Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Комп'ютерних інтелектуальних технологій та систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
Інтелектуальна комп'ютерна система виявлення та відстеження групи автономних БПЛА
(тема)

Виконав:
Здобувач ІІ курсу, групи КІТм-23-1
Сагайдачний О.М.
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп'ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник доцент каф. КІТС. Ілюнін О.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Руденко О.Г.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерної інженерії та управління
Кафедра	Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп'ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____202_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сагайдачному О. М.
(прізвище, ініціали)

1. Тема роботи (проекту) Інтелектуальна комп'ютерна система виявлення та відстеження групи автономних БПЛА

затверджена наказом університету від «28» жовтня 2024р. № 1156 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 20.01.2024 р.

3. Вхідні дані до роботи (проекту)

1) Інтернет-джерела;
2) Науково-технічні публікації;
3) Проекти з відкритим вихідним кодом.

4. Перелік питань, що потрібно опрацювати у роботі

1) Аналіз предметної області та існуючих рішень;
2) Огляд апаратного забезпечення для вирішення задач машинного зору;
3) Аналіз моделей виявлення об’єктів на кадрі;
4) Аналіз алгоритмів та методів відстеження декількох об’єктів;
5) Аналіз алгоритмів та методів повторної ідентифікації об’єктів;
6) Розробка системи відстеження для безпілотних літальних апаратів;
7) Проведення тестів та аналіз результатів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри)

12 слайдів презентаційного матеріалу

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача та узгодження теми проекту	28.10.2024	Виконано
2	Огляд стану проблеми та постановка задачі	30.11-29.11.2024	Виконано
3	Аналіз літератури за напрямком дослідження	30.11- 07.12.2024	Виконано
4	Аналіз алгоритмів та методів виявлення відстеження декількох об’єктів	08.12-10.12.2024	Виконано
5	Аналіз алгоритмів та методів повторної ідентифікації об’єктів;	11.12-16.12.2024	Виконано
6	Розробка системи відстеження об’єктів; інтеграція системи з автопілотом;	17.12-24.12.2024	Виконано
7	Тестування системи та аналіз результатів	25.12-29.12.2024	Виконано
8	Підготовка презентаційного матеріалу	30.12-23.01.2025	Виконано
9	Подання до ЕК	20.01.2025	Виконано
10	Захист проекту		

Дата видачі завдання «28» жовтня 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доцент каф. КІТС. О.О. Ілюнін
(підпис) (посада, ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної магістерської роботи містить: 102 с., 28 рис., 2 дод., 26 джерел.

АВТОНОМНІ ЛІТАЛЬНІ АПАРАТИ, ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ ДЕКІЛЬКОХ ОБ'ЄКТІВ, ПОВТОРНА ІДЕНТИФІКАЦІЯ ОБ`ЄКТІВ

Робота присвячена дослідженню сучасних методів комп'ютерного зору, зокрема моделей виявлення та повторної ідентифікації об’єктів різних класів, алгоритмів відстеження виявлених об’єктів та розробки децентралізованої системи для відстеження декількох об’єктів у реальному часі групою автономних літальних апаратів в динамічних середовищах.

Об'єктами дослідження є автономні безпілотні апарати та сучасні бортові комп’ютери дронів з графічними процесорами. Предметами дослідження є згорткові нейронні мережі для виявлення об'єктів у відеоряді, методи вилучення унікальних ознак та алгоритми повторної ідентифікації об’єктів, а також рішення для трекінгу, що комбінують методи передбачення траєкторії руху цілей з підходами глибокого навчання.

У ході роботи розроблено адаптивну систему для виявлення та відстеження цілей для безпілотних літальних апаратів в динамічних середовищах. Система інтегрує модель YOLOv11, трекер на основі DeepSORT, а також фреймворк Fast-ReID оптимізованими для роботи при обмежених обчислювальних ресурсів.

Розроблену систему буде можливо адаптувати під різні сценарії використання. Запропоновану роботу та отримані результати рекомендується використовувати як основу для подальших дослідженнях в наступних галузях: відстеження декількох об’єктів декількома камерами, відстеження цілей за допомогою БПЛА та для створення рою дронів.

ABSTRACT

The explanatory note of the qualification master's thesis contains: 102 p., 28 fig., 2 appendices, 26 sources.

AUTONOMOUS AERIAL VEHICLES, DETECTION AND TRACKING OF SEVERAL OBJECTS, RE-IDENTIFICATION OF OBJECTS

The work is devoted to the study of modern computer vision methods, in particular, models for the detection and re-identification of objects of different classes, algorithms for tracking detected objects, and the development of a decentralized system for tracking multiple objects in real time by a group of autonomous aerial vehicles in dynamic environments.

The objects of the study are autonomous unmanned aerial vehicles and modern on-board computers of drones with graphics processors. The subjects of the research are convolutional neural networks for detecting objects in video sequences, methods for extracting unique features and algorithms for re-identifying objects, as well as tracking solutions that combine methods for predicting the trajectory of target movement with deep learning approaches.

In the course of the work, an adaptive system was developed for detecting and tracking targets for the unmanned aerial vehicles in dynamic environments. The system integrates the YOLOv11 model, a tracker based on DeepSORT, and the Fast-ReID framework optimized for operation with limited computing resources.

The developed system will be able to be adapted to various usage scenarios. The proposed work and the obtained results are recommended to be used as a basis for further research in the following areas: tracking multiple objects with multiple cameras, tracking targets using drones, and for creating a swarm of UAVs.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Інтелектуальна комп'ютерна система виявлення та відстеження групи автономних БПЛА

Виконав:
здобувач ІІ курсу, група КІТм-23-1
Сагайдачний О. М.
(прізвище, ініціали)

Спеціальність – 123 Комп’ютерна інженерія
Тип програми – освітньо-професійна
Освітня програма – Комп’ютерні інтелектуальні технології

Керівник доцент каф. КІТС. Ілюнін О.О.
(посада, прізвище, ініціали)

2024 р.

АНОТАЦІЯ

Сагайдачний О. М. Інтелектуальна комп'ютерна система виявлення та відстеження групи автономних БПЛА. – Магістерська кваліфікаційна робота.

У магістерській кваліфікаційній роботі вирішено актуальну задачу комютерного зору – алгоритм для виявлення та відстеження декількох об`єктів одночасно в режимі реально часу. Запропонаний алгоритм протестовано та інтегровано з автопілотом для безпілотних літальних апаратів.

Метою кваліфікаційної роботи є розробка легкого та ефективного алгоритму відстеження декількох об’єктів, оптимізованого для використання на безпілотних літальних апаратах. Основна увага зосереджена на створенні системи, здатної працювати в режимі реального часу в умовах обчислювальних обмежень бортових комп’ютерів-супутників, зберігаючи при цьому високу точність у виявленні та відстеженні об’єктів, таких як транспортні засоби та люди.

Об'єктами дослідження цієї роботи є різні типи безпілотних апаратів, рої дронів, а також одноплатні компютери з графічними процесорами в якості бортових комп’ютерах.

Предмети дослідження: згорткові нейронних мережі для виявлення об’єктів у відеопотоках, методи вилучення унікальних характеристик об’єктів та алгоритми повторної ідентифікації для розрізнення схожих об’єктів з часом. Додатково, досліджувалися методики відстеження, які об’єднують моделі прогнозування траєкторії з підходами глибокого навчання для створення надійних рішень відстеження кількох об’єктів, придатних для застосування в повітрі.

У першому розділі представлено вичерпний огляд безпілотників, їх класифікацію та сфери застосування. Розглянуто потенціал використання груп безпілотних літальних апаратів або роїв. Увагу акцентовано на спільних сценаріях, таких як пошуково-рятувальні місії та картографування території, де використання кількох БПЛА значно підвищує продуктивність. У першому розділі також проаналізовано різні моделі зв'язку які використовуються в БПЛА; порівняно централізований і децентралізований підходи. Додатково, розглянуто однопалатні комп'ютери як комп'ютери-супутники для надання достатніх обчислювальних потужностей. В підрозділі про бортові комп'ютери наведено загальний огляд сучасних плат від NVIDIA та Raspberry, їх обчислювальні можливості та обмеження.

В другому підрозділі проведено детальний аналіз штучних нейронних мереж, з особливою увагою до згорткових нейронних мереж (CNN). У розділі описано їхнє значення в обробці візуальних даних і дозволі дронам ефективно аналізувати складні сцени, утворюючи основу багатьох програм комп'ютерного зору. Наприкінці першого розділу розглядаються загальні проблеми комп'ютерного зору, а також деякі існуючі рішення. Розглянуто системи стеження з декількома камерами. Розглянуті проблеми включають асоціацію даних для правильного зіставлення об'єктів між кадрами та повторна ідентифікація об`єкта на різних камерах.

У другому розділі проведено поглиблений аналіз сучасних систем виявлення та відстеження об'єктів. Спершу визначено метрики для аналізу різних моделей та алгоритмів, в наступній частині розглянуто алгоритми, та арзітектури сучасних моделей. Мета другого розділу полягалє в огляді літератури та наукових робіт на обрану тему, визначенні показників оцінки та визначенні загальних тенденцій і проблем у сучасних системах комп'ютерного зору, а також поглиблений аналіз наявних рішень.

В розділі зроблено огляд показників, які використовуються для оцінки детекторів об'єктів, таких як середня середня точність і перетин через

об’єднання. Для систем відстеження кількох об’єктів було визначено та проаналізовано специфічні показники відстеження, такі як MOTA, IDF1 та інші. За допомогою визначених метрик розглянуто та проаналізовано сучасні моделі виявлення об’єктів включаючи їх архітектуру, методи навчання, та час обробки кадрів. В другомиу розділі також проаналзовано моделі для виділення ознак, оскільки вони відіграють вирішальну роль у відстеженні та повторній ідентифікації об’єктів. У розділі розглядалися такі архітектури, як OSNet і ResNet, наголошуючи на їхній продуктивності у вилученні дискримінаційних ознак для людей і транспортних засобів.

Наприкінці другого розділу розглянуто сучасні системи відстеження, у тому числі ті, що впроваджують парадигму асоціації даних та використовують такі стратегії, як зіставлення на основі ознак, фільтрація Калмана та повторна ідентифікація.

У третьому розділі представлено розроблену систему відстеження декількох об`єктів, запропоновану систему протестовано при різних сценаріях, представлено інтеграцію системи з PX4 автопілотом.

Для створення надійної системи відстеження було використано кілька наборів даних: VisDrone; VERI та VehicleID; Market-1501, PRAI-1581 та MRP. VisDrone був основним набором даних для навчання та перевірки моделей виявлення об’єктів. Датасет надав різноманітні сценарії та анотації для виявлення людей і транспортних засобів на аерофотознімках. Набори даних VERI та VehicleID використовувалися для навчання моделі OSNet спеціально для завдань повторної ідентифікації транспортних засобів;Зображення з наборів даних Market-1501, PRAI-1581 та MRP використовувались для навчання моделей, вилучення характеристик, унікальних ознак для людини.

Для проведення досліджень було використано новітні фрейворки ultralitics, fast-ReId, а також класичний набір утиліт torchreid. В якості моделі вивлення об`єктів на кадрі використано YOLOv11 – на момент проведеня

даного дослідження останню модель класу YOLO. Деклько варіантів моделі YOLOv11 були навчені за допомогою набору даних VisDrone із 300 епохами та ранньою зупинкою, реалізованою після 100 епох без покращення. При тренуванні, розмір серії визначався автоматично для стабілізації навчання.

У ході проведення дослідження декілька моделей YOLOv11 різних розмірів (маленькі, середні та нано) були навчені та експортовані в низку форматів, сумісних із розгортанням на бортових комп`ютерах (наприклад, PyTorch, TorchScript, TensorRT). Загалом було створено та проаналізовано 15 моделей, щоб визначити найкращий компроміс між точністю та продуктивністю обчислень.

Для повторної ідентифікації було застосовано дві окремі OSNet моделі для виявлення унакільних ознак: одна модель для транспортних засобів, інша для людей. Обрані моделі були навчені з використанням відповідних наборів даних які включають знімки з дронів на різній висоті. Різноманітність наборів даних забезпечела високу точність вилучення унікальних характеристик для різних класів об’єктів.

Після тренування нейромереж, реалізовано оптимізований трекер об`єктів – DroneSort. Функціонал трекеру включає у себе: модель компенсації руху камери, модуль повторної ідентифікації, та інші. Модель компенсації руху камери реалізовано за допомогою моделі покращеного коефіцієнта кореляції (ECC) для врахування руху літальноuj апарату та повороте камери.

Модуль повторної ідентифікації: працює на основі фіксованого кільцевого буферу ознак об`єктів для кожного треку. Подібність між поточними виявленнями та існуючими обчислювалася за допомогою косинусної або евклідової відстані. Щоб підвищити точність повторної ідентифікації людей і транспортних засобів використовувалися окремі екстрактори ознак. Кожен трек було розширено такими атрибутами, як локальний ідентифікатор, глобальний ідентифікатор, клас і максимальний

показник надійності.

Трекер був інтегрований з автопілотом PX4. Міст uXRCE-DDS ROS2 забезпечив безперебійний зв’язок між комп’ютером-супутником і PX4. Вузол ROS2, DroneImageHandler, був створений для обробки вхідних зображень із передньої камери БПЛА, обробки виявлень через YOLOv11 та виконання операцій відстеження. Результати, включаючи обмежувальні рамки та ідентифікатори об’єктів, були записані у відеофайли для подальшого аналізу. Інтеграція дозволила літальним апаратам відстежувати об`єкти.

Для тестування отриманої системи, виконано моделювання за допомогою PX4 Software-in-the-Loop (SITL) із симулятором Gazebo. Симуляція продемонструвала, що трекер ефективно працював з сценаріями з одним і кількома об’єктами, в режимі реального часу в умовах меншого скупчення людей. Для сцен з об’єктами з високою щільністю спостерігалися компроміси продуктивності, але залишалися в прийнятних межах для додатків БПЛА.

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, АВТОНОМНІ ДРОНИ, НЕЙРОННІ МЕРЕЖІ, МАШИННИЙ ЗІР, ВИЯВЛЕННЯ ОБ'ЄКТІВ, ВІДСТЕЖЕННЯ ДЕКІЛЬКОХ ОБ'ЄКТІВ, БАГАТОКАМЕРНІ СИСТЕМИ, ПОВТОРНА ІДЕНТИФІКАЦІЯ ОБ`ЄКТІВ

Публікації здобувача за темою роботи:

Сагайдачний О. М., Аксак Н. Г. Використання мультиагентних систем для дослідження невідомих середовищ. *РОЗВИТОК СУСПІЛЬСТВА ТА НАУКИ В УМОВАХ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ*. 2024. С. 327–329. URL: https://doi.org/10.62732/liga-inter-31.05.2024.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UAV (Unmanned aerial vehicle) – Безпілотні літальні апарати, БПЛА

GPU (Graphics Processing Unit) – Графічний процесор

CNN (Convolutional Neural Network) – Згорткова нейронна мережа

CV (Computer vision) – Комп'ютерний зір

VT (Visual tracking) – Візуальне відстеження

SOT (Single-object tracking) – Відстеження одного об'єкту

MOT (Multi-object tracking) – Відстеження кількох об'єктів

ODM (Object Detection Model) – Модель виявлення об'єктів

IoU (Intersection over Union) – Перетин через об’єднання

KF (Kalman Filter) – Фільтр Калмана

YOLO (You Only Look Once) – Детектор “You Only Look Once”

SSD (Single-Shot Detector) – Одноступінчастий детектор

R-CNN (Region-Based CNN) – Згорткова нейронна мережа на основі регіону

OSNet (Omni-Scale Network) – Мережа Omni-Scale

NN (Nearest Neighbor Metrics) – Метрика “найближчий сусід”

ROS (Robot Operating System) – Роботична операційна система

ЗМІСТ

ВСТУП

У сучасному світі автономні безпілотні апарати відіграють все більшу роль у різних сферах діяльності людини [1, 3, 4]. Безпілотні системи почали застосовуватись у багатьох напрямках включаючи військову справу, сільське господарство, виконання пошуково рятувальних операціях, логістику та доставку вантажів, та навіть кіноіндустрію. Для того, щоб ефективно виконувати комплексні задачі в зазначених напрямках необхідно мати систему яка б складалась з декількох автономних дронів здатних взаємодіяти між собою для досягнення спільної мети.

В контексті безпілотних систем, основним критерієм автономності є здатність системи виконувати задачі без людського втручання або з мінімальним втручанням. Для створення автономних дронів необхідно навчити керовані дрони сприймати навколишнє середовище за допомогою набору різних сенсорів, а також самостійно приймати рішення на основі отриманих даних. Основним джерелом даних для автономних безпілотних систем є візуальна інформація, тобто дані отримані з різних камер.

Сучасні технології машинного зору поєднують глибокі нейронні мережі та надійні математичні алгоритми для вирішення багатьох задач у даній галузі. Нові моделі та алгоритми надають можливість точно виявляти та відстежувати декілька об’єктів одночасно навіть при наявності перешкод або шумів в відеоряді. Додатково, з стрімким розвитком графічних та тензорних блоків обробки інформації, з’явились компактні одноплатні комп’ютери здатні здійснювати обробку кадрів з відео, а саме виявлення та відстеження цілей, у режимі реального часу.

Групу безпілотних апаратів яка виконує задачу що включає виявлення або відстеження цілей можна класифікувати як динамічну багатокамерну систему відстеження декількох об’єктів. Створення багатокамерної системи на основі автономних агентів відкриє нові області для застосування дронів

у цивільній, індустріальній та військових сферах.

Мета дипломної роботи – створити багатокамерну систему для відстеження декількох об'єктів, яка б могла використовуватися автономними безпілотними апаратами. Така система має забезпечити високу точність і надійність спостереження, що є критично важливим для ефективного виконання завдань безпілотними апаратами.

Особливу увагу в даній роботі буде приділено використанню сучасних методів, таких як виявлення об'єктів, відстеження та повторна ідентифікація. Використання передових алгоритмів машинного навчання та комп'ютерного зору дозволяє досягти високої точності та швидкості обробки даних, що є необхідним для ефективного функціонування багатокамерних систем.

В роботі детально описано архітектуру системи, вибір апаратного забезпечення, програмних засобів та алгоритмів. Особливу увагу приділено інтеграції різних компонентів системи та їхньої взаємодії. Запропонована архітектура забезпечує фундамент для подальших розробок та розвитку системи і реалізації її модифікацій під конкретні умови експлуатації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Безпілотні літальні апарати

1.1.1 Загальні відомості

Безпілотні літальні апарати (БПЛА), також відомі як дрони або безпілотники – це літальні апарати які не мають пілота на борту та можуть керуватись дистанційно або виконувати задачі автономно [2]. Завдяки своїй гнучкості, різним конфігураціям та широким набором характеристик, дрони використовуються в багатьох сферах включаючи військові, комерційні та цивільні напрямки. З розвитком безпілотних систем та інтелектуальних застосунків з’явилась можливість замінити людей автономними дронами для вирішення складних або небезпечних задач без ризику для життя людини. Сучасні дрони забезпечують більш економічні операції та мають більшу маневреність аніж еквівалентні пілотовані системи.

Більшість дронів дистанційно керуються оператором, при такому сценарії команди управління надходять із наземної станції контролю або через пульт дистанційного керування. Також є програмні рішення для дронів які здатні виконувати контрольні операції на борту за допомогою автопілоту та різних датчиків, таких як глобальну систему позиціонування та інерціальні вимірювальні пристрої, системи навігації і картографії на основі візуальних та лідар даних.

Використання різних датчиків, систем навігації, технологій комп’ютерного зору дозволяє створювати автономні дрони які можуть виконувати пілотні задачі без втручання операторів. БПЛА взаємодіють із супутниками, наземними системами управління, комп’ютерами та навіть з

смартфонами через спеціалізовані канали зв’язку. Також дрони можуть виконувати задачі автономно в тих ситуаціях, коли втручання людини є складним або небезпечним.

Безпілотні літальні апарати характеризуються швидким та легким розгортанням, високою масштабованістю та гнучкістю, здатністю до самоорганізації, економічною ефективністю та високою маневреністю. Безпілотники демонструють значну варіативність у конфігурації, розмірах, радіусі дії, вазі, типах двигунів та експлуатаційних характеристиках. Це дозволяє їм нести різноманітні корисні навантаження, включаючи засоби зв’язку, навігаційне обладнання, датчики та камери.

Незважаючи на безліч переваг, існують фактори які обмежують продуктивність та області використання дронів. Такими факторами є обмежена витривалість батареї, обмежена мобільність, обмежена автономність і обмежений час польоту. Обмежений час польоту пояснюється різними факторами, зокрема погодними умовами, точністю датчиків, розміром апарата, а також ресурсом батареї. У більшості випадків для керування дронів які не мають підтримки автопілоту потрібна трудомістка, дорога та інтенсивна підготовка операторів. Багатьом людям важко керувати БПЛА, елементи ручного керування вразливі до атак, можуть бути неефективності а також людський фактор може призвести до критичних помилок.

За останнє десятиліття дрони почали використовувалися в різних сферах застосування. Активний розвиток та використання безпілотних літальних апаратів почалося саме у військовій сфері, проте через деякий час дрони почали використовуватись і в цивільних цілях.

Типові задачі для дронів включають дослідження та картографування важкодоступних або невідомих середовищ, виконання пошукових та рятувальних операцій, перевезення вантажів, охорона периметру, моніторинг ділянок місцевості та інші. Найпоширенішими варіантами

використання дронів є аерофотозйомка, розважальних польотів. Останнім часом дрони стали дуже зручним підходом для логістики, а також для доставки авіапошти. Також дрони почали використовувати для автоматизація роботи в сільському господарстві

1.1.2 Класифікація та характеристики БПЛА

Дрони мають перелік різних характеристик, які визначають та обмежують їх можливості, на рисунку 1.1 зображено класифікацію за різними типами параметрів. Основні характеристики включають:

- Тривалість польоту та радіус дії;
- Максимальна та середня швидкості;
- Вантажопідйомність, стабільність та маневреність;
- Встановлені камери та додаткові сенсори.

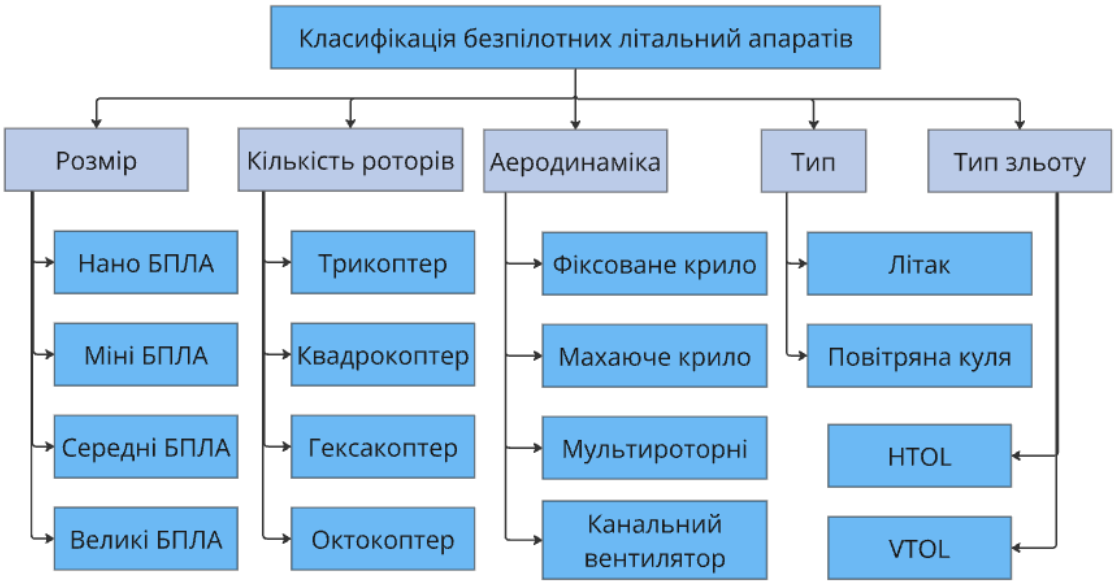

Рисунок 1.1 – Класифікація БПЛА за різними параметрами

Розуміння перелічених характеристик допомагає визначити, який тип БПЛА найбільше підходить для виконання певної місії та забезпечує ефективне використання його можливостей.

Найпростішими та найпоширенішими дронами є квадрокоптери. Квадрокоптери є найбільш часто використовуваними безпілотними літальними апаратами. Квадрокоптери можуть виконувати вертикальний зліт та посадку, мають високу маневровність, швидко реагують на команди оператора. Вони привернули увагу широкого кола користувачів, оскільки є хорошими кандидатами для виконання широкого спектру завдань, включаючи аерофотозйомку, дослідження місцевості, доставку малих вантажів та інші задачі.

Окрім квадрокоптерів існує велика кількість інших конфігурацій дронів. Як правило, дрони поділяють за наступними критеріями: розміри та вага, конструкція, вид двигуна та радіус дії. Насамперед, головним критерієм БПЛА є маса та розмір. Розмір дрону у більшості випадків одразу дозволяє визначити його основне призначення. За конструкцією БПЛА бувають літакового та багатороторного типу, проте також існують гібридні версії, а також конвертоплани.

Безпілотники літакового типу використовують крило для створення підйомної сили та польоту аналогічно до того, як це роблять звичайні літаки. Вони не здатні зависати в повітрі, але можуть рухатися вперед по заданому курсу протягом тривалого часу, поки дозволяє їхнє джерело живлення. Такі БПЛА є більш економічними, швидшими і ідеально підходять для спостереження за великими площами завдяки своїй здатності покривати значні відстані з високою швидкістю.

Багатороторні дрони у свою чергу поділяють на безпілотні гелікоптери, тобто дрон з одним ведучим гвинтом, а також трикоптери, квадрокоптери, гексакоптери та октокоптери. Трикоптери, тобто дрони з трьома роторами забезпечують хорошу маневреність та довший час польоту, але вимагають складної системи управління для стабілізації. Квадрокоптери є найпоширенішими завдяки їхній простій конструкції, стабільності та універсальності, що робить їх популярними в цивільних і

комерційних застосуваннях. Гексакоптери з шістьма роторами та октокоптери з вісьмома роторами мають більшу вантажопідйомність та забезпечують кращу стабільність.

За типом двигуна дрони можна класифікувати на електричні та паливні моделі. Електричні дрони працюють на акумуляторах, що робить їх тихішими за паливні, екологічними та простими в обслуговуванні, проте їхній час польоту обмежений ємністю батареї. З іншого боку, паливні дрони використовують бензинові або дизельні двигуни, які забезпечують значно триваліший час польоту та високу потужність, але вони більш шумні, складніші в реалізації та підтримці, а також потребують регулярного технічного обслуговування.

Гібридні моделі БПЛА, приклад якого зображено на рисунку 1.2, поєднують в собі характеристики багатороторних та літакових дронів. Подібні моделі мають функції вертикального зльоту і посадки, як у мультикоптерів, що робить їх надзвичайно зручними для використання в обмежених просторах або на нерівних поверхнях, а також мають фіксоване крило для планерування.

Рисунок 1.2 – Приклад гібридного БПЛА (VTOL)

Після зльоту гібридні БПЛА переходять у горизонтальний політ, використовуючи крила, як звичайні літаки, що дозволяє їм ефективніше долати великі відстані з меншою витратою енергії. Завдяки такому поєднанню, гібридні дрони можуть виконувати завдання, які зазвичай потребують двох різних типів апаратів.

1.1.3 Бортові комп’ютери та їх характеристики

Бортові комп’ютери необхідні для виконання автономних завдань у безпілотних літальних апаратах. Прикладами таких завдань є одночасна локалізація та картографування, пошук цілей в динамічних середовищах, або як в даній роботі - виявлення та відстеження кількох об’єктів.

Бортові комп’ютери обробляють дані від різноманітних датчиків, включаючи GPS, IMU, LiDAR та камери різних типів, що дозволяє дронам точно орієнтуватися та виконувати поставлені задачі. Вони забезпечують безперебійне виконання завдань, що потребують інтенсивного обчислення, зберігаючи при цьому низьке енергоспоживання – критичний фактор для БПЛА, що працюють від обмеженої ємності акумулятора. Крім того, ці системи забезпечують підключення до мережі для передачі даних у реальному часі, життєво важливого для комерційних і військових застосувань.

Для роботи з нейронними мережами, додаткові обчислювальні пристроїв мають відповідати наступним виогам:

- Наявність прискорювачів для роботи з нейронними мережами;
- Розмір плати для інтеграції в дрони різних типів;
- Енергоефективність.

Сучасні бортові обчислювальні платформи використовують GPU, TPU або FPGA для ефективної обробки згорткових нейронних мереж (CNN). Платформи Raspberry Pi і NVIDIA Jetson є одними з найпопулярніших бортових обчислювальних рішень для БПЛА завдяки своїм компактним розмірам, енергоефективності та надійним можливостям обробки. В таблиях 1.1 та 1.2 порівнюються основні характеристики модулів серіїї NVIDIA Jetson.

Таблиця 1.1 – Порівняння NVIDIA Jetson модулів, частина 1

Назва пристрою	Jetson AGX Orin 64GB	Jetson Orin NX 16GB	Jetson AGX Xavier
Продуктивність ШІ	275 TOPS	100 TOPS	32 TOPS
GPU	2048-core NVIDIA Ampere (64 Tensor Cores)	1024-core NVIDIA Ampere (32 Tensor Cores)	512-core Volta (64 Tensor Cores)
Макс. Частота GPU	1.3 GHz	918 MHz	1377 MHz
CPU	12-core Cortex A78AE	8-core Cortex A78AE	8-core Carmel Arm®v8.2
Макс. Частота CPU	2.2 GHz	2.0 GHz	2.2 GHz
Пам'ять	64GB 256-bit LPDDR5 204.8GB/s	16GB 128-bit LPDDR5 102.4GB/s	32GB 256-bit LPDDR4x 136.5GB/s
Потужність	15W - 60W	10W - 40W	10W - 30W
Розмір пристрою	100mm x 87mm	69.6mm x 45mm	100mm x 87mm

Таблиця 1.2 – Порівняння NVIDIA Jetson модулів, частина 2

Назва пристрою	Jetson Orin Nano Super	Jetson Xavier NX	Jetson Nano
Продуктивність ШІ	67 TOPs	21 TOPS	472 GFLOPS
GPU	1024-core Ampere (32 Tensor Cores)	384-core Volta™ (48 Tensor Cores)	128-core Maxwell™
Макс. Частота GPU	1020 MHz	1100 MHz	921MHz
CPU	8-core Cortex A78AE	6-core Carmel Arm®v8.2	Quad-Core Arm® Cortex®-A57
Макс. Частота CPU	1.7 GHz	1.9 GHz	1.43GHz
Пам'ять	8GB 128-bit LPDDR5 102 GB/s	8GB 128-bit LPDDR4x 59.7GB/s	4GB 64-bit LPDDR4 25.6GB/s"
Потужність	7W - 25W	10W - 20W	5W - 10W
Розмір пристрою	69.6mm x 45mm	69.6mm x 45mm	69.6mm x 45mm

Основною характеристикою наведених пристроїв є «Tera Operations Per Second» (TOPS) – кількость операцій, таких як додавання, множення та логічні обчислення, які процесор може виконати за одну секунду, вимірюється в трильйонах (10^{12}). TOPS визначається шляхом оцінки здатності процесора ефективно виконувати паралельні операції. Фактори, що впливають на TOPS, включають тактову частоту, кількість процесорних ядер і точність операцій (наприклад, INT8, FP16). Для прискорювачів штучного інтелекту вищі показники TOPS часто є результатом завдань з нижчою точністю (наприклад, арифметика INT8), які менш вимогливі до обчислень порівняно з завданнями з вищою точністю (наприклад, FP32).

TOPS служить ключовим індикатором придатності процесора для додатків ШІ в режимі реального часу. Більш високе значення TOPS дозволяє дронам справлятися зі складними робочими навантаженнями, такими як виявлення об’єктів, відстеження та SLAM з мінімальною затримкою. Крім того, процесори з високим співвідношенням TOPS/Watt забезпечують енергоефективні рішення, що має вирішальне значення для БПЛА, що працюють з обмеженим часом автономної роботи.

Відповідно до таблиці 1.2 Jetson Nano: ідеально підходить для малих дронів. Чотириядерний процесор Arm Cortex-A57, що працює на частоті 1,43 ГГц, забезпечує достатню обчислювальну потужність для таких основних завдань, як виявлення та повторна ідентифікація об’єктів, а також відстеження в режимі реального часу, споживаючи при цьому мінімальну енергію. Jetson AGX Orin 64 ГБ, наданий в таблиці 1.1, є оптимальним рішенням для більших безпілотних літальних апаратів великої дальності з розширеними обчислювальними потребами. AGX Orin може легко інтегрувати SLAM системи, а також відстеження кількох об’єктів у реальному часі на основі лідар даних.

На рисунку 1.3 наведенва продуктивність YOLO моделі в форматах: PyTorch, TorchScript і TensorRT на NVIDIA Jetson Nano.

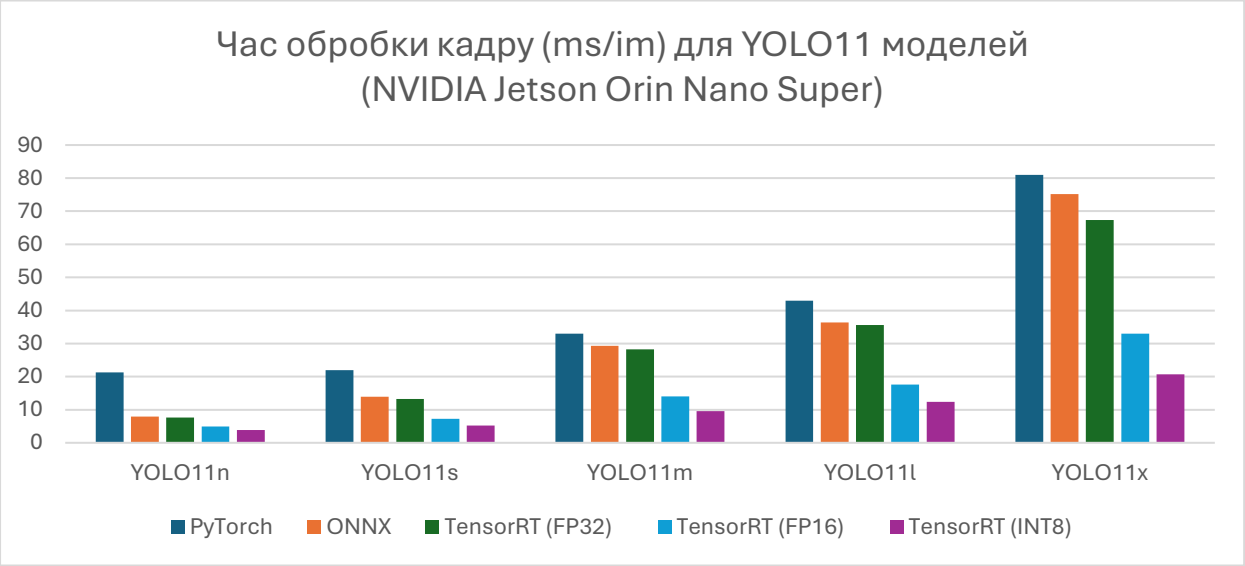

Рисунок 1.3 – Продуктивність YOLO на jetson Nano

Для порівняння, на рисунку 1.4 продемонстровано продуктивність моделей в різних форматах на Raspberry Pi 5. Незважаючи на те, що модель моджливо запустити на RPi 5, отримані результати значно гірші, у порівнянні з NVIDIA Jetson Nano.

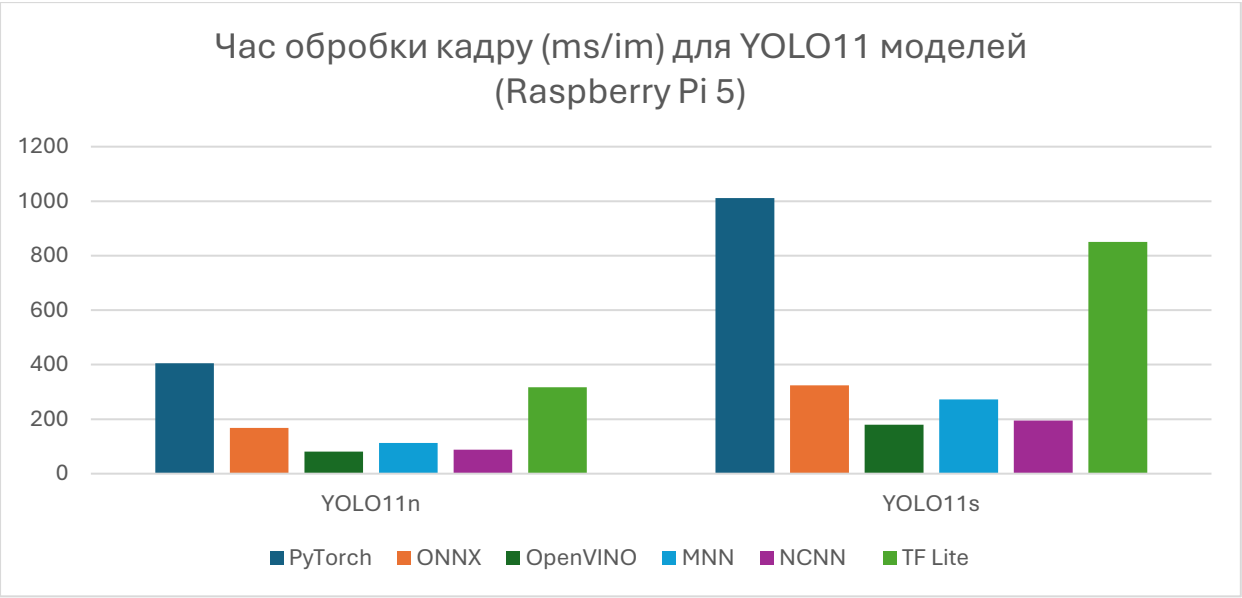

Рисунок 1.4 – Продуктивність YOLO на Raspberry Pi5

1.1.4 Автономні рої дронів

Автономні рої дронів представляють передову концепцію в робототехніці, де кілька безпілотних літальних апаратів працюють спільно для досягнення поставленої задачі. Рої використовують децентралізовані системи контролю та інтелект роїв: замість того, щоб покладатися на централізований контролер, окремі дрони в групі спілкуються за допомогою протоколів, таких як MavLink для передачі повідомлень, або рішень на основі Wi-Fi, таких як WFB-ng для взаємодії з низькою затримкою.

Кожен БПЛА виконує локальні обчислення та обмінюється ключовою інформацією, що дозволяє групі функціонувати злагоджено навіть у динамічному середовищі. Популярні моделі координації роїв включають системи «лідер-послідовник», моделі на основі поведінки та підходи до навчання з підкріпленням, які забезпечують ефективне прийняття рішень та адаптивну реакцію на зміни умов.

Групи безпілотниікв можна класифікувати як напівавтономні або повністю автономні. У свою чергу рої можливо класифікувати як однорівненві та багаторівненві [3, 4]. В одношарових роях кожен БПЛА є лідером і приймає рішення самостійно, у подібних системах зазвичай використовується лише один тип безпілотників. В багатошарових роях кожен дрони розподіляються на підгрупи де кожна група дронів спеціалізується на виконанні окремого завдання. Подібні групи працюють і звітують перед конкретними дронами лідерами на кожному рівні.

Використання роїв безпілотників дає значні переваги в порівнянні з розгортанням одного дрона. Рої за своєю суттю є масштабованими та адаптованими, що робить їх ідеальними для великомасштабних застосувань, таких як реагування на стихійні лиха, сільськогосподарський моніторинг та перевірки інфраструктури, де потрібне широке покриття. Наприклад, під час ліквідації наслідків стихійних лих кілька дронів можуть

одночасно наносити на карту постраждалі території, знаходити тих, хто вижив, і доставляти допомогу швидше, ніж один дрон. Подібним чином у сільському господарстві зграї можуть стежити за посівами на величезних полях, застосовуючи добрива чи пестициди з більшою точністю.

На відміну від цього, розгортання одного дрона часто достатньо для локальних завдань, таких як перевірка окремої вітряної турбіни, зйомка конкретних об’єктів нерухомості або доставка невеликих посилок. Рої найкраще підходять для програм, які вимагають розподіленого робочого навантаження, одночасного збору даних у великих регіонах або резервування для врахування можливих збоїв.

Однак розгортання та експлуатація роїв безпілотників супроводжується значними технічними труднощами. Зв'язок є критично важливим аспектом, оскільки безпілотники потребують обміну даними в режимі реального часу під час роботи в середовищах з обмеженою пропускною здатністю.

Енергоефективність не менш важлива; щоб рій ефективно функціонував, усі дрони повинні підтримувати достатній рівень потужності, що може бути складним у довготривалих місіях. Координація розподілу завдань і забезпечення синхронізації рухів між безпілотниками вимагають складних моделей прийняття рішень, які часто базуються на підкріпленні навчання або евристичних підходах. Нарешті, зграї стикаються з такими вразливими місцями, як окремі точки збою в мережах зв’язку, сприйнятливість до електромагнітних перешкод і проблеми в управлінні загальносистемними оновленнями. Вирішення цих проблем потребує постійного вдосконалення ШІ, робототехніки та комунікаційних технологій, а також ретельного тестування як у змодельованих, так і в реальних середовищах.

1.2 Нейронні мережі

1.2.1 Типи нейронних мереж та методи навчання

Штучні нейронні мережі (ШНМ) – це обчислювальна система, побудована з принципом функціонування біологічних нейронних мереж, які складають мозок тварин або людей. Основною ідеєю нейронних мереж є моделювання процесу обробки інформації в мозку людини, з метою створення систем, здатних до навчання та прийняття рішень на основі вхідних даних.

Ключовими компонентами ШНМ є штучні нейрони, зв'язки між нейронам та ваги для цих зв'язків. Принцип роботи штучної нейронної мережі імітує біологічні мережі та полягає у наступному: кожен нейрон отримує вхідні сигнали від інших нейронів або від зовнішнього середовища, обробляє ці сигнали, застосовуючи функцію активації, і передає результат обробки на вихідні з'єднання. Біологічні нейрони складаються з дендритів - розгалужених відростків нейрону що отримують сигнали, тіла нейрона яке обробляє вхідні сигнали, та аксона мета якого – передати сигнали іншим нейронам. По аналогії, штучні нейрони мають вхідні значення, функцію активації яка імітує процес обробки в тілі нейрона, та вихід аналогічний аксону.

Один з перших та найпростіших видів нейронних мереж – це одношаровий перцептрон, який складається з одного шару нейронів, що приймають вхідні дані і виробляють вихідні результати. Проте для вирішення більш складних завдань використовуються багатошарові перцептрони, які складаються з кількох шарів нейронів, де кожен шар має свою власну функцію активації і ваги зв'язків. Для вирішення більшості задач нейронні мережі складаються з великої кількості нейронів які об'єднані між собою зв'язками з скорегованими значеннями ваг. Основною

метою нейронної мережі є обробка вхідних даних і генерування відповідних вихідних сигналів на основі попереднього навчання.

Принцип роботи нейронних мереж ми можемо описати декількома ключовими етапами. По-перше, кожен нейрон отримує на вхід сигнал: у разі якщо це вхідний шар – сигнали поступають від сенсорів або від будь якого іншого джерела, у всіх інших випадках сигнал отримується від нейронів попереднього шару. Такі сигнали можуть представляти значення пікселів зображення, часовий ряд, значення ознак у наборі даних користувача або будь-якими іншими числовими даними.

Для того, щоб почати використовувати нейромережу першочергово потрібно мережу навчити використовуючи завчасно створений набір даних. Такі набори даних включають у себе приклади вхідних значень, та очікуваний результат. Навчання нейронних мереж – це процес налаштування ваг зв'язків між нейронами на основі вхідних даних та очікуваних результатів.

Розрізняють три парадигми навчання ШНМ, кожна з парадигм використовується для відповідних задач та має власний унікальний підхід до навчання мережі:

- кероване навчання (supervised learning);
- некероване навчання (unsupervised);
- навчання з підкріпленням (reinforcement leaning)

Кероване навчання є найпростішим та широко використовуваним типом машинного навчання. При керованому навчанні модель навчається на великому, структурованому наборі даних де для кожного входу вже надано очікуваний результат. Наприклад, для виявлення різних типів автомобілів строюється набір даних з зображеннями цільових автомобілів з обмежувальними рамками. Під час навчання модель використовує отримані дані для корекції своїх параметрів з метою покращення здатності до генералізації, що дозволяє робити прогнози щодо нових зображень

автомобілів, які не були присутні в тренувальному наборі даних.

Неконтрольоване навчання – це метод навчання, при якому моделі надається набір даних без міток або цільових значень. У цьому випадку модель самостійно виявляє приховані закономірності та структуру в даних. Типовим прикладом неконтрольованого навчання є кластеризація, при якій модель групує схожі точки даних на основі їх характеристик. Наприклад, для набору даних про клієнтів модель може виконувати кластеризацію на основі схожих атрибутів, таких як вік, географічне розташування та поведінкові патерни у витратах.

Навчання з підкріпленням відрізняється від контрольованого та неконтрольованого навчання тим, що модель навчається на основі наслідків своїх дій. У навчанні з підкріпленням агент взаємодіє з середовищем, отримуючи відгуки у вигляді винагороди або покарання залежно від своєї продуктивності. Агент використовує ці відгуки для постійного оновлення алгоритму дій з метою отримання максимальної винагороди. Як приклад, метод навчання з підкріпленням можливо використовувати для навчання автономних дронів, а саме для того, щоб навчити дрон переміщатись до зданих координат у просторі. Класичним прикладом навчання з підкріпленням є навчання моделі грі в шахи або в “хрестики – нулики”, де модель отримує зворотний зв'язок у вигляді виграшу чи програшу, що дозволяє їй коригувати свою стратегію і підвищувати ймовірність успіху у майбутніх іграх.

Слід пам'ятати, всі три типи машинного навчання мають власний унікальний підхід до вирішення задачі та мають бути використані в відповідних програмних рішеннях. Підводячи підсумок, при керованому навчанні модель надається з позначеними даними, некероване навчання дозволяє нам знаходити шаблони в непозначених даних, а при навчанні з підкріпленням модель вчиться на наслідках своїх дій.

Принцип навчання нейронної мережі можливо описати декількома

основними кроками: ініціалізація ваг, пряме поширення сигналу, обчислення похибки та оновлення ваг. На початку навчання всі ваги зв'язків між нейронами встановлюються випадковими значеннями. У деяких випадках можливо використовувати значення ваг з попередньо навчених моделей.

Після ініціалізації запускається процес навчання моделі. Першим кроком є пряме поширення сигналу (forward propagation). Вхідні дані подаються на вхідний шар нейронної мережі, сигнали приймаються прихованими шарами та обробляються відповідно до функції активації та переміщуються до наступного рівня досягаюся вихідного шару, де генеруються результат. Під час прямого розповсюдження попередня активація та активація відбуваються на кожному вузлі прихованого та вихідного рівня нейронної мережі. Функція попередньої активації - це обчислення зваженої суми. Функція активації застосована на основі зваженої суми, щоб зробити потік нейронної мережі нелінійним за допомогою зміщення. При завершенні прямого поширення сигналу обчислюється похибка (error calculation). Похибка, або втрата, обчислюється як різниця між передбаченими значеннями та фактичними, цільовими значеннями. Функція втрат кількісно визначає, наскільки далекі прогнози від справжніх значень.

Після обчислення похибки ми маємо оновити значення ваг для кожного нейрона. Для цього використовується алгоритм зворотнього поширення помилки (backpropagation). Метою зворотного розповсюдження є коригування вагових коефіцієнтів і зміщень у всій нейронній мережі на основі обчисленої вартості, щоб вартість була нижчою в наступній ітерації. Тобто алгоритм змінює ваги для того, щоб мінімізувати витрати. Мета полягає в тому, щоб бути ближче до локального мінімуму після кожного кроку. Ми робимо це, знаходячи градієнт функції витрат використовуючи похідну від вартості відносно ваг. Зворотне розповсюдження – це процес

обчислення похідних, а градієнтний спад – це процес спаду через градієнт, тобто коригування параметрів моделі для переходу через функцію втрат. У градієнтному спуску досягається мінімум функції втрат щодо параметрів, використовуючи похідні, обчислені у зворотному поширенні.

Процес навчання триває до тих пір, поки продуктивність нейронної мережі на навчальних даних не стане задовільною. Потім мережа оцінюється на основі окремого, тестового набору даних. При необхідності параметри нейронної мережі змінюються і мережа швидкість навчання, архітектура тощо) можуть бути скориговані на основі продуктивності набору перевірки. Очікується, що після завершення процесу навчання нейронна мережа вивчить базові закономірності та зв’язки в навчальних даних, що дозволить їй робити точні прогнози щодо нових, невідомих даних.

Важливо відзначити, що нейронні мережі навчаються шляхом ітеративного коригування своїх параметрів на основі градієнтів функції втрат, і цей процес часто потребує обчислень, особливо для великих і глибоких мереж. Отже, навчання нейронних мереж зазвичай виконується на потужному апаратному забезпеченні або спеціалізованих апаратних прискорювачах (GPU або TPU), щоб прискорити процес.

1.2.2 Згорткові нейронні мережі

Людський мозок обробляє величезні обсяги інформації миттєво при сприйнятті зображень. Кожен нейрон функціонує у своєму рецептивному полі та взаємодіє з іншими нейронами забезпечуючи повне охоплення поля зору. Подібно до того, як кожен нейрон реагує на стимули лише в обмеженій області поля зору, яка називається рецептивним полем у системі біологічного зору, кожен нейрон у згорткові нейромережі обробляє дані лише у своєму рецептивному полі. Згорткові нейромережі організовані

таким чином, щоб спочатку виявляти прості ознаки, такі як кути, лінії або криві, та з кожним наступним рівнем більш складніші структури. Використання згорткові нейромережі ми можемо імітувати зір людини та розробити системи комп'ютерного зору. Приклад структури згорткової нейронної мережі зображено на рисунку 1.5.

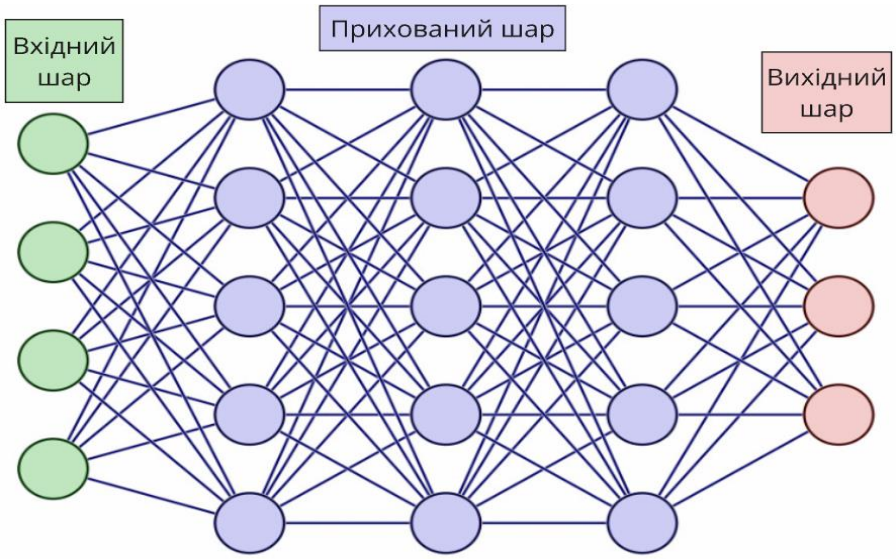

Рисунок 1.5 – Структура згорткової нейронної мережі

Згорткові нейронні мережі належать до класу нейронних мереж прямого поширення. У нейронних мережах прямого поширення нейрони організовані в групи, які називаються шарами.

Зв’язки між нейронами структуровані таким чином, що вхідний шар отримує зовнішній сигнал, обробляє його і передає вихідні значення своїх нейронів наступному шару. У межах кожного шару відсутні внутрішні зв’язки між нейронами, і нейрони кожного наступного шару приймають значення з виходів нейронів попереднього шару. Останній шар мережі називається вихідним шаром, і його вихідні значення формують загальний вихідний сигнал нейронної мережі. Згорткова нейронна мережа, також відома як CNN, є спеціалізованим типом алгоритму глибокого навчання який спеціалізуються на обробці даних з сітчастою топологію, наприклад

зображення, відео або дані з лідару. Зазначені мережі складаються з кількох рівнів, а саме: згортковий рівень, рівень об’єднання та повністю зв’язаний рівень. Згортковий рівень застосовує фільтри до вхідного зображення, щоб виділити особливості, рівень об’єднання зменшує розмірність зображення, щоб зменшити обчислення, а повно-зв'язаний рівень робить остаточний прогноз. Мережа вивчає оптимальні фільтри за допомогою зворотного поширення та градієнтного спуску.

Згортковий рівень є основним блоком згорткової нейромережі оскільки на ньому відбувається більшість обчислень. . Згорткові шари використовують кілька ключових елементів: вхідні дані, фільтри та карти ознак. Припустімо, що вхідними даними є кольорове зображення, представлене як тривимірна матриця пікселів. Це означає, що вхідні дані мають три виміри – висоту, ширину та глибину – відповідні компонентам RGB зображення. Фільтр, або ядро, виконує роль детектора ознак і переміщується по рецептивним полям зображення, перевіряючи наявність певних ознак. Цей процес називається згорткою.

Детектор ознак – це двовимірний масив вагових коефіцієнтів, який представляє частину зображення. Розмір фільтра визначаєтья від параметрів вхідного зображення, проте самий поширений варіант це розмір 3x3. Розмір фільтру визначає розмір рецептивного поля. В процесі навчання завчасно створені фільтри застосовується до області зображення, і обчислюється скалярний добуток між вхідними пікселями та фільтром.

Отриманий скалярний добуток подається у вихідний масив. Після цього фільтр поступово зміщується по всьому зображенню, повторюючи процес, доки ядро не охопить увесь простір. Кінцевий результат серії скалярних добутків між вхідними даними та фільтром називається картою ознак, картою активації або згорнутою функцією. Після кожної операції згортки CNN застосовує перетворення Rectified Linear Unit (ReLU) до карти ознак, вносячи нелінійність у модель. Функція активації ReLU

застосовується після кожної операції згортки. Ця функція допомагає мережі вивчати нелінійні зв’язки між об’єктами на зображенні, таким чином роблячи мережу більш надійною для визначення різних шаблонів. Це також допомагає пом’якшити проблеми зникнення градієнта.

Шари підвибірки (pooling layer), також відомі як зменшення дискретизації, виконують зменшення розмірності, зменшуючи кількість параметрів у вхідних даних. Подібно до згорткового шару, операція підвибірки переміщує фільтр по всьому входу, але на відміну від згорткового шару, цей фільтр не має вагових коефіцієнтів. Замість цього ядро застосовує функцію агрегації до значень у рецептивному полі, заповнюючи вихідний масив. Існує два основних типи підвибірки: середня та максимальна.

Максимальна підвибірка вибирає піксель із максимальним значенням для передачі у вихідний масив. Цей підхід зазвичай використовується частіше, ніж середня підвибірка. Середнє підвибірка під час переміщення фільтра по вхідних даних він обчислює середнє значення в рецептивному полі для передачі у вихідний масив. Незважаючи на те, що на етапі вибірки втрачається певна кількість інформації, цей процес має ряд переваг для CNN. Він допомагає зменшити складність моделі, підвищити обчислювальну ефективність і обмежити ризик перенавчання.

Останній, повнозв’язний шар використовується для генерації результату. Як згадувалося раніше, значення пікселів вхідного зображення не пов’язані безпосередньо з вихідним шаром у частково з’єднаних шарах. Однак на повністю підключеному рівні кожен вузол вихідного рівня з’єднується безпосередньо з вузлом попереднього рівня. Цей рівень виконує завдання класифікації на основі ознак, витягнутих через попередні шари та їхні різні фільтри.

1.3 Комп’ютерний зір

1.3.1 Загальні відомості

Комп'ютерний зір – це галузь штучного інтелекту яка займається розробкою систем здатних аналізувати та отримувати значущу інформацію з цифрових зображень, відео та інших візуальних даних. Сучасні методи для вирішення більшості задач використовують нейроні мережі та методи глибокого навчання на великих обсягах даних для того, щоб навчити комп'ютерні системи “бачити”.

Ідея комп'ютерного зору заснована на ідеї імітації людського зору та його здатності до обробки і розуміння візуальної інформації. Людське око і мозок працюють разом, щоб сприймати, інтерпретувати і реагувати на навколишній світ. Люди можуть розпізнавати об'єкти, оцінювати відстані, розрізняти кольори і форми, а також розуміти контекст сцени завдяки складним когнітивним процесам.

Комп'ютерний зір прагне відтворити ці можливості в комп'ютерних системах. Використовуючи камери для отримання зображень і відео, а також алгоритми для їх обробки та аналізу, комп'ютерний зір намагається імітувати людську здатність до зорового сприйняття. Алгоритми глибинного навчання, зокрема згорткові нейронні мережі, побудовані за принципом роботи людського мозку, допомагають комп'ютерним системам навчатися розпізнавати та класифікувати об'єкти, аналізувати сцени і приймати рішення на основі візуальної інформації. Таким чином, основна концепція комп'ютерного зору полягає в перенесенні принципів роботи людського зору в програмні рішення, дозволяючи їм автоматично обробляти і інтерпретувати візуальні дані для різних застосувань.

Комп'ютерний зір стрімко розвивається завдяки швидкому росту обчислювальних можливостей, особливо завдяки досягненням у глибокому

навчанні. З кожним новим роком удосконалюються алгоритми та моделі виявлення та відстеження об'єктів, з'являються покращенні рішення типових задач у даній галузі. Особливо важливим фактором розвитку даної галузі виявляється використання графічних процесорів (GPU) та спеціалізованих прискорювачів штучного інтелекту, таких як тензорних процесорів (TPU), які значно прискорюють обчислення великих об'ємів даних, що є невід'ємною частиною обробки зображень та відео. З розвитком апаратної частини та з сучасними моделями з'явилась можливість застосовувати комп'ютерний зір в різних галузях, включаючи безпілотні машини та автономні безпілотні літальні апарати.

Варто також зазначити що комп'ютерний зір, обробка та аналіз зображень, а також машинний зір – тісно пов'язані терміни в одній широкій галузі. Термін “Комп'ютерний зір” означає широку галузь, що охоплює всі аспекти отримання, обробки, аналізу та розуміння візуальної інформації. Обробка та аналіз зображень являє собою підгалузь комп'ютерного зору, що зосереджується на техніках вилучення корисної інформації з зображень.

Машинний зір – це напрямок комп'ютерного зору орієнтований на промислове застосування комп'ютерного зору, тобто для автономних дронів та роботів, автоматизованих виробничих процесів, тощо. Обробка зображень в контексті машинного зору може виконуватись як програмно, так і апаратно, здебільшого в реальному часі. Програмна обробка використовує алгоритми і програмні бібліотеки, тоді як апаратна обробка покладається на спеціалізовані пристрої, такі як графічні або тензорні процесори або програмовані вентильні матриці (FPGA), для забезпечення високої продуктивності та швидкості.

Галузь комп'ютерного зору охоплює вирішення багатьох задач, кожна з яких має свої специфічні цілі та області застосування (рисунок 1.6). Базові задачі в комп'ютерному зорі включають у себе:

- Класифікація зображень (images classification);

- Сегментація зображення (images segmentation);
- Виявлення та локалізація об’єктів (object detection);

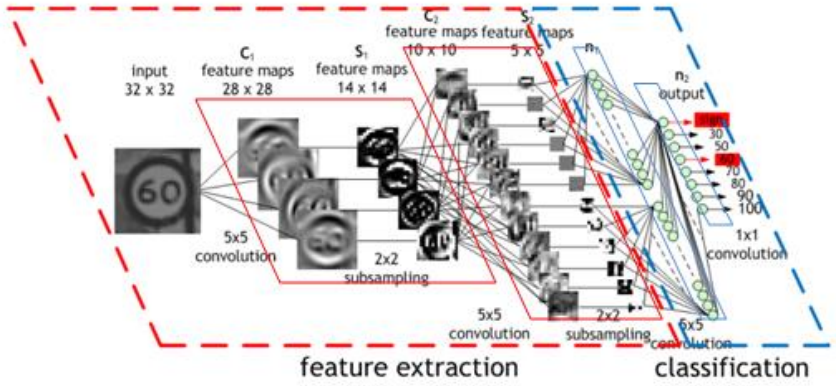

Рисунок 1.6 – Приклад нейромережі для класифікації об’єкта

1.3.2 Виявлення та повторна ідентифікація об'єктів

Відстеження об’єктів – це множина бібліотек комп’ютерного зору, який працює з відеопотоками у реальному часі. Задачу трекінгу можна розглядати як розширення алгоритмів виявлення об’єктів, за допомогою якої один або багато об’єктів виявляються та у подальшому відстежуються в послідовності зображень. Після виявлення кожній цілі присвоюється свій унікальний ідентифікатор. Часто навколо об’єкта який відстежується, є позначка з обмежувальною рамкою що показує користувачеві де знаходиться об’єкт у кожний момент часу.

Основна мета відстеження об’єктів – визначити траєкторію руху об’єкта, зберігаючи його ідентичність з кадру в кадр. Це завдання є критично важливим для багатьох застосувань, таких як відеоспостереження, автоматизовані системи безпеки, автономні транспортні засоби та робототехніка.

Відстеження об’єктів вирішує кілька важливих проблем. По-перше, воно дозволяє виявляти та аналізувати поведінку об’єктів у динамічному

середовищі, що є необхідним для прогнозування їх майбутніх дій і прийняття відповідних рішень. По-друге, забезпечує інтеграцію та синхронізацію візуальної інформації з інших сенсорів, покращуючи загальну точність системи. По-третє, ця технологія дозволяє розпізнавати та розрізняти окремі об’єкти серед багатьох інших, навіть у складних умовах, таких як зміна освітлення, часткове перекриття об’єктів та їх деформація.

Розрізнять два методи відстеження об'єктів: відстеження однієї цілі та відстеження багатьох цілей. При відстеженні багатьох об'єктів комп'ютерна система повинна одночасно стежити за кількома об'єктами, що можуть взаємодіяти або перекривати один одного. Це ще більше ускладнює задачу, але й відкриває можливості для створення більш інтерактивних та інтелектуальних систем.

Відстеження об’єктів є основою для багатьох передових технологій, забезпечуючи надійне та ефективне функціонування систем, що покладаються на аналіз візуальної інформації в реальному часі. Незважаючи на широкий спектр використання алгоритмів відстеження цілей, існує перелік проблем про які слід пам'ятати:

- Продуктивність та ресурсозатратність алгоритмів;
- Швидкість відстеження;
- Наявність перешкод та шумів;
- Повторна ідентифікація об'єктів.

Відстеження об’єктів включає саме виявлення, класифікацію та локалізацію всіх об’єктів на зображені, тому створено багато різних трекерів які мають бути в різних випадках. Наприклад центроїдий трекер (centroid tracker) та звичайний трекер на основі розширеного фільтру Калмана досить легкі та швидкі, можуть бути використані для обробки відео у реальному часі. Деякі трекери мають обмеження на максимальну кількість об’єктів які можливо відстежувати. Наприклад, кореляційний трекер стає непридатним або надто повільним, при відстеженні двох або більше

об’єктів. Також слід пам'ятати що трекери основані на нейронних мережах потребують наявність графічних або тензорних процесорів на фізичному пристрої.

Швидкість відстеження та частота кадрів грають велику роль для систем які працюють в реальному часі. Швидкість реакції системи залежить від швидкості обробки кожного з кадрів. Для пришвидшення роботи системи використовують кадри меншої роздільної здатності, використовують апаратні прискорювачі, а також налаштовують трекерів. Наприклад, кількість кадрів в секунду секунду (FPS) мають великий вплив на відстеження об’єктів. Якщо частота кадрів становить 10 FPS, а автомобіль їде по шосе з великою швидкістю, ми очікуємо, що координати автомобіля будуть значно відрізнятись з кожним новим кадром. Якщо той самий автомобіль буде відстежуватись зі швидкістю 40 або 60 FPS, координати, а точніше обмежувальні рамки будуть перетинатись набагато частіше.

Також на відео часто можуть виникати перешкоди, об’єкт може може різко змінити напрямок або змішуватись з іншими об’єктами або фоном. В такі моменти алгоритми відстеження можуть втратити об'єкт або переплутати ціль з іншим об'єктом. Перешкоди призводить до того, що спочатку відстежуваний об’єкт повторно ідентифікується як новий об’єкт. Додатково алгоритми мають бути стійкими до шумів, зміни яркості та безладу на фоні. Розмиті або однокольорові фони дозволяють трекерам легше виявляти та відстежувати об’єкти у той час як переповнені фони зазвичай спричиняють шум та вносять зайву інформацію.

Для того щоб один об'єкт завжди мав свою унікальну мітку часто використовуються методи повторної ідентифікації об'єктів (re-ID). Під час відстеження об’єкта алгоритм стежить за рухом об’єкта та намагається оцінити або передбачити його положення на відео. У той час, повторна ідентифікація дає нам змогу ідентифікувати один і той самий об’єкт

протягом відеопослідовності. Повторна ідентифікація дає змогу знаходити об’єкти, навіть якщо вони відсутні в кількох послідовних кадрах, або потрапляють у нові зони. Більшість алгоритмів відстеження не зберігає даних про об'єкт, а лише передбачує наступну позицію враховуючи швидкість та напрямок руху об'єкта.

Повторна ідентифікація потребує детального розпізнавання та створення окремої галереї для кожного виявленого об'єкта. У разі виявляння нового об'єкту на відео, цей об'єкт першочергово порівнюється з збереженими даними, та у разі якщо виявляються збіжність унікальних ознак, унікальний ідентифікатор відтворюється та заново присвоюється об'єкту. Сучасні алгоритми відстеження не зовсім здатні виконати повторну ідентифікацію, оскільки не зберігають дані про виявлені об'єкти, або вхідні зображення можуть бути розмиті, або стають розмитими в процесі оброблення кадру. Також існуючі методи повторної ідентифікації шукають найкращу відповідність візуальних характеристик цільового об’єкта в межах набору кадрів. Зважаючи на природу проблеми, покладаючись лише на візуальний вигляд екземплярів об’єктів створюється багато хибних збігів, особливо коли на сцені є кілька об’єктів подібного вигляду або кілька екземплярів одного класу об’єктів. Як результат, зазначений напрямок залишається відкритим до досліджень.

Один з методів вирішення проблеми повторної ідентифікації полягає у процесі асоціації даних. Загалом, популярна парадигма для вирішення задачі відстеження декількох об'єктів складається з двох кроків: виявлення об’єктів у кожному кадрі відео та асоціювання виявлень, які відповідають тому самому об’єкту. Згідно з цією парадигмою, відстеження декількох об'єктів можливо розглядати як проблему асоціації даних.

Асоціація даних – це процес зіставлення об'єктів, виявлених на послідовних кадрах відеопотоку, щоб зберегти їх ідентичність та траєкторії руху. Основна мета асоціації даних правильно прив'язати спостереження

об'єктів у новому кадрі до відповідних об'єктів, виявлених у попередніх кадрах, незважаючи на можливі зміни їх зовнішнього вигляду, положення або перекриття іншими об'єктами. Для вирішення цієї проблеми припускається, що один об’єкт може генерувати не більше одного виявлення, тобто однієї обмежувальної рамки. Також слід бути пам'ятати що нові виявлення можуть бути згенеровані раніше невиявленими цілями, тому слід бути обережним, щоб помилково не призначити одне з цих виявлень вже існуючому треку.

1.3.3 Багатокамерні системи відстеження об'єктів

Використання декількох камер для відстеження об'єктів підвищує стійкість системи, проте потребує більших обчислювальних потужностей, та більш надійних методів повторної ідентифікації об'єктів та алгоритмів передбачення траєкторій руху. Незважаючи на те, що візуальне відстеження об’єктів за допомогою однієї камери привернуло величезну увагу дослідників, багатокамерні системи та повторна ідентифікація та відстеження цілей залишається відкритою проблемою для дослідження.

Багатокамерні системи відстеження декількох об'єктів (MCMOT), на відміну від програмних рішень відстеження об’єктів однією камерою, вимагають координації даних із декількох джерел, для того щоб точно відстежувати об’єкти під час їхнього переміщення між різними ділянками місцевості. З появою недорогих високоякісних камер з високою частотою кадрів і зростанням обчислювальної потужності одноплатних комп'ютерів попит на розгортання багатокамерних мереж значно зростає. Як результат, зростає попит на певний тип алгоритмів відстеження, відомий як «багатокамерні трекери». Подібні трекери розробляються для того, щоб впоратися зі складністю відстеження цілей із використанням багатьох камер у реальному часі.

Перевага багатокамерних мереж полягає можливості досягнути повного покриття контрольованої області та отримати щільність просторово-часової вибірки, яка перевищує те, що може запропонувати одна камера. Таким чином, багатокамерні системи, приклад на рисунку 1.7, відстеження незабаром стануть незамінним інструментом в сферах де надійність, точність та масштабованість є першорядними критеріями.

Рисунок 1.7 – Приклад багатокамерної сисетми відстеження транспортних засобів

Крім того, мережі з декількох камер дозволять поєднувати та обробляти дані з різних типів камер. Незважаючи на те, що мережі з декількома камерами пропонують значні переваги, стеження за цілями в таких системах є більш складним завданням, ніж стеження за допомогою однієї камери. Для створення багатокамерної системи першочергово слід вирішити такі проблеми, як калібрування та синхронізація різних камер, виявлення унікальних ознак об'єктів, створення трекерів стійких до змін зовнішнього вигляду об’єктів та інші. Зазначені проблеми вимагають передових технологій комп’ютерного зору та розроблення нових алгоритмів

для забезпечення точного та надійного відстеження об’єктів.

Оскільки багатокамерне відстеження об'єктів все ще є відносно новим дослідницьким напрямком, при формулюванні проблеми відстеження кількох об’єктів за допомогою кількох камер необхідно враховувати наступні параметри системи та проблеми які можуть виникнути: динамічність зовнішнього вигляду, непередбачуваний рух об’єктів, коефіцієнт перекриття камер, рух камер та найважливіше – асоціація даних.

Динамічність зовнішнього вигляду – в залежності від ракурсу, освітленості та інших факторів один і той самий об'єкт може мати різний вигляд. Для того, щоб відстежувати об'єкт по візуальним ознакам необхідно витягнути дані з цілі для асоціації. Такими даними можуть бути краї або кути, градієнти, траєкторія руху та інші характеристики. Однак варіативність зовнішнього вигляду, яка може виникати через зміни освітленості, оклюзії та перешкоди, тіні, артефакти на камері та інші проблеми – все це може призвести до того, що той самий об’єкт виглядатиме по-різному.

Асоціація даних у додатках відстеження декількох об'єктів з однією камерою вже є складною задачею, але вона стає ще більш складною в багатокамерних системах через динамічність зовнішнього вигляду об'єктів. Вектори ознак можуть суттєво змінюватися, наприклад коли ціль переміщується з внутрішнього місця на вулицю, проте більшість детекторів об’єктів і моделей повторної ідентифікації можуть не враховувати подібні зміни. Це призводить до порушення зв’язку між представленням траєкторії об’єкта та його попередніми векторами ознак, що призводить до неправильного призначення ідентифікаційного номеру. Для системи відстеження збої в асоціації та зв’язків даних у певному кадрі можуть мати довгострокові негативні наслідки. Таким чином, асоціація даних залишається важливою частиною в оцінці ефективності подібних систем.

Коефіцієнт перекриття. Набори даних із кількох камер переважно

мають можливість записувати два типи сценаріїв. Перший варіант – різні камери фокусуються на одній сцені, тим самим поля огляду камер частково або повністю перекривають одне одного. Другий варіант – набор даних складається із зображень із камер, які майже не перекриваються або взагалі не перекриваються. Як результат, коефіцієнт перекриття камер впливає на формулювання проблем відстеження в системах з декількома камерами, і, таким чином, впливає на загальну продуктивність кінцевої моделі відстеження.

Рух камери та рух об’єктів також значно впливають на параметри алгоритмів відстеження та продуктивність системи. Відстежувати та пов’язувати різні види однієї цілі на рухомих камерах у реальному часі є важкою задачею, оскільки зовнішній вигляд виявленої цілі, ракурс та масштаб можуть сильно відрізнятись. Додатково слід враховувати траєкторію руху об'єкта, передбачення траєкторії може допомогти в процесі переідентифікації цілі та асоціації даних. Враховуючи перелічені фактори, можна класифікувати системи відстеження наступним чином:

- Одна статична камера;
- Одна камера що рухається;
- Декілька статичних камер з великим коефіцієнтом перекриття
- Декілька статичних камер з низьким коефіцієнтом перекриття
- Декілька камер що рухаються.

Використання однієї статичної камери є найпростішим варіантом для відстеження об'єктів, оскільки подібна система потребує мінімальних налаштувань найпростіша в реалізації, проте має багато обмежень. Однак, з кожним наступним кроком, коли додаються рухомі камери або збільшується кількість камер, задача відстеження об'єктів стає складнішою, вимагаючи більшого обсягу обчислювальних ресурсів та оптимізованих алгоритмів асоціації даних та повторної ідентифікації цілей.

1.4 Постановка задачі

Для вирішити поставленої задачі, а саме відстеження кількох об’єктів у динамічних середовищах за допомогобю дронів необхідно виконати перелік кроків:

- Аналіз існуючих метрик для оцінки продуктивності;
- Аналіз архітектур сучасних моделей та алгоритмів;
- Розробка оптимізованої системи відстеження об`єктів;
- Інтеграція в отриману систему автопілота та тестування.

Перший крок включає визначення та аналіз метрик, важливих для оцінки продуктивності систем виявлення об’єктів, виділення ознак і відстеження. Для виявлення об’єктів необхідно визначити такі показники, як середня середня точність, відкликання, перетин через з’єднання та інші. Для відстеження виявлених об’єктів необхідно вивчити такі метрики, як точність відстеження кількох об’єктів (MOTA), оцінку F1 (IDF1), перемикачі ідентифікаторів (IDSW), та інші предсталені в роботі [8]. Також для вирішення задачі асоціації даних на основі ззовнішнього вигляду цілі необхідно дослідити такі дискримінаційі показники, як косинусуідальна та евклідова відстані.

Дослідження також має включати аналіз сучасних моделей та архітектур. Необхідно порівняти найсучасніші детектори об’єктів, такі як YOLOv11, Faster R-CNN і SSD, а також моделі вилучення функцій, такі як OSNet [22], ResNet [23] і MobileNet. Рішення для відстеження, включно з SORT, DeepSORT та FeatureSORT, досліджуються, щоб визначити їхні сильні та слабкі сторони. Подібний аналіз надасть можливість обрати найкращі компоненти які будуть придатними для інтеграції в трекер, оптимізований для застосування БПЛА.

Наступна мета – навчання моделей виявлення об`єктів. В ході роботи очікується навчити YOLOv11 модель: нано; маленьку та середню варіачії на

наборі даних, що містить зображення з UAV. Кожну модель необхідно експортати в кілька форматів, включаючи PyTorch, TorchScript, ONNX і TensorRT (з точністю FP32, FP16 і INT8), та провести порівняння.

Моделі вилучення унікальних ознак, такі як OSNet та ResNet, навчити на таких наборах даних, як VERI та Market-1501, щоб забезпечити надійні можливості повторної ідентифікації транспортних засобів і людей. Зазначені моделі необхідно порівняти та обрати найкращі варіації.

Після навчання усіх необхідних моделей необхідно розробити власний трекер, що буде поєднувати переваги існуючих трекерів. Інтеграція з автопілотом PX4 є критично важливим компонентом дослідження. Необхідно розробити ROS2 компонент для обробки вхідних даних камери, їх передачу трекеру.

Розроблений трекер включно з PX4 автопілотом необхіждно протестувати за допомогою симулятора Gazebo, де для оцінки ефективності використовуються сценарії з одним або кількома транспортними засобами. Симуляції мають перевірити здатність системи справлятися з різноманітними проблемами, такими як динамічне середовище, оклюзії та різна щільність об’єктів. Ці тести нададуть важливе уявлення про надійність трекера та його придатність для реальних додатків.

2. АНАЛІЗ СУЧАСНИХ МОДЕЛЕЙ ТА АЛГОРИТМІВ ДЛЯ ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ ЦІЛЕЙ

2.1 Метрики для тестування та аналізу детекторів

Щоб ефективно навчати та оцінювати продуктивність моделей виявлення об’єктів розроблено комплексний набір стандартизованих показників. Ці показники дають змогу кількісно визначати точність і ефективність моделей, гарантуючи що отримані моделі відповідають встановленим вимогам, та можуть бути інтегровані в кінцеві системи та додатки. Для визначення показників ефективності моделей для виявлення та локалізації цілей в кадрі, створено перелік базових метрик:

- Точність (англ. precision) та відклик (англ. recall);
- Перетен через обєднання (англ. intersection over union, IoU);
- Середня точніть (англ. average precision, AP);
- Розширина середня точніть (англ. mean average precision, mAP);
- Оцінка F1 (англ. F1 Score);

Перетин через обєднання (IoU) вимірює перекриття між прогнозованою обмежувальною рамкою та відомою обмежувальною. Дана метрика, (формула 2.1), оцінює точність локалізації.

$$IoU = \frac{|\, B_p \cap B_{gt} \,|}{|\, B_p \cup B_{gt} \,|} \tag{2.1}$$

Де B_p – координати отриманої обмежувальної рамки;

B_{gt} – координати відомої “ground truth” обмежувальної рамки;

$|\, B_p \cap B_{gt} \,|$ – це зона перетину обох обмежувальних рамок;

$|\, B_p \cup B_{gt} \,|$ – це загальна площа, охоплена двома рамками.

Точність (P) визначає частку істинно позитивних прогнозів серед усіх позитивних прогнозів, формула (2.2); Відклик (R) вимірює частку правдивих позитивних прогнозів серед усіх фактичних позитивних результатів, формула (2.3).

$$P = \frac{TP}{TP+FP}, \tag{2.2}$$

$$R = \frac{TP}{TP+FN}, \tag{2.3}$$

де *TP* – кількість правдивих позитивних прогнозів;

FP – кількість неправдивих позитивних прогнозів;

FN – кількість неправдивих негативних прогнозів.

Крива Precision-Recall візуалізує точність як функцію запам’ятовування при різних порогових значеннях, що має вирішальне значення для оцінки продуктивності в незбалансованих наборах даних. Він дає уявлення про здатність моделі підтримувати точність, досягаючи кращого запам’ятовування, допомагаючи оптимізувати можливості виявлення. Крива точності (рисунок 2.1) ілюструє зміну точності з пороговими налаштуваннями, допомагаючи у виборі порогу для мінімізації хибних спрацьовувань. Це особливо корисно в системах, де висока точність є критичною.

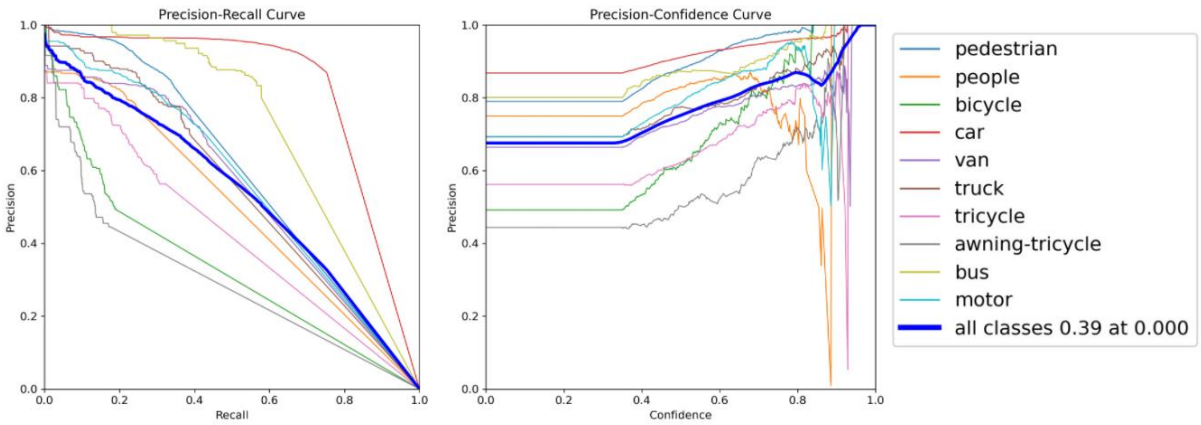

Рисунок 2.1 – Приклад діаграм “PR” та “P” отриманих при тренуванні yolo11m на наборі даних VisDrone

Оцінка F_1 (формула 2.4) поєднує точність та відклик в одну метрику шляхом обчислення їх гармонійного середнього. Зазначена метрика забезпечує збалансоване уявлення про продуктивність моделі, особливо коли набір даних демонструє дисбаланс класів.

$$F_1 = \frac{2}{recall^{-1}+precision^{-1}} = \frac{2TP}{2TP+FP+FN} \tag{2.4}$$

Додатково до самої оцінки, крива F1 ілюструє, як F1 змінюється залежно від різних порогових значень. Аналіз цієї кривої дає цінну інформацію про баланс моделі між хибно-позитивними та хибно-негативними результатами за різних порогів, допомагаючи визначити оптимальний поріг для завдання, приклад кривої зображено на рисунку 2.2.

Метрика «Середня точність» (формула 2.5) – це спосіб узагальнити криву PR в одне значення. Метрика середньої точності є зваженим середнім значенням балів точності, досягнутих на кожному пороговому значенні кривої PR, із збільшенням (Recall) порівняно з використанням попереднього порогового значення

$$AP = \int_{r=0}^{1} P(r)d_r \tag{2.5}$$

Де $P(r)$ – покзник змінення точністі із зміною відклику. формула обчислює інтеграл $P(r)$ в діапазоні значень відкликання від 0 до 1.

Метрика mAP, формула (2.6), узагальнює значення AP для кожного з класів шляхом обчислення середнього точності для всіх класів об’єктів. Цей показник забезпечує цілісну оцінку моделей виявлення в різних сценаріях, забезпечуючи всебічне вимірювання продуктивності всіх класів об’єктів.

$$mAP = \frac{1}{N}\sum_{n=1}^{N} AP(n) \tag{2.6}$$

Де N – кількість класів на якій тестується модель.

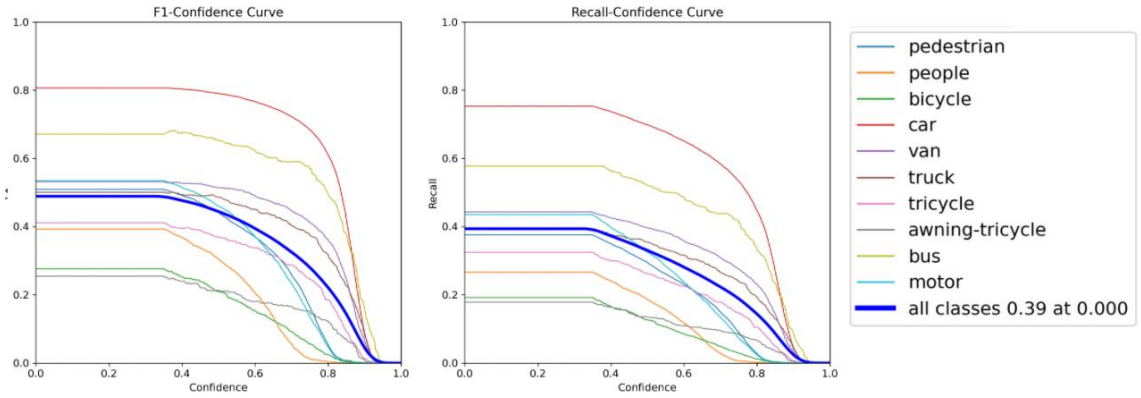

Рисунок 2.2 – Приклад діаграм “F1” та “R” отриманих при тренуванні yolo11m на наборі даних VisDrone

Для візуалізації даних також використовубють “матриці плутанини” (англ. confusion matrix, рисунок 2.3). Нормалізована матриця похибки – це різновид традиційної матриці плутанини, де необроблені дані підрахунку перетворюються на пропорції або відсотки.

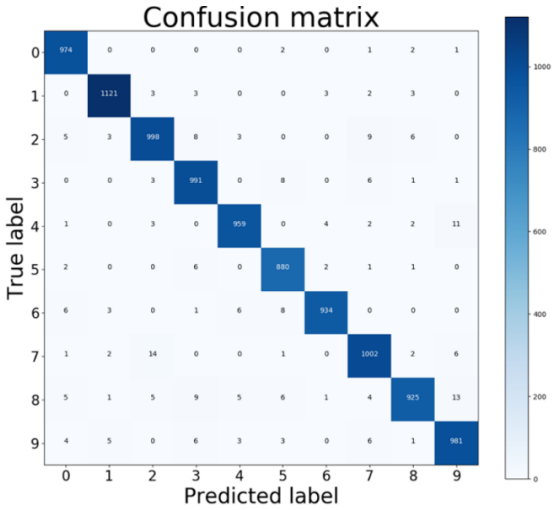

Рисунок 2.3 – Приклад “матриці плутанини”

Нормалізація спрощує порівняння продуктивності моделі в різних класах, незалежно від варіацій частот класів у наборі даних. Кожна комірка нормалізованої матриці представляє відносну частку результатів для певного класу, наприклад частку справжніх позитивних чи хибних негативних результатів.

2.2 Аналіз моделей виявлення об’єктів

2.2.1 YOLO (You Only Look Once)

YOLOv11 (You Only Look Once) [5, 6, 7] представляє сучасну одноетапну модель виявлення об’єктів (ODM), розроблену для додатків у реальному часі. Архітектура YOLO побудована на принципах попередніх моделей YOLO, наголошуючи на швидкості та ефективності без шкоди для точності. YOLOv11 представляє покращену магістраль вилучення функцій із акцентом на ефективність обчислень, покращену такими методами, як CSP (Cross Stage Partial) шари та PANet (Path Aggregation Network) для кращого об’єднання функцій у різних масштабах. Мережа розроблена для прогнозування обмежувальних рамок і ймовірностей класів безпосередньо з карт функцій за один прохід, оптимізуючи наскрізні конвеєри виявлення. Удосконалені механізми без анкерів і покращена функція втрат дозволяють YOLOv11 ефективно обробляти невеликі та перекриваючі об’єкти.

У порівнянні зі своїми попередниками як показано на рисунку 2.4, YOLOv11 демонструє кілька архітектурних і функціональних покращень. Попередні ітерації, такі як YOLOv3, спиралися на магістраль Darknet, яка, незважаючи на ефективність на той час, була перевершена сучасними магістралями, включеними в YOLOv11. Адаптивна генерація прив’язки в YOLOv11 усуває обмеження попередньо визначених прив’язок, які обмежували YOLOv3 і YOLOv4, дозволяючи моделі краще узагальнювати різні набори даних.

Передові методи розширення, включаючи Mosaic і MixUp, сприяють покращенню узагальнення, роблячи YOLOv11 більш надійним для різноманітних сценаріїв реального світу. Крім того, YOLOv11 пропонує масштабованість, пропонуючи кілька варіантів, таких як нано, малий і середній, адаптовані для задоволення різних вимог до обчислень і точності.

Такий рівень масштабованості не був характерним для попередніх моделей, що ще більше вирізняло YOLOv11 як більш універсальну та зручну для розгортання структуру.

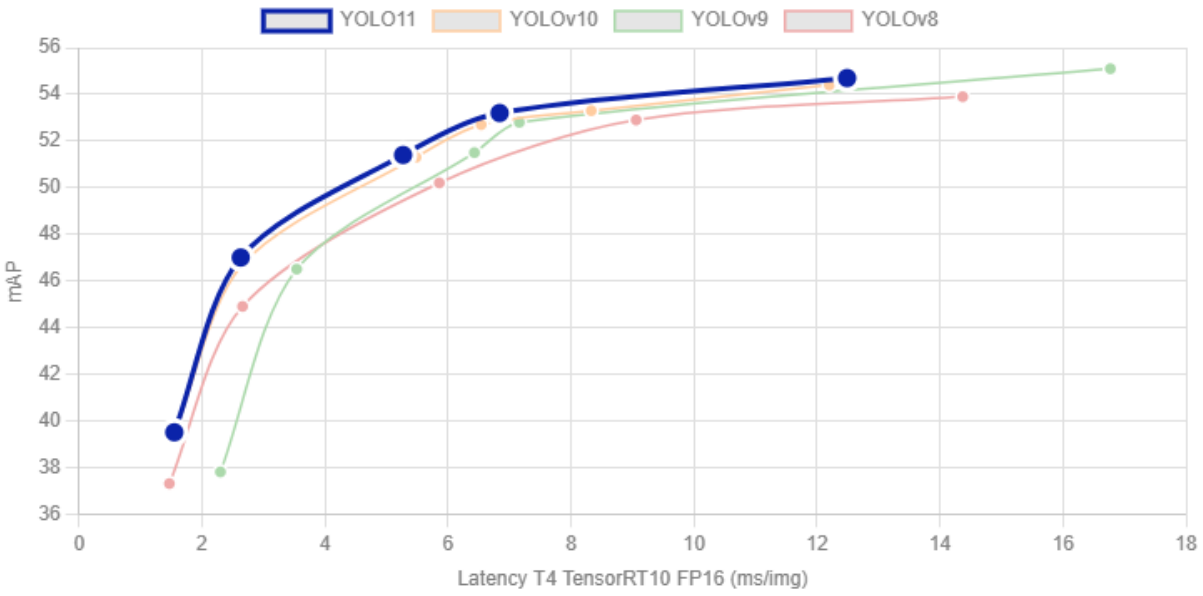

Рисунок 2.4 – Порівняння останніх версій YOLO

YOLOv11 виділяється своєю швидкістю та обчислювальною ефективністю, що робить його ідеальним для додатків у реальному часі на GPU та пристроях з обмеженими ресурсами. Його одноетапна архітектура спрощує конвеєр виявлення, а розширене об’єднання функцій і механізми без закріплення покращують його здатність виявляти невеликі об’єкти або об’єкти, що перекриваються. Масштабованість YOLOv11 із такими варіантами, як нано, малий і середній, ще більше підвищує його адаптивність до різноманітних апаратних і операційних вимог.

Однак YOLOv11 демонструє компроміс між швидкістю та точністю з дещо зниженою точністю порівняно з двоступеневими моделями, як-от Faster R-CNN, особливо в сценах зі щільною об’єктністю. Однопрохідний дизайн мережі може призвести до менш точної локалізації, а продуктивність моделі значною мірою залежить від високоякісних навчальних наборів даних і ефективних доповнень.

2.2.2 SSD (Single Shot MultiBox Detector)

SSD (Single Shot MultiBox Detector) [9], як і моделі YOLO використовують одноетапну архітектуру, призначену для виявлення об’єктів у реальному часі. SSD покладається на піраміду карт функцій для багатомасштабного виявлення, YOLO представляє розширені функції, такі як шари CSP, механізми без прив’язки та розширене об’єднання функцій через PANet. На відміну від YOLO, SSD має залежність від попередньо визначених блоків, що робить його менш конкурентоспроможним у складних сценаріях. Порівняння архітектур зображено на рисунку 2.5.

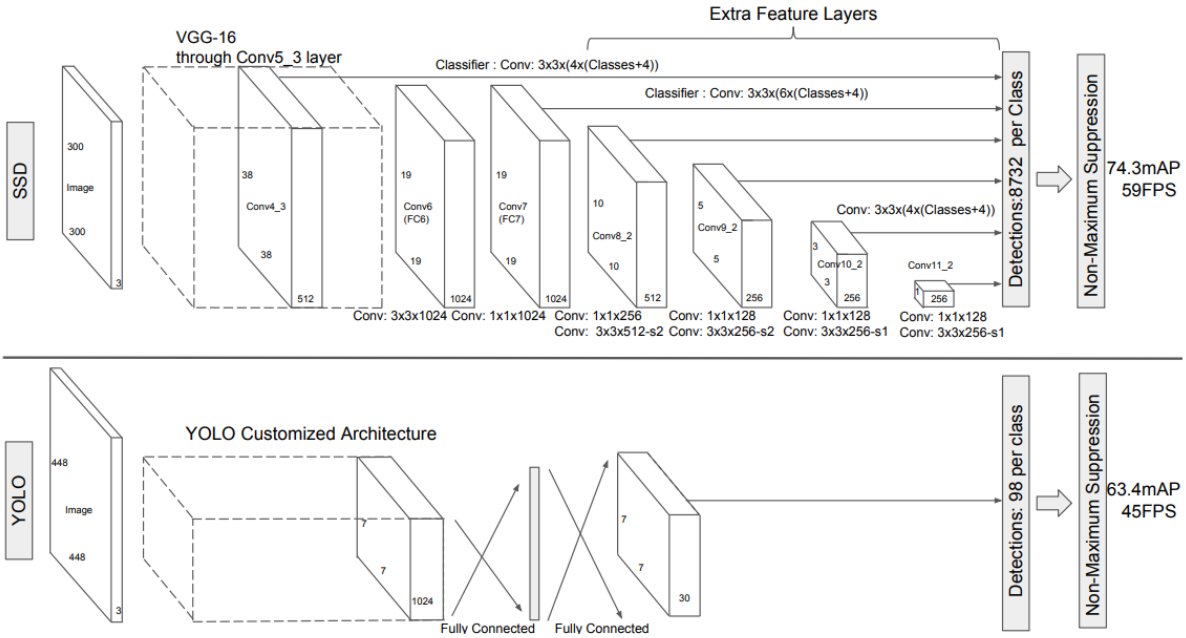

Рисунок 2.5 – Порівняння архітектури SSD з YOLO

Переваги SSD: проста архітектура, подтримує декілька “feature extractors”. Недоліки: погане виявлення дрібних об’єктів, важко виявляти невеликі об’єкти або об’єкти, що перекриваються. Брак функцій, таких як механізми привернення уваги та виявлення без прив’язки, що робить його менш адаптованим до нових викликів.

2.2.3 Faster R-CNN

Faster R-CNN [10, 11] - двоетапна модель виявлення об’єктів, що аикористовує мережу регіональних пропозицій (RPN) для створення регіонів-кандидатів перед виконанням класифікації та уточнення обмежувальної рамки. Подбний дизайн контрастує з новими моделями YOLO, які використовують одноетапну архітектуру, яка безпосередньо передбачає обмежувальні прямокутники та ймовірності класів за один прохід вперед. Модель Faster R-CNN має високу точність, особливо при виявленні невеликих об’єктів або об’єктів, що перекриваються, завдяки спеціальному процесу уточнення регіону. Однак ця точність досягається ціною підвищеної обчислювальної складності та меншої швидкості висновку.

Перевагами Faster R-CNN є точность та надійность. Двоступеневий модель відмінно підходить для сценаріїв, що вимагають точної локалізації та детального виявлення, наприклад, для щільно розташованих об’єктів або малих цілей. RPN дозволяє створювати більш точні пропозиції регіонів, зменшуючи кількість помилкових спрацьовувань і підвищуючи загальну надійність.

Однак Faster R-CNN має помітні недоліки. Через двоетапну архітектуру наявні затримки при наданні результатів, як резульат дана модель є менш придатною для застосунків які мають працюівати в режимі реального часу. Крім того, обчислювальні вимоги моделі вимагають значних апаратних ресурсів, що обмежує її розгортання на периферійних пристроях або в середовищах з обмеженими ресурсами, такими як БПЛА

Як резултьат Faster R-CNN – потужний інструмент для додатків, орієнтованих на точність, але часто перевершує новіші моделі, такі як YOLO, у сценаріях, що вимагають балансу між швидкістю та точністю.

2.3 Метрики для тестування трекерів

Показники відстеження декількох об’єктів (MOT) необхідні для оцінки продуктивності алгоритмів, призначених для виявлення та відстеження кількох об’єктів у відеокадрах. Ці показники забезпечують стандартизовану структуру для оцінки алгоритмічної точності, надійності та ефективності підтримання траєкторій об’єктів та ідентичності з часом.

Розробка цих показників тісно пов’язана з такими ініціативами з порівняльного аналізу, як MOT Challenge, широко визнаною платформою для просування досліджень. Автори MOT20 Challenge надають наступні, ключові метрики:

- Точність відстеження кількох об’єктів (MOTA);
- Оцінка F1 для унікальних ідентифікаторів (IDF1);
- Вища точність відстеження порядку (HOTA [8]);
- Переважно відстежувані об’єкти (MT);
- Перемикання ідентифікаторів (IDSW).

Метрика MOTA вимірює три типи помилок відстеження. Помилки виявлення FN і FP, а також помилка асоціації IDSW. Остаточна оцінка MOTA обчислюється за формолою (2.7).

$$MOTA = 1 - \frac{|FN|+|FP|+|IDSW|}{|gtDet|} \tag{2.7}$$

Де *FN* – кількість неправдивих негативних прогнозів;

FF – кількість неправдивих похитивних прогнозів;

gtDet – набір заздалегіть визначених виявлень;

IDSW – кількість виявлених передаваннь ідентифікаторів.

У MOTA асоціація вимірюється за допомогою концепції перемикача ідентифікації (IDSW). IDSW виникає, коли трекер неправильно змінює

ідентифікатори об’єктів або коли трек було втрачено та повторно ініціалізовано з іншим ідентифікатором. Формально IDSW – це TP, що має прогнозований ідентифікатор, який відрізняється від попереднього прогнозований ідентифікатор. IDSW вимірює лише помилки асоціації порівняно з одним попереднім TP і не враховують помилки, коли той самий передбачений ідентифікатор змінюється на інший, завідомо відомий ідентифікатор (передавання ідентифікатора).

Метрика $IDF1$ (формула 2.8) усуває деякі обмеження MOTA, зосереджуючись на тому, як довго трекер правильно ідентифікує об’єкт, а не просто підраховує помилки. Він заснований на концепції Identification Precision (IDP) та Identification Recall (IDR).

$$IDF1 = \frac{2*IDTP}{2*IDTP+IDFP+IDFN} \tag{2.8}$$

де $IDTP$ – кількість правильно ідентифікованих виявлень;

$IDFP$ – прогнози, які не відповідають жодній реальній дійсності;

$IDFN$ – істинні траєкторії, які не відстежуються.

Метрика $HOTA$ розроблена для усунення обмежень як $MOTA$, так і $IDF1$. Запропонована метрика має на меті забезпечити збалансовану оцінку ефективності виявлення та асоціації. $HOTA$ можна розділити на два критерії: точність виявлення та точність асоціації, що дозволяє окремо аналізувати ці аспекти використовуючи формулу (2.9).

$$HOTA_a = \sqrt{DetA_a AssA_a} \tag{2.9}$$

Де a – представляє різні порогові значення IoU;

$DetA_a$ – точність виявлення;

$AssA_a$ – точність асоціації;

У свою чергу, функцію точність виявлення можливо просто обчисливши $Det\ IoU$, використовуючи кількість TP, FN і FP для всього набору даних за формуолою (2.10):

$$DetA = Det\ IoU = \frac{TP}{TP+FN+FP} \tag{2.10}$$

Асоціація вимірює, наскільки добре трекер пов'язує виявлення протягом певного часу з тими самими ідентифікаторами (ідентифікаторами), враховуючи набір ідентифікаційних зв'язків у базових доріжках. Ми можемо виміряти це, взявши прогнозоване виявлення та наземне виявлення правди, які зіставляються разом, і вимірявши узгодження між цілим прогнозованим виявленням і повним треком наземного виявлення. В результаті IoU асоціації можна за формулою (2.11).

$$AssA = AssA\ IoU = \frac{TPA}{TPA+FNA+FPA} \tag{2.11}$$

де TPA– кількість істинно позитивних асоціацій;

FPA – кількість хибно-позитивних асоціацій;

FNA – кількість хибно-негативних асоціацій.

Варто зазначити що метрика $HOTA$ не ідеально підходить для онлайн-відстеження: оцінка асоціації $HOTA$ залежить від майбутніх асоціацій у всьому відео, що робить дану оцінку менш придатною для перевірки онлайн-відстеження де майбутні дані недоступні. Також $HOTA$ не враховує фрагментацію: не штрафує фрагментовані результати відстеження, оскільки розроблено для зосередження на довгостроковому глобальному відстеженні, яке може не відповідати потребам усіх програм.

2.4 Аналіз алгоритмів відстеження об’єктів

2.4.1 MOT challenge

MOT Challenge [12, 13] – це загальновизнаний тест, призначений для оцінки та порівняння продуктивності систем відстеження кількох об’єктів (MOT). Він надає стандартизовані набори даних і показники, такі як MOTA, IDF1 і перемикачі ідентифікаторів, щоб забезпечити узгоджену оцінку в різних алгоритмах відстеження. На додаток до цих показників, MOT Challenge пропонує вичерпний список трекерів, які були перевірені та підтверджені в різних сценаріях, починаючи від відстеження пішоходів у міських умовах до відстеження транспортних засобів у виді з повітря.

Заначений ресурс дозволяє дослідникам визначати рішення для відстеження, які найкраще підходять для їхніх конкретних випадків використання, сприяючи прогресу в галузі відстеження об’єктів. З наявних трекерів декілька надійних рішень з відкритим вихідним кодом вивчено та проаналізовано. Результати аналізу, а саме метрики алгоритмів представлені в таблиці 2.1. Архітектури цих систем розглянуто, кожне рішення було оцінено підкреслюючи сильні сторони, а також обмеження Зазначений аналіз дає цінну інформацію про придатність цих трекерів для різноманітних застосувань.

Таблиця 2.1 – Алгоритми вдстеження декілької об`єктів в реальному часі.

Трекер	MOTA	IDF1	HOTA	AssA	DetA
SORT [14]	42.7	45.1	36.1	35.9	36.7
DeepSORT [15]	61.4	57.3	41.1	47.7	49.5
FeatureSORT [17]	76.6	75.1	61.3	60.1	62.7

2.4.2 SORT

Алгоритм Simple Online and Realtime Tracking (SORT) [14] – це базовий метод відстеження кількох об’єктів з простотою архітектурою та високою швидкістю. Зазначений алгоритм побудований навколо концепції використання лише інформації про рух для пов’язування виявлень між кадрами, не покладаючись на особливості зовнішнього вигляду. SORT використовує фільтр Калмана (KF) [18] для прогнозування траєкторій об’єктів і угорський алгоритм асоціації даних як покаано на рисунку 2.6, що робить його легким у обчислювальному плані та здатним працювати в режимі реального часу на скромному обладнанні.

Незважаючи на свою ефективність, SORT здебільшого вважається застарілим для багатьох сучасних програм. Відсутність функцій на основі зовнішнього вигляду, таких як вбудовування або візуальні дескриптори, обмежує його надійність у сценаріях із частими оклюзіями, накладанням об’єктів або рухом камери. Ці обмеження часто призводять до вищих показників перемикання ідентифікаторів і зниження загальної точності відстеження в складних або динамічних середовищах. Крім того, SORT має труднощі в налаштуваннях із кількома камерами, де функції зовнішнього вигляду є важливими для узгодженості ідентифікації між камерами.

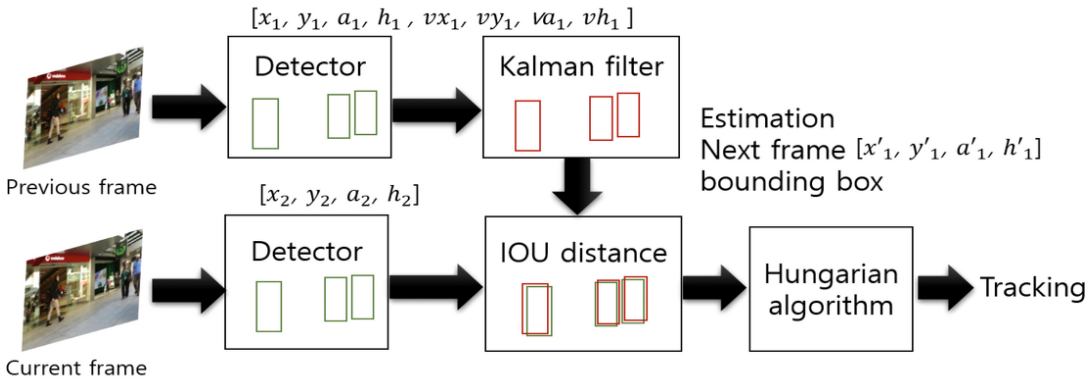

Рисунок 2.6 – Принцип роботи класичного алгоритму SORT

2.4.3 DeepSORT

Deep SORT (Simple online and realtime tracking with deep associaation metric) – це один із надійних алгоритмів відстежування декількох об'єктів [15, 16]. Даний алгоритм поєднує методи класичного алгоритму SORT, глибокі нейроні мережі для отримання ознак для повторної ідентифікації об'єктів на основі зовнішнього вигляду виявлених об'єктів. Для відстеження об'єктів Deep SORT використовує фільтр Калмана та Угорський алгоритм подібним чином, як це реалізовано в класичній версії, рисунок 2.7.

Проблема асоціації виявлених об`єктів та існуючих треків в DeepSORT розв’язується за допомогою Угорського алгоритму. Угорський алгоритм використовується для зіставлення результатів між передбаченими положеннями об'єктів та новими “спостереженнями”. Для постановки проблеми інтегрується інформація про рух та зовнішній вигляд за допомогою комбінації двох відповідних показників.

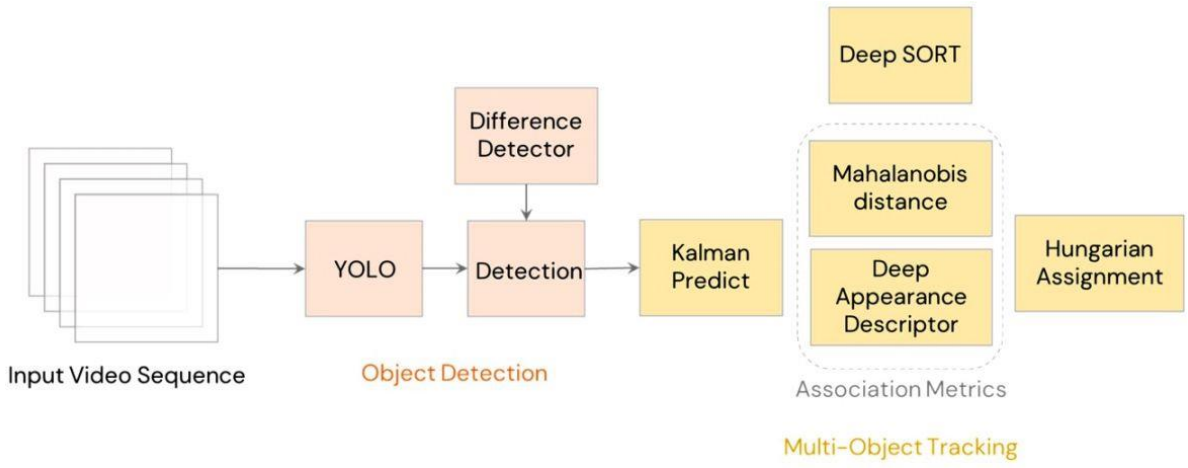

Рисунок 2.7 – Принцип роботи алгоритму DeepSORT

Роботу алгоритму можна описати наступним чином: першочергово об'єкти виявляються в кожному кадрі відео за допомогою відповідної моделі виявлення об’єктів, наприклад YOLO, Faster R-CNN або інші моделі.

Наступним кроком є створення вектору ознак фіксованої довжини (features vector) для кожного виявленого об’єкта, для цього зазвичай використовують глибоку, згорткову нейромережу. Отриманий вектор також часто називають дескриптором об'єкта, цей вектор ознак фіксує характеристики зовнішнього вигляду об’єкта в його обмежувальній рамці. Для створення дескрипторів часто використовуються методи на основі глибокого навчання, такі як сіамські мережі або триплетні мережі.

Після створення дескрипторів Deep SORT використовує угорський алгоритм для зв’язування виявлених об’єктів між кадрами. Deep SORT розглядає як передбачення руху за допомогою алгоритму SORT, так і схожість зовнішнього вигляду з витягнутих векторів ознак. Процес асоціації має на меті зв’язувати об’єкти від кадру до кадру, вирішуючи такі проблеми, як наявність перешкод та тимчасове зникнення.

Після встановлення асоціації використовується фільтр Калмана або подібну методи оцінки стану об'єктів. Фільтр Калмана допомагає передбачити наступний стан об’єкта на основі його попередньої динаміки руху та поточних спостережень. Слід зазначити, що Deep SORT зберігає набір активних треків для поточних відстежуваних об’єктів. Керування життєвими циклами треків відбувається за допомогою кількох параметрів: вводяться такі поняття, як підтвердження треку та віковий параметр.

Трек підтверджується лише після виявлення в кількох послідовних кадрах, таким чином зменшуючи кількість помилкових спрацьовувань. Віковий параметр використовується для видалення старих треків, які нещодавно не були виявлені, гарантуючи, що система підтримує лише активні треки. Після виконання усіх кроків аглоритму, кожному об`єкту присвоюється унікальний ідентифікаторм з пов’язаною інформацією, такою як дескриптор, обмежувальна рамка, напрямок та швидкість ругу. Зазначені параметри використовуютьлся для подальшого аналізу.

3. РОЗРОБКА СИСТЕМИ ВІДСТЕЖЕННЯ ЦІЛЕЙ ДЛЯ UAV

3.1 Підготовка набору даних VisDrone

Головний набір даних викуористаний в даній роботі – VisDrone [20]. Набір даних VisDrone2019 включає 288 відеокліпів, що містять 261 908 кадрів і 10 209 статичних зображень, детальну структуру наведено в табиці 3.1. Набір даних, знятий за допомогою різних камер, встановлених на безпілотниках, охоплює 14 міст Китаю, охоплюючи різноманітні середовища, такі як міські та сільські райони, і різну щільність від рідкісних до багатолюдних місць.

Таблиця 3.1 – Структура та розроділення набору даних VisDrone

	DET	MOT	VID
Тренування	6471 кадрів	24201 кадрів	56 відео
Валідація	548 кадрів	2846 кадрів	7 відео
Тестування	1 580 кадрів	6635	17 відео

Цей набір даних висвітлює широкий спектр об’єктів, включаючи пішоходів, транспортні засоби, велосипеди та трицикли. Надається понад 2,6 мільйона обмежувальних рамок із детальними анотаціями, що забезпечує надійні навчальні дані для таких завдань, як виявлення та відстеження об’єктів. Ключові атрибути, такі як видимість сцени, клас об’єкта та рівні оклюзії, покращують корисність набору даних для реальних програм. Для того, зоб навчити YOLO модель на зазначеносму датасеті необхіддно конвертувати формат обмежувальних рамок в формат “xywh”. Конвертацію можливо зробити локально, а або за допомогою онлейн сервісу “Ultralytics HUB” як показано на рисунку 3.1.

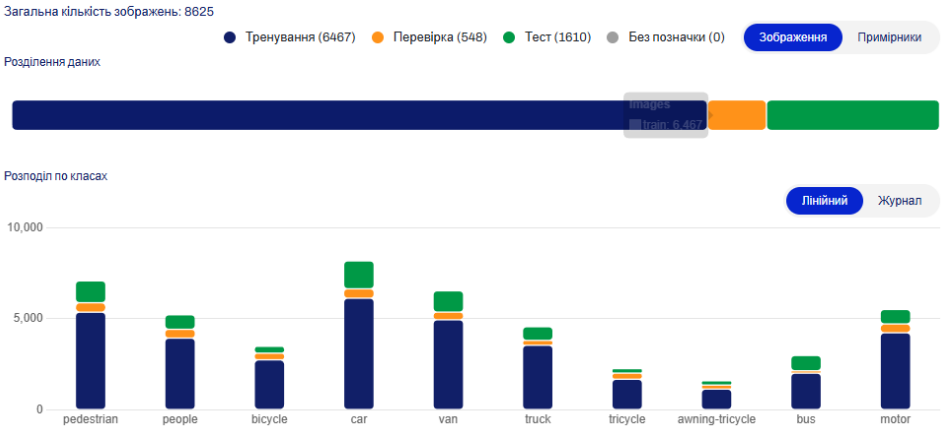

Рисунок 3.1 – Форматування набору дану відповідно до yolo формату

3.2 Підготовка наборів даних для повторної ідентифікації транспортних засобів

Набір даних VRAI [24] містить 137 613 зображень: 13 022 транспортних засобів семи категорій, кожне з яких зроблено принаймні двома БПЛА в різних місцях, під різними кутами огляду та на висоті польоту від 15 до 80 метрів. Це розмаїття посилює варіації всередині класу, роблячи набір даних особливо складним і придатним для розробки надійних моделей ReID. Крім того, анотатори визначили дискримінаційні частини для кожного транспортного засобу, допомагаючи виконувати задачі точного розпізнавання.

Окрім побудови набору даних, автори пропонують алгоритм ReID автомобіля, який використовує ці багаті анотації. Цей метод чітко виявляє дискримінаційні частини для кожного транспортного засобу. VRAI вирішує проблему дефіциту наборів даних ReID транспортних засобів на основі БПЛА та забезпечує цінний ресурс для просування досліджень у цій галузі.

Набір данх VRAI також можливо доповнити використовуючи зобрадення з такизх датасетів як VehicleID та VeRi, рисунок 3.2. В результаті ми отримаємо набір дних з 409,376 зображень виключно для повторної ідентифікації транспортних засобів.

Рисунок 3.2 – Набори даних для ReID транспортних засобів

При обєднанні усіх зазначених наборів даних отримано датасет з загальною кількість зображень 409 376. Загальна кількість транспортних засобів, унікальних ідентифікаторів приблизно 40 065, таблиця 3.2.

Таблиця 3.2 – Структура та розроділення наборів даних для повторної ідентифікації транспортних засобів

Набір даних	VeRi	VehicleID	VRAI	Комбінація
Тренування	37,778	110,178	102,123	250,079
Валідація	643	-	27,522	150,686
Тестування	11,579	111,585	7,968	8,611

3.3 Підготовка наборів даних для повторної ідентифікації людей

Для повторної ідендифікації оглянуто декілька датаестів: Market-1501, MRP та PRAI-1581, їх порванняня днадано в таблиці 3.3.

Market-1501 це загальновизнаний еталон для повторної ідентифікації особи, що містить приблизно 32 668 зображень 1501 особистостей, знятих шістьма стаціонарними камерами в Університеті Цінхуа. Цей набір даних включає знімки з різним фоном, умови освітлення.

Набір даних AVI містить 36 500 зображень 575 транспортних засобів, знятих на різних висотах і в різних точках огляду, підкреслюючи складність ReID літальних апаратів. Разом ці набори даних пропонують комплексні контрольні показники, які охоплюють сценарії зі стаціонарними камерами, наземними та аерофотознімками, що сприяє вдосконаленню ReID для людей і транспортних засобів, римунок 3.3.

Рисунок 3.3 – Набори даних для повторної ідентифікації людей

В роботі «Повторна ідентифікація особи на аерофотознімках» [25] представлено набір даних PRAI-1581. Зазначений набір даних зосереджується на ReID людини на знімках, зроблених БПЛА. Набір даних, отриманий за допомогою БПЛА DJI на висоті від 20 до 60 метрів, містить різноманітну колекцію зображень зі значними варіаціями роздільної здатності, точки огляду та пози.

Таблиця 3.3 – Порівняння наборів даних для ReID людей

Набір даних	Market1501	MRP	AVI	PRAI-1581
Картинок	32668	5597	10863	39461
Ідентифікаторів	1501	51	5124	1581
Висота	< 10m	< 10m	< 10m	20-60m
Кількість UAV	0	9	-	2

3.4 Тренування та тестування нейромереж

Для розробки системи відстеження для дронів, для виявлення об`єктів на кадрі отриманим з передньої камери квадрокоптера було використано модель YOLO останньої версії.

Моделі YOLOv11 [5] були навчені в трьох варіантах: нано, малому та середньому. Характеристики отриманих моделей представлені на таблиці X. Наномодель із 2,58 мільйонами параметрів і потребою 6,3 GFLOP є найлегшою з розміром моделі 5,22 МБ, що робить її ідеальною для середовищ з обмеженими ресурсами. Маленька модель із 9,42 мільйонами параметрів і 21,3 GFLOPs забезпечує баланс між продуктивністю та ефективністю, займаючи 18,2 МБ пам’яті. Модель середнього розміру з 20,04 мільйонами параметрів, 67,7 GFLOPs і обсягом пам’яті 77,0 МБ забезпечує найвищу точність, але вимагає значно більшої обчислювальної потужності та пам’яті.

Таблиця 3.4 – Порівняння характеристик YOLO моделей

Модель	Шарів	Параметрів	FLOPs	Розмір (.pt)
yolo11-nano	238	2,584,102	6.3 GFLOPs	5.22 MB
yolo11-small	238	9,416,670	21.3 GFLOPs	18.2 MB
yolo11-meduim	303	20,037,742	67.7 GFLOPs	77.0 MB

Середня модель сумісна з платою NVIDIA Jetson Nano та подібними бортовими компютерами. При використанні моделей більших розмірів потенційно можливо отримати проблеми з алокацією памятю та отримати затримки при обробки кадру. В результаті модель середнього розміру можливо інтегрувати в бортову систему керування БПЛА без шкоди для розміру платформи чи енергоефективності, що робить зазначену модель оптимальним вибором.

Для тренування моделей було використану бібліотеку PyThorch та пакет ultralytics. Фреймворк Ultralytics при тренуванні автоматично додає аугментацію вхідних зображень, розширюючи набір даних за допомогою різноманітних перетворень під час навчання. Також Ultralytics також надає можливість легко конфігурувати парамети тренування: лістинг 3.1.

Лістинг 3.1 - функція для тренування Yolo моделей:

```
model = YOLO("yolo-uav/yolo11m.pt", task="detect")
results = model.train(data="VisDrone.yaml",
     batch=-1,
     epochs=300,
     imgsz=640,
     device="cuda",
     cache=False,
     plots=True,
     project="yolo-uav",
     name="train-medium", # or nano / small
     patience=100,
     save_period=5)
print(results)
```

Використані параметри для кожної з моделі:

- «imgsz»: Цільовий розмір зображення для навчання. Усі зображення з VisDrone змінюються до розміру 640px перед подачею в модель;

- «batch»: З обраним значенням фреймфорк автоматично вибирає оптимальний розмір пакета на основі доступних системних ресурсів;

- «epochs»: Загальна кількість епох навчання. Кожна епоха представляє повний прохід по всьому набору даних. 300 епох – оптимальне значення, прооте прийшлося пару днів чекати на результати;

- «patience»: Кількість періодів очікування без покращення показників перевірки перед достроковим припиненням навчання.

- «save_period»: Періодичність збереження контрольних точок моделі - кожні 5 епох.

Також, за допомогою опції «plots» було отримано результати заневедені на рисунках 3.4 3.5 та 3.6.

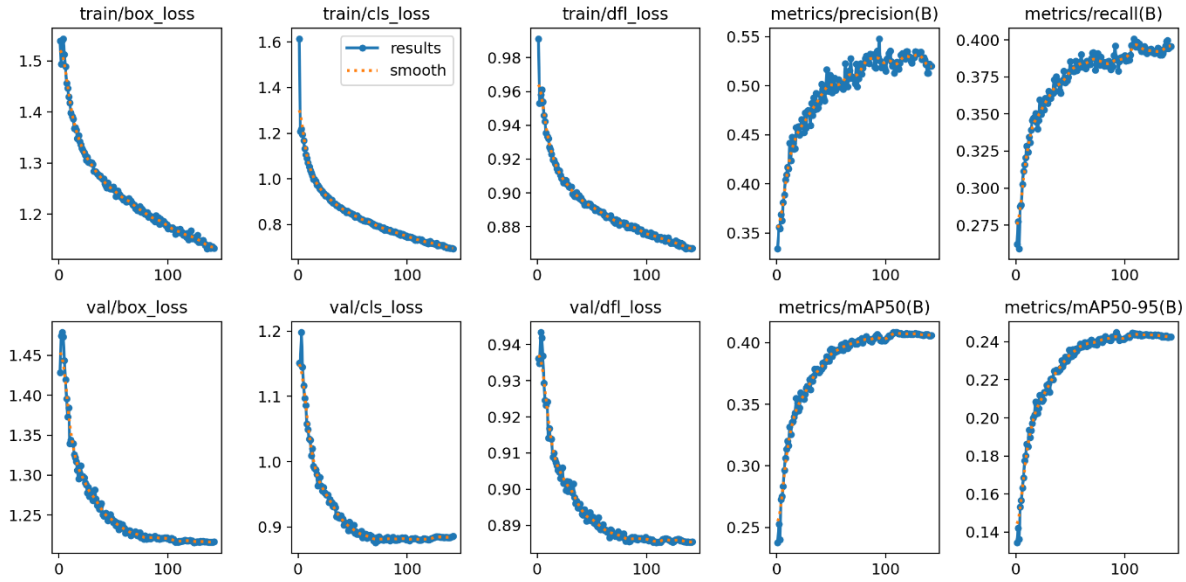

Рисунок 3.4 – Результати тренування yolo11 nano моделі

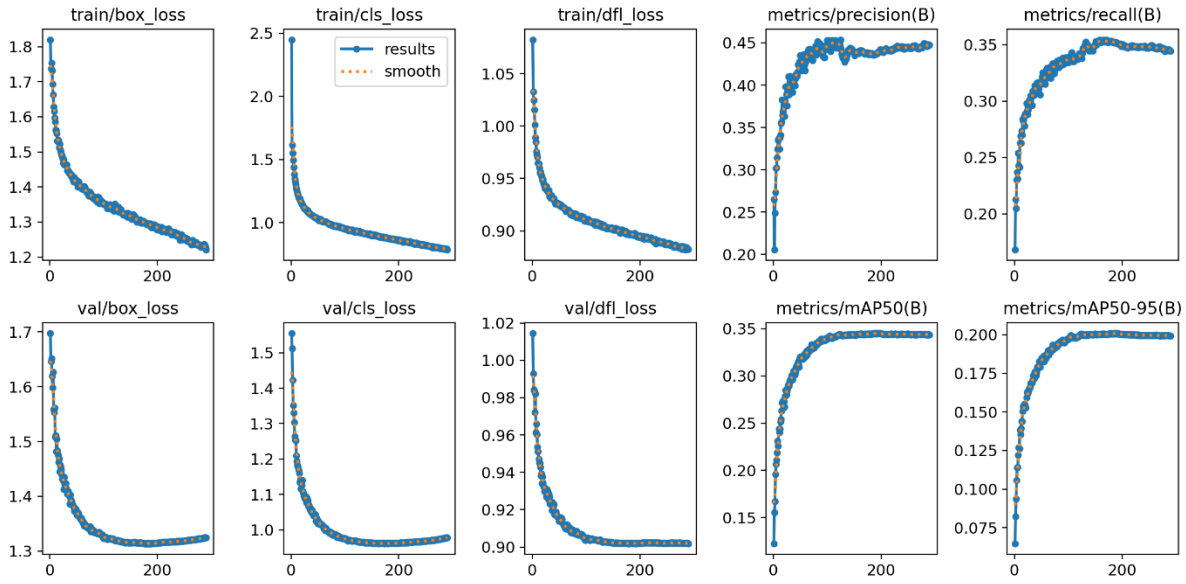

Рисунок 3.5 – Результати тренування yolo11 small моделі

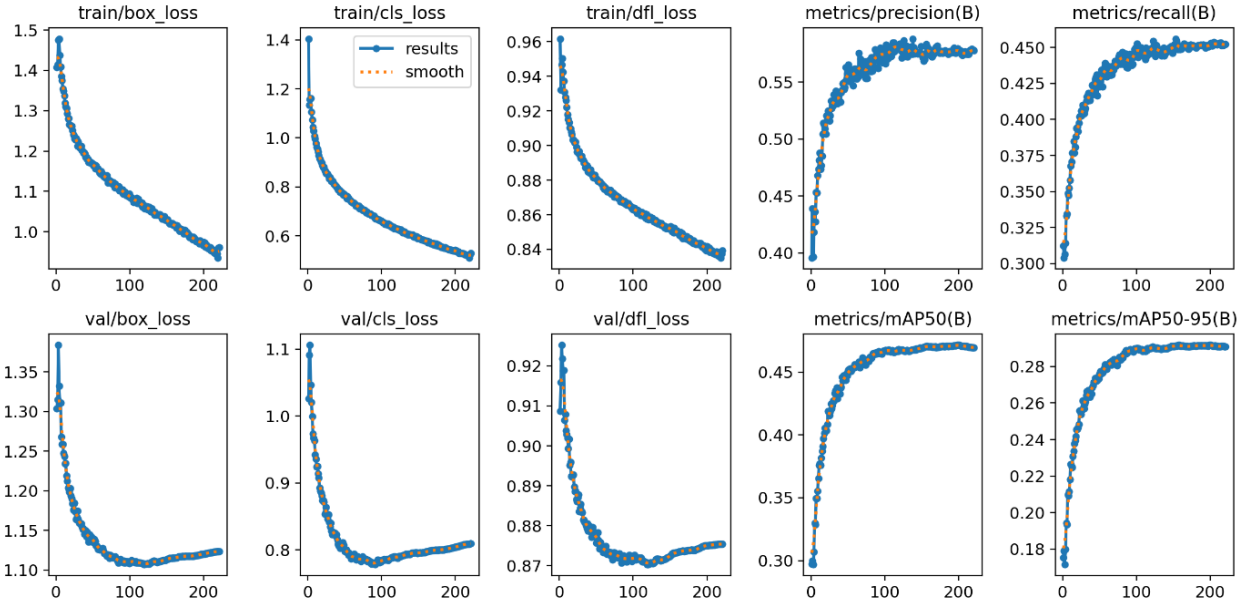

Рисунок 3.6 – Результати тренування yolo11 medim моделі

Метрики «mAP», «precision» та «precision» було розглянуто в попередніх розділах. Метрики «box Loss», «cls loss» та «dfl loss» також використовують при навчанні та валідації моделі.

Box Loss визначає помилку в локалізації обмежувальної рамки. Це гарантує, що передбачені обмежувальні прямокутники точно узгоджуються з завчасно відомими вимірами.

Cls Loss стосується точності класифікації, змушуючи модель призначати правильні мітки класу виявленим об’єктам. Зазвичай базується на перехресній втраті ентропії або фокусній втраті. Зазначена метрика особливо корисна для обробки дисбалансу класів.

Розподіл фокусних втрат (DFL) уточнює передбачення координат обмежувальної рамки, використовуючи для локалізації представлення дискретного біну. DFL оптимізує дрібну просторову точність шляхом включення імовірнісних принципів, що дозволяє моделі ефективно розрізняти тонкі просторові варіації або об’єкти, що перекриваються.

Отримані моделі експотровано в формати: ONNX, TensorRT (FP32), TensorRT (FP16) та TensorRT (INT8) для подальшого аналізу та порівняння. При ексортуванні моделей в TensorRT було використано « квантування після навчання» (PTQ). Зазначений метод включає в себе перетворення вагових коефіцієнтів моделі та активації з високоточних форматів з плаваючою комою у формати нижчої точності, як правило, INT8. Це зменшує обсяг пам’яті моделі та вимоги до обчислень, забезпечуючи швидший і ефективніший висновок, особливо на обладнанні, оптимізованому для операцій із низькою точністю.

Для калібрування використовувались кадри з вибірки для тестування. Репрезентативний набір даних використовується для обчислення коефіцієнтів масштабування, які відобразили значення з плаваючою комою у відповідні цілі числа. Ці коефіцієнти масштабування допомагають максимально зберегти динамічний діапазон і точність вихідної моделі. Хоча PTQ іноді може призвести до незначного зниження точності через зниження точності, це практичний і економічно ефективний підхід для розгортання моделей на крайніх пристроях або в інших середовищах з обмеженими ресурсами.

Кожнва отримана модель була протестована, результати аналізу представлені на таблицях 3.5, 3.6, 3.7, а також на рисунку 3.7.

Порівняння часу обробки кадру. Час висновку для всіх трьох моделей (нано, малої та середньої) з використанням PyTorch значно вищий порівняно з оптимізацією TensorRT. Це підкреслює накладні витрати стандартного PyTorch для розгортання. «FP32 Precision» дає швидший висновок порівняно з PyTorch, але не відповідає продуктивності оптимізацій FP16 та INT8. Час висновку збільшується зі збільшенням розміру моделі, досягаючи піку в 11,3 мс для середньої моделі. «FP16 Precision» забезпечує суттєве прискорення порівняно з FP32, зберігаючи аналогічний рівень точності. Найшвидший висновок реєструється за 1,6 мс

для YOLO11n. «INT8 Precision» забезпечує найшвидший час висновку для всіх трьох моделей з мінімальним погіршенням точності та показників запам’ятовування для нано- та малих моделей. INT8 досягає балансу між швидкістю моделі та зменшеним розміром пам’яті.

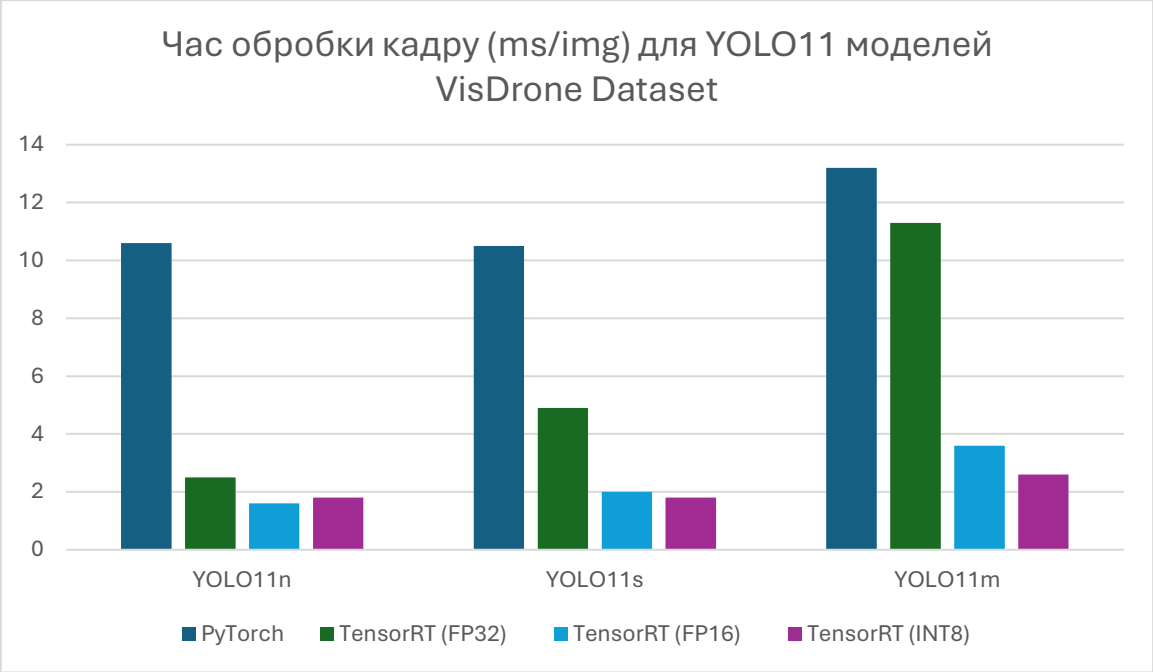

Рисунок 3.7 – Порівняння часу обробки моделей в різних форматах

Порівняння точності та розмірів моделей. Модель YOLO11s у форматі PyTorch досягла найкращого балансу в балах mAP порівняно з іншими форматами. Формати TensorRT (FP32 і FP16) підтримували майже однакові рівні точності, тоді як INT8 версія продемонструвала помітне падіння, особливо враховуючи метрику mAP50-95 (-13,9%). Моделі TensorRT INT8 вимагають менше пам’яті (6,04 МБ), що значно зменшує розмір порівняно з ONNX і FP32, але все ще є прийнятними для легких БПЛА.

Модель YOLO11s у форматі PyTorch забезпечує трохи кращі показники точності та запам’ятовування, ніж інші формати. Точність FP16 майже відповідає PyTorch, тоді як INT8 знову демонструє помітне погіршення запам’ятовування (-11%).

YOLO11m подібна до нано та малих моделей: формати TensorRT (FP32 і FP16) зберігають майже однакову продуктивність; INT8 зазнає значного погіршення mAP50-95 (-3,1%). При розмірі 24,2 МБ розмір моделі INT8 становить майже одну третину розміру PyTorch.

Таблиця 3.5 – Порівняння характеристик nano моделі

Формат	Precision	Recall	mAP50	mAP50-95	Розмір
PyThorch	0.636	0.247	0.443	0.295	5.22 MB
ONNX	0.625	0.243	0.436	0.291	10.0 MB
TensorRT (FP32)	0.625	0.243	0.436	0.291	12.6 MB
TensorRT (FP16)	0.626	0.240	0.436	0.291	7.80 MB
TensorRT (INT8)	0.621	0.163	0.391	0.254	6.04 MB

Таблиця 3.6 – Порівняння характеристик small моделі

Формат	Precision	Recall	mAP50	mAP50-95	Розмір
PyThorch	0.674	0.318	0.497	0.338	18.2 MB
ONNX	0.663	0.315	0.49	0.335	36.0 MB
TensorRT (FP32)	0.663	0.315	0.49	0.335	40.1 MB
TensorRT (FP16)	0.662	0.315	0.49	0.334	20.7 MB
TensorRT (INT8)	0.68	0.207	0.44	0.296	12.9 MB

Таблиця 3.7 – Порівняння характеристик medium моделі

Формат	Precision	Recall	mAP50	mAP50-95	Розмір
PyThorch	0.676	0.394	0.536	0.371	77.0 MB
ONNX	0.673	0.391	0.534	0.37	76.6 MB
TensorRT (FP32)	0.673	0.391	0.534	0.37	86.0 MB
TensorRT (FP16)	0.673	0.392	0.534	0.37	41.0 MB
TensorRT (INT8)	0.691	0.316	0.501	0.34	24.2 MB

3.5 Алгоритм відстеження декількох об’єктів

В ході роботи розроблено трекер DroneSort. Запропонований трекер є розширенням алгоритму DeepSort. Архітектура розробленого трекера поєднує в собі чотири основні модулі:

- модуль виявлення об’єктів на базі YOLO11;
- модуль отримання унікальних ознак для різних типів обжктів;
- модуль повторної ідентифікації який поєднує відстеджження та передбачення траекторії руху об`єктів за допомогою розширеного фільтру Калмана, а також порівняння;
- модуль повторної ідентифікації об`єктів виявлених ішим дроном.

В DroneSort також інтегровано алгоритм компенсації руху камери на основі моделі покращеного коефіцієнта кореляції (ECC). Система відстеження пов’язує виявлені об’єкти з існуючими треками за допомогою двоетапного процесу, використовуючи матриці витрат на основі IoU та зіставлення карт ознак об`єктів. Кожнен трек розгирено з метаданними, що включають локальні та глобальні ідентифікатори, мітки класів і динамічний банк функцій для повторної ідентифікації. Архітектура DroneSort оптимізовано для відстеження декількох об`єктів одночасно та здатна працювати в режимі реального часу.

Модулі виявлення об’єктів, отримання унікальних ознак та повторної ідентифікації поєднані та використані в класі «Tracker», діаграма класів представлена на рисунку 3.8. Класс Трекер у DroneSort служить основним компонентом, відповідальним за зв’язування нових виявлень із наявними доріжці, забезпечуючи безперервне та надійне відстеження кількох об’єктів. Його основною функцією є встановлення відповідності між виявленими об’єктами та існуючими треками за допомогою метрик глибокої асоціації ознак (або косинусної, або евклідової відстані). Коли виявляється новий об’єкт, який не відповідає існуючій доріжці, ініціалізується нова доріжка.

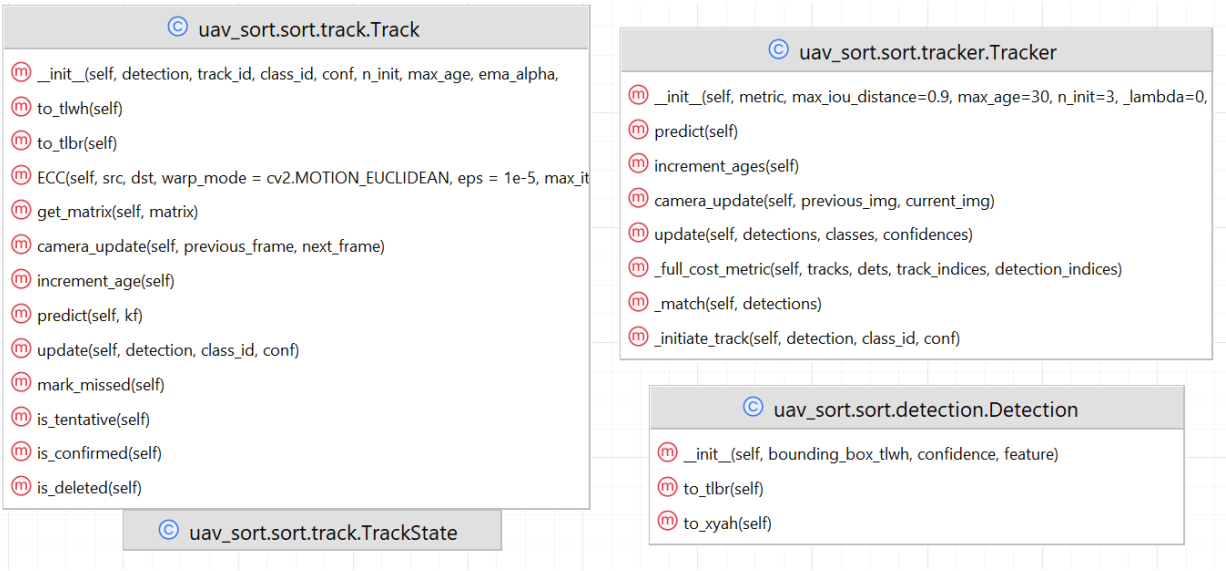

Рисунок 3.8 – Основні компоненти трекеру: «Detection», «Track», «TrackState» та «Tracker»

Трекер використовує кілька ключових параметрів і методів для досягнення поставленої задачі. Використані параметри:

- «metcic»: метрика відстані, яка використовується для асоціації;
- «max_age»: визначає максимальну кількість послідовних ітерацій, які трек може залишати незрівнянним перед видаленням;
- «kf»: фільтр Калмана що відстежує траєкторії об’єктів у просторі зображення, прогнозуючи обмежувальну рамку кожної доріжки.

Основні методи включають:

- «predict»: використовує фільтр Калмана для поширення стану кожної доріжки на один крок вперед;
- «update»: виконує процес асоціації даних, використовуючи функцію _match для асоціації наявних треків із виявленнями. Він оновлює треки та підтримує надійну метрику відстані для ефективного керування змінами;
- «_match»: приватний метод, обробляє завдання асоціації як для підтверджених, так і для непідтверджених треків, використовуючи функції зовнішнього вигляду та критерії на основі IoU.

Класи «KalmanFilter» та «NNDistanceMetric» складають модуль для повторної ідентифікації об`єктів та ассоціції спостережень з наявними треками. Діаграми класів представлені на рисунку 3.9.

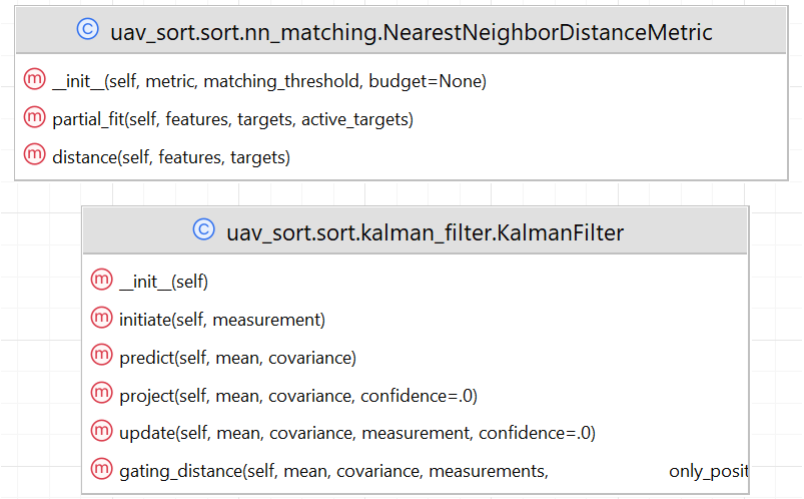

Рисунок 3.9 – Модуль для повторної ідентифікації об`єктів

Філтьтр Калмана є основним компонентом трекера DroneSort, призначеного для оцінки та прогнозування стану об’єктів (обмежувальних рамок) у просторі зображення при тому не використовуючи зовнішні й вигляд об`єкта. Цей фільтр працює в 8-вимірному просторі станів, що представляє центральну позицію (x, y), співвідношення сторін (a), висоту обмежувальної рамки (h) та їхні відповідні швидкості. Модель руху передбачає константну швидкость, де рух об’єкта є лінійним протягом всього часу.

Основні характеристики класу це відображення стану об`єкта та лінійна модель спостереження. Вектор стану включає просторові та геометричні властивості об’єкта, що дозволяє передбачати майбутні позиції та уточнювати поточні спостереження. Розроблені методи класу:

- «initiate»: викликається при ініціалізації трекера, використовує обмежувальну рамкку як початковий стан. При ініціалізації також генерується відповідна коваріаційна матриця, закладаючи основу для наступних прогнозів і оновлень;

- «predict»: просуває матрицю стану та коваріації об’єкта вперед у часі на основі моделі постійної швидкості. Цей метод оцінює майбутнє положення, співвідношення сторін і розмір обмежувальної рамки з урахуванням руху об’єкта;

- «update»: Уточнює прогнозований стан за допомогою нових вимірювань виявлення. Цей крок зменшує помилки передбачення, коригує розташування та розмір об’єкта та забезпечує точнішу оцінку стану об’єкта.

Класс «NNDistanceMetric» використовується для обчислення «найближчої відстані» між характеристиками нещодавно виявлених об’єктів та існуючими цілями для надійного зв’язку даних. Він підтримує два типи метрики відстані: евклідову відстань і косинусну відстань, що забезпечує гнучкість підбору відповідності на основі потреб програми.

Основним методом цбого класу є «distance», цей метод розраховує матрицю вартості для відповідності ознаки об`єкта та цілі, забезпечуючи основу для викорнання повторної івдентифікації на основі зовнашнього вигляду об`єктів. В якості вхідних параметрів метод приймає карту ознак,“features”, у вигляді матриці $N \times M$, де N – кількість ознак виявлення, а M – розмірність кожної з карт, а також перелік виявлених об`єктів: список k ідентифікаторів доріжок, що відповідають активним доріжкам, з якими зіставляються виявлення. Результатом цього методу є матриця вартості розміром (k,N), де кожен елемент (i,j) представляє найближчий квадрат відстані (евклідової або косинусної) між ціллю i та виявленням j.

Для реалізації модуля отримання унікальних ознак для різних типів обжктів використано клас “ReIdBackend”. Зазначений клас використовує “FeatureExtractor” з пакету FastReId [21]. FeatureExtractor служить універсальним інтерфейсом для розгортання та запуску попередньо навчених моделей, адаптованих до завдання повторної ідентифікації. Екстрактор призначений для завантаження певних архітектур, ініціалізації ваг і попередньої обробки вхідних даних, забезпечуючи стандартизований і

ефективний процес вилучення карти функцій.

ReIdBackend ініціалізує FeatureExtractor з обраною моделью на зазначеному пристрої (CPU або GPU). В даній роботі у якості моделей використовувались нейромережі OSNet та ResNet, а твакож їх варації. FeatureExtractor виконує пряме розповсюдження на моделі для обчислення високовимірної карти ознак, що представляє зовнішній вигляд об’єкта, отриманий результат потім використовується в класі “Tracker”.

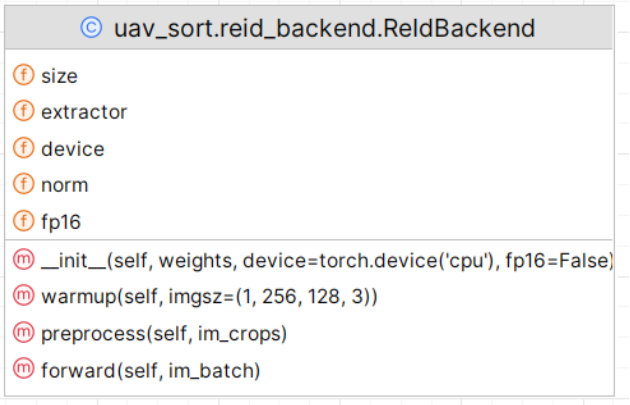

Рисунок 3.10 – Клас для роботи з Feature Extractor

3.6 Тестування розробленої системи на наборі даних VisDrone

Для тестування розробленого алгоритму додано класс TrackerRunner. Зазначений клас дозволяє ініціалізувати трекер з небхідними. Для тесту використовувалось декілька послідовностей з VisDrone-MOT набору даних.

У першій послідовності, знятій на низькій висоті ввечері на перехресті вулиць, трекер працював ефективно, виграючи від відносно невеликої кількості об’єктів у сцені. Це середовище дозволило трекеру працювати майже в режимі реального часу, демонструючи його здатність ефективно справлятися з менш завантаженими та простішими сценаріями.

Кадр з збереженого відео з першій послідовності представлено на рисунку 3.1.

Рисунок 3.11 – Тестування трекеру на першій послідовності

Друга серія - шосе, зняте з великої висоти протягом дня. У цій послідовності трекеру було важко підтримувати постійну продуктивність через високу щільність виявлених об’єктів, що збільшувало обчислювальні вимоги. Це підкреслює компроміс між щільністю об’єктів і швидкістю обробки, демонструючи, що хоча DroneSort добре працює в сценаріях з низькою щільністю об’єктів, його ефекти вність може знизитися за більшого робочого навантаження.

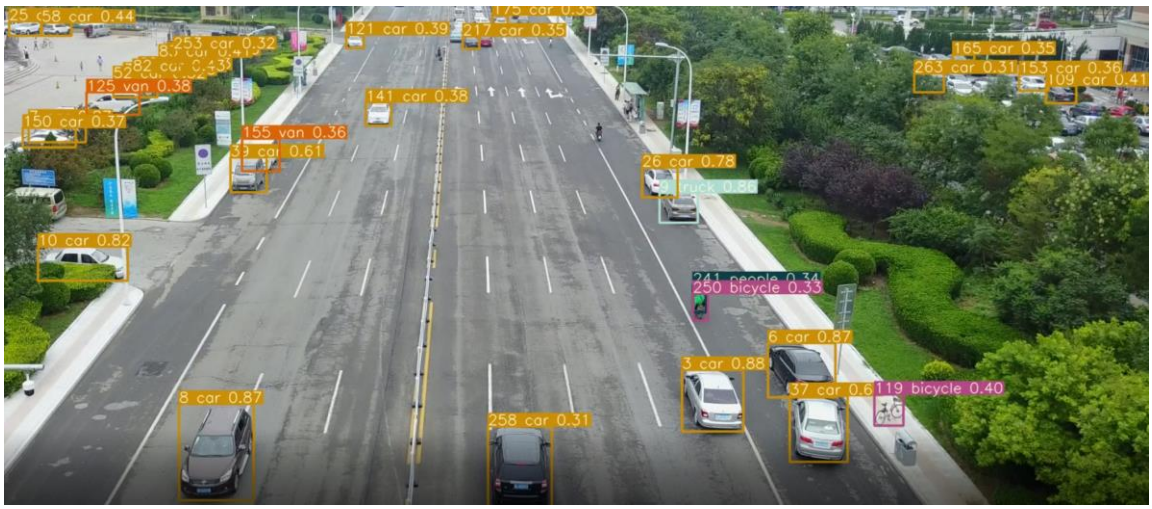

Рисунок 3.12 – Тестування трекеру на другій послідовності

3.7 Інтеграція системи відстеження з ROS2 та PX4

PX4 – це програмне забезпечення для автопілота з відкритим вихідним кодом для безпілотних транспортних засобів, розроблене згідно з дозвільною ліцензією “BSD-3-Clause”. PX4 пропонує надійну та модульну архітектуру, яка підтримує різні типи транспортних засобів, що робить його одним із найбільш гнучких рішень для застосувань дронів різних видів, включаючи гібридні (VTOL) безпілотники. PX4 надає повноцінну документацію, шадійну інфраструктуру для розробки, а також має велику спільноту. Додатково, інтеграція з ROS2 що забезпечується мостом uXRCE-DDS, спрощує зв’язок між автопілотом і комп’ютерами-супутниками, що робить його кращим вибором для проектів, пов’язаних зі складним комп’ютерним зором або автономною навігацією. Порівняно з Ardupilot, PX4 часто забезпечує більш зручну для розробників екосистему, особливо для передових досліджень і розробок.

Для виконання задч повязаних з комп’ютерним зором, до PX4 необхідно підключити комп’ютер-супутник, який забезпечить необхідну обчислювальну потужність для обробки та аналізу даних зображень. На відміну від контролера польоту, який зосереджується на динаміці польоту та контролі в реальному часі, комп’ютер-супутник призначений для розширених завдань, таких як “Optical Flow”, “Motion Capture”, або, як в даній роботі – виявлення та відстеження декількох об`єктів. Очікується, що для інтеграції з PX4 комп’ютер-супутник працюватиме під керуванням ROS2 (Robot Operating System) [26].

ROS 2 представляє собою високо адаптивну та надійну структуру для додатків робототехніки, забезпечуючи глибоку інтеграцію з автопілотом PX4 для розробки складних повітряних систем. ROS 2 полегшує створення режимів польоту, які не відрізняються від тих, що є рідними для PX4, забезпечуючи пряму взаємодію з внутрішніми темами uORB на високій

швидкості передачі даних. Ця функція є особливо вигідною для сценаріїв, що вимагають керування з низькою затримкою, наприклад взаємодії комп'ютера-супутника, використання бібліотек на базі Linux або розробки розширених парадигм керування польотом.

Зв'язок між ROS 2 і PX4 здійснюється за допомогою рівня проміжного програмного забезпечення, що реалізує протокол XRCE-DDS, який перетворює повідомлення PX4 uORB у типи, сумісні з ROS 2. Ця система використовує визначення повідомлень uORB для створення механізмів серіалізації та десеріалізації, забезпечуючи ефективний двонаправлений зв'язок. Крім того, інтерфейс MAVROS, який підтримується проектом MAVROS, пропонує додатковий шлях інтеграції.

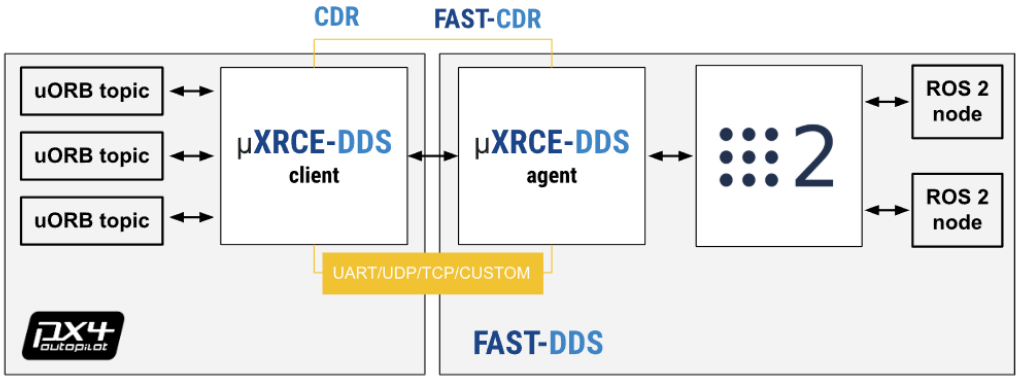

Рисунок 3.13 – Взаємодія між PX4 та ROS2

Для інтеграції трекера DroneSort з PX4 було реалізовано модуль ROS2 під назвою DroneImageHandler для обробки зображень отриманих з передньої камери. Цей модуль підписується на тему "video_frames", отримуючи зображення з розміром черги 10 повідомлень. Бібліотека CvBridge використовується для перетворення повідомлень із зображеннями ROS у формат OpenCV, для подільшої передачі в трекер, що забезпечує сумісність із завданнями обробки зображень.

Кожен кадр спочатку передається до трекера DroneSort, де об'єкти виявляються та відстежуються, отримані результати використовуються для

помічкння виявлених об`єктів на відео та у результаті чого утворюються обмежувальні рамки, помічені унікальними ідентифікаторами. Ці оброблені кадри, що містять накладення обмежувальної рамки, потім зберігаються на диску для подальшого аналізу або автономної оцінки. Поєднуючи обробку та зберігання зображень, вузол гарантує, що операції БПЛА можуть використовувати відстеження в реальному часі, зберігаючи детальні візуальні дані для подальшого перегляду. На лістингу 3.1 наведено реілізації main функції. Життєвий цикл вузла включає ініціалізацію за допомогою rclpy, обертання для безперервної роботи та належне очищення під час вимкнення, що зміцнює його роль у оптимізації завдань комп'ютерного зору в системах БПЛА.

Лістинг 3.2 – «main» функція ROS2 ноди:

```
def main(args=None):
  rclpy.init(args=args) # Ініціалізація бібліотеки rclpy
    image_subscriber = ImageSubscriber() # Створення ноди
    rclpy.spin(image_subscriber) # запуск ноди
  # Знищення ноди при завершенні роботи програми
  image_subscriber.destroy_node()
  rclpy.shutdown()
if __name__ == '__main__':
  main()
```

3.8 Тестування розробленої системи в симуляторі Gazebo

Для тестування трекера DroneSort з автопілотом PX4 створено симуляцію за допомогою застосунка Gazebo. Тестове середовище включає моделі декількох людей та транспортних засобів. Тестування передбачало запуск системи для імітації реальної установки БПЛА. Автопілот PX4 було

ініціалізовано в режимі SITL (Програмне забезпечення в циклі) з моделлю дрона x500 і конкретними координатами пози, що забезпило узгодженість початкових умов для тестування.

Міст зображень ROS2 був створений для потокової передачі зображень камери з симуляції Gazebo, забезпечуючи вхідні дані для трекера DroneSort. Сценарій uav_camera_det.py обробив ці кадри за допомогою інтегрованого трекера, уможлививши виявлення та відстеження об’єктів у реальному часі. Для візуалізації результатів додано вікно “Detected Objects”, рисунок 3.14 на якому зображено виявлені об`єкти.

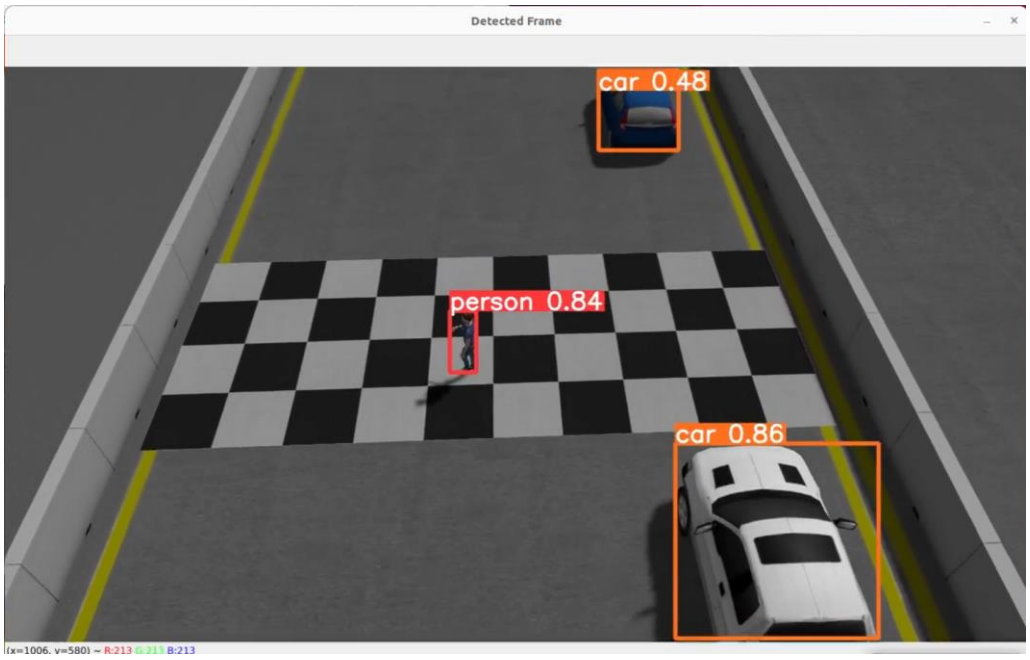

Рисунок 3.14 – Виявлення об`єктів при інтеграції трекера з PX4

Моделювання в симуляторі Gazebo підтвердило здатність системи виконувати виявлення та відстеження в реальному часі використовуючи синтетичні дані, зберігаючи при цьому стабільний зв’язок між трекером, автопілотом PX4. Установка продемонструвала ефективність трекера DroneSort в динамічних середовищах, підтвердивши його готовність до розгортання в реальних додатках БПЛА.

ВИСНОВКИ

В роботі вирішено актуальну задачу компютерного зору для безпілотних летільних апаратів – розроблено систему для виявлення та відстеження декількох об`єктів в режимі реально часу. Запропонвану систему протестовано та інтегровано з PX4 автопілотом.

В ході дослідження порівняні сучасні моделі виявлення об`єктів, трекери, що здатні працювати в режимі реального часу, а також методи повторної ідентифікації людей та транспортних засобів.

Успішно розроблено новий трекер DroneSort. В якості моделі виявлення оєктів використано сучаний алгоритм - YOLOv11. В ході розробки були навчені три моделі: нано, малі та середні варіанти на VisDrone наборі даних. Ортимані моделі проекспортовані в формати PyTorch, TorchScript, ONNX і TensorRT (FP32, FP16 і INT8), в результаті чого отримано 15 варіантів моделей. Кожний модуль було протестовано, знайдено оптимальне поєднання точності та обчислювальної продуктивності. Найефективнішою моделлю став середній варіант YOLOv11 у форматі TensorRT INT8, який поєднує задовільну точність виявлення з мінімальним часом висновку, що робить його ідеальним для розгортання на обладнанні з обмеженими ресурсами, наприклад NVIDIA Jetson Nano.

Розроблений трекер поєднує такі функції як повторна ідентифікація за допомогою OSNet, компенсація руху камери та передбачення траєкторії за допомогою розширеного фільтра Калмана. Інтеграція цього трекера з системою автопілота PX4, досягнута за допомогою інфраструктури на основі ROS2. Інтеграція дозволяє БПЛА здійснювати відстеження кількох об’єктів у режимі реального часу в динамічних середовищах, не заважаючи роботи автопілоту.

Отримані результати вказують на потужний потенціал для практичного застосування в задачах, що вимагають надійного та динамічного виявлення та супроводу об’єктів за допомогою БПЛА. Майбутня робота може включати тестування в реальному світі та подальшу оптимізацію для розширення масштабованості та чутливості в режимі реального часу за вищої щільності об’єктів. Також ця робота встановлює базу для майбутніх досліджень щодо автономних дронів і груп БПЛА. Методології, включаючи оцінку моделей виявлення об’єктів, розробку надійного трекера та інтеграцію розширених можливостей відеобачення з керуванням польотом, забезпечують основу для подальшого прогресу в розвитку автономності дронів. Внески цієї роботи прокладають шлях для таких застосувань, як спостереження на основі роїв, пошуково-рятувальні операції та координація кількох дронів у складних середовищах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Detection and Tracking Meet Drones Challenge / P. Zhu та ін. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021. С. 1. URL: https://doi.org/10.1109/tpami.2021.3119563.

2. Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review / S. A. H. Mohsan та ін. *Drones*. 2022. Т. 6, № 6. С. 147. URL: https://doi.org/10.3390/drones6060147.

3. Swarms of Unmanned Aerial Vehicles – A Survey / A. Tahir та ін. *Journal of Industrial Information Integration*. 2019. Т. 16. С. 100106. URL: https://doi.org/10.1016/j.jii.2019.100106.

4. Unmanned Aerial Vehicles: A Review / A. A. laghari та ін. *Cognitive Robotics*. 2022. URL: https://doi.org/10.1016/j.cogr.2022.12.004.

5. Khanam R., Hussain M. YOLOv11: An Overview of the Key Architectural Enhancements. *arXiv*. 2024. abs/2410.17725. URL: https://doi.org/10.48550/arXiv.2410.17725.

6. YOLOv10: Real-Time End-to-End Object Detection / G. Ding та ін. *arXiv*. 2024. 2405.14458. URL: https://arxiv.org/abs/2405.14458.

7. Varghese, Sambath. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness}. *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*. 2024. С. 1–6. URL: https://doi.org/10.1109/ADICS58448.2024.10533619.

8. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking / J. Luiten et al. *International Journal of Computer Vision*. 2020. URL: https://doi.org/10.1007/s11263-020-01375-2.

9. SSD: Single Shot MultiBox Detector / C. Ning та ін. *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. 2017. С. 549–554. URL: https://doi.org/10.1007/978-3-319-46448-0_2.

10. Girshick R. Fast R-CNN. *arXiv*. 2015. 1504.08083. URL: https://arxiv.org/abs/1504.08083.

11. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / S. Ren та ін. *arXiv*. 2016. 1506.01497. URL: https://arxiv.org/abs/1506.01497

12. MOT20: A benchmark for multi object tracking in crowded scenes / P. Dendorfer та ін. *arXiv*. 2020. 2003.09003. URL: https://arxiv.org/abs/2003.09003.

13. MOTChallenge: A Benchmark for Single-Camera Multiple Target *arXiv*. 2020. 2010.07548. URL: https://arxiv.org/abs/2010.07548.

14. Simple online and realtime tracking / Z. Ge та ін. *2016 IEEE International Conference on Image Processing (ICIP)*. 2016. 3464-3468. URL: https://doi.org/10.1109/ICIP.2016.7533003.

15. Wojke N., Bewley A., Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP)*. 2017. С. 3645–3649. URL: https://doi.org/10.1109/ICIP.2017.8296962.

16. StrongSORT: Make DeepSORT Great Again / Y. Du та ін. *IEEE Transactions on Multimedia*. 2023. С. 1–14. URL: https://doi.org/10.1109/tmm.2023.3240881.

17. Hashempoor H., Koikara R., Hwang Y. D. FeatureSORT: Essential Features for Effective Tracking. *ArXiv*. 2024. abs/2407.04249. URL: https://doi.org/10.48550/arXiv.2407.04249.

18. Ranyang, Ji, Kaifan. Kalman Filter and Its Application. *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. 2015. С. 74–77. URL: https://doi.org/10.1109/ICINIS.2015.35.

19. Microsoft COCO: Common Objects in Context / T.-Y. Lin та ін. *arXiv*. 2015. 1405.0312. URL: https://arxiv.org/abs/1405.0312.

20. Vision Meets Drones: A Challenge / P. Zhu та ін. *arXiv*. 2018. 1804.07437. URL: https://arxiv.org/abs/1804.07437.

21. He L., Liao X., Liu X. FastReID: A Pytorch Toolbox for General Instance Re-identification. *arXiv*. 2020. 2006.02631. URL: https://arxiv.org/abs/2006.02631.

22. Omni-Scale Feature Learning for Person Re-Identification / K. Zhou та ін. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019. С. 3701–3711. URL: https://doi.org/10.48550/arXiv.1905.00953.

23. Deep Residual Learning for Image Recognition/ Xiangyu. Zhang та ін. *arXiv*. *2015* 1512.03385. URL: https://arxiv.org/abs/1512.03385

24. Person Re-Identification in Aerial Imagery / S. Zhang та ін. *IEEE Transactions on Multimedia*. 2021. Т. 23, Institute of Electrical and Electronics Engineers (IEEE). С. 281–291. URL: https://doi.org/10.1109/TMM.2020.2977528.

25. Vehicle Re-identification in Aerial Imagery: Dataset and Approach / P. Wang та ін. *arXiv*. 2019. 1904.01400. URL: https://arxiv.org/abs/1904.01400.

26. Robot Operating System 2: Design, architecture, and uses in the wild / S. Macenski та ін. *Science Robotics*. 2022. Т. 7, № 66. eabm6074. URL: https://doi.org/10.1126/scirobotics.abm6074.