НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК 004.8

До захисту допущено:
В.О. Завідувача кафедри
_____ Михайло Новаторський
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Комп'ютерні системи та мережі» зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Спосіб виявлення об'єктів реального часу на основі моделей трансформерів»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІО-32мп
Пепелов Гліб Геннадійович _____

Керівник:
Проф. каф. ОТ, д.-ф.-м.н. с.н.с
Гордієнко Юрій Григорович _____

Консультант з нормоконтролю:
асистент каф. ОТ, к.т.н.
Кулаков Олексій Юрійович _____

Рецензент:
к.т.н, доцент кафедри ІСТ
Шимкович Володимир Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

Кафедра обчислювальної техніки
(повна назва кафедри)

Рівень вищої – освіти – другий (магістерський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма – «Комп'ютерні системи та мережі»

До захисту допущено:

В.О. Завідувача кафедри

_____ Михайло Новаторський
(підпис) (ім'я, прізвище)

«___»_____ 2024 р.

ЗАВДАННЯ

на дипломний проект студенту

Пепелову Глібу Геннадійовичу

1. Тема проекту *" Спосіб виявлення об'єктів реального часу на основі моделі трансформерів "*, керівник дисертації *Гордієнко Юрій Григорович, доктор фізико-математичних наук* , затверджені наказом по університету від *«08»* листопада 2024р. № 5018-с .

2. Термін подання студентом проекту *24.11.2024р.*

3. Вихідні дані до проекту: *технічне завдання, теоретичні дані, наукові публікації.*

4. Зміст пояснювальної записки: *огляд методів виявлення об'єктів реального часу на основі моделі трансформерів, тестування попередньо навченої моделі і аналіз отриманих результатів, написання алгоритму для побудови та тренування моделі для виявлення об'єктів на власному наборі*

даних, тестування навченої моделі та аналіз моделі та отриманих результатів.

5. Перелік завдань, які потрібно розробити: *огляд методів виявлення об'єктів реального часу на основі моделі трансформерів, тестування попередньо навченої моделі і аналіз отриманих результатів, написання алгоритму для побудови та тренування моделі для виявлення об'єктів на власному наборі даних, тестування навченої моделі та аналіз моделі та отриманих результатів.*

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1.	*проф. д.ф-м.н. с.н.с. Гордієнко Юрій Григорович*		
2.	*проф. д.ф-м.н. с.н.с. Гордієнко Юрій Григорович*		
3.	*проф. д.ф-м.н. с.н.с. Гордієнко Юрій Григорович*		
4.	*проф. д.ф-м.н. с.н.с. Гордієнко Юрій Григорович*		

7. Дата видачі завдання 01.09.2024 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1	*Обговорення теми дисертації. Визначення предмету дослідження*	*1.09.24-7.09.24*	
2	*Дослідження існуючих рішень та проблем виявлення об'єктів в досліджуваній області*	*22.09.24-28.09.24*	
3	*Побудова архітектурного рішення*	*29.09.24-05.10.24*	
4	*Програмна реалізація створеного алгоритму*	*06.10.24-10.10.24*	
5	*Розборка функціоналу по виявленню об'єктів реального часу на основі моделей трансформерів*	*10.10.24-19.10.24*	
6	*Розробка процесу побудови моделі для виявлення об'єктів реального часу*	*10.11.24-13.11.24*	

7	*Розробка стартам-проекту*	*14.11.24-18.11.24*	
8	*Оформлення магістерської дисертації.*	*19.11.21-24.11.21*	

Студент Гліб ПЕПЕЛОВ _____
(підпис)

Керівник Юрій ГОРДІЄНКО _____
(підпис)

РЕФЕРАТ

до магістерської дисертації

виконану на тему: «Спосіб виявлення об'єктів реального часу на основі моделей трансформерів»

студентом: Пепеловим Глібом Геннадійовичем

Робота складається із вступу, п’яти розділів та висновку. Загальний обсяг роботи: 110 аркушів основного тексту, 31 ілюстрацій, 26 таблиці. При підготовці використовувалася література з 24 різних джерел.

Актуальність. Виявлення об'єктів у реальному часі є однією з ключових задач комп'ютерного зору, що знаходить застосування у численних галузях, таких як автономні транспортні засоби, системи відеоспостереження, робототехніка та медична діагностика. Здатність системи ідентифікувати об'єкти в складних сценах і здійснювати це з високою точністю в умовах обмеженої обчислювальної потужності є критичною для забезпечення ефективності та безпеки таких застосувань.

Протягом останніх років досягнення в галузі глибокого навчання суттєво покращили якість та швидкість алгоритмів виявлення об'єктів. Особливої уваги заслуговують трансформерні моделі, які спочатку були розроблені для задач обробки природної мови, але швидко знайшли застосування і в інших областях, включаючи комп'ютерний зір. Завдяки своїй здатності моделювати довгострокові залежності та обробляти складні взаємодії між об'єктами в просторі, трансформери відкривають нові горизонти для виявлення об'єктів.

Мета і завдання дослідження. Метою магістерської роботи є підвищення ефективності існуючих моделей виявлення об’єктів реального часу за рахунок створення способу підбору найкращих архітектур моделей та гіперпараметрів процесу навчання, а також їх тестування на розширеному наборі для перевірки можливостей розроблених мереж в практичних умовах.

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- дослідження структури та архітектури моделей для виявлення об’єктів реального часу;
- створення способу підбору найкращих архітектур моделей та гіперпараметрів процесу навчання;
- ознайомлення з технологіями та наборами даних, що можуть бути використані;
- класифікація та узагальнення методів створення моделі;
- реалізація, навчання та тестування моделей з подальшим аналізом;
- перевірка можливостей покращення ефективності мереж шляхом зміни їх архітектур та гіперпараметрів навчання;
- збір зображень та їх аннотація для створення розширення набору даних;
- перевірка можливостей створених моделей на даних з доповнення.

Об’єкт дослідження – процес виявлення об’єктів реального часу.

Предмет дослідження - метод та програмне забезпечення для виявлення об’єктів реально часу на основі моделей трансформаторів.

Практична цінність. Практична цінність отриманих результатів дослідження полягає у підвищенні ефективності виявлення об'єктів у реальному часі завдяки використанню моделей трансформерів. Запропоновані методи дозволяють зменшити обчислювальні витрати та час обробки даних, що є критично важливим для застосувань у сферах оборони, моніторингу та безпеки. Використання трансформерів дозволяє оптимізувати процес обробки зображень з урахуванням їхніх просторових та масштабних особливостей. Результати можуть бути впроваджені в сучасні системи спостереження, що працюють у реальному часі, для автоматизації та підвищення продуктивності аналізу великих обсягів даних.

Особистий внесок виконавця. Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до

вирішення задачі конструювання алгоритму виявлення об’єктів реального часу на основі моделей транформерів. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Ключові слова: виявлення об’єктів, моделі трансформерів, об’єкти реального часу, класифікація об’єктів, набори даних, аугментація наборів даних, end-to-end навчання, згорткова архітектура.

ABSTRACT

to the master's thesis

entitled: "A Real-Time Object Detection Methods Based on Transformer Models"

Author: Hlib Pepelov

The thesis consists of an introduction, five chapters, and a conclusion. The total length is 110 pages of main text, including 31 illustrations and 26 tables. The research utilized 24 references from various sources.

Relevance. Real-time object detection is one of the key tasks in computer vision, widely applied in domains such as autonomous vehicles, surveillance systems, robotics, and medical diagnostics. The ability of a system to identify objects in complex scenes with high accuracy under limited computational resources is critical for ensuring the efficiency and safety of such applications.

In recent years, advancements in deep learning have significantly improved the accuracy and speed of object detection algorithms. Transformer models, initially designed for natural language processing, have rapidly found applications in other fields, including computer vision. Due to their ability to model long-term dependencies and process complex spatial interactions between objects, transformers open new opportunities for object detection.

Research Objectives and Tasks: The aim of the master's thesis is to enhance the efficiency of existing real-time object detection models by developing a method for selecting the optimal model architectures and hyperparameters for the training process, as well as testing them on an extended dataset to evaluate the capabilities of the developed networks in practical conditions.

To achieve this goal, the following tasks were formulated and accomplished:

- study the structure and architecture of models for real-time object detection;

- development of a method for selecting the optimal model architectures and hyperparameters for the training process
- familiarize with technologies and datasets suitable for use;
- classify and generalize methods for model development;
- implement, train, and test models with subsequent analysis;
- explore opportunities for improving network efficiency by modifying their architectures and training hyperparameters;
- collect images and annotate them to extend the dataset;
- evaluate the performance of the developed models on augmented data.

Research object - is to analyze existing real-time object detection methods and develop an improved method and software for real-time object detection based on transformer models.

Research subject - the process of real-time object detection. Subject of the research: methods and software for real-time object detection based on transformer models.

Practical value of the obtained research results lies in enhancing the efficiency of real-time object detection through the use of transformer models. The proposed methods reduce computational costs and processing time, which is critically important for applications in defense, monitoring, and security. The use of transformers allows optimizing the image processing workflow by considering their spatial and scale-specific features. The results can be implemented in modern real-time surveillance systems to automate and improve the performance of analyzing large volumes of data.

Personal contribution of the student: This master’s research is an independently conducted study that reflects the author's original approach and personally obtained theoretical and practical results related to solving the task of designing a real-time object detection algorithm based on transformer models. The formulation of the research goals and tasks was carried out in collaboration with the academic advisor.

Keywords: object detection, transformer models, real-time objects, object classification, datasets, data augmentation, end-to-end training, convolutional architecture.

ПОЯСНЮВАЛЬНА ЗАПИСКА
до магістерської дисертації

на тему: «Спосіб виявлення об’єктів реального часу на основі моделей трансформерів»

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

YOLO	(You Only Look Once) «Ви дивитеся лише раз»
DETR	(DEtection TRansformer) Трансформер виявлення
RT-DETR	(Real Time - DEtection TRansformer) Трансформер виявлення реального часу
CNN	(Convolutional Neural Network) Згорткова нейронна мережа
GPU	(Graphics Processing Unit) Графічний процесор
CPU	(Central Processing Unit) Центральний процесор
IoU	(Intersection over Union) Перетин до об'єднання
FFN	(Feedforward Neural Networks) Нейронна мережа прямого поширення
COCO	(Common Objects in COntext) Звичайні об’єкти в контексті
AP	(Average Precision) Середня точність

ВСТУП

Виявлення об'єктів — одна з основних задач комп'ютерного зору. Можливість виявляти та класифікувати об'єкти на зображеннях і у відео є необхідною для багатьох сучасних технологій. До них належать автономні транспортні засоби, системи спостереження, робототехніка та навіть генеративний штучний інтелект.

Більшість об'єктних детекторів на основі CNN в основному застосовуються лише для систем рекомендацій. Наприклад, пошук вільних паркувальних місць за допомогою міських відеокамер виконується повільними точними моделями, тоді як попередження про зіткнення автомобілів пов'язане з швидкими, але неточними моделями. Підвищення точності реального часу об'єктних детекторів дозволяє використовувати їх не лише для систем генерації рекомендацій, але й для автономного управління процесами та зменшення людського втручання. Операція об'єктних детекторів реального часу на звичайних графічних процесорах (GPU) дозволяє їх масове використання за доступною ціною. Найточніші сучасні нейронні мережі не працюють у реальному часі та вимагають великої кількості GPU для навчання з великою кількістю вхідних даних (mini-batch). Ці проблеми вирішуються шляхом створення CNN, яка працює в реальному часі на звичайному GPU і для якої навчання вимагає лише один звичайний GPU.

РОЗДІЛ 1
ОГЛЯД КОНТЕКСТУ ДОСЛІДЖЕННЯ, ПРОБЛЕМИ ВИЯВЛЕННЯ ОБ'ЄКТІВ І СУЧАСНИХ МЕТОДОВ ВИРІШЕННЯ

1.1 Проблема виявлення об’єктів

Виявлення об'єктів — одна з основних задач комп'ютерного зору. Можливість виявляти та класифікувати об'єкти на зображеннях і у відео є необхідною для багатьох сучасних технологій. До них належать автономні транспортні засоби, системи спостереження, робототехніка та навіть генеративний штучний інтелект.

Більшість об'єктних детекторів на основі CNN в основному застосовуються лише для систем рекомендацій. Наприклад, пошук вільних паркувальних місць за допомогою міських відеокамер виконується повільними точними моделями, тоді як попередження про зіткнення автомобілів пов'язане з швидкими, але неточними моделями. Підвищення точності реального часу об'єктних детекторів дозволяє використовувати їх не лише для систем генерації рекомендацій, але й для автономного управління процесами та зменшення людського втручання. Операція об'єктних детекторів реального часу на звичайних графічних процесорах (GPU) дозволяє їх масове використання за доступною ціною. Найточніші сучасні нейронні мережі не працюють у реальному часі та вимагають великої кількості GPU для навчання з великою кількістю вхідних даних (mini-batch). Ці проблеми вирішуються шляхом створення CNN, яка працює в реальному часі на звичайному GPU і для якої навчання вимагає лише один звичайний GPU.

1.2 Методи виявлення об’єктів реального часу

Далі наведено відомості про основні методи виявлення об’єктів реального часу та їх порівняння між собою з метою знаходження найбільш оптимального методу.

1.2.1 Огляд алгоритму YOLO

You Only Look Once (YOLO) — це алгоритм, здатний виявляти об'єкти з першого погляду, одночасно виконуючи їх детекцію та класифікацію.

Детекція об'єктів — одна з основних задач комп'ютерного зору. Можливість виявляти та класифікувати об'єкти на зображеннях і у відео є необхідною для багатьох сучасних технологій. До них належать автономні транспортні засоби, системи спостереження, робототехніка та навіть генеративний штучний інтелект.

YOLO став визначальною віхою завдяки своєму інноваційному підходу. Виконуючи детекцію та класифікацію об'єктів одночасно за один прохід через згорткову нейронну мережу (CNN), він поєднав швидкість у реальному часі з високою точністю. Його конвеєрна архітектура, поєднана з певними механізмами для визначення регіонів інтересу, перевершила всі попередні методи, зробивши їх застарілими. Справді, його структура кардинально відрізняється від традиційних технік. Замість того, щоб на початковому етапі пропонувати регіони інтересу, він розділяє вхідне зображення на сітку клітинок, кожна з яких відповідає за прогнозування координат обмежувальних рамок для виявлених об'єктів, а також ймовірностей їхньої належності до різних класів. Простіше кажучи, кожна клітинка прогнозує набір обмежувальних рамок та відповідні оцінки впевненості для кожного класу, безпосередньо на основі ознак, витягнутих CNN. Це усуває потребу проходити зображення кілька разів. Використання згорткової нейронної мережі для вилучення ознак із вхідного зображення є основою YOLO. CNN складається з декількох згорткових шарів і шарів згортання, що дозволяє захоплювати корисні шаблони та ознаки на різних просторових масштабах. Це дозволяє автоматично навчатися відповідним уявленням об’єктів та ефективно виконувати згорткові операції, що значно знижує обчислювальні витрати. Ще однією ключовою особливістю є використання "регіонів інтересу" або "якорів", які є попередньо визначеними обмежувальними рамками різних

розмірів і форм, що слугують як орієнтир для прогнозів. Кожна клітинка сітки асоціюється з певною кількістю якорів, що допомагає YOLO узагальнювати детекції на різні типи об’єктів і масштаби, значно підвищуючи точність детекції.

Безперечно, YOLO пропонує значні покращення порівняно з попередніми методами детекції об’єктів. Це дозволило йому стати одним із найбільш широко використовуваних алгоритмів у сфері комп’ютерного зору. Його головна сила — здатність миттєво виявляти об’єкти та зменшувати загальну кількість необхідних обчислень. Оптимізовано використання ресурсів, оскільки спільні ознаки обчислюються лише один раз. YOLO також вирізняється високою продуктивністю та точністю. Його конвеєрний підхід дозволяє узагальнювати виявлення об’єктів різних форм, що робить його стійким до широкого спектра сценаріїв. Крім того, він відмінно працює з обробкою зображень високої роздільної здатності, оскільки його ефективна архітектура дозволяє працювати з великими розмірами зображень без втрати швидкості. Це є значною перевагою для застосувань, таких як аерофотозйомка чи супутникове виявлення [2].

1.2.2 Огляд End-to-end Object Detectors

Детектори об'єктів у кінцевому режимі відомі своїми спрощеними конвеєрними структурами. Carion та ін. вперше запропонували детектор на основі трансформера, відомий як DETR, що привернув значну увагу завдяки своїм особливим характеристикам. Зокрема, DETR усуває використання вручну створених якорів та компоненту нефінального придушення (NMS). Натомість застосовується біпартитне зіставлення з прямим прогнозуванням набору об’єктів за принципом «один до одного».

Незважаючи на очевидні переваги, DETR має кілька проблем: повільне сходження при навчанні, високі обчислювальні витрати та складність оптимізації запитів. Було запропоновано чимало варіантів DETR для

вирішення цих проблем. Прискорення сходження. Deformable-DETR прискорює сходження під час навчання через багатомасштабні ознаки, підвищуючи ефективність механізму уваги. DAB-DETR та DN-DETR додатково покращують продуктивність, впроваджуючи ітераційне уточнення та навчання з приглушенням шуму. Group-DETR запроваджує групове призначення «багато до одного». Зниження обчислювальних витрат. Efficient DETR та Sparse DETR зменшують обчислювальні витрати за рахунок скорочення кількості шарів енкодера та декодера або кількості оновлюваних запитів. Lite DETR підвищує ефективність енкодера, зменшуючи частоту оновлення ознак нижнього рівня у спосіб чергування.

Оптимізація ініціалізації запитів. Conditional DETR та Anchor DETR знижують складність оптимізації запитів. Запропоновано відбір запитів для двоетапного DETR, а також застосування змішаного відбору запитів для кращої ініціалізації. Нинішні DETR залишаються обчислювально інтенсивними та не орієнтовані на детекцію в реальному часі. RT-DETR активно досліджує зниження обчислювальних витрат і намагається оптимізувати ініціалізацію запитів, перевершуючи сучасні детектори для роботи в реальному часі.

NMS (нефінальне придушення) є широко використовуваним алгоритмом пост-обробки в детекції об'єктів, що застосовується для усунення перекриваючихся вихідних рамок. Для роботи NMS необхідні два пороги: поріг довіри та поріг IoU. Конкретно, рамки зі значеннями нижче порогу довіри безпосередньо відфільтровуються, а коли IoU будь-яких двох рамок перевищує поріг IoU, рамка з нижчим балом буде відкинута. Цей процес виконується ітеративно, поки всі рамки кожної категорії не будуть оброблені. Таким чином, час виконання NMS в основному залежить від кількості рамок та двох порогів.

Для перевірки цього спостереження було використано YOLOv5 (з якорями) та YOLOv8 (без якорів) для аналізу. Спочатку підраховано кількість

рамок, що залишилися після фільтрації вихідних рамок із різними порогами довіри на одному й тому ж вхідному зображенні. Значення були відібрані в діапазоні від 0,001 до 0,25 для порогу довіри, щоб підрахувати кількість залишкових рамок обох детекторів, що було візуалізовано на стовпчиковій діаграмі, яка інтуїтивно показує, що NMS чутливий до своїх гіперпараметрів. Зі збільшенням порогу довіри відфільтровується більше прогнозованих рамок, що призводить до зменшення кількості рамок, для яких необхідно обчислювати IoU, отже, зменшує час виконання NMS.

Крім того, для оцінки точності було використано YOLOv8 на наборі даних COCO val2017 та протестовано час виконання порогу IoU.

Результати показують, що час виконання ядра EfficientNMS збільшується зі зменшенням порогу довіри або зі збільшенням порогу IoU. Це пояснюється тим, що високий поріг довіри безпосередньо фільтрує більше прогнозованих рамок, тоді як високий поріг IoU фільтрує менше прогнозованих рамок у кожному раунді відбору.

Неправильні пороги довіри можуть призвести до значних помилкових позитивних або негативних результатів детектора. При порозі довіри 0,001 та порозі IoU 0,7 YOLOv8 досягає найкращих результатів AP, але відповідний час NMS є вищим. Оскільки детектори YOLO зазвичай звітують про швидкість моделі та не враховують час NMS, необхідно встановити стандарт для оцінки швидкості в кінцевому режимі.

Для забезпечення справедливого порівняння швидкості в кінцевому режимі різних детекторів у реальному часі було встановлено стандарт швидкості в кінцевому режимі. Оскільки час виконання NMS залежить від вхідних даних, необхідно вибрати тестовий набір даних та розрахувати середній час виконання на декількох зображеннях. Вибрано набір даних COCO val2017 як стандарт, а також додано плагін пост-обробки NMS TensorRT для детекторів YOLO.

Конкретно, перевіряється середній час інференції детектора відповідно до порогів NMS для відповідної точності, взятої на тестовому наборі даних, за винятком операцій введення/виведення та копіювання пам'яті. Стандарт використовується для тестування швидкості в кінцевому режимі детекторів на основі якорів, таких як YOLOv5 та YOLOv7, а також детекторів без якорів, таких як PP-YOLOE, YOLOv6 та YOLOv8.

Для забезпечення справедливого порівняння швидкості в кінцевому режимі різних детекторів у реальному часі було встановлено стандарт швидкості в кінцевому режимі. Оскільки час виконання NMS залежить від вхідних даних, необхідно вибрати тестовий набір даних та розрахувати середній час виконання на декількох зображеннях. Вибрано набір даних COCO val2017 як стандарт, а також додано плагін пост-обробки NMS TensorRT для детекторів YOLO.

Конкретно, перевіряється середній час інференції детектора відповідно до порогів NMS для відповідної точності, взятої на тестовому наборі даних, за винятком операцій введення/виведення та копіювання пам'яті. Стандарт використовується для тестування швидкості в кінцевому режимі детекторів на основі якорів, таких як YOLOv5 та YOLOv7, а також детекторів без якорів, таких як PP-YOLOE, YOLOv6 та YOLOv8, на графічному процесорі T4 з використанням TensorRT FP16.

Результати свідчать про те, що детектори без якорів перевершують детектори на основі якорів з еквівалентною точністю для YOLO, оскільки перші потребують менше часу на виконання NMS, ніж другі. Причина цього полягає в тому, що детектори на основі якорів генерують більше прогнозованих рамок, ніж детектори без якорів (у наших тестованих детекторах виявлено утричі більше) [7].

1.2.3 Огляд алгоритму DETR

Два компоненти є критично важливими для прямих прогнозів наборів у виявленні: (1) втрата прогнозу набору, що змушує до унікального зіставлення між прогнозованими та істинними коробками; (2) архітектура, яка передбачає (в одному проході) набір об'єктів і моделює їхні взаємозв'язки.

DETR виводить набір з фіксованим розміром N прогнозів, в одному проході через декодер, де N налаштовується на значно більше значення, ніж типовий обсяг об'єктів у зображенні. Однією з основних труднощів навчання є оцінка прогнозованих об'єктів (клас, позиція, розмір) у відношенні до істини. Наша втрата забезпечує оптимальне біартитне зіставлення між прогнозованими та істинними об'єктами, а потім оптимізує специфічні для об'єкта (коробки) втрати.

Загальна архітектура DETR виявляється простою і зображена на рисунку 1.1 [7]. Вона містить три основні компоненти, які описано нижче: CNN-основу для виділення компактного представлення ознак, трансформер з архітектурою кодувальника-декодувальника та просту нейронну мережу з прямим проходженням (FFN), яка здійснює фінальний прогноз виявлення. На відміну від багатьох сучасних детекторів, DETR можна реалізувати в будь-якій фреймворці глибокого навчання, яка забезпечує загальну CNN-основу та реалізацію архітектури трансформера всього за кілька сотень рядків коду. Код для виведення DETR може бути реалізований менш ніж у 50 рядках в PyTorch. Сподіваємося, що простота нашого методу зацікавить нових дослідників у спільноті виявлення.

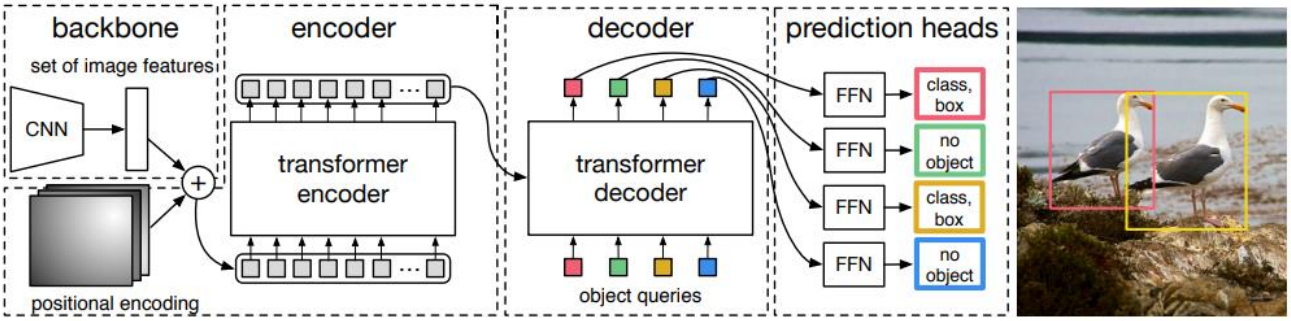

Рис. 1.1. Модель DETR використовує звичайний CNN як основний модуль для навчання 2D-представлення вхідного зображення. Спочатку модель сплющує отримане представлення та доповнює його позиційним кодуванням, після чого передає в трансформерний енкодер. Далі трансформерний декодер приймає на вхід невелику фіксовану кількість навчених позиційних векторів, які ми називаємо об'єктними запитами, і додатково звертається до виходу енкодера. Кожне вихідне представлення декодера передається в спільну мережу прямого поширення (FFN), яка прогнозує або виявлення (клас і координати обмежувальної рамки), або клас "немає об'єкта" [7].

Починаючи з початкового зображення (з 3 кольоровими каналами), звичайна CNN-основа генерує активаційну карту нижчої роздільної здатності.

Трансформатор-кодер. По-перше, 1x1 згортка зменшує розмірність каналу високорівневої активаційної карти f з C до меншої розмірності d, створюючи нову карту ознак z_0. Кодувальник очікує послідовність як вхід, тому ми стискаємо просторові розміри z_0 в одну вимірність, що призводить до карти ознак розміру dxHW. Кожен шар кодувальника має стандартну архітектуру і складається з модуля самовнимання з багатьма головами та мережі з прямим проходженням (FFN). Оскільки архітектура трансформера є перестановчо-інваріантною, ми доповнюємо її фіксованими позиційними кодуваннями, які додаються до вхідних даних кожного шару уваги.

Трансформер-декодувальник. Декодувальник дотримується стандартної архітектури трансформера, перетворюючи N векторів з розміром

d за допомогою механізмів самовнимання з багатьма головами та уваги кодувальник-декодувальник. Відмінність від оригінального трансформера полягає в тому, що наша модель декодує N об'єктів паралельно на кожному шарі декодувальника, тоді як Васвані та ін. використовують авторегресивну модель, яка прогнозує вихідну послідовність по одному елементу за раз. Для тих читачів, які не знайомі з цими концепціями, рекомендуємо звернутися до додаткових матеріалів.

Оскільки декодувальник також є перестановчо-інваріантним, вхідні вектори N повинні бути різними, щоб отримати різні результати. Ці вхідні вектори — це навчальні позиційні кодування, які ми називаємо об'єктними запитами, і, подібно до кодувальника, ми додаємо їх до вхідних даних кожного шару уваги. Об'єктні запити N перетворюються декодувальником у вихідний вектор.

Потім вони незалежно декодуються у координати рамок (bounding box) і класифікаційні мітки за допомогою нейронної мережі з прямим проходженням (описаної в наступному підрозділі), що дає в результаті N фінальних прогнозів. Використовуючи самовнимання та увагу кодувальник-декодувальник над цими векторами, модель здійснює глобальне міркування про всі об'єкти разом, враховуючи парні відношення між ними, при цьому маючи можливість використовувати все зображення як контекст.

Прогнозуючі нейронні мережі з прямим проходженням (FFNs)

Остаточний прогноз обчислюється за допомогою трьохшарової перцептронної мережі з функцією активації ReLU та прихованим розміром d, а також лінійного проекційного шару. FFN передбачає нормалізовані координати центру, висоту та ширину рамки відносно вхідного зображення, а лінійний шар прогнозує класову мітку за допомогою функції softmax.

Оскільки ми прогнозуємо фіксований набір з N рамок, де N зазвичай значно більше, ніж фактична кількість об'єктів, що представляють інтерес, використовується додаткова спеціальна класова мітка ∅, яка вказує на те, що

жоден об'єкт не виявлений у відповідному слоті. Цей клас виконує подібну роль до класу "фон" у стандартних підходах до виявлення об'єктів.

Було виявлено, що використання допоміжних втрат в декодері під час навчання є корисним, особливо для допомоги моделі у виведенні правильної кількості об'єктів кожного класу. Після кожного шару декодера додаються прогностичні нейронні мережі з прямим проходженням (FFN) та угорська втрата. Усі прогностичні FFN ділять свої параметри. Для нормалізації вхідних даних до прогностичних FFN з різних шарів декодера використовується додатковий спільний шар нормалізації [7].

1.2.4 Огляд алгоритму RT-DETR

Серія YOLO стала найпопулярнішою структурою для детекції об'єктів у реальному часі завдяки збалансованому компромісу між швидкістю та точністю. Однак спостерігається, що швидкість і точність YOLO знижуються через нефінальне придушення (NMS). Нещодавно детектори на основі трансформерів (DETR) запропонували альтернативу, яка усуває необхідність у NMS. Проте висока обчислювальна вартість обмежує їх практичність і не дозволяє повною мірою використовувати переваги відмови від NMS. Real-Time DEtection TRansformer (RT-DETR) — перший у своєму роді об'єктний детектор у реальному часі, що вирішує зазначену дилему. RT-DETR будується у два етапи, використовуючи просунутий DETR: спершу відбувається зосередження на збереженні точності з підвищенням швидкості, а потім на збереженні швидкості з підвищенням точності. Було розроблено ефективний гібридний енкодер для швидкої обробки ознак на різних масштабах, розділяючи взаємодію всередині масштабу та злиття між масштабами для підвищення швидкості. Також запропонувано вибір запитів із мінімальною невизначеністю, щоб забезпечити декодеру високоякісні початкові запити, що підвищує точність. RT-DETR підтримує гнучке налаштування швидкості шляхом регулювання кількості шарів декодера для адаптації до різних

сценаріїв без повторного навчання. RT-DETR-R50/R101 досягає 53,1% / 54,3% AP на COCO та 108 / 74 FPS на T4 GPU, перевершуючи попередні версії YOLO як за швидкістю, так і за точністю. Крім того, RT-DETR-R50 перевершує DINO-R50 на 2,2% AP за точністю та приблизно в 21 раз за FPS. Після попереднього навчання з Objects365, RT-DETR-R50 / R101 досягає 55,3% / 56,2% AP (рис. 1.2) [2].

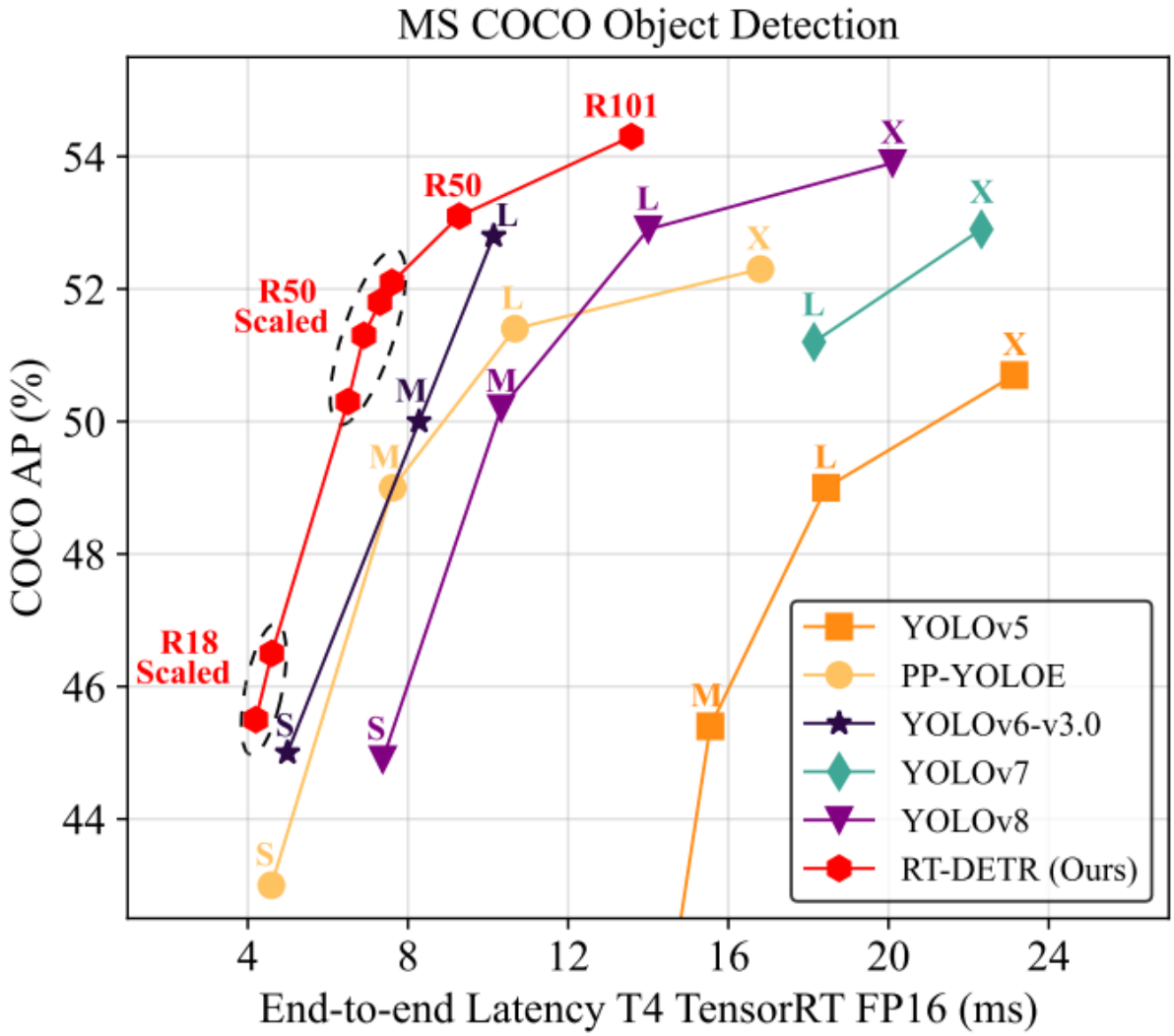

Рис. 1.2. У порівнянні з раніше розробленими реальними об'єктними детекторами, RT-DETR досягає передових показників продуктивності [2].

RT-DETR досягає оптимального компромісу між швидкістю та точністю. Зокрема, RT-DETR-R50 досягає 53,1% AP на COCO val2017 зі

швидкістю 108 FPS на T4 GPU, тоді як RT-DETR-R101 демонструє 54,3% AP і 74 FPS, перевершуючи моделі L та X попередніх YOLO-детекторів як за швидкістю, так і за точністю (рис. 1). Було розроблено масштабовані версії RT-DETR, які, завдяки зменшеним енкодеру та декодеру, перевершують легші моделі YOLO (моделі S та M). Крім того, RT-DETR-R50 перевершує DINO-Deformable-DETR-R50 на 2,2% AP (53,1% AP проти 50,9% AP) за точністю та приблизно в 21 раз за FPS (108 FPS проти 5 FPS), суттєво підвищуючи точність і швидкість DETR. Після попереднього навчання на Objects365 RT-DETR-R50/R101 досягає 55,3% / 56,2% AP, що забезпечує значне покращення результатів.

Основні внески узагальнюються так:

1. запропоновано перший об'єктний детектор у реальному часі — RT-DETR, що перевершує попередні моделі YOLO як за швидкістю, так і за точністю, а також усуває негативний вплив пост-обробки NMS на детекцію в реальному часі;
2. проведено кількісний аналіз впливу NMS на швидкість і точність YOLO-детекторів, встановлено стандарт для оцінки швидкості інференсу реальних детекторів у кінцевому режимі;
3. RT-DETR підтримує гнучке налаштування швидкості через регулювання кількості шарів декодера, що дозволяє адаптацію до різних сценаріїв без необхідності повторного навчання.

RT-DETR складається з основного модуля, ефективного гібридного енкодера та декодера на основі трансформера з допоміжними предиктивними головами. Загальний вигляд RT-DETR ілюструється на малюнку. Конкретно, функції з останніх трьох стадій основного модуля {S3, S4, S5} подаються в енкодер. Ефективний гібридний енкодер перетворює багатошарові функції в послідовність зображень через взаємодію функцій усередині шкали та злиття функцій між шкалами. Потім використовується вибір запитів з мінімальною невизначеністю для відбору фіксованої кількості функцій енкодера, які

слугують початковими запитами об'єктів для декодера. Нарешті, декодер з допоміжними предиктивними головами ітеративно оптимізує запити об'єктів для генерації категорій і рамок.

Аналіз обчислювальних вузьких місць. Введення багатошарових функцій прискорює зближення навчання та покращує продуктивність. Проте, незважаючи на те, що деформований механізм уваги знижує обчислювальні витрати, різке збільшення довжини послідовності все ще призводить до того, що енкодер стає обчислювальним вузьким місцем. За даними, наведені в інших дослідженнях, енкодер становить 49% GFLOPs, але вносить лише 11% в AP у Deformable-DETR.

Для подолання цього вузького місця спочатку проводиться аналіз обчислювальної надмірності, що присутня в багатошаровому трансформерному енкодері. Інтуїтивно зрозуміло, що функції високого рівня, які містять багатий семантичний контент про об'єкти, витягуються з функцій низького рівня, що робить зайвим виконання взаємодії функцій на конкатенованих багатошарових функціях. Тому розробляється набір варіантів з різними типами енкодера для підтвердження того, що одночасна взаємодія функцій усередині шкали та злиття функцій між шкалами є неефективною.

Спеціально, для експериментів використовується DINO-Deformable-R50 з меншим читачем даних та легшим декодером, застосованим у RT-DETR, і спочатку з варіанту A видаляється багатошаровий трансформерний енкодер. Потім до варіанту A вставляються різні типи енкодера для створення серії варіантів, детально описаних нижче (рис. 1.3) [2]:

- A → B: Варіант B вставляє одношаровий трансформерний енкодер в A, який використовує один шар трансформерного блоку. Багатошарові функції ділять енкодер для взаємодії функцій усередині шкали, а потім конкатенуються як вихід.

- B → C: Варіант C вводить злиття функцій між шкалами на основі B і подає конкатеновані функції в багатошаровий трансформерний енкодер для виконання одночасної взаємодії функцій усередині шкали та між шкалами.

- C → D: Варіант D декомпонує взаємодію усередині шкали та злиття між шкалами, використовуючи одношаровий трансформерний енкодер для першої та структуру PANet для другої.

- D → E: Варіант E покращує взаємодію усередині шкали та злиття між шкалами на основі D, впроваджуючи ефективний гібридний енкодер, розроблений автором.

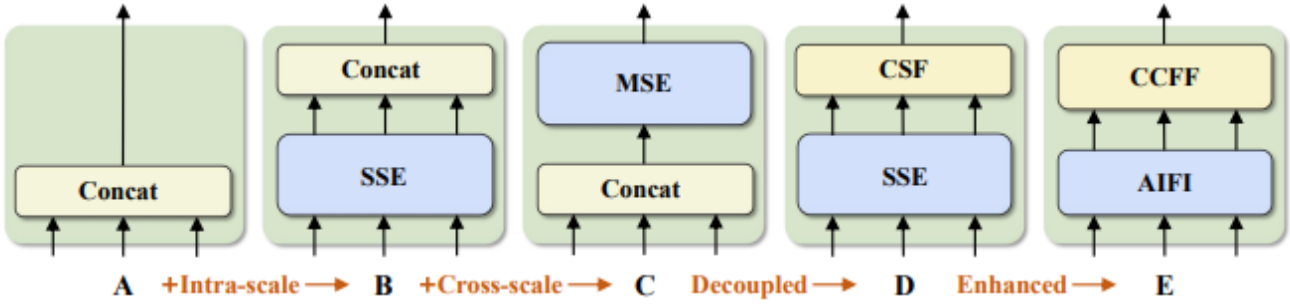

Рис. 1.3. Структура енкодера для кожного варіанту. SSE позначає одношаровий трансформерний енкодер, MSE — багатошаровий трансформерний енкодер, а CSF — злиття між масштабами. AIFI та CCFF — це два модулі, розроблені в нашому гібридному енкодері [2].

Гібридний дизайн. На основі наведеного аналізу була переглянута структура енкодера та запропоновано ефективний гібридний енкодер, що складається з двох модулів: взаємодії функцій на одному масштабі, основаної на увазі (AIFI), та злиття функцій між масштабами на основі CNN (CCFF). Конкретно, AIFI додатково знижує обчислювальні витрати на основі варіанту D, виконуючи взаємодію на одному масштабі лише на S5 за допомогою одношарового трансформерного енкодера. Причина в тому, що застосування операції самостійної уваги до високорівневих функцій з багатшими семантичними концепціями фіксує зв'язок між концептуальними об'єктами,

що полегшує локалізацію та розпізнавання об'єктів наступними модулями. Однак взаємодії на нижчих рівнях є непотрібними через відсутність семантичних концепцій і ризик дублювання та плутанини з високорівневими взаємодіями функцій. Для підтвердження цієї думки взаємодію на одному масштабі виконано лише на S5 у варіанті D, результати експериментів наведені в таблиці 1 (див. рядок DS5).

У порівнянні з варіантом D, варіант DS5 не тільки значно знижує затримки (на 35% швидше), але й покращує точність (на 0,4% AP вище). CCFF оптимізовано на основі модуля злиття функцій між масштабами, що включає кілька блоків злиття, які складаються з згорткових шарів. Роль блоку злиття полягає в об'єднанні двох сусідніх масштабних функцій в нову функцію, структура якого ілюструється на рисунку 1.4 [2]. Блок злиття містить дві згортки 1 × 1 для коригування кількості каналів, а також N RepBlocks, що складаються з RepConv, які використовуються для злиття функцій, і виходи з двох шляхів об'єднуються за допомогою поелементного додавання.

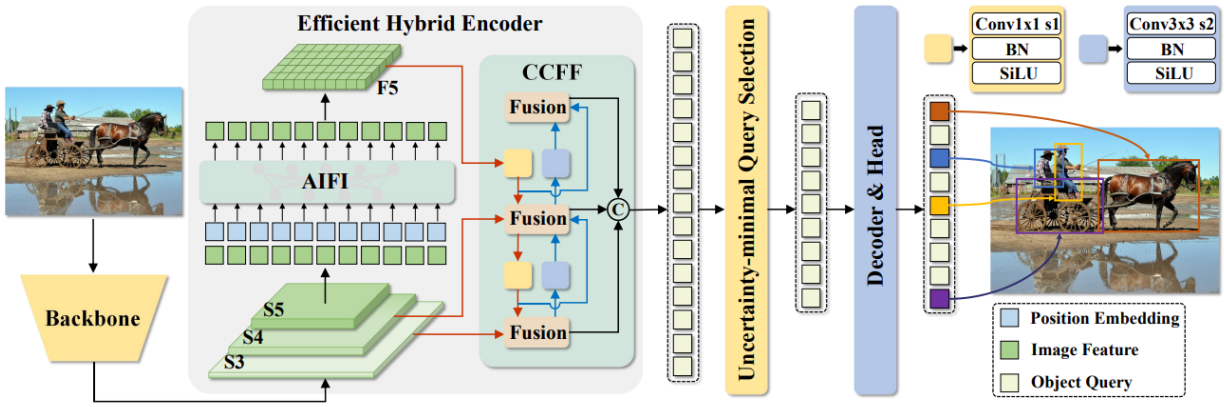

Рис. 1.4. Загальний огляд RT-DETR. Особливості з останніх трьох етапів базової моделі подаються в енкодер. Ефективний гібридний енкодер перетворює багатошарові функції в послідовність зображень за допомогою Взаємодії функцій на основі уваги (AIFI) та Злиття функцій між масштабами на основі CNN (CCFF). Потім відбувається вибір запитів з мінімальною невизначеністю, що вибирає фіксовану кількість функцій енкодера для

подальшого використання в якості початкових запитів для декодера. Нарешті, декодер з допоміжними прогнозними головками ітеративно оптимізує запити об'єктів для генерації категорій та обмежувальних рамок [2].

Для зменшення складності оптимізації запитів об'єктів у DETR кілька подальших робіт пропонують схеми вибору запитів. Усі вони використовують показник упевненості для вибору топ K характеристик з енкодера для ініціалізації запитів об'єктів (або лише позиційних запитів). Показник упевненості відображає ймовірність того, що характеристика містить об'єкти переднього плану. Однак детектору потрібно одночасно моделювати категорію та місцезнаходження об'єктів, обидва з яких визначають якість характеристик. Таким чином, показник продуктивності характеристики є прихованою змінною, яка спільно корелює з класифікацією та локалізацією. На основі цього аналізу поточний вибір запитів призводить до значного рівня невизначеності у вибраних характеристиках, що викликає субоптимальну ініціалізацію для декодера та заважає продуктивності детектора.

Для вирішення цієї проблеми пропонується схема вибору запитів з мінімальною невизначеністю, яка явно конструює та оптимізує епістемічну невизначеність для моделювання спільної прихованої змінної характеристик енкодера, що забезпечує високоякісні запити для декодера. Зокрема, невизначеність характеристики U визначається як різниця між передбаченими розподілами локалізації P та класифікації C. Щоб мінімізувати невизначеність запитів, невизначеність інтегрується в функцію втрат для оптимізації на основі градієнтів.

$$\mathcal{U}(\hat{\boldsymbol{\mathcal{X}}}) = \|\mathcal{P}(\hat{\boldsymbol{\mathcal{X}}}) - \mathcal{C}(\hat{\boldsymbol{\mathcal{X}}})\|, \hat{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^D \quad (2)$$

$$\mathcal{L}(\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\mathcal{Y}}) = \mathcal{L}_{box}(\hat{\mathbf{b}}, \mathbf{b}) + \mathcal{L}_{cls}(\mathcal{U}(\hat{\boldsymbol{\mathcal{X}}}), \hat{\mathbf{c}}, \mathbf{c}) \quad (3)$$

де Yˆ та Y позначають передбачення та істинні значення, Yˆ = {cˆ, bˆ}, cˆ та bˆ представляють категорію та обмежувальну рамку відповідно, Xˆ представляє характеристики енкодера.

Аналіз ефективності. Для аналізу ефективності схеми вибору запитів з мінімальною невизначеністю візуалізуються показники класифікації та IoU вибраних характеристик на COCO val2017. Показники візуалізовано на рис. 1.5 [2]. Створюється розсіювальна діаграма з показниками класифікації, що перевищують 0.5. Фіолетові та зелені точки представляють вибрані характеристики з моделі, навченої з використанням схеми вибору запитів з мінімальною невизначеністю та стандартної схеми вибору запитів відповідно. Чим ближче точка до верхнього правого кута діаграми, тим вищої якості відповідна характеристика, тобто ймовірність того, що передбачена категорія та обмежувальна рамка описують справжній об'єкт, є вищою. Верхня та права щільнісні криві відображають кількість точок для двох типів.

Найбільш помітною рисою розсіювальної діаграми є те, що фіолетові точки зосереджені у верхньому правому куті, тоді як зелені точки зосереджені у нижньому правому. Це свідчить про те, що схема вибору запитів з мінімальною невизначеністю забезпечує більшу кількість високоякісних характеристик енкодера. Крім того, проводиться кількісний аналіз двох схем вибору запитів. Кількість фіолетових точок перевищує кількість зелених точок на 138%, тобто є більше зелених точок з показником класифікації менше або дорівнює 0.5, що може розглядатися як низькоякісні характеристики. Крім того, кількість фіолетових точок перевищує кількість зелених на 120% при обох показниках, що перевищують 0.5. Таке ж висновок можна зробити з щільнісних кривих, де різниця між фіолетовими та зеленими точками є

найбільш помітною у верхньому правому куті діаграми. Кількісні результати ще більше підтверджують, що схема вибору запитів з мінімальною невизначеністю забезпечує більше характеристик з точною класифікацією та локалізацією для запитів, що, у свою чергу, підвищує точність детектора.

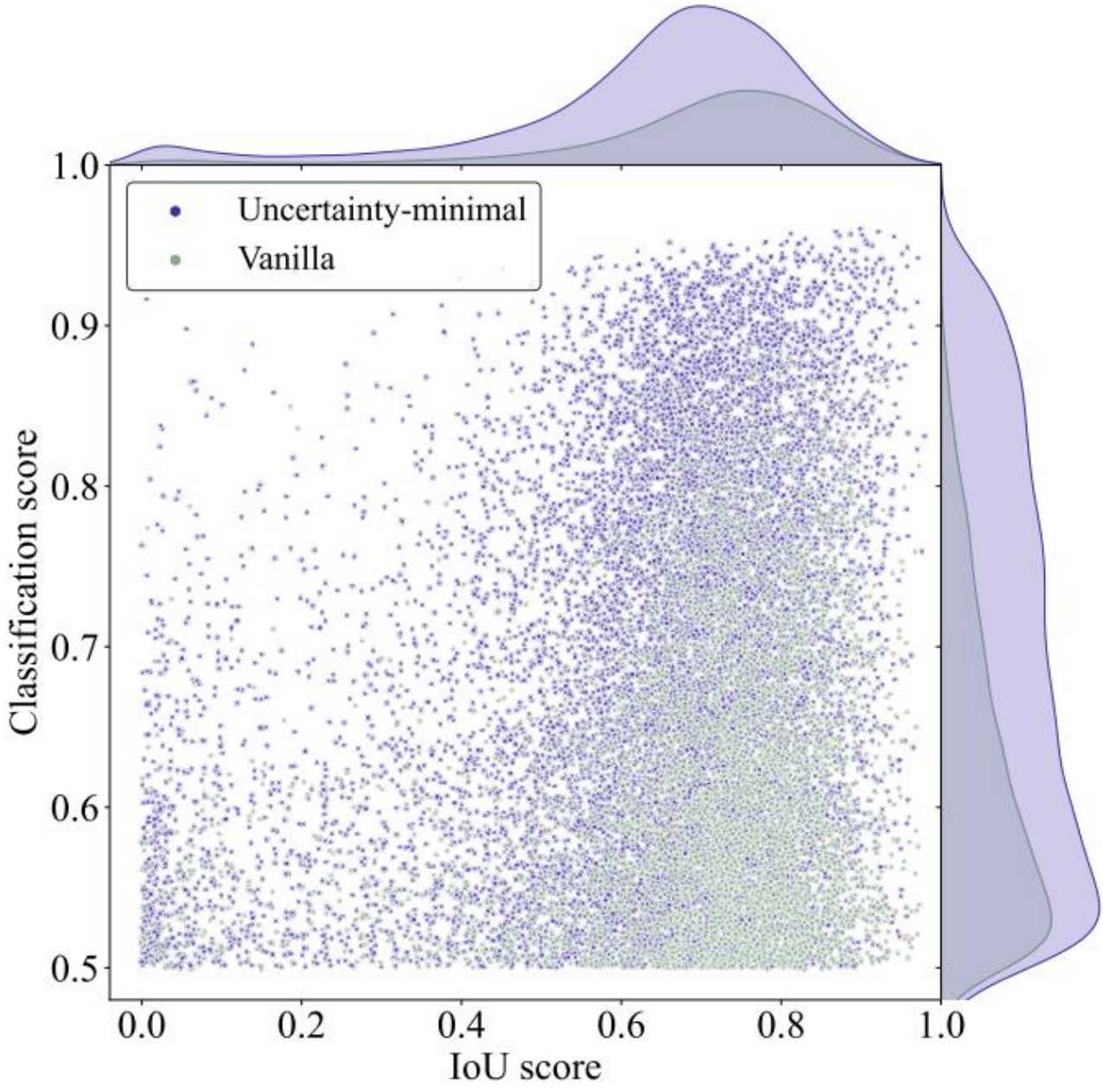

Рис. 1.5. Показники класифікації та IoU вибраних характеристик енкодера. Фіолетові та зелені точки представляють вибрані характеристики з моделі, навченої з використанням схеми вибору запитів з мінімальною невизначеністю та стандартної схеми вибору запитів відповідно [2].

RT-DETR підтримує гнучке масштабування, що відповідає потребам різних сценаріїв, оскільки реальні детектори зазвичай надають моделі в різних масштабах. Конкретно для гібридного енкодера контролюється ширина шляхом регулювання розміру векторів вбудовування та кількості каналів, а глибина—шляхом зміни кількості шарів Трансформера та RepBlocks. Ширина та глибина декодера можуть контролюватися шляхом регулювання кількості об'єктних запитів і шарів декодера. Крім того, швидкість RT-DETR підтримує гнучке регулювання шляхом зміни кількості шарів декодера. Спостережено, що видалення кількох шарів декодера в кінці має незначний вплив на точність, але значно підвищує швидкість виведення. Порівнюються RT-DETR з ResNet50 і ResNet101 з моделями L і X детекторів YOLO. Легші RT-DETR можуть бути спроектовані шляхом використання інших менших (наприклад, ResNet18/34) або масштабованих (наприклад, CSPResNet) основних архітектур із масштабованим енкодером і декодером. Порівняння масштабованих RT-DETR з легшими (S і M) детекторами YOLO наведено в додатку, де RT-DETR перевершують всі моделі S і M як за швидкістю, так і за точністю.

У таблиці 2 проведено порівняння RT-DETR з сучасними реальними детекторами (YOLO) та детекторами з кінцевим входом (DETR), де порівнюються лише моделі L і X детекторів YOLO, а моделі S і M наведені в додатку. RT-DETR і детектори YOLO мають однаковий розмір вхідних даних (640, 640), тоді як інші DETR використовують вхідний розмір (800, 1333). FPS (кадрів на секунду) вказано для T4 GPU з TensorRT FP16, при цьому для детекторів YOLO використано офіційно підготовлені моделі відповідно до запропонованого енд-то-енд бенчмарку швидкості. RT-DETR-R50 досягає 53,1% AP і 108 FPS, тоді як RT-DETR-R101 досягає 54,3% AP і 74 FPS, перевершуючи сучасні YOLO-детектори аналогічного масштабу та DETR з тією ж основною архітектурою як за швидкістю, так і за точністю. Експериментальні налаштування наведені в додатку.

Порівнюються енд-то-енд швидкість та точність RT-DETR з детекторами YOLO. Порівнюються RT-DETR з YOLOv5, PP-YOLOE, YOLOv6 (далі - YOLOv6), YOLOv7 та YOLOv8. У порівнянні з YOLOv5-L / PP-YOLOE-L / YOLOv6-L, RT-DETR-R50 покращує точність на 4,1% / 1,7% / 0,3% AP, збільшує FPS на 100,0% / 14,9% / 9,1% і зменшує кількість параметрів на 8,7% / 19,2% / 28,8%. У порівнянні з YOLOv5-X / PP-YOLOE-X, RT-DETR-R101 покращує точність на 3,6% / 2,0%, збільшує FPS на 72,1% / 23,3% і зменшує кількість параметрів на 11,6% / 22,4%. У порівнянні з YOLOv7-L / YOLOv8-L, RT-DETR-R50 покращує точність на 1,9% / 0,2% AP і збільшує FPS на 96,4% / 52,1%. У порівнянні з YOLOv7-X / YOLOv8-X, RT-DETR-R101 покращує точність на 1,4% / 0,4% AP і збільшує FPS на 64,4% / 48,0%. Це свідчить про те, що RT-DETR досягає сучасної продуктивності в реальному часі.

Також проводиться порівняння RT-DETR з існуючими DETR, що використовують ту ж основну архітектуру. Швидкість DINO-Deformable-DETR тестується відповідно до налаштувань точності, визначених на COCO val2017, тобто швидкість тестується з TensorRT FP16, а вхідний розмір становить (800, 1333). У таблиці 1.1 [2] показано, що RT-DETR перевершує всі DETR з тією ж основною архітектурою за швидкістю та точністю. У порівнянні з DINO-Deformable-DETR-R50, RT-DETR-R50 покращує точність на 2,2% AP і швидкість на 21 раз (108 FPS проти 5 FPS), що є суттєвими покращеннями.

Таблиця 1.1.

Порівняння з SOTA (тільки моделі L і X детекторів YOLO). Швидкість інших DETR не тестується, за винятком DINO-Deformable-DETR для порівняння, оскільки вони не є детекторами в реальному часі. Наш RT-DETR перевершує найсучасніші детектори YOLO та DETR як за швидкістю, так і за точністю [2].

Model	Backbone	#Epochs	#Params (M)	GFLOPs	$FPS_{bs=1}$	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
Real-time Object Detectors											
YOLOv5-L [11]	-	300	46	109	54	49.0	67.3	-	-	-	-
YOLOv5-X [11]	-	300	86	205	43	50.7	68.9	-	-	-	-
PPYOLOE-L [40]	-	300	52	110	94	51.4	68.9	55.6	31.4	55.3	66.1
PPYOLOE-X [40]	-	300	98	206	60	52.3	69.9	56.5	33.3	56.3	66.4
YOLOv6-L [16]	-	300	59	150	99	52.8	70.3	57.7	34.4	58.1	70.1
YOLOv7-L [38]	-	300	36	104	55	51.2	69.7	55.5	35.2	55.9	66.7
YOLOv7-X [38]	-	300	71	189	45	52.9	71.1	57.4	36.9	57.7	68.6
YOLOv8-L [12]	-	-	43	165	71	52.9	69.8	57.5	35.3	58.3	69.8
YOLOv8-X [12]	-	-	68	257	50	53.9	71.0	58.7	35.7	59.3	70.7
End-to-end Object Detectors											
DETR-DC5 [4]	R50	500	41	187	-	43.3	63.1	45.9	22.5	47.3	61.1
DETR-DC5 [4]	R101	500	60	253	-	44.9	64.7	47.7	23.7	49.5	62.3
Anchor-DETR-DC5 [39]	R50	50	39	172	-	44.2	64.7	47.5	24.7	48.2	60.6
Anchor-DETR-DC5 [39]	R101	50	-	-	-	45.1	65.7	48.8	25.8	49.4	61.6
Conditional-DETR-DC5 [27]	R50	108	44	195	-	45.1	65.4	48.5	25.3	49.0	62.2
Conditional-DETR-DC5 [27]	R101	108	63	262	-	45.9	66.8	49.5	27.2	50.3	63.3
Efficient-DETR [42]	R50	36	35	210	-	45.1	63.1	49.1	28.3	48.4	59.0
Efficient-DETR [42]	R101	36	54	289	-	45.7	64.1	49.5	28.2	49.1	60.2
SMCA-DETR [9]	R50	108	40	152	-	45.6	65.5	49.1	25.9	49.3	62.6
SMCA-DETR [9]	R101	108	58	218	-	46.3	66.6	50.2	27.2	50.5	63.2
Deformable-DETR [45]	R50	50	40	173	-	46.2	65.2	50.0	28.8	49.2	61.7
DAB-Deformable-DETR [23]	R50	50	48	195	-	46.9	66.0	50.8	30.1	50.4	62.5
DAB-Deformable-DETR++ [23]	R50	50	47	-	-	48.7	67.2	53.0	31.4	51.6	63.9
DN-Deformable-DETR [17]	R50	50	48	195	-	48.6	67.4	52.7	31.0	52.0	63.7
DN-Deformable-DETR++ [17]	R50	50	47	-	-	49.5	67.6	53.8	31.3	52.6	65.4
DINO-Deformable-DETR [44]	R50	36	47	279	5	50.9	69.0	55.3	34.6	54.1	64.6
Real-time End-to-end Object Detector (ours)											
RT-DETR	R50	72	42	136	**108**	**53.1**	**71.3**	**57.7**	34.8	58.0	70.0
RT-DETR	R101	72	76	259	**74**	**54.3**	**72.7**	58.6	36.0	58.8	**72.1**

Оцінка ефективності варіантів. Ми оцінюємо показники варіантів, розроблених у розділі 4.2, включаючи AP (навчання з конфігурацією 1×), кількість параметрів та затримку, у таблиці 1.2 [2]. Порівняно з базовим варіантом A, варіант B підвищує точність на 1,9% AP та збільшує затримку на 54%. Це доводить, що взаємодія між ознаками всередині шкали є важливою,

але одночасний трансформер з одною шкалою є обчислювально витратним. Варіант C забезпечує поліпшення точності на 0,7% AP у порівнянні з B та збільшує затримку на 20%. Це показує, що крос-шкальна фузія ознак також є необхідною, але трансформер з багатьма шкалами потребує вищих обчислювальних витрат. Варіант D забезпечує покращення точності на 0,8% AP у порівнянні з C, але зменшує затримку на 8%, що свідчить про те, що розділення взаємодії всередині шкали та крос-шкальної фузії не лише зменшує обчислювальні витрати, але й покращує точність. Порівняно з варіантом D, DS5 зменшує затримку на 35%, але забезпечує покращення на 0,4% AP, демонструючи, що внутрішня взаємодія нижчого рівня ознак не є необхідною. Нарешті, варіант E забезпечує покращення на 1,5% AP у порівнянні з D. Незважаючи на збільшення кількості параметрів на 20%, затримка зменшується на 24%, що робить енкодер більш ефективним. Це свідчить про те, що наш гібридний енкодер досягає кращого компромісу між швидкістю та точністю.

Таблиця 1.2.

Показники набору варіантів, зображених на рисунку 2 [2].

Variant	AP (%)	#Params (M)	Latency (ms)
A	43.0	31	7.2
B	44.9	32	11.1
C	45.6	32	13.3
D	46.4	35	12.2
D_{s_5}	46.8	35	7.9
E	47.9	42	9.3

Проведено абляційне дослідження з вибору запитів з мінімальною невизначеністю, результати якого наведені для RT-DETR-R50 з конфігурацією 1× у таблиці 1.3. [2]. Вибір запитів у RT-DETR вибирає топ K (K = 300) ознак з енкодера відповідно до оцінок класифікації як контентні запити, а прогностичні коробки, що відповідають вибраним ознакам, використовуються як початкові позиційні запити. Порівнюються ознаки енкодера, вибрані двома схемами вибору запитів на COCO val2017, та обчислюються пропорції оцінок класифікації, що перевищують 0.5, і оцінок класифікації та IoU, що перевищують 0.5. Результати показують, що ознаки енкодера, вибрані за схемою вибору з мінімальною невизначеністю, не лише збільшують пропорцію високих оцінок класифікації (0.82% проти 0.35%), але й забезпечують більше високоякісних ознак (0.67% проти 0.30%). Також оцінюється точність детекторів, навчений за двома схемами вибору запитів на COCO val2017, де вибір з мінімальною невизначеністю досягає покращення на 0.8% AP (48.7% AP проти 47.9% AP).

Таблиця 1.3.

Результати абляційного дослідження з вибору запитів з мінімальною невизначеністю. Propcls та Propboth представляють частку запитів з оцінкою класифікації та обома оцінками, що перевищують 0.5 відповідно [2].

Query selection	AP (%)	$\text{Prop}_{cls}\uparrow$ (%)	$\text{Prop}_{both}\uparrow$ (%)
Vanilla	47.9	0.35	0.30
Uncertainty-minimal	48.7	0.82	0.67

Таблиця 1.4. [2] показує латентність виводу та точність кожного декодерного шару RT-DETR-R50, навченої з різною кількістю декодерних шарів. Коли кількість декодерних шарів встановлюється на рівні 6, RT-DETR-R50 досягає найкращої точності 53.1% AP. Крім того, спостерігається, що різниця в точності між сусідніми декодерними шарами поступово зменшується з підвищенням індексу декодерного шару. Наприклад, у колонці RT-DETR-R50-Det6 використання 5-го декодерного шару для виводу втрачає лише 0.1% AP (53.1% AP проти 53.0% AP) в точності, при цьому зменшуючи латентність на 0.5 мс (9.3 мс проти 8.8 мс). Таким чином, RT-DETR підтримує гнучку настройку швидкості, регулюючи кількість декодерних шарів без повторного навчання, що підвищує його практичність.

Таблиця 1.4.

Результати дослідження декодера. ID вказує на індекс декодерного шару. Detk представляє детектор з k декодерними шарами. Всі результати наведені для RT-DETR-R50 з конфігурацією 6× [2].

ID	AP(%)				Latency (ms)
	Det^4	Det^5	Det^6	Det^7	
7	-	-	-	52.6	9.6
6	-	-	**53.1**	52.6	9.3
5	-	52.9	53.0	52.5	8.8
4	52.7	52.7	52.7	52.1	8.3
3	52.4	52.3	52.4	51.5	7.9
2	51.6	51.3	51.3	50.6	7.5
1	49.6	48.8	49.1	48.3	7.0

Обмеження. Незважаючи на те, що запропонований RT-DETR перевершує сучасні детектори в реальному часі та кінцеві детектори схожого розміру як за швидкістю, так і за точністю, він має таке ж обмеження, як і інші DETR, а саме: продуктивність на малих об'єктах все ще нижча, ніж у потужних детекторів в реальному часі. Згідно з Таблицею 1, RT-DETR-R50 на 0,5% AP нижче, ніж найвищий APvalS у L моделі (YOLOv8-L), а RT-DETR-R101 на 0,9% AP нижче, ніж найвищий APvalS у X моделі (YOLOv7-X). Сподіваємося, що ця проблема буде вирішена в майбутніх роботах.

Існуючі великі моделі DETR продемонстрували вражаючу продуктивність на COCO test-dev. Запропонований RT-DETR на різних масштабах зберігає декодери, які є однорідними з іншими DETR, що робить можливим дистиляцію нашого легкого детектора з високою точністю, підготовленого на великих моделях DETR. Вважається, що це є однією з переваг RT-DETR над іншими детекторами в реальному часі і може стати цікавим напрямком для майбутніх досліджень [2].

Висновки

У цьому розділі проведено детальний аналіз ефективності моделі RT-DETR, зокрема її здатності до вибору запитів з мінімальною невизначеністю та порівнянням її з іншими сучасними детекторами, такими як YOLO та DETR. Основна увага була приділена впливу вибору запитів на точність і швидкість детекції, а також на зменшення обчислювальних витрат.

Результати показали, що схема вибору запитів з мінімальною невизначеністю забезпечує значні переваги в порівнянні з традиційними методами. Вибір фіч з високими оцінками класифікації та IoU дозволяє досягти кращої точності детекції, зокрема за рахунок збільшення кількості високоякісних характеристик, що приводить до покращення показників AP (Average Precision). Крім того, схема з мінімальною невизначеністю сприяє більш ефективному використанню ресурсів моделі, знижуючи кількість низькоякісних характеристик.

Масштабування RT-DETR також виявилось успішним, оскільки модель здатна підтримувати гнучку настройку, що дозволяє адаптувати її до різних сценаріїв. Порівняння з детекторами YOLO та іншими модифікаціями DETR показало, що RT-DETR досягає кращих результатів у швидкості (FPS) та точності, підтверджуючи її ефективність у реальному часі. Зокрема, RT-DETR-R50 продемонстрував вищу точність і значно більшу швидкість виведення порівняно з аналогічними моделями YOLO, зменшуючи кількість параметрів і підвищуючи ефективність використання ресурсів.

Крім того, проведені абляційні дослідження та аналіз різних варіантів моделі підтвердили важливість компонентів, таких як крос-шкальна фузія ознак і оптимізація декодерів. Порівняння варіантів показало, що зниження обчислювальних витрат не завжди призводить до погіршення точності, а ефективна комбінація різних архітектурних підходів дозволяє досягти кращого компромісу між швидкістю і точністю.

Однак, незважаючи на значні досягнення, RT-DETR має певні обмеження, зокрема виявлену нижчу ефективність на малих об'єктах у порівнянні з деякими іншими детекторами. Ця проблема, ймовірно, буде вирішена в майбутніх роботах, що дозволить RT-DETR стати ще більш конкурентоспроможною моделлю для задач детекції в реальному часі.

Загалом, результати дослідження підтверджують високий потенціал RT-DETR як ефективного детектора, який забезпечує гармонійне поєднання точності та швидкості, а також є перспективним інструментом для застосувань у реальному часі, таких як військова авіація та інші сценарії з високими вимогами до обробки зображень.

РОЗДІЛ 2
СТВОРЕННЯ СПОСОБУ ВИЯВЛЕННЯ ОБ’ЄКТІВ РЕАЛЬНОГО ЧАСУ НА ОСНОВІ МОДЕЛЕЙ ТРАНСФОРМЕРІВ

2.1 Набори даних

2.1.1 Набір даних COCO

Однією з основних цілей комп'ютерного зору є розуміння візуальних сцен. Розуміння сцени включає численні завдання, такі як розпізнавання наявних об'єктів, локалізація об'єктів у 2D та 3D, визначення атрибутів об'єктів і сцени, характеристика відносин між об'єктами та надання семантичного опису сцени. Сучасні набори даних для класифікації та детектування об'єктів допомагають досліджувати перші виклики, пов'язані з розумінням сцени. Наприклад, набір даних ImageNet, який містить безпрецедентну кількість зображень, нещодавно забезпечив прориви як у дослідженнях класифікації, так і в детектуванні об'єктів. Спільнота також створила набори даних, що містять атрибути об'єктів, атрибути сцен, ключові точки та інформацію про 3D-сцени. Це призводить до очевидного запитання: які набори даних найкраще сприятимуть нашому просуванню до остаточної мети — розуміння сцени.

Набір даних, досліджений далі, вирішує три основні дослідницькі проблеми в розумінні сцени: виявлення неіконичних поглядів (або неканонічних перспектив) об'єктів, контекстуальне міркування між об'єктами та точна 2D-локалізація об'єктів. Для багатьох категорій об'єктів існує іконичний вигляд. Наприклад, при виконанні пошуку зображень в Інтернеті для категорії об'єктів "велосипед" найкращі приклади, що знаходяться на верхніх позиціях, з'являються в профіль, без перешкод, близько до центру охайно складеного фото. Відомо, що сучасні системи розпізнавання працюють досить добре на іконичних виглядах, але стикаються з труднощами у розпізнаванні об'єктів в інших умовах.

У дослідженні основними завданнями комп'ютерного зору є розуміння візуальних сцен, що включає в себе такі задачі, як розпізнавання об'єктів,

локалізація об'єктів у 2D та 3D, визначення атрибутів об'єктів та сцен, характеристика відносин між об'єктами та надання семантичного опису сцени. Існуючі набори даних для класифікації та виявлення об'єктів допомагають вивчати перші виклики, пов'язані з розумінням сцен. Зокрема, набір даних ImageNet дозволив досягти значних успіхів у дослідженнях розпізнавання об'єктів.

Набрі даних COCO вирішує три основні проблеми в розумінні сцен: виявлення неканонічних видів об'єктів, контекстуальне мислення між об'єктами та точна 2D-локалізація об'єктів. Для багатьох категорій об'єктів існує іконічний вигляд, але сучасні системи розпізнавання, на нашу думку, мають труднощі з розпізнаванням об'єктів у неканонічних позах або в безладних сценах.

Один із викликів полягає в пошуку природних зображень, що містять кілька об'єктів, оскільки ідентифікація багатьох об'єктів може бути досягнута лише за допомогою контексту. Для просування досліджень у контекстуальному мисленні необхідні зображення, що зображають сцени, а не лише об'єкти в ізоляції. Докладне просторове розуміння розташування об'єктів стане основною складовою аналізу сцени. Розташування об'єкта може бути визначено грубо за допомогою обмежувальної рамки або з використанням точного сегментації на рівні пікселів. Щоб оцінити продуктивність локалізації, важливо, щоб набір даних містив позначення для кожної інстанції кожної категорії об'єктів.

Набір даних Microsoft Common Objects in COntext (MS COCO) містить 91 загальну категорію об'єктів, з яких 82 мають більше 5,000 помічених екземплярів. Загалом набір даних має 2,500,000 помічених екземплярів у 328,000 зображень. У порівнянні з популярним набором даних ImageNet, COCO має менше категорій, але більше екземплярів на категорію, що сприяє навчанню детальних моделей об'єктів, здатних до точної 2D-локалізації. Крім того, важливим відмінністю нашого набору даних є кількість помічених

екземплярів на зображення, що може допомогти в навчанні контекстуальної інформації.

Вибір категорій об'єктів є складним завданням, яке потребує врахування кількох важливих аспектів. Категорії повинні формувати репрезентативний набір усіх можливих категорій, бути релевантними для практичних застосувань та зустрічатися з достатньою частотою, щоб забезпечити збір великого набору даних. Іншими важливими питаннями є включення як "об'єктних", так і "матеріальних" категорій, а також можливість додавання дрібногранних категорій та категорій частин об'єктів.

Категорії "об'єктів" містять елементи, які можна легко позначити, наприклад, людина, стілець або автомобіль, тоді як категорії "матеріалів" включають такі елементи, як небо, вулиця чи трава, які не мають чітко визначених меж. Оскільки основна мета полягає у точній локалізації інстанцій об'єктів, було вирішено включити лише "об'єктні" категорії і не враховувати "матеріальні". Проте, оскільки категорії "матеріалів" можуть надавати значну контекстуальну інформацію, вважається, що їх подальше позначення буде корисним.

Специфіка категорій об'єктів може значно варіюватися. Наприклад, собака може належати до категорій "ссавець", "собака" або "німецька вівчарка". Для забезпечення практичного збору значної кількості інстанцій на категорію було вирішено обмежити набір даних категоріями початкового рівня, тобто категоріями, які зазвичай використовуються людьми для опису об'єктів (собака, стілець, людина). Також можливо, що деякі категорії об'єктів можуть бути частинами інших категорій. Наприклад, обличчя може бути частиною людини. Очікується, що включення категорій частин об'єктів (обличчя, руки, колеса) буде корисним для багатьох реальних застосувань [1].

Як набір даних використовується COCO 2017. COCO 2017: Common Objects in Context 2017 — це набір даних для завдань сегментації, семантичної сегментації та виявлення об’єктів.

У COCO є два завдання виявлення об’єктів: використання вихідних даних обмежувальної рамки або вихідних даних сегментації об’єкта (останній також відомий як сегментація екземпляра).

Набір даних COCO 2017 є компонентом обширного набору даних Microsoft COCO. Це бачення реалізується через компіляцію зображень, що зображують складні повсякденні сцени, де природним чином існують звичайні об’єкти. У наборі даних Microsoft COCO містяться зображення, що відносяться до одного з 91 класу об’єктів. Цей повний набір даних включає в себе 2,5 мільйона позначених екземплярів, розподілених між 328 000 зображень. В свою чергу COCO 2017, містить 2099063 екземпляри із мітками в межах 163 957 зображень, що відносяться до 80 класів.

Набір даних стосується трьох основних дослідницьких проблем розпізнавання об’єктів: виявлення неканонічних перспектив об’єктів, контекстне обґрунтування між об’єктами та точна двовимірна локалізація об’єктів. Вибір категорій об’єктів є нетривіальною вправою. Категорії повинні формувати репрезентативний набір усіх категорій, відповідати практичним застосуванням і траплятися з достатньою частотою, щоб уможливити збір великого набору даних.

Екземпляри зображень у форматі .jpg з набору даних COCO 2017 наведено на рисунку 2.1.

Рис. 2.1. Приклади зображень з набору даних COCO 2017

2.1.2 Набір даних Military Aircraft Detection Dataset

Military Aircraft Detection Dataset [25] — це спеціалізований набір даних, розроблений для задач комп'ютерного зору, спрямованих на виявлення та класифікацію військових літаків. Цей набір даних використовується для навчання і тестування алгоритмів глибокого навчання, що працюють із зображеннями. У даному аналізі розглянуто його структуру, властивості, а також потенційні виклики під час використання.

Основні характеристики датасету:

- **Тип даних:** Датасет складається зі зображень у форматі .jpg і файлів анотацій у форматі .csv.
- **Зображення:** Висока якість візуальних даних забезпечує чітке відображення літаків. Зображення містять різноманітні сцени: літаки

на землі, в польоті, у контексті військових аеродромів або в полі дії інших об'єктів.

- **Анотації:** Містять інформацію про розташування об'єктів (координати межових рамок), їх класи тадодаткові метадані.
- **Кількість даних:** У наборі представлено сотні або тисячі зображень, що робить його придатним для тренування складних моделей, таких як YOLO або RT-DETR.
- **Класифікація об'єктів:** Усі об'єкти в наборі відносяться до однієї категорії (літаки), але можуть бути поділені на підкатегорії, що включають різні типи військових літаків. Це створює умови для розв’язання задачі багатокласової класифікації.
- **Різноманітність даних:** Датасет охоплює зображення з різних ракурсів, під різними умовами освітлення, а також у різних середовищах (літаки на аеродромі, у польоті, на полігоні тощо).

Набір орієнтований на вузьку категорію об’єктів — військові літаки, що дозволяє глибше досліджувати саме цю галузь і створювати високоточні моделі для подібних задач. Зображення відображають реальні сцени, що підвищує здатність моделей узагальнювати знання для застосування в реальних умовах. Датасет легко інтегрується з популярними бібліотеками для тренування моделей, такими як ultralytics, Detectron2 та TensorFlow Object Detection API.

Набір даних включає 30 тисяч зображень військових літаків, де деякі типи об'єднано в одну категорію разом з їх варіантами, розділених на 74 класи.

Структура набору даних

Набір даних організовано в три основні директорії: 'dataset', 'crop' і 'annotated_samples:

1. **Директорія dataset.** Містить зображення у форматі JPEG (.jpg) та відповідні файли анотацій у форматі CSV з однаковими іменами файлів. Кожен файл анотації CSV описує об'єкти на зображенні,

використовуючи формат PASCAL VOC (рис. 2.2). Колонки у файлах анотацій CSV такі:

- **filename:** Ідентифікатор для зображення та відповідного файлу анотацій.
- **width** і **height**: Розміри зображення в пікселях.
- **class**: Тип літака.
- **xmin**, **ymin**, **xmax**, **ymax**: Координати обмежувального прямокутника.

filename	width	height	class	xmin	ymin	xmax	ymax
000aa01b25574f28b654718db0700f72	2048	1365	F35	852	177	1998	503
000aa01b25574f28b654718db0700f72	2048	1365	JAS39	169	769	549	893
000aa01b25574f28b654718db0700f72	2048	1365	JAS39	125	908	440	1009
000aa01b25574f28b654718db0700f72	2048	1365	B52	277	901	1288	1177

Рис. 2.2. Приклад анотацій

2. **Директорія crop.** Містить зображення, вирізані з оригінальних зображень у папці dataset, відповідно до анотацій обмежувальних прямокутників. Ця папка спеціально підготовлена для полегшення навчання моделей класифікації. Вирізані зображення організовані в папки, назви яких відповідають іменам їх класів.
3. **Директорія annotated_samples.** Містить вибірку зразків зображень з візуалізованими обмежувальними прямокутниками, щоб забезпечити швидке візуальне посилання та приклади анотованих зображень з набору даних.

Екземпляр зображень у форматі .jpg з набору даних Military Aircraft Detection Dataset наведено на рисунку 2.3.

Рис. 2.3. Приклад зображень з набору даних Military Aircraft Detection Dataset

2.2 Інструменти і засоби розробки

Під час виконання роботи було використано декілька бібліотек Python, кожна з яких мала чітко визначене призначення, спрямоване на обробку, аналіз і візуалізацію даних, а також автоматизацію процесів підготовки набору даних для тренування моделей виявлення об’єктів (рис. 2.4):

1. **numpy**

   Бібліотека слугувала основним інструментом для роботи з багатовимірними масивами числових даних. Вона забезпечує швидкі й ефективні математичні операції, що використовувалися для обчислення статистичних характеристик, перетворення координат, і маніпуляцій із зображеннями у числовому форматі.

2. **pandas**

Було використано для роботи з табличними даними, такими як файли формату .csv, що містять анотації об’єктів. Бібліотека дозволяє зручно зчитувати, змінювати і фільтрувати дані, що було критично важливо під час підготовки набору даних і перевірки консистентності анотацій.

3. **matplotlib.pyplot**

   Ця бібліотека застосовувалася для візуалізації результатів обробки даних. Її основне призначення полягало у створенні графіків, гістограм і відображенні зображень. Це сприяло аналізу розподілу даних і виявленню аномалій у наборі.

4. **glob**

   Забезпечувала пошук файлів у файловій системі за певними шаблонами. Її використання дозволило автоматизувати збір даних (зображень та файлів анотацій) з заданої директорії.

5. **shutil**

   Використовувалася для управління файлами і директоріями, зокрема, копіювання і переміщення файлів між різними папками. Це стало ключовим етапом під час організації структури набору даних для тренування, валідації та тестування.

6. **cv2**

   (OpenCV) Бібліотека призначена для роботи із зображеннями. Її використання охоплювало читання, зміну розміру, перетворення кольорів та візуалізацію об'єктів. Вона також дозволила додатково обробляти зображення, якщо це було потрібно для покращення якості даних.

7. **os**

   Основний інструмент для роботи з файловою системою, який забезпечував створення директорій, перевірку їх існування та інші дії, пов’язані з організацією структури проєкту.

8. **tqdm**

   Використовувалася для відображення прогрес-барів під час виконання ітеративних операцій, таких як копіювання файлів або обробка зображень. Це дозволило зручно моніторити перебіг виконання завдань.

9. **seaborn**

   Ця бібліотека сприяла створенню більш якісних і зрозумілих візуалізацій, зокрема теплових карт або комплексних діаграм. Вона використовувалася для аналізу розподілу об'єктів у наборі даних.

10. **ultralytics**

    Ця бібліотека призначена для роботи з сучасними моделями комп'ютерного зору, зокрема YOLOv8 та RT-DETR. Вона забезпечує просте тренування, оцінку та розгортання моделей для задач виявлення об'єктів, сегментації та класифікації. Основне призначення бібліотеки полягає у забезпеченні швидкої інтеграції глибокого навчання у реальні проєкти, оптимізації продуктивності під час роботи з великими наборами даних і підтримці сучасних форматів (COCO, YOLO).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob
import shutil
import cv2
import os
from tqdm import tqdm
import seaborn as sns
import ultralytics
```

Рис. 2.4. Лістинг коду ініціалізації використаних бібліотек

2.3 Організація потоку робочих операцій

2.3.1 Платформа для тренування та тестування моделей

Тренування та тестування всіх моделей було здійснено за допомогою платформи Kaggle.

Kaggle — це одна з найбільших платформ для роботи з даними та машинного навчання. Вона надає фахівцям із даних, дослідникам і розробникам інструменти для аналізу, побудови моделей і навчання на реальних задачах.

Платформа активно використовується в освітніх цілях, для професійного розвитку та створення інноваційних рішень у сфері штучного інтелекту та аналізу даних.

Для виконання поставлених задач було обрану дану платформу через її наступні особливості:

Набори даних (Datasets): Платформа містить велику кількість публічних і приватних наборів даних, які можна використовувати для досліджень, аналізу та навчання моделей. Користувачі також можуть ділитися власними наборами даних.

Kaggle Kernels (Notebooks): Інтегроване середовище для розробки, яке дозволяє писати, виконувати та обмінюватися кодом у Python або R без необхідності налаштування локального середовища. Платформа забезпечує доступ до обчислювальних ресурсів, таких як GPU та TPU.

Обговорення (Discussion): Форум для обміну ідеями, пошуку рішень та співпраці з іншими учасниками спільноти.

При виконанні завдань проекту на платформі було створено два різних середовища для розробки та ініціалізації коду. Платформа Kaggle забезпечила доступ до бібліотек, що було наведено в пункті 2.2.

Для процесу тренування та тестування в якості прискорювача було обрано два графічних процесори NVIDIA Tesla T4, доступ до яких забезпечила платформа Kaggle.

NVIDIA Tesla T4 — це потужний графічний процесор (GPU), створений для прискорення задач машинного навчання, глибокого навчання, аналітики даних і віртуалізації. Він базується на архітектурі Turing і широко використовується в серверних середовищах та хмарних платформах.

Tesla T4 — це універсальний GPU, оптимізований для обчислювальних задач у хмарних середовищах і центрах обробки даних, що забезпечує баланс між продуктивністю, енергоефективністю та вартістю.

Основні характеристики:

Архітектура Turing: GPU підтримує тензорні ядра (Tensor Cores) третього покоління, які забезпечують високопродуктивні обчислення для задач глибокого навчання, зокрема роботи з матрицями.

Продуктивність: Пікова продуктивність у задачах FP32: 8.1 TFLOPS Для задач із змішаною точністю (FP16/INT8): 65 TFLOPS (FP16), 130 TOPS (INT8) Підтримка INT4, що забезпечує ще більшу продуктивність у певних задачах.

Об'єм пам'яті: 16 ГБ GDDR6

Пропускна здатність пам'яті: 320 ГБ/с

Масштабованість: Розроблений для роботи в центрах обробки даних з можливістю встановлення в компактних серверах (форм-фактор PCIe). Підтримка NVENC/NVDEC для прискорення відеокодеків, що корисно для потокового відео та мультимедійних застосунків.

Підтримка програмного забезпечення: Підтримка NVIDIA CUDA, cuDNN, TensorRT, що дозволяє використовувати GPU у широкому спектрі програм для машинного навчання. Інтеграція з популярними фреймворками, такими як TensorFlow, PyTorch, та ONNX Runtime.

2.3.2 Навчені моделі

За базову модель при розробці покращеного методу виявлення об'єктів реального часу на основі моделей трансформерів було обрано модель RT-

DETR Large. Ця модель є представником ефективних об'єкт-детекторів, розроблених для задач реального часу.

Основні характеристики RT-DETR Large:

Архітектура: RT-DETR Large побудована на трансформерній архітектурі з гібридним енкодером, що поєднує переваги згорткових нейронних мереж (CNN) та механізму самопильності (self-attention). Цей підхід забезпечує глибоке врахування просторових і масштабних залежностей вхідних даних.

Гібридний енкодер:

- Attention-Inspired Feature Interaction (AIFI): Енкодер взаємодіє з багаторівневими ознаками, що покращує виявлення об'єктів різних розмірів.
- Cross-Scale Feature Fusion (CSFF): Застосовується для інтеграції ознак з різних масштабів, що підвищує точність моделі на зображеннях зі складними сценами.

Декодер: Трансформерний декодер використовує динамічні запити для передбачення положень і класів об'єктів. Забезпечує мінімізацію невизначеності при виборі запитів, що підвищує ефективність виявлення.

Продуктивність: Підтримує обробку зображень у реальному часі з високою швидкістю (FPS). Досягає балансу між швидкістю і точністю, демонструючи конкурентоспроможність із моделями, такими як YOLOv8 та DINO. Модель оптимізована для зменшення затримок під час інференсу на сучасних GPU.

Застосування: Модель RT-DETR Large підходить для задач комп’ютерного зору в реальному часі, таких як: Виявлення об’єктів на відеопотоках, системи спостереження, автоматизація аналізу зображень у промисловості, транспорті та безпеці. Обчислювальні вимоги: RT-DETR Large є масштабованою і може працювати на серверних GPU, таких як NVIDIA T4,

A100 тощо. Забезпечує продуктивність навіть на обмежених апаратних ресурсах, що робить її універсальним вибором для різних середовищ.

Точність: Використання трансформерів дозволяє підвищити точність детекції за рахунок більш точного передбачення положень і класів об'єктів. Забезпечує хорошу продуктивність навіть на складних наборах даних із високою варіативністю об'єктів.

Таким чином, RT-DETR Large була обрана як базова модель завдяки її високій швидкості, точності та гнучкості, що робить її ідеальною для побудови покращених методів виявлення об'єктів у реальному часі.

Також додатково було розглянуто модель RT-DETR Extra Large, що має такі відмінності від моделі RT-DETR Large: збільшену кількість параметрів, вищу продуктивність у складних задачах детекції завдяки покращеному енкодеру та декодеру, а також вищі вимоги до обчислювальних ресурсів. Ця модель забезпечує ще точніше виявлення об'єктів, особливо для великих і складних наборів даних, але зі зменшеною швидкістю інференсу.

Моделі RT-DETR Large та RT-DETR Extra Large були натреновані на широко використовуваних наборах даних для виявлення об’єктів, зокрема COCO (Common Objects in Context), що був розглянутий в пункті 2.1.1.

Висновки

В даному розділі було розглянуто інструментальні засоби, набори даних та програмне забезпечення, що було використано для розробки покращеного методу виявлення об’єктів реального часу на основі моделей трансформерів, який описано в наступному розділі.

В даному розділі було розглянуто інструментальні засоби, набори даних та програмне забезпечення, що було використано для розробки покращеного методу виявлення об’єктів реального часу на основі моделей трансформерів, який описано в наступному розділі. До цього списку належать набір даних COCO 2017 для завдань виявлення об'єктів, а також спеціалізований набір даних для виявлення та класифікації військових літаків, що містить високоякісні зображення з відповідними анотаціями. Для аналізу та підготовки даних активно використовувалися бібліотеки Python, такі як numpy, pandas та matplotlib, які забезпечують можливість ефективно обробляти та візуалізувати великі обсяги даних.

Для виконання експериментів і розгортання моделей було обрано платформу Kaggle, що дозволяє ефективно використовувати потужності хмарних обчислень для виконання ресурсомістких завдань. Особлива увага приділялася використанню моделі RT-DETR, яка є вдосконаленим підходом для виявлення об'єктів у реальному часі. Ця модель, завдяки своєму ефективному гібридному енкодеру та трансформерному декодеру з додатковими предсказувальними головами, демонструє значні переваги в точності та швидкості виявлення порівняно з іншими методами.

РОЗДІЛ 3
РОЗРОБКА ПОКРАЩЕННОГО МЕТОДУ ВИЯВЛЕННЯ ОБ'ЄКТІВ РЕАЛЬНОГО ЧАСУ

3.1 Тестування попередньо навчених моделей

Тестування готових моделей RT-DETR (Real Time Detection Transformer), а саме: RT-DETR-Large та RT-DETR-Extra-Large, було проведено на онлайн-платформі Kaggle з використанням вбудованного прискорювача GPU T4 [25]. Ваги навчених моделей (rtdetr-l.pt та rtdetr-x.pt) було отримано за допомогою python-інтерпретатору *ultralytics*.

Для тестування було використано піднабір val2017 набору COCO 2017, що містить 5000 зображень, , що містять об'єкти, які відносяться до 80 різних класів.

Було виконано тестування готових моделей з метою отримання значень якості AP та швидкості FPS моделей. Отримані результати зображено на рисунках 3.1, 3.2. Порівняння метрик для обох моделей, отриманих внаслідок тестування, з метриками, отриманими розробниками моделей [2], наведено в таблиці 3.1.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.524
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.706
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.568
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.336
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.570
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.705
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.383
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.636
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.696
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.506
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.736
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.873
```

Рис. 3.1. Метрики моделі RT-DETR-Large

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.540
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.720
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.587
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.346
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.590
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.717
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.391
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.648
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.709
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.505
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.759
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.872
```

Рис. 3.2. Метрики моделі RT-DETR-Extra-Large

Таблиця 1

Порівняння отриманих метрик з авторами статі.

	Model	Input shape	Params (M)	GFLOPs	FPS	AP_{val}	AP_{val50}	Ap_{val75}	AP_{valS}	AP_{valM}	AP_{valL}
Метрики, отримані командою розробників	RT-DETR-L	640	32	110	108	53.1	71.3	57.7	34.8	58.0	70.0
	RT-DETR-X	640	67	234	74	54.3	72.7	58.6	36.0	58.8	72.1
Власні отриманні метрики	RT-DETR-L	640	32,97	108,3	19	52.4	70.6	56.8	33.6	57.0	70.5
	RT-DETR-X	640	67,47	232,7	16	54.0	72.0	58.7	34.6	59.0	71.7

Порівнявши і проаналізувавши отримані результати, видно, що метрики якості (Average Precision для різних IoU та розмірів зон) моделей, отриманих в результаті тестувань та наведених розробником, майже повністю збігаються (найбільша різниця складає 1,4%).

При цьому, незважаючи на ідентичність прискорювачів (GPU T4), натренованих моделей та набору даних в обох випадках, показники швидкості суттєво відрізняються. Це обумовлено тим, що при виконанні тестування командою розробників було використано високоефективний інструмент оптимізації моделей TensorRT для апаратних прискорювачів NVIDIA. Цей інструмент більше підходить для виконання інференції (inference), в той час,

коли Kaggle використовує менш оптимізовані для подібних завдань TensorFlow та PyTorch. В свою чергу TensorFlow та PyTorch ефективніше застосовуються при навчанні моделей. Також, TensorRT краще оптимізовано для низьких значень batch size (в нашому випадку batch size = 1).

Візуалізація передбачень моделі RT-DETR наведено на рисунку 3.3.

Рис. 3.3. Візуалізація RT-DETR передбачень

Детальніше розглядаючи метрики якості для кожного з наявних класів об’єктів, можна простежити, що для певних класів показники точності є значно меншими, ніж для інших. Наприклад, проаналізувавши показники точності для окремих класів, що містить набір даних COCO 2017, видно, що середня точність (IoU 50-95) для таких класів як «човен» або «книга» є більше ніж в 2 рази меншою за середню точність для класу «потяг», «автомобіль» та «людина» (рис. 3.4). Узагальнивши, можна простежити, що до класів, що

мають нижчі показники точності, відносяться об’єкти малого розміру та об’єкти, що частково перекриваються на зображені іншими об’єктами. Причиною цього може бути обмежена кількість піксельної інформації як при навчанні моделі, так і під час тестування на нових зображеннях, або велика різниця у масштабах між малими і великими об’єктами, що призводить до більшому приділенню «уваги» трансформера більш контрастним об’єктам та об’єктам більшого розміру.

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	5000	36335	0.743	0.654	0.702	0.521
person	2693	10777	0.848	0.746	0.845	0.607
bicycle	149	314	0.775	0.573	0.646	0.396
car	535	1918	0.743	0.664	0.735	0.498
motorcycle	159	367	0.814	0.714	0.803	0.56
airplane	97	143	0.9	0.885	0.935	0.775
bus	189	283	0.878	0.813	0.855	0.738
train	157	190	0.901	0.913	0.934	0.765
truck	250	414	0.662	0.548	0.643	0.458
boat	121	424	0.705	0.55	0.606	0.36

Рисунок 3.4. Показники середньої точності для декількох класів

Низькі показники для окремих класів в свою чергу впливають на середні показники точності самої моделі. Для покращення результатів можна розглядати такі напрямки:

1. **Аугментація вхідних даних:** використання даних із штучно збільшеними дрібними об’єктами або застосування зображень вищої роздільної здатності та підвищення фокусу на деталях у мікрообластях.
2. **Фокусоване навчання:** Використання динамічних масок, які акцентують увагу на дрібних об'єктах у процесі тренування. Застосування більше "гострих" масок для точнішого представлення малих об’єктів.

3. **Комбінування:** Використання моделей, спеціалізованих на виявленні дрібних об’єктів. Наприклад, комбінування з ResNet або Swin Transformer для багаторівневої деталізації.
4. **Вузька спеціалізація:** використовувати меншу кількість типів об’єктів в наборі даних, що використовуються для навчання моделі.

3.2 Підготовка даних до тренування

При розробці покращеного методу виявлення об’єктів реального часу на основі моделей трансформерів було здійснено тренування та тестування готової моделі RT-DETR-Large на онлайн-платформі Kaggle з використанням вбудованного прискорювача GPU T4 [26]. Ваги навченої моделі (rtdetr-l.pt) було отримано за допомогою python-інтерпретатору *ultralytics*.

Для тренування та тестування було використано набір Military Aircraft Detection Dataset, який було розглянуто в пункті 2.1.2.

Для підготовки даних для тренування, валідації та тестування моделі було виконано серію операцій, спрямованих на структурування зображень та відповідних анотацій. Ці дії були необхідні для забезпечення правильного функціонування алгоритму, а також для створення чіткої структури набору даних, придатного для моделювання.

Перший етап полягав у зборі та впорядкуванні шляхів до вхідних файлів. Для забезпечення доступу до даних з директорії було здійснено пошук файлів, що відповідають вказаним критеріям. Шаблони пошуку включали два типи даних: файли формату .csv, які можуть містити метаінформацію або координати об’єктів на зображеннях, та файли формату .jpg, які безпосередньо є візуальними даними.

Щоб забезпечити коректний порядок обробки файлів, усі отримані шляхи було відсортовано. Це дозволяє синхронізувати зображення та їх анотації під час наступних етапів. Для контролю за обсягом доступних даних

було виведено на екран кількість знайдених файлів .csv, які в даному випадку використовувалися як індикатор обсягу набору даних.

Другий етап стосувався створення необхідної файлової структури для підготовки набору даних. Для забезпечення організованого зберігання даних у проекті було створено декілька окремих директорій, призначених для зображень та анотацій, розподілених за категоріями: тренувальні, валідаційні та тестові дані.

Було розроблено наступну структуру (рис. 3.5):

1. Директорії для зображень тренувальної частини набору даних.
2. Директорії для анотацій, що відповідають тренувальним зображенням.
3. Директорії для зображень та анотацій, призначених для валідаційної та тестової частин набору.

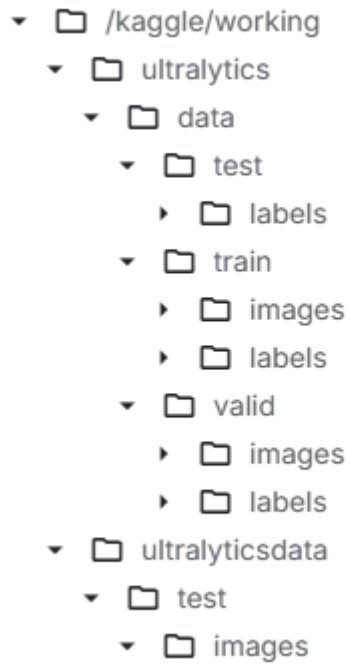

Рис. 3.5. Структура готового для навчання набору даних

Під час створення кожної директорії було враховано можливість їх попереднього існування, тому вказано параметр, що дозволяє уникнути помилок у разі дублювання операцій. Результат забезпечує підготовлений

простір для подальшої роботи з набором даних. Вхідні файли відсортовані, а структура директорій організована таким чином, щоб легко інтегрувати наступні етапи обробки: копіювання файлів, конвертація форматів та їх використання для тренування моделі.

Було виконано підготовку зображень та їхніх анотацій для подальшого використання в процесі тренування, перевірки та тестування моделі. Спочатку були визначені зображення і відповідні файли анотацій у форматі CSV, які містять дані про координати межових рамок об'єктів на зображеннях. Кожен файл CSV відповідає одному зображенню, а анотації описують місце розташування об'єктів і їх класи.

Було виконано сортування зображень за їх назвами, що дозволило визначити, до якого набору даних вони належать: тренувальний, перевірочний або тестовий. Зображення, назви яких починалися з цифр або літер від 'a' до 'b', були перенесені до відповідної директорії для тренування. Для кожного з цих зображень також було створено текстові файли, що містять анотації про об'єкти на зображеннях, у форматі, сумісному з моделями виявлення об'єктів.

Для набору перевірки було відібрано зображення, назви яких починалися з літер 'c' до 'f'. Ці зображення були перенесені до відповідної директорії для перевірки. Для того щоб збільшити варіативність, було змінено порядок символів у назвах файлів перед їх копіюванням.

Зображення, що не підпадали під попередні категорії, було використано для тестового набору даних. Назви таких зображень також були змінені, і їх було копійовано до відповідної директорії для тестування. Для кожного з цих зображень також були створені текстові файли з анотаціями.

Для кожного зображення було створено текстові файли, що містять анотації. В цих файлах записувалася інформація про об'єкти: клас об'єкта, координати його центральної точки, а також ширина і висота межових рамок, нормовані по відношенню до розмірів зображення. У разі наявності помилок в анотаціях, такі файли і зображення не були включені в оброблений набір.

Після завершення процесу підготовки даних було підраховано кількість файлів анотацій у кожній з категорій (тестова, перевірочна, тренувальна), щоб перевірити правильність виконаної роботи.

3.4 Візуалізація набору даних

Розподіл об’єктів за класами. Було обчислено і візуалізовано нормалізований розподіл об'єктів за класами у кожному наборі даних (рис. 3.6). Це дозволило оцінити баланс даних між наборами, використовуючи горизонтальну стовпчикову діаграму, де кожен клас відображався окремо. Кольори для кожного набору були обрані, щоб забезпечити чітку відмінність.

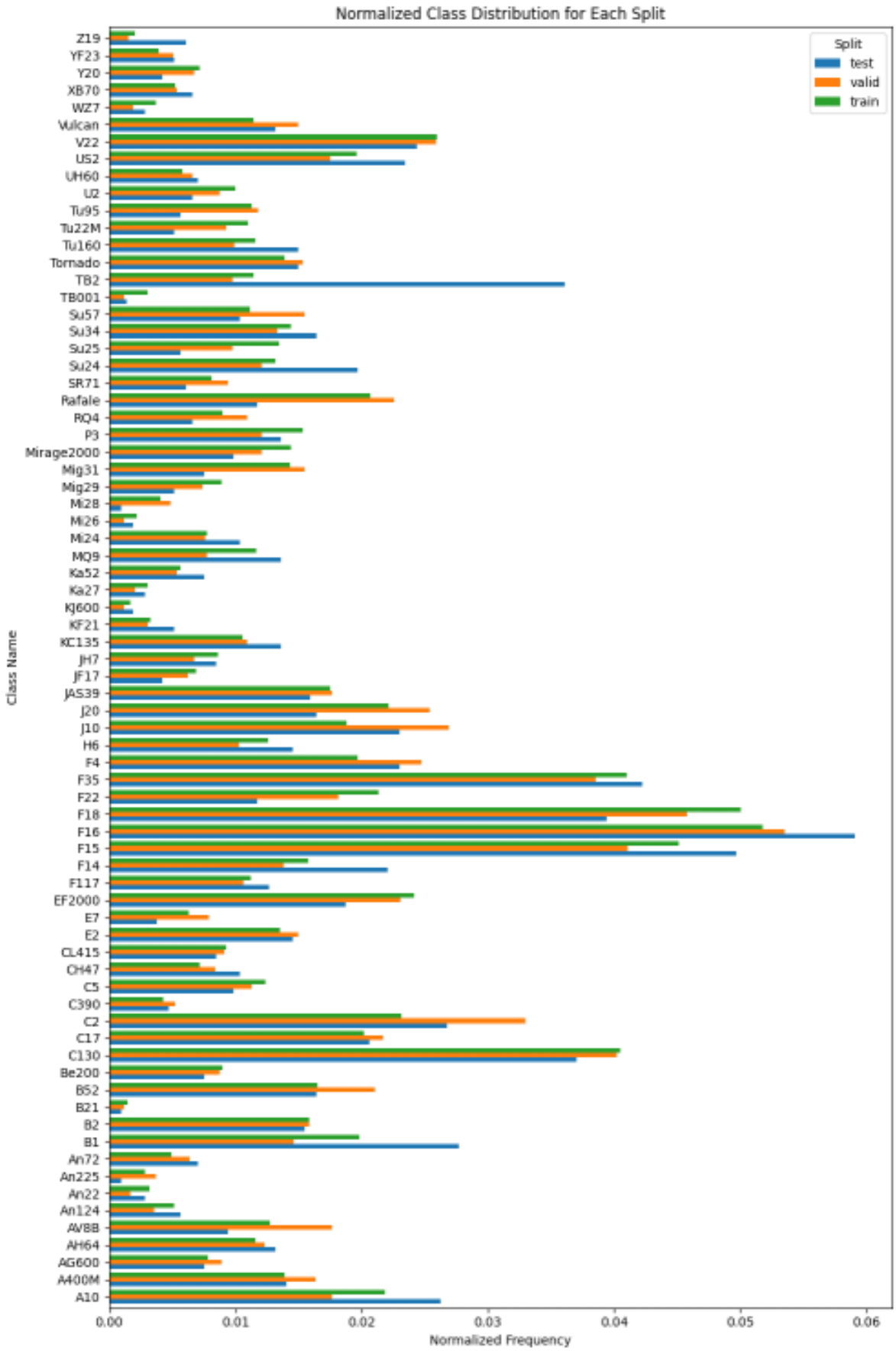

Рис. 3.6. Нормалізований розподіл об'єктів за класами у кожному наборі даних

Параметри межових рамок. Далі було проаналізовано параметри об'єктів у межах анотацій, зокрема ширину та висоту межових рамок. Для кожного об'єкта обчислювалася нормалізована ширина і висота шляхом ділення на розміри відповідного зображення. Було створено діаграму розсіювання, що відображала співвідношення цих двох параметрів для всіх об'єктів, з додаванням середніх значень ширини та висоти, які позначалися червоним "X" (рис. 3.7).

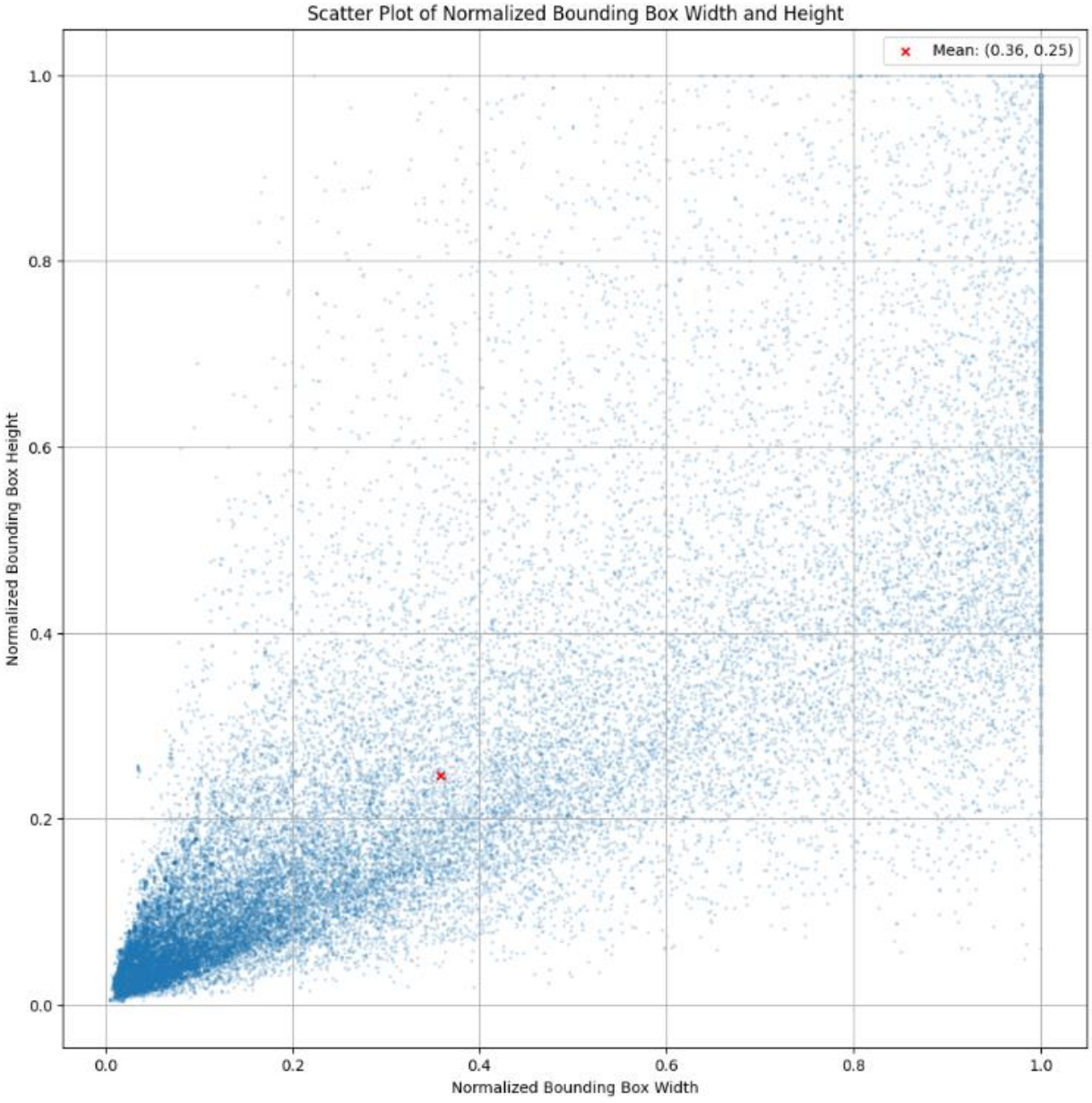

Рис. 3.7. Нормалізовані ширина та висота межових рамок об’єктів з наборів даних

Відношення ширини до висоти об’єктів. Було обчислено аспектне співвідношення межових рамок, яке відображає відношення ширини до висоти кожного об’єкта. Для цього параметра було побудовано гістограму в логарифмічному масштабі для кращої інтерпретації діапазону значень і частот їхнього виникнення (рис. 3.8). Це дозволило оцінити типові співвідношення розмірів для об’єктів у наборі даних.

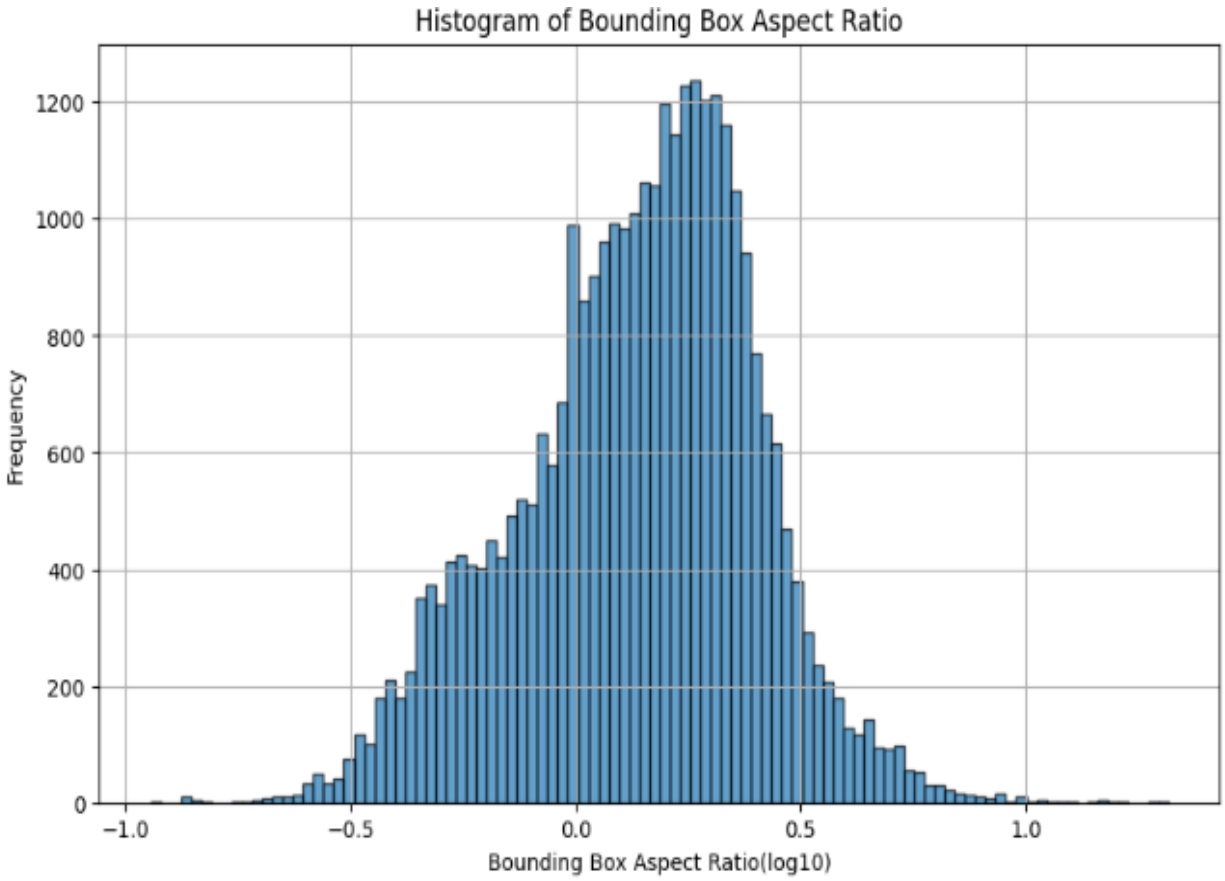

Рис. 3.8. Відношення ширини до висоти об’єктів з набору даних і частота їх зустрічі

Відношення площі межових рамок до площі зображення. Додатково було обчислено площу межових рамок об'єктів відносно площі зображення. Отримане співвідношення площі відображало, яку частину простору зображення займає об'єкт. Для цієї метрики також було побудовано гістограму,

яка дозволила оцінити, наскільки великі чи малі об'єкти присутні в наборі даних (рис. 3.9).

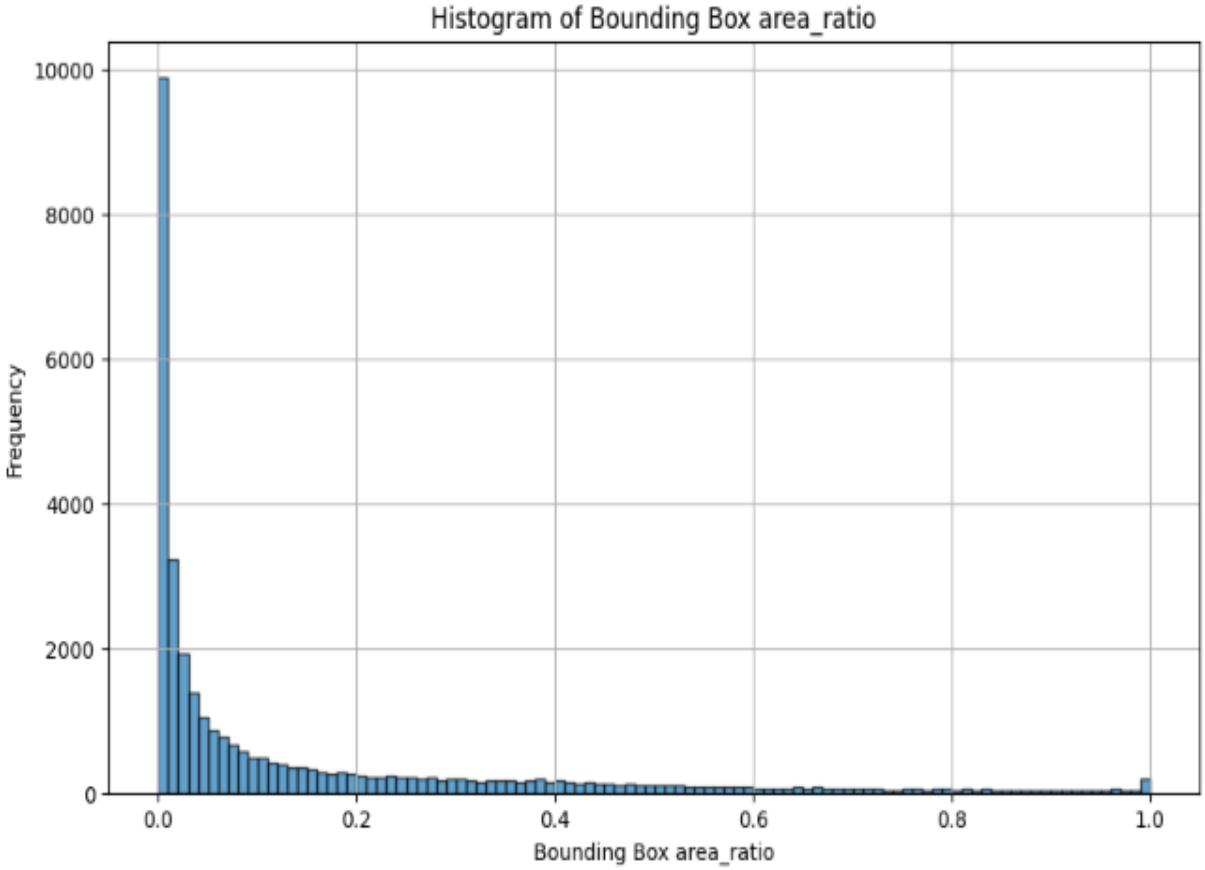

Рис. 3.9. Відношення розміру межових рамок до загальної площі зображень з набору даних і частота їх зустрічі

Аналіз аспектного співвідношення зображень. Для кожного зображення було обчислено аспектне співвідношення шляхом ділення ширини на висоту. Це дозволило оцінити розподіл пропорцій зображень, який було представлено у вигляді гістограми. Значення було подано в логарифмічному масштабі для полегшення аналізу широкого діапазону пропорцій (рис. 3.10).

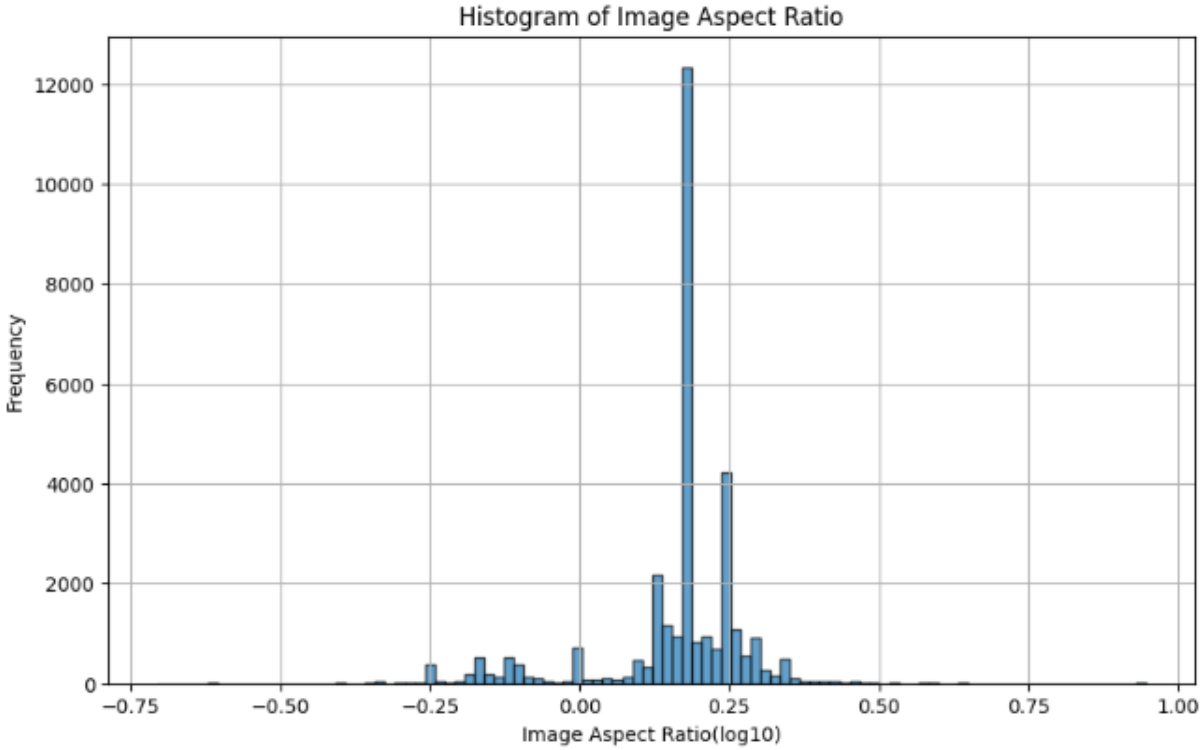

Рис. 3.10. Аспектне співвідношення зображень

Розподіл кількості межових рамок за типами літаків. Було підраховано загальну кількість межових рамок для кожного класу об’єктів. Результати цього аналізу візуалізовано у вигляді стовпчикової діаграми, що демонструє частоту представлення кожного типу літаків (рис. 3.11). Цей розподіл дозволив оцінити, які класи є домінантними, а які представлені недостатньо.

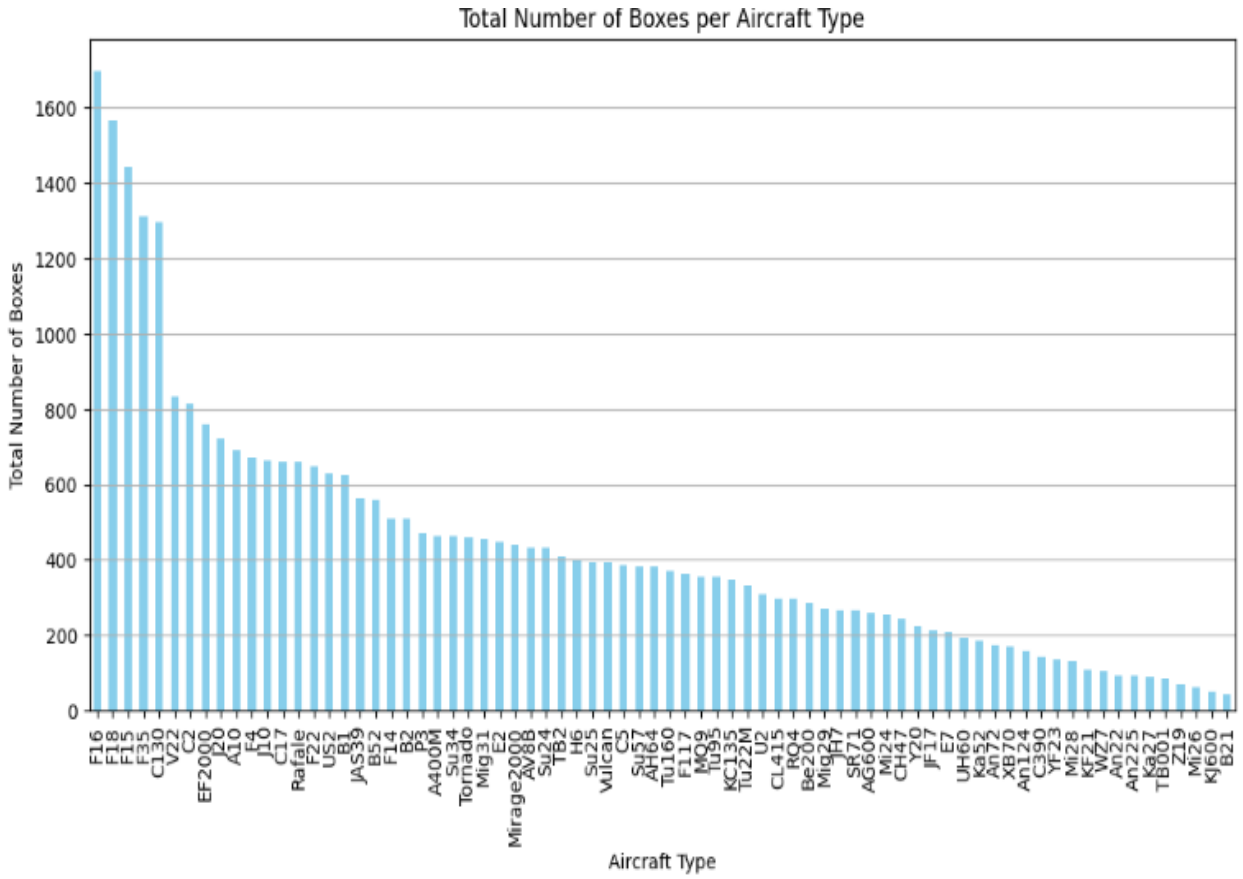

Рис. 3.11. Розподіл кількості межових рамок за типами літаків

Середня кількість межових рамок. Було обчислено середню кількість межових рамок на зображення без урахування класу, що дало загальну картину наповненості набору даних. Додатково було визначено середню кількість межових рамок для кожного класу, обчисливши співвідношення між кількістю межових рамок і кількістю зображень, де представлено кожен клас. На стовпчиковій діаграмі червоною лінією було позначено загальне середнє значення (рис. 3.12).

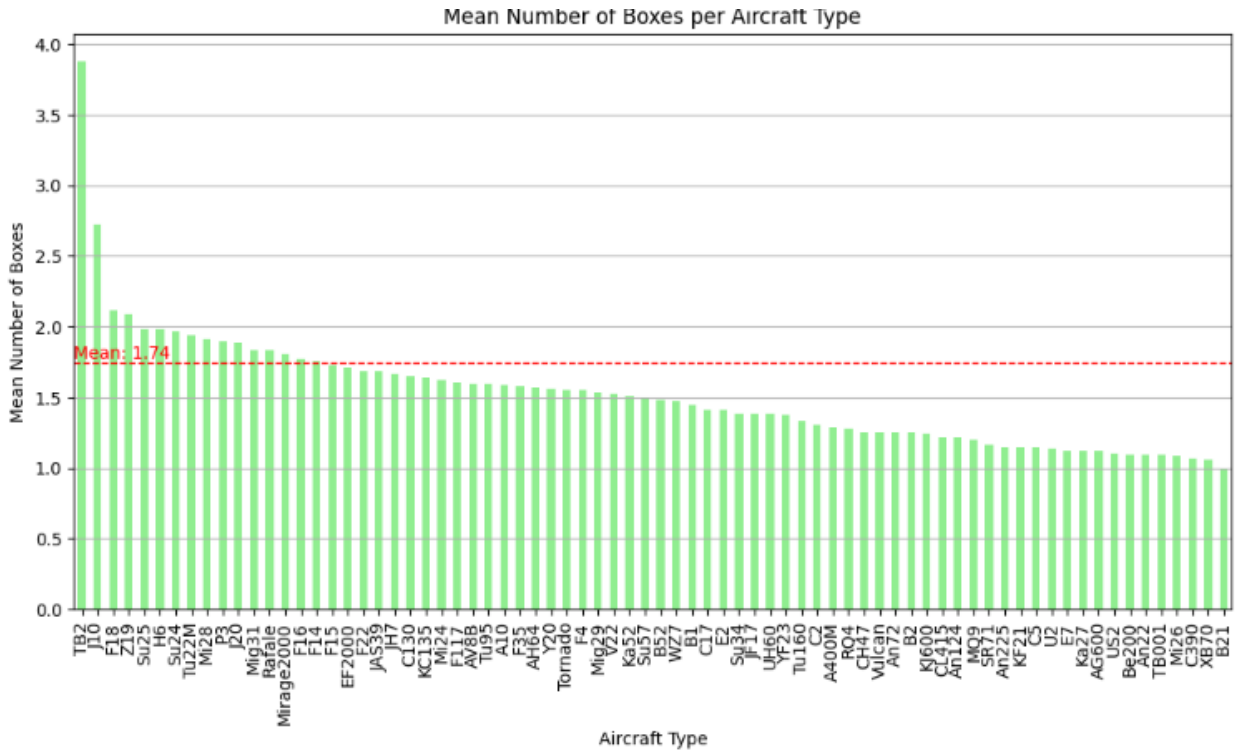

Рис. 3.12. Середня кількість межових рамок

Співіснування класів у межах одного зображення. Для кожного зображення було визначено унікальний набір класів об'єктів, що дозволило аналізувати співіснування різних типів літаків. Створено матрицю, що відображає кількість пар класів, які зустрічаються разом на одному зображенні (рис. 3.13). На тепловій карті було представлено співвідношення у логарифмічному масштабі, а діагональні значення (пари одного класу) були виключені для покращення сприйняття.

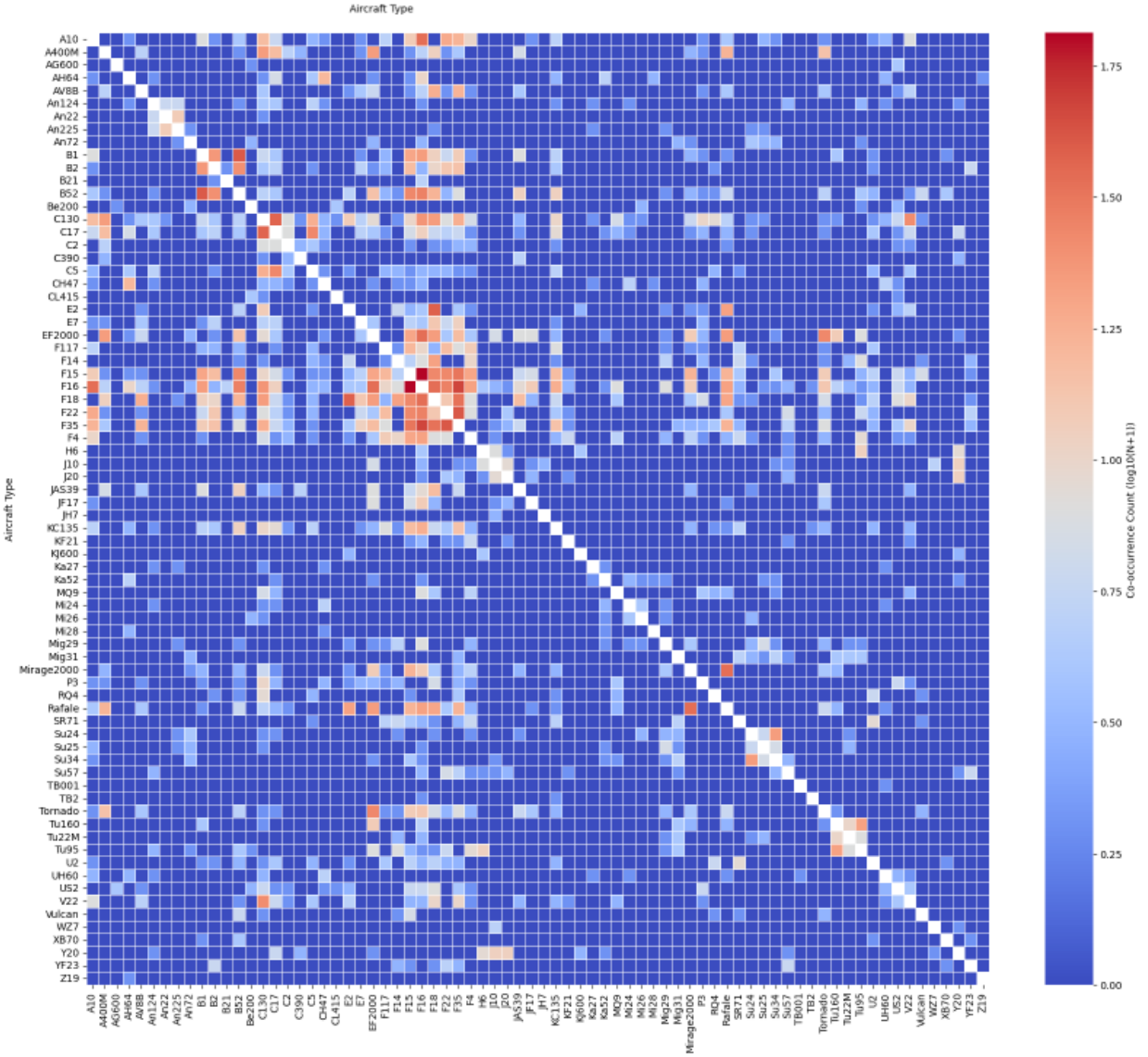

Рис. 3.13. Матриця, що відображає кількість пар літаків, що зустрічаються разом на одному зображенні

Розподіл аспектного співвідношення межових рамок. Було побудовано гістограму аспектного співвідношення межових рамок (відношення ширини до висоти) в логарифмічному масштабі (рис. 3.14). Цей аналіз виявив закономірності у формах межових рамок, які можуть вплинути на точність виявлення об'єктів моделлю.

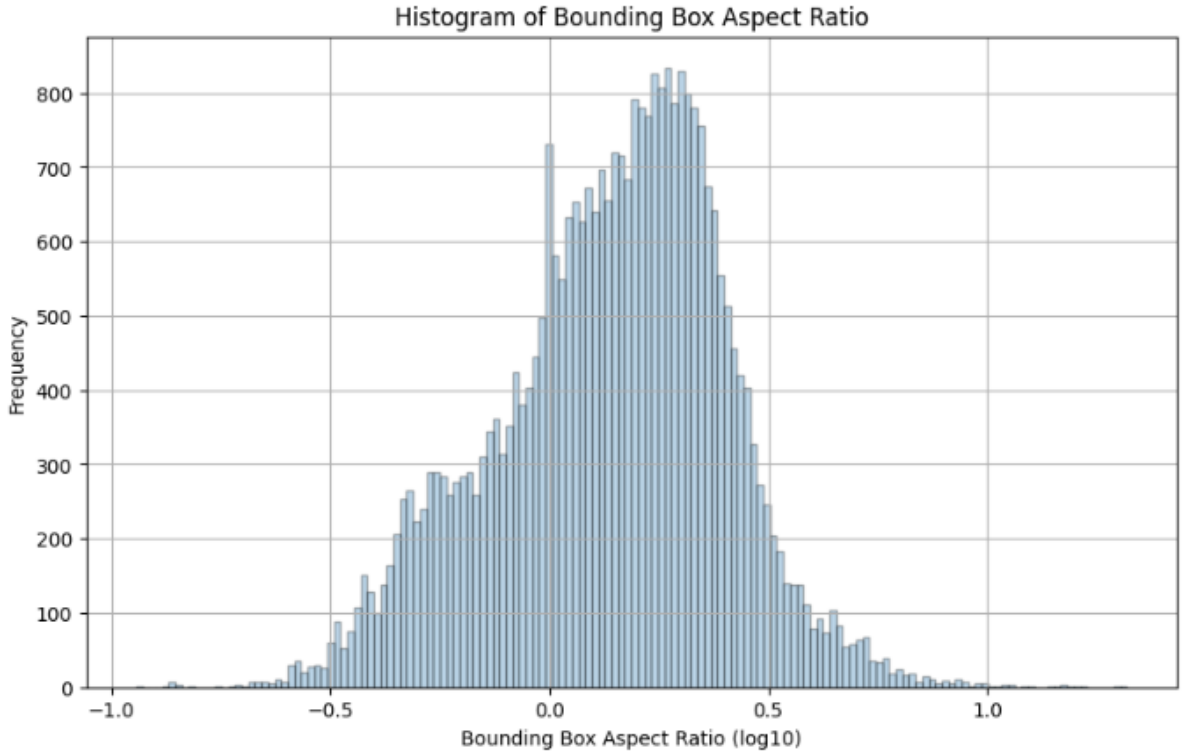

Рис. 3.14. Аспектне співвідношення межових рамок

3.4 Налаштування гіперпараметрів та аугментації даних перед початком навчання моделі

При тренуванні моделі RT-DETR використовується низка гіперпараметрів та інших елементів, які впливають на процес навчання (рис. 3.15).

```python
if MODE=='train':
    from ultralytics import RTDETR
    model = RTDETR(model='rtdetr-l.pt')

    result=model.train(
        name='RT-DETR_01',
        epochs=50,
        imgsz=640,
        batch=16,
        #optimizer='auto',

        # augmentation
        close_mosaic=0,
        hsv_h=0.02,
        hsv_s=0.75,
        hsv_v=0.45,
        degrees=0.1,
        translate=0.25,
        scale=0.55,
        shear=0.1,
        perspective=0.0,
        flipud=0.1,
        fliplr=0.5,
        mosaic=1.0,
        mixup=0.5,

        pretrained='/kaggle/working/rtdetr-l.pt',
        data='/kaggle/working/ultralytics/data/mad.yaml',
    )
```

Рис. 3.15. Лістинг коду для ініціалізації процесу навчання моделі

Основні параметри:

name='RT-DETR_01' Назва експерименту для тренування. Цей параметр використовується для ідентифікації поточного тренування та результатів, що зберігатимуться в окремій директорії.

epochs=50 Визначає кількість епох навчання. Епоха — це повний прохід моделі через весь тренувальний набір даних. Чим більше епох, тим краще модель може адаптуватися до даних, але це може призвести до перенавчання.

imgsz=640 Розмір зображень, які будуть використовуватись для тренування. Усі зображення будуть масштабовані до розміру 640×640 пікселів.

batch=16 Розмір пакета (batch size), що визначає кількість зображень, які будуть оброблятися одночасно в одній ітерації. Менші пакети вимагають менше пам'яті, але можуть бути менш стабільними для навчання.

Аугментація (augmentation):

Аугментація використовується для штучного збільшення кількості даних шляхом застосування випадкових змін до зображень і координат боксів. Це сприяє покращенню узагальнювальної здатності моделі.

close_mosaic=0 Застосування мозаїки (mosaic) для збільшення варіацій у зображеннях. Значення 0 вказує на відсутність закриття мозаїчних змін.

hsv_h=0.02 Випадкові зміни кольорового відтінку (hue) в межах 2%. Це робить модель стійкішою до змін освітлення.

hsv_s=0.75 Зміни насиченості (saturation) кольорів. Значення 0.75 означає, що насиченість може варіюватись у межах 75% від початкового.

hsv_v=0.45 Зміни яскравості (value) зображення в межах 45%.

degrees=0.1 Випадковий поворот зображення на кут до 0.1 градуса. Допомагає моделі розпізнавати об’єкти незалежно від їхнього кута орієнтації.

translate=0.25 Випадкове зміщення зображення по горизонталі та вертикалі до 25% від розмірів.

scale=0.55 Масштабування зображення в межах ±55% від початкового розміру.

shear=0.1 Викривлення (shearing) зображення з максимальним кутом до 0.1 радіана.

perspective=0.0 Випадкове спотворення перспективи. Значення 0.0 означає, що перспектива не змінюється.

flipud=0.1 Імовірність перевертання зображення по вертикалі. Значення 0.1 означає 10% ймовірності.

fliplr=0.5 Імовірність перевертання зображення по горизонталі. Значення 0.5 означає 50% ймовірності.

mosaic=1.0 Імовірність використання техніки мозаїки. Значення 1.0 означає, що мозаїка буде застосовуватись до всіх зображень.

mixup=0.5 Імовірність використання mixup — техніки комбінування двох зображень і відповідних анотацій. Значення 0.5 означає, що mixup буде застосовуватись до половини зображень.

Параметри моделі:

model='rtdetr-l.pt' Назва попередньо натренованої моделі RT-DETR. Використовується для початкової ініціалізації ваг.

pretrained='/kaggle/working/rtdetr-l.pt' Шлях до файлу з попередньо натренованими вагами, які будуть використані як стартова точка для подальшого навчання.

Параметри даних:

data='/kaggle/working/ultralytics/data/mad.yaml' Шлях до YAML-файлу, що містить інформацію про набір даних. У цьому файлі вказуються:

- Шляхи до тренувальних, валідаційних та тестових даних.
- Кількість класів (nc).
- Назви класів (names).

3.5 Тренування і аналіз результатів

Під час тренування моделі RT-DETR було досягнуто значних результатів у виявленні об'єктів. Тренування тривало 50 епох із розміром зображень 640×640 пікселів і розміром пакета 16. Використання GPU сягало 14.3 ГБ пам’яті, а час на одну епоху становив близько 16 хвилин 38 секунд.

Основні метрики оцінювання на тренувальному наборі даних: точність моделі (Precision) становила 91.3%, що свідчить про низький рівень хибно-позитивних спрацьовувань, повнота (Recall) досягла 87.7%, тобто модель коректно ідентифікує більшість об'єктів. Середнє значення точності (mAP@50) становило 91.1%, а середнє значення точності при різних порогах

IoU (mAP@50-95) – 89.4%, що підтверджує здатність моделі до узагальнення (рис. 3.16).

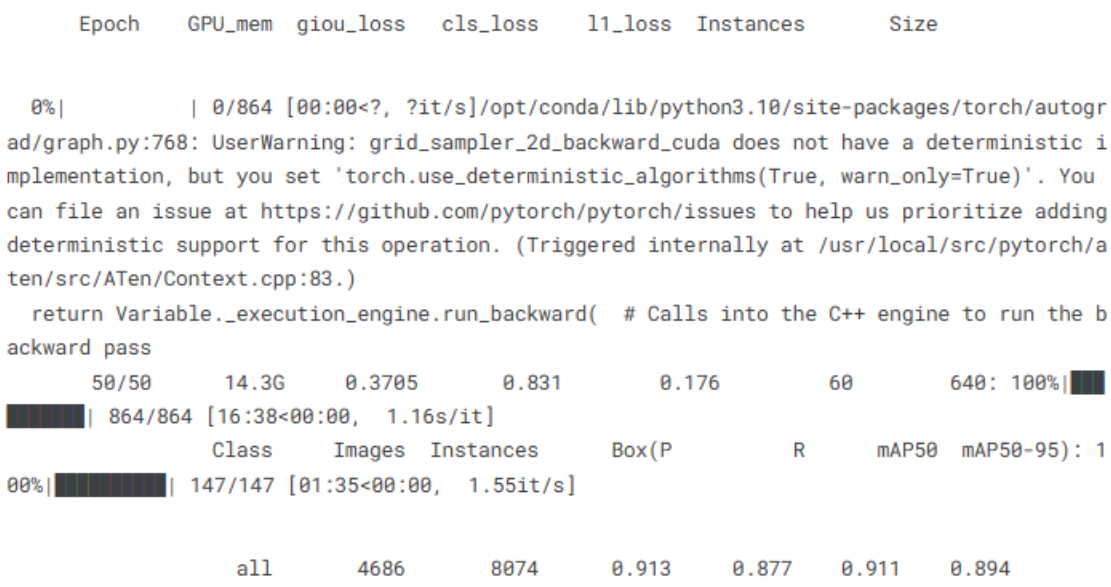

Рис. 3.16. Результати тренування моделі

Під час тренування відстежувалися втрати, пов’язані з різними аспектами моделі:

- giou_loss (втрати геометричної відповідності передбачених рамок із реальними);
- cls_loss (втрати класифікації об'єктів);
- l1_loss (втрати локалізації через абсолютні відхилення координат).

Усі типи втрат зменшувались протягом епох, що демонструє успішне навчання моделі.

Виявлено попередження щодо недетермінованої реалізації grid_sampler_2d_backward_cuda, що може впливати на відтворюваність результатів.

Узагальнено, модель RT-DETR показала високі результати у виявленні об'єктів, особливо за значеннями точності та середньої точності. Однак повнота моделі залишає можливості для вдосконалення, особливо у виявленні

малопомітних об'єктів. Модель готова до практичного використання з можливістю подальшого налаштування для покращення продуктивності.

Для перевірки навчених ваг було виконано процес оцінки ефективності моделі на валідаційному наборі даних. Для цього було використано інструмент, призначений для роботи з нейронними мережами, що застосовуються для задач комп'ютерного зору, зокрема виявлення об'єктів (рис. 3.17).

```
results = model.val(
        data="/kaggle/working/ultralytics/data/mad.yaml",
        split="val",
        save_json=True,
        device=0,
        batch=35,
        imgsz=1280
    )
```

Рис. 3.17. Лістинг коду для ініціалізації процесу валідації моделі

Процедура валідації включала наступні дії:

1. **Завантаження моделі з попередньо навченими вагами:** Було обрано навчену модель з вагами, отриманими в результаті навчання.
2. **Використання набору даних для перевірки:** Шлях до набору даних у форматі YAML було вказано як вхідний параметр. Файл YAML визначає структуру даних, включаючи розташування зображень для валідації, кількість класів і їхні назви.
3. **Налаштування розміру пакету:** Було встановлено розмір пакету на 35. Це визначає кількість зображень, оброблених моделлю одночасно, що впливає на швидкість обробки та ефективність використання обчислювальних ресурсів. Задання розміру зображення: Для забезпечення високої точності розпізнавання було обрано розмір

зображення 1280 пікселів. Це дозволяє моделі отримати більше деталей, особливо для об'єктів, що займають невелику площу на зображенні.

Було виконано процедуру валідації моделі для оцінки її здатності виявляти та класифікувати об'єкти на зображеннях із валідаційного набору даних (рис. 3.18). Для перевірки було використано модель RT-DETR Large, яка має 502 шари та 32,135,810 параметрів, забезпечуючи обчислювальну потужність у 103.8 GFLOPs.

```
val: Scanning /kaggle/working/ultralytics/data/valid/labels... 4686 images, 0 ba
val: New cache created: /kaggle/working/ultralytics/data/valid/labels.cache
```

Class	Images	Instances	Box(P	R	mAP50	m
all	4686	8074	0.951	0.855	0.888	0.838
A10	100	161	0.993	0.89	0.924	0.856
A400M	88	127	0.991	0.888	0.929	0.897
AG600	58	69	0.985	0.928	0.98	0.915
AH64	64	101	0.891	0.89	0.918	0.823
AV8B	76	125	0.979	0.896	0.906	0.882
An124	26	33	0.887	0.939	0.982	0.956
An22	16	16	1	1	0.995	0.995
An225	22	24	0.939	0.644	0.716	0.691
An72	35	53	0.975	0.746	0.816	0.767
B1	98	146	0.949	0.932	0.955	0.905
B2	100	127	0.956	0.89	0.945	0.9
B21	9	9	0.964	0.778	0.775	0.761
B52	119	160	0.94	0.886	0.941	0.884
Be200	65	68	0.988	0.956	0.955	0.933
C130	197	318	0.949	0.896	0.93	0.85
C17	120	173	0.91	0.861	0.92	0.834
C2	191	253	0.98	0.96	0.981	0.958
C390	38	41	0.875	0.78	0.831	0.777
C5	82	88	0.922	0.941	0.961	0.926
CH47	64	72	0.951	0.803	0.86	0.794
CL415	61	72	0.968	0.889	0.898	0.854
E2	76	120	0.973	0.909	0.931	0.876
E7	48	55	0.961	0.873	0.903	0.888
EF2000	114	177	0.94	0.881	0.917	0.871
F117	63	90	0.988	0.88	0.927	0.888
F14	72	129	0.982	0.846	0.879	0.847
F15	202	350	0.956	0.949	0.955	0.905
F16	263	444	0.904	0.825	0.873	0.802
F18	190	356	0.963	0.88	0.923	0.875
F22	86	133	0.946	0.865	0.875	0.845
F35	216	319	0.934	0.865	0.922	0.849
F4	120	196	0.976	0.845	0.884	0.822
H6	48	92	0.976	0.893	0.935	0.873
J10	65	209	0.978	0.845	0.865	0.813
J20	94	186	0.963	0.839	0.872	0.826

JAS39	89	139	0.927	0.821	0.869	0.825
JF17	37	46	0.925	0.891	0.929	0.894
JH7	39	58	0.945	0.881	0.912	0.886
KC135	60	94	0.952	0.849	0.887	0.837
KF21	26	29	0.923	0.862	0.868	0.854
KJ600	10	11	0.773	0.636	0.571	0.539
Ka27	17	18	0.99	1	0.995	0.954
Ka52	31	48	0.997	0.958	0.96	0.918
MQ9	64	75	0.986	0.907	0.933	0.9
Mi24	39	67	0.969	0.922	0.923	0.879
Mi26	10	11	1	0.686	0.784	0.738
Mi28	14	31	0.894	0.839	0.857	0.813
Mig29	41	55	0.891	0.691	0.718	0.689
Mig31	66	108	0.965	0.926	0.952	0.886
Mirage2000	55	93	0.975	0.846	0.91	0.843
P3	63	101	0.956	0.911	0.942	0.925
RQ4	66	79	0.893	0.848	0.874	0.801
Rafale	94	159	0.922	0.887	0.922	0.883
SR71	52	69	1	0.891	0.925	0.891
Su24	58	114	0.944	0.879	0.918	0.793
Su25	47	70	0.923	0.886	0.904	0.829
Su34	83	114	0.976	0.886	0.902	0.845
Su57	68	114	0.983	0.851	0.887	0.86
TB001	10	10	0.858	0.9	0.972	0.847
TB2	30	135	0.979	0.703	0.924	0.794
Tornado	82	123	0.985	0.821	0.845	0.833
Tu160	68	91	0.972	0.879	0.901	0.858
Tu22M	30	66	0.878	0.742	0.755	0.68
Tu95	54	82	0.975	0.945	0.96	0.906
U2	64	66	0.942	0.955	0.954	0.921
UH60	31	54	0.924	0.667	0.673	0.602
US2	143	154	0.949	0.974	0.973	0.935
V22	133	206	0.965	0.808	0.862	0.749
Vulcan	88	117	0.981	0.88	0.939	0.892
WZ7	14	17	0.991	0.588	0.684	0.593
XB70	45	46	1	0.873	0.902	0.838
Y20	41	49	0.976	0.821	0.879	0.829
YF23	25	41	0.968	0.976	0.976	0.946
Z19	10	22	0.879	0.409	0.407	0.399

Speed: 1.0ms preprocess, 68.3ms inference, 0.0ms loss, 0.3ms postprocess per image
Results saved to **runs/detect/val**

Рис. 3.18. Результати валідації моделі з даними для кожного з класів

Загальні показники точності моделі:

- Precision (точність): 0.951. Модель виявила 95.1% об'єктів серед усіх передбачень як коректно класифіковані.

- Recall (повнота): 0.855. Із усіх реальних об'єктів модель коректно розпізнала 85.5%.
- mAP50: 0.888. Середня точність при порозі перекриття 50%.
- mAP50-95: 0.838. Середня точність за різними порогами перекриття (від 50% до 95%).

Показники для окремих класів: Для кожного класу об'єктів було оцінено наступні метрики:

- Precision: Відсоток передбачених об'єктів, які є правильними.
- Recall: Частка виявлених об'єктів серед усіх присутніх.
- mAP50 та mAP50-95: Якість прогнозу за різними порогами перекриття.

Наприклад:

- Для класу An22 досягнуто найвищих показників із Precision і Recall, які дорівнюють 1.0.
- Клас KJ600 показав нижчі результати (Precision: 0.773, Recall: 0.636).
- Для найскладніших об'єктів, таких як Z19, показники точності були найнижчими (Precision: 0.879, Recall: 0.409).

Всі результати валідації було збережено у відповідну папку для подальшого аналізу та візуалізації. Результати валідації демонструють високі показники точності та повноти для більшості класів. Найвищих значень mAP було досягнуто для класів із чітко окресленими характеристиками (наприклад, An22, Ka27). Натомість, найскладнішими для моделі виявилися класи з меншою кількістю прикладів або складними для розпізнавання особливостями, такими як Z19.

Для аналізу результатів роботи моделі було виконано процедуру візуалізації зображень, що містять передбачені моделлю об'єкти та їхні межі. Для забезпечення послідовності у відображенні, було зчитано всі шляхи до файлів у вказаній директорії, що зберігають результати. Було враховано файли у форматах PNG і JPG.

Для кожного зображення було побудовано окремий графік із використанням бібліотеки візуалізації даних. Графік мав розмір, достатній для детального перегляду зображення, та відображав його без втрати важливої інформації. Було забезпечено візуалізацію зображення в його оригінальному розмірі та кольорах.

Крива F1-Confidence є графічним представленням, що демонструє залежність метрики F1-score від рівня впевненості (Confidence Threshold), який застосовується до передбачень моделі. Ця крива є корисним інструментом для оцінки ефективності моделі та вибору оптимального значення порогу впевненості (рис. 3.19). F1-score - метрика, яка є гармонійним середнім між точністю (Precision) і повнотою (Recall).

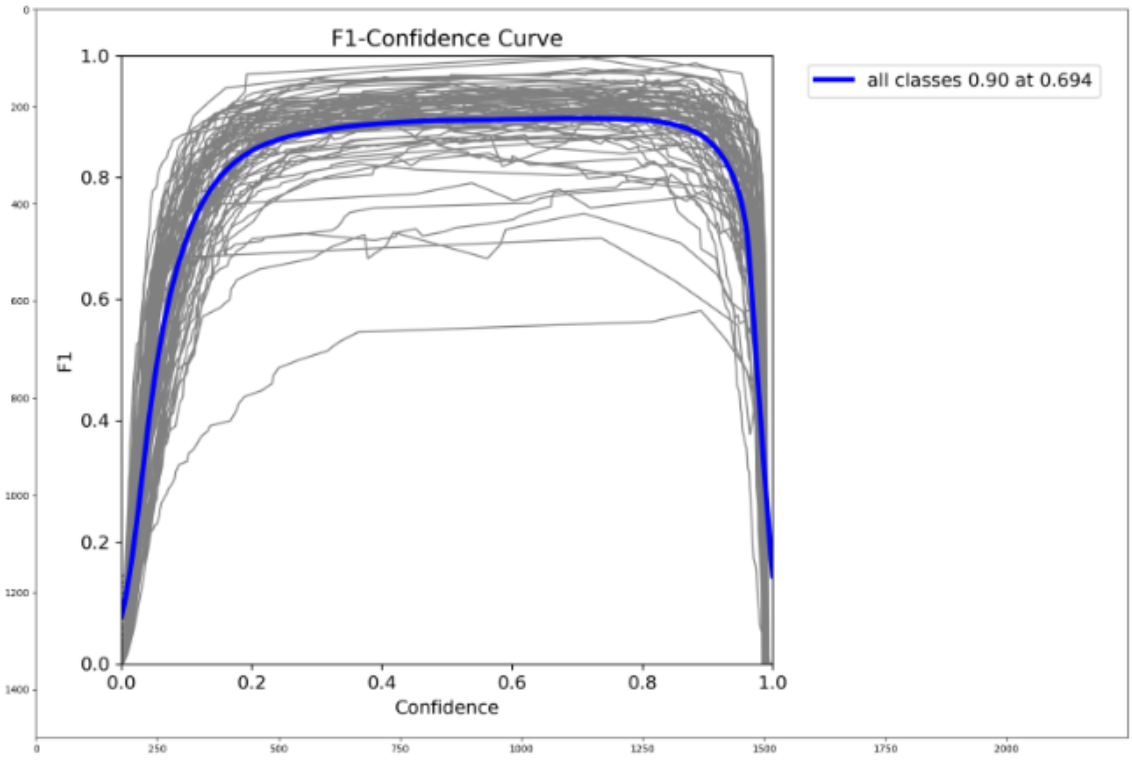

Рис. 3.19. – крива F1-Confidence

Крива Precision-Recall (P-R) — це графічне представлення залежності між метриками точності (Precision) і повноти (Recall) для різних значень порогу впевненості моделі. Ця крива є важливим інструментом для оцінки

продуктивності моделей класифікації, особливо в задачах із незбалансованими класами (рис. 3.20).

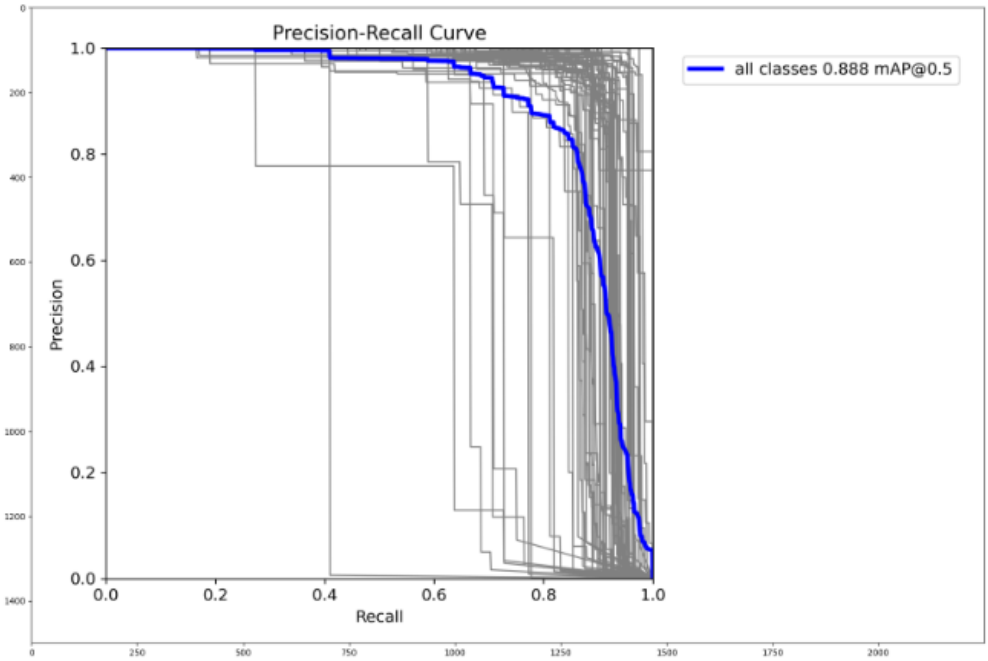

Рисунок 3.20 Крива Precision-Recall

Крива **Precision-Confidence** відображає залежність точності моделі (Precision) від порогу впевненості (Confidence Threshold), використаного для класифікації передбачень (рис. 3.21).

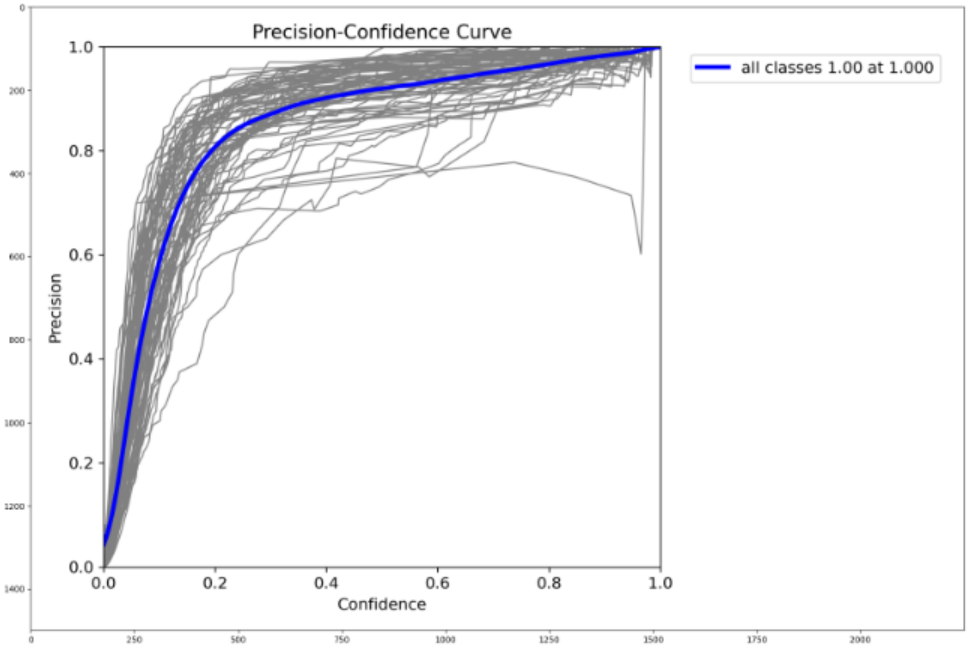

Рисунок 3.21 Крива Precision-Confidence

Крива Recall-Confidence демонструє залежність показника Recall від порогу впевненості (Confidence Threshold) для класифікації передбачень. Вона використовується для оцінки того, як зміни в порозі впевненості впливають на здатність моделі знаходити всі позитивні приклади (рис. 3.22).

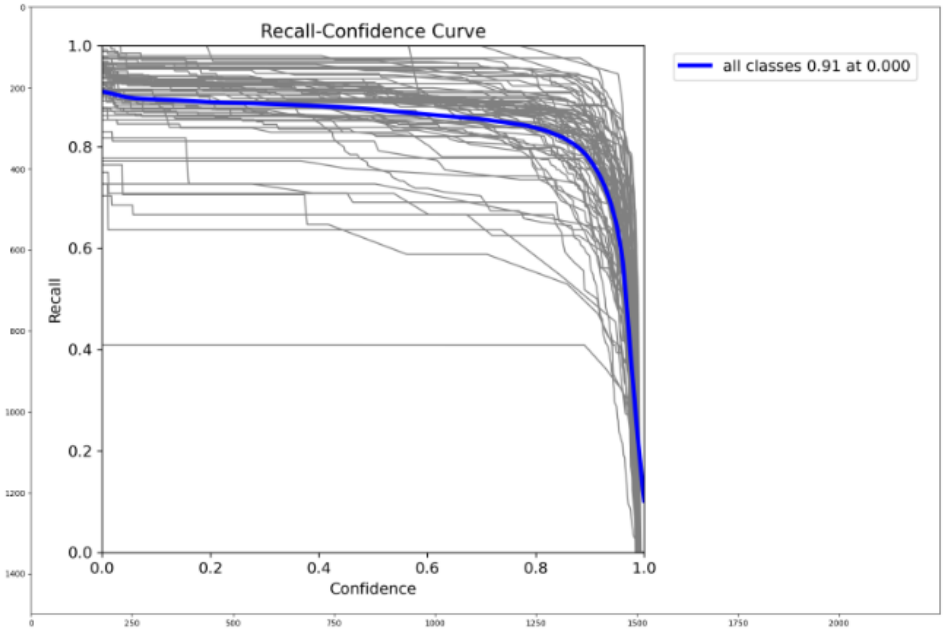

Рисунок 3.22 Крива Recall-Confidence

В результаті було забезпечено відображення всіх доступних зображень із результатами роботи моделі. Така процедура дозволяє проводити візуальний аналіз точності моделі, оцінювати її здатність виявляти об'єкти у межах зображення, а також виявляти можливі випадки неправильного розпізнавання або пропущення об'єктів.

Візуалізація передбачень моделі RT-DETR наведено на рисунку 3.23

Рис. 3.23. Візуалізація передбачень моделі RT-DETR

Висновки

Було проведено тестування попередньо навчених моделей RT-DETR LARGE та RT-DETR EXTRA LARGE на наборі даних COCO 2017.

У результаті проведених тестувань моделей RT-DETR-Large та RT-DETR-Extra-Large на платформі Kaggle було отримано важливі метрики якості та швидкості, що дозволяють оцінити їх ефективність для задач виявлення об’єктів у реальному часі. Порівняння результатів з метриками, наданими розробниками, показало високу точність моделей, з невеликою різницею в якості (до 1,4%). Однак, незважаючи на однакові умови тестування, швидкість обробки зображень суттєво відрізнялась через використання різних інструментів оптимізації (TensorFlow/PyTorch vs. TensorRT). Візуалізація передбачень моделі продемонструвала деякі проблеми з точністю для малих і частково перекритих об’єктів, що вказує на потребу в подальшій оптимізації для обробки таких випадків.

Було здійснено тренування моделі RT-DETR LARGE на наборі даних Military Aircraft Detection Dataset.

Під час проведення тренування та валідації моделі RT-DETR було досягнуто значних результатів у задачі виявлення і класифікації об'єктів на зображеннях. Процес тренування тривалістю 50 епох з розміром зображень 640×640 пікселів і розміром пакета 16 дозволив досягти високої точності моделі, зокрема середній значення точності mAP@50 досягло 91.1%. Незважаючи на успішні результати в загальних показниках, було виявлено можливості для покращення повноти моделі, зокрема у виявленні малопомітних об'єктів. Оцінка моделі на валідаційному наборі даних показала добрі результати з точністю 95.1% і повнотою 85.5%, що свідчить про ефективність моделі в більшості класів. Однак для окремих класів, таких як Z19, точність і повнота були нижчими, що вказує на складність розпізнавання цих об'єктів. Використання таких методів візуалізації, як графіки Precision-Recall і F1-Confidence, дозволяє здійснити детальний аналіз і вибір

оптимальних порогів впевненості для подальшого покращення продуктивності моделі. Всі результати були зафіксовані та використані для подальшого аналізу, що дозволяє здійснювати оптимізацію моделі в напрямку більш точного виявлення складних або малопомітних об'єктів.

РОЗДІЛ 4
РОЗРОБКА СТАРТАП ПРОЕКТУ

4.1. Загальна характеристика стартап-проекту

Основною ідеєю даного стартап проекту реалізація можливості застосування методу виявлення літальних апаратів військового типу та їх класифікації для підвищення ефективності дії засобів протиповітряної оборони.

Більш детальна інформація по особливостям та характеристиці проекту наведена у таблицях 4.1 – 4.3.

Таблиця 4.1.

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Застосування методів виявлення об’єктів реального часу для підвищення ефективності засобів протиповітряної оборони	1. Автоматичне виявлення та класифікація повітряних цілей	Застосування моделі виявлення для автоматичного виявлення і класифікації різних типів військових літальних апаратів, включаючи безпілотні летальні апарати, вертольоти, бойові літаки та ракети. Це дозволяє системам ППО своєчасно і точно визначити загрози, що надходять, та класифікувати їх за типом.
	2. Пріоритетність та оцінка загрози	Використання виявлених об'єктів для оцінки рівня загрози, визначаючи

		пріоритет для знищення, базуючись на таких характеристиках, як швидкість, висота, траєкторія руху і тип літального апарату. Це дозволяє забезпечити ефективне використання ресурсів системи ППО.
	3. Оптимізація часу реагування	Моделі виявлення об'єктів можуть значно скоротити час, необхідний для виявлення та прийняття рішень щодо нейтралізації загрози.
	4. Моніторинг та аналіз ситуації в реальному часі	Виявлення об'єктів дозволяє створити систему моніторингу повітряного простору в реальному часі, що дає можливість прогнозувати можливі шляхи атак та вчасно коригувати дії сил ППО.
	5. Розпізнавання камуфляжних та малопомітних цілей	Моделі, навчена на різноманітних типах літальних апаратів, можуть забезпечити високу точність виявлення навіть малопомітних або камуфльованих об'єктів.

Продукт, заснований на методах виявлення об'єктів реального часу для підвищення ефективності засобів протиповітряної оборони, має кілька унікальних особливостей, що визначають його конкурентні переваги на ринку оборонних технологій:

Таблиця 4.2.

Опис унікальних властивостей продукту

№ п/п	Назва властивості	Пояснення
1	Інтелектуальна модель виявлення	Продукт використовує потужну модель виявлення, треновану на спеціалізованому наборі даних, що містить різноманітні типи військових літальних апаратів.
2	Швидкість обробки в реальному часі	Завдяки оптимізації алгоритмів та використанню сучасних обчислювальних потужностей, продукт забезпечує швидку обробку вхідних даних у реальному часі.
3	Масштабованість і адаптивність	Модель спроектована таким чином, щоб її можна було адаптувати під конкретні умови та вимоги користувача. Це дозволяє ефективно інтегрувати систему в існуючі засоби протиповітряної оборони, а також адаптувати її під нові типи загроз, що з'являються в майбутньому.
4	Аналіз і прогнозування загроз	Продукт не обмежується лише виявленням об'єктів, але й здатний аналізувати траєкторії руху цілей, оцінюючи їхні потенційні шляхи. Це дозволяє точно прогнозувати можливі

		напрямки атаки і приймати рішення про нейтралізацію загрози ще до того, як вона наблизиться до критичних точок.
5	Інтеграція з багатофункціональними системами	Продукт може бути інтегрований з іншими засобами ППО, такими як радіолокаційні станції, сенсори зображень, тепловізори та зенітні ракети. Така інтеграція дозволяє створити єдину автоматизовану систему управління обороною, яка здійснює координацію всіх елементів у реальному часі.
6	Підвищена стійкість до камуфляжних технологій	Модель демонструє високу ефективність виявлення навіть малопомітних, камуфльованих або швидко маневруючих об'єктів.
7	Оптимізація використання ресурсів	Система адаптує своє реагування на основі поточної ситуації та доступних ресурсів, що дозволяє ефективно використовувати обмежені потужності ППО, оптимізуючи час реакції та розподіл ресурсів між різними загрозами.

Таблиця 4.3.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтраль-на сторона)	S (сильна сторона)
		Мій проект	Конкурент 1	Конкурент 2			
1	Бюджетне фінансування	Розробка за класний рахунок	Використання інвестицій	Розробка за класний рахунок	Відсутність фінансування	Часткове фінансування	Повне фінансування проекту
2	Використання сучасної техніки та якісної матеріальної бази	Використання сучасних технологій	Використання сучасних технологій	Використання сучасних технологій	Застосування застарілих технологій	Часткова технічна комплектація	Застосування новітніх технологій
3	Наявність маркетингової стратегії	Достатнє використання реклами	Достатнє використання реклами	Недостатнє використання реклами	Відсутність рекламування	Часткова трансляція	Активна трансляція
4	Високий професійний рівень розробки	Алгоритм розроблений згідно стандартів	Розроблено групою професіоналів	Потребує допрацювання	Неналежним чином реалізований продукт	Необхідність у систематичній підтримці продукту	Належним чином розроблений продукт
5	Використання бізнес-послуг	Наявна	Наявна	Відсутня	Відсутність консультування	Часткове консультування	Повне ведення проекту

4.2. Технологічний аудит ідеї проекту

Таблиця 4.4.

Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
1	Застосування методу виявлення об’єктів реального часу для підвищення ефективності засобів протиповітряної оборони	Ultralytics	Наявна	У відкритому доступі
2		TensorFlow	Наявна	У відкритому доступі
3		EfficientNetB7	Наявні	У відкритому доступі

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Відносно загальних характеристик ринку відносно даної сфери, можна сказати, він знаходиться, по більшій мірі, на стадії активного розвитку. Існує дуже велика кількість досліджень та удосконалень існуючих рішень по застосуванню методів виявлення об’єкту реального часу, що є достатньо вигідним часом по впровадженню стартап рішень. У таблиці 4.5 наведено попередню характеристику потенційного ринку стартап-проекту.

Таблиця 4.5.

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, usd/ум.од	760000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Дослідження випереджують реальне впровадження
5	Специфічні вимоги до стандартизації та сертифікації	Присутні
6	Середня норма рентабельності в галузі (або по ринку), %	87
7	Попит	На стадії зростання
8	Технологія	На стадії активного розвитку
9	Зміни в міжнародній торгівлі і прямих інвестиціях	На стадії зростання
10	Інтенсивність конкуренції	На стадії зростання
11	Швидкість зміни умов ринку	Ринкові умови змінюються достатньо швидко, що потребує додаткових витрат на пристосування.
12	Доступність інвестицій	Необмежено

Таблиця 4.6.

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності поведінки потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Зміцнення національної безпеки, захист від повітряних атак	Національні уряди та військові структури	Потребують високої точності та надійності систем, можливості швидкого реагування в реальному часі. Вищі вимоги до інтеграції з існуючими оборонними системами.	Система повинна бути інтегрована з іншими засобами ППО, мати високу точність, здатність до швидкого реагування та адаптації до нових загроз.
2	Підвищення ефективності оборонних технологій	Оборонні підприємства, що розробляють ППО	Вимоги до адаптації технологій до нових типів загроз і можливості тестування на різних	Висока масштабованість, підтримка різних типів загроз, можливість налаштування під специфічні потреби

			платформах. Цікавляться новітніми технологіями, що мають потенціал для подальшого вдосконалення.	оборонних систем.
3	Розвиток передових технологій у військовій сфері	Міжнародні організації, що займаються безпекою	Потребують універсальних рішень, які можна застосувати в різних країнах з урахуванням специфіки локальних загроз. Зазвичай мають високі вимоги до технічної підтримки та навчання.	Проста інтеграція з іншими системами, легкість у впровадженні та навчанні персоналу.
4	Зменшення ризиків для комерційної авіації	Авіаційні компанії та органи цивільної авіації	Мають специфічні вимоги щодо виявлення загроз для цивільних	Висока точність виявлення загроз, швидка реакція, інтеграція з існуючими

			літаків. Потрібен баланс між високою точністю та швидким реагуванням.	системами моніторингу авіації.

Таблиця 4.7.

Аналіз загроз запуску та роботи стартап-проекту

№ п/п	Фактор загрози	Опис загрози	Планове реагування компанії
1	Технічні труднощі в розробці моделі	Можливі проблеми з адаптацією моделі для роботи з різними типами об'єктів, зміна характеристик даних або складність інтеграції з існуючими оборонними системами.	Підвищена увага до тестування моделей на різноманітних типах даних, постійне оновлення та вдосконалення алгоритмів. Залучення фахівців для забезпечення безперервної підтримки та розвитку технології.
2	Висока конкуренція на ринку	Зростаюча кількість компаній, що працюють у сфері оборонних технологій, може	Стратегічне позиціонування продукту через унікальність та технічні переваги, акцент на інноваційність і

		ускладнити вихід на ринок.	підтримку клієнтів. Прогнозування та аналітика для адаптації до змін на ринку.
3	Регуляторні та юридичні бар'єри	Можливі зміни в міжнародному праві, особливо щодо оборонних технологій і експортних обмежень, що можуть обмежити доступ до деяких ринків.	Співпраця з юридичними консультантами для моніторингу регуляцій у різних країнах. Розробка альтернативних стратегій для виходу на ринки з різними вимогами.
4	Фінансова нестабільність або відсутність інвестицій	Труднощі у залученні необхідного капіталу або нестабільність на фінансових ринках може вплинути на швидкість розвитку проекту.	Диверсифікація джерел фінансування, стратегічне планування та економія витрат, залучення інвесторів з великою увагою до стабільності фінансів.
5	Труднощі у підтримці масштабування	При збільшенні обсягів обробки даних та інтеграції з іншими системами може виникнути потреба в значних ресурсах для	Використання гнучких архітектур та хмарних рішень для швидкого масштабування, постійне вдосконалення інфраструктури,

		масштабування технології.	адаптація до змінних потреб клієнтів.

Таблиця 4.8.

Аналіз можливостей по реалізації стартап-проекту

№ п/п	Фактор можливості	Зміст можливості	Можлива реакція компанії
1	Розвиток технологій в сфері ШІ та машинного навчання	Прогрес у галузі штучного інтелекту та машинного навчання відкриває нові можливості для покращення точності моделей виявлення об'єктів у реальному часі.	Активне впровадження нових методів машинного навчання для покращення ефективності моделей, партнерство з університетами і науковими центрами для досліджень.
2	Зростання попиту на засоби протиповітряної оборони	Підвищення міжнародної напруженості та потреба в сучасних системах ППО створюють значний попит на новітні технології виявлення загроз.	Позиціонування продукту як інноваційного рішення для забезпечення національної безпеки, активне залучення до оборонних тендерів і співпраця з урядами.
3	Зростання важливості кібербезпеки в	Потреба в безпечних системах ППО, що захищають	Інтеграція високих стандартів кібербезпеки в розробку продукту,

	оборонних технологіях	від кіберзагроз, створює можливість для розвитку продуктів, які поєднують виявлення об'єктів і кіберзахист.	співпраця з фахівцями в галузі кіберзахисту для забезпечення надійності продукту.

Таблиця 4.9.

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	Прояв характеристики конкурентного середовища	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції	Чиста конкуренція	В умовах чистої конкуренції, де існує безліч конкурентів, компанії потрібно акцентувати увагу на унікальності продукту та технологічній інноваційності.
Рівень конкурентної боротьби	Національний	Національний рівень конкуренції вимагає адаптації до специфіки національного ринку та орієнтації на локальні потреби.
Галузева ознака	Внутрішньогалузева	Внутрішньогалузева конкуренція означає необхідність постійного вдосконалення технологій, оптимізації процесів виробництва та інтеграції нових технологічних рішень.

Конкуренція за видами товарів	Товарно-родова	У рамках товарно-родової конкуренції, де змагаються аналогічні продукти з різними виробниками, компанія повинна акцентувати на унікальності своїх товарів, таких як їхня технічна ефективність, надійність, інтеграція з іншими системами або додаткові функції, що роблять продукт більш привабливим для споживачів.
Характер конкурентних переваг	Нецінова	Оскільки конкуренція нецінова, компанія повинна зосередитись на наданні додаткових цінностей для споживачів, таких як інноваційні технології, високий рівень обслуговування, довговічність і точність продукту, індивідуальні налаштування для специфічних потреб замовника, а також на забезпеченні надійності та безпеки продукції.
Інтенсивність конкуренції	Марочна	В умовах марочної конкуренції, коли конкуренти змагаються за унікальність бренду, компанії слід розвивати сильний бренд, який асоціюється з високою якістю, інноваціями та надійністю.

Таблиця 4.10.

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
	Продукти і компанії	Потреба у додаткових інвестиціях	Залучення постачальників послуг	Орієнто-ваність на клієнтські потреби	Прийняття рішень в процесі детального дослідження
Висновки	Основною загрозою є поява нових конкурентів у доменній області. Конкуренція на міжнародному ринку, наразі, є вкрай складною та вимагає професійн	Недостатня кількість або повна відсутність інвестицій може стати великим бар'єром для виходу на великий ринок	Показник залучення постачальників є низьким	Для даного рішення маємо більш-менш стабільну ситуацію. Більш можливими є варіанти по впровадженню додаткового програмного забезпечення у якості нових фічей.	Ускладненням може слугувати виключно відсутність кваліфікованої експертизи

	ого підходу.				

Таблиця 4.11.

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкуренто-спроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Цінова політика	Встановлення ціни на продукт має бути вигідним для обох сторін, постачальника продукту та клієнтів
2	Репутація виробника	Маючи бездоганну репутацію, довіра до компанії з боку клієнта зростає. Також вона може сприяти вдалому виходу на міжнародний рівень
4	Унікальність сервісних послуг	Надання якісного та ефективного сервісного рішення як продукту є важливим аспектом для користувача

Таблиця 4.12.

Порівняльний аналіз сильних та слабких сторін стартап-проекту

№ п/п	Фактор конкуренто-спроможності	Бали 1-20	Порівняння рейтингу товарів-конкурентів						
			-3	-2	-1	0	+1	+2	+3
1	Лідерські позиції на ринку	20					+		
2	Цінова політика компанії	18						+	

3	Торговий маркетинг	18				+			
4	Репутація компанії	19					+		
5	Лояльність до бренду	16				+			
6	Інвестиційний бюджет	19			+				

На основі попереднього аналізу різних факторів загроз та можливостей стартап-проекту по застосуванню методів виявлення об’єктів реального часу на основі трансформерів, у таблиці 4.13 наведено матрицю SWOT. Дана схема відображає загальну картину про розвитку проекту на ринку.

Таблиця 4.13.

SWOT- аналіз стартап-проекту

Сильні сторони: 1. Інноваційність технології 2. Потужна модель виявлення 3. Потенціал для інтеграції з існуючими системами протиповітряної оборони 4. Наявність досвіду в обробці великих даних і машинному навчанні	**Слабкі сторони**: 1. Високі початкові витрати на розробку 2. Залежність від якості даних 3. Високі вимоги до обчислювальних потужностей
Можливості: 1. Зростаючий попит на вдосконалення оборонних технологій	**Загрози**: 1. Сильна конкуренція 2. Зміни в законодавстві та регулюваннях

2. Розширення на міжнародні ринки 3. Співпраця з урядовими та військовими структурами 4. Інтеграція з іншими технологіями	3. Технологічні зміни 4. Безпека даних і кіберзагрози

Таблиця 4.14. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Активне опанування ринку та укладення нових договорів із підприємцями	Висока	Короткі
2	Слідування стратегії по забезпеченню конкурентоспроможності	Середня	Короткі

4.4 Розробка ринкової стратегії проекту

При планування ринкової стратегії проекту варто зосередити увагу на цільових групах потенційних споживачів.

Таблиця 4.15.

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової	Інтенсивність конкуренції в сегменті	Простота входу у сегмент

	потенційних клієнтів		групи (сегменту)		
1	Державні та міські адміністрації	Готовий	Високий	Середня	Складно
2	Інформаційні служби та приватні підприємства	Готовий	Високий	Висока	Складно
3	Рекламні агенства, власники підприємств, служби по наданню товарів та послуг	Готовий	Високий	Низька	Складно
Обрані цільові групи: стратегія диференційованого маркетингу					

Згідно аналізу потенційних комерційних груп споживачів на запропонований продукт, було обрано стратегію диференційованого маркетингу, оскільки компанія працює водночас із кількома сегментами, з розробкою окремих програм для ринкового впливу.

Таблиця 4.16.

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Стратегія зростання	Прагнення компаній до активних темпів економічного зростання	Стратегія концентрованого зростання, внутрішнє розширення продукту, а також виробництво нових продуктів	Стратегія диференційованого маркетингу

Таблиця 4.17.

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першо-прохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Стратегія поєднання нових та старих споживачів продукту	Відсутність потреби	Стратегія наслідування лідера

Таблиця 4.18.

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Забезпечення високої якості продукту	Стратегія диференційо-ваного маркетингу	Професіональний підхід до розробки продукту згідно стандартів	Якість, надійність, безпека
2	Вигідна ціна на продукт	Стратегія наслідування лідера	Широкий спектр можливостей по застосуванню додаткового пз	Доступність, унікальність, новизна

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.19

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Кластеризація мережі	Товар забезпечує ефективне формування груп вузлів	Якісний процес кластеризації. новизна

2	Скорочення витрат на інфраструктуру	Особливість даного методу сприяє скороченню витрат	Ефективність обробки даних

Таблиця 4.20

Формування системи збуту стартап-проекту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Цільові клієнти, зокрема урядові та військові установи, зазвичай приймають рішення на основі ретельного аналізу надійності технології, точності її роботи та здатності до інтеграції з існуючими системами.	Постачальник має виконувати функції консультування, впровадження технології, налаштування та навчання персоналу клієнта. Окрім цього, необхідно забезпечити технічну підтримку, оновлення програмного забезпечення, а також супровід у разі оновлення або модернізації системи протиповітряної оборони. Важливою є також допомога у проведенні випробувань на місці.	Значна	B2B, коли продажі здійснюються безпосередньо через укладення контрактів з державними і приватними компаніями, що забезпечують високий рівень безпеки.

Таблиця 4.21

Концепція маркетингових комунікацій стартап-проекту

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Адаптивність у використанні	Мережа Інтернет	Гарантування надійності	Транслювання товару	Ефективна транспортна мережа

Висновки

Проект з виявлення військових літальних апаратів на основі методів реального часу має значний потенціал у забезпеченні безпеки та підвищенні ефективності протиповітряної оборони. Використання інноваційних технологій, таких як глибокі нейронні мережі для виявлення та класифікації об'єктів, здатне значно покращити швидкість та точність виявлення загроз у реальному часі. Це дозволяє знизити витрати на операції, забезпечуючи високу надійність та ефективність оборонних систем.

Як показують дослідження та аналіз ринку, існує значний попит на такі технології серед державних установ і оборонних підприємств, що відкриває можливості для комерціалізації цього проекту. Попри високий рівень конкуренції в цій галузі, запропоноване рішення забезпечує значні конкурентні переваги завдяки своїй точності та адаптивності, що дозволяє ефективно виявляти і класифікувати різні типи військових літальних апаратів.

Зважаючи на виявлені можливості та загрози, подальший розвиток проекту та його інтеграція у військові та оборонні структури є обґрунтованим і перспективним. Створення системи виявлення з високою точністю і швидкістю може стати ключовим елементом для удосконалення протиповітряної оборони, що робить проект доцільним для впровадження в найближчому майбутньому.

ВИСНОВКИ

У ході виконання даної роботи було проведено детальний аналіз ефективності моделі RT-DETR, зокрема її здатності до вибору запитів з мінімальною невизначеністю та порівнянням її з іншими сучасними детекторами, такими як YOLO та DETR. Основна увага була приділена впливу вибору запитів на точність і швидкість детекції, а також на зменшення обчислювальних витрат.

Результати показали, що схема вибору запитів з мінімальною невизначеністю забезпечує значні переваги в порівнянні з традиційними методами. Вибір фіч з високими оцінками класифікації та IoU дозволяє досягти кращої точності детекції, зокрема за рахунок збільшення кількості високоякісних характеристик, що приводить до покращення показників AP (Average Precision). Крім того, схема з мінімальною невизначеністю сприяє більш ефективному використанню ресурсів моделі, знижуючи кількість низькоякісних характеристик.

Було проведено огляд та тестування моделі з вже натренованими вагами з аналізом отриманих метрик точності та швидкості для визначення ймовірних напрямків покращення алгоритму. Так, було простежено, що до класів, що мають нижчі показники точності, відносяться об’єкти малого розміру та об’єкти, що частково перекриваються на зображені іншими об’єктами. Причиною цього може бути обмежена кількість піксельної інформації як при навчанні моделі, так і під час тестування на нових зображеннях, або велика різниця у масштабах між малими і великими об’єктами, що призводить до більшому приділенню «уваги» трансформера більш контрастним об’єктам та об’єктам більшого розміру.

Далі було здійснено тренування власної моделі на спеціалізованому наборі даних, розробленого для задач комп'ютерного зору, спрямованого на виявлення та класифікацію військових літаків. Було проведено попереднє розділення набору даних на 3 піднабори для тренування, валідації і тестування

моделі. Процес тренування тривалістю 50 епох з розміром зображень 640×640 пікселів і розміром пакета 16 дозволив досягти високої точності моделі, зокрема середній значення точності mAP@50 досягло 91.1%. Незважаючи на успішні результати в загальних показниках, було виявлено можливості для покращення повноти моделі, зокрема у виявленні малопомітних об'єктів. Оцінка моделі на валідаційному наборі даних показала добрі результати з точністю 95.1% і повнотою 85.5%, що свідчить про ефективність моделі в більшості класів.

Використання таких методів візуалізації, як графіки Precision-Recall і F1-Confidence і інші, дозволило здійснити детальний аналіз і вибір оптимальних порогів впевненості для подальшого покращення продуктивності моделі. Всі результати були зафіксовані та використані для подальшого аналізу, що дозволяє здійснювати оптимізацію моделі в напрямку більш точного виявлення складних або малопомітних об'єктів.

Практичне значення методу виявлення об'єктів на основі моделі трансформерів RT-DETR, донавченої для набору даних Military Aircraft Detection Dataset, полягає в здатності ефективно ідентифікувати об'єкти в умовах надзвичайних ситуацій або стихійних лих. Завдяки високій точності та швидкості обробки зображень, ця модель може бути застосована для моніторингу повітряного простору, швидкого виявлення об'єктів та автоматизації процесів збору інформації в критичних умовах. Покращена модель RT-DETR дозволяє забезпечити надійне спостереження в реальному часі, що є важливим для швидкого реагування на зміни ситуації. Розширення можливостей моделі для специфічних умов підвищує її ефективність у сценаріях з обмеженими ресурсами та високими вимогами до часу обробки даних.

Було розроблено стартап-проект, що ґрунтувався на дослідженому методі виявлення об’єктів реального часу. Проект з виявлення військових літальних апаратів на основі методів реального часу має значний потенціал у

забезпеченні безпеки та підвищенні ефективності протиповітряної оборони. Використання інноваційних технологій, таких як глибокі нейронні мережі для виявлення та класифікації об'єктів, здатне значно покращити швидкість та точність виявлення загроз у реальному часі. Це дозволяє знизити витрати на операції, забезпечуючи високу надійність та ефективність оборонних систем.

СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

1. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. 2014. Microsoft COCO: Common Objects in Context. (дата звернення: 21.10.2024).
2. Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, Jie Chen. 2023. DETRs Beat YOLOs on Real-time Object Detection. (дата звернення: 21.10.2024). https://arxiv.org/abs/2304.08069
3. Official RT-DETR (RTDETR paddle pytorch), Real-Time DEtection TRansformer, DETRs Beat YOLOs on Real-time Object Detection.. https://github.com/lyuwenyu/RT-DETR/tree/main
4. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *Yolov4: Optimal speed and accuracy of object detection*. arXiv preprint arXiv:2004.10934, 2020.
5. Daniel Bogdoll, Maximilian Nitsche, and J. Marius Zollner. *Anomaly detection in autonomous driving: A survey*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4488–4499, 2022.
6. Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. *Reversible column networks*. In International Conference on Learning Representations, 2022.
7. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. *End-to-end object detection with transformers*. In European Conference on Computer Vision, pages 213–229. Springer, 2020.
8. Qiang Chen, Xiaokang Chen, Gang Zeng, and Jingdong Wang. *Group detr: Fast training convergence with decoupled one-to-many label assignment*. arXiv preprint arXiv:2207.13085, 2022.

9. Qiang Chen, Jian Wang, Chuchu Han, Shan Zhang, Zexian Li, Xiaokang Chen, Jiahui Chen, Xiaodi Wang, Shuming Han, Gang Zhang, et al. *Group detr v2: Strong object detector with encoder-decoder pretraining*. arXiv preprint arXiv:2211.03594, 2022.
10. Cheng Cui, Ruoyu Guo, Yuning Du, Dongliang He, Fu Li, Zewu Wu, Qiwen Liu, Shilei Wen, Jizhou Huang, Xiaoguang Hu, Dianhai Yu, Errui Ding, and Yanjun Ma. *Beyond self-supervision: A simple yet effective network distillation alternative to improve backbones*. CoRR, abs/2103.05959, 2021.
11. Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. *Repvgg: Making vgg-style convnets great again*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13733–13742, 2021.
12. Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. *Fast convergence of detr with spatially modulated co-attention*. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 3621–3630, 2021.
13. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.
14. Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 558–567, 2019.
15. Xin Huang, Xinxin Wang, Wenyu Lv, Xiaying Bai, Xiang Long, Kaipeng Deng, Qingqing Dang, Shumin Han, Qiwen Liu, Xiaoguang Hu, et al. Pp-yolov2: A practical object detector. arXiv preprint arXiv:2104.10419, 2021.
16. Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. Yolov6 v3.0: A full-scale reloading. arXiv preprint arXiv:2301.05586, 2023.

17. Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13619–13627, 2022.
18. Feng Li, Ailing Zeng, Shilong Liu, Hao Zhang, Hongyang Li, Lei Zhang, and Lionel M Ni. Lite detr: An interleaved multi-scale encoder for efficient detr. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18558–18567, 2023.
19. Junyu Lin, Xiaofeng Mao, Yuefeng Chen, Lei Xu, Yuan He, and Hui Xue. Dˆ 2etr: Decoder-only detr with computationally efficient cross-scale attention. arXiv preprint arXiv:2203.00860, 2022.
20. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European Conference on Computer Vision, pages 740–755. Springer, 2014.
21. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2980–2988, 2017.
22. Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8759–8768, 2018.
23. Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. In International Conference on Learning Representations, 2021.
24. Military Aircraft Detection Dataset https://www.kaggle.com/datasets/a2015003713/militaryaircraftdetectiondataset

25. RT-DETR Large/Extra Large Validate on COCO 2017
https://www.kaggle.com/code/hlibpepelov/notebook5656176946
26. RT-DETR Large Train and Validate on Military Aircraft Detection Dataset
https://www.kaggle.com/code/hlibpepelov/notebookddf54f55f0

ДОДАТОК

Частина програмного коду до розділу 2

```
%pip install ultralytics
from ultralytics import RTDETR
import time
model = RTDETR("rtdetr-l.pt")
# model = RTDETR("rtdetr-x.pt")


def measure_fps(model, dataset_path, iterations):
    start_time = time.time()

    results = model(dataset_path)

    elapsed_time = time.time() - start_time
    fps = iterations / elapsed_time
    return fps

def validate_model():
    results = model.val(
        data="/kaggle/input/yaml-coco4/custom_coco.yaml",
        split="val",
        save_json=True,
        device=0
    )

    metrics = {
        "APval": results.box.map,
        "APval_50": results.box.map50,
        "APval_75": results.box.map75,
    }

    return metrics


if __name__ == "__main__":
    metrics = validate_model()
    print("Validation Metrics:")
    for key, value in metrics.items():
        print(f"{key}: {value:.3f}")
```

```
    fps = measure_fps(model, dataset_path="/kaggle/input/coco-data2/coco/images/val2017", iterations=5000)
    print(f"FPS (batch size = 1): {fps:.2f}")
```

Частина програмного коду до розділу 3

Підготовка набору даних

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob
import shutil
import cv2
import os
from tqdm import tqdm
import seaborn as sns

MODE = 'train'# train, valid

%pip install ultralytics
import ultralytics
ultralytics.checks()

classes = np.array(
    [
        'A10', 'A400M', 'AG600', 'AH64', 'AV8B', 'An124', 'An22', 'An225', 'An72',
        'B1', 'B2', 'B21', 'B52', 'Be200', 'C130', 'C17', 'C2', 'C390', 'C5', 'CH47', 'CL415',
        'E2', 'E7', 'EF2000', 'F117', 'F14', 'F15', 'F16', 'F18', 'F22', 'F35', 'F4', 'H6',
        'J10', 'J20', 'JAS39', 'JF17', 'JH7', 'KC135', 'KF21', 'KJ600', 'Ka27', 'Ka52',
        'MQ9', 'Mi24', 'Mi26', 'Mi28', 'Mig29', 'Mig31', 'Mirage2000', 'P3', 'RQ4',
        'Rafale', 'SR71', 'Su24', 'Su25', 'Su34', 'Su57', 'TB001', 'TB2', 'Tornado',
        'Tu160', 'Tu22M', 'Tu95', 'U2', 'UH60', 'US2', 'V22', 'Vulcan', 'WZ7',
        'XB70', 'Y20', 'YF23', 'Z19'
    ]
)
print(len(classes))
```

```
csv_paths = glob.glob('../input/militaryaircraftdetectiondataset/dataset/*.csv')
jpg_paths = glob.glob('../input/militaryaircraftdetectiondataset/dataset/*.jpg')
csv_paths.sort()
jpg_paths.sort()
print('number of images:',len(csv_paths))


os.makedirs('/kaggle/working/ultralytics/data/train/images', exist_ok=True)
os.makedirs('/kaggle/working/ultralytics/data/train/labels', exist_ok=True)

os.makedirs('/kaggle/working/ultralytics/data/valid/images', exist_ok=True)
os.makedirs('/kaggle/working/ultralytics/data/valid/labels', exist_ok=True)

os.makedirs('/kaggle/working/ultralyticsdata/test/images', exist_ok=True)
os.makedirs('/kaggle/working/ultralytics/data/test/labels', exist_ok=True)

# split
for i, (csv_path, jpg_path) in enumerate(tqdm(zip(csv_paths, jpg_paths))):
    annotations = np.array(pd.read_csv(csv_path))

    if os.path.basename(csv_path)[0] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b']:
        if MODE=='train':
            jpg_file_path = '/kaggle/working/ultralytics/data/train/images/' + os.path.basename(jpg_path)
            txt_file_path = '/kaggle/working/ultralytics/data/train/labels/' + os.path.basename(csv_path)[:-4]+'.txt'
            shutil.copy(jpg_path, jpg_file_path)
        else:
            continue
    elif os.path.basename(csv_path)[0] in ['c', 'd', 'e', 'f']:#['c', 'd', 'e']:cde:val, f:test. f is also used as val for demo here
        name = os.path.splitext(os.path.basename(jpg_path))[0]
        new_name = ''.join(np.random.permutation(list(name)))
        jpg_file_path = '/kaggle/working/ultralytics/data/valid/images/' + new_name + '.jpg'
        txt_file_path = '/kaggle/working/ultralytics/data/valid/labels/' + new_name + '.txt'
```

```
            shutil.copy(jpg_path,  jpg_file_path)
      else:
            name  =  os.path.splitext(os.path.basename(jpg_path))[0]
            new_name  =  ''.join(np.random.permutation(list(name)))
            jpg_file_path  =  '/kaggle/working/ultralytics/data/test/images/'
+  new_name  +  '.jpg'
            txt_file_path  =  '/kaggle/working/ultralytics/data/test/labels/'  +
new_name  +  '.txt'
            shutil.copy(jpg_path,  jpg_file_path)

      with  open(txt_file_path,  mode='w')  as  f:
            try:
                  for  annotation  in  annotations:
                        width  =  annotation[1]
                        height  =  annotation[2]
                        class_name  =  annotation[3]
                        xmin  =  annotation[4]
                        ymin  =  annotation[5]
                        xmax  =  annotation[6]
                        ymax  =  annotation[7]
                        x_center  =  0.5*(xmin+xmax)
                        y_center  =  0.5*(ymin+ymax)
                        b_width  =  xmax  -  xmin
                        b_height=  ymax  -  ymin
                        class_num  =  np.where(classes==class_name)[0][0]
                        output_string  =  '{} {} {} {} {}\n'.format(class_num,

x_center/width,

y_center/height,

b_width/width,

b_height/height)
                        f.write(output_string)
            except:
                  print(txt_file_path)
                  0/0

print('test:',len(glob.glob('/kaggle/working/ultralytics/data/test/labels/*.txt'
)))
print('valid:',len(glob.glob('/kaggle/working/ultralytics/data/valid/labels/*.t
xt')))
```

```
print('train',len(glob.glob('/kaggle/working/ultralytics/data/train/labels/*.txt')))

df_all = pd.concat([pd.read_csv(csv_path) for csv_path in csv_paths])
df_all = df_all.reset_index(drop=True)

def determine_split(filename):
    first_char = filename[0].lower()
    if '0' <= first_char <= '9' or 'a' <= first_char <= 'b':
        return 'train'
    elif 'c' <= first_char <= 'e':
        return 'valid'
    elif first_char == 'f':
        return 'test'
    else:
        return 'unknown'

df_all['split'] = df_all['filename'].apply(determine_split)
class_distribution = df_all.groupby('split').apply(lambda x: x['class'].value_counts(normalize=True)).unstack().fillna(0)
class_distribution_reordered = class_distribution.T[['test', 'valid', 'train']]

# Plot the normalized class distribution for each split with distinctive colors
fig, ax = plt.subplots(figsize=(10, 15))

colors = ['#1f77b4', '#ff7f0e', '#2ca02c']    # Colors for train, valid, test

class_distribution_reordered.plot(kind='barh', ax=ax, width=0.8, color=colors)
ax.set_xlabel('Normalized Frequency')
ax.set_ylabel('Class Name')
ax.set_title('Normalized Class Distribution for Each Split')
plt.legend(title='Split')

plt.tight_layout()
plt.show()


# Load the dataset
#file_path = 'all_label.csv'
data = df_all
```

```
# Calculate bounding box width and height, and normalize by image width and height
data['bbox_width'] = (data['xmax'] - data['xmin']) / data['width']
data['bbox_height'] = (data['ymax'] - data['ymin']) / data['height']

# Scatter plot of bounding box width and height (normalized)
plt.figure(figsize=(12, 12))
plt.scatter(data['bbox_width'], data['bbox_height'], alpha=0.15, s=2)
plt.xlabel('Normalized Bounding Box Width')
plt.ylabel('Normalized Bounding Box Height')
plt.title('Scatter Plot of Normalized Bounding Box Width and Height')

# Add mean bounding box width and height as a red 'X'
mean_bbox_width = data['bbox_width'].mean()
mean_bbox_height = data['bbox_height'].mean()
print(f'mean_bbox_width:{mean_bbox_width}, mean_bbox_height:{mean_bbox_height}')
plt.scatter(mean_bbox_width, mean_bbox_height, color='red', marker='x', s=25, label=f'Mean: ({mean_bbox_width:1.2f}, {mean_bbox_height:1.2f})')
plt.legend()
plt.grid(True)
plt.show()

# Calculate bounding box aspect ratio
data['bbox_aspect_ratio'] = data['bbox_width'] / data['bbox_height']

# Plot histogram of bounding box aspect ratio
plt.figure(figsize=(10, 6))
plt.hist(np.log10(data['bbox_aspect_ratio']), bins=100, edgecolor='black', alpha=0.7)
plt.xlabel('Bounding Box Aspect Ratio(log10)')
plt.ylabel('Frequency')
plt.title('Histogram of Bounding Box Aspect Ratio')
plt.grid(True)
plt.show()



# Plot histogram of bounding box aspect ratio
area_ratio = (data['xmax'] - data['xmin'])* (data['ymax'] - data['ymin'])/(data['width']*data['height'])
plt.figure(figsize=(10, 6))
plt.hist(area_ratio, bins=100, edgecolor='black', alpha=0.7)
```

```
plt.xlabel('Bounding Box area_ratio')
plt.ylabel('Frequency')
plt.title('Histogram of Bounding Box area_ratio')
plt.grid(True)
plt.show()



# Calculate image aspect ratio
data['image_aspect_ratio'] = data['width'] / data['height']

# Plot histogram of image aspect ratio
plt.figure(figsize=(10, 6))
plt.hist(np.log10(data['image_aspect_ratio']), bins=100, edgecolor='black', a
lpha=0.7)
plt.xlabel('Image Aspect Ratio(log10)')
plt.ylabel('Frequency')
plt.title('Histogram of Image Aspect Ratio')
plt.grid(True)
plt.show()

# Total number of bounding boxes per aircraft type
bbox_count_per_type = data['class'].value_counts()

# Plot bar of total number of boxes per aircraft type, sorted by number
plt.figure(figsize=(12, 6))
bbox_count_per_type.sort_values(ascending=False).plot(kind='bar', color='sk
yblue')
plt.xlabel('Aircraft Type')
plt.ylabel('Total Number of Boxes')
plt.title('Total Number of Boxes per Aircraft Type')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.show()

# Mean number of boxes per image (class-agnostic)
mean_boxes_per_image = len(data) / data['filename'].nunique()

# Mean number of boxes per aircraft type
images_containing_type = data.groupby('class')['filename'].nunique()
mean_boxes_per_type = bbox_count_per_type / images_containing_type

# Plot bar of mean number of boxes per aircraft type, sorted by number
```

```
plt.figure(figsize=(12, 6))
mean_boxes_per_type.sort_values(ascending=False).plot(kind='bar', color='lightgreen')
plt.axhline(mean_boxes_per_image, color='red', linestyle='dashed', linewidth=1)
plt.text(-0.5, mean_boxes_per_image, f'Mean: {mean_boxes_per_image:.2f}', color='red', verticalalignment='bottom')
plt.xlabel('Aircraft Type')
plt.ylabel('Mean Number of Boxes')
plt.title('Mean Number of Boxes per Aircraft Type')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.show()

# Create a set of unique aircraft types for each image
unique_aircraft_per_image = data.groupby('filename')['class'].apply(set)

# Extract unique aircraft types
unique_aircraft_types = sorted(data['class'].unique())

# Create a new DataFrame to store unique pairs of aircraft types per image
pair_counts = pd.DataFrame(0, index=unique_aircraft_types, columns=unique_aircraft_types)

# Count co-occurrences of aircraft types (unique pairs per image)
for aircraft_set in unique_aircraft_per_image:
    for aircraft1 in aircraft_set:
        for aircraft2 in aircraft_set:
            if aircraft1 != aircraft2:
                pair_counts.at[aircraft1, aircraft2] += 1

# Sort the pair counts matrix by aircraft type (alphabetical order)
pair_counts_sorted = pair_counts.sort_index(axis=0).sort_index(axis=1)

# Create a mask for the diagonal
mask = np.zeros_like(pair_counts_sorted, dtype=bool)
np.fill_diagonal(mask, True)

# Plot heatmap of unique pair co-occurrence without numeric annotations, with 90-degree rotated x-ticks and white diagonal
plt.figure(figsize=(20, 17))
```

```
sns.heatmap(np.log10(pair_counts_sorted+1), annot=False, cmap='coolwar
m', linewidths=0.5, cbar_kws={'label': 'Co-occurrence Count (log10(N+1))'},
mask=mask)
plt.xlabel('Aircraft Type')
plt.ylabel('Aircraft Type')
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.show()


plt.figure(figsize=(10, 6))
plt.hist(np.log10(data['bbox_aspect_ratio']), bins=150, edgecolor='black', al
pha=0.3)
plt.xlabel('Bounding Box Aspect Ratio (log10)')
plt.ylabel('Frequency')
plt.title('Histogram of Bounding Box Aspect Ratio')
plt.grid(True)
plt.show()
```

Тренування моделі

```
if MODE=='train':
    from ultralytics import RTDETR
    model = RTDETR(model='rtdetr-l.pt')

    result=model.train(
        name='RT-DETR_01',
        epochs=50,
        imgsz=640,
        batch=16,
        #optimizer='auto',

        # augmentation
        close_mosaic=0,
        hsv_h=0.02,
        hsv_s=0.75,
        hsv_v=0.45,
        degrees=0.1,
        translate=0.25,
        scale=0.55,
        shear=0.1,
        perspective=0.0,
        flipud=0.1,
```

```
            fliplr=0.5,
            mosaic=1.0,
            mixup=0.5,

            pretrained='/kaggle/working/rtdetr-l.pt',
            data='/kaggle/working/ultralytics/data/mad.yaml',
        )
```

Валідація моделі

```
results = model.val(
            data="/kaggle/working/ultralytics/data/mad.yaml",
            split="val",
            save_json=True,
            device=0
        )
```

Візуалізація результатів

```
for path in sorted(glob.glob('/kaggle/working/runs/detect/val/*.png')):
        image = cv2.imread(path)[:,:,::-1]
        plt.figure(figsize=(20,20))
        plt.imshow(image)
        plt.show()

for path in sorted(glob.glob('/kaggle/working/runs/detect/val/*.jpg')):
        image = cv2.imread(path)[:,:,::-1]
        plt.figure(figsize=(20,20))
        plt.imshow(image)
        plt.show()
```