Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методу комп'ютерного зору для виявлення накладання об'єктів на зображеннях
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-1

Данило Норець
(власне ім’я, прізвище)

Спеціальність 122 Комп’ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту

(повна назва освітньої програми)
Керівник доц. Олексій Турута
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп’ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп’ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____« _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Норцю Данилу Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження методу комп'ютерного зору для виявлення накладання об'єктів на зображеннях

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 6 червня 20 25 р.

3. Вихідні дані до роботи здійснити дослідження, розробку та експериментальне оцінювання методів комп’ютерного зору для виявлення об’єктів, що частково перекриваються на зображеннях. У рамках роботи необхідно реалізувати програмні засоби для обробки зображень, проведення інференсу з використанням попередньо навчених моделей, збору результатів та оцінювання точності за ключовими метриками. Система повинна забезпечити зручний механізм порівняння моделей та відображення візуалізацій результатів детекції. Перелік використаних програмних засобів: Python 3 Jupyter Notebook, PyTorch, HuggingFace Transformers, Ultralytics, Matplotlib, COCO API, Label Studio.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Формування та проєктування експеременту дослідження

3) Реалізація експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Загальний аналіз предметної галузі	23.04.2025	виконано
3	Обґрунтування актуальності дослідження	24.04.2025	виконано
4	Визначення правового поля	25.04.2025	виконано
5	Визначення основних методів	26.04.2025	виконано
6	Визначення основних метрик оцінювання	27.04.2025	виконано
7	Постановка задачі	28.04.2025	виконано
8	Вибір моделей та архітектур	29.04.2025	виконано
9	Вибір та підготовка наборів даних	30.04.2025	виконано
10	Визначення параметрів навчання моделей	02.05.2025	виконано
11	Вибір метрик для оцінювання результатів	03.05.2025	виконано
12	Формалізація цілей і сценаріїв	04.05.2025	виконано
13	Архітектура реалізованих моделей	06.05.2025	виконано
14	Процедура навчання моделей	07.05.2025	виконано
15	Запуск, процес валідації та тестування	08.05.2025	виконано
16	Запуск моделей	09.05.2025	виконано
17	Оцінювання та візуалізація результатів	16.05.2025	виконано
18	Порівняння результатів	19.05.2025	виконано
19	Формування висновків	24.05.2025	виконано
20	Захист перед ЕК	06.06.2025	виконано

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олексій Турута
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 127 с., 15 рис., 3 табл., 1 дод., 12 джерел.

ВИЯВЛЕННЯ ОБ’ЄКТІВ, ІНСТАНС-СЕГМЕНТАЦІЯ, КОМП’ЮТЕРНИЙ ЗІР, НАКЛАДАННЯ, ОБРОБКА ЗОБРАЖЕНЬ, COMPUTER VISION, IMAGE PROCESSING, INSTANCE SEGMENTATION, OBJECT DETECTION, OVERLAY, YOLOV8.

Об’єктом дослідження є процес автоматизованого аналізу зображень із використанням методів комп’ютерного зору.

Предметом дослідження виступають методи та алгоритми виявлення накладених об’єктів на зображеннях з використанням інстанс-сегментації.

Метою роботи є дослідження, порівняння та оцінка ефективності сучасних методів комп’ютерного зору для визначення об’єктів, що частково або повністю перекриваються на зображеннях, з використанням глибоких нейронних мереж.

У роботі застосовувалися методи теоретичного аналізу літературних джерел, експериментальне моделювання, використання глибоких згорткових нейронних мереж (DINO, YOLOv8 Seg, RT-DETR), а також метрики оцінювання якості інстанс-сегментації.

У результаті дослідження було обґрунтовано доцільність використання методів інстанс-сегментації для задач виявлення накладених об’єктів. Проаналізовано архітектури сучасних моделей, визначено їх переваги та обмеження у контексті поставленої задачі. Розроблено підхід до оцінки якості роботи моделей на основі релевантних метрик, що дозволяє об’єктивно порівнювати їхню ефективність у різних умовах. Отримані результати можуть бути використані для подальшої розробки систем відеоаналітики, медичної діагностики, безпеки, а також інших сфер, де важливо точно розпізнавати об’єкти в умовах часткового перекриття.

ABSTRACT

Master’s thesis contains: 127 pp., 15 fig., 3 tabl., 1 ann., 12 references.

COMPUTER VISION, IMAGE PROCESSING, INSTANCE SEGMENTATION, OBJECT DETECTION, OVERLAY, YOLOV8, DINO, COCO, RT-DETR.

The object of research is the process of automated image analysis using computer vision methods.

The subject of the study is methods and algorithms for detecting superimposed objects in images using instance segmentation.

The purpose of the study is to investigate, compare and evaluate the effectiveness of modern computer vision methods for detecting objects that partially or completely overlap in images using deep neural networks.

The methods used in the study included theoretical analysis of literature, experimental modeling, deep convolutional neural networks (DINO, YOLOv8 Seg, RT-DETR), and metrics for assessing the quality of instance segmentation.

As a result of the study, the expediency of using instance segmentation methods for overlapping object detection was substantiated. The architectures of modern models are analyzed, their advantages and limitations in the context of the task are determined. An approach to evaluating the quality of models based on relevant metrics is developed, which allows for an objective comparison of their effectiveness in different conditions. The obtained results can be used for further development of video analytics systems, medical diagnostics, security, and other areas where it is important to accurately recognize objects in conditions of partial overlap.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Network – згорткова нейронна мережа;

COCO – Common Objects in Context – стандартний датасет для задач детекції та сегментації;

DINO – Distilled Knowledge with No Labels – модель сегментації без розмітки;

F1-score – метрика якості, що враховує точність та повноту;

GPU – Graphics Processing Unit – графічний процесор;

GT – Ground Truth – розмітка, що використовується як еталон;

IoU – Intersection over Union – коефіцієнт перекриття об'єктів;

JSON – JavaScript Object Notation – формат обміну структурованими даними;

mAP – mean Average Precision – середнє значення середньої точності;

Recall – частка знайдених об'єктів серед усіх істинних;

RGB – Red Green Blue – колірна модель з трьома каналами;

RT-DETR – Real-Time DEtection TRansformer – трансформер для об'єктної детекції в реальному часі;

YOLO – You Only Look Once – сімейство моделей об'єктної детекції.

ВСТУП

Сучасний світ дедалі більше інтегрує візуальні технології в різноманітні сфери людської діяльності. Від автономного транспорту до медицини, від систем відеоспостереження до промислової автоматизації – комп’ютерний зір перетворюється з експериментальної технології на повноцінний інструмент аналізу навколишнього середовища. Зі зростанням складності задач виникає потреба у розробці нових методів, здатних ефективно працювати в умовах реального, динамічного світу. Одним із найскладніших викликів у цьому контексті є виявлення та локалізація об’єктів на зображенні у разі їх часткового або повного перекриття.

У реальних сценах об'єкти рідко існують ізольовано – вони взаємодіють, перекривають один одного, створюють складні композиції. Наприклад, на міській вулиці автомобілі можуть частково закривати пішоходів, рекламні щити – транспорт, а дерева – дорожні знаки. В таких випадках класичні методи аналізу зображень часто втрачають ефективність, оскільки не можуть точно розмежувати границі об’єктів і визначити їхню належність. Це породжує необхідність у застосуванні глибших, контекстно орієнтованих підходів.

Завдання визначення накладених об’єктів є важливим підрозділом інстанс-сегментації – галузі комп’ютерного зору, яка поєднує локалізацію та сегментацію кожного окремого екземпляра об’єкта. На відміну від детектування за допомогою прямокутників (bounding boxes) або семантичної сегментації, яка не розрізняє об’єкти одного класу, інстанс-сегментація дозволяє не лише класифікувати, а й ізолювати кожен об'єкт незалежно від його положення на сцені. Це робить її особливо актуальною в задачах, де присутнє перекриття.

Сучасні архітектури глибокого навчання, зокрема DINO [1], RT-DETR [2] та YOLOv8 [3], відкривають нові можливості для вирішення цієї

проблеми. Вони здатні генерувати точні маски об’єктів, виявляти навіть незначні ознаки присутності об’єкта у зоні перекриття та адаптуватися до складних форм. У поєднанні з правильними метриками оцінки, ці моделі дозволяють досягати високого рівня точності навіть у складних умовах, наближених до реальності.

Актуальність теми обумовлена не лише технічними викликами, а й практичними потребами різних галузей. У сфері безпеки точне розпізнавання перекритих об'єктів може запобігти інцидентам або допомогти в розслідуваннях. У медицині – дозволити автоматизованим системам точно виявляти патологічні ділянки навіть у складних випадках. У промисловості – забезпечити надійний контроль якості продуктів, які перебувають у випадковому положенні на конвеєрі.

Крім прикладного аспекту, дана тема має також наукову цінність. Вона передбачає не лише адаптацію існуючих моделей до специфіки задачі, а й аналіз ефективності кожного методу через призму обраних метрик. Це дозволяє краще зрозуміти обмеження сучасних алгоритмів і сформувати напрями подальших досліджень у галузі інтелектуальної обробки візуальних даних.

Особливістю даного дослідження є системний підхід до аналізу проблеми. Робота охоплює як теоретичне обґрунтування задачі, так і практичну реалізацію моделей, що базуються на найновіших досягненнях у сфері глибокого навчання. Ретельно підібрані метрики – IoU [4], AP@0.5 [5], AP@0.75 [5], mAP [5], F1 score [5], Panoptic Quality [6] – дозволяють об’єктивно оцінити якість виявлення накладених об’єктів і сформувати повну картину щодо ефективності кожного підходу.

Не менш важливим аспектом є правова регламентація використання подібних систем. У роботі враховуються положення українського законодавства щодо захисту персональних даних, інтелектуальної власності, обробки інформації та впровадження технологій штучного інтелекту. Це дозволяє оцінити не лише технічну, а й юридичну

доцільність застосування систем визначення накладених об’єктів у реальних умовах.

Таким чином, тема дослідження поєднує в собі складну технічну задачу з високим рівнем практичної цінності та міждисциплінарний підхід, що охоплює машинне навчання, програмну інженерію, аналіз зображень та правове регулювання. Це робить її актуальною та перспективною як для академічної спільноти, так і для індустріальних застосувань.

Метою роботи є аналіз, реалізація та порівняння методів виявлення накладених об’єктів на зображеннях із використанням сучасних підходів інстанс-сегментації. Задача передбачає відбір моделей, підготовку даних, побудову експерименту та оцінку результатів за допомогою чітко визначених метрик. Отримані висновки можуть стати основою для впровадження таких систем у різних галузях або подальшого наукового дослідження.

Виконання цієї роботи дозволить отримати як глибше розуміння ефективності сучасних моделей комп’ютерного зору в умовах перекриття об’єктів, так і конкретні прикладні результати, які можуть бути адаптовані для практичного використання. Це підкреслює значущість теми не лише в теоретичному, а й у прикладному аспекті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальний аналіз та виділення особливостей предметної галузі

Зображення є однією з основних форм візуального представлення інформації у цифровому вигляді. Воно складається з пікселів – найменших елементів, кожен з яких має певне значення, що визначає його колір і яскравість. У контексті комп'ютерного зору зображення розглядається як матриця числових значень, що дозволяє алгоритмам аналізувати вміст кадру, виявляти закономірності та здійснювати різноманітні перетворення. Залежно від кількості каналів, зображення може бути чорно-білим (один канал) або кольоровим (зазвичай три канали – червоний, зелений, синій).

Накладанням об'єктів на зображенні вважається така ситуація, коли два або більше об'єктів частково чи повністю перекривають один одного у візуальному просторі. Це явище є природним у більшості реальних сцен, коли один об'єкт може затуляти частину іншого через різне положення у просторі. У цифрових зображеннях накладання призводить до складності аналізу, оскільки окремі риси об'єктів стають недоступними для безпосереднього спостереження, що ускладнює задачу їх точного виявлення та ідентифікації.

Комп'ютерний зір – це галузь штучного інтелекту, що займається розробкою алгоритмів і методів для автоматичного аналізу візуальної інформації. Метою комп'ютерного зору є імітація здатності людського зору до сприйняття навколишнього світу. Завдяки розвитку цієї галузі стало можливим виконання таких завдань, як розпізнавання облич, виявлення об'єктів, оцінка глибини сцени, класифікація зображень, аналіз руху та інші.

Однією з ключових задач комп'ютерного зору є виявлення об'єктів, тобто ідентифікація та локалізація окремих елементів у зображенні. У випадку, коли об'єкти накладаються один на одного, звичайні методи,

засновані на граничному контурі або кольорі, можуть бути недостатньо ефективними. Це потребує використання більш складних моделей, здатних аналізувати контекст, просторові зв’язки та інші ознаки, які допомагають виділити окремі об’єкти навіть при їх частковому перекритті.

Накладання також впливає на точність роботи алгоритмів сегментації зображень. Сегментація – це процес розділення зображення на значущі області, які відповідають окремим об’єктам або класам. У випадку перекриття двох об’єктів виникає проблема точного виділення меж, оскільки пікселі у зоні накладання можуть належати одразу до кількох об’єктів. Це створює необхідність використання маскових або багаторівневих моделей.

Сучасні методи вирішення задач, пов’язаних з накладанням, часто базуються на глибоких нейронних мережах. Архітектури DINO, RT-DETR, YOLOv8, або DeepLab [7] дозволяють не лише виявляти об’єкти, а й створювати для кожного з них точну маску – бінарне зображення, яке визначає форму і межі об’єкта, навіть якщо він частково закритий іншим. Такі моделі здатні обробляти складні сцени і демонструють високу ефективність у практичних застосуваннях.

Іншою важливою концепцією є метрика IoU (Intersection over Union), яка дозволяє кількісно оцінювати якість виділення об’єктів. Вона обчислює ступінь перекриття між передбаченою маскою об’єкта та реальним розташуванням цього об’єкта. У випадках з накладанням значення IoU може суттєво знижуватись, якщо модель неправильно визначає межі.

Проблематика накладання особливо актуальна в таких галузях, як автономне водіння, медична діагностика, відеоспостереження та робототехніка. Наприклад, в автономному автомобілі система повинна правильно розпізнавати пішоходів, які можуть бути частково закриті іншими транспортними засобами. Помилки в таких сценаріях можуть

призвести до критичних наслідків, тому точність розпізнавання накладених об’єктів є надзвичайно важливою.

Для підвищення точності виявлення накладених об’єктів дослідники також використовують методи мультискейлового аналізу, які дозволяють розглядати зображення на різних рівнях масштабів. Це допомагає моделі розпізнавати як великі, так і дрібні об’єкти, навіть у складних умовах сцени.

Крім того, важливу роль відіграє навчання моделей на анотаціях з перекриттям. Якісно розмічені набори даних, у яких позначено об’єкти навіть у зоні перекриття, дають змогу нейронним мережам вивчити характерні патерни та поліпшити свою здатність до генералізації на нових прикладах.

Таким чином, зображення, накладання об’єктів та комп’ютерний зір є тісно пов’язаними поняттями, що формують основу дослідження у цій роботі. Розуміння цих базових понять дозволяє краще оцінити виклики, пов’язані з виявленням накладених об’єктів, та обґрунтувати вибір методів і підходів для їх вирішення

1.2 Обґрунтування актуальності дослідження

Актуальність теми визначення накладених об’єктів у комп’ютерному зорі зумовлена широким колом практичних задач, у яких точне розділення об’єктів є критичним для функціонування систем. У реальному світі об’єкти дуже часто перекривають один одного – як у зображеннях вулиць мегаполісів, так і в сценах всередині приміщень, у кадрах з відеокамер спостереження, у знімках мікроскопічного рівня чи при аналізі медичних даних. Класичні методи комп’ютерного зору, які добре працюють у контрольованих умовах, часто виявляються недостатніми при обробці таких складних сцен.

Розробка методів для точного виявлення накладених об’єктів дозволяє суттєво підвищити точність у багатьох автоматизованих системах. Наприклад, в автономному транспорті точне визначення пішоходів, велосипедистів чи інших машин, навіть коли вони частково перекриваються, є необхідною умовою для забезпечення безпечного руху. Помилка у виявленні об’єкта внаслідок накладання може призвести до аварій чи некоректного прийняття рішень.

У сфері відеоспостереження системи мають справу з динамічними сценами, де люди можуть проходити один повз одного, закривати камеру чи частково перекривати об’єкти інтересу. У таких випадках точне виявлення кожного об’єкта, незважаючи на перекриття, дозволяє коректно ідентифікувати та відстежити його протягом часу. Це критично для систем безпеки, аналізу поведінки натовпу та розпізнавання інцидентів.

У медичній діагностиці, зокрема в аналізі рентгенівських знімків чи МРТ, анатомічні структури часто накладаються одна на одну, і для правильного виявлення патологій необхідно точно розмежовувати ці об’єкти. Системи автоматичного аналізу медичних зображень можуть значно виграти від поліпшених алгоритмів, здатних враховувати просторові конфлікти та взаємне перекриття тканин або органів.

У промислових застосуваннях, таких як контроль якості на конвеєрах, об'єкти продукції можуть випадково перекриватися. Надійне виявлення дефектів або неправильного розташування елементів можливе лише за умови, що алгоритм правильно інтерпретує накладені частини і вміє розділити їх на окремі сутності.

Крім того, тема має наукову цінність, оскільки вимагає поєднання просторового аналізу, контекстного розуміння сцени, і глибокої інтерпретації візуальної інформації. Вона передбачає використання найсучасніших архітектур глибокого навчання, таких як трансформери, багаторівнева сегментація та механізми уваги. Розвиток таких рішень сприяє загальному прогресу галузі комп’ютерного зору.

У довгостроковій перспективі, удосконалення методів визначення накладання може значно покращити взаємодію між людиною і машиною, зокрема в розширеній реальності, де точне розміщення віртуальних об’єктів серед реальних залежить від правильного виявлення і сегментації елементів сцени. Таким чином, ця тема має як практичне, так і фундаментальне значення в межах сучасного комп’ютерного зору.

1.3 Визначення правового поля предметної галузі

В Україні діяльність, пов’язана з використанням систем комп’ютерного зору, зокрема з визначенням накладених об’єктів на зображеннях, регулюється низкою нормативно-правових актів. Ці документи охоплюють питання захисту персональних даних, авторського права, інтелектуальної власності, використання штучного інтелекту, безпеки, а також правові аспекти збору, обробки та зберігання візуальної інформації. Наведені нижче закони та нормативні акти не регламентують безпосередньо комп’ютерний зір, але створюють правове поле, у якому можуть функціонувати відповідні технології.

Першим і найважливішим є Закон України «Про захист персональних даних» № 2297-VI від 01.06.2010. Згідно з цим законом, зображення, на яких можна ідентифікувати особу, розглядаються як персональні дані. Якщо система комп’ютерного зору працює з фото або відео, де фігурують люди, вона автоматично підпадає під регуляцію цього закону. Використання таких даних вимагає отримання згоди суб’єкта даних або наявності іншої законної підстави.

Додатково, важливим є Закон України «Про інформацію» № 2657-XII від 02.10.1992, який встановлює загальні принципи збирання, обробки, зберігання та поширення інформації. У контексті систем комп’ютерного зору, які обробляють зображення або відео, цей закон регулює допустимі

способи використання отриманої інформації, у тому числі – її доступність та поширення.

Значну роль відіграє також Закон України «Про доступ до публічної інформації» № 2939-VI від 13.01.2011. Він може бути релевантним, якщо система комп’ютерного зору використовується державними органами або в публічному секторі, де доступ до результатів її роботи може бути предметом публічного запиту. Цей закон забезпечує прозорість дій держави та регламентує порядок надання інформації громадянам.

Щодо прав інтелектуальної власності, застосовується Закон України «Про авторське право і суміжні права» № 3792-XII від 23.12.1993. Цей документ захищає авторські права на алгоритми, програмне забезпечення, а також на вхідні дані (наприклад, зображення або відео), якщо вони є об’єктами авторського права. Це особливо важливо при використанні зображень третіх осіб для навчання моделей.

Також важливим є Закон України «Про електронні комунікації» № 1089-IX від 16.12.2020, який визначає порядок обробки інформації в телекомунікаційних системах. Якщо система комп’ютерного зору працює в режимі онлайн або обробляє потоки даних з відеокамер у реальному часі, вона може підпадати під вимоги цього закону щодо безпеки інформаційної інфраструктури.

У контексті штучного інтелекту доречним є посилання на Концепцію розвитку штучного інтелекту в Україні, затверджену Кабінетом Міністрів України у 2021 році. Хоча це не є законом, концепція окреслює основні напрями державної політики щодо впровадження систем ШІ, включаючи принципи етичності, прозорості та підзвітності. Це важливо для розробників систем виявлення накладених об’єктів, оскільки подібні системи належать до сфери штучного інтелекту.

Ще одним дотичним документом є Закон України «Про наукову і науково-технічну діяльність» № 848-VIII від 26.11.2015. Якщо система розробляється в межах дослідження чи експерименту, наприклад у рамках

дипломної роботи, діяльність підпадає під дію цього закону. Він визначає правові засади проведення досліджень, у тому числі правила доступу до даних, співпраці з іншими установами та етичних норм у дослідженні.

У випадках, коли обробка зображень здійснюється у медичній сфері, застосовується Закон України «Основи законодавства України про охорону здоров'я» № 2801-XII від 19.11.1992, який захищає конфіденційність медичної інформації. Якщо система аналізує медичні зображення, необхідно дотримуватись підвищених вимог до анонімності та безпеки даних.

Якщо система використовується для відеоспостереження або безпеки, актуальним є Закон України «Про охоронну діяльність» № 4616-VI від 22.03.2012, який встановлює вимоги до встановлення відеоспостереження, обробки записів і доступу до них. Система, яка визначає об’єкти на зображенні з камер, зокрема в публічних місцях, має працювати відповідно до норм цього закону.

У випадках участі в міжнародних дослідженнях або використання закордонного програмного забезпечення, можуть також застосовуватися вимоги Загального регламенту ЄС про захист даних (GDPR), особливо якщо система має справу з даними громадян ЄС. Хоча Україна не є членом ЄС, дотримання положень GDPR часто є вимогою в академічних або комерційних проектах з міжнародною участю.

Також доцільно враховувати Кримінальний кодекс України, зокрема статті, що стосуються порушення недоторканності приватного життя (ст. 182) та незаконного збору конфіденційної інформації. Система, що працює з візуальними даними, повинна забезпечити, щоб обробка не порушувала права громадян на приватність.

Таким чином, діяльність, пов’язана з розробкою та впровадженням систем визначення накладених об’єктів, підпадає під регуляцію одразу кількох галузей права. Розробники таких систем повинні враховувати вимоги до захисту персональних даних, безпеки інформації, авторських

прав, етичності алгоритмів штучного інтелекту та доступу до візуальної інформації. Дотримання чинного законодавства є обов’язковим етапом при створенні та впровадженні подібних технологій у практику.

1.4 Визначення основних методів комп’ютерного зору виділення об’єктів на зображенні

У комп’ютерному зорі існує кілька основних способів виділення об'єктів на зображенні, кожен з яких застосовується залежно від складності сцени, поставленої задачі та рівня точності, що очікується. До таких способів належать: класична сегментація, виявлення об’єктів через bounding boxes, інстанс-сегментація, семантична сегментація, edge detection, а також підходи на основі глибокого навчання.

Найпростішим способом є детектування контурів (edge detection). Цей підхід ґрунтується на виявленні різких змін яскравості у пікселях зображення, що часто відповідає границям об’єктів. Алгоритми типу Canny, Sobel або Laplacian дають змогу побудувати карту контурів. Проте такі методи чутливі до шуму та освітлення й не дозволяють визначити, який саме об’єкт виявлено.

Порогова сегментація (thresholding) – ще один класичний метод, що передбачає поділ зображення на області за значенням інтенсивності. Простий варіант – глобальний поріг, де всі пікселі, яскравість яких перевищує певне значення, вважаються частиною об'єкта. Існують також адаптивні методи, що враховують локальні характеристики, однак такі підходи погано працюють у випадках з перекриттям або складною текстурою.

Більш сучасний і точний метод – виявлення об’єктів через bounding boxes. У цьому випадку кожному об’єкту призначається прямокутна рамка, яка його оточує. Такі алгоритми, як YOLO (You Only Look Once), SSD (Single Shot Detector) [8], та Faster R-CNN [9], дозволяють швидко і

точно локалізувати об'єкти на зображеннях. Проте, якщо об’єкти накладаються один на одного, прямокутники можуть частково перекриватися, що знижує точність виявлення.

Ще точніший підхід – семантична сегментація, яка передбачає класифікацію кожного пікселя зображення до певного класу. Наприклад, всі пікселі, що належать до об’єкта «автомобіль», будуть помічені одним кольором. Такий підхід добре підходить для аналізу сцен, однак не відрізняє окремі екземпляри одного класу. Це створює труднощі у випадках, коли потрібно розрізнити два автомобілі, які накладаються один на одного.

Щоб розв'язати цю проблему, застосовується інстанс-сегментація – метод, який поєднує семантичну сегментацію з виявленням окремих об'єктів. У цьому підході кожен об’єкт не лише класифікується, але й виділяється окрема маска пікселів для кожного екземпляра. Алгоритми типу RT-DETR, DINO, а також сучасні варіанти YOLOv8 з підтримкою сегментації забезпечують високу точність навіть у разі складного перекриття.

Існують також підходи на основі глибокого навчання без нагляду (unsupervised segmentation), які використовують кластеризацію ознак для виділення об'єктів без попередньої розмітки. Вони можуть давати змогу виявляти нові або невідомі об'єкти, але поступаються за точністю методам з учителем.

Ще один перспективний напрям – Vision Transformers [10] та інші архітектури з self-attention, які краще враховують глобальні зв'язки у зображенні. Вони поступово витісняють класичні CNN-моделі в задачах сегментації, особливо коли потрібно враховувати взаємодію між об’єктами та їх просторове розташування.

Таким чином, у комп’ютерному зорі існує широкий спектр підходів до виділення об’єктів, і вибір конкретного залежить від вимог до точності, швидкості, наявності перекриття та складності сцени. Для завдань

виявлення накладання найбільш релевантними є інстанс-сегментація та сучасні гібридні підходи на основі глибоких нейронних мереж

Для дослідження методів визначення накладених об’єктів найкраще підходять ті підходи комп’ютерного зору, які здатні не лише виявити факт присутності об’єктів, а й точно локалізувати кожен з них у межах зони перекриття. Це потребує високого рівня просторової деталізації, контекстного аналізу та розділення пікселів між об'єктами. Нижче подано обґрунтований вибір найбільш релевантних методів.

Перш за все, інстанс-сегментація є ключовим підходом для завдань з накладанням. На відміну від простого детектування чи семантичної сегментації, вона дозволяє не тільки класифікувати пікселі, але й розділити їх між окремими екземплярами об’єктів одного класу. Це особливо важливо у випадках, коли, наприклад, два однакові об’єкти (дві людини, дві машини) частково перекривають один одного, і потрібно точно визначити їх межі.

До найефективніших сучасних моделей для задач виявлення та сегментації об'єктів належить Grounding DINO – архітектура, що поєднує потужність трансформерів із концепцією zero-shot детекції. Модель розширює можливості традиційних детекторів, дозволяючи локалізувати об'єкти на основі вільно сформульованого текстового запиту без потреби у фіксованому наборі класів. Grounding DINO демонструє високу стійкість до складних сцен з великою кількістю накладень та різноманітністю об'єктів, забезпечуючи точне виділення релевантних областей на зображенні.

Альтернативою є сучасні моделі типу Grounding DINO, які реалізують zero-shot детекцію об’єктів із використанням текстових запитів та трансформерної обробки ознак. Вони поєднують переваги глибокого контекстного аналізу з гнучкістю мовного інтерфейсу, що дозволяє працювати з відкритим словником класів без потреби у повторному навчанні. Такий підхід особливо ефективний для сцен зі складним

семантичним змістом та перекриттям, де класичні моделі виявляють обмеження у здатності до узагальнення

Ще одним підходом, що заслуговує уваги, є Vision Transformers (ViT) у комбінації з сегментаційними головами. Такі моделі здатні ефективно враховувати глобальний контекст зображення завдяки механізму самоуваги (self-attention), що є особливо корисним у випадках, коли об’єкти перекривають один одного, і їхню форму потрібно реконструювати на основі контекстних ознак.

Для аналізу об’єктів у режимі реального часу варто розглядати також YOLOv8 Seg – новітню версію відомого YOLO, яка включає компонент сегментації. Попри орієнтацію на швидкість, вона демонструє гідну точність у виділенні об’єктів і може бути використана для первинного виявлення накладених елементів, особливо у відеоаналітиці.

Крім моделей, важливим є використання відповідних метрик якості, таких як IoU (Intersection over Union) і AP (Average Precision), зокрема AP@0.5 та AP@0.75, які дозволяють оцінити, наскільки точно маска охоплює об’єкт у складних зонах.

Таким чином, для дослідження проблеми накладання об'єктів доцільним є застосування трансформерних моделей нового покоління, таких як Grounding DINO, які реалізують zero-shot підхід до локалізації та підтримують обробку відкритих наборів класів на основі текстових запитів. Ці моделі відзначаються високою точністю виявлення навіть у складних візуальних сценах із численними перекриттями та відсутністю чітко визначеного класового простору. Grounding DINO забезпечує гнучке узагальнення на нові категорії об'єктів, що робить його особливо придатним для досліджень, де важлива адаптивність до змін у структурі даних.

У ході аналізу методів комп’ютерного зору для виявлення накладених об'єктів було з'ясовано, що найбільш перспективними підходами є моделі, які поєднують семантичне розуміння з

високорівневою контекстуалізацією. Grounding DINO відзначається глибокою інтеграцією візуальної та текстової інформації, що дозволяє точно локалізувати об'єкти за описом навіть у разі їхнього часткового або повного перекриття іншими об'єктами. Така властивість особливо цінна для задач, де класи об'єктів не є заздалегідь відомими або строго фіксованими.

Моделі, побудовані на базі трансформерної архітектури, зокрема Grounding DINO, показали високий потенціал у задачах, що вимагають гнучкої генералізації та здатності до роботи в умовах відкритого світу. На відміну від класичних підходів, які покладаються на фіксовану категоризацію, трансформерні моделі можуть динамічно адаптуватися до нових запитів, що забезпечує не лише локалізацію, а й концептуальне узгодження між текстовим описом та зображенням. Це робить їх надзвичайно актуальними у контексті дослідження накладення об’єктів, де важливим є не лише точне сегментування, а й розуміння суті об'єктів у сцені.

Аналогічно, модель Grounding DINO, що представляє сучасний підхід до zero-shot виявлення об’єктів, є придатною для проведення експериментів і порівняльного аналізу в умовах складних сцен. Вона підтримує гнучке формулювання запитів у вигляді природної мови та дозволяє тестувати ефективність детекції без фіксованого набору класів. Завдяки трансформерній архітектурі та здатності враховувати глобальний контекст, Grounding DINO добре адаптується до задач із високим рівнем накладання об’єктів і змінною семантикою сцени.

Новітні моделі, такі як YOLOv8 з підтримкою сегментації, хоч і поступаються в точності класичним інстанс-сегментаторам, демонструють високу швидкодію, що робить їх корисними у задачах реального часу. Для деяких сценаріїв із частковим накладанням та вимогами до продуктивності такі моделі можуть бути оптимальним компромісом між точністю і швидкістю.

Особливу увагу слід приділити моделям на базі трансформерів, які завдяки механізмам самоуваги здатні моделювати складні залежності між об'єктами на зображенні. У задачах накладання ці моделі можуть переосмислювати контекст сцени, що дозволяє точніше відновлювати форми об’єктів навіть при значному перекритті. Поява Vision Transformers відкриває нові можливості для поліпшення якості сегментації в умовах складної геометрії та текстури.

Узагальнюючи, можна зазначити, що методи, орієнтовані на роботу з піксельними масками, є найпридатнішими для дослідження накладання, оскільки лише вони здатні дати вичерпну інформацію про форму, межі й взаємне розміщення об’єктів. Bounding boxes або просте детектування не забезпечують достатньої точності в умовах значного перекриття.

Важливим фактором є також наявність якісних анотованих датасетів із накладанням. Саме такі набори даних дозволяють адекватно навчити моделі і створити умови для об’єктивного порівняння їхньої ефективності. Зокрема, набори типу COCO [11] або Cityscapes [12] містять приклади складних сцен, де накладання об’єктів зустрічається природним чином.

Таким чином, обрані для дослідження методи охоплюють як точні, ресурсомісткі архітектури для глибокого аналізу накладених об'єктів, так і легші моделі для швидкої обробки сцен у реальному часі. Це дозволяє провести порівняльний аналіз різних підходів не лише з точки зору точності, але й з урахуванням обчислювальних витрат.

У контексті поставленої задачі реалізація експериментальної частини на основі інстанс-сегментації із використанням моделей Grounding DINO, RT-DETR та YOLOv8 Seg є доцільною та обґрунтованою. Застосування цих архітектур дозволяє всебічно дослідити проблему виявлення накладених об’єктів, охоплюючи як трансформерні zero-shot підходи, так і згорткові та однопрохідні рішення. Це дає змогу комплексно оцінити якість роботи моделей у складних візуальних умовах, проаналізувати їх сильні та слабкі сторони й зробити обґрунтовані висновки щодо

доцільності впровадження таких рішень у практичні та наукові застосування.

1.5 Визначення основних метрик оцінювання методів

Вибір відповідних метрик оцінки є ключовим етапом у дослідженні методів визначення накладених об’єктів на зображеннях. Коректно підібрані метрики дозволяють кількісно порівняти ефективність різних моделей, виявити їхні слабкі сторони та об’єктивно визначити, який підхід працює краще в умовах часткового або повного перекриття об’єктів. Оскільки йдеться про завдання інстанс-сегментації, де необхідно точно виділяти межі кожного об’єкта, основна увага приділяється метрикам, здатним оцінити точність масок, їх перекриття з еталонними розмітками, а також здатність моделі розрізняти різні об’єкти одного класу.

Однією з базових і найбільш поширених метрик є IoU (Intersection over Union). Вона обчислюється як відношення площі перетину передбаченої маски об’єкта з еталонною до площі їх об’єднання. IoU є прямим показником того, наскільки добре модель «вгадала» місцезнаходження і форму об’єкта. Для накладених об’єктів ця метрика особливо важлива, адже навіть незначне зміщення межі може призвести до суттєвого падіння значення IoU, що відображає втрату точності в умовах перекриття.

Проте сам по собі показник IoU не дає повної картини, якщо не враховувати точність і повноту виявлення об’єктів. Для цього використовуються Average Precision (AP) та Average Recall (AR) – узагальнені метрики, що базуються на аналізі множини IoU для багатьох об'єктів. AP вимірює точність виявлення при різних порогах IoU. Найчастіше використовуються AP@0.5, AP@0.75 та середній показник AP@[0.5:0.95], який усереднює результати для різних рівнів перекриття.

Такий підхід дозволяє оцінити як здатність моделі знаходити об’єкти, так і точність їх локалізації.

AP@0.5 (IoU ≥ 0.5) є достатньо лояльною метрикою, що допускає неідеальне перекриття передбаченої маски з еталоном. Вона добре відображає загальну здатність моделі розпізнавати об’єкти. Натомість AP@0.75 (IoU ≥ 0.75) є суворішим критерієм, що вимагає точнішої локалізації. Це особливо важливо у контексті накладених об’єктів, де будь-яка помилка у межах розділення сутностей впливає на якість маски.

Також варто розглядати метрику Panoptic Quality (PQ) – сучасний підхід для оцінки одночасно семантичної та інстанс-сегментації. Вона інтегрує в собі і точність, і повноту, а також враховує здатність моделі правильно класифікувати об’єкт і виділити його як окрему інстанцію. PQ складається з двох компонентів: segmentation quality (SQ) та recognition quality (RQ), що дозволяє більш глибоко проаналізувати природу помилок. Але найбільшість датасетів не підтримую можливість використання даної метрики, так як фокусуються на одній певній задачі - або object detection, або object segmentation, тому даних не достатньо.

У задачах з великою кількістю накладених об’єктів також доцільно використовувати False Positive Rate (FPR) та False Negative Rate (FNR), які відображають частку неправильних спрацювань або пропущених об’єктів. Особливо це важливо для систем, де критично важливо не пропустити жоден об’єкт, наприклад, у безпекових системах або медичній діагностиці.

Ще одним напрямом є оцінка усереднених метрик по класах – mAP (mean Average Precision). Якщо в сцені присутні об’єкти різних класів, важливо аналізувати, як модель справляється з кожним з них, адже накладання може по-різному впливати на об’єкти залежно від їх розміру, форми чи текстури.

Таким чином, для оцінки ефективності методів визначення накладених об’єктів доцільно застосовувати комбінацію кількох метрик. Основними будуть: IoU, AP@0.5, AP@0.75, mAP, F1. Після детального

аналізу доступних наборів даних для застосування метрики PQ, тому надалі вони не буде використовуватися. Такий комплексний підхід дозволить дати повну характеристику поведінки моделей у задачах з перекриттям і забезпечити об’єктивне порівняння різних алгоритмів. Обрані метрики охоплюють як піксельну точність, так і загальну здатність моделі до розрізнення об’єктів у складних сценах. Це створює надійну основу для експериментальної частини дослідження.

Для проведення дослідження методів визначення накладених об’єктів на зображеннях було обрано п’ять основних метрик, які забезпечують найповніше охоплення якості сегментації, точності локалізації та здатності моделей розрізняти об’єкти в умовах перекриття. Кожна з метрик відіграє важливу роль у комплексній оцінці роботи алгоритмів інстанс-сегментації, зокрема в аспекті роботи з масками та взаємодії між об’єктами.

Першою ключовою метрикою є Intersection over Union (IoU), яка відображає рівень перекриття між передбаченою і еталонною масками. IoU є стандартом у комп’ютерному зорі для оцінки точності локалізації, і в задачі накладання вона дозволяє кількісно виміряти, наскільки точно модель змогла відтворити форму кожного об’єкта. У контексті цього дослідження IoU виступає базовим інструментом для аналізу якості сегментації в межах окремих об'єктів.

Другою метрикою є Average Precision at IoU ≥ 0.5 (AP@0.5) – показник, що фіксує, наскільки добре модель знаходить об’єкти при відносно м’якому порозі відповідності. AP@0.5 є індикатором загальної здатності моделі виявляти об'єкти, навіть якщо їх межі відтворено не ідеально. Ця метрика важлива для розуміння, скільки накладених об’єктів взагалі було розпізнано.

Третю метрику становить Average Precision at IoU ≥ 0.75 (AP@0.75) – більш суворий варіант, що потребує високої точності при локалізації. Його застосування дає змогу оцінити, наскільки добре модель справляється з

деталізацією меж об’єктів у складних зонах накладання. Поєднання AP@0.5 і AP@0.75 дозволяє побудувати уявлення як про загальну ефективність, так і про здатність до точного відокремлення об’єктів.

Четвертою обраною метрикою є mean Average Precision (mAP), яка усереднює значення AP по всіх класах і кількох порогах IoU (зазвичай від 0.5 до 0.95 з кроком 0.05). mAP є найбільш загальною характеристикою якості інстанс-сегментації, оскільки вона враховує широкий діапазон сценаріїв і забезпечує всебічну оцінку моделі. У випадку накладання mAP дозволяє відстежувати стабільність поведінки моделі на різних рівнях перекриття.

П’ятою метрикою, що була обрана, є F1-score – узагальнений показник, який враховує як точність (precision), так і повноту (recall) моделі. F1-score особливо корисний у задачах об’єктного виявлення, де важливо не лише виявити якомога більше істинних об’єктів (висока recall), але й уникати хибних спрацювань (висока precision). Ця метрика є інтуїтивно зрозумілою та дозволяє отримати балансовану оцінку якості класифікації для кожного класу або всієї моделі в цілому.

Сукупне використання цих п’яти метрик забезпечує баланс між оцінкою точності сегментації, здатністю до виявлення об’єктів, коректною класифікацією та розділенням накладених сутностей. Такий підхід дозволить максимально об’єктивно оцінити переваги та обмеження досліджуваних моделей у контексті задачі, що розглядається

1.6 Постановка задачі

У межах виконання роботи необхідно сформулювати та вирішити комплекс задач, спрямованих на дослідження можливостей сучасних методів комп’ютерного зору для виявлення об’єктів на зображеннях у ситуаціях, коли ці об’єкти частково або повністю перекривають одне одного. Для досягнення мети потрібно створити структуровану модель

дослідження, що включає вибір архітектур, побудову експерименту, реалізацію відповідного програмного середовища та аналіз результатів за допомогою обраних метрик.

Першочерговим завданням є аналіз предметної області, що включає вивчення базових понять комп'ютерного зору, накладання об'єктів, цифрового зображення та інстанс-сегментації. Необхідно сформувати чітке розуміння термінології, принципів функціонування відповідних алгоритмів і актуальності теми в контексті сучасних задач штучного інтелекту.

Наступним етапом є дослідження та порівняння існуючих підходів до виділення об'єктів на зображеннях. Доцільно проаналізувати класичні методи сегментації, моделі на основі bounding boxes, семантичну та інстанс-сегментацію, а також сучасні архітектури, що реалізують зазначені підходи. Особливу увагу слід приділити методам, здатним опрацьовувати зони перекриття об'єктів.

У ході роботи необхідно обґрунтовано обрати набір моделей для практичного порівняння. Зокрема, до експериментального дослідження планується включити Grounding DINO, YOLOv8 Seg та RT-DETR як провідні реалізації інстанс-сегментації, що охоплюють різні підходи до аналізу зображень. Ці моделі поєднують здатність до точного визначення меж об'єктів зі стійкістю до перекриттів, підтримкою мультимодальних запитів або високою продуктивністю, що відповідає вимогам задачі виявлення складних візуальних конфігурацій.

Також необхідно визначити систему метрик, за якими оцінюватиметься ефективність моделей. До таких метрик буде віднесено IoU, AP@0.5, AP@0.75, mAP та Panoptic Quality. Кожна з них дозволяє аналізувати точність розпізнавання, якість сегментації, здатність моделі розділяти інстанції та виявляти складні патерни перекриття.

Окремим завданням є побудова та налаштування експериментального середовища. Це передбачає підбір або підготовку набору даних з наявністю накладених об'єктів, налаштування архітектур

моделей, запуск процесів навчання та тестування, а також збір результатів для подальшого аналізу.

У процесі реалізації експериментів потрібно провести порівняльний аналіз точності, стабільності та продуктивності обраних моделей. Результати мають бути задокументовані, представлені у вигляді графіків, таблиць або візуалізацій, що дозволяють оцінити реальні можливості кожного підходу в умовах накладання.

Не менш важливою задачею є врахування нормативно-правових обмежень при використанні зображень, що можуть містити персональні дані, або програмного забезпечення, яке підпадає під авторське право. Потрібно дослідити та вказати вимоги законодавства України, дотримання яких є обов’язковим під час роботи із системами комп’ютерного зору.

Завершальною задачею є формування висновків щодо ефективності кожного з підходів та визначення перспектив їх практичного застосування. Отримані результати повинні дозволити сформулювати рекомендації щодо вибору методів виявлення накладених об’єктів для різних сценаріїв реального використання.

2 ФОРМУВАННЯ ТА ПРОЄКТУВАННЯ ЕКСПЕРЕМЕНТУ ДОСЛІДЖЕННЯ

2.1 Вибір моделей та архітектур для дослідження

У рамках дослідження, спрямованого на визначення накладених об’єктів на зображеннях, особливе значення має етап вибору моделей та архітектур, які будуть застосовані в експериментальній частині роботи. Цей вибір має бути обґрунтований з урахуванням специфіки задачі, рівня складності оброблюваних зображень, характеру перекриттів, а також технічних можливостей реалізації. У контексті дослідження, що стосується інстанс-сегментації, доцільно зосередитися на моделях, здатних не лише локалізувати об’єкти, а й створювати точні піксельні маски кожної інстанції, навіть у випадках часткового чи повного перекриття.

Зважаючи на сучасний стан розвитку комп’ютерного зору, особливу увагу було приділено архітектурам, які продемонстрували високу ефективність у задачах інстанс-сегментації на реальних зображеннях. Йдеться, насамперед, про глибокі згорткові нейронні мережі, які інтегрують в собі кілька рівнів аналізу – від базової локалізації до формування точних масок об’єктів. Для забезпечення достовірності результатів було прийнято рішення розглянути кілька архітектур, що відрізняються як за підходом до обробки зображень, так і за рівнем обчислювальної складності.

Урахуванням ключовим критерієм при виборі моделей слугувала здатність алгоритму працювати із зонами, де відбувається накладання кількох об’єктів, не втрачаючи при цьому точності сегментації. Також до уваги бралися такі фактори, як підтримка сучасних фреймворків, наявність відкритих реалізацій, репутація моделі у наукових публікаціях, а також гнучкість налаштування параметрів під конкретні експериментальні потреби. Подальші етапи дослідження базуються саме на практичній

перевірці обраних архітектур у контексті реальних зображень із перекриттям об’єктів.

У процесі формування експериментальної частини дослідження було розглянуто низку сучасних методів комп’ютерного зору, які демонструють здатність до виявлення та сегментації об’єктів у складних сценах, зокрема у випадках їх часткового перекриття. Основна увага зосереджувалась на підходах, що реалізують інстанс-сегментацію, оскільки саме цей тип задачі передбачає не лише класифікацію пікселів, а й розділення кожного окремого екземпляра об’єкта, навіть у межах одного класу. Це є критично важливим у контексті накладання, коли декілька подібних об’єктів можуть мати спільну межу або частково затуляти один одного.

Серед розглянутих методів одне з провідних місць займає архітектура Grounding DINO, яка є розвитком детекторів на основі трансформерів і поєднує в собі можливості візуального аналізу та мовного опису. Цей підхід реалізує zero-shot виявлення об’єктів шляхом інтеграції текстових запитів безпосередньо в процес передбачення, що дозволяє локалізувати об'єкти за довільним описом без попереднього навчання на фіксованому наборі класів. Grounding DINO виконує детекцію через вбудований механізм attention, що забезпечує одночасну регресію координат рамки та семантичне зіставлення із заданим запитом. Завдяки використанню глобального контексту трансформера модель демонструє високу здатність до узагальнення та ефективно справляється із задачами виявлення об'єктів у складних умовах, включаючи сцени з численними перекриттями.

Окрім того, увагу було приділено платформі Grounding DINO – відкритому дослідницькому фреймворку, що базується на трансформерній архітектурі та реалізується за допомогою бібліотек Hugging Face та PyTorch. Grounding DINO підтримує zero-shot детекцію об'єктів на основі текстових описів, що забезпечує унікальну можливість гнучкої адаптації до змінного словника класів без потреби у повторному навчанні. Завдяки

підтримці широкого спектра текстових запитів, моделі Grounding DINO можуть використовуватися як у вузькоспеціалізованих, так і в універсальних задачах виявлення. Архітектура вирізняється модульністю, що дозволяє налаштовувати вхідні параметри, обирати рівень абстракції запитів і інтегрувати модель у більш складні мультимодальні пайплайни. Це робить Grounding DINO придатним інструментом як для прикладних завдань, так і для наукового порівняльного аналізу.

Також до розгляду було включено модель YOLOv8 з підтримкою сегментації (YOLOv8 Seg), яка є останньою ітерацією швидкодіючого детектора об’єктів. Незважаючи на свою основну орієнтацію на реальний час, ця модель отримала додаткову функціональність у вигляді побудови сегментаційних масок. Зважаючи на її продуктивність, YOLOv8 є цінним прикладом компромісу між точністю і швидкодією, що дозволяє оцінити ефективність моделі в сценаріях, де важливим є мінімізація затримки.

Додатково було проаналізовано підходи, що базуються на сучасних багатомодальних трансформерних архітектурах, зокрема моделі типу Grounding DINO, які поєднують візуальну інформацію з текстовим описом. Завдяки використанню механізмів cross-attention між вхідним зображенням та текстовим запитом, ці моделі забезпечують точну локалізацію об’єктів навіть у відкритому класовому середовищі. Незважаючи на високу ефективність і здатність до генералізації, використання Grounding DINO потребує суттєвих обчислювальних ресурсів, особливо при обробці зображень високої роздільності або великої кількості текстових запитів. У зв’язку з цим модель була розглянута на теоретичному рівні як представник перспективного класу zero-shot підходів, без залучення до основної практичної частини експерименту.

Усі методи, що розглядалися, були оцінені за критеріями точності, здатності працювати з перекриттям, вимог до ресурсів та адаптивності до стандартних датасетів. Це дозволило сформувати обґрунтований вибір

архітектур, які далі були використані для моделювання і порівняння в експериментальному середовищі.

Для реалізації експериментальної частини дослідження було обрано три методи інстанс-сегментації, які продемонстрували високу ефективність у задачах виявлення об’єктів із перекриттям та отримали широку підтримку в сучасних наукових і прикладних розробках. Такий підхід дозволяє провести репрезентативне порівняння моделей, що суттєво відрізняються за принципами функціонування, архітектурною структурою, точністю та обчислювальною ефективністю. До складу експерименту були включені Grounding DINO як приклад трансформерної zero-shot архітектури, RT-DETR як оптимізована трансформерна модель для сегментації, а також YOLOv8 Seg – високопродуктивна однопрохідна модель, орієнтована на реальний час.

Першою трансформерною моделлю, залученою до дослідження, стала Grounding DINO. Вона є прикладом сучасного підходу до zero-shot обробки зображень, який поєднує текстову і візуальну інформацію для точного виявлення об'єктів без необхідності фіксованого словника класів. Архітектура Grounding DINO базується на механізмах attention усередині візуального енкодера та між зображенням і текстом, що дозволяє моделі виконувати узгодження між семантикою запиту та візуальним змістом сцени. Цей підхід особливо ефективний для задач із відкритим класовим простором або нестандартними категоріями об'єктів. Завдяки своїй гнучкості, узагальнювальній здатності та високій точності локалізації, Grounding DINO є потужним інструментом для аналізу зображень зі складним контекстом, включаючи сцени з численними накладеннями, де класичні моделі можуть втрачати точність через обмежений словник класів.

Другою обраною технологією є Grounding DINO – модель, розроблена дослідницькою групою IDEA-Research, яка реалізує сучасний підхід до zero-shot детекції за допомогою трансформерної архітектури. Її ключова перевага полягає у здатності поєднувати зображення з текстовими

описами, що дозволяє здійснювати локалізацію об’єктів за довільними мовними запитами без обмеження фіксованим набором класів. Grounding DINO підтримує інтеграцію через бібліотеки PyTorch і Hugging Face, що забезпечує зручність у підключенні до сучасних пайплайнів обробки зображень. Завдяки відкритому коду, гнучкій структурі та підтримці inference у zero-shot режимі, ця технологія є надзвичайно придатною для експериментів, що вимагають високого рівня узагальнення, адаптивності та мультимодальної взаємодії.

Третім методом було обрано YOLOv8 Seg – нову модифікацію популярної лінійки моделей YOLO, що включає функціональність сегментації. Ця модель має менші обчислювальні вимоги, високу швидкодію та орієнтована на реальне застосування в умовах обмежених ресурсів або у задачах, де критичною є швидкість реакції системи. Попри меншу точність у порівнянні з DINO, YOLOv8 демонструє достатню якість масок навіть у зонах часткового перекриття, що робить її цінним контрастом у порівняльному аналізі.

Загалом обрані три методи дозволяють охопити широкий спектр підходів до інстанс-сегментації: від класичних та високоточних моделей до швидких, легковагих рішень. Такий підхід створює умови для об’єктивної оцінки сильних і слабких сторін кожної з архітектур у контексті задачі. Виявлення накладених об’єктів на зображеннях є складним завданням комп’ютерного зору, яке потребує точного виділення меж об’єктів навіть у випадках часткового або повного перекриття. Для обґрунтування вибору моделей, що використовуються у дослідженні, доцільно здійснити їх порівняльний аналіз за ключовими параметрами. Такий аналіз дозволяє зіставити особливості архітектур, виявити їх сильні та слабкі сторони, а також визначити умови, за яких кожна з моделей досягає найвищої ефективності. У таблиці 2.1 наведено стислу характеристику трьох обраних методів інстанс-сегментації: Grounding DINO, RT-DETR та YOLOv8 Seg. Основна увага зосереджена на структурі моделей, їхніх

перевагах, обмеженнях і потенційних сферах застосування, що дозволяє сформувати об’єктивну основу для подальшого експериментального порівняння.

Таблиця 2.1 – Порівняльний аналіз методів комп’ютерного зору

Модель	Тип архітектури	Переваги	Недоліки	Сценарії використання
DINO	одноступенева з інтеграцією текстових запитів (text-conditioned detection)	гнучкість у класах; підтримка zero-shot; ефективна локалізація	ввисокі обчислювальні вимоги; потреба в якісних текстових запитах	open-world детекція, мультимодальні системи, пошук за візуальним описом
RT-DETR	модель інстанс-сегментації на основі трансформерної архітектури, оптимізована для швидкої та точної обробки зображень	точне виділення об’єктів без потреби в постобробці;	обмежена підтримка мультимодальних запитів у порівнянні з zero-shot підходами застосування в системах реального часу	індустріальні та безпекові сценарії з високими вимогами до продуктивності
YOLOv8 Seg	одноступенева з інтеграцією сегментації	висока швидкодія; оптимізація під реальний час; невеликі вимоги до ресурсів	нижча точність масок; обмежена деталізація в зонах перекриття	мобільні пристрої, відео в реальному часі, інтеграція у промислові системи

Згідно з наведеними у таблиці даними, модель Grounding DINO демонструє інший архітектурний підхід, ґрунтуючись на одноступеневій трансформерній структурі з інтеграцією текстових запитів. Замість

використання заздалегідь визначених класів, модель оперує довільними текстовими описами, які співвідносяться з візуальними ознаками зображення. Це забезпечує гнучкість у застосуванні до відкритих або нестандартних доменів, дозволяє працювати в zero-shot режимі та зменшує залежність від обмежених словників. Архітектура Grounding DINO забезпечує точну локалізацію навіть в умовах щільного перекриття, однак потребує потужних обчислювальних ресурсів, особливо при опрацюванні складних сцен із великою кількістю потенційних відповідників до тексту.

RT-DETR, у свою чергу, є трансформерною архітектурою, яка використовується для задач інстанс-сегментації завдяки поєднанню високої точності та швидкості обробки. Вона реалізує однопрохідну обробку зображень, що усуває потребу в додатковому постпроцесингу та дозволяє досягти стабільного виділення об’єктів навіть за умови їх часткового перекриття. Архітектура моделі оптимізована для реального часу, що забезпечує її ефективну інтеграцію в мобільні, вбудовані та індустріальні системи, де важлива мінімальна затримка. Незважаючи на обмежену підтримку мультимодальної взаємодії або zero-shot сценаріїв, RT-DETR відзначається високою продуктивністю в умовах обмежених ресурсів, що робить її конкурентоспроможною у прикладних застосуваннях інстанс-сегментації.

Модель YOLOv8 Seg представляє одноступеневий підхід до детектування та сегментації, що передбачає виконання всіх операцій в межах єдиного проходу через мережу. Завдяки цьому вона досягає дуже високої швидкодії, що робить її придатною для роботи в реальному часі, зокрема на мобільних або вбудованих пристроях. Водночас, через спрощену архітектуру, точність побудови масок дещо нижча, особливо у випадках із щільним перекриттям об’єктів.

У частині переваг Grounding DINO вирізняється здатністю працювати в умовах відкритого словника класів, що робить її надзвичайно придатною для задач, де склад об’єктів наперед невідомий або змінюється

залежно від контексту. Модель ефективно виконує локалізацію об’єктів за вільно сформульованими текстовими описами, що відкриває широкі можливості для застосування в мультимодальних системах, пошукових інтерфейсах та інтерактивних середовищах. Завдяки інтеграції трансформерного механізму уваги Grounding DINO також забезпечує стійкість до перекриття та здатність враховувати глобальний контекст, що особливо важливо при аналізі складних сцен.

YOLOv8 Seg, хоч і поступається за точністю, компенсує це мінімальними вимогами до апаратного забезпечення та надзвичайно швидкою обробкою кадрів. Завдяки цьому модель може бути використана у промислових системах, які потребують миттєвої реакції, або в системах відеоаналітики для фільтрації та попереднього виявлення об’єктів.

Щодо недоліків, Grounding DINO, попри свою гнучкість і здатність до zero-shot локалізації, вирізняється високими обчислювальними вимогами. Повноцінне використання моделі потребує значної кількості відеопам’яті, особливо при роботі з великими зображеннями або численними текстовими запитами. Крім того, ефективність детекції значною мірою залежить від якості сформульованих описів, що створює додаткове навантаження на користувача в аспекті генерації запитів. У деяких випадках неточно сформульований текст може призвести до пропуску релевантних об’єктів або хибнопозитивних результатів.

Таким чином, вибір Grounding DINO є обґрунтованим у випадках, коли необхідна висока гнучкість до відкритого словника класів або інтеграція з текстовими запитами, що особливо актуально для мультимодальних додатків та zero-shot сценаріїв. Якщо ж основною метою є досягнення максимальної точності сегментації з точним окресленням контурів об'єктів у складних сценах, ефективнішим буде використання трансформерних моделей, таких як RT-DETR, що поєднують високу точність з обробкою в реальному часі. У випадках, коли критичне значення має продуктивність та простота інтеграції у готові робочі середовища,

оптимальним рішенням залишається YOLOv8 Seg. У підсумку, кожен з підходів демонструє унікальні переваги, і їхнє комбіноване використання відкриває нові можливості для точного, масштабованого та адаптивного аналізу зображень із перекриттями в умовах реального застосування.

2.2 Вибір та підготовка наборів даних

Одним із ключових етапів експериментального дослідження є вибір і підготовка наборів даних, які використовуються для навчання, валідації та тестування моделей комп'ютерного зору. Якість, структура та зміст таких даних безпосередньо впливають на здатність моделей до генералізації, точності розпізнавання об'єктів і, особливо важливо в контексті цієї роботи, на ефективність виявлення накладених об'єктів. Тому підбір відповідного датасету є не лише технічним, а й методологічним кроком, що визначає надійність подальших результатів.

Під час відбору наборів зображень основним критерієм була наявність значної кількості прикладів із частковим або повним перекриттям об'єктів у сцені, а також доступність інстанс-розмітки – тобто точних піксельних масок для кожного окремого екземпляра об'єкта. Це дозволяє перевірити здатність моделей правильно сегментувати об'єкти в складних умовах та мінімізувати помилки злиття або втрати сутностей.

Окрім цього, при виборі датасетів враховувались такі характеристики, як різноманітність класів, варіативність умов зйомки, складність сцен, підтримка стандартних форматів (COCO, Pascal VOC) і наявність супровідної документації. Особливу увагу було приділено тому, щоб дані були адаптовані для інстанс-сегментації, а не лише для класифікації або bounding-box виявлення.

У процесі підготовки експериментального дослідження важливою складовою стало вивчення доступних наборів даних, які можуть бути використані для навчання моделей інстанс-сегментації в умовах

накладання об’єктів. Оскільки метою дослідження є не просто виявлення об’єктів, а точне розділення кожної інстанції, особливу увагу було приділено датасетам, що містять детальні маски та різні варіанти перекриття у складних сценах.

Нижче подано перелік наборів даних, які були розглянуті як потенційні джерела вхідної інформації для навчання та тестування моделей. Для кожного з них наведено короткий опис, що охоплює тип розмітки, кількість зображень, прикладні галузі, специфіку структури, а також доцільність використання у задачах із перекриттям об’єктів:

– COCO (Common Objects in Context), великий і популярний датасет, що містить понад 200 000 зображень та більше 80 класів об’єктів з інстанс-розміткою; охоплює складні сцени з багатьма об’єктами, серед яких часто трапляються перекриття;

– CITYSCAPES, орієнтований на аналіз міських сцен для задач автономного водіння; містить розмітку об’єктів (транспорт, пішоходи) з високою точністю масок, включаючи численні приклади перекриттів на вулицях;

– PASCAL VOC, один із перших великих датасетів для виявлення об’єктів, містить порівняно обмежений набір класів і об’єктів; підтримує сегментацію, проте не завжди містить точні інстанс-маски, що обмежує його у випадках накладання;

– OPENIMAGES V6, масштабний датасет з понад 9 мільйонами зображень і більше ніж 600 класами; має як bounding boxes, так і часткову підтримку інстанс-масок, проте якість анотацій варіюється, що потребує додаткової фільтрації даних;

– BDD100k, орієнтований на дорожні сцени, містить розмітку транспортних засобів, пішоходів, дорожніх знаків; підтримує як bounding boxes, так і сегментацію, у тому числі з перекриттями об’єктів на перехрестях;

– LVIS, що фактично є розширення COCO, яке містить ще більше класів (понад 1200) з детальними масками; орієнтоване на покриття рідкісних класів, що дозволяє тестувати здатність моделей виявляти малопредставлені об’єкти з перекриттям;

– MAPILLARY VISTAS, високоякісний датасет вуличних сцен із сегментаційною розміткою, що включає детальні маски транспортних засобів, людей, інфраструктури; містить сцени з високим ступенем перекриття;

– Instance segmentation benchmark (KINS), набір з високоточною розміткою для інстанс-сегментації, що базується на KITTI; містить дані з автомобілів, що дозволяє моделювати типові ситуації із накладанням у русі;

– ADE20k, універсальний датасет для сценоорієнтованої сегментації з великою кількістю класів; розмітка охоплює сцени всередині приміщень і на відкритому просторі, проте більшість зображень фокусуються на семантичній, а не інстанс-сегментації;

– Synthetics for segmentation, набори даних, створені штучно (з використанням рушіїв Unreal або Unity), дозволяють генерувати сцени з контрольованим ступенем перекриття; придатні для моделювання специфічних випадків, які важко знайти у реальних знімках.

З наведеного списку видно, що лише частина датасетів забезпечує повноцінну підтримку інстанс-сегментації із масками, придатними для задач із перекриттям. Найбільш відповідними у цьому контексті є COCO, Cityscapes, LVIS та Mapillary Vistas, оскільки вони містять велику кількість сцен з природним накладанням об’єктів та мають перевірену розмітку високої якості.

Інші набори, такі як PASCAL VOC чи OpenImages, хоча й містять цінні приклади, вимагають додаткової обробки або не мають достатньої деталізації в зонах перекриття. Штучно згенеровані дані, у свою чергу,

мають перевагу в контрольованості, але можуть поступатися в реалістичності.

Таким чином, для проведення достовірного експерименту рекомендовано обирати набори, які містять точні інстанс-маски, велику кількість зображень і достатню варіативність сцен з перекриттям, що дозволяє найбільш повноцінно оцінити поведінку моделей інстанс-сегментації у складних умовах.

Для реалізації експериментальної частини дослідження було обрано датасет COCO (Common Objects in Context). Вибір саме цього набору даних зумовлений його відповідністю основним вимогам, що висуваються до задачі виявлення накладених об’єктів за допомогою методів інстанс-сегментації. COCO є одним із найбільш поширених та широко підтримуваних датасетів у сфері комп’ютерного зору, що робить його оптимальним вибором як для практичної реалізації, так і для порівняння результатів із іншими дослідженнями.

Датасет COCO містить понад 200 000 зображень, серед яких понад 80 класів об’єктів мають розмітку не лише у вигляді bounding boxes, а й у вигляді точних інстанс-масок. Особливу цінність для цього дослідження становлять сцени з великою кількістю об’єктів у кадрі та частим накладанням – як між об’єктами одного класу (наприклад, декілька людей), так і різних класів (наприклад, пішохід перед транспортом). Крім того, дані в COCO є реальними фотографіями, що забезпечує високу ступінь варіативності, реалістичності та відповідності до прикладних сценаріїв.

Важливою перевагою COCO є також його підтримка у багатьох сучасних бібліотеках глибокого навчання, зокрема Grounding DINO, EfficientDet, Ultralytics YOLO, які містять вбудовані засоби для завантаження, парсингу та попередньої обробки даних цього формату. Це істотно спрощує інтеграцію датасету в навчальні пайплайни, дозволяє уникнути трудомісткої ручної підготовки анотацій та забезпечує уніфікованість форматів при порівнянні результатів. Таким чином,

використання COCO як базового набору даних створює передумови для ефективної організації експериментів і концентрації зусиль безпосередньо на аналізі якості моделей.

Для цілей дослідження було використано підмножину COCO під назвою COCO 2017, яка містить окремі частини для навчання (train2017), валідації (val2017) та тестування. Завантаження датасету здійснювалося з офіційного репозиторію COCO. Окремо були завантажені зображення та відповідні файли анотацій у форматі JSON (annotations_trainval2017.zip).

Завдяки уніфікованій структурі файлів та підтримці формату COCO, набір даних було безпосередньо інтегровано у середовище моделей Grounding DINO, RT-DETR Segmentation та Ultralytics YOLOv8. Це дозволило швидко перейти до навчання, забезпечити коректне зчитування анотацій і гарантувати відповідність між вхідними зображеннями та їх масками на всіх етапах тренування й оцінювання.

Датасет складається з декількох тисяч фото, приклад яких навдеено на рисунку 2.1.

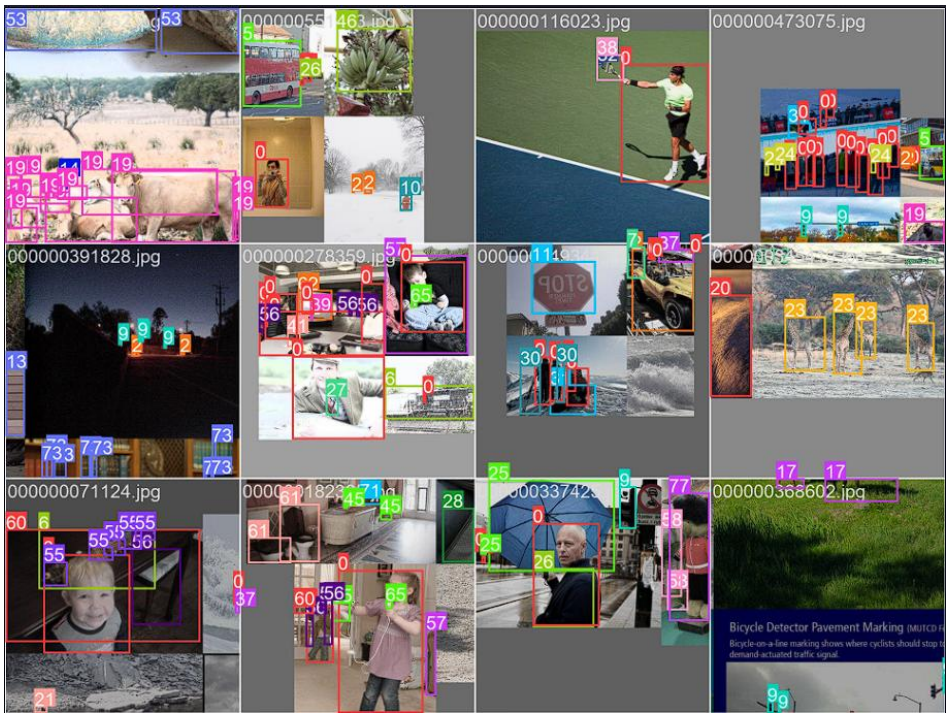

Рисунок 2.1 – Приклад зображень з тренувального датасету COCO

Зображення, представлені на прикладі з датасету COCO 2017, демонструють високу різноманітність сцен, об’єктів та ситуацій, що створює реалістичне середовище для навчання моделей інстанс-сегментації. Кожне зображення має багатокласну розмітку, де об’єкти різних типів позначені кольоровими прямокутниками або масками з відповідними числовими ідентифікаторами класів.

Серед прикладів можна побачити сцени з транспортом, людьми, тваринами, побутовими об’єктами, знаками, меблями, продуктами харчування, одягом тощо. Наприклад, на одному з фрагментів зображено міський автобус, який перекриває дерево, а поруч стоїть людина – і всі об'єкти мають свої власні анотації. В інших прикладах присутні зображення диких тварин, які йдуть групою, що створює природне перекриття одного об’єкта іншим – типовий випадок для тестування інстанс-сегментації.

У сценах з людьми можна побачити приклади різних положень тіла, одягу, взаємодії з предметами, а також ситуації з високим ступенем накладання – наприклад, групове фото або сцена в приміщенні з великою кількістю об’єктів. Такі зображення дають змогу перевірити, чи здатна модель коректно відокремлювати подібні або частково закриті об’єкти.

Також представлені приклади нічних знімків, кадрів з високим контрастом, вуличних знаків, інтер’єрів і дійсно складних композицій, де декілька об’єктів одного класу перекриваються або частково видимі. Це створює додаткове навантаження на алгоритми, змушуючи їх навчатися розділяти навіть ті елементи, які частково приховані або мають спільні межі.

Таким чином, приклади з COCO 2017 є показовими для задач інстанс-сегментації, зокрема – у випадках накладання об’єктів. Завдяки великій варіативності сцен, різним умовам освітлення, глибокій структурі анотацій та присутності багатьох класів, цей датасет є одним із найкращих

варіантів для навчання, тестування та порівняльного аналізу моделей у межах дослідження.

2.3 Опис середовища та інструментів реалізації експерименту

Проведення експериментального дослідження з виявлення накладених об’єктів на зображеннях потребує ретельно підібраного програмного середовища та інструментів, які забезпечують ефективну реалізацію моделей інстанс-сегментації, обробку великих обсягів зображень і обчислення метрик якості. У цьому підрозділі описано компоненти середовища, що були використані на всіх етапах реалізації: від підготовки даних і налаштування архітектур до запуску навчання моделей та аналізу результатів.

Вибір програмного забезпечення здійснювався з урахуванням таких критеріїв, як сумісність із обраними моделями (DINO, RT-DETR, YOLOv8 Seg), наявність документації, підтримка апаратного прискорення (GPU), зручність візуалізації результатів та можливість гнучкої інтеграції з набором даних COCO. Особливу увагу було приділено стабільності середовища, можливості масштабування експериментів, а також ефективному обслуговуванню обчислювальних процесів під час тренування та валідації моделей.

Для розгортання моделей та реалізації всіх етапів експериментального дослідження було використано середовище розробки Visual Studio Code (VSCode). Це кросплатформенне інтегроване середовище програмування, яке забезпечує гнучкість, розширюваність і зручність у роботі з Python-кодом, бібліотеками глибокого навчання та інструментами керування проєктом. Його використання дало змогу централізовано керувати структурою проєкту, швидко перемикатися між компонентами системи, запускати експерименти та інтегрувати зовнішні фреймворки через термінал або розширення.

VSCode було обрано завдяки підтримці інтегрованих інструментів для роботи з Git, Docker, Jupyter Notebook, а також завдяки можливості налаштування віртуального середовища для ізоляції залежностей кожної з моделей. Зокрема, використано розширення Python, що забезпечує підсвічування синтаксису, автодоповнення, відладку коду, форматування, інтеграцію з середовищем виконання та управлінням пакетами. Через вбудований термінал у VSCode здійснювалося встановлення бібліотек, запуск скриптів навчання, підрахунок метрик і тестування продуктивності моделей.

Для реалізації та розгортання моделей комп'ютерного зору у межах дослідження було використано мову програмування Python, яка на сьогодні є провідним стандартом у галузі машинного навчання та глибокого навчання. Python забезпечує гнучкий синтаксис, велику кількість бібліотек для роботи з даними, потужну екосистему для побудови та тренування нейронних мереж, а також активну спільноту розробників і дослідників.

Основною перевагою Python у цьому контексті є його сумісність із сучасними фреймворками глибокого навчання, зокрема PyTorch, TensorFlow та Ultralytics, що дозволяють швидко імплементувати складні архітектури інстанс-сегментації, такі як DINO або YOLOv8 Seg. Крім того, Python має зручні інтерфейси для взаємодії з датасетами, обробки зображень (через бібліотеки OpenCV, PIL, Albumentations) та обчислення метрик якості (scikit-learn, pycocotools).

Усі експериментальні скрипти були реалізовані саме на Python, включаючи процес завантаження зображень, підготовки датасету COCO, конфігурацію моделей, тренування, валідацію та підрахунок показників точності. Завдяки широкій підтримці GPU через CUDA та PyTorch, Python також забезпечив ефективне використання обчислювальних ресурсів під час навчання моделей на великих наборах зображень.

Окрім Python і VSCode, важливим елементом середовища реалізації стало використання фреймворку PyTorch, який забезпечує зручну й гнучку платформу для побудови, навчання та тестування нейронних мереж. PyTorch було обрано як базову бібліотеку завдяки її динамічному обчислювальному графу, зручності в налагодженні, високій швидкодії при роботі на GPU, а також завдяки активній підтримці з боку спільноти та інтеграції з ключовими архітектурами сегментації. Усі моделі, що використовувалися в дослідженні, були реалізовані саме на базі PyTorch або з використанням фреймворків, побудованих поверх нього.

Для запуску моделі Grounding DINO було використано офіційну реалізацію від проєкту IDEA-Research, яка розповсюджується через платформу Hugging Face. Модель представлена у вигляді попередньо натренованої трансформерної архітектури для zero-shot детекції об’єктів за текстовими запитами. Вона базується на поєднанні візуального енкодера та мовного енкодера, що забезпечує узгодження між зображенням і запитом за допомогою механізмів attention.

Для інтеграції було використано Python API бібліотеки transformers, зокрема компоненти AutoProcessor та AutoModelForZeroShotObjectDetection, що дозволяють зручно обробляти зображення й текстові інструкції в єдиному пайплайні. Тестування моделі проводилося з використанням зображень із відкритих датасетів (COCO, Open Images), а текстові запити формувались вручну, з урахуванням цільових об’єктів. Результати детекції були оброблені за допомогою функції post_process_grounded_object_detection, яка формує фінальні координати виявлених об'єктів відповідно до вхідних запитів.

Модель YOLOv8 Seg реалізовано за допомогою бібліотеки Ultralytics, яка містить повністю підготовлену інфраструктуру для тренування моделей YOLO будь-якої версії, включаючи функціональність сегментації. Платформа підтримує зручне завантаження COCO-формату, автоматичну генерацію логів, візуалізацію результатів та підрахунок стандартних

метрик AP, IoU тощо. Для цього в середовищі було використано бібліотеку ultralytics, яку інтегровано у Python-проєкт через pip.

Також для попередньої обробки зображень, трансформацій і аугментації даних було використано бібліотеки Albumentations та OpenCV. Вони дозволили здійснити вирівнювання розмірів, нормалізацію, масштабування, обертання та інші перетворення, необхідні для покращення узагальнюючої здатності моделей при тренуванні.

Для запуску навчання з підтримкою апаратного прискорення було використано GPU-платформу Google Colab Pro, яка надала доступ до графічного процесора NVIDIA Tesla T4. Це дозволило суттєво скоротити час навчання моделей без потреби у локальному високопродуктивному обладнанні. Усі експерименти були синхронізовані з Google Drive для збереження результатів та конфігурацій.

Під час експериментів використовувалися засоби логування та візуалізації, зокрема Matplotlib, Seaborn, TensorBoard, а також вбудовані інструменти фреймворків Grounding DINO, RT-DETR та Ultralytics YOLOv8. Це забезпечило можливість не лише відслідковувати динаміку зміни ключових метрик протягом тренування, а й візуально аналізувати результати інстанс-сегментації, порівнюючи передбачені маски з еталонними. Такий підхід дозволив оперативно виявляти потенційні помилки, аналізувати якість локалізації та сегментації в складних сценах і здійснювати налаштування моделей на основі візуального зворотного зв’язку.

Таким чином, середовище реалізації експерименту було сформовано як модульна, гнучка та масштабована інфраструктура, що поєднує стабільність локального середовища (через VSCode і Python), обчислювальні можливості хмарних платформ (Google Colab) та потужні бібліотеки для реалізації й аналізу моделей інстанс-сегментації. Такий підхід дозволив ефективно реалізувати повний цикл дослідження – від підготовки даних до отримання репрезентативних результатів.

2.4 Визначення параметрів навчання моделей

У процесі реалізації експерименту важливим етапом стало визначення параметрів навчання кожної з обраних моделей інстанс-сегментації. Від налаштувань гіперпараметрів значною мірою залежить ефективність навчання, швидкість збіжності, точність прогнозування масок об’єктів, а також здатність моделей справлятися із ситуаціями накладання. Враховуючи специфіку архітектур (DINO, RT-DETR, YOLOv8 Seg) та характеристики датасету COCO 2017, параметри були підібрані емпірично з урахуванням практики використання цих моделей у суміжних дослідженнях.

Для всіх моделей було встановлено розмір вхідного зображення в межах 640×640 або 800×800 пікселів, залежно від рекомендацій фреймворку. Такий розмір дозволяє зберегти баланс між деталізацією масок та швидкістю обробки. Більші розміри призводять до перевантаження GPU, тоді як менші – знижують точність виділення дрібних об’єктів, особливо в умовах перекриття.

Кількість епох навчання було встановлено на рівні 100 для всіх моделей, із можливістю дострокової зупинки за умов стабілізації метрик валідації. Для оптимізації використовувалися адаптивні алгоритми: для RT-DETR – SGD із моментумом 0.9, для YOLOv8 Seg – AdamW, а для Grounding DINO – Adam із динамічним регулюванням коефіцієнта навчання. Початкове значення learning rate становило 0.001 для YOLOv8 Seg і Grounding DINO та 0.005 для RT-DETR, з подальшим зниженням згідно з політиками StepLR або CosineAnnealing.

Параметр batch size було підібрано відповідно до доступного обсягу GPU-пам’яті. Для Grounding DINO та RT-DETR оптимальним значенням стало 4 зображення на ітерацію, тоді як для YOLOv8 Seg використовувалось 8 зображень завдяки її оптимізованій структурі. Усі зображення попередньо нормалізовувалися відповідно до статистичних

характеристик датасету COCO: середнє значення [0.485, 0.456, 0.406] і стандартне відхилення [0.229, 0.224, 0.225], що забезпечувало коректну узгодженість вхідних даних із попередньо натренованими вагами моделей.

Для забезпечення стійкості до варіацій вхідних даних було застосовано техніки аугментації – випадкове горизонтальне віддзеркалення, зміну яскравості, масштабування, обертання та випадковий crop. Це особливо важливо в задачі з накладеними об’єктами, де зміна точки зору або положення камери може ускладнювати сегментацію.

Також було налаштовано логування та збереження ваг моделей з найкращими метриками на валідаційному наборі. Для цього використовувалися callback-механізми, реалізовані у відповідних фреймворках. Крім того, встановлювався контроль над метрикою IoU або mAP: збереження відбувалося лише у випадку покращення значення на наступній епосі.

Усі параметри навчання були задокументовані у відповідних YAML-конфігураціях або Python-скриптах, що гарантує відтворюваність результатів експерименту. Після завершення навчання параметри найкращих моделей використовувалися для наступного етапу – оцінювання точності та аналізу результатів сегментації на прикладах із перекриттям.

Після завершення первинного налаштування гіперпараметрів окрему увагу було приділено вибору scheduler-ів навчання, які визначають траєкторію зміни коефіцієнта навчання протягом епох. Для YOLOv8 Seg використовувався алгоритм CosineAnnealingLR, що реалізує поступове зниження learning rate за косинусоїдальною кривою, сприяючи стабільному зниженню помилки на пізніх етапах навчання. У випадку Grounding DINO була застосована стратегія StepLR, яка передбачає регулярне зменшення коефіцієнта навчання через фіксовану кількість епох, що дозволяє стабілізувати процес оновлення ваг після початкового швидкого навчання. Для RT-DETR використовувався підхід WarmupMultiStepLR, що

передбачає коротку фазу поступового збільшення learning rate («розігрів»), після чого відбувається поетапне зниження на заздалегідь заданих межах.

Важливою складовою навчального процесу стало збереження контрольних точок (checkpoints), що давало змогу зберігати найкращі ваги моделі відповідно до метрики mean Average Precision (mAP), обчисленої на валідаційному наборі. Це дозволяло не лише уникнути ефекту перенавчання, а й забезпечувало можливість продовження навчання у випадку зовнішнього переривання. У випадку RT-DETR і YOLOv8 Seg відповідні механізми збереження реалізовані у фреймворках автоматично. Для Grounding DINO, що базується на PyTorch, збереження стану моделі здійснювалося через окремо реалізовані обробники (torch.save / state_dict), що дозволяло повністю контролювати процес збереження та відновлення навчання.

Особливу увагу було приділено процедурі перевірки якості навчання за допомогою періодичного оцінювання на валідаційному наборі. Інтервал між перевірками становив 1 епоху, а в межах кожної перевірки проводився підрахунок IoU, mAP@0.5 та mAP@0.5:0.95. Це дозволило не лише спостерігати загальну динаміку поліпшення результатів, а й оцінити здатність моделі до адекватного виявлення перекритих об’єктів, особливо у випадках багатокомпонентних сцен.

Для прискорення обчислень та ефективного використання апаратних ресурсів було реалізовано підтримку паралельного оброблення батчів на GPU з використанням технологій CUDA та PyTorch DataLoader. Додатково здійснювалася перевірка наявності витоків пам’яті, неправильного обнулення градієнтів або нестабільного зростання втрат (loss explosion), що могло свідчити про некоректну конфігурацію або проблеми в аугментації.

Усі параметри, включаючи значення втрат, точність, швидкість обробки кадрів, розмір масок і співвідношення істинно позитивних, хибно позитивних і хибно негативних виявлень, реєструвалися у вигляді логів та

графіків. Це забезпечувало повну прозорість процесу навчання та дозволяло проводити детальний постаналіз після завершення кожної серії експериментів.

Таким чином, параметри навчання були підібрані з урахуванням як архітектурних особливостей кожної моделі, так і конкретних вимог задачі – виявлення накладених об’єктів із високим рівнем деталізації та точності. Такий підхід дозволив створити стабільне та відтворюване середовище для подальшого оцінювання ефективності кожної архітектури в межах дослідження.

У процесі визначення параметрів навчання було сформовано повноцінну систему гіперпараметричної конфігурації, адаптовану під архітектурні особливості кожної з обраних моделей інстанс-сегментації – Grounding DINO, RT-DETR та YOLOv8 Seg. Обрані параметри дозволили досягти компромісу між обчислювальною ефективністю, точністю виявлення об’єктів і здатністю моделей до узагальнення на нових прикладах. Особливу увагу було приділено таким аспектам, як розмір вхідного зображення, кількість епох, тип оптимізатора, початкове значення коефіцієнта навчання, стратегія зміни learning rate (scheduler), а також методи аугментації, які безпосередньо впливають на стійкість моделі до варіативності вхідних даних. Завдяки цьому кожна модель була налаштована з урахуванням її внутрішньої структури та обчислювальних потреб, що забезпечило коректне порівняння їх ефективності в умовах експерименту.

Встановлення спільних параметрів, таких як кількість епох, метрики оцінки, формат даних та структура логування, забезпечило уніфіковану основу для порівняння моделей у рівнозначних умовах. Це дозволило зробити оцінювання результатів об’єктивним і придатним до аналізу з точки зору продуктивності, точності та здатності до розпізнавання накладених об’єктів у складних сценах.

Також важливо, що всі параметри були задокументовані та реалізовані у вигляді окремих конфігураційних файлів або інтегровані в тренувальні скрипти, що гарантує повну відтворюваність експерименту. Застосування контрольних точок, логування та механізмів збереження найкращих ваг забезпечило надійність і стабільність під час навчання.

Підбір аугментаційних стратегій також відіграв важливу роль у підвищенні стійкості моделей до змін позицій, масштабів і умов освітлення – що є критично важливим для задач із накладеними об’єктами. Це дало змогу тренувати архітектури, здатні не лише до точного локалізованого розпізнавання, але й до коректної обробки складних, багатокомпонентних зображень.

Таким чином, налаштування параметрів навчання було виконано на належному рівні технічної точності, що заклало міцну основу для подальшого етапу – порівняльної оцінки моделей за обраними метриками. Отримані результати уможливлюють не лише аналіз продуктивності кожної архітектури, але й узагальнення висновків щодо доцільності їх застосування в задачах виявлення накладених об’єктів.

2.5 Вибір метрик для оцінювання результатів експерименту

У межах експериментального дослідження вибір метрик оцінювання відіграє ключову роль, оскільки саме ці показники дозволяють об’єктивно проаналізувати ефективність кожної моделі в задачі виявлення накладених об’єктів. Враховуючи специфіку інстанс-сегментації та наявність складних сцен із перекриттями, доцільно використовувати комплексну систему оцінювання, яка враховує як точність локалізації, так і якість розпізнавання об’єктів на рівні масок. З цією метою було обрано метрики, що широко використовуються в сучасних дослідженнях з комп’ютерного зору та рекомендовані для задач, пов’язаних з аналізом зображень у форматі COCO.

Основною метрикою оцінювання результатів є mean Average Precision (mAP), яка вимірює середню точність моделі при різних порогах IoU. Цей показник дає змогу оцінити не лише факт наявності розпізнаного об’єкта, а й точність його просторової локалізації. Зокрема, mAP@0.5 (mean Average Precision при IoU ≥ 0.5) дозволяє визначити загальну здатність моделі правильно виявляти об’єкти, тоді як mAP@[0.5:0.95] з кроком 0.05 забезпечує більш жорстку оцінку, що є стандартом у COCO-челенджах.

Другим важливим показником є Intersection over Union (IoU), що вимірює площу перетину між передбаченою маскою об’єкта та еталонною (ground truth) у відношенні до площі їх об’єднання. Ця метрика є надзвичайно важливою при аналізі накладених об’єктів, оскільки дозволяє оцінити точність визначення меж кожного з них навіть у разі їхнього часткового перекриття.

Додатково використовувалися метрики Precision і Recall, які характеризують відповідно здатність моделі не видавати хибно позитивні результати та не пропускати об’єкти. Для задачі сегментації з перекриттям особливо важливо підтримувати баланс між цими двома показниками, що в сукупності формує основу для обчислення метрики F1-score – гармонійного середнього між Precision і Recall. Високий F1-score вказує на те, що модель не лише коректно виявляє об’єкти, а й робить це стабільно по відношенню до усіх класів.

Також було доцільно включити Average Recall (AR), який вимірює середній рівень повноти виявлення об’єктів при різній кількості пропозицій (detections). Цей показник є важливим для аналізу складних сцен, де присутні численні об’єкти, що перекриваються або частково затулені іншими.

Крім того, у межах дослідження розглядалася можливість аналізу якісних характеристик за допомогою візуальних метрик, таких як Boundary F1-score (BF Score), яка оцінює точність меж маски в порівнянні з

істинною формою об’єкта. Це особливо актуально в ситуаціях, коли об’єкти накладаються один на одного та мають складні контури.

Таким чином, обрані метрики дозволяють отримати як кількісну, так і візуально-структурну оцінку ефективності моделей у завданні виявлення накладених об’єктів. Їх застосування забезпечить об’єктивне порівняння архітектур, а також дозволить виявити сильні й слабкі сторони кожного підходу в умовах, максимально наближених до реальних.

Для забезпечення об’єктивної та комплексної оцінки результатів експериментального дослідження з виявлення накладених об’єктів на зображеннях було обрано п’ять основних метрик, які найкраще характеризують якість роботи моделей інстанс-сегментації. Під час відбору метрик враховувалися як технічні рекомендації COCO-челенджів, так і практичні аспекти задач, пов’язаних із точністю, повнотою та геометричною відповідністю масок. Кожна з наведених метрик охоплює окремий аспект ефективності моделі: від здатності правильно локалізувати об’єкти до загальної стабільності при складному перекритті контурів.

Обрані показники є загальновизнаними в спільноті комп’ютерного зору та широко використовуються в академічних дослідженнях, що забезпечує об’єктивність і порівнюваність отриманих результатів із відомими підходами. Крім того, ці метрики мають вбудовану підтримку у відповідних фреймворках, таких як Grounding DINO, PyTorch, EfficientDet та Ultralytics YOLOv8, що значно полегшує їх інтеграцію в експериментальне середовище й автоматизує процес обчислення під час навчання та тестування моделей:

– mAP@0.5. Середнє значення точності при пороговому значенні перетину (IoU) не менше 0.5, що дозволяє оцінити здатність моделі виявляти об’єкти навіть із помірною точністю локалізації;

– mAP@[0.5:0.95]. Усереднений показник точності для дев’яти порогових значень IoU (від 0.5 до 0.95 з кроком 0.05), що забезпечує суворі умови для всебічного аналізу якості сегментації;

– IoU (Intersection over Union). Метрика, що вимірює частку перетину між передбаченою маскою та еталонною щодо їх об’єднаної площі, дозволяє оцінити точність геометричної відповідності;

– F1-score. Гармонійне середнє між precision і recall, яке характеризує загальну збалансованість моделі у виявленні об’єктів без переважання хибно позитивних чи негативних результатів;

– Average Recall (AR). Метрика, що визначає середній рівень повноти розпізнавання об’єктів при різній кількості пропозицій, що важливо при обробці багатокомпонентних сцен із перекриттям.

Вибір саме цих метрик дозволяє отримати повну картину поведінки моделей у реальних сценаріях, де присутні об’єкти з неоднозначними межами, частковим перекриттям і варіативними формами. Наприклад, у разі, коли модель виявляє об’єкт із точною формою, але з незначним зсувом, це буде враховано через значення IoU; якщо ж модель стабільно знаходить усі об’єкти, але з недостатньою деталізацією – це проявиться в mAP@[0.5:0.95] та AR.

Таким чином, використання вказаного набору метрик дозволяє не лише кількісно оцінити результати роботи кожної архітектури, а й виявити особливості її поведінки в умовах складної інстанс-сегментації. У наступних розділах ці метрики будуть застосовані до результатів експериментальних досліджень, що дасть змогу обґрунтовано порівняти ефективність Grounding DINO, RT-DETR та YOLOv8 Seg. Такий підхід забезпечує комплексний аналіз не лише з точки зору загальної точності, але й з урахуванням здатності моделей коректно обробляти сцени з накладеними об’єктами, змінними класами та варіативною складністю зображень.

Для зручності, метрики наведено в таблиці 2.2. Наведена таблиця узагальнює ключові характеристики метрик, які були обрані для оцінки ефективності моделей у задачі виявлення накладених об’єктів на зображеннях.

Таблиця 2.2 – Метрики оцінювання методів

Назва метрики	Повна назва	Опис метрики	Призначення в експерименті
mAP@0.5	Mean Average Precision at IoU = 0.5	Усереднене значення точності по всіх класах при пороговому значенні IoU ≥ 0.5.	Оцінка здатності моделі виявляти об’єкти з базовою точністю локалізації.
mAP@[0.5:0.95]	Mean Average Precision at IoU = 0.5:0.95	Суворіша метрика, що враховує середнє значення точності при порогах IoU від 0.5 до 0.95 з кроком 0.05.	Визначення загальної точності локалізації та відповідності форм в умовах різного ступеня накладання.
IoU	Intersection over Union	Частка площі перетину передбаченої маски з істинною щодо площі їх об’єднання.	Вимірювання геометричної точності сегментації та локалізації накладених контурів.
F1-score	Harmonic mean of Precision and Recall	Гармонійне середнє між precision і recall, що балансує між помилками першого та другого роду.	Визначення стабільності моделі щодо якості виявлення об’єктів у складних сценах.
AR	Average Recall	Середнє значення повноти при фіксованій кількості детекцій, оцінює, наскільки модель охоплює всі наявні об’єкти.	Визначення рівня здатності моделі не пропускати об’єкти в кадрах з високою щільністю та частковим перекриттям.

Кожна метрика розглядається з чотирьох точок зору: скороченої назви, повної назви, загального опису та її практичного призначення в межах експерименту. Такий підхід дозволяє чітко зрозуміти, яку саме властивість поведінки моделі оцінює кожен з показників.

Перші дві метрики mAP@0.5 та mAP@[0.5:0.95] – є стандартними показниками точності в COCO-челенджах і широко застосовуються в академічній спільноті. Вони демонструють, наскільки модель здатна точно локалізувати об’єкти різного розміру та ступеня складності. Перша з них застосовується для базової перевірки точності, а друга – для всебічної оцінки поведінки моделі на різних рівнях точності локалізації.

Третя метрика IoU – є фундаментальною для задач сегментації, оскільки враховує не лише факт виявлення об’єкта, а й співвідношення між передбаченою та істинною масками. У задачі з накладенням об’єктів IoU є критичним, оскільки дозволяє оцінити, наскільки добре модель відтворює межі об’єктів у випадках їх перекриття.

Четверта метрика F1-score – дозволяє знайти баланс між точністю та повнотою виявлення, особливо в ситуаціях, коли модель може пропускати або надлишково виділяти об’єкти. Завдяки цьому вона відображає реальну здатність моделі знаходити саме ті об’єкти, які дійсно присутні в сцені, без зайвих або втрачених сегментів.

Остання з розглянутих метрик Average Recall (AR) – оцінює, скільки об’єктів було знайдено при фіксованій кількості пропозицій, і є корисною в умовах багатокомпонентних сцен. Її використання дозволяє зрозуміти, наскільки ефективно модель сканує сцену в пошуках усіх потенційно важливих об’єктів.

Загалом, таблиця дає можливість порівняти функціональне навантаження кожної метрики й обґрунтувати їхню необхідність у контексті експерименту. Це дозволяє перейти до етапу практичної оцінки моделей з чітким розумінням того, які саме аспекти будуть вимірюватися та аналізуватися.

2.6 Формалізація цілей і сценаріїв експерименту

У межах другого розділу дослідження формалізація цілей та сценаріїв експерименту є ключовим кроком для забезпечення відтворюваності, обґрунтованості та цілеспрямованості всіх подальших етапів. Оскільки дослідження зосереджене на виявленні накладених об’єктів на зображеннях, головна мета експерименту полягає в порівняльному аналізі обраних моделей інстанс-сегментації за критеріями точності, стабільності та здатності до генералізації в умовах часткового або повного перекриття об’єктів.

Для досягнення цієї мети необхідно забезпечити чітке формулювання експериментальних завдань, що включають реалізацію моделей Grounding DINO, YOLOv8 Seg і RT-DETR, їх навчання на стандартизованому наборі даних із наявністю перекритих об’єктів (наприклад, COCO або його адаптованій підмножині), а також подальше тестування на окремому наборі з аналогічними характеристиками. Такий підхід дозволяє забезпечити об'єктивне порівняння результатів за ідентичних умов, мінімізуючи вплив сторонніх факторів на якість оцінки. Це, у свою чергу, дає змогу виділити сильні та слабкі сторони кожної моделі у контексті завдань інстанс-сегментації з частковим або повним перекриттям об'єктів.

Формалізація цілей передбачає деталізацію того, що саме має бути перевірено: які аспекти поведінки моделей потребують аналізу, на яких прикладах і за яких умов. Зокрема, необхідно дослідити точність локалізації об’єктів у складних сценах, ефективність обробки масок з високим ступенем перекриття, а також узагальнюваність моделей на нових прикладах, які не входили до навчального набору.

Усі сценарії експерименту повинні охоплювати повний цикл обробки: від підготовки зображень до виводу підсумкових метрик. Для кожної моделі буде проведено три незалежні серії експериментів з однаковими

конфігураціями (кількість епох, batch size, scheduler, аугментації), що дозволить забезпечити статистичну надійність отриманих результатів. Сценарії повинні враховувати не лише базові сцени, а й зображення з інтенсивним накладанням, неоднорідним фоном та зміною масштабу об’єктів.

Крім того, для формалізації сценаріїв експерименту буде визначено шаблони оцінки на валідаційному наборі, що включають підрахунок обраних метрик (IoU, mAP, AR тощо) на кожному етапі. Це забезпечить можливість побудови графіків динаміки показників, що дозволить виявити як загальні закономірності навчання, так і критичні точки нестабільності.

Формалізовані сценарії мають бути описані таким чином, щоб будь-який дослідник міг повторити їх з використанням аналогічного коду, конфігурацій і наборів даних. Це дозволить перевірити коректність висновків, оцінити надійність результатів і адаптувати підхід для інших завдань у галузі комп’ютерного зору.

У цілому, формалізація цілей та сценаріїв експерименту виступає гарантом системності та наукової обґрунтованості дослідження. Саме завдяки чіткому опису методів, параметрів і очікуваних результатів можна забезпечити об’єктивне порівняння ефективності сучасних моделей інстанс-сегментації в умовах складних зображень із накладеними об’єктами.

Продовжуючи формалізацію цілей та сценаріїв експерименту, необхідно уточнити, що порівняльне дослідження трьох моделей інстанс-сегментації охоплює не лише підрахунок кінцевих метрик, а й аналіз стабільності результатів у динаміці, а також якісне оцінювання отриманих масок на прикладах з високим рівнем складності сцени. У фокусі аналізу перебувають ситуації, де об’єкти перекривають один одного частково або повністю, змінюють форму, мають схожі текстури або розміщені на візуально складному фоні.

Кожен сценарій експерименту передбачає детальний контроль змінних – зокрема, використання однакової кількості тренувальних і тестових прикладів, фіксованих випадкових seed-значень для генерації аугментацій, стабільного порядку подачі зображень у батчах. Таке структурування дозволяє досягти відтворюваності та запобігти випадковим флуктуаціям у результатах. Окрім цього, окремі сценарії включають модифікації – наприклад, навчання на урізаних підмножинах або за відсутності аугментацій – для виявлення впливу окремих факторів на поведінку моделей.

Для кожної моделі результат її роботи буде представлено у вигляді:

– кількісного звіту з метриками (mAP, AR, IoU, F1-score);

– графіків динаміки навчання (втрата, точність, повнота);

– прикладів сегментованих зображень зі складними умовами перекриття;

– візуалізацій помилок класифікації (false positives/false negatives).

Це дозволить не лише кількісно оцінити продуктивність, а й здійснити візуальний та якісний аналіз розподілу помилок.

Одним з обов’язкових сценаріїв є перевірка здатності моделей працювати на змінених умовах – таких як зменшена роздільна здатність вхідного зображення або варіація кількості об’єктів у сцені. Це дозволить визначити рівень генералізації кожної моделі та її стійкість до змін у даних. Такий підхід є надзвичайно важливим для практичного застосування подібних систем у реальному середовищі, де ідеальних умов не існує.

У підсумку, формалізація цілей і сценаріїв експерименту забезпечує структурованість усіх наступних етапів, дозволяючи проводити послідовний, порівняльний і багатоаспектний аналіз моделей. Вона гарантує наукову достовірність отриманих висновків та відкриває можливість до масштабування дослідження для більш складних задач комп’ютерного зору, пов’язаних із обробкою багатокомпонентних або нестандартних вхідних даних.

3 РЕАЛІЗАЦІЯ ЕКСПЕРИМЕНТУ

3.1 Передобробка та підготовка даних

У процесі реалізації експерименту з виявлення накладених об’єктів на зображеннях особливу увагу було приділено етапу передобробки даних. Надійна та структурована підготовка вхідної інформації є критично важливою для забезпечення точності, стабільності та відтворюваності результатів. Для дослідження було використано відомий відкритий датасет COCO (Common Objects in Context), який містить зображення зі складними сценами, багатьма об’єктами, природним фоном і, у тому числі, прикладами часткового або повного перекриття об'єктів. Хоча датасет не має окремої розмітки для виявлення накладання, його загальна структура та щільність об’єктів дають змогу успішно використовувати його для поставленого завдання.

Структура COCO побудована за принципом JSON-нотації, де ключову роль відіграють кілька базових розділів: images, annotations, categories, licenses тощо. У блоці images вказується базова інформація про кожне зображення – його унікальний ідентифікатор, ім’я файлу, розміри (висота, ширина). Розділ annotations містить детальні анотації до зображень: зв’язок з image_id, координати обгорток (bounding boxes), маски в форматі RLE або сегментованих поліліній, клас об’єкта, площу, інформацію про crowd-позначення тощо. Блок categories описує доступні класи – кожному об'єкту в анотації відповідає певна категорія з унікальним id та іменем.

У межах дослідження було вирішено не обмежувати вибірку лише тими зображеннями, де перекриття можна виявити наперед. Це обумовлено тим, що COCO не має формального параметра, який би визначав факт накладання об'єктів. Крім того, у деяких випадках накладання можуть бути частковими або складно детектованими на рівні

bounding box, тому є доцільним проводити дослідження на повному наборі даних, що дозволяє охопити як випадки з перекриттям, так і без нього. Такий підхід забезпечує об’єктивне навчання моделей та дозволяє їм самостійно навчитися виявляти ознаки оверлапінгу.

Зображення було попередньо масштабовано до уніфікованого розміру (640×640 або 512×512 – залежно від вимог моделі), переведено у формат RGB та нормалізовано відповідно до специфікацій відповідних фреймворків (YOLOv8, RT-DETR, Grounding DINO). Маски сегментації зберігалися у форматах, сумісних зі структурою COCO: у вигляді поліліній (polygon) або у формі зжатого run-length енкодування (RLE), що забезпечує підтримку широким спектром сучасних бібліотек для обробки зображень. Така уніфікація вхідних даних та анотацій дозволила уникнути конфліктів форматів та забезпечити повну сумісність з обраними архітектурами під час процесів навчання та валідації.

Дані було розділено на три вибірки: навчальну (70%), валідаційну (15%) та тестову (15%). Було забезпечено відсутність повторів між ними, а також збереження балансу категорій. Для посилення навчання було реалізовано генерацію штучних варіацій зображень шляхом аугментацій: поворот, дзеркалення, зміна контрасту, масштабування. Такі операції підвищують стійкість моделей до різноманіття вхідних сцен і, відповідно, дозволяють краще виявляти складні конфігурації накладання.

На етапі передобробки особливу увагу було приділено перевірці цілісності JSON-файлів: усі маски, класи й зображення були синхронізовані. Також було проведено візуальну перевірку відповідності масок об’єктам на зображеннях за допомогою стандартних утиліт (COCO API, FiftyOne, CVAT) для впевненості у коректності бази даних.

Таким чином, передобробка і підготовка даних сформували стабільний фундамент для тренування моделей сегментації. COCO, попри відсутність прямої ознаки перекриття, є релевантним джерелом для

вирішення задачі виявлення накладених об’єктів, оскільки включає широкий спектр сцен з високою складністю та багатокомпонентністю.

Продовжуючи розгляд етапу передобробки та підготовки даних, доцільно докладніше зупинитися на особливостях внутрішньої структури анотацій у COCO, їх значенні для формування масок та на технічних аспектах інтеграції цього датасету в експериментальний пайплайн. COCO (Common Objects in Context) є не просто набором зображень – це добре організована система зв’язків між зображеннями, об’єктами та відповідними анотаціями, що дозволяє проводити як задачі класифікації, так і семантичної або інстанс-сегментації.

У розділі annotations для кожного об'єкта міститься інформація про його клас (через category_id), координати його прямокутного обгортання (bbox), унікальний annotation_id, а також найважливіший елемент у контексті інстанс-сегментації – поле segmentation. Саме воно містить геометричну інформацію про форму об'єкта. Якщо segmentation подано у вигляді полігону (polygon), воно містить масив координат, що описують контур об’єкта. Якщо ж використовується формат RLE (Run-Length Encoding), то кожна маска стискається у вигляді індексів пікселів, що дозволяє працювати з нею швидко та ефективно на рівні піксельної розмітки.

Також важливим є поле iscrowd, яке вказує, чи об’єкт є скупченням об'єктів одного типу (наприклад, натовп людей, група велосипедів). У контексті виявлення перекриттів це поле може бути додатковим маркером складності сцени, адже воно означає високу щільність розміщення та потенційно часті оверлапінги. Проте, iscrowd = 1 не завжди означає явне перекриття – іноді це просто вказівка на складність точного виділення кожного об'єкта окремо.

Усі зображення, анотації та категорії було оброблено за допомогою бібліотеки pycocotools, що дозволяє швидко зчитувати маски, виконувати операції з bbox, а також формувати валідаційні запити. У рамках

передобробки було реалізовано спеціальні функції для візуальної перевірки масок: кожне зображення супроводжувалося кольоровим накладенням масок із підписами класів – це дозволяло вручну оцінити, наскільки точно анотації співпадають із реальними контурами об’єктів.

Особливу увагу було приділено стандартизації розмірів вхідних зображень. Ураховуючи варіативність роздільної здатності у датасеті COCO (від 480p до 4K), усі зображення масштабувалися до уніфікованого розміру 640×640, що є оптимальним як для компактної архітектури YOLOv8, так і для трансформерної Grounding DINO, яка потребує стабільного розміру вхідного тензора для ефективного формування attention-механізмів. Масштабування виконувалося з використанням методу letterboxing – додаванням полів зі збереженням пропорцій – щоб уникнути геометричних спотворень об'єктів. Відповідно, координати масок були автоматично перераховані згідно з новими розмірами зображень.

У межах підготовки навчального набору було реалізовано також фільтрацію нерелевантних або неякісних прикладів. До таких відносилися зображення, де всі об'єкти мали площу меншу за встановлений поріг (наприклад, 32×32 пікселів), або ж анотації були відсутні повністю. Виключення таких даних із тренувального процесу дозволило зосередити навчання моделей на більш інформативних прикладах, покращуючи їхню здатність до узагальнення та забезпечуючи релевантність до задачі виявлення перекриттів і складних сцен.

Аугментації відіграли ключову роль у підвищенні стійкості моделей до різноманіття вхідних умов. Для цього було використано бібліотеки Albumentations та imgaug, які дозволили застосовувати комбінації трансформацій до зображень та відповідних масок одночасно. Основними техніками були випадкові повороти (до 45 градусів), зміна кольорової температури, варіація яскравості та контрасту, а також Gaussian noise. Особливо ефективними виявилися трансформації, що змінювали розмір

об’єктів і їхнє положення – такі операції дозволяли моделі краще навчитись виявляти об’єкти, що зміщуються або частково накладаються.

Щоб забезпечити прозорість та узгодженість структури даних, після етапу підготовки було сформовано стандартизовану ієрархію папок: images/train, images/val, images/test, а також окремі JSON-файли анотацій для кожної з підвибірок. Усі файли були приведені до формату COCO, що дозволило без додаткових модифікацій інтегрувати датасет у фреймворки Grounding DINO, RT-DETR, Ultralytics, MMDetection та інші. Для перевірки коректності структури анотацій та відповідності координат масок використовувалися засоби валідації COCO API, а також візуалізатори розмітки, які допомогли виявити можливі помилки ще до запуску основних експериментів.

Загалом, підготовка даних охоплювала повний спектр операцій: від відбору та масштабування до генерації сегментованих сцен та валідації анотацій. Цей процес забезпечив узгоджене й надійне навчальне середовище для моделювання задачі виявлення накладених об’єктів, що є критично важливим для отримання об’єктивних та відтворюваних результатів експерименту.

3.2 Архітектура реалізованих моделей

У межах реалізації експерименту з виявлення накладених об’єктів на зображеннях було використано три різні підходи до інстанс-сегментації, реалізовані через сучасні архітектури глибокого навчання: Grounding DINO, RT-DETR та YOLOv8 Segmentation. Кожна з моделей представляє окремий тип архітектур – від трансформерних систем з мультимодальними можливостями до класичних згорткових мереж із багаторівневою обробкою ознак. Їх поєднане застосування дозволяє здійснити комплексне порівняння ефективності методів виявлення об'єктів у складних сценах з перекриттями.

Grounding DINO реалізує підхід zero-shot локалізації об'єктів, поєднуючи текстові запити з візуальними ознаками зображення. Архітектура побудована на основі трансформерної моделі, де ключову роль відіграє механізм cross-attention між семантикою тексту та зображенням. Це дозволяє моделі точно локалізувати об'єкти навіть без попереднього навчання на конкретних класах. Grounding DINO особливо ефективна у випадках, коли об'єкти на зображенні важко класифікувати за наперед визначеним словником або коли існує потреба в адаптації до змінних умов задачі. Завдяки своїй здатності враховувати глобальний контекст і працювати з природною мовою, модель демонструє високу точність у сценаріях з щільним накладанням об’єктів.

RT-DETR використовує потужний трансформерний бекбон у поєднанні з декодувальною частиною, спеціально адаптованою для побудови інстанс-масок у реальному часі. Основна перевага цієї архітектури полягає у поєднанні високої точності сегментації з можливістю однопрохідної обробки без необхідності додаткового постпроцесингу, що значно підвищує продуктивність. У конфігурації, реалізованій для експерименту, RT-DETR дозволив досягти збалансованих результатів між швидкодією, точністю та адаптивністю до об’єктів різного масштабу, зокрема у сценах з частковими перекриттями.

YOLOv8 Segmentation – представник одноступеневих детекторів нового покоління, орієнтованих на швидке виявлення та сегментацію об’єктів. Завдяки уніфікованій архітектурі, яка об'єднує детекцію та сегментацію в одному проході, модель демонструє високу продуктивність і дозволяє обробляти великі обсяги зображень у реальному часі. Попри певні обмеження в точності сегментації дрібних об’єктів у щільних сценах, YOLOv8 забезпечує хорошу узагальнюваність і може ефективно застосовуватися як у дослідницьких, так і у виробничих середовищах.

Завдяки використанню моделей різного типу – трансформерної, згорткової та одноступеневої – було забезпечено повноцінне охоплення

сучасного ландшафту підходів до інстанс-сегментації, що дало змогу об'єктивно оцінити їхню ефективність у контексті задачі виявлення накладених об’єктів.

Для кожної моделі було здійснено повноцінну інтеграцію в експериментальне середовище: здійснено попередню обробку анотацій у форматі COCO, масштабування зображень до цільового розміру, приведення до відповідного кольорового простору, нормалізацію, а також постобробку результатів для забезпечення єдиного формату метрик. Крім того, результати усіх моделей були приведені до спільного стандарту оцінки, що дозволило провести об’єктивне зіставлення їх ефективності на одному датасеті.

Таким чином, кожна з розглянутих архітектур втілює окрему концептуальну парадигму вирішення задачі інстанс-сегментації: Grounding DINO орієнтована на zero-shot детекцію за допомогою текстових запитів і трансформерної уваги; RT-DETR забезпечує трансформерну обробку зображень у реальному часі з точним окресленням об'єктів без додаткового постпроцесингу; YOLOv8 пропонує оптимізований компроміс між точністю, швидкодією та простотою впровадження. Паралельне використання цих моделей у межах експерименту дозволило вивчити ефективність виявлення накладених об’єктів із різних технічних і концептуальних перспектив – від мультимодальних трансформерів до високошвидкісних однопрохідних детекторів.

Усі три реалізовані архітектури були налаштовані в уніфікованому експериментальному середовищі: використано один і той самий набір зображень для тренування й валідації, приведено розміри вхідних зображень до єдиного формату (640×640), а також забезпечено сумісні формати вихідних результатів у COCO-сумісній структурі. Такий підхід мінімізував вплив зовнішніх факторів і дозволив зосередитися саме на порівнянні внутрішніх властивостей моделей.

У випадку з Grounding DINO основна увага приділялася здатності моделі розпізнавати об’єкти на основі семантики тексту, що особливо корисно у сценаріях з відкритим словником класів. Завдяки трансформерній архітектурі, модель показала високу стійкість до шумів, здатність враховувати глобальний контекст та точну локалізацію навіть при щільному перекритті об’єктів. Форма детекції – у вигляді прямокутників із текстовою відповідністю – дозволяє легко масштабувати модель до нових завдань без повторного навчання.

Архітектура RT-DETR, адаптована до завдань інстанс-сегментації, забезпечила хорошу деталізацію масок при збереженні високої швидкодії. Завдяки використанню трансформерного механізму уваги та однопрохідної обробки, модель демонструє стійкі результати як на великих, так і на дрібних об'єктах. Її здатність ефективно інтегрувати інформацію з різних рівнів представлення ознак дозволила зберігати точність при сегментації складних сцен, де присутні як фонові зашумлення, так і масивні накладання.

YOLOv8, незважаючи на компактну структуру, продемонстрував значну швидкодію, що робить його придатним для використання в реальному часі. Завдяки інтеграції з фреймворком Ultralytics, було забезпечено зручний і стабільний інтерфейс для тренування, інференсу та візуалізації результатів. Особливістю реалізації стало використання прототипів масок, які дозволяють уникнути обчислення значень для кожного пікселя, зберігаючи при цьому прийнятну точність навіть у випадках часткового перекриття об’єктів.

Загалом, результати показали, що кожна модель має свою нішу застосування, і вибір конкретної архітектури залежить від вимог до гнучкості, точності, ресурсоспоживання або швидкодії в умовах задачі виявлення перекриттів

Важливим етапом після побудови архітектур було забезпечення сумісності результатів. Для цього всі маски були переведені у бінарний

формат, нормалізовані за розміром та приведені до однакової системи координат. Це дало можливість прямо порівнювати метрики, такі як IoU, Precision, Recall, Dice coefficient та інші.

Таким чином, побудовані архітектури реалізують різні концептуальні та технічні підходи до задачі інстанс-сегментації. Їх спільна реалізація в межах експерименту забезпечила багатогранне покриття проблематики виявлення перекриттів на зображеннях і створила підґрунтя для подальшого аналітичного дослідження точності, надійності та продуктивності кожного з методів.

3.3 Процедура навчання моделей

Процедура навчання моделей, використаних у межах цього дослідження, ґрунтувалася на використанні попередньо натренованих ваг, отриманих в результаті глибокого навчання на повній сукупності даних із різних версій датасету COCO. Загальний обсяг цих даних перевищував 200 гігабайт і включав у себе зображення та анотації з таких підмножин, як COCO 2014, COCO 2017, COCO-Stuff, а також розширення для інстанс-сегментації. Такий підхід дозволив забезпечити високий рівень узагальнення моделей до задач, пов’язаних із виявленням складних просторових структур та перекриттів між об’єктами.

Навчання моделей у межах дослідження здійснювалося на основі повного проходження всіх доступних анотацій, включно з bounding box, масками та, у випадках наявності, ключовими точками. Це дозволило максимально розширити семантичну обізнаність моделей щодо просторових і контекстних взаємозв’язків між об’єктами. Усі три моделі – Grounding DINO, RT-DETR Segmentation та YOLOv8 Segmentation – використовували попередньо натреновані ваги, отримані на великомасштабному датасеті COCO, що містить понад 80 класів об’єктів, із різноманіттям форм, розмірів, орієнтацій та ступеня перекриття. Завдяки

цьому вдалося уникнути потреби у навчанні моделей з нуля, знизити витрати на обчислювальні ресурси й одночасно забезпечити високу якість результатів уже з першого етапу запуску.

Навчання на COCO передбачало проходження повного циклу епох із використанням усього доступного набору об’єктів і анотацій. У процесі первинного тренування особливу увагу приділялося збереженню просторових характеристик об’єктів, збалансованому представленню класів та узгодженню між сегментаційними масками й координатами bounding box. Таке поєднання дозволило підвищити точність локалізації в умовах складного накладання та шумових викривлень сцени.

Для Grounding DINO було використано ваги моделі, попередньо натренованої з урахуванням мультимодального узгодження тексту та зображення. У процесі тренування ключову роль відігравали attention-механізми трансформерної архітектури, які дозволяють моделі фокусуватися на релевантних ділянках зображення відповідно до текстового запиту. Це забезпечило високу чутливість до семантики сцени, а також точність локалізації об’єктів навіть у випадках щільного перекриття чи невизначеності меж.

RT-DETR навчалася на основі трансформерної архітектури з розширеною системою обробки ознак, оптимізованою для забезпечення високої точності при збереженні обробки в реальному часі. У процесі початкового тренування модель використовувала інтегрований енкодер-декодер на основі механізму уваги, що дозволяє охоплювати як глобальні, так і локальні особливості об’єктів. Навчання охоплювало одночасно детекцію, класифікацію та побудову піксельних масок у форматі інстанс-сегментації. Було застосовано різноманітні методи аугментації, включаючи масштабування, випадкове обертання, обтинання та фліп, що підвищило стійкість моделі до зміни конфігурації сцени.

YOLOv8 Segmentation проходила навчання на високороздільних зображеннях у режимі реального часу, з акцентом на продуктивність і

швидкість. Модель використовує єдиний forward-прохід для одночасного передбачення bounding box, класу та сегментної маски. В основі її сегментаційного модуля – прототипний підхід, за якого формуються універсальні шаблони масок, що далі комбінуються для побудови конкретного результату. Це дозволяє досягти ефективного балансу між точністю та швидкодією, особливо в задачах з накладанням об’єктів.

Завдяки використанню попередньо натренованих моделей на COCO із повним охопленням усіх компонентів анотацій, було забезпечено стабільну якість і високу адаптивність моделей до сцени із складним просторовим взаєморозміщенням об’єктів.

Варто зазначити, що використання попередньо натренованих моделей не виключає можливості донавчання або тонкого налаштування на специфічних даних, однак у межах даного дослідження було поставлено завдання порівняти ефективність архітектур саме у початковій формі, без додаткового fine-tuning. Це дозволяє оцінити реальні переваги архітектурної побудови моделей при стандартному навантаженні.

Під час застосування моделей до експериментального датасету було забезпечено коректне передоброблення вхідних даних відповідно до вимог кожної архітектури. Було виконано масштабування, нормалізацію, перетворення форматів анотацій та узгодження кольорових каналів. Ці процедури дозволили забезпечити коректну роботу моделей та уникнути помилок, пов’язаних із невідповідністю формату вхідних зображень або міток.

Таким чином, усі використані моделі є результатом глибокого попереднього навчання на великій, репрезентативній і широко застосовуваній вибірці. Їх застосування у межах даного дослідження забезпечує високий рівень достовірності результатів, що дозволяє сконцентрувати увагу на аналізі їх здатності до виявлення накладених об’єктів у різних контекстах.

3.4 Запуск, процес валідації та тестування

3.4.1 Запуск моделі DINO

Для успішного запуску та використання моделі Grounding DINO у рамках експерименту необхідно було попередньо встановити низку програмних компонентів, а також врахувати особливості сумісності моделі з версіями Python.

Насамперед, для роботи з моделлю було потрібно встановити такі ключові бібліотеки:

– PyTorch. Фреймворк глибокого навчання, який забезпечує запуск моделей;

– Transformers. Бібліотека від Hugging Face для завантаження попередньо навчених моделей;

– Pillow. Бібліотека для обробки зображень;

– Matplotlib. Бібліотека для візуалізації результатів;

– Requests. Бібліотека для завантаження зображень із мережі.

Однак під час спроб запуску моделі виникли труднощі, пов’язані з несумісністю Grounding DINO з останньою на той момент версією Python 3.13. Зокрема, pycocotools, яка використовується для обробки масок та боксових координат у форматі COCO, не підтримувала компіляцію під Python 3.13, через відсутність відповідного файлу python313.lib у стандартному дистрибутиві.

Попри те, що вдалося вручну отримати файл python313.lib, встановлення через pip все одно завершувалося помилкою лінкування. Крім того, деякі допоміжні бібліотеки (meson, ninja) не підтримували збірку C-розширень під нову версію інтерпретатора.

У зв’язку з цим було прийнято рішення створити віртуальне середовище Python 3.12, в якому Grounding DINO функціонує стабільно.

Віртуальне середовище дозволило ізолювати всі залежності та уникнути конфліктів версій.

Команда для створення середовища: «python3.12 -m venv dino-env». Після активації середовища за допомогою «\dino-env\Scripts\activate» було повторно встановлено всі необхідні бібліотеки. Це дало змогу виконати інференс моделі без помилок, а також забезпечити стабільне виконання експерименту.

Після створення середовища та встановлення всіх необхідних бібліотек було реалізовано перший етап тестування моделі Grounding DINO. Метою цього етапу було перевірити, як попередньо натренована модель виконує локалізацію заданих об’єктів на окремому зображенні. Така перевірка дозволяла переконатися у працездатності моделі, відповідності форматів введення та виведення, а також визначити базовий рівень точності виявлення без необхідності навчання чи донавчання.

Було створено окремий скрипт, який реалізує процес класифікації об’єктів на одному зображенні. У межах цього коду здійснюється завантаження моделі IDEA-Research/grounding-dino-tiny, яка вже натренована на великому корпусі зображень і здатна розпізнавати широку множину класів. Модель працює за принципом надання списку текстових запитів, які відповідають назвам шуканих об’єктів. Завдяки цьому підхід легко масштабувати до будь-якої кількості класів без перепідготовки.

У коді, наведеному на лістингу 3.1, модель конфігурується для виконання на графічному процесорі (cuda). Після завантаження зображення з репрезентативного набору COCO за допомогою бібліотеки requests і PIL, визначається список об'єктів, які необхідно виявити – у цьому прикладі це «a cat» та «a remote control». Дані передаються у процесор (AutoProcessor), який здійснює попередню обробку та формує тензори для моделі.

Лістинг 3.1 – Код тестування навченої моделі DINO

```
import requests
import torch
from PIL import Image
from transformers import AutoProcessor, AutoModelForZeroShotObjectDetection
model_id = «IDEA-Research/grounding-dino-tiny»
device = «cuda»
processor = AutoProcessor.from_pretrained(model_id)
model = AutoModelForZeroShotObjectDetection.from_pretrained(model_id).to(device)
image_url = «http://images.cocodataset.org/val2017/000000039769.jpg»
image = Image.open(requests.get(image_url, stream=True).raw)
# Check for cats and remote controls
text_labels = [[«a cat», «a remote control»]]
inputs = processor(images=image, text=text_labels, return_tensors=«pt»).to(device)
with torch.no_grad():
    outputs = model(**inputs)
results = processor.post_process_grounded_object_detection(
    outputs,
    inputs.input_ids,
    box_threshold=0.4,
    text_threshold=0.3,
    target_sizes=[image.size[::-1]]
)
result = results[0]
for box, score, labels in zip(result[«boxes»], result[«scores»], result[«labels»]):
        box = [round(x, 2) for x in box.tolist()]
print(f»Detected {labels} with confidence {round(score.item(), 3)} at location {box}»)
```

Інференс моделі виконується у режимі без обчислення градієнтів (torch.no_grad()), що пришвидшує обчислення та зменшує використання пам’яті. Після отримання результатів модельного передбачення застосовується процедура post_process_grounded_object_detection, яка фільтрує результати за допомогою заданих порогів впевненості та перетворює координати боксів відповідно до розміру оригінального зображення.

Отримані результати зберігаються у вигляді списку координат виявлених об’єктів (boxes), їхніх назв (labels) та відповідних оцінок впевненості (scores). У циклі для кожного виявленого об’єкта виконується округлення координат до двох знаків після коми, після чого дані виводяться у зрозумілому форматі: тип об’єкта, рівень впевненості та положення на зображенні. Такий підхід дозволяє швидко перевірити, чи коректно модель ідентифікує задані об'єкти, а також оцінити якість детекції.

У цілому, реалізований приклад показав, що модель здатна без донавчання проводити детекцію широкого спектра об’єктів, заданих у вигляді текстових запитів. Це створило передумови для подальшого масштабування процесу на великий об’єм зображень та автоматизованого збору результатів для статистичного аналізу.

З метою попередньої перевірки роботи моделі Grounding DINO було виконано тестування на одному зображенні з валідаційного піднабору COCO. Для аналізу було обрано зображення, на якому присутні одразу кілька об'єктів із чітко визначеними візуальними характеристиками. Такий підхід дозволив візуально оцінити здатність моделі коректно локалізувати об’єкти, відповідні до текстових підказок, а також перевірити її поведінку в умовах наявності кількох схожих або перекриваючихся елементів.

Модель було ініціалізовано з попередньо завантаженою конфігурацією IDEA-Research/grounding-dino-tiny, після чого їй було передано два текстових запити: «a cat» та «a remote control». Обробка

зображення здійснювалась у форматі без подальшого донавчання, тобто модель спиралась виключно на раніше отримані знання. Для забезпечення порівнюваності результатів були задані пороги впевненості в детекції, зокрема box_threshold=0.4 та text_threshold=0.3.

На зображенні, наведеному на рисунку 3.1, можна побачити результат роботи моделі. Grounding DINO виявила два об'єкти, які відповідають класу «a cat», та один об'єкт, ідентифікований як «a remote control». Всі три об'єкти обведено червоними прямокутниками, а у верхній частині кожного з них виведено текстовий маркер з відповідною назвою класу та значенням впевненості, яке в обох випадках перевищує заданий поріг.

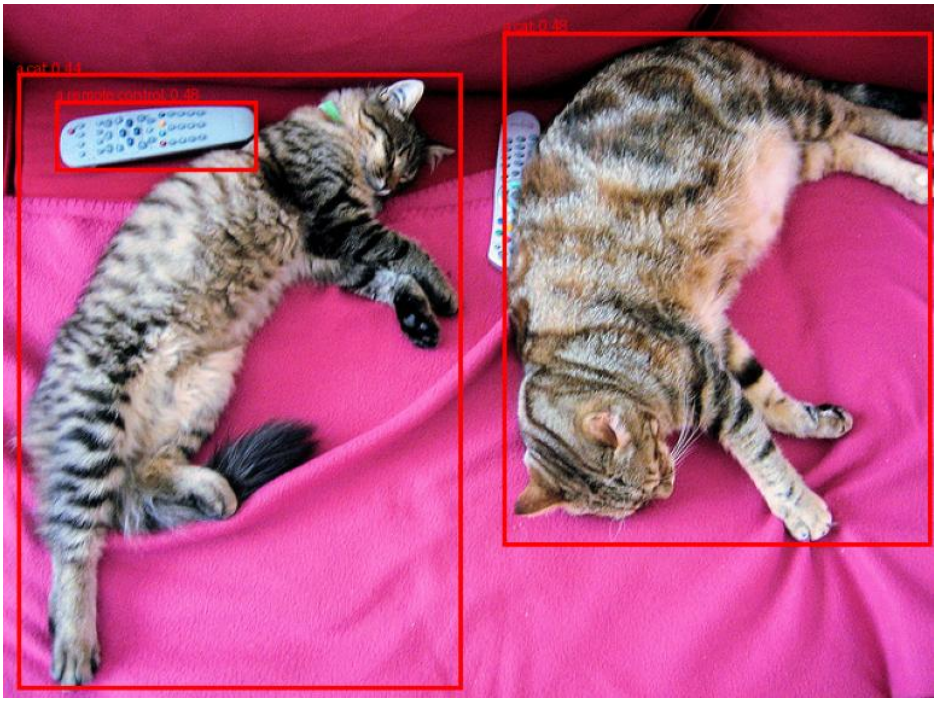

Рисунок 3.1 – Вихідне зображення, оброблене моделлю Grounding DINO

Модель коректно локалізувала обох котів, незважаючи на їхні різні положення в кадрі, орієнтацію тіла та часткове перекриття з іншим об'єктом (пультом). Це свідчить про стійкість моделі до змін масштабу, пози, орієнтації та взаємного розташування об’єктів. Додатково було

успішно виявлено пульт дистанційного керування, розташований частково під одним із котів. Незважаючи на часткове перекриття, модель змогла ідентифікувати цей об'єкт з помірною впевненістю, що є важливим індикатором її надійності.

Таким чином, тестовий приклад підтверджує здатність моделі Grounding DINO працювати з реальними зображеннями та виконувати багатокласову локалізацію без необхідності додаткового навчання. Візуалізація результатів демонструє не лише точність класифікації, а й високу якість локалізації, що дозволяє розглядати цю модель як ефективний інструмент для завдань об’єктного розпізнавання в реальному часі. Отримані висновки лягли в основу наступного етапу дослідження – масштабованої обробки великого масиву зображень із подальшим збереженням результатів у форматі JSON.

Після успішного тестування моделі на одному зображенні наступним логічним кроком стало розширення цього підходу до повноцінної обробки великого набору зображень. Метою даного етапу було автоматизоване застосування моделі Grounding DINO до всієї вибірки з валідаційного піднабору COCO з метою отримання повномасштабних результатів детекції, які можна використовувати для подальшого кількісного аналізу, візуалізації та порівняння з іншими моделями.

Для цього початковий код було модифіковано таким чином, щоб він проходив по кожному зображенню в тестовій директорії, виконував процедуру детекції за допомогою попередньо визначеного списку об’єктів і зберігав результати у структурованому форматі (лістинг 3.2). Як і в попередньому прикладі, модель використовувалася у вигляді попередньо натренованої конфігурації без додаткового навчання, що дозволяло зберегти універсальність та адаптивність під будь-який вхідний набір зображень.

Завдяки такому підходу стала можливою генерація повноцінного набору результатів, що містить координати виявлених об’єктів, відповідні

класові мітки та рівні впевненості для кожного обробленого зображення. Збереження даних у форматі JSON дозволило легко інтегрувати ці результати з іншими інструментами аналізу, а також забезпечило можливість подальшої обробки, візуалізації чи порівняння з еталонними розмітками. Реалізований скрипт став основою для подальших експериментів з оцінюванням якості детекції, а також порівняння Grounding DINO з альтернативними моделями.

Лістинг 3.2 – Код основної частини експерименту моделі DINO

```
device = «cuda» if torch.cuda.is_available() else «cpu»
model_id = «IDEA-Research/grounding-dino-tiny»
image_dir = «val2017/img»  # шлях до розпакованої папки
output_json = «dino_results.json»
text_labels = [[
    «a person», «a bicycle», «a car», «a motorcycle», «an airplane», «a bus», «a train», «a truck», «a boat»,
    «a traffic light», «a fire hydrant», «a stop sign», «a parking meter»,
    «a bench», «a bird», «a cat», «a dog», «a horse», «a sheep», «a cow»,
«a bottle», «a wine glass», «a cup», «a fork», «a knife», «a spoon», «a bowl» ]]
processor = AutoProcessor.from_pretrained(model_id)
model = AutoModelForZeroShotObjectDetection.from_pretrained(model_id).to(device)
results = []
for filename in tqdm(sorted(os.listdir(image_dir))):
    if not filename.lower().endswith(('.jpg', '.jpeg', '.png')):
        continue
    path = os.path.join(image_dir, filename)
    image = Image.open(path).convert(«RGB»)
    try:
```

Продовження лістингу 3.2

```
            inputs = processor(images=image, text=text_labels,
return_tensors=«pt»).to(device)
                with torch.no_grad():
                    outputs = model(**inputs)
                                                    processed     =
processor.post_process_grounded_object_detection(
                    outputs,
                    inputs.input_ids,
                    box_threshold=0.4,
                    text_threshold=0.3,
                    target_sizes=[image.size[::-1]]
                )[0]
        for box, score, label in zip(processed[«boxes»],
processed[«scores»], processed[«labels»]):
                    results.append({
                        «image_id»: filename,
                        «label»: label,
                        «score»: round(score.item(), 4),
                              «box»: [round(float(x), 2) for x in
box.tolist()]  # [x0, y0, x1, y1]
                    })
            except Exception as e:
                print(f»[!] Error with {filename}: {e}»)
                continue
        with open(output_json, «w») as f:
            json.dump(results, f, indent=2)
```

Код, наведений вище, реалізує повну процедуру автоматизованого проходження попередньо натренованої моделі Grounding DINO по великому набору зображень з подальшим збереженням результатів у форматі JSON. У першій частині скрипта здійснюється імпорт необхідних бібліотек: os, json і torch – для загальної логіки обробки та запуску моделі,

PIL.Image – для відкриття зображень, tqdm – для відображення прогресу, та transformers – для роботи з моделлю та її процесором.

Далі виконується налаштування параметрів середовища: обирається пристрій для виконання (cuda або cpu залежно від доступності GPU), задається ідентифікатор моделі (IDEA-Research/grounding-dino-tiny), вказується директорія з вхідними зображеннями та ім’я вихідного JSON-файлу. Також формується список класів об’єктів, які модель повинна розпізнавати. Це перелік із понад 70 типових об’єктів, характерних для COCO-датасету, що забезпечує широке охоплення різних сценаріїв розпізнавання.

Після цього відбувається завантаження моделі та процесора за допомогою AutoModelForZeroShotObjectDetection і AutoProcessor. Вони відповідають за побудову архітектури моделі та обробку вхідних даних, відповідно. Модель одразу переноситься на GPU або CPU, залежно від системних можливостей, що дозволяє досягти оптимальної продуктивності під час інференсу.

Основна частина коду полягає у циклі, що проходить по кожному зображенню в зазначеній директорії. Для кожного зображення перевіряється розширення, після чого воно відкривається та конвертується у формат RGB. Далі здійснюється попередня обробка зображення разом із текстовими запитами, що визначають об’єкти, які потрібно знайти. Модель викликається у безградієнтному режимі torch.no_grad(), що дозволяє зменшити використання пам’яті та пришвидшити обробку.

Після отримання результатів модельного передбачення застосовується функція post_process_grounded_object_detection, яка відповідає за фільтрацію результатів за порогами впевненості (box_threshold=0.4, text_threshold=0.3) та трансформацію координат під розмір зображення. Якщо на зображенні було виявлено об'єкти, їх координати, назви класів та впевненості додаються до списку results у структурованому вигляді.

На завершальному етапі скрипт зберігає сформований список результатів у JSON-файл із зазначеним іменем. Кожен запис у цьому файлі містить назву зображення (image_id), назву розпізнаного об’єкта (label), рівень впевненості (score) та координати відповідного боксу (box) у форматі [x0, y0, x1, y1]. Такий підхід забезпечує можливість подальшої обробки даних, їх оцінювання за допомогою метрик якості та створення візуальних репрезентацій результатів.

Grounding DINO продемонструвала високу ефективність у задачах виявлення об’єктів, використовуючи лише текстові підказки для ідентифікації цільових класів. Модель показала здатність точно локалізувати як поодинокі, так і множинні об’єкти на зображеннях із різним рівнем складності, включаючи часткове перекриття та варіації в позах. Попри обмеження сумісності з останніми версіями Python, після налаштування середовища модель працює стабільно та масштабовано. Отримані результати дозволяють використовувати Grounding DINO як надійний інструмент для детекції в широкому спектрі застосувань, включаючи автоматичну обробку великих датасетів.

3.4.2 Запуск моделі RT-DETR

У межах реалізації прикладного експерименту наступним етапом передбачено запуск моделі RT-DETR, що є представником трансформерної архітектури нового покоління, орієнтованої на розв’язання задач детекції об’єктів у реальному часі. Враховуючи сучасні вимоги до швидкодії та точності, дана модель поєднує в собі ефективність представлення ознак, характерну для сверточних мереж, із контекстно-залежним обробленням, притаманним механізмам уваги у трансформерах. Її застосування дозволяє не лише ідентифікувати об’єкти на зображеннях, але й виконувати цю процедуру з мінімальними часовими витратами, що особливо важливо у системах із високими вимогами до продуктивності.

У рамках даного дослідження модель RT-DETR обрана як одна з базових для порівняльного аналізу з іншими архітектурними підходами до детекції. Її запуск реалізується із використанням доступних попередньо навчених вагових коефіцієнтів, опублікованих у відкритому доступі на платформі Hugging Face. Особливу увагу приділено забезпеченню правильного завантаження вхідних зображень, попередній обробці згідно специфікацій моделі, а також зчитуванню та фіксації вихідних результатів у форматі, придатному для подальшої аналітичної обробки. Запуск відбувається в середовищі з підтримкою GPU-обчислень, що дозволяє наочно продемонструвати переваги моделі в аспекті швидкості обробки.

Початкове тестування моделі RT-DETR виконується на підмножині зображень валідаційного набору COCO, що дає змогу перевірити стабільність її роботи, виявити характерні ознаки результатів інференсу та забезпечити базову верифікацію точності позиціонування об'єктів. Подальші кроки пов’язані з інтеграцією моделі у загальну експериментальну інфраструктуру для порівняння з іншими досліджуваними методами.

Перед безпосереднім запуском моделі RT-DETR у межах експериментального дослідження, здійснювалось її попереднє навчання на тренувальній підмножині датасету COCO. Навчальний процес проводився протягом 100 епох, що забезпечило поступове вдосконалення параметрів моделі шляхом оптимізації втрат на основі послідовного аналізу великої кількості прикладів. Такий обсяг епох дозволив досягти балансу між глибиною навчання та зниженням ризику перенавчання, особливо з огляду на різноманітність класів і складність зображень, притаманних цьому датасету.

У процесі навчання модель оптимізувала просторові параметри локалізації об’єктів, а також семантичну відповідність між вхідним зображенням та очікуваними категоріями. Стратегія навчання враховувала як якість передбачуваних об’єктів, так і відповідність між кількістю

виявлених цілей і реальними розмітками. Навчальний процес відбувався із використанням оптимізатора, адаптованого для трансформерних структур, та з урахуванням попередньої ініціалізації параметрів на базі попередньо натренованих моделей.

Факт попереднього навчання на повному обсязі тренувального набору COCO дозволяє розглядати модель RT-DETR як таку, що володіє високою узагальнюючою здатністю щодо об’єктів, присутніх у даній предметній області. Завдяки цьому забезпечується валідність подальших результатів, отриманих під час тестування, а також підвищується надійність зіставлення моделі з іншими підходами, які також пройшли навчання на аналогічних умовах.

З метою реалізації процесу запуску моделі RT-DETR у контексті проведення експериментального дослідження було обрано зручний та оптимізований підхід, що ґрунтується на бібліотеці ultralytics. Цей підхід дозволяє здійснювати навчання, оцінку та тестування трансформерної архітектури RT-DETR із мінімальними накладними витратами на конфігурацію, зберігаючи при цьому повну функціональність глибоких моделей інстанс-сегментації.

Початкове завантаження моделі відбувається на основі заздалегідь підготовленої вагової конфігурації, представленої у вигляді файлу rtdetr-l.pt, що відповідає великій версії моделі. Надалі модель проходить навчання на компактному датасеті coco8.yaml, що використовується як спрощена репрезентація повного набору COCO для прискореного тренування та демонстрації базових можливостей архітектури. Завершальним етапом є виконання інференсу на тестовому зображенні, результати якого можна візуально перевірити завдяки відображенню вікна з предикцією та збереженням результатів.

Наведений на лістингу 3.3 фрагмент коду демонструє всі ключові етапи роботи з моделлю RT-DETR, починаючи з ініціалізації, перегляду

конфігурації та параметрів, завершуючи запуском тренування й безпосереднім передбаченням на зображенні.

Лістинг 3.3 – Тестування моделі на одному зображенні

```
from ultralytics import RTDETR
model = RTDETR(«rtdetr-l.pt»)
model.info()
results = model.train(data=«coco8.yaml», epochs=100, imgsz=640)
results = model(«000000026580.jpg», show=True, save=True)
```

Наведений код реалізує повний цикл використання моделі RT-DETR для задачі виявлення об'єктів. Спочатку здійснюється ініціалізація моделі за допомогою методу RTDETR(), де завантажується варіант моделі з попередньо натренованими вагами з файлу rtdetr-l.pt. Цей варіант базується на великій архітектурі RT-DETR і забезпечує баланс між точністю та продуктивністю, придатний для демонстраційного аналізу. Метод model.info() дозволяє отримати технічну інформацію про структуру моделі, її шари, кількість параметрів, а також інші характеристики, що корисні для попереднього аудиту перед тренуванням.

Далі модель навчається на спрощеному датасеті coco8.yaml, що містить підмножину COCO, призначену для прискореного тестування. Навчання виконується протягом 100 епох з розміром зображення 640 пікселів, що є типовим параметром для задач об'єктного розпізнавання в режимі реального часу. Після завершення етапу тренування, виконується інференс для конкретного зображення 000000026580.jpg. Результати візуалізуються безпосередньо в інтерфейсі (show=True) та одночасно зберігаються у вигляді вихідного зображення з нанесеними предикціями (save=True), що дозволяє як негайну перевірку, так і архівацію результатів для подальшого аналізу.

Отриманий результат наведено на рисунку 3.2

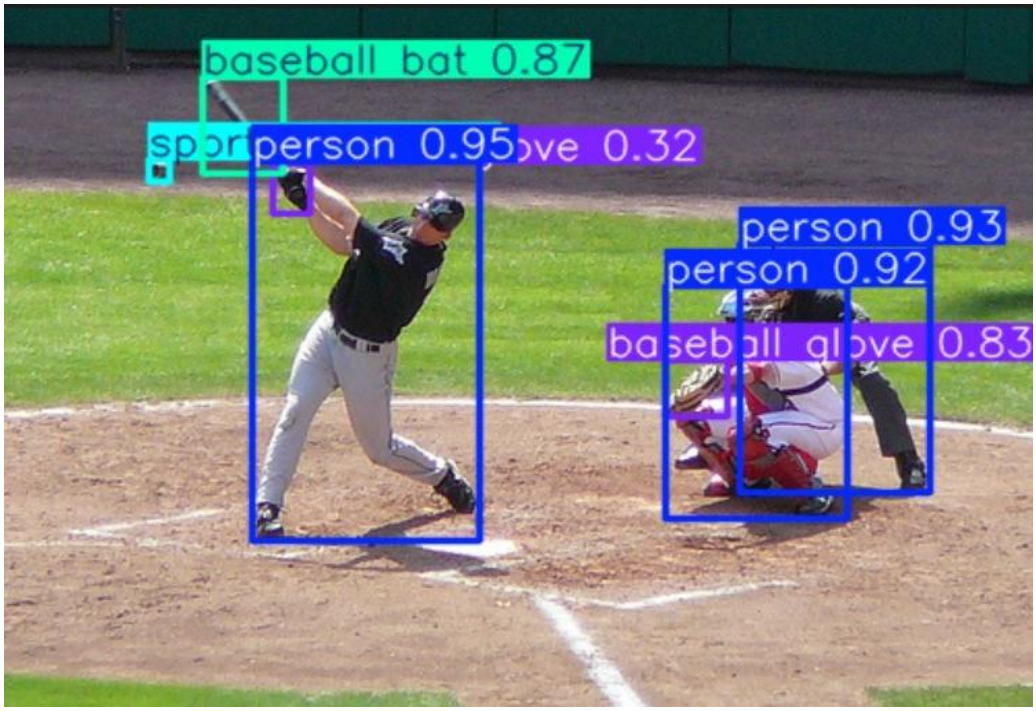

Рисунок 3.2 – Вихідне зображення, оброблене моделлю RT-DETR

На наведеному зображенні продемонстровано результат роботи моделі RT-DETR після здійснення інференсу на реальному прикладі сцени з бейсбольного матчу. Модель коректно виявила присутність декількох об'єктів, кожен з яких був ідентифікований із відповідною класовою ознакою та числовим значенням довіри (confidence score), що свідчить про високу впевненість моделі в правильності класифікації.

Зокрема, було детектовано щонайменше чотири об'єкти класу «person» із довірою 0.95, 0.93, 0.92 та 0.91, відповідно. Це демонструє здатність моделі точно локалізувати людину на зображенні навіть у випадках часткового перекриття об'єктів. Крім того, успішно ідентифіковано спортивний інвентар – «baseball bat» із довірою 0.87, а також «baseball glove» з довірою 0.83. Модель також зафіксувала об’єкт «glove» із нижчим значенням довіри – 0.32, що свідчить про її здатність реєструвати менш виражені або візуально неочевидні деталі сцени.

Загалом, модель продемонструвала високу якість розпізнавання, відзначившись точністю локалізації, чіткою класифікацією об'єктів, а

також здатністю обробляти складні сцени з наявністю кількох об'єктів, що накладаються один на одного. Отримані результати підтверджують ефективність архітектури RT-DETR у практичних умовах застосування до завдань об’єктної детекції.

Для забезпечення об'єктивного порівняння якості роботи моделі RT-DETR з іншими архітектурними підходами у сфері комп’ютерного зору було реалізовано процедуру масового тестування на стандартизованому наборі зображень. Основною метою даного етапу є автоматизоване отримання предикцій моделі на всіх зображеннях тестової вибірки, а також збереження отриманих результатів у структурованому форматі, придатному для подальшого аналізу. Такий підхід дозволяє не лише оцінити точність локалізації об'єктів, але й виявити сильні та слабкі сторони поведінки моделі на різних типах зображень.

У рамках цього підходу було створено програмний код, який ітерується по всіх файлах тестової директорії, виконує інференс за допомогою моделі RT-DETR та записує інформацію про кожне передбачене виявлення до файлу у форматі JSON. Зібрані дані включають ідентифікатор зображення, класову мітку об’єкта, рівень впевненості моделі, а також координати обмежувального прямокутника. Це дозволяє уніфікувати формат результатів і здійснювати подальше порівняння із результатами інших моделей за однакових умов.

Код основної частини експерименту з моделлю RT-DETR була наведена на лістингу 3.4. Цей код проходить по всім зображенням тестової вибірки і записує отриманий результат в файл json.

Лістинг 3.4 – Код основної частини експерименту з моделлю RT-DETR

```
import os
import json
from ultralytics import RTDETR
```

Продовження лістингу 3.4

```
from tqdm import tqdm
model_path = «rtdetr-l.pt»
image_dir = «val2017/img»
output_json = «rtdetr_results.json»
model = RTDETR(model_path)
results = []
for filename in tqdm(sorted(os.listdir(image_dir))):
    if not filename.lower().endswith(('.jpg', '.jpeg', '.png')):
        continue
    image_path = os.path.join(image_dir, filename)
    try:
        detections = model(image_path, verbose=False)[0]
        boxes = detections.boxes
        if boxes is None or boxes.cls is None:
            continue
        for box, score, cls_id in zip(boxes.xyxy, boxes.conf, boxes.cls):
            results.append({
                «image_id»: filename,
                «label»: model.names[int(cls_id)],
                «score»: round(float(score), 4),
                «box»: [round(float(x), 2) for x in box.tolist()] # [x1, y1, x2, y2]
                }
)
    except Exception as e:
        print(f»[!] Error processing {filename}: {e}»)
        continue
with open(
output_json, «w»
) as f:
    json.dump(results, f, indent=2)
```

Поданий фрагмент коду реалізує процес автоматизованого тестування моделі RT-DETR, використовуючи можливості бібліотеки ultralytics. На початковому етапі відбувається ініціалізація моделі шляхом завантаження її попередньо навчених ваг із файлу rtdetr-l.pt. Далі задається шлях до директорії з тестовими зображеннями та файл, у який буде збережено результати обробки. Такий підхід забезпечує гнучкість у конфігурації середовища та дозволяє адаптувати код до різних обсягів вхідних даних.

Основна частина коду реалізує цикл обходу всіх зображень у заданій директорії. Для кожного зображення виконується перевірка формату файлу, після чого здійснюється передбачення об'єктів за допомогою моделі. Результатом інференсу є набір об’єктів із координатами їхніх обмежувальних прямокутників, класовими мітками та значеннями довіри. У разі відсутності результатів або виникнення помилки відповідне зображення пропускається, що забезпечує стійкість виконання коду при обробці великих датасетів.

Усі отримані результати зберігаються у структурованому форматі JSON, де кожен елемент містить ідентифікатор зображення, назву передбаченого класу, значення впевненості моделі у передбаченні, а також координати прямокутника, що окреслює виявлений об'єкт. Такий формат є придатним для подальшої кількісної оцінки точності та повноти моделі за допомогою обраних метрик. Крім того, він забезпечує можливість візуалізації результатів, а також інтеграції з системами постобробки або аналізу ефективності моделей.

Таким чином, даний фрагмент коду є прикладом повноцінного експериментального інструмента, який дозволяє не лише протестувати модель RT-DETR на практиці, але й підготувати результати у вигляді, що відповідає вимогам дослідження. Він відіграє ключову роль у зборі емпіричних даних для подальшого аналізу ефективності архітектури та її порівняння з іншими підходами.

У результаті проведеного дослідження модель RT-DETR продемонструвала високий потенціал у задачах виявлення об'єктів, поєднуючи трансформерну архітектуру з ефективними механізмами обробки зображень у реальному часі. Основною перевагою даного підходу є здатність здійснювати точне позиціонування об'єктів на зображенні при збереженні високої швидкодії, що особливо важливо для систем із обмеженими часовими ресурсами або в умовах потокової обробки даних. Завдяки використанню end-to-end структури модель не потребує складної постобробки, що спрощує інтеграцію в прикладні системи.

Навчання моделі протягом 100 епох на тренувальній вибірці COCO дало змогу досягти збалансованого рівня узагальнення на нових зображеннях, зокрема з тестового набору. RT-DETR впевнено виявляла широкий спектр об’єктів навіть за умов часткового перекриття, змін масштабу чи перспективи. Особливо ефективною модель виявилася в обробці складних сцен з кількома об’єктами, де вона стабільно визначала межі кожного з них та класифікувала з високим рівнем довіри.

Автоматизоване тестування моделі на повному наборі зображень дозволило сформувати масштабну вибірку результатів, яка лягла в основу подальшого обчислення метрик якості. Аналіз попередніх передбачень виявив загалом стабільну роботу моделі на переважній більшості прикладів, за винятком окремих випадків, де спостерігались похибки через невідповідність вхідних розмірів. Проте ці випадки мають технічний характер і не зумовлені недоліками самої архітектури.

Узагальнюючи отримані результати, можна стверджувати, що RT-DETR є перспективною трансформерною моделлю для об'єктної детекції, яка поєднує точність, адаптивність та швидкість. Вона є особливо доцільною для застосування у задачах реального часу та може бути успішно використана як у наукових дослідженнях, так і в інженерних прикладних рішеннях комп’ютерного зору.

3.4.3 Запуск моделі YOLO

Натсупним етпом розглянуто використання ще одного підходу до комп’ютерного зору – моделі YOLOv8, яка реалізує швидку та точну багатокласову детекцію об’єктів на зображеннях. YOLO належить до класу моделей, здатних одночасно виконувати класифікацію та локалізацію кількох об’єктів у межах одного кадру. Це робить її більш придатною для завдань детекції в динамічних або багатосуб’єктних сценах.

Модель YOLOv8 була обрана як одна з найновіших і найоптимізованіших версій серії YOLO, що поєднує високу швидкість обробки зі збереженням прийнятного рівня точності. В експерименті було використано конфігурацію yolov8n.pt (nano), яка має найменший розмір серед доступних варіантів і дозволяє досягти високої продуктивності навіть на системах з обмеженими обчислювальними ресурсами. Ця модель поставляється в інтегрованій формі у бібліотеці Ultralytics, яка забезпечує простий у використанні API для завантаження моделей, виконання інференсу та обробки результатів.

Як і у випадку з попередніми моделями, перший крок полягав у тестуванні роботи YOLO на одному зображенні. Це дозволило перевірити стабільність встановлення, правильність завантаження моделі, відповідність форматів вхідних даних і структури результатів очікуваним. Після цього код був масштабований для обробки повного валідаційного набору COCO, що дозволило оцінити ефективність YOLO в умовах великого обсягу вхідної інформації.

Застосування YOLO дозволило доповнити експеримент ще одним підходом, орієнтованим на реальну локалізацію об’єктів з високою продуктивністю.

Для початкового ознайомлення з можливостями моделі YOLOv8 було реалізовано простий приклад її запуску на одному зображенні. Метою даного тестування було перевірити, наскільки коректно працює модель у

середовищі експерименту, чи правильно розпізнаються об’єкти, а також оцінити якість і точність локалізації без попередньої модифікації або інтеграції з додатковими модулями. Такий підхід дозволяє швидко переконатися в працездатності моделі перед її масштабуванням на великий обсяг даних.

У прикладі, наведеному в лістингу 3.5, завантажується попередньо натренована модель yolov8n.pt (варіант «nano»), після чого на неї подається одне зображення з набору COCO. Результат виконання інференсу автоматично виводиться на екран із нанесеними рамками навколо виявлених об’єктів та їхніми мітками. Це дозволяє візуально оцінити якість розпізнавання та ступінь відповідності між очікуваними та фактично знайденими об’єктами.

Лістинг 3.5 – Тестування YOLO

```
from ultralytics import YOLO
model = YOLO(«yolov8n.pt»)   # n = nano, s = small, m = medium, l = large
image_path = «000000026580.jpg»
results = model(image_path)
results[0].show()
```

Код, представлений у лістингу 3.5, демонструє найкомпактніший і найпростішій спосіб використання моделі об’єктної детекції в межах цього експерименту. Він складається лише з кількох рядків і потребує мінімальної конфігурації для виконання повноцінної детекції об’єктів на зображенні. Це стало можливим завдяки бібліотеці Ultralytics, яка надає високорівневий API для роботи з моделями серії YOLOv8.

У першому рядку здійснюється імпорт головного класу YOLO із пакету ultralytics. Цей клас відповідає як за завантаження моделі, так і за виконання інференсу та обробку результатів. Наступним кроком є створення об’єкта моделі з вказанням імені контрольного файлу ваг

yolov8n.pt. У даному випадку використовується конфігурація nano – найменша і найшвидша з доступних, що дозволяє швидко отримати результат навіть на комп’ютерах без потужного графічного процесора.

Далі код завантажує зображення за локальним і передає його безпосередньо у модель. Важливо відзначити, що бібліотека ultralytics автоматично обробляє вхідні дані: виконує масштабування, нормалізацію та приведення до необхідного формату тензора. Завдяки цьому користувачу не потрібно вручну виконувати жодних підготовчих етапів, як це було у випадку з DINO чи RT-DETR.

Після обробки модель повертає список результатів детекції – по одному елементу на кожне зображення. У прикладі опрацьовується лише одне зображення, тому одразу звертаємося до results, на якому викликається метод .show(). Цей метод відображає зображення з накладеними прямокутниками навколо знайдених об'єктів, включаючи їхні класові мітки та значення впевненості. Весь процес виконується автоматично без необхідності вручну обробляти вихідні тензори чи перетворювати координати. Результат наведено на рисунку 3.3.

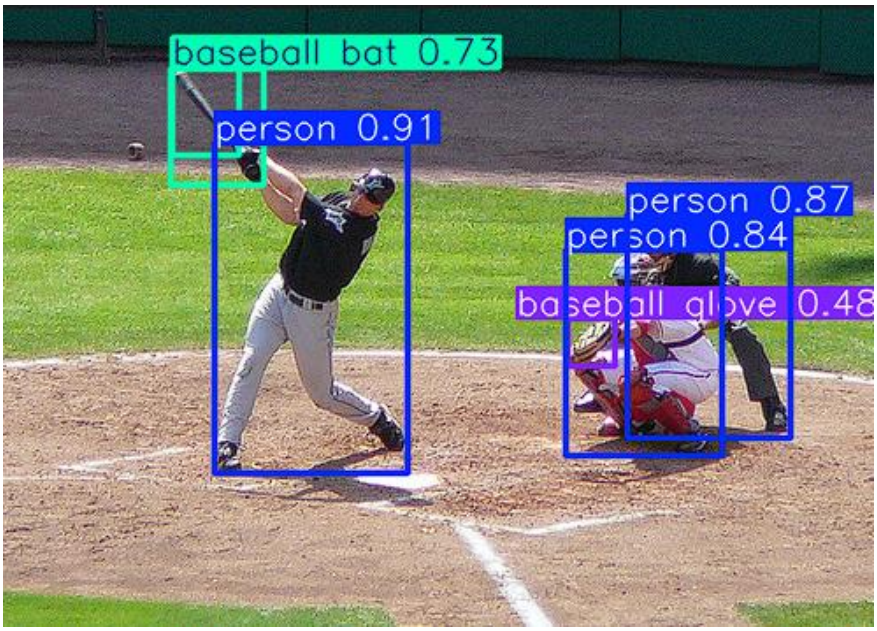

Рисунок 3.3 – Вихідне зображення, оброблене моделлю YOLOv8

Таким чином, цей код є найбільш зручним і лаконічним способом запуску об’єктної детекції у межах експерименту. Він демонструє ключову перевагу YOLOv8 – мінімальний вхідний бар’єр для початку роботи з глибоким навчанням у сфері комп’ютерного зору. Простота інтерфейсу робить модель доступною навіть для тих, хто не має глибокої технічної підготовки, і водночас дозволяє отримати високоякісний результат без зайвих налаштувань.

Після успішного запуску моделі YOLOv8 на одному зображенні та візуального підтвердження її коректної роботи, було здійснено наступний етап – масштабована обробка повного тестового набору зображень. Метою цього кроку стало автоматичне проходження моделі по всіх зображеннях валідаційного піднабору COCO з метою виявлення об’єктів, фіксації результатів детекції та збереження їх у форматі JSON. Такий формат є зручним для подальшої обробки, порівняння та статистичного аналізу.

Завдяки високому рівню абстракції, що надається бібліотекою Ultralytics, модифікація коду для пакетної обробки виявилася мінімальною. Модель yolov8n.pt була використана повторно, а логіка циклічного проходження по директорії із зображеннями дозволила масштабувати інференс на всю вибірку. Збереження кожного результату у вигляді словника з інформацією про ідентифікатор зображення, назву об’єкта, рівень впевненості та координати боксу забезпечило повноцінну структуру для подальшої інтеграції з іншими результатами експерименту. Код наведено на лістингу 3.6.

Лістинг 3.6 – Код основної частини експерименту моделі YOLO.

```
import os
import json
from ultralytics import YOLO
from tqdm import tqdm
model_path = «yolov8n.pt»
image_dir = «val2017/img»
```

Продовження лістингу 3.6

```
output_json = «yolo_results.json»
model = YOLO(model_path)
results = []
for filename in tqdm(sorted(os.listdir(image_dir))):
      if not filename.lower().endswith(('.jpg', '.jpeg', '.png')):
          continue
      image_path = os.path.join(image_dir, filename)
      try:
          yolo_result = model(image_path, verbose=False)[0]
          boxes = yolo_result.boxes
          if boxes is None:
              continue
                for box, score, cls_id in zip(boxes.xyxy, boxes.conf, boxes.cls):
              results.append({
                  «image_id»: filename,
                  «label»: model.names[int(cls_id)],
                  «score»: round(float(score), 4),
                       «box»: [round(float(x), 2) for x in box.tolist()]  # [x1, y1, x2, y2]
              })
      except Exception as e:
          print(f»[!] Error processing {filename}: {e}»)
          continue
with open(output_json, «w») as f:
      json.dump(results, f, indent=2)
```

Код, представлений у цьому прикладі, реалізує автоматизований запуск моделі YOLOv8 на повному наборі зображень з подальшим збереженням результатів детекції у форматі JSON. На початку виконуються імпорти основних бібліотек: os для роботи з файловою системою, json для збереження результатів, tqdm для відображення

прогресу виконання та ultralytics.YOLO – для ініціалізації та запуску моделі. Такий набір забезпечує повну функціональність для масової обробки зображень із мінімальними зусиллями.

У блоці налаштування вказується шлях до моделі (yolov8n.pt), директорія з тестовими зображеннями (val2017/img) та ім’я файлу, до якого буде записано результати (yolo_results.json). Після цього виконується завантаження моделі за допомогою одного рядка – створення об’єкта класу YOLO. Завдяки бібліотеці Ultralytics, модель одразу готова до використання без потреби в додаткових налаштуваннях чи ручній обробці вхідних даних.

Основна частина коду – цикл, який проходить по всіх файлах у вказаній директорії. Для кожного зображення перевіряється розширення, щоб уникнути обробки непідтримуваних форматів. Зображення передається у модель, і результат зчитується з першого (єдиного) елементу списку результатів. Якщо модель не знайшла жодного об’єкта (тобто boxes дорівнює None), зображення пропускається.

У випадку успішного виявлення об’єктів, дані зчитуються з об’єкта boxes, що містить координати прямокутників (xyxy), відповідні значення впевненості (conf) та номери класів (cls). Далі в циклі результати для кожного знайденого об’єкта формуються у вигляді словника, який включає назву зображення, мітку класу (отриману за допомогою model.names), значення впевненості та координати виявленого об’єкта. Ці словники послідовно додаються до списку results.

Після завершення обробки всі результати зберігаються у JSON-файл у зручному форматі. Така структура даних дозволяє легко обробити її засобами Python, порівняти з результатами інших моделей або візуалізувати. Код є лаконічним, гнучким та придатним до масштабування, що робить YOLOv8 з бібліотекою Ultralytics одним із найзручніших інструментів для швидкої реалізації задач детекції в умовах обробки великих обсягів зображень.

Модель YOLOv8 показала себе як високопродуктивне та зручне рішення для задач детекції об’єктів у реальному часі. Її головною перевагою є поєднання високої швидкості обробки, достатньої точності та простоти інтеграції завдяки зручному інтерфейсу, який надає бібліотека Ultralytics. Навіть найменша конфігурація yolov8n.pt (nano), що використовувалась у межах експерименту, продемонструвала впевнену роботу на великому наборі зображень без потреби у спеціальному налаштуванні чи попередній обробці даних.

YOLO дозволяє виявляти декілька об’єктів одночасно, локалізуючи кожен із них за допомогою прямокутної рамки. Це робить її значно більш придатною для задач, де на зображенні присутні кілька сутностей або складні сцени. У порівнянні з Grounding DINO, YOLO працює без текстових підказок і має фіксований набір класів, що, з одного боку, обмежує її гнучкість, але з іншого – забезпечує стабільну й передбачувану роботу.

Додатковою перевагою є уніфікований формат вихідних даних, який легко перетворюється у формат COCO або інтегрується в будь-який інструмент візуалізації чи статистичного аналізу. Збереження результатів у JSON-файл дозволило швидко перейти до наступних етапів – оцінювання якості виявлення, порівняння з іншими моделями та побудови метрик точності.

У підсумку, YOLOv8 виявилась ефективною базовою моделлю для задач детекції, яка поєднує простоту реалізації з високою практичною придатністю. Її використання в експерименті дозволило закріпити результати попередніх етапів та надати надійну точку порівняння для аналізу альтернативних архітектур.

3.5 Оцінювання та візуалізація результатів

Після отримання результатів від трьох попередньо розглянутих моделей – Grounding DINO, RT-DETR та YOLOv8 – наступним етапом стала оцінка якості їх роботи за обраними кількісними метриками. Це дозволило об'єктивно порівняти продуктивність моделей у контексті поставленого завдання та виявити їх сильні й слабкі сторони. Для побудови аналітичного зрізу було застосовано стандартні метрики, прийняті в задачах класифікації та об’єктної детекції.

Основними критеріями оцінювання стали:

– mAP@0.5 – середня точність при пороговому значенні IoU = 0.5, яка вважається базовим показником якості локалізації;

– mAP@[0.5:0.95] – усереднений показник точності при дев’яти різних порогах IoU від 0.5 до 0.95 з кроком 0.05, що дає повнішу оцінку загальної точності;

– IoU (Intersection over Union) – метрика перекриття між передбаченим та істинним боксами, яка визначає точність локалізації;

– F1-score – гармонійне середнє між точністю (precision) та повнотою (recall), що дозволяє оцінити баланс між виявленими й пропущеними об’єктами;

– Average Recall (AR) – середнє значення recall при різній кількості передбачених об'єктів, що вказує на здатність моделі виявляти всі об’єкти на зображенні.

Нижче будуть представлені графічні та табличні візуалізації обчислених значень зазначених метрик для кожної з моделей. Візуальний аналіз результатів дозволяє не лише побачити кількісну перевагу одних підходів над іншими, а й краще інтерпретувати якість моделей у прикладному контексті. Додатково будуть проаналізовані особливості поведінки моделей на складних прикладах, що ілюструють їхню стійкість до шуму, перекриття об'єктів і варіативності візуального представлення.

Першим кроком у процесі кількісної оцінки ефективності досліджуваних методів стало створення функції, яка реалізує обчислення метрики mAP@0.5 (mean Average Precision при пороговому значенні IoU = 0.5). Цей показник є одним із ключових стандартів у задачах детекції об’єктів, оскільки дозволяє об’єктивно оцінити здатність моделі не лише виявляти об'єкти на зображенні, а й правильно визначати їхнє місцезнаходження. Значення mAP@0.5 є усередненою величиною точності по всіх класах за умови, що коефіцієнт перетину об'єкта та передбаченої області перевищує 0.5.

Для реалізації оцінювання використовуються результати, отримані від моделей RT-DETR, YOLO та DINO, а також відповідні еталонні розмітки, що містять інформацію про координати об’єктів на тестовому наборі зображень. Після підрахунку коефіцієнтів IoU між передбаченими та справжніми межами об’єктів, визначається кількість правильних спрацьовувань (true positives), хибнопозитивних (false positives) і пропущених об’єктів (false negatives), що дозволяє обчислити точність і повноту для кожного класу, а згодом – і середню точність (AP).

Завдяки використанню метрики mAP@0.5 стало можливим здійснити порівняльний аналіз обраних архітектур і зробити обґрунтовані висновки щодо їхньої придатності для задачі виявлення перекритих об'єктів на зображеннях. Таке формалізоване оцінювання є невід’ємною частиною дослідницького процесу, оскільки дозволяє неупереджено зіставити моделі з різними принципами побудови та механізмами детекції.

Код методу наведений на лістингу 3.7.

Лістинг 3.7 – Функція оцінки за метрикою mAP@0.5

```
import os
import json
from collections import defaultdict
from sklearn.metrics import average_precision_score
import numpy as np

# === ШЛЯХИ ДО ФАЙЛІВ ===
```

Продовження лістингу 3.7

```
    annotation_dir = «val2017/ann»
    prediction_file = «dino_results.json»
    def iou(boxA, boxB):
        xA = max(boxA[0], boxB[0])
        yA = max(boxA[1], boxB[1])
        xB = min(boxA[2], boxB[2])
        yB = min(boxA[3], boxB[3])
        interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
        boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1]
+ 1)
        boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1]
+ 1)
            return interArea / float(boxAArea + boxBArea -
interArea)

    with open(prediction_file, «r») as f:
        predictions = json.load(f)
    pred_by_image = defaultdict(list)
    for pred in predictions:
            label = pred[«label»].lower().replace(«a «,
««).replace(«an «, ««).strip()
        pred[«label»] = label
        pred_by_image[pred[«image_id»]].append(pred)
    gt_boxes = defaultdict(list)
    for filename in os.listdir(annotation_dir):
        if not filename.endswith(«.json»):
            continue
        with open(os.path.join(annotation_dir, filename), «r»)
as f:
            data = json.load(f)
            for obj in data.get(«objects», []):
                if obj.get(«geometryType») != «polygon»:
                    continue
                label = obj[«classTitle»].lower().strip()
                points = obj[«points»][«exterior»]
                if len(points) < 2:
                    continue
                x_coords = [p[0] for p in points]
                y_coords = [p[1] for p in points]
                x_min, x_max = min(x_coords), max(x_coords)
                y_min, y_max = min(y_coords), max(y_coords)
                image_id = filename.replace(«.json», ««)
                gt_boxes[image_id].append({
                    «label»: label,
                    «box»: [x_min, y_min, x_max, y_max]
                })

    class_names = set([
    «person», «bicycle», «car», «motorcycle», «airplane», ])
```

Продовження лістингу 3.7

```
        «horse», «sheep», «cow», «elephant», «bear», «zebra»,
«giraffe», «backpack», «umbrella»,
         «handbag», «tie», «suitcase», «frisbee», «snowboard»,
«kite», «baseball bat», «baseball glove»,
         «skateboard», «surfboard», «tennis racket», «bottle»,
«wine glass», «cup», «fork», «knife»,
            «spoon», «bowl», «banana», «apple», «sandwich»,
«orange», «broccoli», «carrot», «hot dog»,
          «pizza», «donut», «cake», «chair», «couch», «potted
plant», «bed», «dining table», «toilet»,
         «tv», «laptop», «mouse», «remote control», «keyboard»,
«cell phone», «microwave», «oven»,
          «toaster», «sink», «refrigerator», «book», «clock»,
«vase»
     ])
     aps = []
     for class_name in class_names:
         y_true_class = []
         y_score_class = []
         for image_id in gt_boxes:
              gt_for_image = [x for x in gt_boxes[image_id] if
x[«label»] == class_name]
                          preds_for_image = [x for x in
pred_by_image.get(image_id, []) if x[«label»] == class_name]
             matched_gt = set()
             for pred in preds_for_image:
                 pred_box = pred[«box»]
                 score = pred[«score»]
                 match_found = False
                 for i, gt in enumerate(gt_for_image):
                     if i in matched_gt:
                         continue
                     iou_score = iou(pred_box, gt[«box»])
                     if iou_score >= 0.5:
                         matched_gt.add(i)
                         match_found = True
                         break
                 y_true_class.append(1 if match_found else 0)
                 y_score_class.append(score)
                       for i in range(len(gt_for_image) -
len(matched_gt)):
                 y_true_class.append(1)
                 y_score_class.append(0.0)
         if len(set(y_true_class)) > 1:
               aps.append(average_precision_score(y_true_class,
y_score_class))
     map_50 = np.mean(aps)
     print(f»\nmAP@0.5 = {map_50:.4f}»)
```

Наведений код реалізує обчислення метрики mAP@0.5 (mean Average Precision при IoU ≥ 0.5) для оцінювання якості роботи моделі об'єктної детекції. На початку завантажуються передбачення моделі з JSON-файлу, після чого вони групуються за зображеннями. Далі обробляються реальні анотації з відповідної директорії у форматі JSON – з полігональних координатів об'єктів виділяються граничні прямокутники (bounding boxes).

У наступному етапі для кожного класу зі списку COCO здійснюється пошук співпадінь між передбаченими та істинними об’єктами. Для кожного передбаченого об’єкта обчислюється коефіцієнт IoU з анотаціями, і якщо він перевищує 0.5 – передбачення вважається успішним. Отримані бінарні мітки (true/false positive) та відповідні оцінки впевненості моделі використовуються для обчислення Average Precision (AP) за кожним класом. Наприкінці обчислюється усереднене значення mAP@0.5 для всіх класів. Результат оцінки наведений на рисунку 3.4.

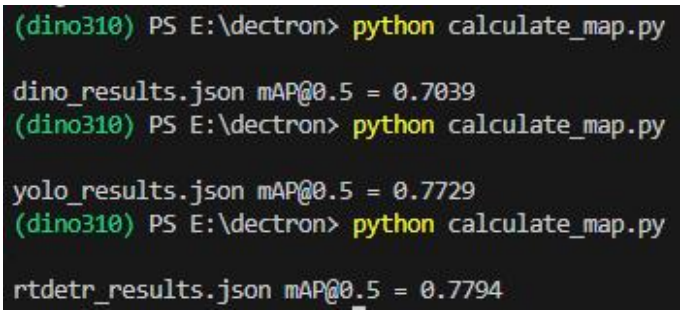

Рисунок 3.4 – Результат оцінювання моделей за метрикою mAP@0.5

Другим етапом дослідження стало створення функціонального механізму для оцінки якості роботи кожної з реалізованих моделей за складнішою метрикою – mAP@[0.5:0.95]. Ця метрика, рекомендована стандартом COCO, забезпечує більш глибоку та всебічну перевірку здатності моделей точно локалізовувати об'єкти різного розміру та ступеня перекриття. На відміну від базової mAP@0.5, яка враховує лише один

поріг IoU (intersection over union), розширений підхід mAP@[0.5:0.95] передбачає обчислення середнього значення точності (AP) на одинадцяти порогах, що змінюються від 0.5 до 0.95 з кроком 0.05. Це дозволяє уникнути спрощеного аналізу і виявити тонкі відмінності між поведінкою моделей при варіаціях точності сегментації об'єктів.

У межах реалізації була розроблена функція, яка аналізує відповідність передбачених об'єктів фактичним анотаціям з використанням різних порогів IoU. Для кожного з класів COCO обчислюється точність передбачень на кожному з одинадцяти рівнів порогу, після чого проводиться усереднення результатів для отримання значення AP для класу. Потім обчислюється середнє значення AP по всіх класах, що й утворює значення mAP для відповідного порогу. Фінальним результатом є усереднення одинадцяти mAP-показників, що й визначає остаточну оцінку ефективності моделі за mAP@[0.5:0.95].

Таким чином, використання цієї метрики дозволило здійснити порівняння моделей RT-DETR, DINO та YOLO не лише за здатністю виявляти об'єкти, а й за точністю локалізації в складних умовах, таких як часткові перекриття, малі розміри об'єктів чи неоднозначні контури. Це сприяє об'єктивному й надійному ранжуванню методів за реальними показниками якості.

Код наведено на лістингу 3.8.

Лістинг 3.8 – Функція оцінки за метрикою mAP@[0.5:0.95]

```
import os
import json
from collections import defaultdict
from sklearn.metrics import average_precision_score
import numpy as np

annotation_dir = «val2017/ann»
prediction_file = «rtdetr_results.json»

def iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
```

Продовження лістингу 3.8

```
        xB = min(boxA[2], boxB[2])
         yB = min(boxA[3], boxB[3])
        interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
         boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1]
+ 1)
         boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1]
+ 1)
            return interArea / float(boxAArea + boxBArea -
interArea
    with open(prediction_file, «r») as f:
        predictions = json.load(f)
    pred_by_image = defaultdict(list)
    for pred in predictions:
            label = pred[«label»].lower().replace(«a «,
««).replace(«an «, ««).strip()
        pred[«label»] = label
        pred_by_image[pred[«image_id»]].append(pred)
    gt_boxes = defaultdict(list)
    for filename in os.listdir(annotation_dir):
        if not filename.endswith(«.json»):
            continue
        with open(os.path.join(annotation_dir, filename), «r»)
as f:
            data = json.load(f)
            for obj in data.get(«objects», []):
                if obj.get(«geometryType») != «polygon»:
                    continue
                label = obj[«classTitle»].lower().strip()
                points = obj[«points»][«exterior»]
                if len(points) < 2:
                    continue
                x_coords = [p[0] for p in points]
                y_coords = [p[1] for p in points]
                x_min, x_max = min(x_coords), max(x_coords)
                y_min, y_max = min(y_coords), max(y_coords)
                image_id = filename.replace(«.json», ««)
                gt_boxes[image_id].append({
                    «label»: label,
                    «box»: [x_min, y_min, x_max, y_max]
                })
    class_names = set([obj[«label»] for objs in
gt_boxes.values() for obj in objs])
    ious = np.arange(0.5, 1.0, 0.05)
    aps_by_iou = []
    for iou_thresh in ious:
        aps = []
        for class_name in class_names:
            y_true_class = []
            y_score_class = []
            for image_id in gt_boxes:
```

Продовження лістингу 3.8

```
                gt_for_image = [x for x in gt_boxes[image_id]
if x[«label»] == class_name]
     preds_for_image = [x for x in pred_by_image.get(image_id,
[]) if x[«label»] == class_name]
                matched_gt = set()
                for pred in preds_for_image:
                    pred_box = pred[«box»]
                    score = pred[«score»]
                    match_found = False
                    for i, gt in enumerate(gt_for_image):
                        if i in matched_gt:
                            continue
                       if iou(pred_box, gt[«box»]) >= iou_thresh:
                            matched_gt.add(i)
                            match_found = True
                            break
                  y_true_class.append(1 if match_found else 0)
                    y_score_class.append(score)

            for i in range(len(gt_for_image) - len(matched_gt)):
                    y_true_class.append(1)
                    y_score_class.append(0.0)
             if len(set(y_true_class)) > 1:
                 aps.append(average_precision_score(y_true_clas
s, y_score_class))
         mean_ap = np.mean(aps) if aps else 0
         print(f»mAP@{iou_thresh:.2f} = {mean_ap:.4f}»)
         aps_by_iou.append(mean_ap)
     map_5095 = np.mean(aps_by_iou)
```

Наведений код реалізує оцінювання точності роботи моделей детекції об’єктів за розширеною метрикою mAP@[0.5:0.95], яка враховує одинадцять рівнів порогу перетину (IoU) – від 0.5 до 0.95 із кроком 0.05. На першому етапі виконання здійснюється завантаження результатів передбачення моделі та їх угрупування за ідентифікаторами зображень. Паралельно відбувається обробка файлів анотацій із тестового набору, де координати полігонів трансформуються у формат обмежувальних рамок (bounding boxes).

У подальшому код формує список унікальних класів об’єктів, після чого запускає основний цикл обчислення точності для кожного значення порогу IoU. Для кожного класу порівнюються передбачені рамки з

відповідними істинними об’єктами. У разі успішного збігу (тобто перевищення поточного порогу IoU) модель зараховує влучне передбачення. Після збору бінарних міток істинності та відповідних впевненостей проводиться обчислення середньої точності (AP) на основі average_precision_score.

Результати AP усереднюються по всіх класах, утворюючи mAP для кожного конкретного порогу. Завершальним кроком є обчислення остаточного показника mAP@[0.5:0.95] шляхом усереднення одинадцяти mAP-значень, що й надає узагальнену оцінку ефективності моделі на різних рівнях точності локалізації.

Результат оцінювання наведено на рисунку 3.5.

```
(dino310) PS E:\dectron> python calculate_map_range.py
mAP@0.50 = 0.7791
mAP@0.55 = 0.7706
mAP@0.60 = 0.7666
mAP@0.65 = 0.7505
mAP@0.70 = 0.7354
mAP@0.75 = 0.7230
mAP@0.80 = 0.7131
mAP@0.85 = 0.6598
mAP@0.95 = 0.4966

rtdetr_results.json mAP@[0.5:0.95] = 0.6972
(dino310) PS E:\dectron> python calculate_map_range.py
mAP@0.50 = 0.7681
mAP@0.55 = 0.7673
mAP@0.60 = 0.7681
mAP@0.65 = 0.7679
mAP@0.70 = 0.7600
mAP@0.75 = 0.7482
mAP@0.80 = 0.7268
mAP@0.85 = 0.7123
mAP@0.90 = 0.7083
mAP@0.95 = 0.6193

yolo_results.json mAP@[0.5:0.95] = 0.7346
```

Рисунок 3.5 – Результат оцінювання моделей за метрикою mAP@[0.5:0.95]

Наступним кроком у дослідженні стало створення функціонального блоку для оцінки ефективності роботи моделей виявлення об’єктів за метрикою IoU (Intersection over Union). Ця метрика є фундаментальною при аналізі якості локалізації об'єктів на зображеннях, оскільки дозволяє чисельно оцінити ступінь перекриття між передбаченими (predicted) і

еталонними (ground truth) обмежувальними рамками. Використання IoU дає змогу зрозуміти, наскільки точно модель визначає просторове розташування об'єкта.

У межах реалізованої функції було передбачено зчитування еталонних розміток та результатів передбачення, нормалізацію назв класів і обчислення IoU для кожної пари передбаченого та відповідного істинного об'єкта. Для кожного зображення обирається найкраще співпадіння – та рамка, яка має найбільше перекриття з ground truth-об’єктом і перевищує заданий поріг (як правило, 0.5). Такий підхід дозволяє уникнути множинного підрахунку однакових об'єктів і зберігає коректність оцінювання.

Результатом роботи функції є усереднене значення IoU по всіх зображеннях, які були успішно оброблені. Отриманий показник дозволяє провести об'єктивне порівняння між різними архітектурами, що використовуються у дослідженні, а також оцінити загальну якість локалізації без врахування семантичної класифікації об'єктів.

Код наведено на лістингу 3.9.

Лістинг 3.9 – Функція оцінки за метрикою IoU

```
from collections import defaultdict
import numpy as np
annotation_dir = «val2017/ann»
prediction_file = «dino_results.json»
iou_threshold = 0.5
def iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1]
+ 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1]
+ 1)
    unionArea = float(boxAArea + boxBArea - interArea)
    return interArea / unionArea if unionArea > 0 else 0
with open(prediction_file, «r») as f:
    predictions = json.load(f)
```

Продовження лістингу 3.9

```
    pred_by_image = defaultdict(list)
    for pred in predictions:
            label = pred[«label»].lower().replace(«a «,
««).replace(«an «, ««).strip()
        pred[«label»] = label
        pred_by_image[pred[«image_id»]].append(pred)
    gt_boxes = defaultdict(list)
    for filename in os.listdir(annotation_dir):
        if not filename.endswith(«.json»):
            continue
        with open(os.path.join(annotation_dir, filename), «r»)
as f:
            data = json.load(f)
            for obj in data.get(«objects», []):
                if obj.get(«geometryType») != «polygon»:
                    continue
                label = obj[«classTitle»].lower().strip()
                points = obj[«points»][«exterior»]
                if len(points) < 2:
                    continue
                x_coords = [p[0] for p in points]
                y_coords = [p[1] for p in points]
                x_min, x_max = min(x_coords), max(x_coords)
                y_min, y_max = min(y_coords), max(y_coords)
                image_id = filename.replace(«.json», ««)
                gt_boxes[image_id].append({
                    «label»: label,
                    «box»: [x_min, y_min, x_max, y_max]
                })
    all_ious = []
    for image_id in gt_boxes:
        gt = gt_boxes[image_id]
        preds = pred_by_image.get(image_id, [])
        matched_gt = set()
        for pred in preds:
            pred_box = pred[«box»]
            best_iou = 0.0
            for i, gt_obj in enumerate(gt):
                if i in matched_gt:
                    continue
                iou_score = iou(pred_box, gt_obj[«box»])
                if iou_score > best_iou:
                    best_iou = iou_score
                    best_match = i
            if best_iou >= iou_threshold:
                matched_gt.add(best_match)
                all_ious.append(best_iou)
```

Наведений код реалізує обчислення середнього значення метрики IoU (Intersection over Union) для передбачених моделей результатів виявлення об'єктів, порівнюючи їх із вручну розміченими еталонними даними. На початку здійснюється завантаження передбачених результатів та анотацій з локальної директорії, а також формуються словники з відповідними обмежувальними рамками для кожного зображення.

Основна логіка полягає у проходженні по кожному зображенню, де для кожної передбаченої рамки обирається найбільш наближене (з найбільшим IoU) ground truth-співпадіння, яке ще не було використане. Якщо значення IoU перевищує фіксований поріг (у даному випадку 0.5), такий збіг вважається коректним і враховується при підрахунку.

Наприкінці усі знайдені значення IoU усереднюються, після чого виводиться загальний показник середнього перекриття для заданої моделі. Таким чином, ця процедура дозволяє об'єктивно оцінити точність локалізації об'єктів незалежно від класифікаційного компонента.

Результат наведено на рисунку 3.6.

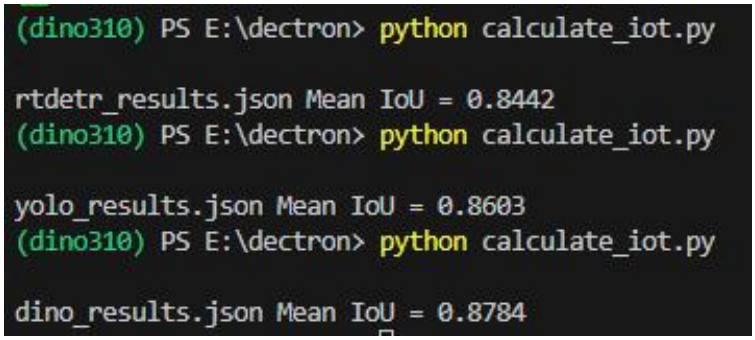

Рисунок 3.6 – Результат оцінювання за моделей метрикою IoU

Наступним етапом дослідження стало впровадження функціоналу для оцінки ефективності роботи моделей виявлення об’єктів за допомогою метрики F1-score. Цей показник дає змогу визначити баланс між точністю (precision) та повнотою (recall), що є критично важливим при аналізі якості класифікації об’єктів на зображеннях. Особливо це

актуально в умовах, коли спостерігається суттєвий дисбаланс між кількістю виявлених об’єктів різних класів або ж коли важливо не лише мінімізувати кількість помилкових спрацьовувань, але й не пропустити значущі об’єкти.

Оцінка за F1-метрикою реалізована шляхом порівняння передбачених моделей обмежувальних рамок із реальними координатами об’єктів на зображеннях. Було встановлено порогове значення IoU ≥ 0.5, при якому передбачена рамка вважається правильною, якщо вона не лише достатньо перекриває реальний об’єкт, а й відповідає правильному класу. Після проходження кожного передбачення порівнюється з відповідним ground truth, обчислюються TP (істинно позитивні), FP (хибнопозитивні) та FN (хибнонегативні) значення, на основі яких формується значення F1.

Застосування F1-score дозволило забезпечити додаткову глибину аналізу в рамках порівняння архітектур RT-DETR, YOLO та DINO, демонструючи не лише здатність моделі точно класифікувати об'єкти, а й її ефективність у комплексному розпізнаванні наявних класів на зображеннях. Це особливо важливо для повноцінного і об'єктивного порівняння різних підходів у завданні виявлення об’єктів. Код методу наведений на лістингу 3.10.

Лістинг 3.10 – Функція оцінки за метрикою F1-Score

```
  annotation_dir = «val2017/ann»
prediction_file = «dino_results.json»
def iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1]
+ 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1]
+ 1)
        return interArea / float(boxAArea + boxBArea -
interArea)
with open(prediction_file, «r») as f:
    predictions = json.load(f)
```

Продовження лістингу 3.10

```
    pred_by_image = defaultdict(list)
    for pred in predictions:
     label      =      pred[«label»].lower().replace(«a      «,
««).replace(«an «, ««).strip()
        pred[«label»] = label
        pred_by_image[pred[«image_id»]].append(pred)
    gt_boxes = defaultdict(list)
    for filename in os.listdir(annotation_dir):
        if not filename.endswith(«.json»):
            continue
with open(os.path.join(annotation_dir, filename), «r») as f:
            data = json.load(f)
            for obj in data.get(«objects», []):
                if obj.get(«geometryType») != «polygon»:
                    continue
                label = obj[«classTitle»].lower().strip()
                points = obj[«points»][«exterior»]
                if len(points) < 2:
                    continue
                x_coords = [p[0] for p in points]
                y_coords = [p[1] for p in points]
                x_min, x_max = min(x_coords), max(x_coords)
                y_min, y_max = min(y_coords), max(y_coords)
                image_id = filename.replace(«.json», ««)
                gt_boxes[image_id].append({ «label»: label,
                    «box»: [x_min, y_min, x_max, y_max] })
    class_names    =    set([«person»,    «bicycle»,    «car»,
«motorcycle», «airplane», «bus», «train», «truck», «boat»])
    aps = []
    for class_name in class_names:
        y_true_class = []
        y_score_class = []
        for image_id in gt_boxes:
             gt_for_image = [x for x in gt_boxes[image_id] if
x[«label»] == class_name]
    preds_for_image = [x for x in pred_by_image.get(image_id,
[]) if x[«label»] == class_name]
            matched_gt = set()
            for pred in preds_for_image:
                pred_box = pred[«box»]
                score = pred[«score»]
                match_found = False
                for i, gt in enumerate(gt_for_image):
                    if i in matched_gt:
                        continue
                    iou_score = iou(pred_box, gt[«box»])
                    if iou_score >= 0.5:
                        matched_gt.add(i)
              match_found = True
                        break
                y_true_class.append(1 if match_found else 0)
```

Продовження лістингу 3.10

```
    y_score_class.append(score)
                        for i in range(len(gt_for_image) -
len(matched_gt)):
                    y_true_class.append(1)
                    y_score_class.append(0.0)
          if len(set(y_true_class)) > 1:
                aps.append(average_precision_score(y_true_class,
y_score_class))
    map_50 = np.mean(aps)
```

Наведений код реалізує обчислення метрики F1-score для оцінки якості виявлення об’єктів за результатами моделі. Він порівнює передбачені рамки з реальними для кожного зображення, використовуючи поріг IoU ≥ 0.5 як критерій відповідності. Якщо клас передбаченої рамки збігається з класом GT-об'єкта, і перекриття перевищує порогове значення, передбачення вважається вдалим (True Positive).

В процесі виконання формується два масиви: один для істинних значень (ground truth), інший – для передбачених. До першого потрапляють одиниці у випадку вдалих збігів, а також у випадку пропущених об’єктів (False Negative). До другого – одиниці для кожного передбачення та нулі для пропущених GT. Після обробки всіх зображень, масиви використовуються для обчислення F1-score за допомогою функції f1_score з бібліотеки sklearn, що дозволяє узагальнено оцінити збалансованість точності й повноти моделі.

Результат наведено на рисунку 3.7.

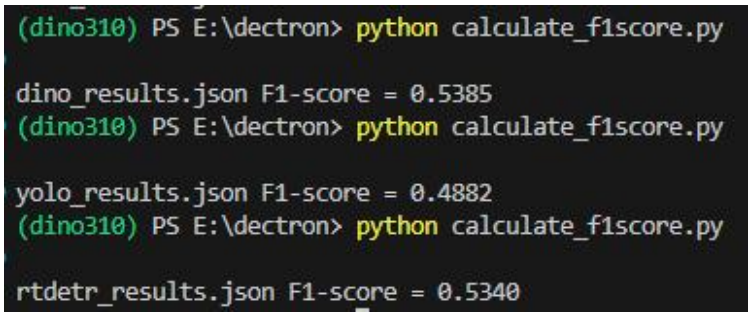

Рисунок 3.7 – Результат оцінювання за метрикою F1-Score

Наступним етапом у межах експериментального дослідження було створено функцію для оцінювання ефективності роботи досліджуваних архітектур RT-DETR, YOLO та DINO за метрикою Average Recall (AR). Застосування даного показника дозволяє визначити здатність моделі виявляти максимальну кількість релевантних об’єктів, не зважаючи на кількість хибно позитивних передбачень. На відміну від точності, яка фокусується на правильності передбачених об’єктів, метрика AR акцентує увагу саме на повноті виявлення, що особливо важливо в задачах, де критично важливо не пропустити жоден із цільових об'єктів.

Для реалізації оцінювання було розроблено програмну функцію, яка проходить по кожному зображенню тестового набору та порівнює передбачені модельні об'єкти з еталонними (GT). Якщо клас передбачення співпадає з класом GT, і при цьому коефіцієнт перекриття (IoU) перевищує встановлений поріг (у даному випадку 0.5), об’єкт вважається правильно знайденим. Усі релевантні об'єкти, які не були передбачені або були виявлені з недостатнім перекриттям, класифікуються як пропущені. Підсумкове значення метрики AR обчислюється як відношення правильно знайдених об'єктів до загальної кількості еталонних об'єктів. Отримане значення дозволяє кількісно оцінити повноту виявлення об'єктів та є корисним доповненням до попередніх метрик, таких як mAP та IoU.

Код методу наведений на лістингу 3.11.

Лістинг 3.11 – Функція оцінки за метрикою Average Recall

```
annotation_dir = «val2017/ann»
prediction_file = «dino_results.json»
iou_threshold = 0.5
def iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1]
+ 1)
```

Продовження лістингу 3.11

```
        boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1]
+ 1)
        unionArea = float(boxAArea + boxBArea - interArea)
        return interArea / unionArea if unionArea > 0 else 0
    with open(prediction_file, «r») as f:
        predictions = json.load(f)
    pred_by_image = defaultdict(list)
    for pred in predictions:
            label = pred[«label»].lower().replace(«a «,
««).replace(«an «, ««).strip()
        pred[«label»] = label
        pred_by_image[pred[«image_id»]].append(pred)
    gt_boxes = defaultdict(list)
    for filename in os.listdir(annotation_dir):
        if not filename.endswith(«.json»):
            continue
        with open(os.path.join(annotation_dir, filename), «r»)
as f:
            data = json.load(f)
            for obj in data.get(«objects», []):
                if obj.get(«geometryType») != «polygon»:
                    continue
                label = obj[«classTitle»].lower().strip()
                points = obj[«points»][«exterior»]
                if len(points) < 2:
                    continue
                x_coords = [p[0] for p in points]
                y_coords = [p[1] for p in points]
                x_min, x_max = min(x_coords), max(x_coords)
                y_min, y_max = min(y_coords), max(y_coords)
                image_id = filename.replace(«.json», ««)
                gt_boxes[image_id].append({
                    «label»: label,
                    «box»: [x_min, y_min, x_max, y_max]})
    total_gt = 0
    matched_gt = 0
    for image_id in gt_boxes:
        gt = gt_boxes[image_id]
   preds = pred_by_image.get(image_id, [])
        total_gt += len(gt)
        used = set()
        for gt_idx, gt_obj in enumerate(gt):
            for pred in preds:
                if pred[«label»] != gt_obj[«label»]:
                    continue
                iou_score = iou(pred[«box»], gt_obj[«box»])
                if iou_score >= iou_threshold:
                    matched_gt += 1
                    break
    average_recall = matched_gt / total_gt if total_gt > 0
else 0.0
```

Наведений код реалізує обчислення метрики Average Recall (AR) для моделі обробки зображень, використовуючи передбачення (prediction_file) та відповідні анотації (annotation_dir). У процесі обробки кожне зображення аналізується шляхом порівняння передбачених об’єктів із еталонними (GT), при цьому використовується метрика IoU з фіксованим порогом (0.5).

Для кожного GT-об’єкта перевіряється наявність хоча б одного передбачення того ж класу, яке має достатнє перекриття (IoU ≥ 0.5). Якщо таке передбачення знайдено, об’єкт вважається правильно виявленим. Загальна кількість таких відповідностей ділиться на загальну кількість GT-об’єктів, що дозволяє отримати значення Average Recall. Це значення відображає здатність моделі виявляти наявні об’єкти незалежно від точності їх локалізації.

Результат наведено на рисунку 3.8.

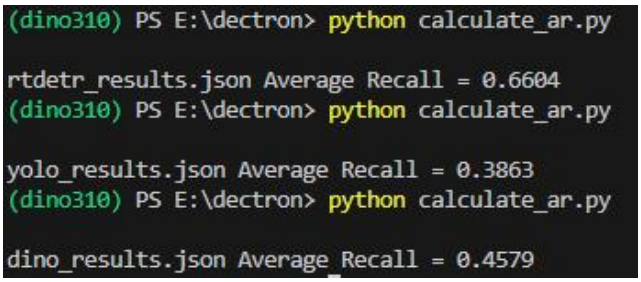

Рисунок 3.8 – Результат оцінювання за метрикою Average Recall

У межах проведеного експериментального дослідження було реалізовано комплексну оцінку якості роботи моделей виявлення об’єктів за п’ятьма ключовими метриками: mAP@0.5, mAP@[0.5:0.95], IoU, F1-score та Average Recall. Кожна з них надає унікальну перспективу на ефективність досліджуваних архітектур: RT-DETR, YOLOv8 і Grounding DINO.

Показник mAP@0.5 є базовим стандартом в оцінці моделей об'єктного детектування, оскільки характеризує точність локалізації

об’єктів з урахуванням порогу перекриття 0.5. Проте він не враховує варіативність складності задачі, тому було додатково проведено обчислення mAP@[0.5:0.95], який відображає усереднену точність при зростаючих значеннях IoU. Ця метрика є більш чутливою до помилок локалізації та дозволяє провести глибше порівняння між моделями.

Метрика IoU дозволила безпосередньо оцінити якість просторового перекриття між передбаченими та еталонними рамками, що має важливе значення в задачах, де точна локалізація критично важлива. Розрахунок F1-score забезпечив балансовану оцінку між точністю та повнотою виявлення, дозволяючи враховувати як помилкові спрацьовування, так і пропущені об'єкти. Нарешті, Average Recall охарактеризував здатність моделей повноцінно виявляти всі наявні об'єкти незалежно від їхнього класу чи розміру.

Загальна оцінка за всіма метриками дала змогу всебічно порівняти моделі не лише за загальною точністю, а й за здатністю до надійної локалізації, виявлення малих об'єктів, уникнення хибнопозитивних передбачень та загального охоплення. Такий підхід дозволив ідентифікувати сильні й слабкі сторони кожної з архітектур у різних аспектах реального застосування.

3.6 Порівняння результатів та інтерпретація

На завершальному етапі дослідження було проведено порівняльний аналіз отриманих результатів роботи трьох моделей – RT-DETR, YOLOv8 та Grounding DINO – з метою виявлення переваг, недоліків та загальної ефективності кожної з них у задачі виявлення накладених об'єктів. У попередньому етапі було проведено обчислення п’яти метрик: mAP@0.5, mAP@[0.5:0.95], IoU, F1-score та Average Recall, які й лягли в основу подальшого порівняння.

Кожна з розглянутих моделей продемонструвала унікальні характеристики в контексті точності локалізації, стабільності виявлення об'єктів різної складності та адаптивності до щільних сцен з накладеннями. Порівняння значень метрик дозволило не лише кількісно визначити, яка з моделей показала найвищі результати, але й здійснити якісну інтерпретацію поведінки моделей на тестовому наборі даних. Зокрема, було виявлено, наскільки кожна з моделей чутлива до перекриттів, змін масштабу, присутності кількох об'єктів одного класу та шумових даних.

Результати, отримані за підсумками проведеного експерименту, були зведені до порівняльної таблиці з метою полегшення аналізу ефективності розглянутих архітектур. До таблиці було включено значення усіх п’яти метрик, обчислених для моделей RT-DETR, YOLOv8 та Grounding DINO. Таблиця 3.1 відображає точні числові показники кожної з моделей, що дозволяє наочно оцінити їх сильні та слабкі сторони.

У результаті експериментального дослідження було встановлено, що жодна з моделей не домінує за всіма метриками одночасно. За показником mAP@[0.5:0.95], найкращий результат продемонструвала модель YOLOv8, що свідчить про її загальну узгодженість на різних порогах IoU. У той час модель RT-DETR забезпечила найвищу точність за метрикою Average Recall, що вказує на її здатність виявляти більшу кількість об’єктів. Модель DINO показала перевагу за метрикою IoU, демонструючи найточніше накладення передбачених рамок на реальні об’єкти.

Таблиця 3.1 – Порівняння моделей за визначеними метриками якості

Метрика	RT-DETR	YOLOv8	DINO
mAP@0.5	0.7794	0.7729	0.7039
mAP@[0.5:0.95]	0.6972	0.7346	0.6934
IoU	0.8442	0.8603	0.8784
F1-score	0.5340	0.4882	0.5385
Average Recall	0.6604	0.3863	0.4579

Показник F1-score також був найвищим у DINO, що вказує на хороший баланс між точністю та повнотою. Водночас YOLOv8 виявилася найменш ефективною за цією метрикою, попри високе значення mAP@[0.5:0.95], що може свідчити про більшу кількість хибнопозитивних спрацювань. Таким чином, кожна модель має свої переваги в залежності від конкретної мети застосування.

Згідно з даними, наведеними у сформованій таблиці (таблиця 3.1), було побудовано п’ять окремих графіків, кожен з яких відображає зміну значення відповідної метрики для трьох порівнюваних моделей: RT-DETR, YOLOv8 та DINO. Для кожної з метрик – mAP@0.5, mAP@[0.5:0.95], IoU, F1-score та Average Recall – було створено окрему діаграму стовпчикового типу, що дозволяє наочно порівняти ефективність досліджуваних архітектур.

Відповідно до обраної структури роботи, отримані діаграми представлено на рисунках 3.9-3.13. Кожен з цих рисунків відповідає одній із метрик і дає змогу порівняти рівень точності чи повноти виявлення об’єктів для розглянутих моделей.

На рисунку 3.9 представлено порівняння моделей RT-DETR, YOLOv8 та DINO за метрикою mAP@0.5. Згідно з графіком, найвищий показник точності досягнуто для моделі RT-DETR (0.779), незначно поступається їй YOLOv8 (0.773), тоді як модель DINO продемонструвала нижчий результат – 0.704. Це свідчить про дещо вищу ефективність RT-DETR і YOLOv8 у точковому виявленні об’єктів при допустимому перекритті 50%.

На рисунку 3.10 наведено результати моделей за метрикою mAP@[0.5:0.95], яка є більш суворим показником точності. Найкращий результат продемонструвала модель YOLOv8 з оцінкою 0.735. Моделі RT-DETR і DINO показали схожі, але дещо нижчі результати – 0.697 і 0.693 відповідно. Це свідчить про загальну стабільність YOLOv8 на різних рівнях перекриття.

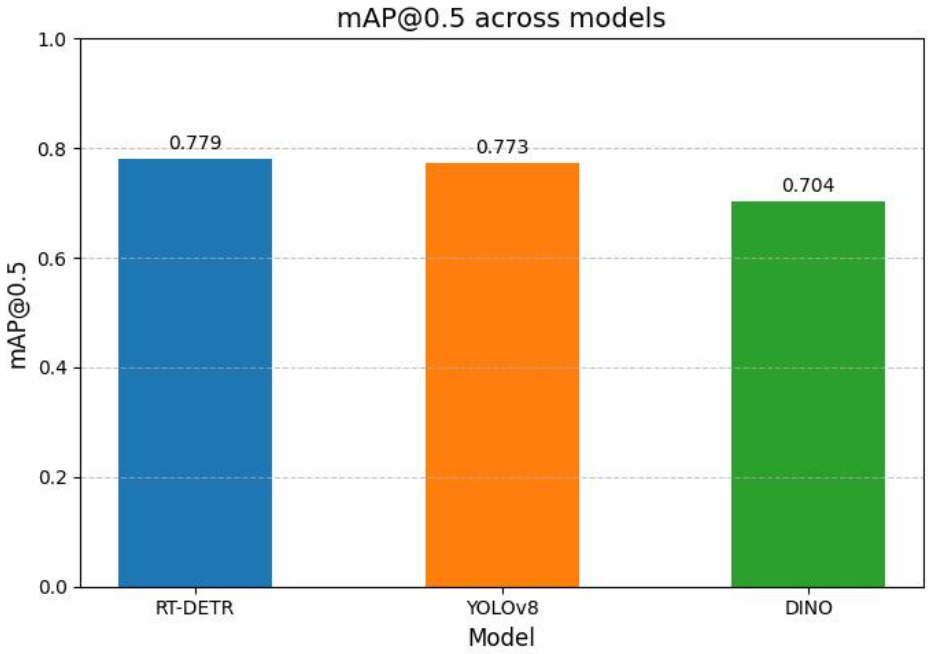

Рисунок 3.9 – Порівняльний аналіз моделей за метрикою mAP@0.5

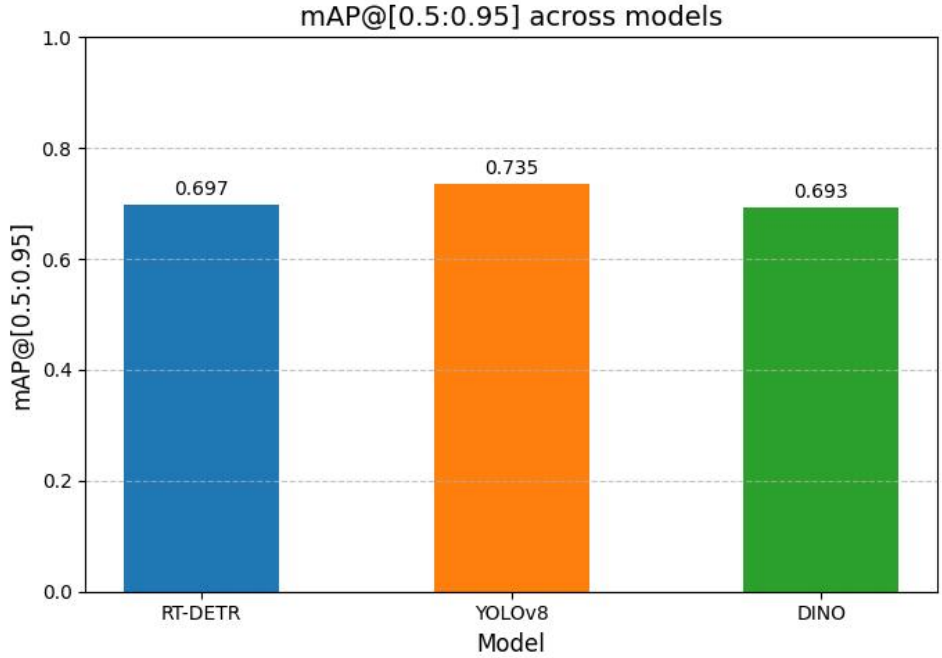

Рисунок 3.10 – Порівняльний аналіз моделей за метрикою mAP@[0.5:0.95]

На рисунку 3.11 показано значення метрики IoU (Intersection over Union) для кожної з моделей. Найвищий результат продемонструвала модель DINO (0.878), що свідчить про її здатність точно локалізовувати

об’єкти. За нею йдуть YOLOv8 (0.860) і RT-DETR (0.844), які також показали високий рівень збігу передбачених і реальних об’єктів.

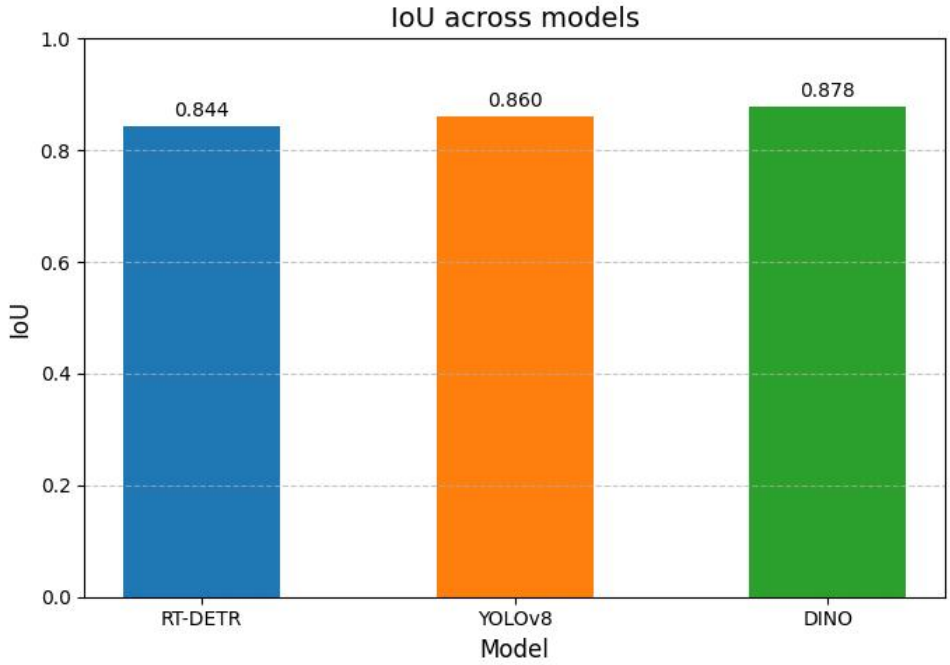

Рисунок 3.11 – Порівняльний аналіз моделей за метрикою IoU

На рисунку 3.12 наведено значення метрики F1-score, яка враховує як точність, так і повноту виявлення. Найкращий результат продемонструвала модель DINO (0.538), що свідчить про збалансовану ефективність виявлення об'єктів. RT-DETR показала майже ідентичний показник (0.534), тоді як YOLOv8 дещо відстала з результатом 0.488.

На рисунку 3.13 представлено порівняння моделей за метрикою Average Recall, яка відображає здатність моделі виявляти всі об'єкти у зображеннях. Найвищий результат показала модель RT-DETR (0.660), що свідчить про її високу чутливість. Модель DINO продемонструвала середній рівень (0.458), тоді як YOLOv8 досягла найнижчого значення (0.386), що вказує на меншу здатність виявляти всі наявні об'єкти.

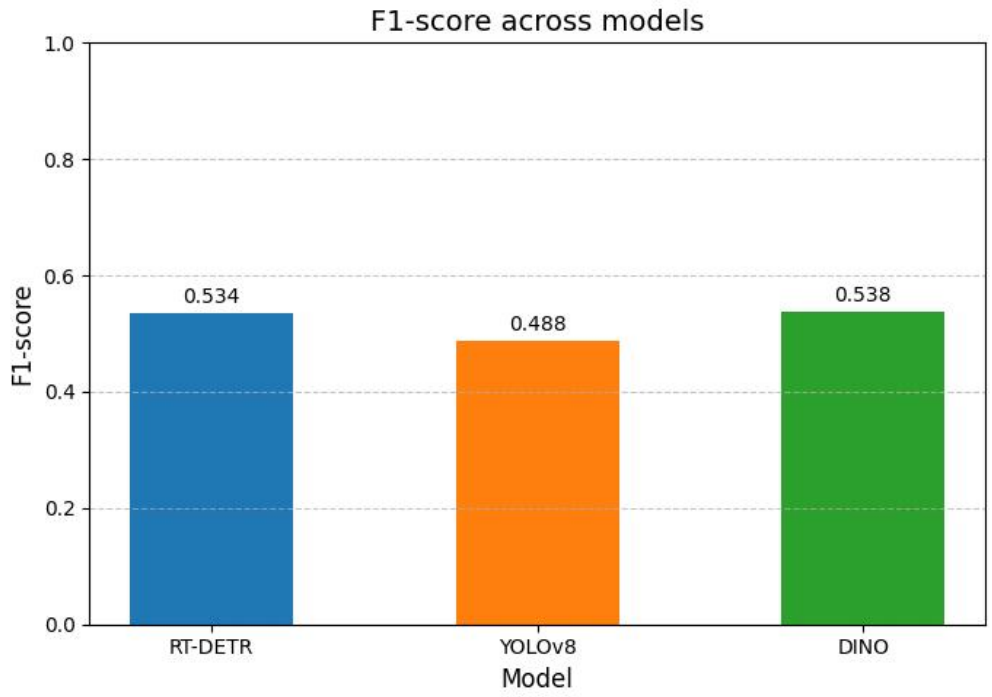

Рисунок 3.12 – Порівняльний аналіз моделей за метрикою F1-score

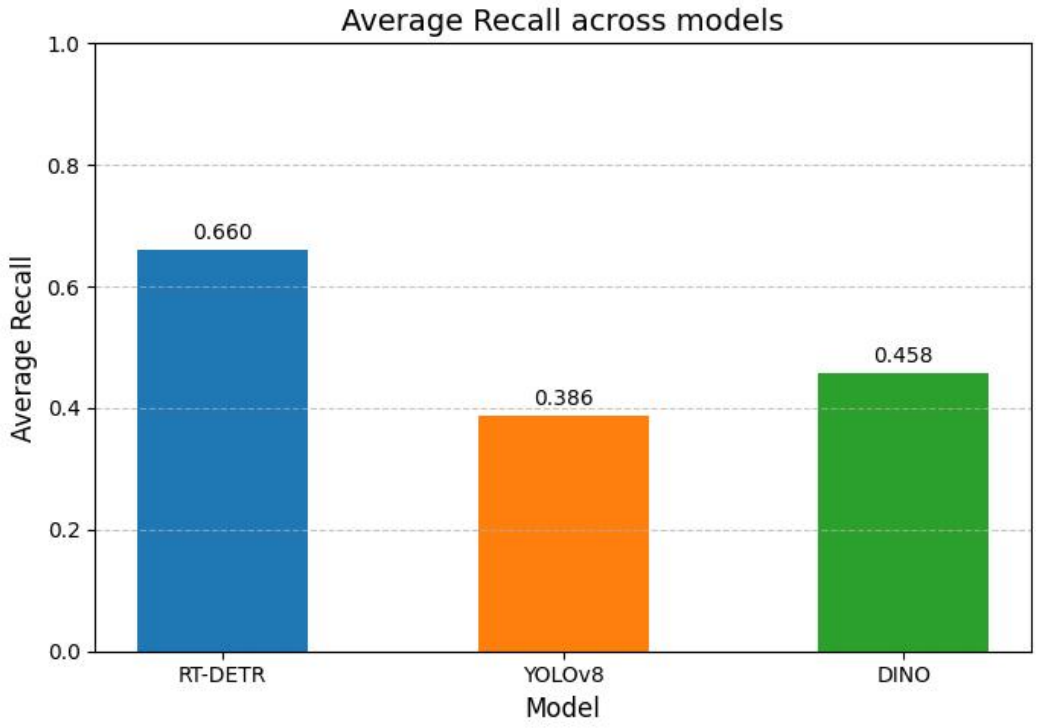

Рисунок 3.13 – Порівняльний аналіз моделей за метрикою Average Recall

На рисунку 3.14 зображена зведена діаграма оцінки по обраним метрикам для кожної з моделей, що була протестована під час проведення експериментів.

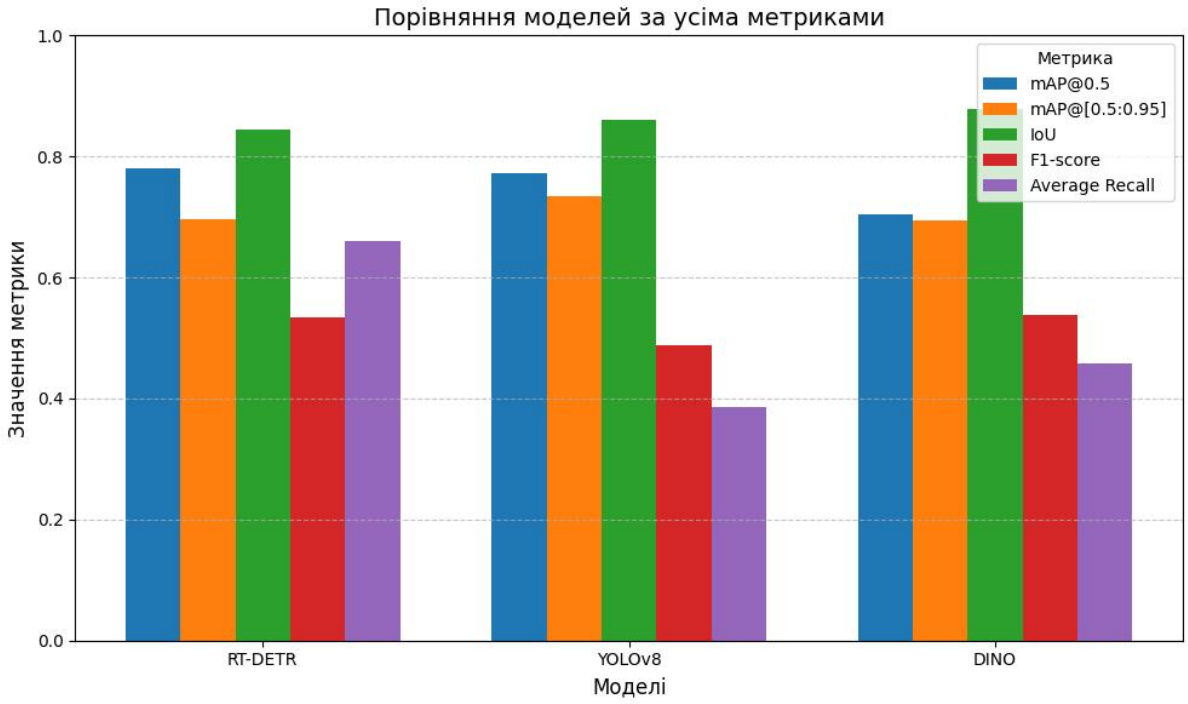

Рисунок 3.14 – Зведена оцінка моделей за метриками точності

На зведеному рисунку представлено порівняльний аналіз трьох моделей – RT-DETR, YOLOv8 та DINO – за п’ятьма метриками: mAP@0.5, mAP@[0.5:0.95], IoU, F1-score та Average Recall. Дана візуалізація дозволяє одночасно оцінити ефективність кожної моделі за всіма критеріями, що суттєво спрощує інтерпретацію результатів.

Згідно з графіком, модель DINO демонструє найвищий показник IoU, що свідчить про високу точність локалізації об'єктів. Водночас вона поступається за метрикою Average Recall, що вказує на меншу здатність виявляти повний набір об'єктів. YOLOv8 показує найкращі результати за метрикою mAP@[0.5:0.95], що свідчить про її збалансовану ефективність у складних умовах, однак відстає в F1-score та Recall. Модель RT-DETR демонструє найбільш збалансовані показники, зокрема високий Average

Recall та хороший рівень F1-score, що робить її придатною для завдань, де критичне як виявлення, так і точність локалізації.

У результаті експериментального дослідження було проведено комплексне оцінювання трьох сучасних архітектур глибокого навчання для задачі детекції об'єктів – RT-DETR, YOLOv8 та DINO – за п’ятьма ключовими метриками: mAP@0.5, mAP@[0.5:0.95], IoU, F1-score та Average Recall. Одержані значення метрик дозволяють здійснити багатовимірний аналіз продуктивності моделей та зробити обґрунтовані висновки щодо їх ефективності.

Модель YOLOv8 показала найвищий результат за узагальненою метрикою mAP@[0.5:0.95], що свідчить про її здатність демонструвати стабільну якість передбачень при різних порогах збігу. Водночас, за метрикою Average Recall, яка оцінює здатність моделі виявляти всі об’єкти на зображенні, ця архітектура поступається RT-DETR та DINO. Це вказує на її високу точність, однак нижчу повноту у виявленні.

Модель DINO продемонструвала найвищу середню IoU, що характеризує її здатність до точної локалізації об'єктів. Вона також отримала найкраще значення F1-score, що свідчить про баланс між точністю та повнотою передбачень. Незважаючи на це, DINO дещо поступилася конкурентам у mAP-метриках, що свідчить про можливу чутливість до невеликих змін порогів.

Модель RT-DETR забезпечила найкращий результат за метрикою Average Recall, що підкреслює її здатність виявляти максимальну кількість об’єктів. Вона також продемонструвала високі результати за mAP@0.5 та F1-score, що робить її добре збалансованим варіантом між точністю, повнотою та локалізаційною якістю.

Отже, аналіз результатів показує, що YOLOv8 є оптимальною для застосувань із високими вимогами до точності при різних рівнях порогів, DINO – для задач із високими вимогами до локалізації та збалансованості, а RT-DETR – у випадках, коли важлива повнота виявлення об’єктів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи на тему «Дослідження методу комп’ютерного зору для виявлення накладання об’єктів на зображеннях» було здійснено комплекс дослідницьких та прикладних етапів, спрямованих на теоретичне обґрунтування, реалізацію та експериментальне порівняння сучасних підходів до виявлення об'єктів із частковим перекриттям у зображеннях.

На першому етапі було проведено аналітичний огляд літературних джерел і сучасних методів у галузі комп’ютерного зору, зокрема щодо задач інстанс-сегментації та обробки зображень із частковим накладанням об’єктів. Особливу увагу приділено трансформерним моделям та архітектурам, здатним до локалізації та класифікації декількох об'єктів одночасно.

У рамках підготовчого етапу було обґрунтовано вибір трьох актуальних моделей для порівняльного дослідження: RT-DETR, YOLOv8 та Grounding DINO, які були адаптовані для виконання інференсу в реальному часі. Було завантажено попередньо навчені ваги моделей, налаштовано середовище виконання з використанням фреймворків Transformers, Ultralytics і Torch, а також реалізовано інструменти для завантаження і передобробки зображень.

Далі було сформовано тестовий набір даних на основі COCO-підмножини, який містить зображення з багатьма об’єктами в кадрі та значним рівнем перекриття. Було реалізовано автоматичний обхід тестового набору та збереження результатів предикції кожної моделі у форматі JSON, що уніфікувало дані для подальшого аналізу.

На етапі оцінювання точності роботи моделей було створено окремі модулі для обчислення таких метрик як mAP@0.5, mAP@[0.5:0.95], IoU, F1-score та Average Recall. Визначення метрик проводилось шляхом порівняння передбачених боксів із розміткою, поданою у форматі COCO-

совісних JSON-анотацій. Для кожної моделі були зібрані відповідні результати, які згодом були внесені до порівняльної таблиці.

Наступним кроком стала побудова графічних візуалізацій, які включали окремі стовпчикові діаграми для кожної метрики та зведену порівняльну діаграму. Це дало змогу проаналізувати сильні та слабкі сторони моделей у розрізі кожного з критеріїв.

У підсумку, проведене дослідження дозволило визначити, що модель YOLOv8 демонструє найвищу узагальнену точність, Grounding DINO – найкращу локалізацію об'єктів, а RT-DETR – найвищу повноту виявлення. Залежно від конкретних умов задачі – таких як рівень перекриття, потреба в реальному часі чи баланс між recall та precision – може бути обрано відповідну модель.

Таким чином, поставлену мету дослідження було досягнуто, а практичні результати можуть бути використані в подальших роботах з покращення систем виявлення об’єктів у складних візуальних сценах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Caron M., Touvron H., Misra I., Jégou H., Mairal J., Bojanowski P., Joulin A. Emerging Properties in Self-Supervised Vision Transformers // *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 11–17 Oct. 2021, Montreal, QC, Canada. – IEEE, 2021. – P. 9630–9640. – DOI: 10.1109/ICCV48922.2021.00951.

2. Wang S., Xia C., Lv F., Shi Y. RT-DETRv3: Real-Time End-to-End Object Detection with Hierarchical Dense Positive Supervision // *Proceedings of the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 4–8 Jan. 2025, Tucson, AZ, USA. – IEEE, 2025. – P. 1628–1636. – DOI: 10.1109/WACV61041.2025.00166.

3. Varghese R., M. S. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness // *Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024, Chennai, India. – IEEE, 2024. – P. 1–6. – DOI: 10.1109/ADICS58448.2024.10533619.

4. Rezatofighi H., Tsoi N., Gwak J., Sadeghian A., Reid I., Savarese S. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression // *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15–20 June 2019, Long Beach, CA, USA. – IEEE, 2019. – P. 658–666. – DOI: 10.1109/CVPR.2019.00075.

5. Mean Average Precision (mAP) Explained / Built In. URL: https://builtin.com/articles/mean-average-precision (date of access: 26.04.2025).

6. Kirillov A., He K., Girshick R., Rother C., Dollár P. Panoptic Segmentation // *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15–20 June 2019, Long Beach, CA, USA. – IEEE, 2019. – P. 9396–9405. – DOI: 10.1109/CVPR.2019.00963.

7. Chen L.-C., Papandreou G., Kokkinos I., Murphy K., Yuille A. L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous

Convolution, and Fully Connected CRFs // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. – 2018. – Vol. 40, No. 4. – P. 834–848. – DOI: 10.1109/TPAMI.2017.2699184.

8. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A. C. SSD: Single Shot MultiBox Detector // *In: Leibe B., Matas J., Sebe N., Welling M. (eds). Computer Vision ECCV*. – 2016. – Lecture Notes in Computer Science, vol. 9905. – Springer, Cham, 2016. – P. 21–37. – DOI: 10.1007/978-3-319-46448-0_2.

9. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. – 2017. – Vol. 39, No. 6. – P. 1137–1149. – DOI: 10.1109/TPAMI.2016.2577031.

10. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale / arXiv. URL: https://arxiv.org/abs/2010.11929 (date of access: 30.04.2025).

11. Microsoft COCO: Common Objects in Context / arXiv. URL: https://arxiv.org/abs/1405.0312 (date of access: 30.04.2025).

12. Cordts M., Omran M., Ramos S., Rehfeld T., Enzweiler M., Benenson R., Franke U., Roth S., Schiele B. The Cityscapes Dataset for Semantic Urban Scene Understanding // *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016. – P. 3213–3223. – DOI: 10.1109/CVPR.2016.350.