КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки, освітньо-наукова програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

“ Розробка інтелектуальної системи ідентифікації рухомих об'єктів методами Data Science**”**

Студента 2-го курсу групи ІАВ-21

БІЛУХИ Назара Романовича
(прізвище, ім'я, по батькові)

(підпис студента)

Науковий керівник:

Кандидат технічних наук
(науковий ступінь, вчене звання)

ЄРЕМЕНКО Богдан Михайлович
(прізвище, ім'я, по батькові)

(дата) (підпис)

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри технологій управління

(підпис) (прізвище, ініціали) (дата)

Київ – 2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень
Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Морозов В.В.
_____

«___» _____20__року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Білуха Н.Р.

Група ІАВ-21

1. Тема кваліфікаційної роботи

Розробка інтелектуальної системи ідентифікації рухомих об'єктів методами Data Science

Затверджена наказом по від «___»_____20__р. № ___.

2. Строк подання студентом готової роботи – "__"_____20__р.

3. Цільова установка та вихідні дані до роботи

Навчитися виявляти рухомі об'єкти за допомоги методів Data Science з подальшим створенням автоматизованої системи. Використовувати сучасні архітектури для виявлення образів, розглянути методи аугменації даних для покращення стабільності системи. Для реалізації поставленого завдання передбачається використання як релевантних публічних наборів даних так і власноруч зібраних та анотованих візуальних даних, що містять рухомі об'єкти в різних умовах.

4. Зміст роботи

У змісті роботи передбачено аналіз сучасних методів Data Science та нейронних мереж для ідентифікації рухомих об'єктів з відеоданих БПЛА, зокрема в різних умовах видимості, з подальшим підбором, обробкою візуальних даних та побудовою й удосконаленням відповідних моделей глибокого навчання (наприклад, YOLO, Faster R-CNN, RetinaNet). Буде розроблено програмний прототип інтелектуальної системи, проведено експериментальну оцінку його ефективності за допомогою релевантних метрик, проаналізовано перспективи застосування отриманих результатів та сформульовано висновки.

5. Перелік графічного матеріалу (слайдів)

Презентація складається з 21 слайду.

6. Календарний план виконання роботи:

№ п/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	01.10.24	01.10.24
2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	27.12.24	27.12.24
3.	Формування перелік у нормативних матеріалі, літератури з проблематики дипломної роботи	10	08.01.25	07.01.25
4.	Складання розгорнутого плану кваліфікаційної роботи	5	18.01.25	18.01.25
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	19.01.25 - 20.01.25	20.11.25
6.	Підготовка розділу 1 «Аналіз теоретико-методологічних основ використання нейронних мереж в задачах керування проектною	10	12.02.25	13.02.25

	діяльністю компанії»			
7.	Підготовка розділу 2 «Структури нейронних мереж та способи їх навчання»	14	08.03.25	08.03.25
8.	Підготовка розділу 3 «Застосування нейронних мереж для задач відбору перспективних проектів»	14	01.04.25	01.04.25
9.	Підготовка розділу 4 «Проект впровадження нейронних мереж для задач відбору перспективних проектів»	13	20.04.25	20.04.25
10.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	15	03.05.25	03.05.25
11.	Передача кваліфікаційної роботи науковому керівникові	2	04.05.25	04.05.25
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	11.05.25	11.05.25
13.	Попередній захис т кваліфікаційної роботи	5	17.05.25	17.05.25

Дата видачі завдання «____» _____2025 р.

Керівник роботи ___к.т.н. Єременко Богдан Михайлович_____
(посада, прізвище, ім’я, по батькові)

_____
(підпис)

Завдання прийняв до виконання студент групи ІАВ-21 Білуха Н.Р.

_____
(підпис)

ЗМІСТ

АНОТАЦІЯ 6

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ 8

ВСТУП 9

РОЗДІЛ 1. АНАЛІЗ МЕТОДОЛОГІЙ ВИКОРИСТАННЯ НЕЙРОМЕРЕЖ В ЗАДАЧАХ ВИЯВЛЕННЯ ОБ'ЄКТІВ 14

1.1. Формулювання та опис задачі ідентифікації рухомих об'єктів ..14
1.2. Інтелектуальний аналіз даних та сфери застосування 19
1.3. Аналіз методологій застосування нейронних мереж в задачах виявлення об'єктів.....24
1.4. Аналіз методів застосування в задачі виявлення рухових об'єктів в сфері оборони.....30
1.5. Постановка задачі дослідження.....30

РОЗДІЛ II. СТРУКТУРИ ТА ОСОБЛИВОСТІ АРХІТЕКТУР МЕРЕЖ ДЛЯ ІДЕНТИФІКАЦІЇ РУХОМИХ ОБ'ЄКТІВ 34

2.1. Модель YOLO.....34
2.2. Модель Faster R-CNN. 42
2.3. Модель RetinaNet.....46
2.4. Архітектура Visual Transformer (ViT).....50

РОЗДІЛ III. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ТА ПОБУДОВА МОДЕЛІ ВИЯВЛЕННЯ РУХОМИХ ОБ'ЄКТІВ 55

3.1. Формалізація бази знань виявлення рухомих об'єктів.....55
3.2. Вибір засобів для реалізації моделі 64
3.3. Вибір інструментів розмітки даних 68
3.4. Реалізація моделі 72

РОЗДІЛ IV. ЗАСТОСУВАННЯ ТА ПОДАЛЬШЕ ВИКОРИСТАННЯ МОДЕЛІ ВИВЛЕННЯ РУХОМИХ ОБ'ЄКТІВ..... 76

4.1. Принцип роботи цілісної системи 76
4.2. Особливості архітектур та розгортання 80
4.3. Застосування розробки.....81

ВИСНОВОК 84

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 86

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 - Комп’ютерні науки,
освітня програма “Інформаційна аналітика та впливи”

Дипломна робота магістра Білухи Назара Романовича.

Тема роботи – «Розробка інтелектуальної системи ідентифікації рухомих об'єктів методами Data Science».

Мета дипломної роботи магістра – підвищення точності та швидкодії виявлення рухомих об’єктів за допомоги нейронних мереж.

Об’єкт дослідження – процеси обробки відеоданих з безпілотних літальних апаратів.

Предмет дослідження – моделі та методи та технології комп'ютерного зору та машинного навчання для автоматизованого виявлення рухомих об'єктів.

Наукова новизна роботи – розширення теоретичної бази виявлення рухомих об'єктів в умовах динамічного відеопотоку з безпілотних літальних апаратів, з використанням методів глибокого навчання.

У роботі досліджуються існуючі підходи до автоматизованого виявлення об'єктів у відеопотоці, отриманому з безпілотних літальних апаратів, та аналізуються особливості їх застосування у військовій сфері. Розробляється методика застосування моделі глибокого навчання для виявлення рухомих ворожих об'єктів, а також обґрунтовується доцільність та необхідність її впровадження в системи обробки даних з БПЛА.

Дипломна робота складається зі вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього налічує 95 сторінок та перелік посилань з 49 джерел на 3 сторінках та додатки.

Ключові слова: рухомий об’єкт, нейромережа, детектинг, ідентифікація.

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

БПЛА – безпілотний літальний апарат

CV – computer vision

ML – machine learning

FPV – first person view

ДСНС – державна служба надзвичайних ситуацій

CNN – convolutional neural networks

RPN – region proposal network

FPN – feature pyramid network

NMS – non maximal suppression

PANet – path aggregation network

NLP – natural language processing

ВСТУП

Стрімкий розвиток технологій комп’ютерного зору здійснив революцію в галузі виявлення об’єктів, зокрема у сфері виявлення та розпізнавання рухомих об’єктів. Цей напрям привертає увагу через широке коло застосувань – у системах відеоспостереження, автономних транспортних засобах, робототехніці та інфраструктурі розумних міст. Розробка інтелектуальної системи для виявлення рухомих об’єктів є складним завданням, адже вона потребує не лише точного виявлення, а й здатності працювати в динамічному середовищі та забезпечувати обробку в реальному часі.

Актуальність теми даної роботи визначається зростаючою потребою у високоточних та швидкодіючих технологіях виявлення та ідентифікації рухомих об’єктів у реальному часі, особливо в умовах повномасштабної російсько-української війни. Дрони відіграють ключову роль в сучасних умовах ведення військових дій, забезпечуючи повітряну розвідку, спостереження та коригування вогню в режимі реального часу. Оснащення дронів інтелектуальними системами виявлення цілей значно підвищує їхню ефективність, точність та автономність під час виконання бойових і розвідувальних завдань. Система здатна дозволяти дронам збирати більше корисної інформації, здійснювати автоматичне наведення на ціль, проводити розвідки, коригувати вогонь та підсвічувати необхідну інформацію операторам БПЛА. Також це дозволяє оперативно виявляти загрози, підвищувати ситуаційну обізнаність і приймати своєчасні рішення, що безпосередньо впливає на збереження життя військових і цивільних. Також такі технології важливі для безпечної роботи автономного транспорту, ефективного відеоспостереження, робототехніки й міської інфраструктури. Підвищення точності й швидкодії виявлення та відстеження об’єктів безпосередньо впливає на

рівень безпеки громадян, оптимізує логістичні процеси й сприяє цифровій трансформації міст.

Метою даного дослідження є розробка інтелектуального застосунку для ідентифікації рухомих об’єктів із використанням сучасних технологій комп’ютерного зору. Робота спрямована на вирішення завдань точного виявлення та відстеження об'єктів у режимі реального часу. Основними завдання дослідження є:

1. Розробка та застосування алгоритмів виявлення об'єктів різних типів, розмірів в різних погодних умовах.
2. Створення ефективних механізмів відстеження, які зберігають наявність об’єктів між кадрами.
3. Оптимізація системи для роботи в реальному часі.
4. Забезпечення адаптивності до змін умов, таких як освітлення чи перешкоди.
5. Оцінка ефективності системи в різних сценаріях та порівняння з сучасними методами.
6. Аналіз можливостей застосування в сферах оборони, робототехніки, транспорту та агросекторі.

Інтелектуальна система ідентифікації рухомих об’єктів повинна враховувати кілька ключових аспектів таких як надійні алгоритми виявлення об’єктів, що здатні працювати з різними типами, розмірами та орієнтаціями об’єктів; ефективні механізми відстеження для збереження ідентичності об’єкта між кадрами. Система повинна мати можливість обробляти дані в реальному часі для забезпечення своєчасного прийняття рішень; адаптивність до змін у навколишньому середовищі, таких як освітлення та перекриття; інтеграція з іншими сенсорами та джерелами даних для підвищення точності та обізнаності про контекст.

Об'єктом дослідження є процес виявлення та відстеження рухомих об'єктів в динамічних середовищах за допомогою інтелектуальних систем computer vision. Предметом дослідження є CV алгоритми для застосувань у відеоспостереженні та автономних системах. Дослідження зосереджене на розробці методів, що забезпечують високу точність і адаптивність до змінних умов навколишнього середовища.

У процесі розробки застосунку було використано комплекс методів, що поєднує теоретичні та практичні підходи. Теоретичний аналіз наукових джерел дозволив структурувати сучасні підходи до виявлення та відстеження об'єктів, оцінити ефективність існуючих алгоритмів і визначити напрями їх удосконалення. Для побудови моделей було застосовано CV методи, зокрема нейронні мережі типу YOLO, RetinaNet та Fast RCNN, що забезпечують високу точність виявлення і стабільне відстеження об'єктів у відеопотоці. Практична частина дослідження передбачає створення та навчання моделей на попередньо підготовленому наборі даних, з урахуванням різних умов зйомки та типів об’єктів. Для підвищення якості вхідних даних було використано базові методи попередньої обробки зображень, зокрема нормалізацію та фільтрацію шумів. Для оцінювання ефективності системи застосовувалися такі кількісні метрики як Precision, Recall, F1-score та mAP (mean Average Precision). Крім того, було використано моделювання сценаріїв реального часу для перевірки працездатності системи в умовах, наближених до практичного застосування, зокрема відеорозвідка та виявлення рухомих образів в умовах бойових дій. Аналіз отриманих результатів здійснювався з урахуванням порівняння з сучасними базовими рішеннями у сфері комп’ютерного зору.

У дослідженні запропоновано інтегровану систему виявлення та відстеження рухомих об'єктів, яка поєднує сучасні нейронні мережі з ефективними трекерами для обробки відео в реальному часі. Враховано

адаптацію до змін зовнішніх умов та можливість інтеграції з додатковими сенсорами для підвищення точності та стійкості системи. Результати дослідження можуть бути використані для розробки систем відеоспостереження, безпілотних літальних апаратів, автономного транспорту та роботизованих платформ.

Розроблена інтелектуальна система була протестована на реальних відеоматеріалах, отриманих із дронів під час виконання завдань з виявлення та ураження техніки противника, а також у процесі відеорозвідки. Випробування проводилися в умовах симуляції бойового середовища, що включали змінне освітлення, димові перешкоди та часткові перекриття об’єктів. Усі відеоматеріали були відібрані з дійсних джерел, що підтверджує їхню достовірність та створює умови досить наближені до реальних. Система продемонструвала здатність стабільно виявляти та відстежувати цілі різного типу, розміру й швидкості, зберігаючи точність і швидкодію. Отримані результати підтвердили її ефективність для оперативного аналізу даних з БПЛА та прийняття рішень у реальному часі.

Результати роботи були апробовані та представлені на Міжнародній науковій конференції *“Urban Transformation in the EU: Challenges and Opportunities”*, 17 квітня 2025 року, за підсумками якої опубліковано одну працю *“Facial Matching Technologies in Smart Cities: Enhancing Public Safety and Personalized Urban Services”*, ключові положення якої лягли в основу для написання даної роботи.

У вступі описана практична цінність роботи, зокрема для оборонної сфери, що полягає у значному підвищенні ефективності, точності та автономності безпілотних літальних апаратів при виконанні бойових і розвідувальних завдань, завдяки автоматизованому виявленню та ідентифікації ворожих цілей. Це значно сприяє покращенню ситуаційної

обізнаності, оперативності прийняття рішень та збереженню життів особового складу і цивільного населення в сучасних умовах ведення бойових дій.

РОЗДІЛ 1. АНАЛІЗ МЕТОДОЛОГІЙ ВИКОРИСТАННЯ НЕЙРОМЕРЕЖ В ЗАДАЧАХ ВИЯВЛЕННЯ ОБ'ЄКТІВ

1.1. Формулювання та опис задачі ідентифікації рухомих об'єктів

Однією з актуальних науково-технічних проблем використання ML алгоритмів в умовах бойового застосування БПЛА є розробка ефективних методів ідентифікації об'єктів, що переміщуються, на основі аналізу відеопотоку з камер. Завдання, що розглядається в цій магістерській роботі, полягає у створенні інтелектуальної системи, здатної забезпечити надійне розпізнавання таких об'єктів у складних та динамічних умовах бойового середовища.

Специфіка бойового застосування БПЛА створює низку серйозних викликів. Динамічність відеоданих, спричинена рухом самого апарата, призводить до постійних змін ракурсу, масштабу, освітленості та появи перешкод. Додатковими викликами для задачі є складне й змінне фонове оточення – руйнування, дим, ландшафт – що ускладнює відокремлення цілей. Система повинна ідентифікувати різнорідні типи об'єктів, таких як танки, бойові машини піхоти, бронеавтомобілі, артилерійські установки, зенітні ракетні комплекси, інженерна техніка, транспорт, який забезпечує постачання логістики та усі інші транспортні засоби, що можуть становити тактичний перевагу в умовах ведення бойових дій.

Велика варіативність вхідних даних значно ускладнює створення унікальної системи, яка здатна адаптуватись до різних типів БПЛА, зокрема через різноманітність камер пристроїв. Тепловізійні зображення, що фіксують теплове випромінювання об'єктів, характеризуються незалежністю від умов освітлення та здатністю виділяти об'єкти з відмінною температурною сигнатурою. Однак, вони мають обмежену деталізацію текстур та кольорів, а ефективність виявлення залежить від

наявності достатнього теплового контрасту між об'єктом та фоном. Зображення, отримані з приладів нічного бачення, підсилюють слабке освітлення, забезпечуючи кращу деталізацію порівняно з тепловізійними, проте їхня робота залежить від наявності хоча б мінімального рівня освітленості.

Рисунок 1.1. Варіативність вхідних даних

Відеопотоки з FPV дронів відрізняються високою динамічністю, швидкою зміною кадру та широким кутом огляду, що може призводити до геометричних спотворень та артефактів руху. Зображення з розвідувальних дронів, отримані в видимому спектрі, надають багату кольорову інформацію та високу роздільну здатність, але їх якість суттєво залежить від умов освітлення.

Ця значна варіативність візуальних даних створює низку проблем для системи ідентифікації. Навчені на одному типі зображень моделі можуть

демонструвати низьку ефективність при обробці іншого типу. Система повинна бути спроектована таким чином, щоб мати можливість обробляти різні формати, роздільні здатності та колірні простори вхідних даних. Створення універсальної моделі, здатної ефективно працювати з усім спектром можливих вхідних зображень, є складним завданням, яке може призвести до зниження точності ідентифікації для окремих типів даних. Для успішного навчання робастної моделі потрібні великі обсяги навчальних даних, що охоплюють усі потенційні варіації вхідних зображень, характерні для різних типів дронів та умов їх застосування.

Подолання цих проблем може бути досягнуто шляхом застосування різноманітних стратегій. Одним з підходів є розробка окремих моделей, спеціалізованих для обробки кожного конкретного типу зображень, хоча це може збільшити обчислювальні витрати та складність системи. Іншим перспективним напрямом є використання мультимодальних підходів, що передбачають об'єднання інформації, отриманої з різних сенсорів, для підвищення надійності ідентифікації. Техніки адаптації домену можуть бути використані для зменшення розриву між різними типами вхідних даних та покращення здатності моделі до узагальнення. Застосування спеціалізованих архітектур нейронних мереж, розроблених з урахуванням особливостей певних типів зображень, а також методів донавчання (transfer learning), що передбачають попереднє навчання моделі на великих різноманітних наборах даних з подальшим донавчанням на специфічних типах, також можуть сприяти підвищенню ефективності системи.

Обмеженість ресурсів програмного забезпечення у дронів висуває вимоги до обчислювальної ефективності, а потреба в оперативності – до високої швидкодії алгоритмів при збереженні точності. Завдання

ускладнюється також через використання маскування, зміну погодних умов і часу доби, що впливає на якість вхідних даних. Методи глибокого навчання, зокрема сучасні підходи до виявлення об’єктів, демонструють високу ефективність у задачах виявлення та локалізації об’єктів.

Важливим етапом є також розробка ефективних алгоритмів попередньої обробки вхідних відеоданих. Застосування таких алгоритмів має бути спрямоване на покращення загальної якості зображення та мінімізацію впливу негативних факторів, таких як шуми або нестабільність зображення. Крім того, для підвищення стійкості розробленої моделі до різноманітних варіацій візуальних даних, характерних для бойового середовища, необхідно використовувати методи збільшення обсягу та різноманітності навчальних даних.

Методи збільшення обсягу та різноманітності навчальних даних являють собою набір технік, спрямованих на штучне розширення тренувального набору шляхом створення модифікованих копій існуючих зображень або відеокадрів. Ці модифікації мають на меті імітувати різноманітні варіації, які можуть зустрічатися в реальних умовах експлуатації системи, підвищуючи таким чином робастність та узагальнюючу здатність навченої моделі. До основних категорій методів збільшення даних, які можуть бути ефективними для даної задачі, належать геометричні трансформації, колірні модифікації, застосування фільтрів та композиційні методи.

Геометричні трансформації включають такі операції, як випадкові повороти зображень на невеликі кути, масштабування, зсуви по горизонталі та вертикалі, а також віддзеркалення. Застосування цих трансформацій дозволяє навчити модель бути інваріантною до незначних

змін ракурсу спостереження та положення об'єктів у кадрі, що є типовим для відео, отриманого з рухомого дрона.

Колірні модифікації передбачають зміну яскравості, контрастності, насиченості та гами кольорів зображень. Ці методи допомагають моделі стати більш стійкою до змін освітлення та погодних умов, які можуть суттєво варіюватися в умовах бойових дій. Випадкові зміни колірних характеристик навчальних зображень змушують модель вивчати більш загальні ознаки об'єктів, не прив'язуючись до конкретних колірних відтінків.

Застосування різноманітних фільтрів, таких як розмиття за Гауссом або медіанний фільтр, може імітувати погіршення якості зображення, спричинене атмосферними явищами або особливостями роботи камери. Навчання на таких штучно спотворених даних підвищує стійкість моделі до реальних деградацій зображення.

Композиційні методи включають більш складні трансформації, такі як випадкове обрізання частин зображення з подальшим масштабуванням, вставка випадкових об'єктів або регіонів на зображення, а також змішування кількох зображень. Ці методи сприяють навчанню моделі розпізнавати об'єкти частково прихованими або на складному фоні, що є актуальним для умов бойових дій, де об'єкти можуть бути замасковані або знаходитися в захаращеному середовищі.

Ефективність застосування конкретних методів data augmentation та їхніх параметрів повинна визначатися емпірично, шляхом проведення експериментів та оцінки впливу різних комбінацій трансформацій на продуктивність навченої моделі на валідаційному наборі даних. Важливо забезпечити, щоб застосовані аугментації були реалістичними та не

призводили до створення артефактів, які не зустрічаються в реальних умовах застосування системи.

1.2. Інтелектуальний аналіз даних та сфери застосування

Для розробки ефективної системи ідентифікації рухомих об'єктів, інтелектуальний аналіз даних є невід’ємною складовою. Цей процес охоплює не лише методи виявлення об’єктів, а й ширший спектр аналітичних маніпуляцій, спрямованих на отримання важливої інформації з візуальних даних.

Збір даних для навчання системи є поступовим процесом, що залежить від типу безпілотного авіаційного комплексу, встановленого на ньому сенсорного обладнання та конкретних завдань, які виконуються. Оператори дронів відіграють ключову роль у процесі збору даних, оскільки саме вони здійснюють керування польотом, наведенням камер та фіксацію необхідних візуальних матеріалів. Процес передачі зображень, як правило, відбувається в режимі реального часу або з певною затримкою, залежно від використовуваних каналів зв'язку та відстані між дроном і наземною станцією управління. У багатьох сучасних безпілотних системах використовується цифровий радіозв'язок, що забезпечує передачу відеопотоку високої якості. Оператор має можливість спостерігати відео в режимі першої особи (FPV) або використовувати інтерфейс управління камерою для зміни кута огляду, масштабування та фокусування.

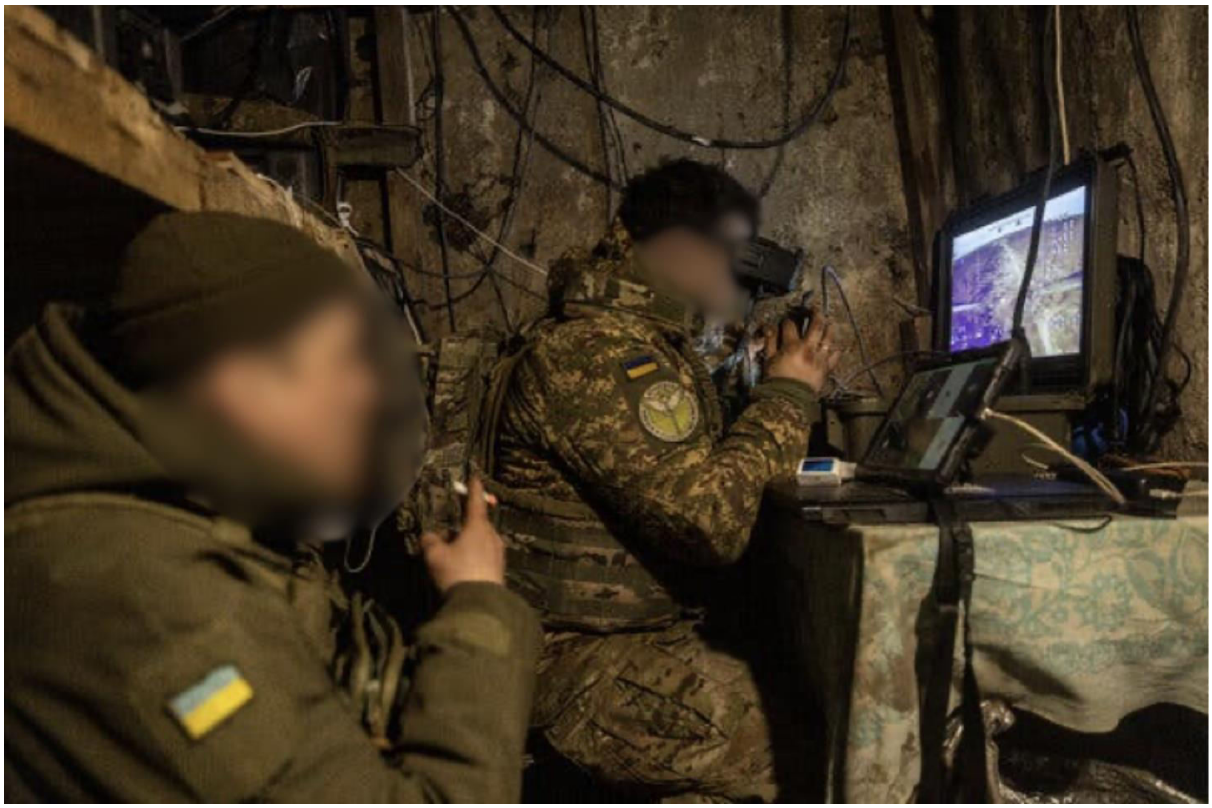

Рисунок 1.2. Процес роботи оператора FPV-дронів, запис відеоматеріалів та збір інформації.

Фото від New York Times Томаса Гіббон-Неффа та Юрія Шивала

Фіксація зображень або відео здійснюється оператором за допомогою відповідних команд, що надсилаються на бортовий комп'ютер дрона. Збережені дані можуть передаватися на наземну станцію управління в режимі реального часу для оперативного аналізу або зберігатися на бортових носіях інформації для подальшого опрацювання після завершення польоту.

У процесі збору даних оператори керуються поставленими завданнями, які можуть включати моніторинг певних районів, ідентифікацію конкретних типів об'єктів або документування певних подій, оператори FPV-дронів в свою чергу мають лише завдання ураження техніки супротивника. Оператори повинні мати відповідну

підготовку для ефективного використання обладнання дрона та розпізнавання потенційно важливих об'єктів у візуальному потоці. Їхні навички у виборі оптимального ракурсу, висоти польоту та часу зйомки можуть суттєво впливати на якість зібраних даних.

Крім даних, зібраних операторами в реальних умовах, для навчання системи можуть використовуватися існуючі бази даних, що містять зображення військової техніки, особового складу та інших релевантних об'єктів. Однак, такі бази даних можуть не повністю відображати специфіку зображень, отриманих саме з дронів в умовах бойових дій, а саме: специфічні кути огляду, динаміка руху камери, особливості освітлення та атмосферних умов. У зв'язку з цим, важливим напрямом є збір даних безпосередньо під час навчань, тренувань або в умовах реального застосування. Такі дані є найбільш цінними, оскільки вони відображають реальні умови експлуатації системи.

Також перспективним є використання методів симуляції для генерації синтетичних даних. За допомогою спеціалізованих програмних засобів можна створювати віртуальні сцени, що імітують різні бойові ситуації, типи місцевості, погодні умови та рух різноманітних об'єктів. Перевагою синтетичних даних є можливість контролювати умови зйомки, отримувати велику кількість анотованих даних та імітувати рідкісні або небезпечні сценарії.

Проте військова справа далеко не єдина сфера застосування даної технології. Існує чимало завдань, для агросектору, екологічної галузі, транспорту та ДСНС. У сфері сільського господарства інтелектуальний аналіз візуальних даних, отриманих з безпілотних літальних апаратів, супутників або наземних камер, відкриває значні можливості для оптимізації виробничих процесів та підвищення врожайності. Методи

ідентифікації рухомих об'єктів можуть застосовуватися для автоматичного виявлення сільськогосподарської техніки на полях, контролю за її переміщенням та ефективністю використання. Крім того, аналіз зображень може допомогти у виявленні ділянок з нерівномірним ростом рослин, ознак захворювань або ураження шкідниками на ранніх стадіях, що дозволяє оперативно вживати необхідних заходів. Моніторинг стану посівів на великих площах за допомогою дронів, оснащених камерами різного спектрального діапазону, та подальший інтелектуальний аналіз отриманих даних сприяє точному землеробству, оптимізації внесення добрив та засобів захисту рослин, що призводить до зниження витрат та підвищення екологічності сільськогосподарського виробництва. Оцінка врожайності на основі аналізу аерофотознімків також є важливим застосуванням інтелектуального аналізу, що дозволяє прогнозувати обсяги збору врожаю та оптимізувати логістичні процеси.

У галузі екологічного моніторингу інтелектуальний аналіз візуальних даних є потужним інструментом для спостереження за станом навколишнього середовища та виявлення негативних антропогенних впливів. Методи ідентифікації рухомих об'єктів можуть використовуватися для моніторингу популяцій диких тварин, відстеження їхніх міграційних шляхів та виявлення випадків браконьєрства. Аналіз аерокосмічних знімків дозволяє виявляти незаконні вирубки лісів, розливи нафти та інші види забруднення навколишнього середовища, а також оцінювати масштаби стихійних лих, таких як повені або лісові пожежі. Автоматизований аналіз відеопотоків з камер спостереження, встановлених у природних заповідниках, може допомогти у виявленні порушень природоохоронного законодавства та контролі за поведінкою відвідувачів.

У сфері транспортної інфраструктури, окрім управління дорожнім рухом, інтелектуальний аналіз візуальних даних може застосовуватися для моніторингу стану інженерних споруд, таких як мости, тунелі та залізничні колії. Аналіз зображень, отриманих за допомогою дронів або спеціалізованих сканувальних систем, дозволяє виявляти тріщини, корозію, деформації та інші дефекти на ранніх стадіях, що сприяє своєчасному проведенню ремонтних робіт та запобіганню аваріям. Моніторинг залізничних колій за допомогою інтелектуальних систем може допомогти у виявленні сторонніх об'єктів на коліях або поблизу них, що підвищує безпеку руху поїздів. Аналіз відеоданих з камер, встановлених на рухомому складі, може використовуватися для контролю за станом контактної мережі та іншого обладнання.

Ще однією важливою сферою застосування є діяльність ДСНС. Під час здійснення рятувальних операцій інтелектуальний аналіз візуальних даних є незамінним інструментом для швидкого та ефективного пошуку постраждалих в умовах надзвичайних ситуацій, таких як землетруси, повені або техногенні катастрофи. Аналіз зображень, отриманих з дронів або інших повітряних платформ, може допомогти у виявленні людей серед завалів або на затоплених територіях. Методи ідентифікації рухомих об'єктів можуть бути адаптовані для виявлення ознак людської присутності, таких як теплові сигнатури або рух. Автоматизований аналіз великих обсягів візуальних даних значно скорочує час, необхідний для пошуку постраждалих, та підвищує шанси на їхнє успішне порятунок.

1.3. Аналіз методологій застосування нейронних мереж в задачах виявлення об’єктів

Згорткові нейронні мережі (CNN) є основою сучасних систем виявлення об'єктів, забезпечуючи автоматичне навчання розрізнювальних ознак безпосередньо з зображень. CNN складаються з послідовності шарів, кожен з яких виконує відповідну функцію. Згорткові шари застосовують набори фільтрів до вхідного зображення для виявлення локальних залежностей, таких як краї, кути та текстури. Шари підвибірки, такі як максимальна або середня підвибірка, зменшують просторову розмірність карт ознак, роблячи представлення більш компактним та інваріантним до невеликих зсувів об'єкта. Функції активації, вводять нелінійність, дозволяючи мережі навчатися складнішим залежностям. Повнозв'язні шари зазвичай розташовуються наприкінці мережі та використовуються для класифікації на основі вивчених ознак.

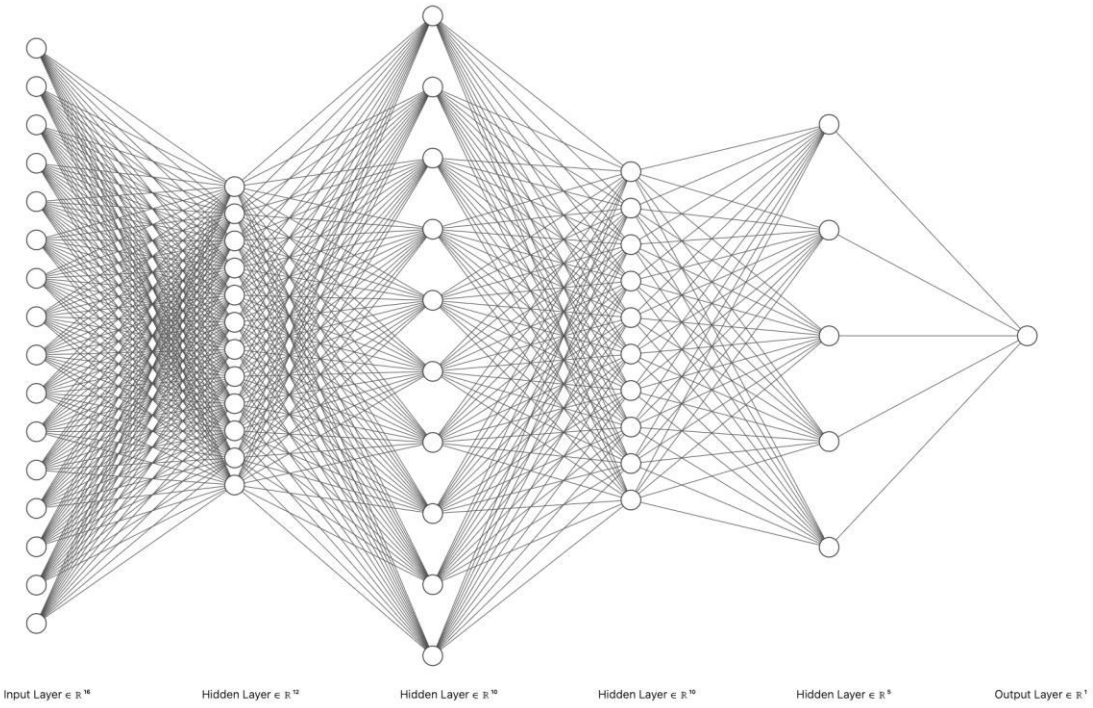

Рисунок 1.3. Графічне подання згорткових нейронних мереж

Поєднання цих шарів дозволяє CNN будувати ієрархічне представлення візуальної інформації: від простих ознак на нижніх рівнях до складних, семантично значущих концепцій на вищих рівнях. Популярні базові архітектури, такі як VGG, ResNet, DarkNet яка використовується в YOLO та EfficientNet [10,19], слугують потужними екстракторами ознак для багатьох моделей виявлення об'єктів. Вибір конкретної базової архітектури є критичним проектним рішенням, що безпосередньо впливає на компроміс між точністю та обчислювальною ефективністю. Глибші та складніші архітектури, наприклад, різні варіанти ResNet, зазвичай забезпечують вищу точність, але вимагають більше часу на обробку та значних обчислювальних ресурсів, що робить їх менш придатними для застосувань у реальному часі на периферійних пристроях.

Іншим класом моделей є двоетапні детектори об'єктів [20], які реалізовують процес виявлення у два послідовні етапи: генерацію пропозицій регіонів, що потенційно містять об'єкти, та їхню подальшу класифікацію з уточненням обмежувальних рамок. Однією з перших успішних реалізацій стала R-CNN, яка використовувала зовнішній алгоритм для генерації пропозицій, що оброблялися згорткою незалежно. Низька швидкість обробки була її основним недоліком. Наступна покращена модель Fast R-CNN значно пришвидшила процес, застосовуючи згортку до всього зображення одноразово для отримання глобальної карти ознак. Ознаки для кожної пропозиції, все ще генерованої зовнішнім алгоритмом, вилучалися за допомогою шару RoI Pooling. Класифікація та регресія обмежувальних рамок виконувалися спільно. Дана версія моделі інтегрувала етап генерації пропозицій, замінивши зовнішній алгоритм на внутрішню мережу пропозицій регіонів (RPN).

RPN генерує пропозиції безпосередньо з карти ознак базової згортки. Ці пропозиції потім обробляються аналогічно Fast R-CNN.

Еволюція сімейства R-CNN демонструє тенденцію до наскрізного навчання та підвищення обчислювальної ефективності шляхом інтеграції окремих етапів обробки в архітектуру нейронної мережі. Така інтеграція сприяє оптимізації всіх компонентів системи. Двоетапні детектори відомі високою точністю виявлення, особливо малих або складних об'єктів, та якісною локалізацією, проте зазвичай поступаються в швидкості одноетапним детекторам.

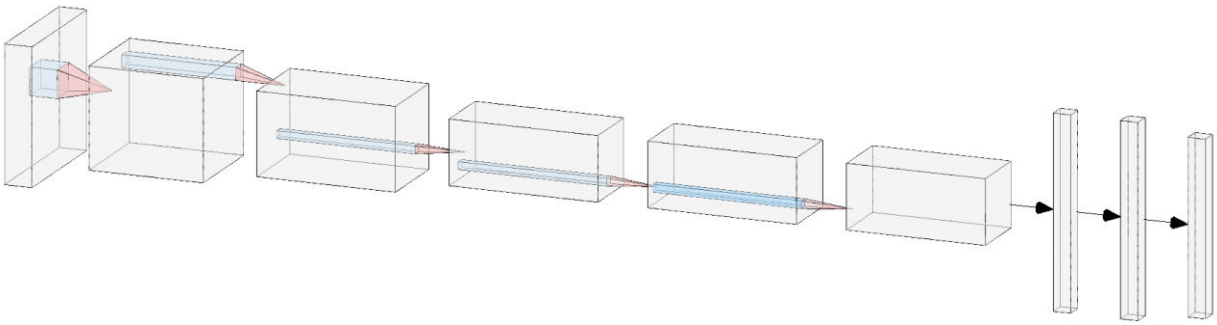

Рисунок 1.4. Архітектура згорткової нейромережі за прикладом AlexNet

Одноетапні детектори являють собою клас архітектур нейронних мереж, що здійснюють одночасну локалізацію та класифікацію об'єктів за один прохід обробки зображення. Цей підхід забезпечує значно вищу швидкість обробки порівняно з двоетапними детекторами, що робить їх придатними для застосувань у реальному часі. Найпопулярніша модель YOLO є однією з найбільш відомих архітектур одноетапних детекторів, яка розглядає задачу виявлення як проблему регресії. Зображення поділяється на сітку комірок, кожна з яких прогнозує обмежувальні рамки для об'єктів, чиї центри потрапляють у цю комірку, а також імовірності класів. YOLO вирізняється високою швидкістю обробки, що робить його ефективним для відеоаналітики в реальному часі. Подальші версії YOLO

впроваджують удосконалення для підвищення точності, зокрема у виявленні малих об'єктів.

SSD є ще одним представником одноетапних детекторів, що використовує карти ознак з різних шарів базової згорткової моделі для виявлення об'єктів різних масштабів. Для кожної комірки на цих картах SSD прогнозує зміщення відносно набору стандартних обмежувальних рамок різних розмірів та співвідношень сторін, а також імовірності класів. RetinaNet була розроблена для вирішення проблеми дисбалансу між фоновими та передньоплановими прикладами при навчанні одноетапних детекторів. Вона використовує мережу пірамід ознак (FPN) для ефективного виявлення об'єктів на різних масштабах та фокусну функцію втрат, яка зменшує вплив легко класифікованих прикладів, фокусуючи навчання на складних об'єктах.

Одноетапні детектори значно змінили підхід до виявлення об'єктів у реальному часі, переформулювавши його як задачу регресії. Це забезпечило значне зростання швидкості, хоча на початковому етапі супроводжувалося певною втратою точності, особливо для малих або перекритих об'єктів. Подальші інновації, такі як FPN та архітектурні вдосконалення в наступних версіях YOLO, спрямовані на мінімізацію цього компромісу, зберігаючи при цьому переваги у швидкості.

Останнім часом спостерігається тенденція до застосування архітектур на основі трансформерів, зображеної на Рисунку 1.5 [47], у комп'ютерному зорі, зокрема у виявленні об'єктів. DETR є показовим прикладом, який розглядає виявлення як задачу прямого передбачення множини. DETR використовує енкодер-декодерну архітектуру трансформера разом із згортковою моделлю для вилучення ознак, усуваючи необхідність у багатьох ручних компонентах, таких як якірні

рамки та постобробка NMS.

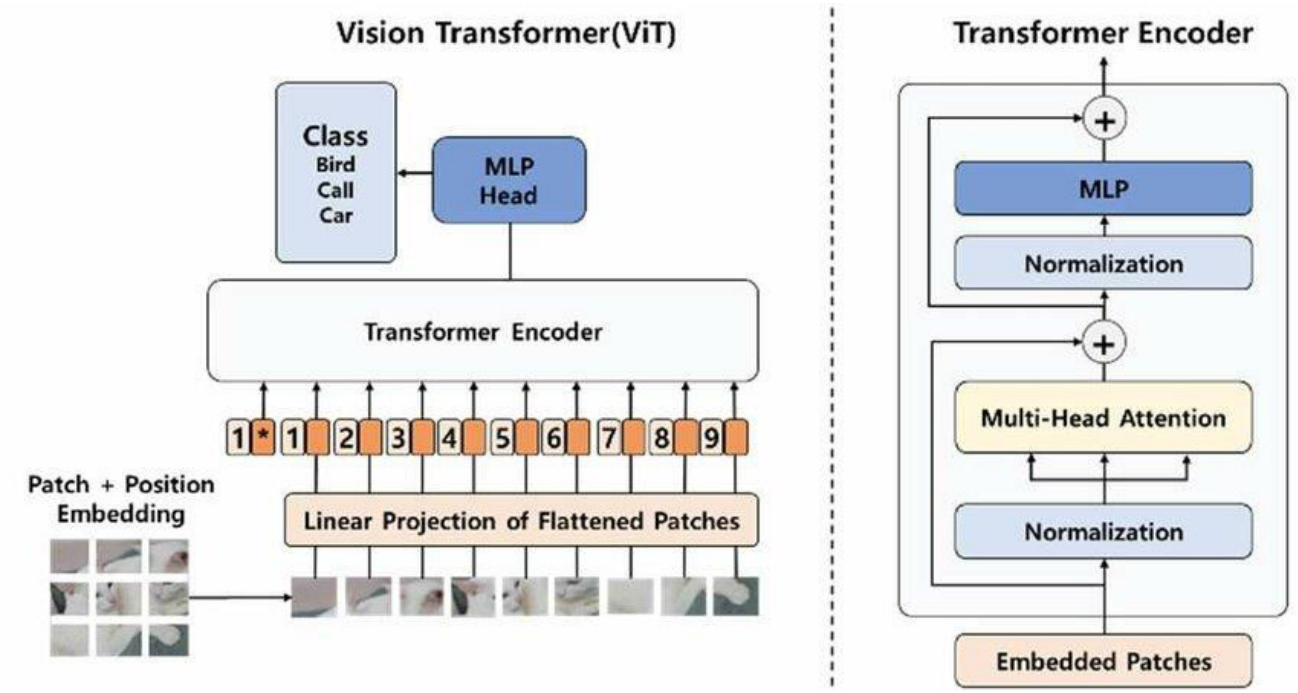

Рисунок 1.5. Архітектура Vision Transformers [47]

Моделі на основі трансформерів демонструють багатообіцяючі результати завдяки здатності захоплювати глобальний контекст зображення за допомогою механізмів уваги. Однак на даному етапі вони часто вимагають значних обчислювальних ресурсів та великих обсягів даних для навчання. Їхня продуктивність у виявленні малих об'єктів може бути нижчою порівняно з оптимізованими згортковими моделями, і вони залишаються активною областю досліджень та оптимізації для практичного застосування. Застосування трансформерів у комп'ютерному зорі свідчить про зближення архітектурних парадигм з різних галузей штучного інтелекту, відкриваючи нові перспективи для розробки потужніших систем виявлення.

Таблиця 1.1. Порівняльна характеристика ключових моделей виявлення об’єктів

Модель	Тип	Типовий FPS	Архітектурні особливості	Переваги	Недоліки
Faster R-CNN	Двоетапний	~5-15 FPS	RPN, RoI Pooling	Висока точність локалізації, добре для малих/складних рухомих об'єктів	Низька швидкість, може не встигати за дуже швидкими об'єктами
YOLO v3	Одноетапний	~30-60 FPS	Сітка, пряма регресія рамок, багатомасштабні виходи	Дуже висока швидкість, добре для відстеження в реальному часі	Може мати нижчу точність для дуже малих або сильно перекритих об'єктів
YOLO v11	Одноетапний	~50-200 FPS	Покращена архітектура (C2f, SPPF), безякірний підхід (anchor-free), гнучкість масштабування	Висока швидкість та покращена точність порівняно з YOLOv3, гнучкість	Складність виявлення дуже дрібних об'єктів все ще може бути присутня
SSD	Одноетапний	~20-50 FPS	Багатомасштабні карти ознак, стандартні рамки (default boxes)	Хороша швидкість, добре виявляє об'єкти різного розміру	Точність може бути нижчою, ніж у двоетапних, особливо для малих об'єктів
RetinaNet	Одноетапний	~10-20 FPS	FPN, Focal Loss	Хороший баланс точності та швидкості,	Швидкість нижча, ніж у YOLO/SSD

				добре працює з різними масштабами	
Visual Transformer (ViT)	Новітній, енкодер + декодер	~10-20 FPS	Розбиття зображення на батчі, застосування трансформерів	Захоплення глобального контексту, потенційна стійкість до зсувів	Високі обчислювальні вимоги, потреба у великих обсягах даних для навчання

Продовження Таблиці 1.1.

1.4. Аналіз методів застосування в задачі виявлення рухових об’єктів в сфері оборони

Нейронні мережі та методи виявлення об'єктів набувають все більшого значення в сучасних військових операціях, зокрема з поширеним використанням БПЛА. Ключові сценарії застосування охоплюють ідентифікацію та відстеження таких цілей як: бойової техніки, артилерійських систем, самохідних установок. Автоматизований аналіз даних розвідки, спостереження та рекогносцировки, автономне наведення боєприпасів та прикордонний контроль для виявлення незаконної діяльності. Військова сфера є значним рушієм у розробці надійних, швидких та автономних систем виявлення об'єктів з БПЛА і саме це стимулює дослідження у напрямку підвищення автономності систем та їхньої здатності функціонувати у складних та ворожих умовах.

Для військових завдань загальні моделі виявлення об'єктів, такі як YOLO, SSD та Faster R-CNN, часто адаптуються або донавчаються на специфічних військових наборах даних. Особлива увага приділяється сенсорній фузії, яка передбачає поєднання даних з різних сенсорів таких

як RGB-камер, інфрачервоних камер, радарів, LiDAR, радіочастотних та акустичних сенсорів для підвищення надійності виявлення та відстеження. Нейронні мережі використовуються для інтеграції та спільної обробки цих різнорідних даних, що компенсує недоліки окремих сенсорів та підвищує загальну ефективність системи, особливо у контексті протидії БПЛА. Розвиток спрямований на мультимодальну сенсорну фузію та методи штучного інтелекту, здатні робити висновки в умовах невизначеності, наприклад, нейросимвольний ШІ, що вказує на перехід до більш цілісного сприйняття та аналізу сцени.

Для навчання та оцінки моделей використовуються різні набори даних, такі як DOTA (аерознімки з різними категоріями об'єктів), UAVDT (виявлення та відстеження транспортних засобів з БПЛА у міських умовах) та спеціалізовані набори даних для протидії БПЛА (DroneRF, Halmstad Drone, Anti-UAV Challenge datasets та інші). Також використовуються спеціалізовані набори даних, створені для конкретних завдань. Прикладами застосування є адаптація YOLOv11 для виявлення військових об'єктів на основі DOTA, використання YOLOv8 для виявлення та відстеження дронів та застосування YOLO для ідентифікації бойових машин у реальному часі.

Військове використання БПЛА для ідентифікації рухомих об'єктів характеризується низкою специфічних та складних перешкод. Однією з основних проблем є виявлення малорозмірних, високодинамічних або непередбачувано маневруючих цілей, які на значній відстані з висоти польоту можуть мати мініатюрні розміри, що суттєво ускладнює їхню детекцію та точну локалізацію. Додаткові труднощі створює активне застосування противником засобів маскування, спрямованих на зниження візуальної, теплової та радіолокаційної помітності об'єктів, що вимагає

розробки методів, стійких до камуфляжу. Гетерогенність ландшафтів, де часто відбуваються військові операції формує складне та неоднорідне тло, що ускладнює відділення цілей від оточення. Забезпечення обробки інформації в режимі реального часу безпосередньо на борту БПЛА є критично важливим, проте ускладнюється обмеженими обчислювальними ресурсами, енергоспоживанням та пропускною здатністю каналів зв'язку, що вимагає розробки високоефективних та оптимізованих алгоритмів. Значною проблемою є також дефіцит різноманітних та якісно анотованих військових наборів даних, оскільки збір та розмітка інформації, що відображає реалістичні військові сценарії та сучасне озброєння, є складним, ресурсомістким та часто конфіденційним процесом. Крім того, системи виявлення повинні демонструвати стійкість до ворожих засобів радіоелектронної боротьби, зберігаючи працездатність в умовах спроб противника створити перешкоди або підмінити сигнали навігаційних систем, каналів зв'язку та сенсорів

1.5. Постановка задачі дослідження

Ключовим завданням цього дослідження є розробка системи виявлення рухомих об’єктів та адаптація методів машинного навчання. Важливим аспектом є забезпечення високої точності та швидкодії процесу здатність виявляти різноманітні типи об'єктів. Для досягнення поставленої мети передбачається розв'язання низки взаємопов'язаних підзадач, що включає дослідження та аналіз існуючих методологій виявлення рухомих об'єктів, вибір та реалізацію архітектур нейронних мереж, що є найбільш придатними для обробки відеоданих з БПЛА, а також розробку ефективних стратегій навчання моделей з урахуванням специфіки вхідних даних та вимог до продуктивності системи в реальному часі. Особлива увага буде приділена формалізації бази знань

про характеристики рухомих об'єктів, що впливають на процес їх ідентифікації. Важливим етапом є експериментальна оцінка розробленої системи з використанням релевантних метрик якості.

ВИСНОВКИ ЗА РОЗДІЛОМ

У розділі І розглянуто проблему ідентифікації рухомих об'єктів на основі відеоданих, що є актуальним завданням у різних сферах, зокрема при використанні БПЛА. Проаналізовано основні складнощі, пов'язані з динамікою зображень та варіативністю умов спостереження. Розглянуто сучасні підходи машинного навчання, що застосовуються для виявлення об'єктів, включаючи різні архітектури нейронних мереж та їхні характеристики. Окрему увагу приділено специфічним вимогам та обмеженням, що виникають у певних областях застосування. На основі проведеного аналізу існуючих методів, визначено ключові аспекти, які необхідно враховувати при розробці ефективних систем ідентифікації рухомих об'єктів.

РОЗДІЛ ІІ. СТРУКТУРИ ТА ОСОБЛИВОСТІ АРХІТЕКТУР МЕРЕЖ ДЛЯ ІДЕНТИФІКАЦІЇ РУХОМИХ ОБ’ЄКТІВ

2.1. Модель YOLO.

В основі парадигми YOLO лежить фундаментальна концепція одноетапної детекції, яка якісно відрізняється від попередніх двохетапних підходів. На противагу послідовній генерації потенційних регіонів інтересу з подальшою їх класифікацією, YOLO інтегрує процес детекції об'єктів у єдину задачу регресії. Нейронна мережа здійснює обробку всього вхідного зображення за один прямий прохід, одночасно прогнозуючи параметри обмежувальних рамок та відповідні ймовірності що належать до визначених класів для всіх наявних об'єктів. Такий підхід забезпечує значне підвищення швидкості обробки, що робить дану архітектуру високоефективною для застосувань, які вимагають оперативного реагування, зокрема при аналізі відеопотоку з безпілотних літальних апаратів для відстеження рухомої військової техніки.

Ключовим елементом функціонування YOLO є дискретизація вхідного зображення шляхом його поділу на умовну сітку розмірністю $S \times S$ комірок. Кожна комірка цієї сітки набуває відповідальності за виявлення тих об'єктів, чиї геометричні центри потрапляють у межі даної комірки. Отже, якщо центр об'єкта локалізується в певній комірці сітки, саме ця комірка ініціює генерацію прогностичних даних для цього об'єкта. Для кожної комірки сформованої сітки модель YOLO здійснює прогнозування фіксованої кількості обмежувальних рамок, кожна з яких описується п'ятьма параметрами:

$$(x, y, w, h, confidence) \quad (1)$$

Координати центру обмежувальної рамки прогнозуються відносно лівого верхнього кута відповідної комірки сітки та нормалізуються розмірами цієї комірки. Ширина та висота обмежувальної рамки прогнозуються відносно розмірів всього вхідного зображення та підлягають аналогічній нормалізації. Параметр confidence являє собою оцінку впевненості моделі в тому, що прогнозована рамка дійсно охоплює об'єкт, а також відображає точність відповідності між прогнозованою та реальною обмежувальною рамкою об'єкта. Математично оцінка впевненості визначається як добуток ймовірності наявності об'єкта в межах комірки та значення метрики Intersection over Union (IoU) між спрогнозованою рамкою та істинною обмежувальною рамкою об'єкта.

$$IoU = \frac{TP}{(TP+FP+FN)} \tag{2}$$

$$\text{Mean } IoU = \frac{1}{C}\sum_C IoU_C \tag{3}$$

IoU, у свою чергу, є кількісною мірою перекриття двох рамок, що обчислюється як відношення площі їх перетину до площі їх об'єднання. Окрім прогнозування обмежувальних рамок, кожна комірка сітки також генерує ймовірності належності виявленого об'єкта до кожного з C можливих класів.

Принциповою особливістю архітектури YOLO є передбачення лише одного набору з C умовних ймовірностей класів:

$$P(Class_i|Object) \tag{4}$$

для кожної комірки сітки, незалежно від кількості (B) обмежувальних рамок, що нею прогнозуються. Це означає, що всі B рамок, асоційованих з певною коміркою, матимуть однаковий розподіл ймовірностей класів.

Такий підхід накладає певне обмеження при обробці зображень, на яких центри кількох об'єктів різних класів потрапляють в одну й ту саму комірку сітки. У таких випадках модель здатна коректно класифікувати лише один з цих об'єктів.

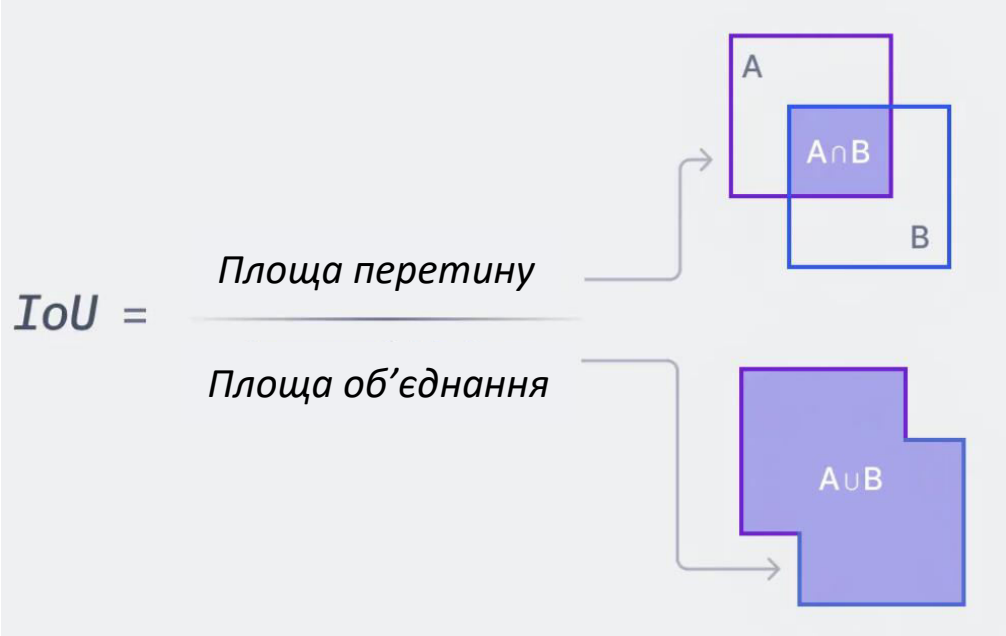

Рисунок 2.1. Принцип розрахунку IoU

Це обмеження особливо відчутне при детекції щільних скупчень дрібних об'єктів, наприклад, при виявленні групи військової техніки або особового складу, де центри кількох об'єктів різних типів можуть опинитися в межах однієї комірки. Модель буде змушена визначити домінуючий клас для цієї комірки, що може призвести до пропуску або неправильної класифікації інших об'єктів у цій же області.

Незважаючи на високу швидкість обробки, властиву одноетапному підходу, ранні ітерації YOLO демонстрували дещо нижчу точність локалізації об'єктів порівняно з двохетапними детекторами. Це пояснюється тим, що двохетапні методи спочатку генерують пропозиції регіонів, а потім здійснюють їх детальну класифікацію, що дозволяє

більш точно налаштовувати параметри обмежувальних рамок. YOLO, виконуючи всі операції за один прохід, оптимізує швидкість, проте регресія параметрів рамок відбувається одночасно з класифікацією на відносно грубій сітці, що може негативно впливати на точність їх позиціонування. Цей компроміс між швидкістю та точністю є важливим фактором при виборі моделі для застосування на БПЛА, де швидкість реакції є критично важливою, але помилки в локалізації цілей можуть мати серйозні наслідки.

Подальша обробка результатів детекції включає застосування алгоритму немаксимального пригнічення [NMS]. У процесі роботи YOLO часто генерує множинні обмежувальні рамки для одного й того ж об'єкта, особливо якщо об'єкт має значні розміри або розташований на межі кількох комірок сітки. Для усунення цих надлишкових детекцій та формування єдиної, оптимальної рамки для кожного виявленого об'єкта використовується NMS. Алгоритм NMS сортує всі запропоновані рамки на основі їхніх оцінок впевненості. Рамка з найвищою оцінкою впевненості вибирається як остаточна детекція, а всі інші рамки, що мають значне перекриття з нею, пригнічуються, тобто видаляються. Цей ітеративний процес повторюється для решти рамок до тих пір, поки не залишаться лише найбільш впевнені та просторово непересічні детекції. Застосування NMS є критично важливим етапом постобробки, що значно підвищує чіткість та точність кінцевих результатів детекції об'єктів.

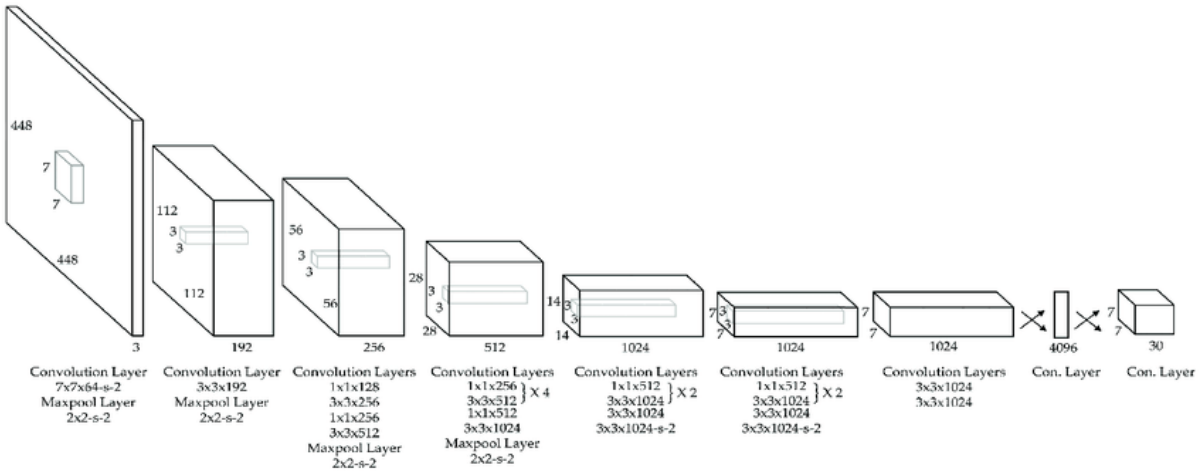

Рисунок 2.2. Архітектура моделі YOLO [48]

На Рисунку 2.2 [48] зображена архітектура YOLOv3, яка є наразі досить застарілою, проте гарно описує принцип її роботи. Архітектура сучасних моделей YOLO, подібно до багатьох інших детекторів об'єктів, зазвичай складається з трьох основних функціональних блоків: основи [Backbone], шиї [Neck] та голови [Head].

Основа являє собою згорткову нейронну мережу, відповідальну за вилучення ієрархічного набору карт ознак з вхідного зображення. Як правило, в якості основи використовуються попередньо навчені на великих наборах даних для класифікації зображень мережі, такі як Darknet, CSPDarknet або інші ефективні архітектури. Процес обробки в основі характеризується поступовим зменшенням просторової роздільної здатності зображення при одночасному збільшенні глибини карт ознак, що дозволяє фіксувати як низькорівневі характеристики, так і високорівневі семантичні ознаки об'єктів. Шия виконує функцію сполучної ланки між основою та головою. Її основне завдання полягає в агрегації та комбінуванні ознак, отриманих з різних шарів основи. Це дозволяє збагатити представлення ознак шляхом об'єднання семантично насичених ознак з глибших шарів основи з ознаками високої роздільної

здатності з більш ранніх шарів. Поширеними архітектурами для шиї є FPN та Path Aggregation Network [PANet], які забезпечують ефективний потік інформації між ознаками різних масштабів. Голова відповідає за формування кінцевих прогнозів на основі оброблених ознак, що надходять від шиї. У випадку YOLO, голова генерує тензор, що містить прогнози обмежувальних рамок, зображених на Рисунку 2.3 [1], а саме координати, ширина, висота, оцінки впевненості для кожної рамки та ймовірності класів для кожної комірки сітки. Структура голови може варіюватися залежно від конкретної версії YOLO, проте її основна функція полягає у перетворенні багаторівневих ознак, отриманих від шиї, у кінцеві результати детекції об'єктів.

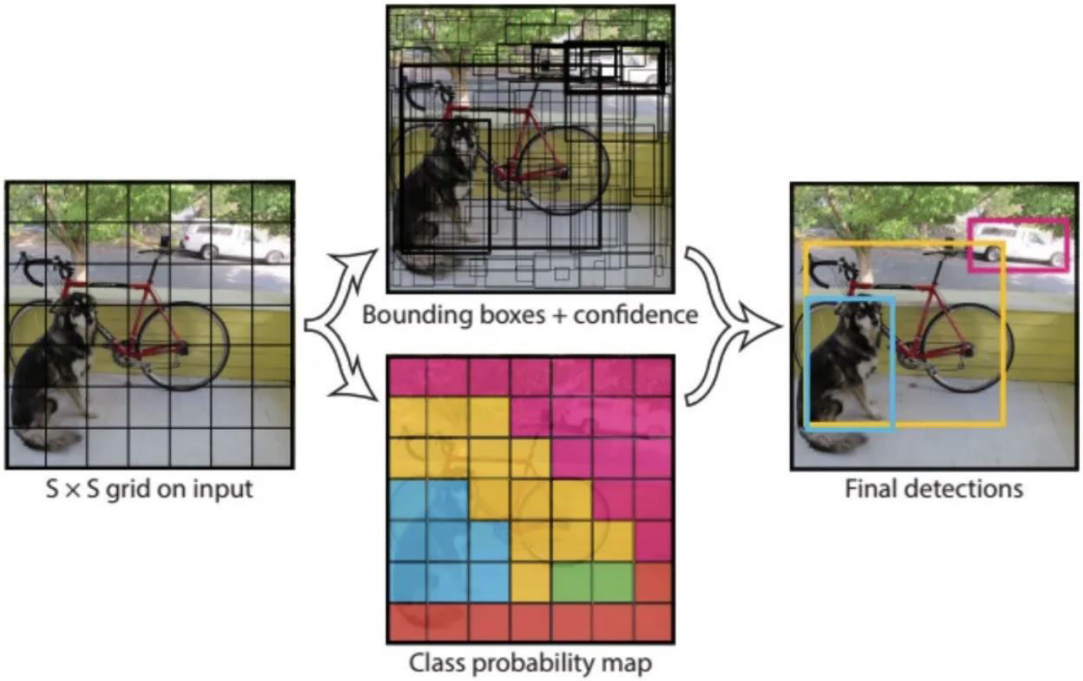

Рисунок 2.3. Принцип роботи обмежувальний рамок та класифікація виділення рамок за класом [1]

В новітніх версіях моделі присутні нові методи, які допомагають збільшувати продуктивність моделей. Однією з ключових модифікацій в архітектурі основи є впровадження модуля C2f на заміну модулю C3 [5].

Принципова відмінність полягає в стратегії обробки вихідних даних внутрішніх основних модулів. У той час як модуль C3 використовував лише вихід останнього блоку основи, модуль C2f здійснює об'єднання виходів усіх модулів основи, інтегрованих у його структуру. Такий підхід сприяє збагаченню потоку ознак та градієнтів у межах мережі, що позитивно впливає на ефективність навчання та потенційно скорочує час тренування. Зазначена архітектурна зміна забезпечує більш раціональне використання ознак, отриманих на різних етапах обробки всередині блоку.

YOLO також характеризується переходом до повністю без'якірного підходу до виявлення об'єктів. На відміну від попередніх ітерацій, які використовували набір попередньо визначених якірних рамок для прогнозування положення та розміру об'єктів, YOLO здійснює пряме прогнозування характеристик об'єкта, таких як координати його центральної точки та розміри обмежувальної рамки [3]. Цей методологічний зсув зумовлює низку переваг таких. Перший та найважливіший це покращена генералізація: відмова від фіксованих якорів зменшує залежність моделі від статистичних характеристик розмірів та співвідношень сторін об'єктів у навчальному наборі даних, що потенційно підвищує її здатність до узагальнення на нових, раніше невідомих даних або на спеціалізованих наборах даних з унікальними об'єктами. Спрощення конвеєра обробки: усувається необхідність у ретельному виборі та оптимізації якірних рамок, що спрощує етапи розробки та налаштування моделі. Прискорення післяобробки: без'якірний підхід часто призводить до зменшення загальної кількості вихідних прогнозів моделі, що, в свою чергу, може сприяти пришвидшенню роботи алгоритму немаксимального пригнічення, який застосовується для фільтрації надлишкових виявлень [6].

Ефективне застосування моделі YOLO у військовій галузі [22,23], зокрема для задач виявлення рухомої техніки з БПЛА, вимагає не лише вибору базової архітектури, але й критично важливого етапу спеціалізованого донавчання. Цей процес передбачає використання великих та репрезентативних наборів даних, що відображають специфічні типи об'єктів та сценарії, характерні для реальних умов експлуатації. Застосування стандартних моделей, попередньо навчених на загальних наборах даних, таких як COCO, ймовірно призведе до незадовільних результатів при обробці зображень військових цілей, враховуючи їх різноманітність, використання камуфляжу, специфічні ракурси з БПЛА та складні метеорологічні умови. Дослідження, як-от розробка YOLO-E або MSC-YOLO [21], підкреслюють необхідність створення спеціалізованих наборів даних та модифікації існуючих архітектур для вирішення конкретних завдань, включаючи детекцію замаскованих об'єктів або малорозмірних цілей на аерофотознімках.

Крім того, слід зазначити, що навіть "легкі" версії YOLO можуть виявитися надмірно ресурсовитратними для деяких БПЛА з обмеженими обчислювальними можливостями. Це зумовлює активні дослідження в напрямку квантування моделей, прунінгу та інтеграції спеціалізованих апаратних прискорювачів нейронних мереж на борту БПЛА. Зазначені підходи виходять за межі простого вибору архітектури, проте є ключовими практичними аспектами для забезпечення функціональності систем детекції на безпілотних платформах у реальних умовах.

2.2. Модель Faster R-CNN.

Faster R-CNN є однією з фундаментальних та надзвичайно впливових архітектур у галузі детекції об'єктів, що належить до сімейства двоетапних детекторів. Ця модель стала значним прогресом порівняно зі своїми попередниками R-CNN та Fast R-CNN, завдяки впровадженню ефективного механізму генерації пропозицій регіонів безпосередньо в межах нейронної мережі. Перший етап полягає у генерації пропозицій регіонів. На цьому етапі спеціалізований модуль, відомий як мережа генерування пропозицій, здійснює аналіз карти ознак, отриманої від базової згорткової мережі, та формує набір прямокутних можливих областей. Ці пропозиції являють собою ділянки зображення, які, за оцінкою мережі, з високою ймовірністю можуть містити об'єкти. Другий етап включає класифікацію та уточнення пропозицій. На цьому етапі згенеровані пропозиції регіонів використовуються детектором, подібним до Fast R-CNN. Для кожної пропозиції відбувається вилучення ознак, які потім використовуються класифікатором для визначення класу об'єкта та регресором для точного налаштування координат обмежувальної рамки.

Впровадження мережі генерування пропозицій [МГП], яка розділяє обчислення згорткових шарів з основною детекторною мережею, стало вирішальним кроком у розвитку архітектури. Це дозволило усунути недоліки продуктивності, характерне для попередніх моделей R-CNN та Fast R-CNN, де генерація пропозицій регіонів покладалася на повільні зовнішні алгоритми, такі як селективний пошук, що часто виконувалися на центральному процесорі. Завдяки такому підходу генерація пропозицій стала значно ефективнішою з точки зору обчислювальних витрат, оскільки вона повторно використовує вже обчислені згорткові ознаки, що суттєво прискорило загальний процес виявлення. Для генерації пропозицій регіонів різних розмірів та співвідношень сторін

МГП використовує концепцію якірних рамок, зображених на Рисунку 2.5 [50]. У кожній позиції ковзного вікна на карті ознак розміщується набір k якірних рамок, які мають попередньо визначені масштаби та співвідношення сторін. Якори слугують референсними рамками, відносно яких МГП прогнозує кінцеві пропозиції регіонів.

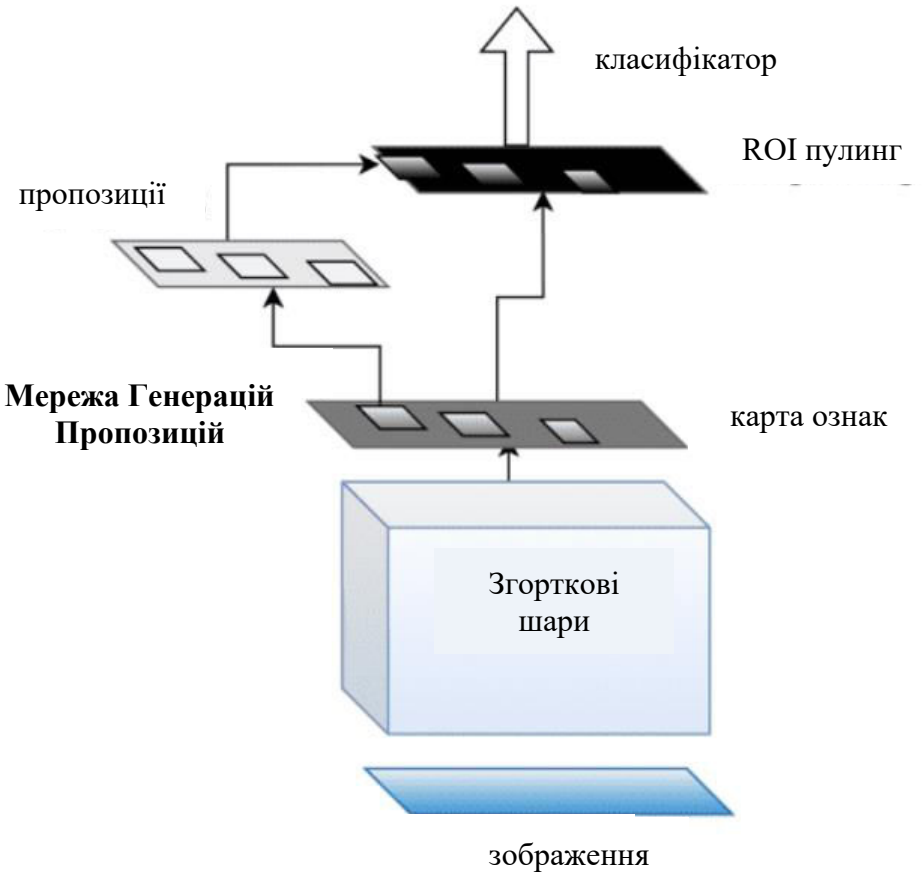

Рисунок 2.4. Архітектура моделі Faster R-CNN

На відміну від YOLO, де якорі визначалися за допомогою кластеризації, у Faster R-CNN вони зазвичай є фіксованими та розробленими для охоплення типових форм та розмірів об'єктів у використовуваному наборі даних. Для кожного з k якорів у кожній позиції ковзного вікна МГП прогнозує два типи вихідних даних: оцінку об'єктності, яка є результатом бінарної класифікації та вказує на

ймовірність того, що якір містить об'єкт (позитивний приклад) або є фоном (негативний приклад), та регресію координат рамки, яка представлена чотирма значеннями, що відображають поправки до координат відповідного якоря для більш точного охоплення потенційного об'єкта. Згенеровані МГП пропозиції регіонів можуть мати різні розміри та співвідношення сторін. Однак, наступні повнозв'язні шари в детекторі Fast R-CNN вимагають на вхід ознаки фіксованого розміру. Для вирішення цієї проблеми застосовується шар RoI Pooling. RoI Pooling використовує карту ознак та координати кожної пропозиції регіону, поділяє цю пропозицію на фіксовану кількість під-регіонів та застосовує операцію максимального пулінгу до кожного підрегіону. Завдяки цьому, для кожної пропозиції регіону генерується вектор ознак фіксованої довжини. Проте, RoI Pooling вносить операції квантування при відображенні координат пропозиції регіону на карту ознак та при поділі регіону на підрегіони. Це може призвести до незначних зміщень та втрати просторової точності, що є особливо критичним для детекції малих об'єктів або для завдань, що вимагають високої точності локалізації, таких як сегментація.

Для подолання цього обмеження було розроблено шар RoI Align, вперше представлений у моделі Mask R-CNN. RoI Align уникає будь-якого квантування координат, використовуючи білінійну інтерполяцію для обчислення точних значень ознак у фіксованій кількості точок вибірки всередині кожної під-комірки пропозиції регіону. Потім до цих значень застосовується операція агрегації. Це забезпечує значно краще вирівнювання між вилученими ознаками та оригінальним регіоном інтересу, що призводить до підвищення точності детекції та сегментації. Еволюція від RoI Pooling до RoI Align ілюструє, як навіть незначні

неточності на проміжних етапах обробки ознак можуть суттєво впливати на кінцеву якість детекції.

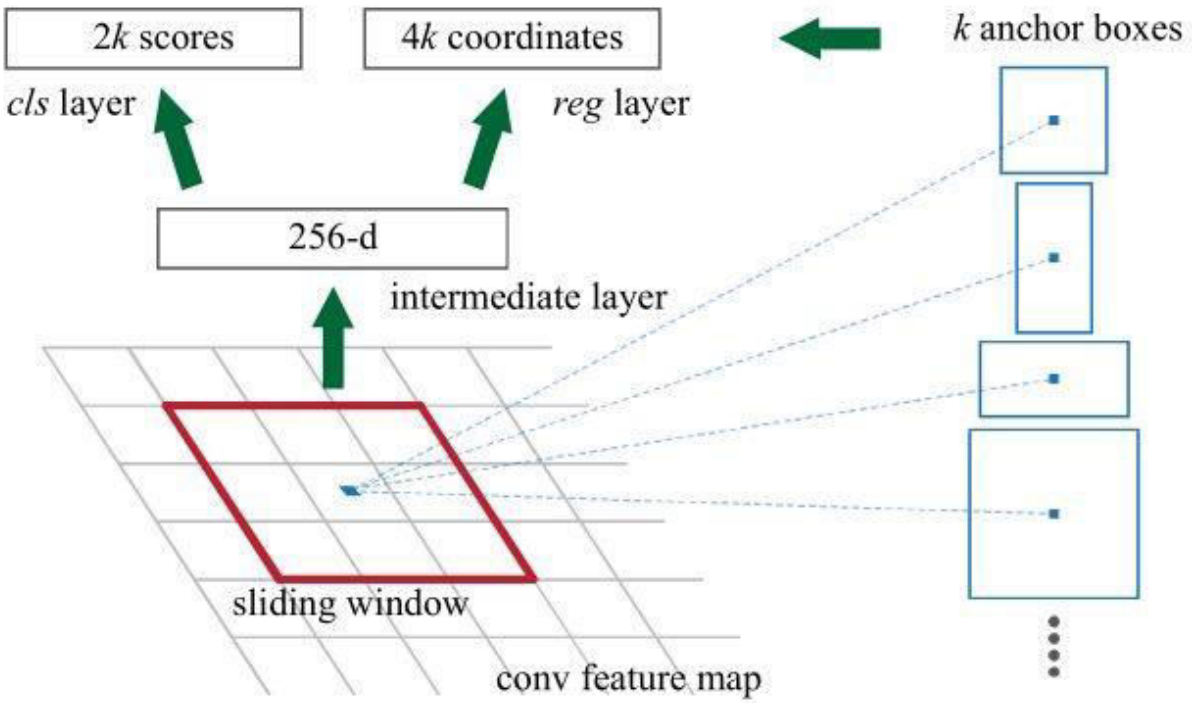

Рисунок 2.5. Приклад використання якірних рамок (anchor boxes) [50]

У військових застосуваннях, де точність локалізації є критично важливою, покращення, що надає RoI Align, є значущим, наприклад, при точному визначенні меж малорозмірних безпілотних літальних апаратів або вразливих зон бронетехніки. Після приведення ознак кожної пропозиції регіону до фіксованого розміру за допомогою RoI Pooling або RoI Align, вони подаються на вхід кількох повнозв'язних шарів. Ці шари, у свою чергу, розгалужуються на дві паралельні "голови": класифікаційну голову, яка прогнозує клас об'єкта всередині пропозиції регіону, зазвичай використовуючи функцію softmax для видачі ймовірностей по K класам об'єктів плюс один додатковий клас для фону, та регресійну голову, яка прогнозує остаточні поправки до координат обмежувальної рамки для кожного з K класів об'єктів, щоб забезпечити ще точнішу локалізацію.

2.3. Модель RetinaNet

RetinaNet є впливовою архітектурою одностадійної детекції об'єктів, ключовою інновацією якої стало ефективне вирішення проблеми значного дисбалансу між класами переднього плану та фону, що дозволило досягти точності, порівнянної з двостадійними аналогами.

RetinaNet, подібно до YOLO, є одностадійним детектором, що здійснює одночасне прогнозування обмежувальних рамок та класифікацію об'єктів за один прохід мережею, без окремого етапу генерації пропозицій регіонів. Розроблена для ефективної "щільної" детекції об'єктів, RetinaNet аналізує велику кількість потенційних локацій, систематично відібраних по всьому зображенню за допомогою анкерних рамок на різних рівнях піраміди ознак.

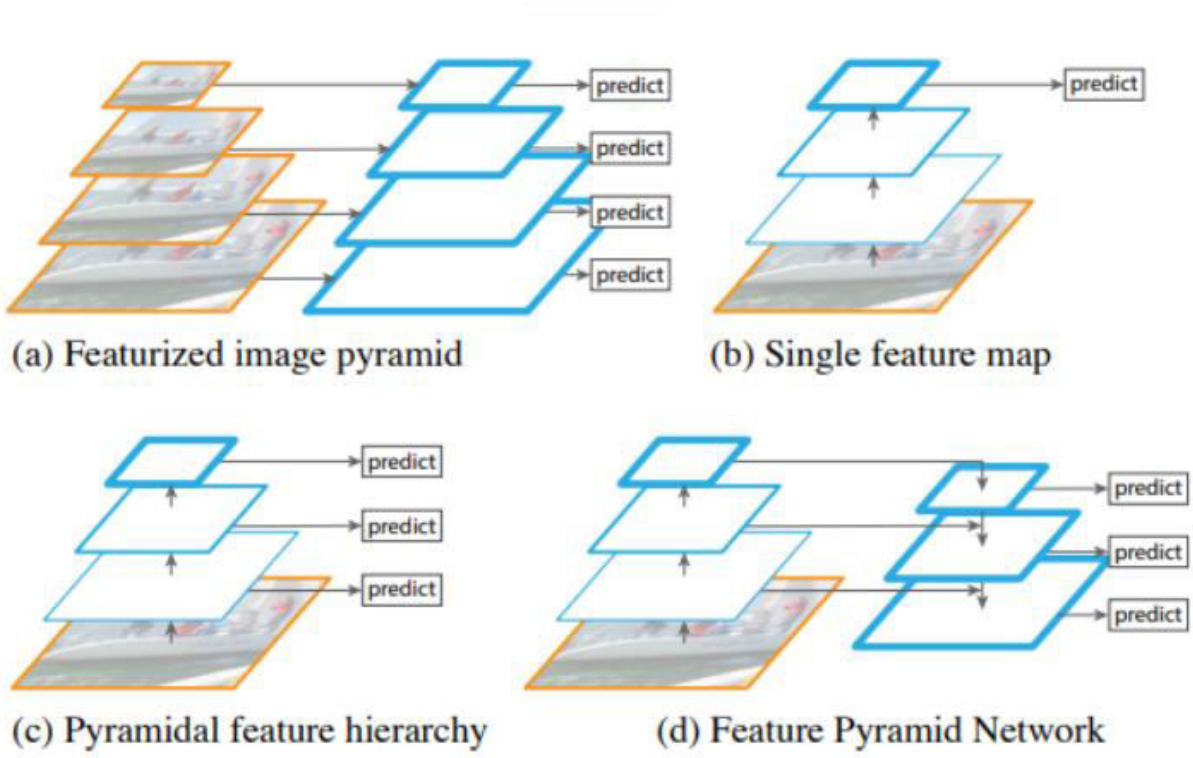

Рисунок 2.6. Принцип роботи мережі пірамід ознак (FPN)$s^{[52]}$

Успіх RetinaNet ґрунтується на двох основних компонентах: мережі пірамід ознак, зображена на Рисунку 2.6 [52], яка забезпечує вилучення багаторівневих карт ознак для детекції об'єктів різних масштабів шляхом поєднання семантично сильних ознак з низькою роздільною здатністю та ознак високої роздільної здатності зі збереженням просторової інформації, та фокусній функції втрат, спеціально розробленій для боротьби з екстремальним дисбалансом класів шляхом динамічного зменшення ваги легко класифікованих фонових прикладів, що дозволяє моделі зосередитися на складних випадках. Саме фокальна функція втрат стала ключовим елементом, що дозволив одностадійним детекторам, таким як RetinaNet, значно підвищити точність порівняно з двостадійними підходами, які раніше демонстрували кращі результати через проблему перенасичення навчання великою кількістю легких негативних прикладів.

Архітектура RetinaNet включає базову мережу [backbone], мережу пірамід ознак та дві паралельні підмережі для виконання завдань класифікації та регресії обмежувальних рамок. В якості основної мережі RetinaNet використовує потужну згорткову нейронну мережу, таку як ResNet, попередньо навчену на великих наборах даних для класифікації зображень. Ця мережа відповідає за вилучення ієрархічних карт ознак з вхідного зображення на різних рівнях з різною просторовою роздільною здатністю. Мережа пірамід ознак є інтегральною частиною RetinaNet і призначена для створення багаторівневої піраміди ознак, що ефективно представляє об'єкти різних масштабів. FPN складається з висхідного шляху, що являє собою стандартне пряме поширення сигналу через базову мережу з генерацією карт ознак на кожному етапі зі зменшенням просторової роздільної здатності та посиленням семантичних ознак; низхідного шляху, що починається з найбільш семантично сильної карти

ознак та послідовно збільшує її просторову роздільну здатність шляхом апсемплінгу; та бічних з'єднань, де карти ознак з низхідного шляху об'єднуються з відповідними картами ознак з висхідного шляху однакової просторової роздільної здатності. Це збагачує семантично сильні ознаки точною просторовою інформацією, формуючи набір карт ознак на різних рівнях піраміди, призначених для детекції об'єктів певного масштабу.

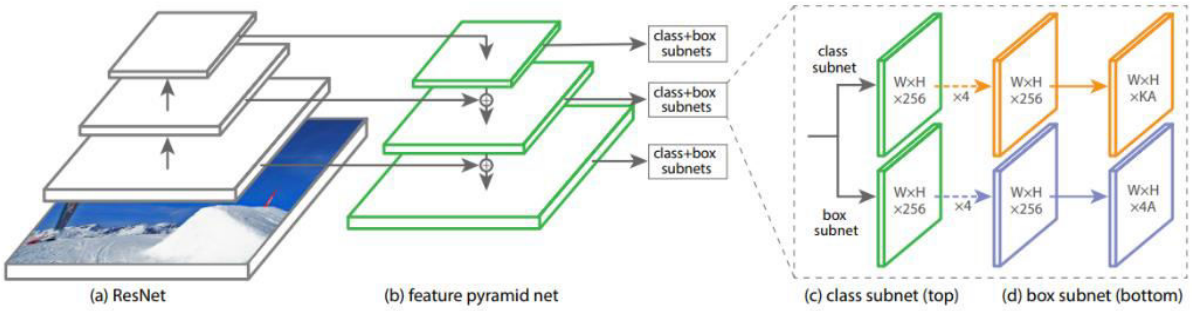

Рисунок 2.7. Архітектура моделі RetinaNet

До кожного рівня піраміди ознак приєднані дві ідентичні за структурою, але з різними навченими параметрами, невеликі повністю згорткові підмережі для класифікації та регресії обмежувальних рамок. Підмережа класифікації приймає карту ознак з певного рівня FPN та для кожної просторової позиції і кожного з A якорів прогнозує ймовірності належності до K класів об'єктів, генеруючи $A \times K$ виходів на кожній позиції. Підмережа регресії обмежувальних рамок також приймає ту саму карту ознак і для кожної просторової позиції та кожного з A якорів прогнозує 4 зміщення відносно анкерної рамки до реальної рамки об'єкта, генеруючи $A \times 4$ виходи на кожній позиції. Параметри цих підмереж є спільними для всіх рівнів піраміди ознак, що підвищує ефективність моделі.

RetinaNet використовує якірні рамки на різних рівнях FPN для забезпечення щільного покриття зображення пропозиціями різних форм та розмірів. На кожному рівні піраміди розміщується щільна сітка якорів з різними базовими розмірами, що дозволяє кожному рівню спеціалізуватися на детекції об'єктів відповідного масштабу. До кожного базового розміру застосовуються кілька співвідношень сторін та підмасштабів, що призводить до генерації *A* якорів у кожній просторовій позиції на кожному рівні, забезпечуючи необхідне щільне покриття для ефективної одностадійної детекції.

Математичне формулювання та інтуїтивне пояснення фокусної відстані базується на стандартній функції крос-ентропії *(CE)* для бінарної класифікації, яка визначається як:

$$CE(p_t) = -\log(p_t) \tag{5}$$

де pt є ймовірністю, спрогнозованою моделлю для істинного класу. Фокусна відстань модифікує кросс-ентропію шляхом введення модулюючого фактора.

Важливу роль відіграють параметр фокусування γ та балансуючий параметр α. Параметр γ контролює ступінь зменшення впливу легко класифікованих прикладів. Зі збільшенням γ > 0, вплив легких прикладів експоненційно зменшується. Експериментально встановлено, що оптимальним значенням для γ є 2. Для подальшого балансування між позитивними та негативними прикладами використовується α-зважена версія фокусної вістані:

$$FL(p_t) = -a_t(1-p_t)^{\gamma}\log(p_t) \tag{6}$$

де $a_t = a$ для позитивного класу та $a_t = (1 - a)$ для негативного класу. Параметр a зазвичай в діапазоні [0,1] дозволяє надати більшу вагу рідкісному класу, яким часто є об'єкти.

Механізм подолання дисбалансу класів у задачах щільної детекції об'єктів полягає у значній перевазі кількості потенційних фонових локацій над кількістю локацій, що містять об'єкти. За використання стандартної крос-ентропії сукупний внесок втрат від численних "легких" фонових прикладів може значно перевищувати внесок від значно меншої кількості складних прикладів об'єктів, що призводить до зміщення навчання моделі у бік класифікації фону. Фокусна функція вирішує цю проблему шляхом значного зменшення вагового внеску легко класифікованих фонових анкерів за допомогою модулюючого фактора, одночасно збільшуючи відносну вагу важких, неправильно класифікованих прикладів у загальній функції втрат. Це спонукає модель приділяти більше уваги навчанню на складних випадках, що сприяє підвищенню точності детекції об'єктів. Концепція фокусної відстані є універсальною та може бути застосована в інших задачах класифікації з істотним дисбалансом класів, включаючи військові застосування, де певні типи цілей можуть бути значно рідкіснішими за фон або інші нецільові об'єкти, що робить використання подібних механізмів зважування втрат важливим для підвищення надійності виявлення критично важливих, але рідкісних цілей.

2.4. Архітектура Visual Transformer (ViT)

Поява Vision Transformer (ViT) сприяла значному концептуальному зсуву в галузі комп'ютерного зору, продемонструвавши ефективність застосування архітектур, що ґрунтуються на механізмі уваги, який раніше домінував в обробці природної мови, до візуальних задач, що стало

альтернативою традиційним згортковим нейронним мережам. Vision Transformer адаптує стандартну архітектуру Transformer для обробки зображень, розглядаючи вхідне зображення не як двовимірну сітку пікселів, а як послідовність дискретних патчів.

На першому етапі відбувається розбиття зображення на патчі, де вхідне зображення поділяється на фіксовану кількість непересічних квадратних фрагментів заданого розміру. Кожен патч згодом піддається лінійному проектуванню, в процесі якого він перетворюється у вектор та лінійно проектується у векторне представлення фіксованої розмірності *N*. Отримана послідовність векторів патчів функціонально аналогічна послідовності токенів у задачах NLP. З огляду на те, що стандартна архітектура Transformer є інваріантною до порядку елементів у послідовності, до векторів патчів додаються позиційні ембедінги, які є навченими векторними представленнями, що кодують інформацію про відносне або абсолютне просторове розташування кожного патча в межах зображення, що є критично важливим для розуміння його просторової структури.

За аналогією з моделлю BERT, на початок послідовності ембедінгів патчів може бути доданий спеціальний токен класифікації, вихідний стан якого після проходження через енкодер Transformer використовується для здійснення фінальної класифікації всього зображення. Підготовлена таким способом послідовність векторів подається на вхід стандартного енкодера Transformer, що складається з множини ідентичних шарів.

Ключовим компонентом кожного шару енкодера Transformer є механізм багатоголовкової самоуваги [Multi-Head Self-Attention], зображеного на Рисунку 2.8 [51]. Механізм самоуваги дозволяє моделі визначати ваговий

внесок кожного патча в послідовності відносно всіх інших патчів при обчисленні нового представлення для кожного з них.

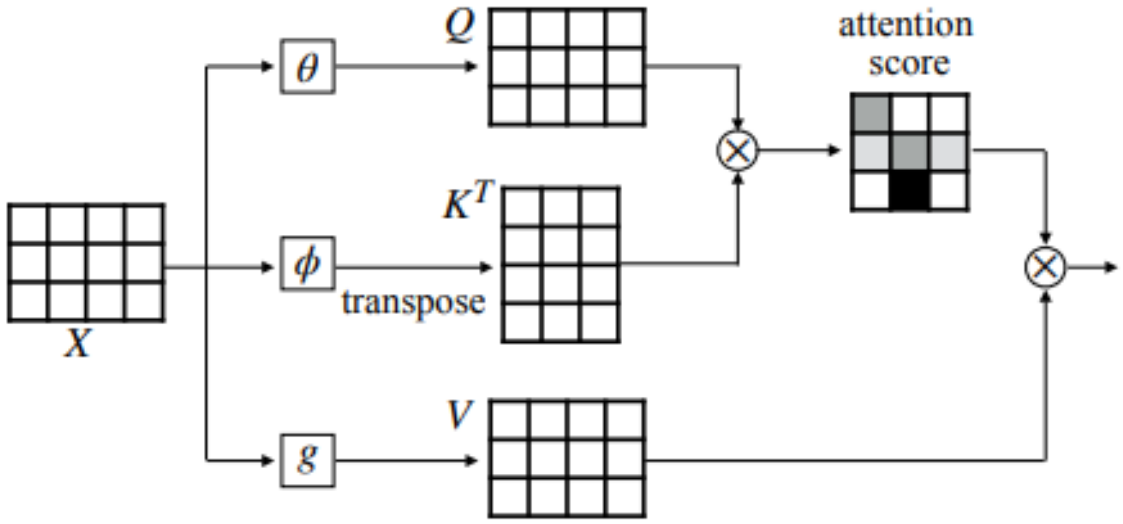

Рисунок 2.8. Принцип роботи Self-Attention механізму [51]

Кожен патч генерує три вектори: запит (Query, Q), ключ (Key, K) та значення (Value, V). Міра "схожості" між вектором запиту одного патча та векторами ключів усіх інших патчів визначає ступінь впливу значень цих інших патчів на оновлене представлення першого патча. На відміну від згорткових шарів у CNN, які мають обмежене рецептивне поле та оперують переважно з локальними ознаками, механізм самоуваги в ViT надає можливість моделювати глобальні залежності між будь-якими двома патчами зображення, незалежно від їхньої просторової віддаленості, що потенційно сприяє кращому розумінню загального контексту сцени. Замість використання одного набору вагових коефіцієнтів для Q, K та V, механізм уваги виконується паралельно кілька разів з різними, незалежно навченими наборами ваг. Це дозволяє моделі одночасно концентруватися на різних аспектах та підпросторах інформації, представленої послідовністю патчів. Виходи з усіх голів потім об'єднуються та лінійно проектуються для отримання фінального виходу

шару уваги. Після шару самоуваги в кожному блоці енкодера зазвичай застосовується повнозв'язний багатошаровий перцептрон, що обробляє кожен патч незалежно, а також використовуються залишкові з'єднання та нормалізація шарів для стабілізації процесу навчання глибоких нейронних мереж.

Через велику кількість варіантів архітектур візуальних трансформерів, кожна з яких має свої переваги залежно від характеру та складності поставленого завдання. Вибір конкретної моделі значною мірою визначається вимогами до точності, обчислювальними ресурсами, доступними для навчання та інференсу, а також особливостями вхідних даних.

Таблиця 2.1. Порівняння Vision Transformer моделей

Модель	**Кількість параметрів**	**FLOPs (ImageNet)**	**Top-1 Accuracy (%)**	**Особливості**
ViT-B/16	86M	17.7B	77.9	Перша базова ViT, патчі 16×16
DeiT-B	86M	17.7B	81.8	Навчена без великих даних, distillation
Swin-T	29M	4.5B	81.2	Ієрархічна структура, зсувні вікна
Swin-B	88M	15.4B	83.5	Більша версія Swin, масштабована структура
PVTv2-B2	25M	4.0B	82.0	Легка модель з хорошою продуктивністю
ConvNeXt-B	89M	15.4B	83.8	Конволюційна альтернатива ViT, inspired by ViT

Focal-T	29M	4.9B	82.2	Фокусування на локальних + глобальних зв’язках
ViT-G/14	1.8B	> 1T	90.5	Дуже велика модель, потребує TPU-підтримку

Продовження Таблиці 2.1.

ВИСНОВКИ ЗА РОЗДІЛОМ

В даному розділі було розглянуто ключові архітектури нейронних мереж, що лежать в основі сучасних систем виявлення об'єктів. Досліджено одно та двоетапні типи архітектур та їх класифікацію. Особливу увагу було приділено принципам функціонування, ключовим інноваціям та архітектурним особливостям кожної розглянутої моделі, включаючи їхні переваги, потенційні обмеження та напрямки подальшого розвитку в контексті завдань комп'ютерного зору. Проаналізовано компроміси між швидкістю та точністю, а також вплив архітектурних рішень на ефективність виявлення об'єктів різних розмірів та у складних сценаріях. Окремо підкреслено значення інноваційних підходів, таких як механізми уваги, у подоланні обмежень традиційних згорткових мереж та відкритті нових можливостей для задач виявлення об’єктів.

РОЗДІЛ ІІІ. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ТА ПОБУДОВА МОДЕЛІ ВИЯВЛЕННЯ РУХОМИХ ОБ’ЄКТІВ

3.1. Формалізація бази знань виявлення рухомих об’єктів

Формалізація бази знань для автоматизованого виявлення рухомих ворожих об'єктів є комплексним процесом, що вимагає глибокого розуміння предметної області, особливостей сенсорних даних та принципів представлення знань. Її метою є створення структурованого, чітко визначеного та операційно придатного сховища експертної інформації, яка може бути ефективно використана системами штучного інтелекту для розпізнавання та відстеження потенційних загроз.

На початковій фазі здійснюється глибокий аналіз та концептуалізація предметної області, що передбачає всебічне дослідження військової доктрини, характеристик озброєння та військової техніки противника, типових сценаріїв бойових дій та особливостей застосування безпілотних літальних апаратів у розвідці. Визначається повний спектр об'єктів інтересу, включаючи танки, бронетранспортери (БТР), бойові машини піхоти (БМП), бронеавтомобілі, артилерію та транспорт, що забезпечує логістику. Для кожного класу об'єктів проводиться попередній аналіз їхніх ключових атрибутів та відмінних рис. Наступним етапом є детальна специфікація характеристик об'єктів, яка передбачає розробку докладних описів для кожного типу ворожого об'єкта, що охоплюють різні модальності сприйняття. Для кожного типу техніки визначаються характерні візуальні дескриптори, такі як силует, який особливо важливий при спостереженні з висоти польоту дрона, особливості корпусу, наявність та конфігурація башти для броньованих машин, а також елементи ходової частини таких як колеса або гусениці.

Аналізуються типові розмірні характеристики об'єктів на різних відстанях, особливості камуфляжного забарвлення та наявність розпізнавальних знаків. Крім того, на основі фізичних принципів тепловиділення та емпіричних даних визначаються характерні розподіли температурних полів для різних елементів військової техніки та особового складу, що є важливим при аналізі даних тепловізійних камер в умовах обмеженої видимості або нічного часу. Важливим аспектом є розширений аналіз контекстуальної інформації, де визначаються та формалізуються різноманітні зовнішні фактори, що можуть впливати на виявлення та інтерпретацію даних. Класифікуються типові види місцевості, що є релевантними для спостереження, зокрема поля, ліс, лісопосадки та польові стежки, з урахуванням їхніх характеристик, які впливають на видимість об'єктів, їхні траєкторії руху та можливість маскування. Враховуються також часові умови, такі як час доби та пора року, а також поточні погодні умови, оскільки ці фактори істотно впливають на якість візуальних та теплових даних, а також на поведінку об'єктів. Вхідні дані, що надходять від сенсорних систем для виявлення рухомих ворожих об'єктів, характеризуються значною варіативністю залежно від дистанції між сенсором та об'єктом спостереження. Ця дистанційна залежність зумовлює якісні та кількісні зміни в представленні об'єкта, що необхідно враховувати при розробці алгоритмів обробки та аналізу даних.

При близькій дистанції об'єкт спостереження займає значну частину поля зору сенсора, що забезпечує високу деталізацію його візуальних характеристик. Стають чітко розрізнюваними дрібні структурні елементи, особливості форми та текстури поверхні, а також специфічні деталі обладнання. Однак, при близькій дистанції може виникати проблема

обмеження поля зору, що ускладнює сприйняття об'єкта в контексті його безпосереднього оточення.

Рисунок 3.1. Особливості різноманітності даних. Близька дистанція, середня дистанція, значна дистанція.

На середній дистанції об'єкт спостереження займає меншу частину поля зору, що призводить до зменшення рівня деталізації його візуальних ознак. Дрібні елементи можуть ставати нерозрізнюваними, а контури об'єкта узагальнюються. Теплові сигнатури відображають загальний температурний профіль об'єкта, проте деталізація розподілу температурних полів знижується. На цій дистанції зростає можливість сприйняття об'єкта в ширшому контексті навколишнього середовища, що може сприяти аналізу його поведінки та взаємодії з оточенням.

При значній дистанції об'єкт спостереження займає мінімальну частину поля зору, що суттєво обмежує обсяг доступної візуальної інформації. Розрізнення окремих структурних елементів стає практично неможливим, а ідентифікація об'єкта ускладнюється. Візуально об'єкт може бути представлений як невелика пляма або точка. Теплові сигнатури, за наявності, можуть бути єдиною помітною ознакою, що вказує на наявність тепловипромінюючого об'єкта, проте його класифікація на основі лише теплових даних стає менш надійною. На великих дистанціях контекстуальна інформація, така як тип місцевості або напрямок руху, набуває особливого значення для попередньої ідентифікації потенційної загрози.

Подальшим кроком є детальне моделювання поведінкових патернів для кожного типу ворожого об'єкта в залежності від контекстуальних умов. Танки характеризуються переважним переміщенням по відкритій місцевості, артилерія зазвичай замаскована в лісосмугах, БМП та БТР використовують попередньо підготовлені стежки, а логістичний транспорт переміщується дорогами. Не менш важливим є комплексна характеристика метаданих сенсорів, де детально описуються технічні характеристики та особливості даних, що надаються оптичними камерами в динаміці такі як FPV дрони, камерами високої роздільності для детальної розвідки та тепловізійними камерами для нічного спостереження, включаючи їхню роздільну здатність, кут огляду, спектральні діапазони та особливості обробки даних.

Рисунок 3.2. Особливості поведінкових патернів різної техніки в умовах ведення бою

Артилерійські системи, як правило, характеризуються стаціонарним розташуванням з метою ведення вогню, при цьому спостерігається тенденція до їхнього маскування в межах лісових масивів або лісосмуг для ускладнення виявлення засобами розвідки, де стаціонарність положення може бути ключовою ознакою при їхньому виявленні протягом тривалих періодів часу, особливо в поєднанні з характерними ознаками маскування. На противагу цьому, танки демонструють високу мобільність та здатність до пересування поза дорожньою мережею, зокрема на відкритих полях та непроїзних ділянках місцевості, що відповідає їхній тактичній ролі у здійсненні швидких маневрів та проривів оборони, де динаміка переміщення на відкритій місцевості може слугувати важливим індикатором їхньої активності та потенційних напрямків атаки. Бойові машини піхоти та бронетранспортери вирізняються універсальністю щодо вибору маршрутів пересування, оскільки можуть ефективно функціонувати як у лісистій місцевості,

використовуючи наявні просіки та міждеревні проходи, так і на підготовлених стежках, забезпечуючи маневреність піхотних підрозділів та їхню вогневу підтримку в різних умовах бойової обстановки.

Врахування цих характерних особливостей є важливим для оптимізації алгоритмів виявлення та прогнозування можливих місць знаходження та напрямків руху ворожої техніки. Враховуючи ці усталені патерни, системи виявлення повинні адаптивно застосовувати різні стратегії пошуку та аналізу ознак залежно від передбачуваного типу об'єкта та ймовірних умов його знаходження.

Рисунок 3.3. Дефекти вхідних даних та нестабільність зв’язку в БПЛА

Процес отримання вхідних даних для систем автоматизованого аналізу зображень, зокрема у складних умовах дистанційного

спостереження, може супроводжуватися низкою артефактів та перешкод, що істотно впливають на якість візуальної інформації. Однією з проблем є наявність дефектів зображення, які можуть виникати внаслідок особливостей роботи сенсора, умов зйомки або процесів передачі даних. До таких дефектів належать шуми різного походження, спотворення кольору, розмиття контурів об’єктів та локальні аномалії піксельних значень.

Іншою суттєвою перешкодою є нестабільність каналів зв’язку, що може проявлятися у вигляді переривань передачі даних або перебоїв у потоці зображень. Ці явища призводять до фрагментарності візуальної інформації, втрати окремих кадрів або частин зображень, що ускладнює безперервне спостереження за рухомими об’єктами та аналіз їхньої динаміки.

Крім того, можливе виникнення ситуацій, коли зображення демонструють ефект “зависання”, що характеризується тимчасовою зупинкою оновлення візуальної інформації. Це явище може бути спричинене як проблемами з передачею даних, так і обмеженнями обчислювальних ресурсів на стороні сенсора або приймаючого пристрою. “Зависання” зображення унеможливлює відстеження руху об’єктів у реальному часі та може призводити до втрати важливої інформації про їхню траєкторію та поведінку. Зазначені особливості вхідних даних, пов’язані з дефектами зображення та нестабільністю каналів зв’язку, становлять значний виклик для розробки надійних систем автоматизованого виявлення та аналізу. Ефективне функціонування таких систем вимагає застосування спеціалізованих методів попередньої обробки зображень для зменшення впливу артефактів, а також розробки алгоритмів, стійких до втрати даних та здатних до інтерполяції

пропущеної інформації для забезпечення безперервного та якісного аналізу візуальної сцени.

На етапі обґрунтування вибору методу формалізації знань здійснюється вибір формального апарату для представлення накопиченої інформації, враховуючи критерії виразності, можливості автоматизованого виведення, зручності оновлення та інтеграції. Може бути обрано онтологічний підхід або фреймові системи, залежно від складності предметної області та вимог до системи виявлення. Наступним кроком є систематична реалізація бази знань, де здійснюється безпосереднє кодування знань у вибраній формальній системі шляхом створення класів, визначення атрибутів, встановлення зв’язків та формулювання правил. З метою забезпечення надійності розроблена база знань піддається валідації та верифікації, яка оцінює повноту, коректність та несуперечливість представлених знань, а також її ефективність на реальних та синтетичних даних.

На завершальному етапі відбувається інтеграція та еволюція бази знань з іншими компонентами системи виявлення, такими як модулі обробки сенсорних даних та прийняття рішень, з передбаченням механізмів постійного оновлення та розширення знань у відповідь на зміни в тактичній обстановці та появу нових типів ворожих об’єктів. Отримана інформація використовується для оперативного надання відомостей та формування ситуаційної картини шляхом передачі в режимі реального часу верифікованих даних про виявлені об’єкти, їхню типологічну належність, просторову локалізацію та динамічні параметри руху, з подальшою візуалізацією на геоінформаційних платформах для формування цілісного розуміння оперативно-тактичної обстановки.

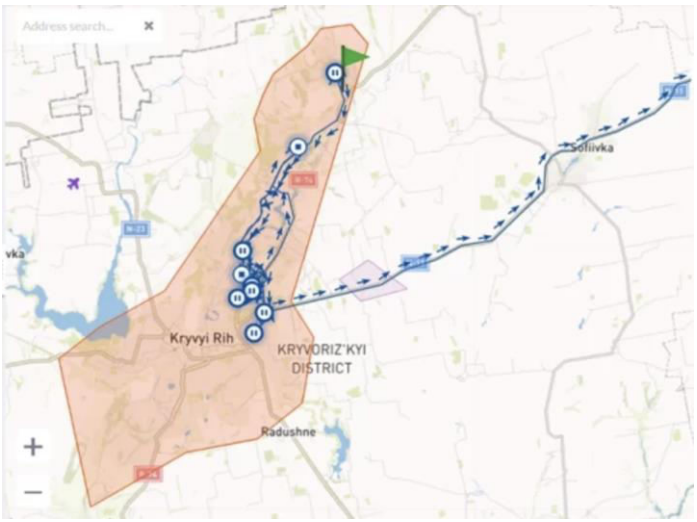

Рисунок 3.4. Симуляційний приклад трекингової системи рухомих об’єктів

Крім того, ідентифіковані об’єкти та зафіксовані особливості їхньої поведінки слугують інформаційною основою для підтримки процесів прийняття рішень у часовому континуумі, що може включати ініціювання відповідних заходів реагування, таких як оповіщення зацікавлених підрозділів, наведення вогневих засобів та коригування траєкторії польоту безпілотних літальних апаратів. Акумульовані дані підлягають аналітичній обробці та прогностичному моделюванню з метою виявлення стійких тенденцій, кореляційних залежностей у моделях переміщення ворожої техніки та розробки ймовірнісних прогнозів щодо їхніх майбутніх дій, а також ідентифікації зон підвищеної концентрації ризиків.

Дані, отримані від системи автоматизованого виявлення, використовуються як верифікаційний інструмент для підтвердження або спростування відомостей з інших розвідувальних джерел, підвищуючи загальну достовірність та інтегрованість інформаційного поля. Система також здійснює автоматизовану реєстрацію подій та документування всіх епізодів виявлення об’єктів, включаючи часові мітки, географічні

координати, класифікаційні атрибути та супутні метадані, забезпечуючи формування деталізованого хронологічного журналу. Нарешті, записи реальних сценаріїв виявлення, задокументовані системою, можуть слугувати цінним дидактичним матеріалом для підготовки та підвищення кваліфікації операторів та аналітичного персоналу.

3.2. Вибір засобів для реалізації моделі

Етап практичної реалізації розробленої концепції передбачає обґрунтований вибір інструментальних засобів, що забезпечать ефективну розробку, навчання та подальше функціонування моделі. Серед широкого спектру доступних технологій, мова програмування Python вирізняється своєю універсальністю, розвиненою екосистемою бібліотек та широкою підтримкою спільноти, що робить її оптимальним вибором для даного проекту. Python, як інтерпретована мова програмування високого рівня, характеризується лаконічним синтаксисом, що сприяє швидкій розробці та читабельності коду. Її динамічна типізація та автоматичне керування пам'яттю спрощують процес програмування, дозволяючи дослідникам зосередитися на алгоритмічній складності задачі, а не на низькорівневих деталях.

Особливу цінність для реалізації моделі виявлення рухомих об'єктів становить розгалужена екосистема Python-бібліотек, спеціалізованих для наукових обчислень, машинного навчання та комп'ютерного зору. Бібліотеки NumPy та SciPy забезпечують потужні інструменти для виконання числових операцій, лінійної алгебри та статистичного аналізу, що є фундаментальними для обробки сенсорних даних та реалізації математичних моделей. Для побудови та навчання нейронних мереж, які є ключовим елементом сучасних систем детекції об'єктів, бібліотеки

TensorFlow та PyTorch пропонують гнучкі та ефективні фреймворки з підтримкою графічних процесорів для прискорення обчислень.

Також бібліотеки OpenCV та Pillow надають широкий спектр функцій для завантаження, попередньої обробки, аналізу та візуалізації зображень, що є необхідним для роботи з даними, отриманими з оптичних та тепловізійних сенсорів. Для реалізації формалізованої бази знань можуть бути використані бібліотеки, що підтримують семантичні мережі та онтології, забезпечуючи структуроване представлення експертної інформації та можливість логічного виведення. Крім того, Python інтегрується з численними інструментами для розгортання та масштабування розроблених моделей, що є важливим для їхнього практичного застосування на борту безпілотних літальних апаратів або у складі наземних станцій управління.

Визначальною перевагою мови програмування Python у контексті розробки систем виявлення рухомих ворожих об'єктів є її безперешкодна інтеграція з широким спектром зовнішніх сервісів та готових програмних рішень. Екосистема Python забезпечує ефективну взаємодію з хмарними платформами обробки даних та машинного навчання, що спрощує масштабування обчислювальних ресурсів та розгортання розроблених моделей. Наявність стандартизованих інтерфейсів та бібліотек полегшує інтеграцію з геоінформаційними системами для візуалізації та аналізу просторових даних про виявлені об'єкти. Крім того, Python підтримує взаємодію з різноманітними API для доступу до попередньо навчених моделей, сервісів аналізу відеопотоку та інших готових компонентів, що скорочує час розробки та підвищує функціональність кінцевого рішення. Інтеграційні можливості Python сприяють створенню гнучкої та

ефективної системи виявлення шляхом використання переваг існуючих технологічних напрацювань.

Keras, як високорівневий API для нейронних мереж, інтегрований у потужні фреймворки глибокого навчання, такі як TensorFlow, PyTorch та JAX, є дієвим інструментом для оперативного проектування та експериментування зі складними моделями машинного навчання. Його функціональна основа полягає у принципах модульності та абстракції, що дозволяє дослідникам зосереджуватися на архітектурних аспектах нейронних мереж, мінімізуючи необхідність заглиблення в низькорівневі деталі обчислювальних процесів. В основі побудови моделі лежить концепція послідовного або функціонального компонування шарів, кожен з яких являє собою автономний модуль, що здійснює певну трансформацію вхідних тензорних даних. Keras надає розгалужений набір попередньо імплементованих шарів, що охоплюють широкий спектр нейронних мережних структур. Конфігурація процесу навчання здійснюється шляхом визначення оптимізатора, функції втрат та метрик оцінки продуктивності. Навчання моделі ініціюється, що автоматизує етапи прямого та зворотного поширення сигналу, обчислення градієнтів та оновлення вагових коефіцієнтів.

Переваги використання Keras є значними. Простота використання та висока швидкість розробки забезпечуються інтуїтивно зрозумілим API та мінімальною кількістю необхідного коду для побудови складних моделей. Модульність та гнучкість архітектури, що базується на незалежних шарах, сприяють створенню різноманітних топологій мереж для вирішення широкого кола завдань. Будучи інтегрованою частиною провідних фреймворків глибокого навчання, Keras отримує доступ до їхньої потужності та оптимізацій, а широка підтримка спільноти та велика

кількість доступної документації полегшують процес освоєння та застосування. Крос-платформність Keras та можливості розгортання моделей на різних апаратних платформах є важливими перевагами для практичного застосування розроблених рішень. Орієнтованість на користувача робить Keras зручним інструментом для виконання типових завдань машинного навчання. У контексті задачі виявлення рухомих ворожих об'єктів, Keras надає ефективні засоби для побудови згорткових нейронних мереж, реалізації механізмів уваги, визначення специфічних функцій втрат та використання оптимізаторів для навчання моделей, придатних для роботи в умовах реального часу та обмежених ресурсів.

TensorFlow, розроблений Google, являє собою потужну open-source платформу для машинного навчання, що забезпечує комплексний набір інструментів для побудови, навчання та розгортання різноманітних моделей, включаючи глибокі нейронні мережі. В основі TensorFlow лежить концепція потоків даних, де обчислення представляються у вигляді орієнтованого графа. Вузли графа відображають математичні операції, а ребра – багатовимірні масиви даних, що називаються тензорами. Ця парадигма дозволяє ефективно паралелізувати обчислення на різних апаратних прискорювачах, таких як центральні процесори, графічні процесори та спеціалізовані тензорні процесорні блоки, що є критично важливим для обробки великих обсягів даних та навчання складних моделей.

TensorFlow надає низькорівневий API, що забезпечує гнучкий контроль над кожним аспектом розробки моделі, від визначення архітектури шарів до оптимізації процесу навчання. Його функціональність включає широкий спектр математичних операцій, лінійну алгебру, функції активації, оптимізатори, функції втрат та

інструменти для роботи з даними. Для спрощення розробки та прискорення прототипування, TensorFlow інтегрує високорівневий API Keras, який пропонує більш інтуїтивно зрозумілий інтерфейс для побудови нейронних мереж.

TensorFlow забезпечує необхідну інфраструктуру для ефективної обробки сенсорних даних, отриманих з безпілотних літальних апаратів. Його можливості паралельних обчислень дозволяють прискорити навчання згорткових нейронних мереж, які є ключовими для аналізу зображень. TensorFlow також надає інструменти для роботи з різними форматами даних, включаючи зображення та відеопотоки, а його підтримка розподіленого навчання дозволяє використовувати кластери обчислювальних ресурсів для навчання особливо складних моделей на великих наборах даних. Крім того, TensorFlow пропонує розвинені засоби для розгортання навчених моделей на різних платформах, від серверних систем до вбудованих пристроїв з обмеженими обчислювальними ресурсами, що є важливим для застосування моделей безпосередньо на борту БПЛА. TensorFlow є потужною та універсальною платформою, що забезпечує надійну основу для розробки та впровадження ефективних систем виявлення рухомих ворожих об'єктів.

3.3. Вибір інструментів розмітки даних

Label Studio являє собою open-source платформу для спільної анотації даних, що набула значної популярності в галузі машинного навчання та комп'ютерного зору. Її архітектура розроблена з метою забезпечення ефективного та масштабованого процесу розмітки різноманітних типів даних, включаючи зображення, текст, аудіо та відео, що робить її цінним інструментом на етапі підготовки навчальних вибірок для моделей штучного інтелекту. Основний принцип роботи Label Studio полягає у

централізованому управлінні проектами анотації, де адміністратори можуть визначати набори даних, завдання з розмітки та схеми анотації, адаптовані до специфіки конкретної задачі. Платформа надає інтуїтивно зрозумілий веб-інтерфейс для анотаторів, що включає набір інструментів для виконання різних типів розмітки, таких як окреслення обмежувальних рамок, полігональна сегментація, розмітка ключових точок, класифікація зображень та тексту, транскрибування аудіо тощо.

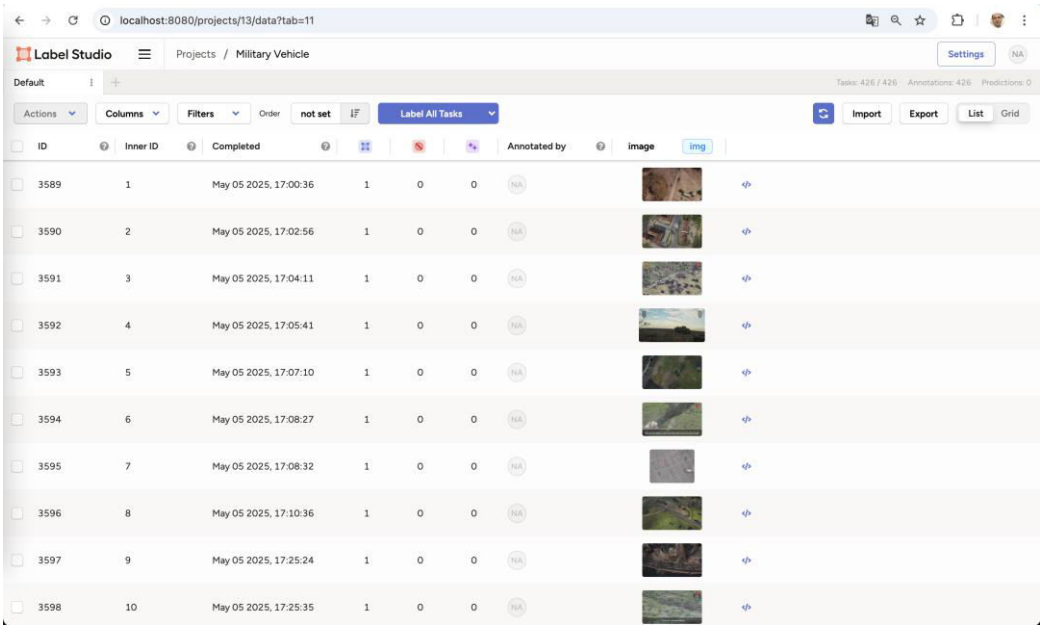

Рисунок 3.5. Інтерфейс інструменти розмітки Label Studio

Однією з ключових переваг Label Studio є її гнучкість та розширюваність. Платформа підтримує налаштування інтерфейсу розмітки за допомогою мови розмітки завдань, що дозволяє адаптувати інструменти та процес анотації до унікальних вимог кожного проекту. Крім того, Label Studio надає API для інтеграції з існуючими системами зберігання даних, моделями машинного навчання та іншими інструментами обробки даних, що сприяє автоматизації окремих етапів процесу анотації та інтеграції розмічених даних у навчальні цикли моделей.

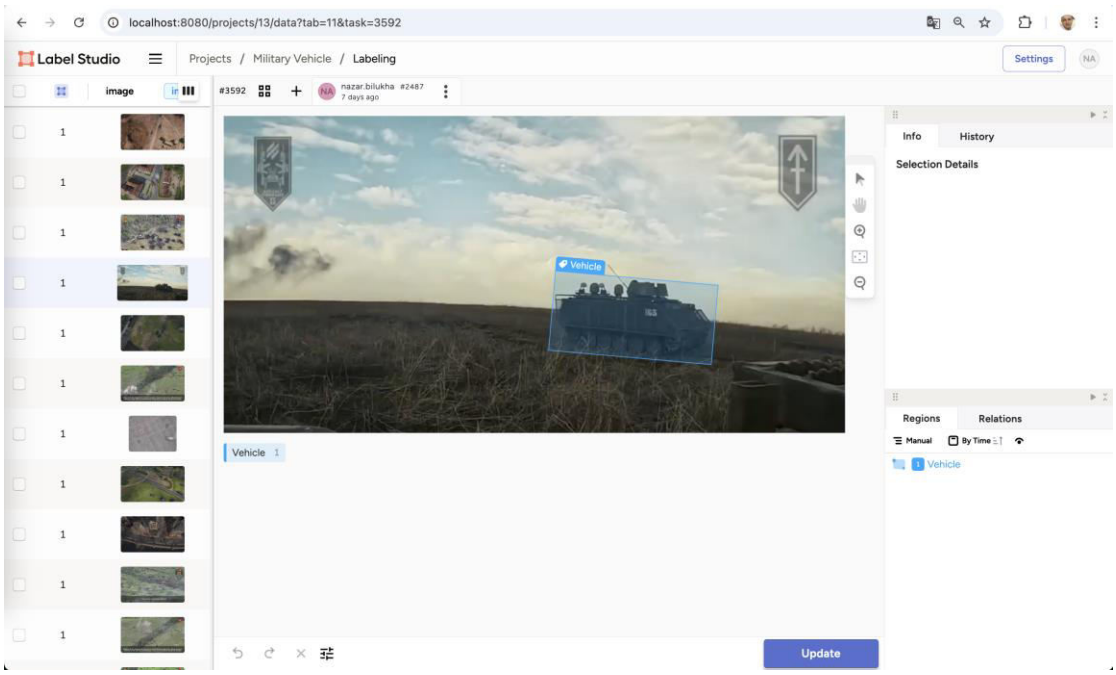

Рисунок 3.6. Приклад розмітки об’єктів в Label Studio

Для задачі виявлення рухомих ворожих об'єктів з використанням даних безпілотних літальних апаратів, Label Studio може бути використана для створення якісно розміченого навчального набору даних. Оператори-анотатори можуть використовувати інструменти платформи для окреслення обмежувальних рамок навколо різних типів ворожої техніки (танки, БТР, БМП, артилерія, транспорт), а також для класифікації цих об'єктів. Для задач сегментації може бути використана полігональна розмітка для точного визначення меж об'єктів, що є важливим для навчання моделей, здатних до піксельної класифікації.

Платформа також підтримує роботу з послідовностями зображень, що є релевантним для аналізу динаміки руху об'єктів. Анотатори можуть розмічати об'єкти на окремих кадрах відеопотоку, а також відстежувати їхні траєкторії протягом часу. Крім того, Label Studio надає можливості для контролю якості анотації, включаючи інструменти для перегляду та

порівняння розміток різних анотаторів, а також механізми консенсусної розмітки для підвищення об'єктивності результатів.

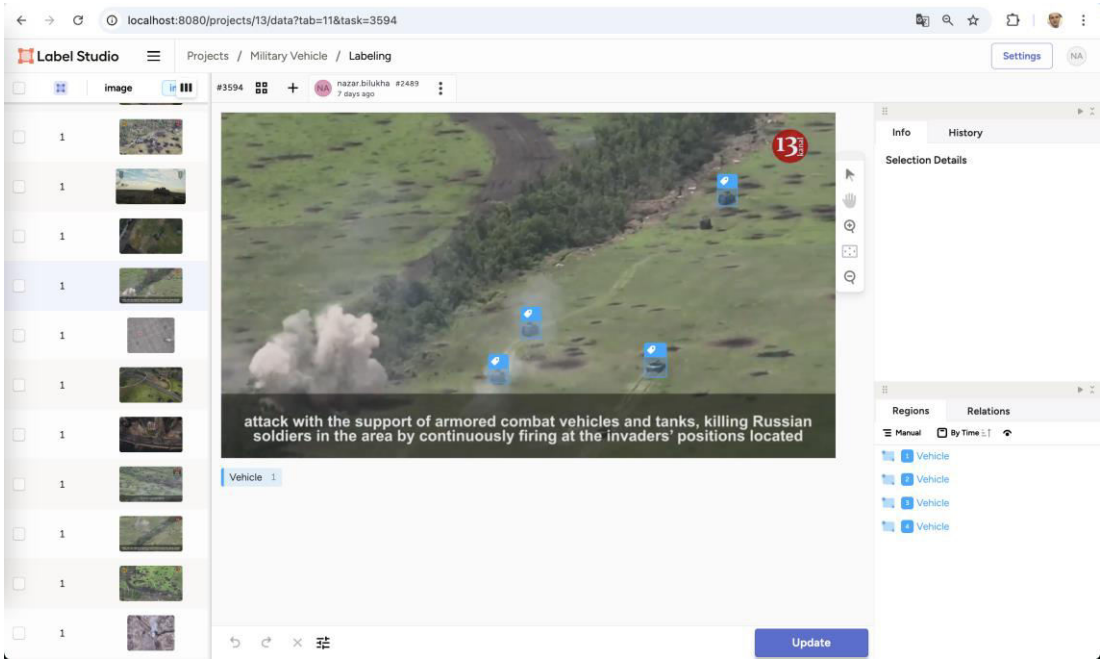

Рисунок 3.7. Приклад розмітки декількох об’єктів на фото

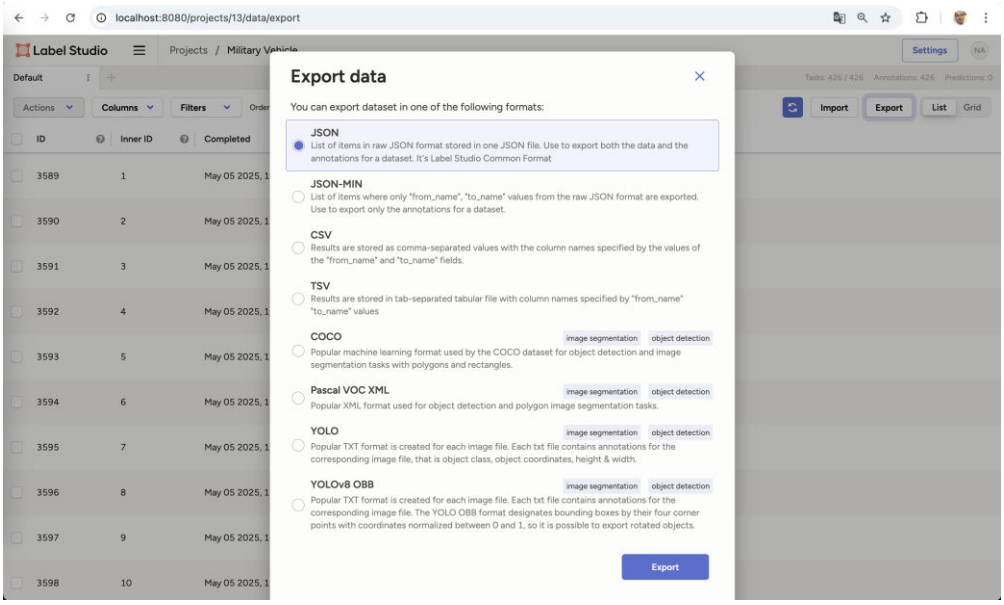

Рисунок 3.8. Список підтримуваних форматів експорту

3.4. Реалізація моделі

Jupyter Notebook являє собою інтерактивне обчислювальне середовище, що набуло широкого застосування в наукових дослідженнях, аналізі даних, машинному навчанні та освіті. Його фундаментальною особливістю є поєднання виконуваного коду, розміченого тексту, математичних формул, візуалізацій та мультимедійного контенту в єдиному документі, відомому як "блокнот".

Принцип роботи Jupyter Notebook базується на клієнт-серверній архітектурі. Користувач взаємодіє з веб-інтерфейсом, який відображає блокнот. За лаштунками працює ядро – окремий процес, відповідальний за виконання коду на певній мові програмування. Коли користувач запускає комірку з кодом, запит надсилається до ядра, яке виконує код та повертає результати назад до клієнта для відображення у блокноті. Однією з ключових переваг Jupyter Notebook є його інтерактивність. Користувачі можуть експериментувати з кодом у невеликих, незалежних комірках, миттєво бачити результати та ітеративно розширювати свій аналіз або розробку. Такий підхід сприяє дослідницькому програмуванню та полегшує розуміння складних обчислювальних процесів.

В результаті було здійснено навчання чотирьох репрезентативних моделей: одностадійного детектора YOLOv8, гіпотетичної версії YOLOv11, одностадійного детектора RetinaNet та двоетапного детектора Faster R-CNN. Процес навчання кожної з зазначених моделей був ініційований з використанням конфігураційних параметрів, що відображали застосування обчислювальних потужностей графічного процесора для прискорення тензорних операцій, фіксовану кількість ітерацій навчання, визначену у 300 епох, та уніфікований розмір вхідних зображень, встановлений на рівні 640x640 пікселів, з метою забезпечення

порівнянності умов експерименту. По завершенні багаторазових ітерацій оптимізації вагових коефіцієнтів для кожної архітектури було сформовано структуровані результати, що кількісно та якісно характеризують їхню здатність до виявлення та класифікації об'єктів на незалежному валідаційному наборі даних, не залученому до процесу навчання. Ці результати слугуватимуть основою для подальшого порівняльного аналізу та висновків щодо придатності кожної архітектури для вирішення поставленої задачі.

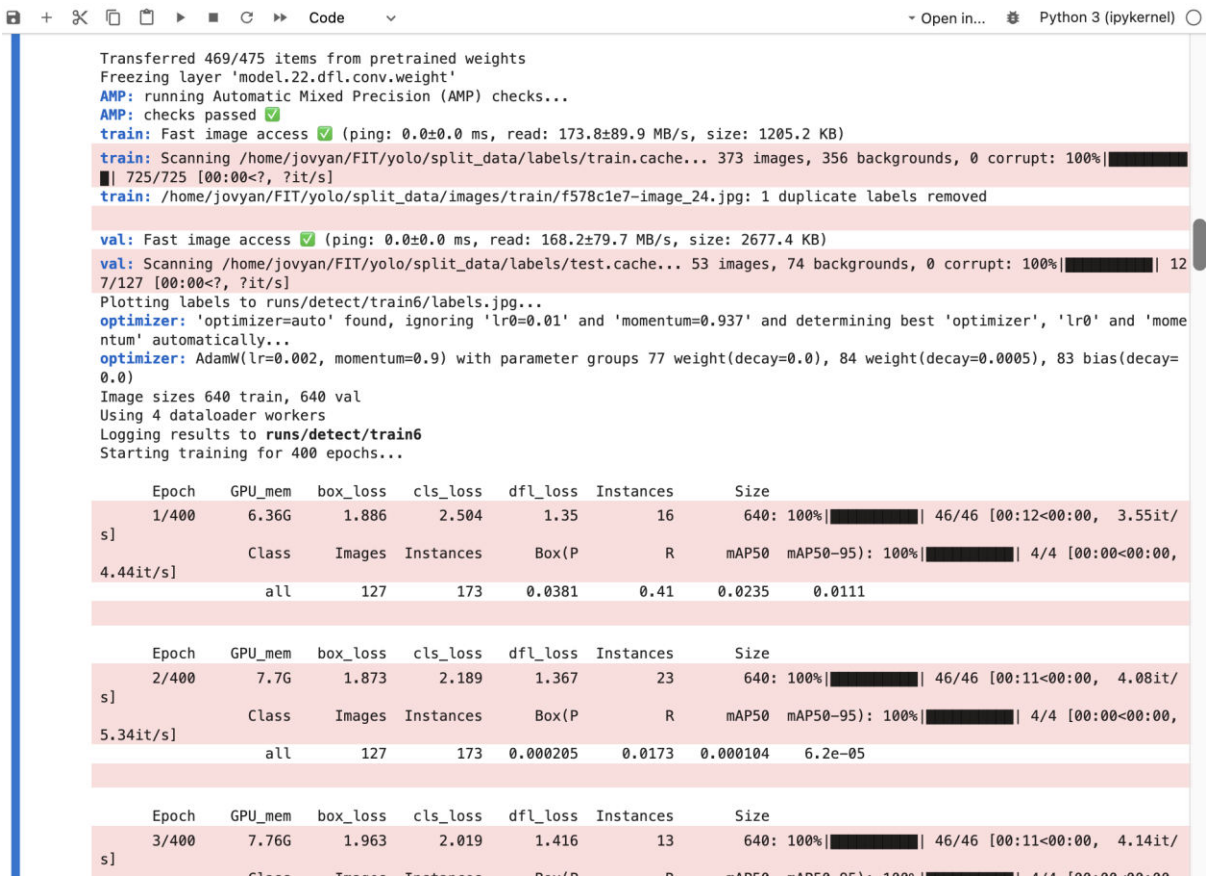

Рисунок 3.9. Процес тренування моделі

На основі представлених результатів тренування, що відображають динаміку зміни функції втрат та метрики точності протягом трьохсот епох навчання, здійснено порівняльний аналіз продуктивності чотирьох архітектур.

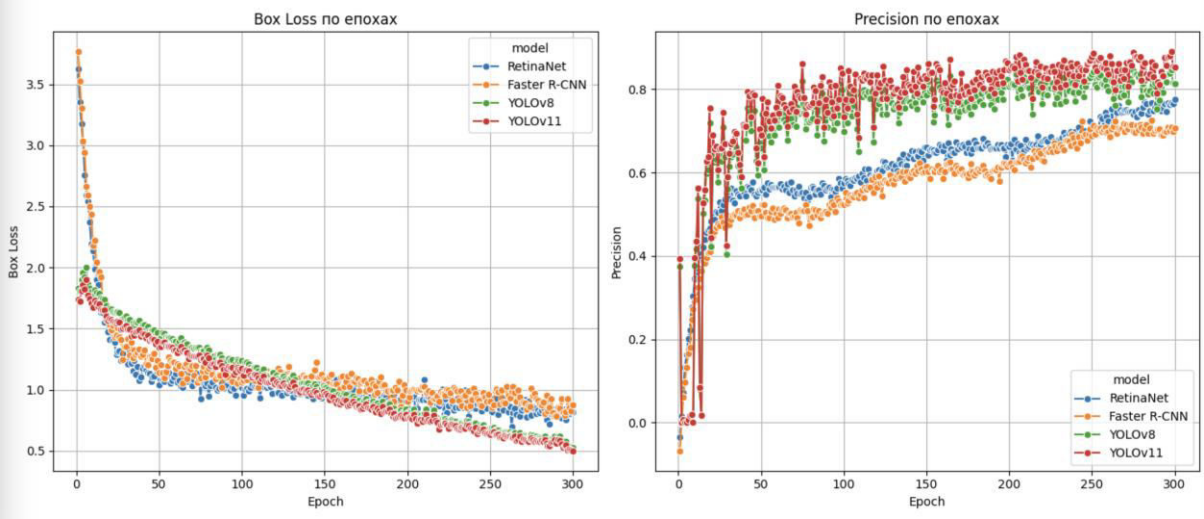

Рисунок 3.10. Результати тренування моделей

Аналіз динаміки функції втрат засвідчив загальну тенденцію до її зниження для всіх досліджуваних моделей, що є індикатором їхньої здатності до навчання. Водночас, спостерігаються відмінності у швидкості збіжності та досягнутих кінцевих значеннях. У початкових епохах моделі YOLOv8 та YOLOv11 демонстрували вищу швидкість збіжності порівняно з RetinaNet та Faster R-CNN, що може свідчити про їхню ефективність у первинному засвоєнні закономірностей даних. Після ста епох темп зниження втрат сповільнився для всіх моделей, проте на завершальних етапах навчання YOLOv8 та YOLOv11 досягли дещо нижчих значень функції втрат, вказуючи на потенційну перевагу в точності локалізації об'єктів.

Аналіз динаміки метрики точності виявив загальне стрімке зростання на початкових етапах навчання, що відображає покращення здатності моделей до класифікації виявлених об'єктів. Подальша стабілізація кривих точності продемонструвала досягнення вищих значень моделями YOLOv8 та YOLOv11 порівняно з RetinaNet та Faster R-CNN, що свідчить про їхню потенційну здатність до зменшення кількості хибнопозитивних

спрацювань та більш точної детекції. Крім того, криві точності для YOLOv8 та YOLOv11 характеризувалися меншою волатильністю та вищою стабільністю, особливо на пізніх етапах навчання, що може вказувати на їхню стійкість до перенавчання та кращу узагальнюючу здатність.

Узагальнюючи результати порівняльного аналізу динаміки функції втрат та метрики точності, можна констатувати, що моделі YOLOv8 та YOLOv11 продемонстрували загалом вищу продуктивність у вирішенні поставленої задачі виявлення рухомих ворожих об'єктів. Досягнення ними нижчих значень функції втрат та вищих значень precision вказує на їхню потенційну перевагу як у точності локалізації, так і в точності класифікації об'єктів. Моделі RetinaNet та Faster R-CNN, хоча й продемонстрували здатність до навчання, показали дещо гірші результати. Стабільність метрики Precision для YOLOv8 та YOLOv11 також свідчить про їхню потенційну стійкість до перенавчання та кращу здатність до узагальнення. Майбутні дослідження можуть бути спрямовані на оптимізацію гіперпараметрів моделей, порівняння їхньої ефективності на різних наборах даних та розробку гібридних архітектур для досягнення оптимальних результатів у задачі виявлення рухомих ворожих об'єктів.

ВИСНОВКИ ЗА РОЗДІЛОМ

В даному розділі було розглянуто процес формалізації бази знань, необхідного для автоматизованого виявлення рухомих ворожих об'єктів. Описано ключові етапи цього процесу, починаючи з глибокого аналізу предметної області та закінчуючи вибором методів представлення знань. Детально охарактеризовано специфіку даних, поведінкові залежності різних типів ворожої техніки. Окрему увагу було приділено

обґрунтуванню вибору інструментальних засобів для розмітки даних та розробки моделей машинного навчання. Представлено результати навчання та порівняльного аналізу чотирьох архітектур нейронних мереж:YOLOv8, YOLOv11, RetinaNet, Faster R-CNN у задачі виявлення рухомих ворожих об'єктів, що відображають динаміку функції втрат та метрики точності.

РОЗДІЛ IV. ЗАСТОСУВАННЯ ТА ПОДАЛЬШЕ ВИКОРИСТАННЯ МОДЕЛІ ВИВЛЕННЯ РУХОМИХ ОБ’ЄКТІВ

4.1. Принцип роботи цілісної системи

В результаті передбачається 2 архітектури, які можуть бути використані в подальшому для здійснення задач виявлення. Перша архітектура передбачає збір відеоданих дроном та передача окремих кадрів або стислих відеофрагментів на сервер, де запущено Flask-додаток з інтегрованою моделлю.

Дрон – Клієнт:

- Оснащений однією з камер, оптична, тепловізійна або комбінована.
- Здійснює відеозйомку заданої території.
- Реалізує механізм для надсилання зображень або відеофрагментів на сервер. Це може бути періодичне надсилання окремих ключових кадрів або стислих відеопотоків.
- Може мати базові алгоритми попередньої обробки для оптимізації передачі даних.

Сервер Flask – Backend:

- Запускає веб-сервер на основі фреймворка Flask.
- Має API-ендпоінт для приймання POST-запитів з зображеннями або відеофрагментами від дрона.
- Містить завантажену та готову до використання модель YOLOv11, яку оцінили як найкращу для виявлення рухомих об'єктів.
- Реалізує логіку обробки вхідних даних моделлю.
- Формує відповідь у структурованому форматі, що містить інформацію про виявлені об'єкти.
- Повинен мати додаткові модулі для:
    - Логування запитів та результатів.
    - Зберігання історії виявлень.
    - Візуалізації результатів, накладання обмежувальних рамок на вихідне зображення.
    - Інтеграції з іншими системами.

Принцип роботи системи виявлення, полягає в наступному. На першому етапі дрон здійснює безперервну або періодичну відеозйомку цільової території. Далі, за визначеним тригером, дрон ініціює передачу отриманого зображення або стислого відеофрагменту на спеціальний API-ендпоінт сервера. Сервер Flask, отримавши запит, обробляє його за допомогою відповідного маршруту, який витягує передане зображення або відеофрагмент з тіла запиту. Потім ці дані передаються на вхід моделі виявлення рухомих об'єктів, яка попередньо завантажена та готова до використання на сервері. Модель аналізує отримані візуальні дані та ідентифікує на них рухомі об'єкти, визначаючи їхній та обчислюючи координати обмежувальних рамок навколо кожного виявленого об'єкта, а

також ступінь впевненості у правильності виявлення. Результати обробки, отримані від моделі, формуються сервером у стандартизований формат JSON. Цей JSON-об'єкт містить масив виявлень, де для кожного об'єкта вказано його клас, координати обмежувальної рамки та ступінь впевненості. Після формування JSON-відповідь надсилається сервером Flask назад дрону. Отримавши ці структуровані дані, дрон може використовувати їх для виконання різноманітних завдань, таких як автономна навігація, цілеспрямоване спостереження за виявленими об'єктами, передача інформації оператору в реальному часі тощо.

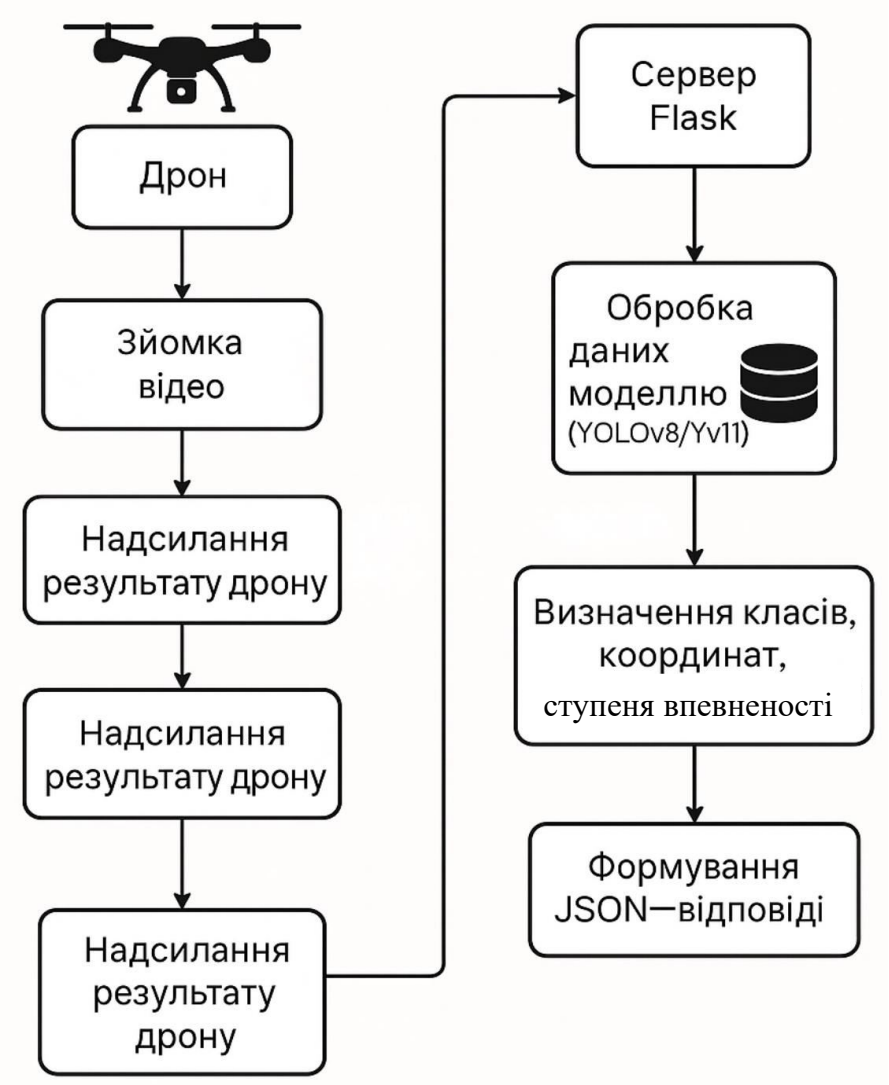

Рисунок 4.1. Блок-схема системи

Друга архітектура передбачає, що модель виявлення рухомих об'єктів інтегрується безпосередньо в обчислювальний модуль дрона.

Дрон – Автономна система:

- Оснащений однією з камер, оптична, тепловізійна або комбінована.
- Має потужний вбудований обчислювальний модуль такий як NVIDIA Jetson, Intel Movidius, який здатний виконувати операції машинного навчання в режимі реального часу.
- Містить завантажену та оптимізовану для вбудованих систем модель виявлення рухомих об'єктів.
- Реалізує алгоритми обробки відеопотоку, детекції об'єктів та їх відстеження.
- Повинен мати модулі для:
    - Попередньої обробки зображень.
    - Фільтрації результатів детекції.
    - Відстеження виявлених об'єктів у часі.
    - Прийняття автономних рішень на основі виявлених об'єктів.
    - Передачі оператору метаданих про виявлені об'єкти.

Принцип роботи вбудованої системи виявлення рухомих об'єктів на дроні полягає в наступній послідовності дій. Насамперед, дрон веде безперервну відеозйомку навколишнього середовища. Отриманий відеопотік у режимі реального часу надходить до вбудованого обчислювального модуля. На даному модулі запущена модель машинного навчання, навчена виявляти рухомі об'єкти. Ця модель аналізує кожен окремий кадр відео або ключові кадри, щоб ідентифікувати потенційні цілі. У разі виявлення рухомого об'єкта, модель визначає його клас та окреслює його на зображенні за допомогою обмежувальної рамки, а

також надає оцінку впевненості у правильності свого виявлення. Далі, вбудоване програмне забезпечення дрона використовує цю інформацію про виявлені об'єкти. Якщо дрон обладнаний екраном для оператора, на ньому відображаються виділені об'єкти. Система також може автоматично відстежувати рух цих об'єктів у відеопотоці.

4.2. Особливості архітектур та розгортання

Вбудована обробка на дроні вирізняється насамперед автономністю, оскільки не потребує постійного зв'язку із зовнішнім сервером чи інтернет-з'єднання. Це забезпечує низьку затримку обробки, що є критично важливим для застосувань, де потрібне миттєве прийняття рішень у реальному часі. Крім того, локальна обробка даних підвищує безпеку, оскільки чутлива інформація не передається зовнішніми каналами. Однак, цей підхід стикається з обмеженнями обчислювальних ресурсів вбудованих систем, що може обмежувати складність використовуваних моделей машинного навчання та швидкість їхньої роботи. Для ефективної роботи потрібна спеціальна оптимізація моделей, що може бути складним завданням. Також, оновлення моделі на великій кількості дронів є більш складним процесом, а необхідність виконання інтенсивних обчислень на борту призводить до вищого енергоспоживання дрона.

На противагу цьому, обробка даних на віддаленому сервері використовує потужні обчислювальні ресурси, що дозволяє застосовувати складні та ресурсомісткі моделі з вищою точністю. Оновлення та керування моделлю є простішим завдяки централізованій інфраструктурі. Крім того, дрон може бути легшим та споживати менше енергії, оскільки основне навантаження з обробки перекладається на сервер. Основними недоліками серверного підходу є залежність від

стабільного інтернет-з'єднання, затримки в якому можуть негативно впливати на час обробки та прийняття рішень. Існує також ризик втрати даних у разі нестабільного з'єднання. Крім того, виникає потреба у розгортанні та підтримці серверної інфраструктури, що може бути пов'язано з додатковими витратами.

Для розгортання веб-серверу необхідно створити Dockerfile, який визначає інструкції для побудови Docker-образу. Команда для створення докер файлу:

```
FROM python:3.9-slim-buster
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY model.h5 .
COPY app.py .
EXPOSE 5000
CMD ["python", "app.py"]
```

Основними командами для побудови Docker-образу є Команда `docker build -t object_detection .` створює Docker-образ на основі Dockerfile. Образ містить усе необхідне для запуску застосунку. Запуск Docker-контейнера здійснюється за допомоги команди `docker run -p 5000:5000 object_detection` яка запускає контейнер з образом. Опція `-p` відображає порт контейнера на порт хост-системи, роблячи сервіс доступним.

4.3. Застосування розробки

Моделювання реального сценарію використання системи передбачає встановлення камери на борту БПЛА та інтегрована передача даних в систему обробки на наземній станції управління. На бортовому обчислювальному модулі БПЛА, такому як NVIDIA Jetson або аналогічна вбудована система, розгортається спеціально оптимізована модель

YOLOv11. Ця модель, розроблена з урахуванням обмежених енергетичних ресурсів та необхідності швидкої обробки даних, попередньо навчається для ідентифікації військової техніки противника, включаючи танки, бронетранспортери, бойові машини піхоти, артилерійські установки, особовий склад, а також інших важливих військових об'єктів, таких як мінометні позиції та командні пункти.

Під час виконання розвідувального польоту, відеопотік, що надходить з встановленої на БПЛА камери, безперервно аналізується цією бортовою моделлю. Процес аналізу відбувається в реальному часі, де модель виявляє на зображеннях рухомі об'єкти, здійснює їхню класифікацію за заданими категоріями та визначає їхні географічні координати. Останнє стає можливим завдяки комбінуванню даних глобальної навігаційної супутникової системи GPS/GNSS БПЛА з координатами виявленого об'єкта на отриманому зображенні.

Рисунок 4.2. Приклад роботи системи. Виявлення ворожого транспорту

Система налаштовується таким чином, щоб пріоритетно виявляти об'єкти, що класифікуються як ворожа військова техніка, які демонструють характерні для військової діяльності патерни поведінки, наприклад, переміщення поза дорогами загального користування або розгортання в бойові порядки. Водночас, система здійснює фільтрацію нерелевантних об'єктів, таких як птахи або цивільні транспортні засоби, якщо алгоритми розпізнавання дозволяють їх диференціювати.

Рисунок 4.3. Приклад множинного виявлення об’єктів

У випадку виявлення об'єкта, що відповідає встановленим критеріям військової загрози, БПЛА автоматично ініціює процес передачі сповіщення на наземну станцію управління. Цей процес сповіщення відбувається в режимі реального часу, забезпечуючи оперативне інформування оператора про потенційну загрозу.

ВИСНОВОК

Дипломна робота присвячена розробці інтелектуальної системи ідентифікації рухомих об'єктів. Основною задачою системи є точне виявлення та відстеження об'єктів у режимі реального часу, зокрема в умовах ведення бойових дій. Ключовими аспектами є розробка алгоритмів виявлення об'єктів для різних типів, розмірів та погодних умов, створення ефективних механізмів відстеження для збереження ідентичності об'єктів у різних кадрах, оптимізація системи для роботи в режимі реального часу. Завдяки забезпеченню адаптивності до мінливих умов, система здатна розпізнавати зображення різних форматів. Технологія здатна виявляти попередні загрози зокрема з сфери оборони, робототехніки, транспорті та сільському господарстві. Система була протестована на реальних відеоданих з безпілотників, які виконували завдання виявлення та ураження ворожої техніки та розвідка.

Досягнуто підвищення точності виявлення за рахунок оптимізації архітектури мережі та стратегій їх навчання, а також збільшення швидкодії обробки відеоданих з БПЛА, що є критично важливим для застосування в реальному часі. Результати дослідження показали, що експериментальна модель YOLOv11 виявилась найкращою для виявлення об’єктів. Значення метрики precision для даної моделі складає 0.87687, що перевищує її конкурентів YOLOv8 зі скором 0.833, модель RetinaNet зі значенням точності 0.7898 та Faster RCNN зі скором 0.7765. Це дозволяє інтегрувати модель з основним застосунком та створює потенційні можливості для повноцінної інтеграції з передовими сучасними технологіями. Система здатна поєднувати модель з ефективними трекерами для обробки відео в реальному часі. Вона враховує адаптацію до мінливих зовнішніх умов та інтеграцію з додатковими датчиками для підвищення точності та надійності.

Подальші кроки для покращення системи є створення мультимоделей для адаптації під кожен з типів зображення у випадку ситуацій, коли модель не передбачає особливі випадки. Універсальний підхід може бути неефективним для покриття усіх типів задач, що постають перед системою. Зокрема, поточний стан та валідація технології показують що вона спроможна покривати базові потреби та слугувати рекомендаційним сервісом для прийняття рішень як операторам дронів так і командирам. В подальшому система може бути навчена конкретно для виявлення розвідувальних дронів, видані доступи до керуванням дроном та такі системи дозволять самостійно вражати техніку супротивника.

З метою покращення взаємодії та забезпечення ефективного використання інформації, отриманої системою, перспективним є розвиток візуальних інтерфейсів та аналізу даних. Ключовим напрямком подальших досліджень залишається збільшення обсягу та покращення якості навчальних даних, включаючи дослідження методів синтезування даних та активного навчання. Також вартує уваги, вивчення можливостей апаратної акселерації обчислень з обмежених ресурсів, що сприятиме підвищенню продуктивності системи в реальних умовах експлуатації та не навантажуватиме девайс, на якому використовується дана модель.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Detection of Moving Objects in a Metro Rail CCTV Video Using YOLO Object Detection Models, A Abhinand, P A Aparna, Jaison Mulerikkal, Anu C Jaison, Anil Antony
2. Moving objects detection for mobile mapping, Hao Sun, Naser El-Sheimy, Cheng Wang
3. Detection of static moving objects using multiple nonparametric background models, Raquel Martinez, Daniel Berjon, Carlos Cuevas, Narciso Garcia
4. Detecting ground moving objects using panoramic system, Fuyuan Xu, Jing Wang, Guohua Gu
5. An Intelligent System for Road Moving Object Detection, Mejdi Ben Dkhil, Adel M Alimi, Ali Wali
6. Detection of Moving Objects in a Metro Rail CCTV Video Using YOLO Object Detection Models, A Abhinand, P A Aparna, Jaison Mulerikkal, Anu C Jaison, Anil Antony
7. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Hastie, Tibshirani, Friedman
8. Pattern Recognition and Machine Learning, Bishop
9. Deep Learning, Goodfellow, Bengio, Courville
10. Python for Data Analysis, McKinney
11. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, Géron
12. Data Science from Scratch: First Principles with Python, Grus
13. Applied Predictive Modeling, Kuhn, Johnson
14. Machine Learning Yearning, Ng
15. Continuously evolving rewards in an open-ended environment, Richard M.Bailey

16. NeurIPS 2024 Experiment on Improving the Paper-Reviewer Assignment
17. Computer Vision: Algorithms and Applications, Szeliski
18. Digital Image Processing, Gonzalez, Woods
19. Multiple View Geometry in Computer Vision, Hartley, Zisserman
20. Deep Learning for Computer Vision, Brownlee
21. Object Detection: 20 Years of Progress, Zhao et al.
22. Real-Time Object Detection with YOLO, YOLOv2, YOLOv3, Redmon et al.
23. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren et al.
24. Mask R-CNN, He et al.
25. International Journal of Computer Vision (IJCV)
26. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
27. Visual Tracking: An Experimental Survey, Yilmaz et al.
28. Online Multi-Object Tracking: A Benchmark, Milan et al.
29. DeepSORT: Simple Online and Realtime Tracking with a Deep Association Metric, Wojke et al.
30. A Survey on Deep Learning for Visual Object Tracking, Bhat et al.
31. Motion-Based Object Tracking: A Survey, Mittal, Davis
32. Human Detection and Tracking: A Survey, Gavrila
33. Artificial Intelligence: A Modern Approach, Russell, Norvig
34. Decision Making Under Uncertainty: Theory and Application, Morgan, Henrion
35. Reinforcement Learning: An Introduction, Sutton, Barto
36. APPLICATION OF DRONES WITH ARTIFICIAL INTELLIGENCE FOR MILITARY PURPOSES, ALEKSANDAR PETROVSKI
37. Pre-trained Deep Learning Networks for Advanced Visible Imagery Drone Detection and Recognition, Hassan J. Muhammad Bilal
38. Human pose recognition based on computer vision, Yahao Wu

39. TensorFlow Documentation (tensorflow.org)
40. PyTorch Documentation (pytorch.org)
41. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), High-Speed Tracking with Kernelized Correlation Filters, Henriques et al
42. Neural Information Processing Systems (NeurIPS), Long Short-Term Memory Networks for Anomaly Detection in Trajectories
43. IEEE Transactions on Image Processing, Robust Object Tracking via Sparsity-Based Collaborative Model
44. Journal of Machine Learning Research (JMLR), Online Learning of Object Detectors from Video Streams
45. Proceedings of the European Conference on Computer Vision (ECCV), Multi-Object Tracking by Lifted Multicut and Appearance Clustering
46. Artificial Intelligence Review, Intelligent Video Surveillance Systems: A Survey
47. CA-CMT: Coordinate Attention for Optimizing CMT Networks, Ji-Hyeon, Bang Sung-Wook, Park Jun-Yeong, Kim Jun Park
48. A Real Time Malaysian Sign Language Detection Algorithm Based on YOLOv3, Shafaf Ibrahim MARA, Itaza afiani Mohtar
49. Rich feature hierarchies for accurate object detection and semantic segmentation, Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik
50. Car detection from low-altitude UAV imagery with the faster R-CNN, Yongzheng xu, Guizhen Yu
51. Toward Interpretable Music Tagging with Self-Attention Minz Won, Sanghyuk Chun, Xavier Serra
52. Feature Pyramid Networks for Object Detection Tsung-Yi Lin, Piotr Dollar, Ross Girshick

ДОДАТКИ

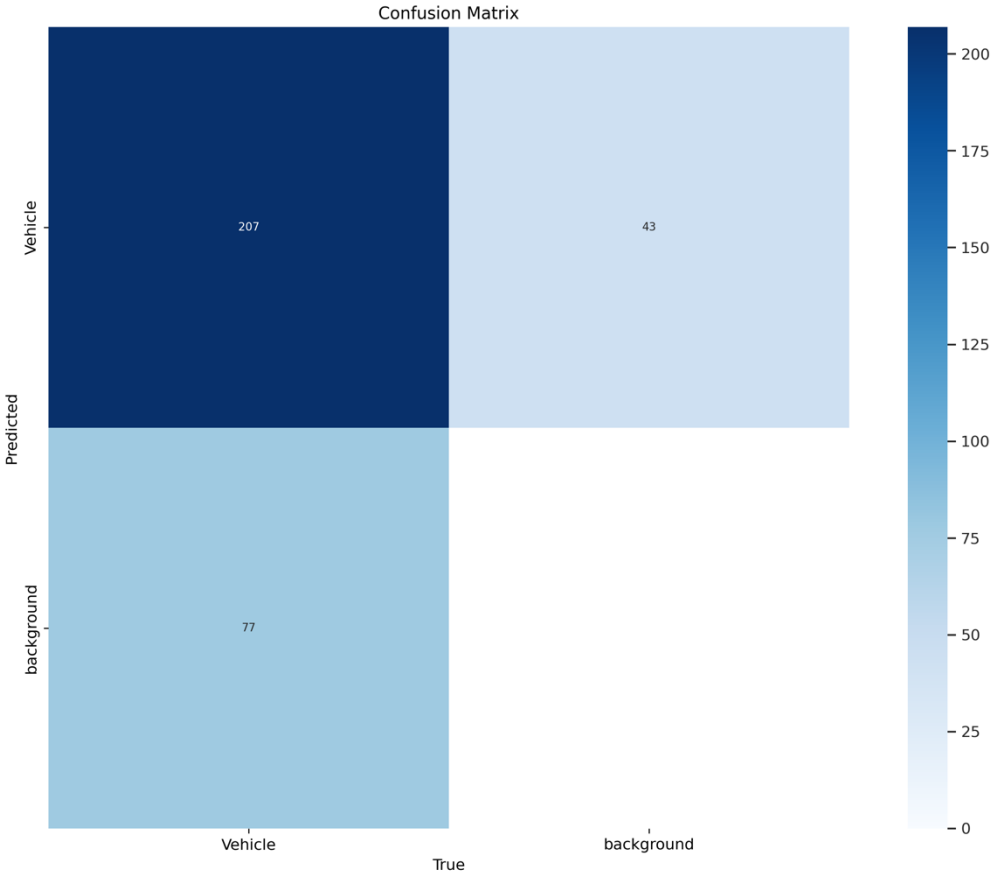

Рисунок 5.1. Confusin Matrix моделі YOLOv11

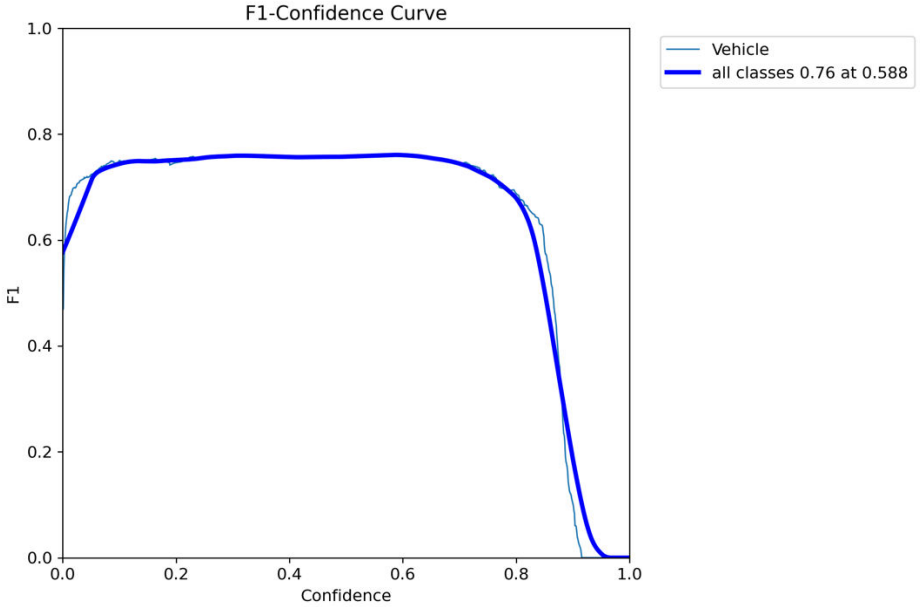

Рисунок 5.2. F1-Confidence Curve моделі YOLOv11

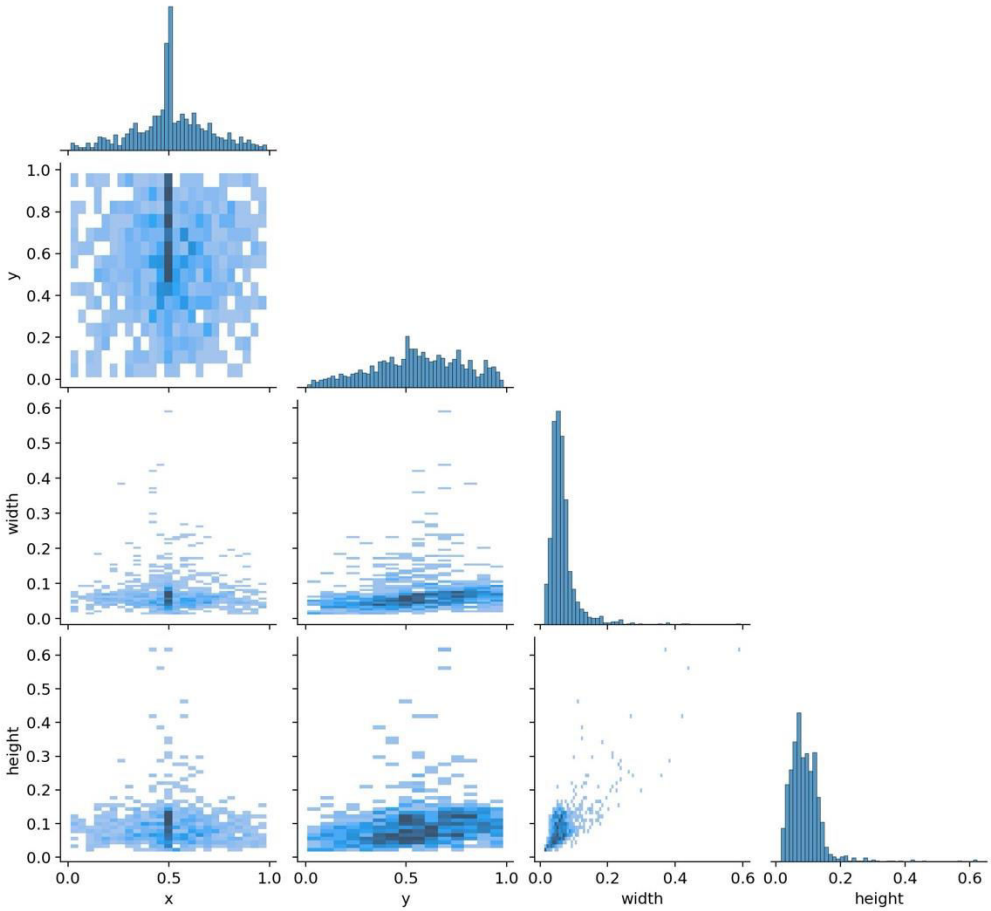

Рисунок 5.3. Корелограма моделі

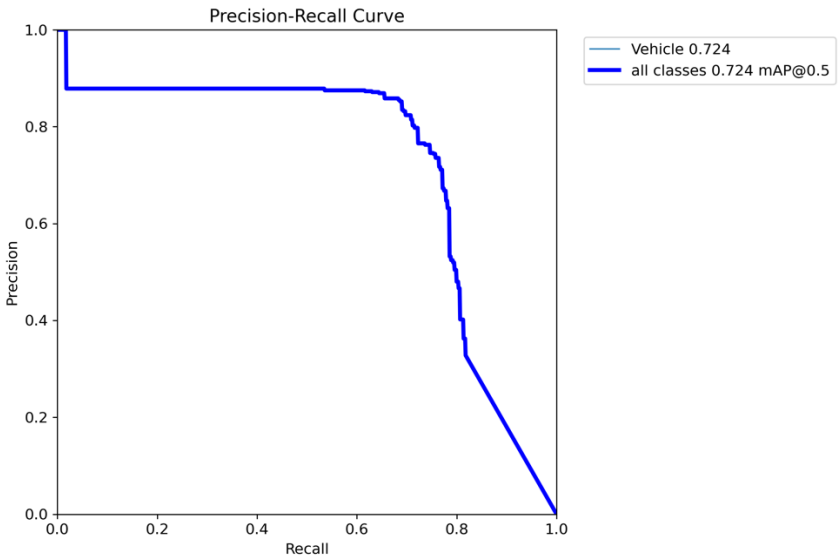

Рисунок 5.4. Крива Precisin-Recall моделі

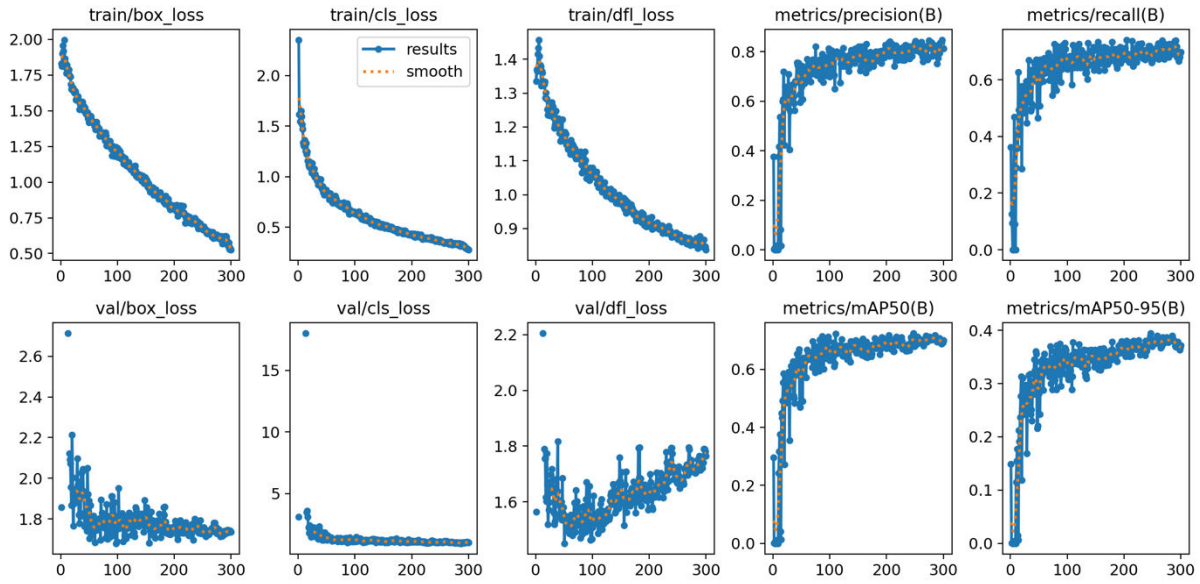

Рисунок 5.5. Деталізований результат тренування моделі

Рисунок 5.6. Приклади тренувальних передбачень моделлю YOLOv11

Рисунок 5.7. Приклади валідаційних передбачень моделлю YOLOv11

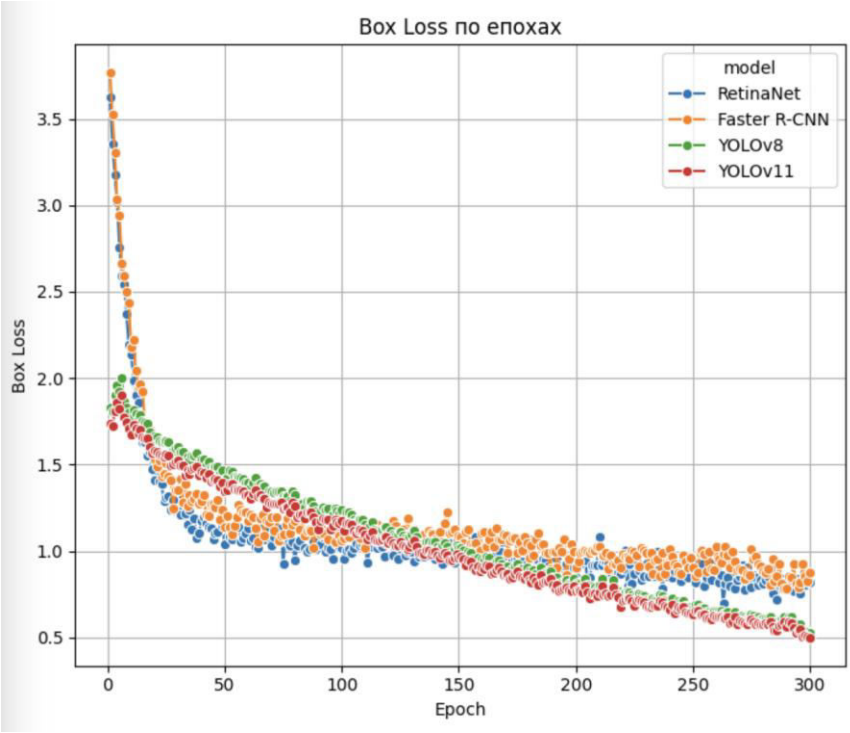

Рисунок 5.8. Box Loss моделей

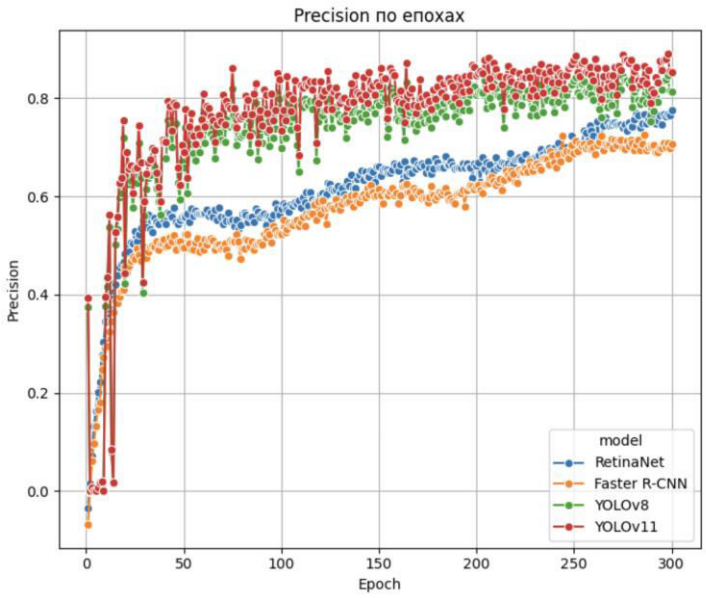

Рисунок 5.9. Precision моделей

Програмний код застосунку

```
import os
import cv2
from ultralytics import YOLO
from flask import Flask, request, jsonify, Response, stream_with_context
import tempfile
import threading
import time
import torch

# --- Конфігурація ---
DEFAULT_MODEL_PATH = 'yolov11n.pt' UPLOAD_FOLDER = 'uploads'
PROCESSED_FOLDER = 'processed_videos'
DEFAULT_TRAINING_DATA_YAML = 'data/dataset_config.yaml' # YAML файл для
    тренування за замовчуванням

os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(PROCESSED_FOLDER, exist_ok=True)
os.makedirs('data', exist_ok=True) # Для прикладу data.yaml

class YOLODetectionSystem:
   def __init__(self, model_path: str = None):
      self.model_path = model_path if model_path else DEFAULT_MODEL_PATH
      self.device = 'cuda' if torch.cuda.is_available() else 'cpu'
      self.model = YOLO(self.model_path)
      self.model.to(self.device)
      print(f"YOLO модель завантажена: {self.model_path} на пристрої: {self.device}")
```

```
def _get_model_for_training(self, base_model_path: str = DEFAULT_MODEL_PATH):

   if self.model_path == DEFAULT_MODEL_PATH or not
 os.path.exists(self.model_path):
      if not os.path.exists(base_model_path) and base_model_path ==
 DEFAULT_MODEL_PATH:
          _ = YOLO(DEFAULT_MODEL_PATH) # Завантажить, якщо немає
      return YOLO(base_model_path)
   return YOLO(self.model_path)


def train(self, data_yaml_path: str, epochs: int = 50, imgsz: int = 640, batch_size: int = 16,
 device_train: str = None, experiment_name: str = "yolo_train_run"):
   try:
      active_device = device_train if device_train else self.device

      # Використовуємо self.model, якщо це вже завантажена модель для донавчання
      training_model_instance = self._get_model_for_training()
      training_model_instance.to(active_device)


      results = training_model_instance.train(
         data=data_yaml_path,
         epochs=epochs,
         imgsz=imgsz,
         batch=batch_size,
         device=active_device,
         name=experiment_name,
         exist_ok=True # Дозволяє перезаписувати попередні експерименти з такою ж
 назвою
      )

      # Оновлюємо поточну модель на щойно натреновану
      self.model_path = results.save_dir / 'weights/best.pt' # results.save_dir це pathlib.Path
      self.model = YOLO(self.model_path)
      self.model.to(self.device) # Переміщуємо оновлену модель на робочий пристрій

      return {"status": "success", "message": "Навчання завершено.", "model_path":
 str(self.model_path)}
   except Exception as e:
      return {"status": "error", "message": f"Помилка під час навчання: {str(e)}"}

def process_image(self, image_path: str):
   try:
      results = self.model.track(image_path, persist=True, verbose=False,
 device=self.device)
      detections = []
      annotated_image_bytes = None
```

```
        if results and results[0].boxes:
            for box in results[0].boxes:
                class_id = int(box.cls)
                class_name = self.model.names[class_id]
                confidence = float(box.conf)
                xyxy = box.xyxy[0].cpu().tolist()
                obj_id = int(box.id.cpu().item()) if box.id is not None else None
                detections.append({
                    "object_id": obj_id, "class_id": class_id, "class_name": class_name,
                    "confidence": confidence, "bounding_box": xyxy
                })

            # Отримання анотованого зображення у вигляді байтів
            annotated_frame = results[0].plot()
            is_success, buffer = cv2.imencode(".jpg", annotated_frame)
            if is_success:
                annotated_image_bytes = buffer.tobytes()

        return {"status": "success", "detections": detections}, annotated_image_bytes
    except Exception as e:
        return {"status": "error", "message": str(e), "detections": []}, None

def stream_video_processing(self, video_source):
    cap = cv2.VideoCapture(video_source)
    if not cap.isOpened():
        error_msg = f"Помилка: Не вдалося відкрити відеоджерело: {video_source}"
        print(error_msg)
        # Для потокової передачі помилки, можна генерувати кадр з текстом помилки
        img = cv2.putText(cv2.zeros((480, 640, 3), dtype=np.uint8), error_msg, (50, 240),
 cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1, cv2.LINE_AA)
        _, buffer = cv2.imencode('.jpg', img)
        frame_bytes = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
        return

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

            results = self.model.track(frame, persist=True, verbose=False, device=self.device)
            annotated_frame = results[0].plot()

            is_success, buffer = cv2.imencode('.jpg', annotated_frame)
            if not is_success:
                continue
            frame_bytes = buffer.tobytes()
            yield (b'--frame\r\n'
```

```
                b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
        finally:
            cap.release()
            print(f"Відеопотік завершено для джерела: {video_source}")

    def process_video_to_file(self, input_video_path: str, output_filename_base: str =
     "processed_video"):
        output_filename = f"{output_filename_base}_{int(time.time())}.mp4"
        output_path = os.path.join(PROCESSED_FOLDER, output_filename)

        cap = cv2.VideoCapture(input_video_path)
        if not cap.isOpened():
            return {"status": "error", "message": f"Неможливо відкрити відеофайл:
     {input_video_path}"}

        frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = cap.get(cv2.CAP_PROP_FPS)
        if fps == 0: fps = 25 # Значення за замовчуванням, якщо fps не вдалося отримати

        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        out_writer = cv2.VideoWriter(output_path, fourcc, fps, (frame_width, frame_height))

        try:
            while True:
                ret, frame = cap.read()
                if not ret:
                    break
                results = self.model.track(frame, persist=True, verbose=False, device=self.device)
                annotated_frame = results[0].plot()
                out_writer.write(annotated_frame)

            return {"status": "success", "message": "Відео оброблено.", "output_path":
      output_path}
        except Exception as e:
            return {"status": "error", "message": str(e)}
        finally:
            cap.release()
            out_writer.release()


app = Flask(__name__)
yolo_system =
    YOLODetectionSystem(model_path=os.getenv('TRAINED_MODEL_PATH',
    DEFAULT_MODEL_PATH))

training_status = {"status": "idle", "message": "", "model_path": ""}
training_thread = None

@app.route('/train', methods=['POST'])
```

```
def endpoint_train():
    global training_thread, training_status, yolo_system
    if training_thread and training_thread.is_alive():
        return jsonify({"status": "error", "message": "Процес навчання вже запущено."}), 409

    try:
        data = request.get_json()
        if not data: return jsonify({"status": "error", "message": "Відсутні дані JSON."}), 400

        data_yaml = data.get('data_yaml_path', DEFAULT_TRAINING_DATA_YAML)
        epochs = int(data.get('epochs', 10)) # Менше епох для швидкого тестування API
        imgsz = int(data.get('imgsz', 640))
        batch = int(data.get('batch_size', 8))
        device_t = data.get('device', yolo_system.device)
        exp_name = data.get('experiment_name', 'api_train')

        if not os.path.exists(data_yaml):
            return jsonify({"status": "error", "message": f"Файл конфігурації даних
     {data_yaml} не знайдено."}), 400

        def run_training_task():
            global training_status, yolo_system
            training_status = {"status": "running", "message": f"Навчання запущено з
     {data_yaml}..."}
            result = yolo_system.train(data_yaml, epochs, imgsz, batch, device_t, exp_name)
            training_status.update(result)
            if result['status'] == 'success':
                print(f"Оновлено модель yolo_system на: {yolo_system.model_path}")


        training_thread = threading.Thread(target=run_training_task)
        training_thread.start()
with tf.device('/GPU'):
    results = model.train(data=os.path.join(os.getcwd(), "yolo/data.yaml"), epochs=400,
     imgsz=640)
        return jsonify({"status": "pending", "message": "Запит на навчання прийнято.
     Перевірте /train/status."}), 202
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500

@app.route('/train/status', methods=['GET'])
def endpoint_train_status():
    global training_status
    return jsonify(training_status)

@app.route('/predict/image', methods=['POST'])
def endpoint_predict_image():
    if 'file' not in request.files:
        return jsonify({"status": "error", "message": "Файл зображення не надано."}), 400
```

```
    file = request.files['file']
    if file.filename == '':
        return jsonify({"status": "error", "message": "Файл зображення не вибрано."}), 400

    try:
        temp_dir = tempfile.mkdtemp()
        temp_image_path = os.path.join(temp_dir, file.filename)
        file.save(temp_image_path)

        results_data, annotated_image_bytes = yolo_system.process_image(temp_image_path)

        os.remove(temp_image_path)
        os.rmdir(temp_dir)

        if results_data['status'] == 'success' and annotated_image_bytes:
            # Можна повернути JSON з детекціями та саме зображення окремо,
            # або закодувати зображення в base64 у JSON, або просто детекції
            # Для прикладу, повернемо детекції. Анотацію можна отримати через
     video_stream endpoint якщо потрібно.
             return jsonify(results_data)
        return jsonify(results_data), 500 if results_data['status'] == 'error' else 200
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500


@app.route('/predict/video_stream', methods=['GET'])
def endpoint_predict_video_stream():
    video_source_param = request.args.get('source', '0') # '0' для веб-камери за
     замовчуванням

    # Спробуємо конвертувати source в int (для ID камери), якщо не вийде - це шлях до
     файлу
    try:
        video_source = int(video_source_param)
    except ValueError:
        video_source = video_source_param # Це шлях до файлу

    return
     Response(stream_with_context(yolo_system.stream_video_processing(video_source)),
                 mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/predict/video_upload', methods=['POST'])
def endpoint_predict_video_upload():
    if 'file' not in request.files:
        return jsonify({"status": "error", "message": "Відеофайл не надано."}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({"status": "error", "message": "Відеофайл не вибрано."}), 400
```

```
    try:
        temp_dir = tempfile.mkdtemp()
        temp_video_path = os.path.join(temp_dir, file.filename)
        file.save(temp_video_path)

        results = yolo_system.process_video_to_file(temp_video_path,
     os.path.splitext(file.filename)[0])

        os.remove(temp_video_path) # Видаляємо тимчасовий завантажений файл
        os.rmdir(temp_dir) # Видаляємо тимчасову директорію

        return jsonify(results)
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500


if __name__ == '__main__':
    # Створення прикладу data/dataset_config.yaml, якщо його немає
    if not os.path.exists(DEFAULT_TRAINING_DATA_YAML):
        print(f"Створення прикладу файлу конфігурації:
     {DEFAULT_TRAINING_DATA_YAML}")
        os.makedirs(os.path.dirname(DEFAULT_TRAINING_DATA_YAML), exist_ok=True)
        with open(DEFAULT_TRAINING_DATA_YAML, 'w') as f:
            f.write("# Шляхи до тренувальних та валідаційних даних\n")
            f.write("train: ../dataset/images/train  # шлях до зображень для навчання\n")
            f.write("val: ../dataset/images/val    # шлях до зображень для валідації\n")
            f.write("# (опціонально) test: ../dataset/images/test # шлях до тестових
     зображень\n\n")
            f.write("# Кількість класів\n")
            f.write("nc: 2 # Наприклад, 2 класи\n\n")
            f.write("# Назви класів (має відповідати nc)\n")
            f.write("names: ['person', 'car'] # Замініть на ваші класи\n")

        dummy_dataset_root = 'dataset' # відносно до розташування dataset_config.yaml
        os.makedirs(os.path.join(dummy_dataset_root, 'images/train'), exist_ok=True)
        os.makedirs(os.path.join(dummy_dataset_root, 'images/val'), exist_ok=True)
        os.makedirs(os.path.join(dummy_dataset_root, 'labels/train'), exist_ok=True)
        os.makedirs(os.path.join(dummy_dataset_root, 'labels/val'), exist_ok=True)
        print(f"Будь ласка, налаштуйте {DEFAULT_TRAINING_DATA_YAML} та
     заповніть папки '{dummy_dataset_root}' даними.")

    import numpy as np # Потрібно для stream_video_processing, якщо виникає помилка
     відкриття відео
    app.run(host='0.0.0.0', port=5000, debug=True)
```