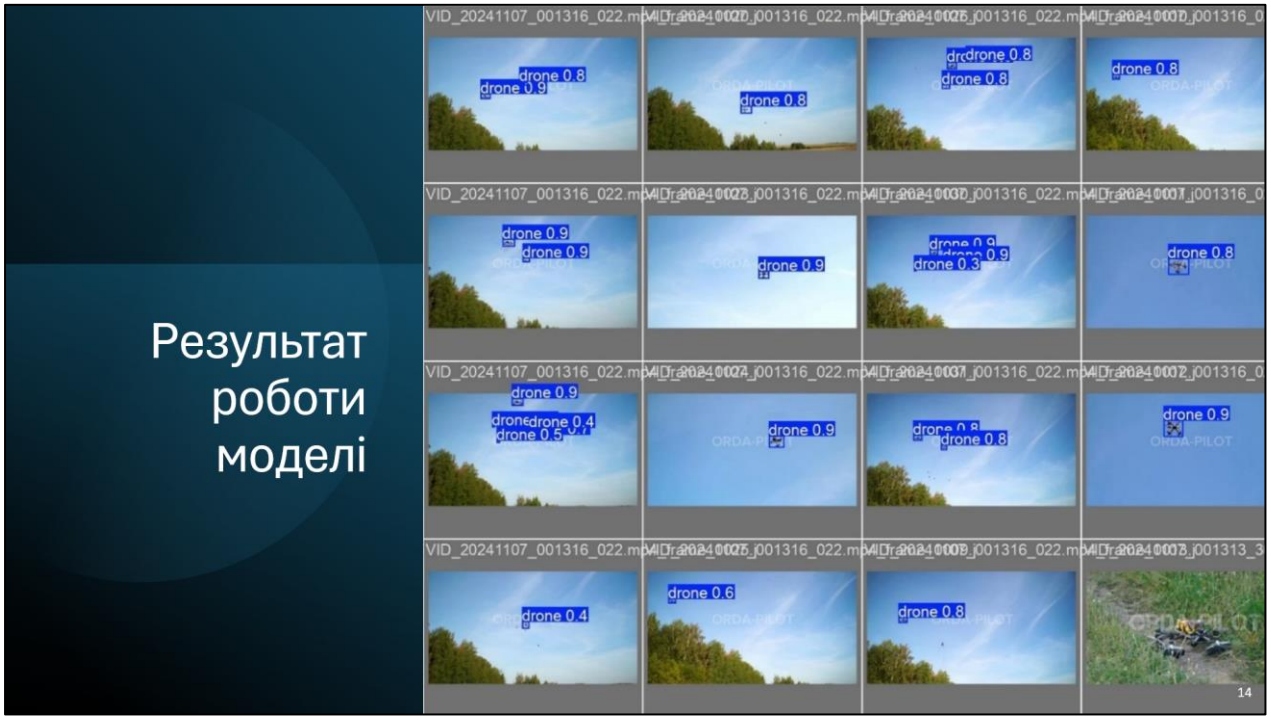
11

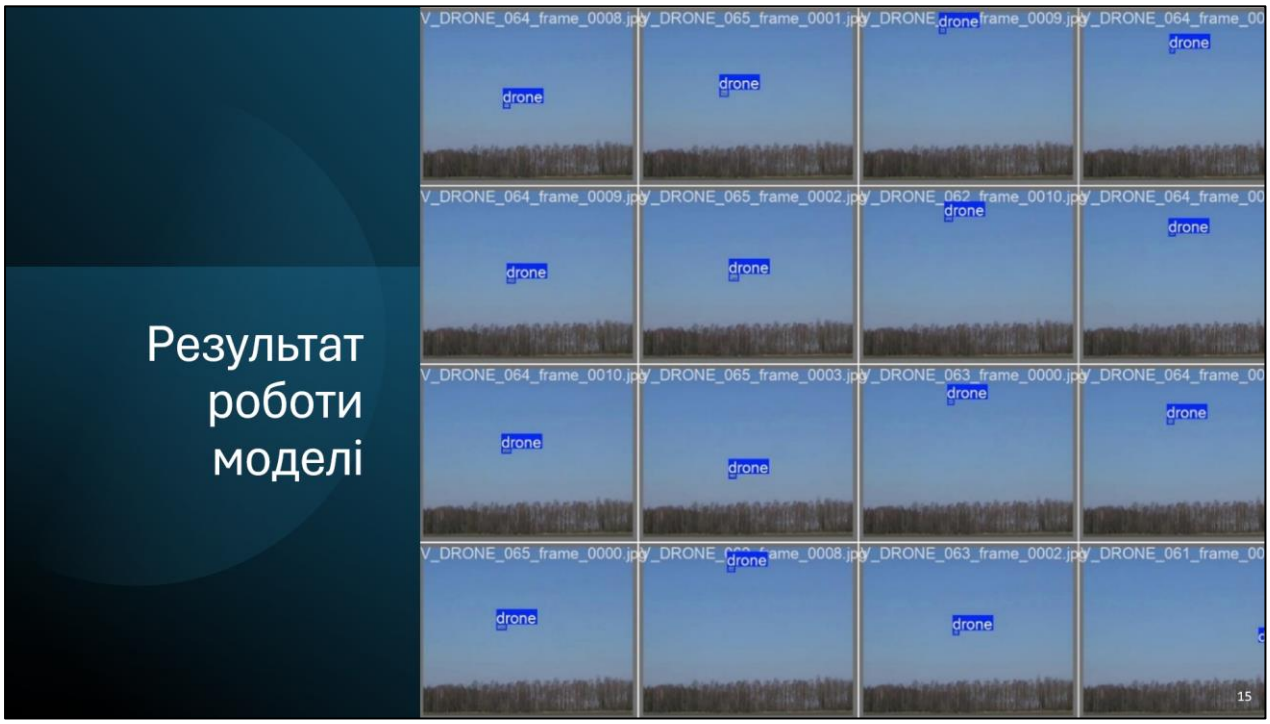
Precision-Recall крива

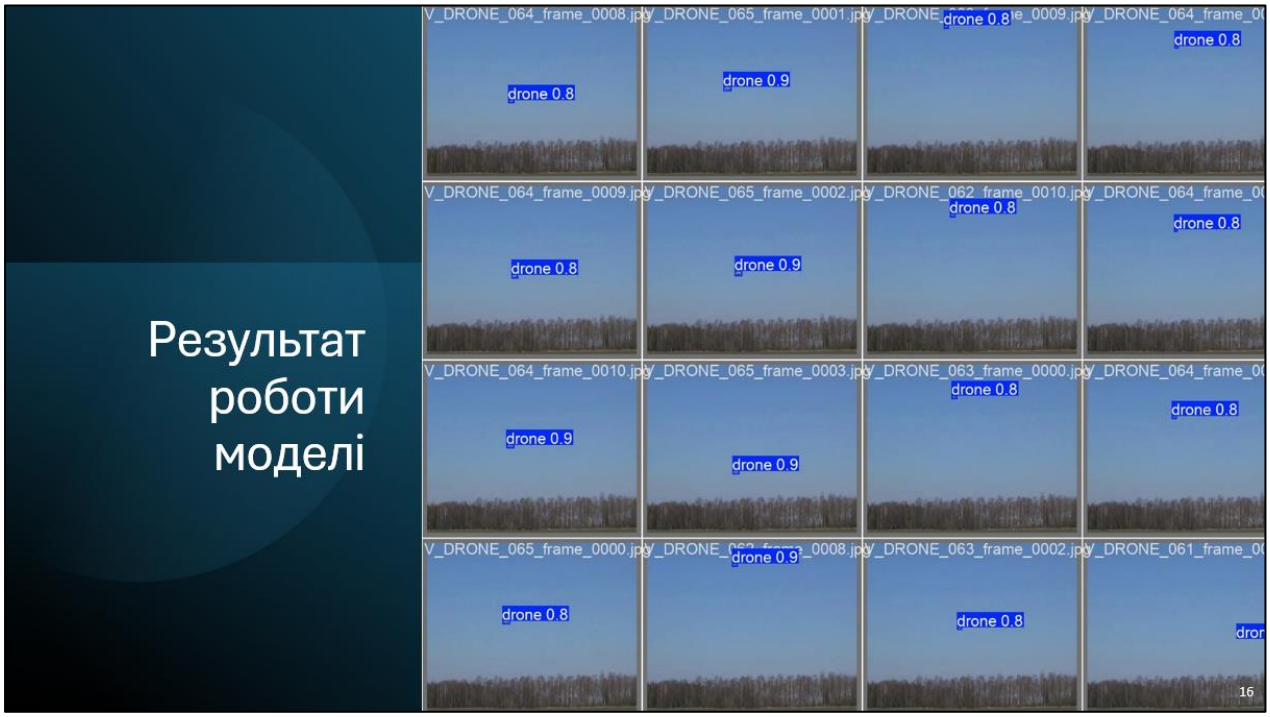


12


Результат роботи моделі
13


Результат роботи моделі
14


Результат роботи моделі
15


Результат роботи моделі
16

Огляд методів захоплення об’єктів

- DeepSORT використовує калманівський фільтр і апаратний аналіз ознак об'єктів, що дозволяє відстежувати об'єкти навіть за умов тимчасових втрат детекції.
- ByteTrack ефективно інтегрує інформацію про детекції об'єктів і їхнє пересування, зберігаючи точність трекінгу навіть у ситуаціях із великою кількістю об'єктів у кадрі.
- KCF (Kernelized Correlation Filters) забезпечує трекінг на основі кореляційних фільтрів із ядровими функціями, що дає змогу швидко й точно відстежувати об'єкти за умови стабільного освітлення та малої кількості змін у сцені.
- CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) перевершує інші кореляційні методи завдяки використанню просторово-канального аналізу, що дозволяє більш точно відстежувати об'єкти в умовах складних змін фону чи освітлення.

10

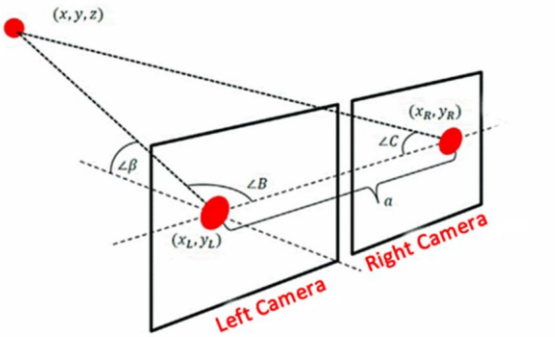
Результат захоплення

11

Опис алгоритму стереоскопічного зору

$distance = \frac{F \cdot C}{disparity}$, де:

- F – фокусна відстань камер(мм),
- C – відстань між камерами(м),
- disparity – різниця координат об'єкта на двох зображення.



12

Результат роботи підходу

13

Висновки

• У ході дослідження розроблено систему для виявлення, трекінгу та визначення відстані до швидкісних об’єктів за допомогою сучасних методів комп’ютерного зору та машинного навчання. На першому етапі створено якісний датасет із зображеннями дронів та інших об’єктів, що забезпечило ефективне навчання моделі YOLO з високою точністю детекції. Серед протестованих алгоритмів трекінгу найкращі результати показав CSRT, завдяки здатності ефективно обробляти швидкі зміни траєкторії. Для визначення відстані реалізовано стереоскопічний метод, який дозволяє точно розраховувати координати об’єктів у реальному часі. Розроблена система характеризується високою точністю та швидкодією, що робить її корисною для задач безпеки, моніторингу та автономних систем.

14

Дякую за увагу!

15