Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Кваліфікаційна робота

за спеціальністю «Комп'ютерні науки» – 122

освітній ступінь – бакалавр

на тему: «Пришвидшена розмітка даних для виявляння об'єктів»

Керівник кваліфікаційної роботи

кандидат фіз.-мат. наук, доцент

Швай Надія Олександрівна

_____

(підпис)

"____" _____ 2024 р.

Виконала студентка 4 курсу

Лахтюк А.В.

"____" _____ 2024 р.

Київ 2024

Зміст

Анотація

У роботі досліджено методи пришвидшення розмітки зображень для виявлення об’єктів. Проаналізовано їх ефективність та подано загальний підсумковий огляд доступних інструментів. Окрім того, центральною частиною роботи є створення власного застосунку з функціоналом, що дозволяє виконувати пришвидшене анотування об’єктів на зображеннях.

Вступ

Одним із важливих завдань розробки моделей комп’ютерного зору є розмітка даних для виявлення об’єктів. Системи, що потребують виконання цього типу розмітки широко використовуються в медицині, системах спостереження та оповіщення, автомобілях із функцією автопілоту тощо. Розробка подібних систем та забезпечення ефективності їх роботи вимагають застосування великих наборів даних, що повинні належним чином бути розмічені.

Актуальність, наукове та практичне значення обраної теми

Разом зі стрімким розвитком сфери штучного інтелекту, машинного навчання, комп’ютерного зору продовжує зростати потреба в анотуванні даних. Процес розмітки є ресурсозатратним і часто займає значну частину часу призначеного на розробку проєкту. Для того аби натренувати нову модель потрібно використати велику кількість якісно анотованих даних. Точність моделі отриманої в результаті тренування значною мірою залежить саме від того наскільки чітко було розмічено вхідні дані. Мануальна розмітка (розмітка виконана людиною) має свої переваги та недоліки. До переваг належать: розуміння необхідного контексту та неоднозначно виражених деталей, можливість контролю якості даних та їх розмітки, вміння обробки складних даних та швидке реагування на нові умови, проте недоліки також є досить вагомими: швидкість виконання є

низькою і відповідно забирає багато часу, людина у ролі анотатора даних може бути упередженою і таким чином створювати похибки в розмітках, через часозатратність може стати неможливим реагування на різкі зміни. З іншого боку автоматична розмітка даних може бути швидкою, дешевшою та більш доступною, але не має можливості розпізнавати необхідний контекст, не передбачає контролю якості та потребує постійного оновлення використовуваних систем. Саме тому існує необхідність в частковій оптимізації та, як наслідок, пришвидшенні виконання цього завдання. Основна ідея полягає в поєднанні переваг обох методів та забезпеченні контролю розмітки даних людиною з одночасним пришвидшенням виконання завдання за рахунок часткового та, ймовірно, недостатньо точного розпізнавання об'єктів на зображеннях.

Розмітка даних для виявлення об'єктів наразі є дуже актуальною галуззю, що постійно розвивається та потребує додаткових досліджень та покращень. Відповідно до наведених раніше пояснень пришвидшення розмітки представляє широке поле для досліджень. Окрім того, відповідно до даних представлених у Report "Data Collection and Labelling" Overview [1] ринок розмітки зображень та застосунків, сервісів для виконання цього завдання буде стабільно збільшуватися щонайменше до 2030 року, а розмітка відео та зображень займає 35% від усього ринку, що є найбільшим показником.

Розділ 1

1.1 Розмітка даних для виявлення об'єктів

Анотування (розмітка) даних – це процес маркування або додавання метаданих до зображення призначений для того, щоб надати додаткову інформацію про вміст зображення та як наслідок зробити його придатним до використання в тренуванні моделей машинного навчання.

Анотування даних для завдання виявлення об’єктів полягає у створенні відповідних метаданих у вигляді обмежувальних коробок для тих об’єктів, що присутні на зображенні з подальшим виокремленням координат розмічених елементів з показниками типів об’єктів у окремі структури.

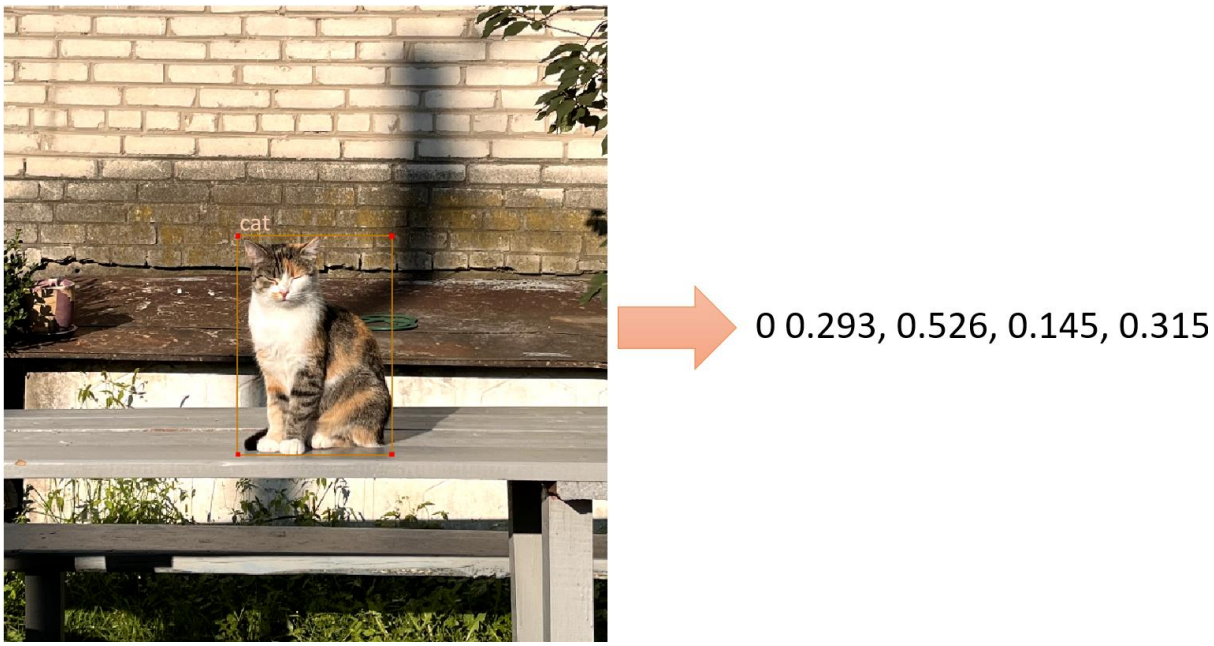

Рисунок 1. Приклад даних отриманих у результаті розмітки для виявлення об’єктів

У результаті процесу розмітки зображень шляхом додавання обмежувальних коробок в залежності від мети та моделі, що буде спроектована або використана з-поміж претренованих для подальшого тренування або, відповідно, тонкого настроювання, отримані дані можуть бути збережені у різних форматах. Найпоширенішими є txt, json та csv. Метадані можуть зберігатися в одному файлі, що містить інформацію відразу про всі зображення в наборі даних або в окремих файлах для кожного зображення або в кількох файлах відповідно до призначення.

Пришвидшення анотування полягає у зменшенні часу затраченого на розмітку даних для того чи іншого завдання та спрощенні самого процесу.

1.2 Методи пришвидшення розмітки зображень

Для процесу виявлення об’єктів у залежності від типу даних можна виокремити розмітку зображень та відео даних. Розмітка відео даних

передбачає подібну процедуру до тієї, що й при розмітці зображень, проте, відео може бути тривалістю, наприклад, 5 хвилин, fps 30, тож у такому разі необхідно буде проанотувати 5*60*30=9000 зображень, на кожному зображенні будуть виявленні різні між собою об’єкти при використанні прямого підходу виявлення об’єктів. Саме тому для розмітки відео даних, тобто кадрів, з яких відео складається використовують методи відстеження об’єкту.

Відстеження об’єктів полягає в їх виявленні, ідентифікації і подальшому відстежуванні їх траєкторій. Воно виконується шляхом аналізу кожного кадру, щоб визначити об'єкт, про який йде мова, і додати навколо нього обмежувальну рамку. Об'єкт ефективно відстежується протягом усього відео, шляхом виконання цієї операції на всіх кадрах. Відстеження об’єктів широко використовується в системах вбудованих в самокеровані автомобілі, для забезпечення відстеження руху та стану об’єктів дослідження в середовищі існування, та в різного роду систем спостереження тощо.

З-поміж підходів до реалізації відстеження об’єктів можна виокремити 2 категорії: відстеження одного об’єкту та відстеження кількох об’єктів. Для виконання цього завдання використовують алгоритми з використанням фільтру Калмана , зсув середнього (mean shift), постійно адаптивний зсув середнього, згорткові кореляційні фільтри, DeepSORT та ін. Вони переважно працюють таким чином, що відбувається початкове виявлення об’єктів із подальшим застосуванням певного алгоритму для їх відстеження, в подібних системах варто враховувати можливість виникнення нових об’єктів на зображенні, якщо завдання стосується виявлення кількох об’єктів.

Пришвидшити розмітку зображень можна за рахунок використання претренованих моделей, що наявні у відкритому доступі. Великі моделі,

натреновані на значних обсягах даних часто можуть розпізнавати десятки класів об’єктів. Також в залежності від завдання, що необхідно виконати, від об’єктів, які потрібно розпізнавати на зображеннях можна самостійно натренувати модель на наборі даних чи кількох наборах, що вже розмічені та викладені у відкритий доступ. Проте, це завдання вимагає часу та певних обчислювальних потужностей, а також умінь працювати з даними.

Для певних специфічних завдань, що потребують виконання виявлення унікальних типів об’єктів можливе використання методу, що передбачає початкову розмітку незначної кількості даних з подальшою генерацією синтетичних, розмічених даних. Надалі з використанням цих даних необхідно виконати тренування моделі для виявлення об’єктів, щоб використовувати її під час анотування з можливістю редагувати отримані метадані.

Ще одним способом пришвидшення розмітки даних є використання претренованої моделі та виконання її тонкого настроювання в процесі розмітки. Таким чином розмічені дані, що представляють частину загального набору даних передаються на вхід нейронній мережі для виконання тонкого настроювання після чого ця модель з певним невеликим показником точності, проте може використовуватися для пришвидшення розмітки даних, що наявні в наборі та не були розмічені до цього. Процес можна повторювати кілька разів щоразу покращуючи показники моделі та швидкість розмітки [2]. Проте, такий спосіб не є ефективним, так як вимагає чималих обчислювальних потужностей і регулярного тренування.

Використання мульти-модальних претренованих нейронних мереж, що можуть визначати об’єкти на зображенні відповідно до їх текстового опису (часто лише назви). Прикладом такої моделі може бути GroundingDINO[3][4], що на основні вхідних зображення та фрази (тексту,

що позначає необхідний об’єкт чи об’єкти, що мають бути визначені на зображенні) визначає певну кількість знайдених об’єктів. Цей спосіб є важливим кроком на шляху до автоматизації розмітки даних, оскільки робить це завдання найменш залежним від втручання людини, все, що потрібно подати на вхід: зображення та назви об’єктів. Окрім того, досить новим методом реалізації систем для виявлення об’єктів є системи, що пристосовані для визначення об’єктів відкритого набору даних.

Отже, методи пришвидшення розмітки зображень для виявлення об’єктів можна поділити на 2 категорії в залежності від типу даних: розмітка зображень та розмітка відео. Друга категорія не виключає можливості використання методів із першої, так як при анотуванні кадрів відео наявна необхідність виявляти нові об’єкти на окремих кадрах. Також з-поміж методів, які базуються на використанні претренованих моделей можна виокремити: використання претренованих моделей, що наявні у відкритому доступі, тренування моделі на незначній кількості даних (розмічених людиною або завантажених з наявних у відкритому доступі, або розмічених людиною з наступною генерацією синтетичних даних) з подальшим тонким настроюванням. Окрім того, ці методи можна класифікувати за типом архітектури, що характерна використовуваній моделі. Окремою категорією варто визначити використання моделей, що уможливлюють виявлення об’єктів з відкритого набору даних.

Розділ 2

2.1 Застосунок для виконання пришвидшеної розмітки зображень

Практична частина роботи – це створений з використанням PyQt6 анотатор призначений для ефективного виконання розмітки даних. Процес пришвидшено за рахунок використання претренованих моделей машинного навчання, що дозволяють скоротити час розмітки за рахунок

виявлення об’єктів внаслідок пропускання зображень через модель, попередньо обрану користувачем.

Використані бібліотеки:

- PyQt6 – інструментарій GUI Qt6 для Python [5]
- opencv-python – інструментарій OpenCV для Python
- pandas – додаткові структури даних та функції, корисні для використання в машинному навчанні
- uuid – генерування унікальних ідентифікаторів
- scikit-image – бібліотека для обробки зображень
- ultralytics – бібліотека для роботи з моделями YOLO [6]

Базова структура застосунку складається з основного вікна, поділеного на частини – основне поле, верхня панель та права панель.

Основне поле, що представлене базовим класом `ImageAnnotator` містить зображення та обмежувальні рамки. Верхня панель надає можливість відкрити одне зображення, папку із зображеннями, папку із зображеннями та їх анотаціями (у відповідному форматі), зберегти анотації, перейти в режим розмітки, перейти в режим редагування розмітки, видалити об’єкти з правої панелі та вибрати претреновану модель зі списку для автоматичної обробки зображення нею.

Права панель складається з таких частин:

- список типів об’єктів – надає можливість створювати нові типи та використовувати їх для розмітки об’єктів у основному полі;
- список даних про окремі обмежувальні рамки – містить необхідні дані про всі рамки, присутні на основному полі (їх ідентифікатори, типи антованих об’єктів та відносні координати обмежувальних полей)

- список зображень, що можуть бути відкриті та проанотовані.

Створення нових класів для анотування відбувається за рахунок введення даних у поле «Add Label» та натискання клавіші Enter. Після цього новий клас з'являється нижче у розділі «Labels List» та може використовуватись для необхідних операцій із зображенням.

У режимі анотування програма відображає позицію курсора. Лише в цьому режимі користувач має можливість зробити власні обмежувальні рамки, що будуть відображатись на правій панелі у відповідному розділі. Також з'являється можливість відредагувати вже визначені рамки, для цього потрібно натиснути кнопку верхньої панелі «Edit» та на основному полі виконати операції з переміщення (натискаючи та перетягуючи курсором всередині візуалізації коробки) або зміни розміру (виконуючи ті самі операції, але на одному із кутків коробки). Після цього зміни в даних автоматично відображаються у відповідних об'єктів справа та записуються в дані про поточну сесію.

Кожен елемент зі списків на правій панелі може бути видалений відповідною кнопкою на верхній панелі. Для цього потрібно розставити відповідні позначки на потрібних елементах списків та натиснути «Delete» на верхній панелі. Видалення об'єктів зі списку зображень видаляє їх лише з панелі, залишаючи об'єкти на диску.

Анотатор підтримує експорт метаданих у 3 типах txt, JSON та csv. Перший є типовим для тренування моделей YOLO [6] його загальна структура виглядає так: $n_i\ x_j\ y_j\ w_j\ h_j$, де a_i – номер призначений певному класу чи типу об'єктів, i – номер типу об'єкту, із загальної кількості n, що мають бути визначені (ця кількість може збільшуватися в процесі анотування зображень x_j , y_j – відносні координати середини обмежувальної коробки, що позначає об'єкт a_i , j – номер обмежувальної

коробки із загальної кількості обмежувальних коробок визначених на зображенні, w_j h_j – відносні ширина та висота обмежувальної коробки.

JSON формат є COCO [7]

```
{
    "images": [
        {
            "file_name": “img1.jpg”
            "height": 460,
            "width": 640,
            "id": 0
        }….
    ],
    "categories": [
        {
            "supercategory": "",
            "id": 1,
            "name": "Object category"
        }
    ],
    "annotations": [
        {
            "id": 1,
            "image_id": 0,
            "category_id": 1,
            "bbox": [
                110.0,
                67.0,
                194.0,
                264.0
            ]
         },….
     ]
}
```

А також csv формат, що дозволяє зберігати дані конкретної сесії розмітки у вигляді:

'bbox uuid’, 'image name', 'label name', 'label index', 'bx', 'by', 'bw', 'bh'

де 'bbox uuid' - унікальне id обмежувальної коробки, 'image name' – назва зображення, 'label name' – назва типу, класу об'єкта, 'label index' - номер типу об'єкту, із загальної кількості n, що мають бути визначені, 'bx', 'by', 'bw', 'bh' – відносні координати x, у середину обмежувальної коробки та відносні ширина і висота відповідно.

2.2 Імплементовані методи пришвидшення розмітки

Задля можливості виконання пришвидшеної розмітки зображень для виявлення об'єктів в межах створеного застосунку було обрано дві моделі глибинного навчання та додано функціонал для передавання зображення їм на вхід і оброблення отриманого результату. Перша модель yolov8 [6] претренована на наборі даних COCO та може розпізнавати 79 категорій поширених об'єктів, також ця модель може бути тонко настроєна для виявлення додаткових категорій об'єктів.

2.2.1 Yolov8

Yolo (you only look once) – це сімейство моделей з архітектурою, що базується на CNN (Convolutional Neural Network), широко використовуються для вирішення завдань комп'ютерного зору: сегментація, класифікація, виявлення об'єктів.

Опис архітектури yolo

Наразі точна структура yolov8 з її описом та деталями не була представлена авторами в офіційній статті чи в певному дослідженні. Проте, репозиторій з вихідним кодом доступний для перегляду. [8]

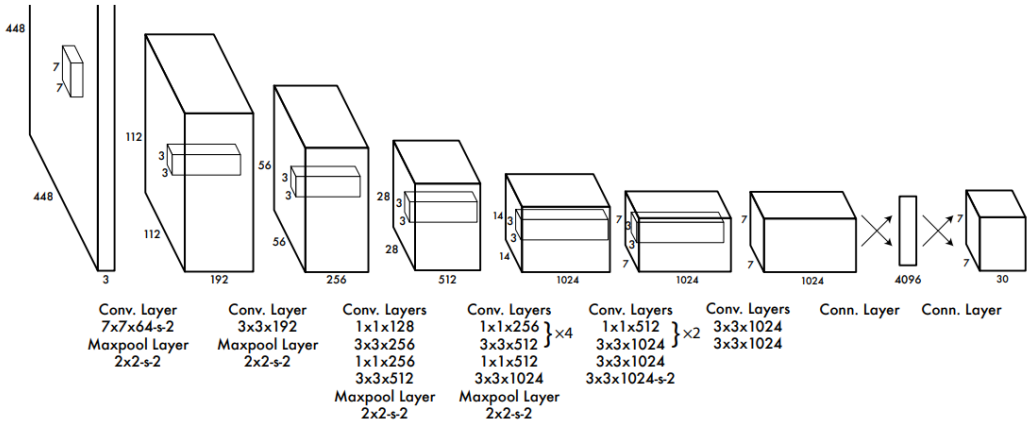

Рисунок 2. Базова архітектура YOLO [9]

На рисунку зображено базову архітектуру моделей YOLO, що була представлена ще в 2015 році [9] . Вона містить 24 згорткових шари, після яких наявні ще 2 повноз'єднаних шари. Архітектура моделі yolov8 дуже подібна до тієї, що представлена з моделлю yolov5 і містить певні модифікації. Структурно модель 5 покоління містить backbone, neck та head [10]. У свою чергу backbone представляє Darknet-53, neck з'єднує backbone із head та виконує завдання об'єднання та уточнення ознак отриманих із backbone та підсилення семантичної та просторової інформації. Складена з 3 потоків head, виконує на кожному передбачення обмежувальних рамок об'єктів на різних розмірах решітки та на останньому етапі виконується NMS (Non-maximum Suppression) для фільтрації отриманих обмежувальних коробок з виключенням значного перекриття.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Рисунок 3. Архітектура Darknet-53 [11]

Yolov8 є anchor-free моделлю, що дозволяє визначати безпосередньо розташування об’єкту без необхідності обрахування зсуву відносно певного anchor-box. У свою чергу це зменшує кількість об’єктів, що передбачаються моделлю загалом та пришвидшує виконання NMS.[20]

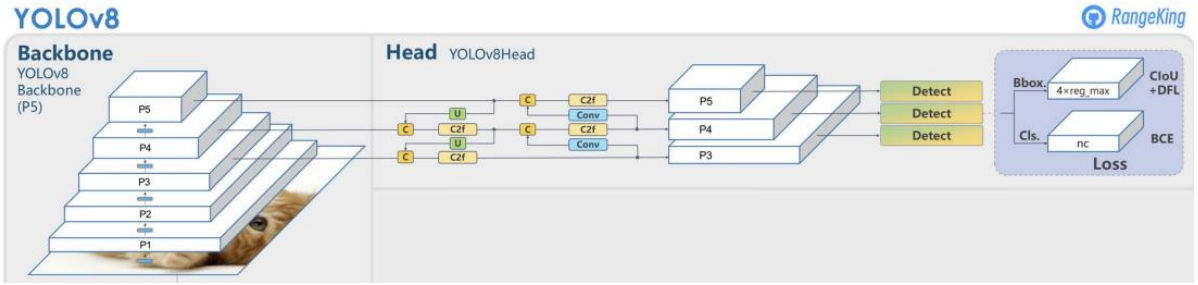

Рисунок 4. Структура моделі YOLOv8[12]

2.2.2 GroundingDino

Опис архітектури GroundingDino

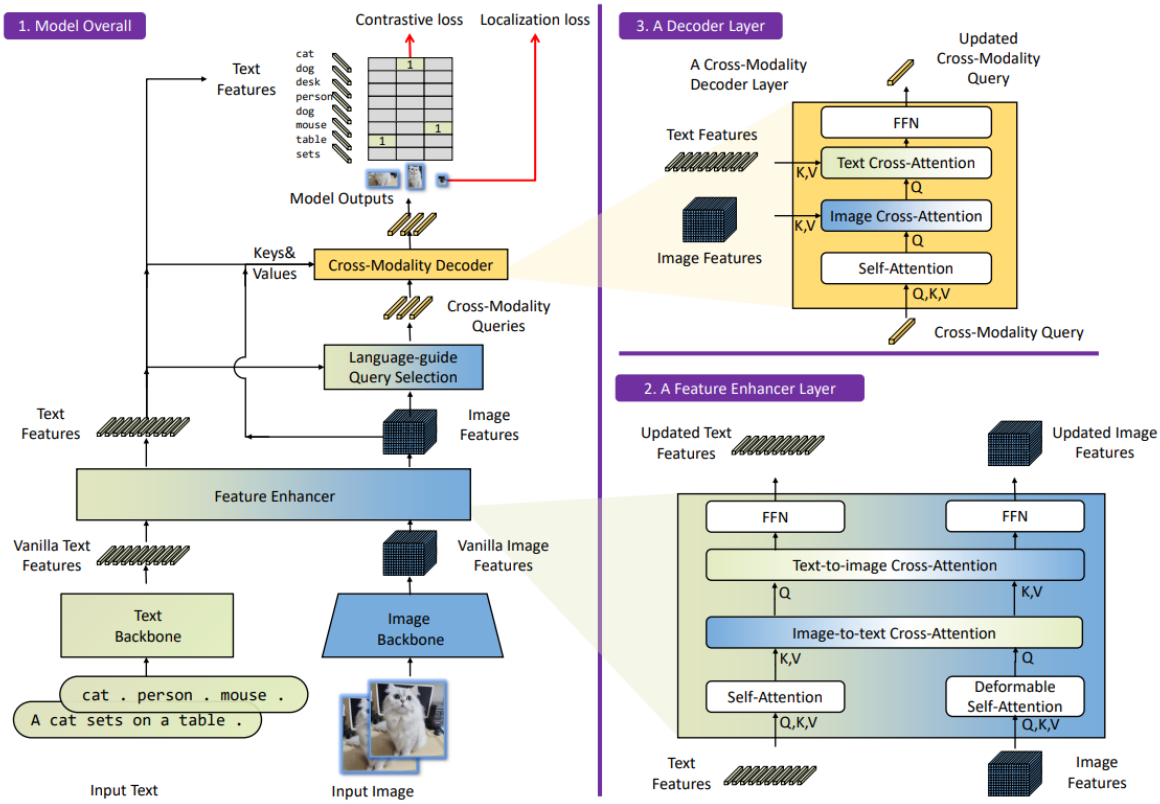

Рисунок 5. Архітектура моделі GroundingDINO

GroundingDino представляє систему, що може визначати на зображеннях довільні об’єкти базуючись на вхідному текстовому запиті [13]. Її унікальна архітектура базується на використанні трансформерів, що робить простішою обробку даних різних типів: зображень та тексту. DINO є моделлю з архітектурою DETR (DEtection TRansformer) [14]. GroundingDINO приймає на вхід пари (зображення, тестова назва) і повертає кілька пар (обмежувальна коробка, текстова назва). Базова архітектура моделі містить початкову модель виявлення ознак зображень та початкову модель виявлення ознак у тексті. Далі ці ознаки передаються для обробки модулю підсилювання ознак (feature enhancer module) для забезпечення крос-модальності. Після цього дані передаються декодеру, що визначає необхідні ознаки з обох модальностей для виявлення об’єктів та відповідних текстових фраз. Важливими компонентами архітектури цієї системи є модуль вибору запиту з керуванням мовою (language-guided

query selection module), крос-модальний декодер, Sub-Sentence рівень текстових ознак та функції втрат, а саме L1 і GIOU.

Розділ 3

3.1DETR архітектура

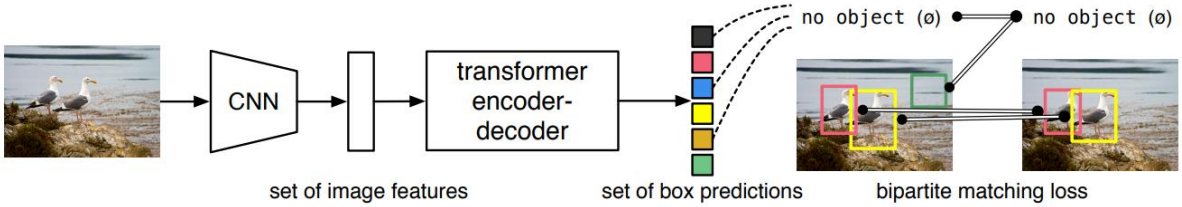

Рисунок 6. Базова архітектура DETR

DETR (DEtection TRansformer) – це модель глибинного навчання з побудованою архітектурою на основі використання трансформерів, що була запропонована Facebook AI Research у 2020 році[]. Ця модель відрізняється від переважної більшості поширених моделей виявлення об’єктів певними структурними компонентами. Особливостями запропонованої структури є використання функції втрат, що базується на обрахункові глобального значення втрат на наборі всіх передбачень та encoder-decoder архітектура.

Першим елементом DETR є CNN (Convolutional Neural Network), що є початковим кодувальником, ця модель повертає ознаки вищого рівня (high-level features) вхідного зображення такі як інформація про об’єкти, певний контекст елементів на зображенні. Надалі вводяться позиційні кодування (positional encoding) для представлення зв'язків між частинами зображення, що дозволяє трансформеру отримати дані про відносні та абсолютні положення елементів. Трансофрмер здійснює кодування та подальше декодування визначеного набору ознак і визначає відразу певну кількість об’єктів, що представляють повну сукупність об’єктів передбачення. Останній етап передбачення здійснюється FFN (Feedforward

neural network), що передбачає відносні координати середини, ширину та довжину обмежувальної коробки. Оскільки загальна кількість визначених рамок N є переважно більшою за кількість об'єктів на зображенні, вводиться порожній клас, що позначає ті об'єкти, які не відносяться до жодного з визначених попередньо класів. Цей клас схожий до типового для завдань виявлення об'єктів класі фону зображення.

3.1.1 Bipartite matching loss

Нехай $y = \{y_i\}_{u=1}^{M}$ набір об'єктів, $\hat{y} = \{\hat{y}_i\}_{u=1}^{N}$ – набір передбачень, кожне з яких представляє пару з визначеного класу та обмежувальної рамки, N > M. Приведення у до розміру N з ∅ тобто відсутністю об'єкта позначено $\bar{y}$. Тоді Hungarian loss, угорська функція втрат визначатиметься наступним чином:

$$\mathcal{L}_H(\bar{y}, \hat{y}) = \sum_{i=1}^{N} [\mathcal{L}_{class}^{i,\hat{\sigma}(i)} + 1_{\{\bar{y}\neq\emptyset\}}\mathcal{L}_{box}^{i,\hat{\sigma}(i)}] \quad (1)$$

$\mathcal{L}_{class}^{i,\hat{\sigma}(i)}$ та $\mathcal{L}_{box}^{i,\hat{\sigma}(i)}$ є значеннями втрати класифікації та регресивної втрати обмежувальної рамки між i^{th}ground truth та $\hat{\sigma}(i)^{th}$передбаченням. У свою чергу $\hat{\sigma}$ це оптимальний bipartite matching, що може бути визначений наступним чином:

$$\hat{\sigma} = arg\min_{\sigma\in Q_N}\sum \mathcal{L}_m(\bar{y}_i, \hat{y}_{\sigma(i)}) \quad (2)$$

Де Q_N – набір усіх N перестановок пар та $\mathcal{L}_m$ – значення втрат між кожним $\bar{y}$ та $\hat{y}_{\sigma(i)}$. Таким чином за рахунок використання Hungarian алгоритму можна визначити оптимальний bipartite matching. [15]

3.2 Використання створеного анотатора

З-поміж наборів даних доступних на платформі Kaggle нами було обрано 2 різних набори, що створені для тренування моделей виявлення

об’єктів типу дорожні знаки. Traffic Signs Detection (Signs Detection For Self-Driving Cars) [16] та Traffic Signs Detection [17]. Перший набір містить 3530 зображень для тренування, 801 зображення для валідації та 638 тестових зображень, а також відповідну кількість файлів з метаданими у форматі YOLO (txt). Цей набір даних призначений для використання у вирішенні завдань виявлення об’єктів, що належать до 15класів: «Green Light», «Red Light», «Speed Limit 10», «Speed Limit 100», «Speed Limit 110», «Speed Limit 120», «Speed Limit 20», «Speed Limit 30», «Speed Limit 40», «Speed Limit 50», «Speed Limit 60», «Speed Limit 70», «Speed Limit 80», «Speed Limit 90», «Stop». Другий набір містить більше 5000 зображень для тренування, більше 2000 зображень для валідації та відповідну кількість файлів з метаданими також у форматі YOLO. Він надає дані про дорожні знаки з 55 різних класів, які можна віднести до 4 категорій: forb, info, mand, warn, кожен клас має в назві один з таких префіксів із зазначенням категорії транспортного знаку: forbidden (заборонений), informational (інформаційний), mandatory (обов'язковий) та warning (попереджувальний), відповідно. З-поміж цих наборів даних випадковим чином було обрано близько 220 зображень і поділено їх на дані для тренування, валідації та тестування таким чином, що перша категорія містить близько 110 зображень, друга близько 60 та третя також близько 60. Після чого усі ці зображення було проанотовано з використанням створеного анотатора та із частковим використанням моделі GroundingDINO для пришвидшення виконання завдання. Усі об’єкти з проанотованого набору належать до класу дорожніх знаків. Ефективність роботи GroundingDINO є досить високою, проте варто зважати, що через специфіку цієї системи в окремих випадках створюється кілька обмежувальних рамок, що не відповідають тому класу, який потрібно анотувати або повторюють подібні обмежувальні коробки.

У ході виконання роботи було виконане тренування моделі з DETR архітектурою для визначення об’єктів, що належать до класу «дорожній знак». Дані, що використовувалися, були розмічені з використанням створеного застосунку. Усього було виконано 200 ітерацій на 114 зображення із набору для тренування та 61 зображення валідаційного набору. Попередньо було виконано спроби тренування на більшій кількості даних із, згаданих раніше наборів, проте, DETR вимагає значних обчислювальних потужностей, що майже унеможливлює тренування на великій кількості даних без створених для цього умов. Натренована модель використовує ваги претренованої моделі detr-resnet-50 []. Отримані результати показують, що модель може визначати дорожні знаки з достатньою чіткістю визначення обмежувальної рамки.

Test metric	**Value**
test/loss	0.3963673412799835
test_loss_ce	0.010869328863918781
test_loss_bbox	0.04830474779009819
test_loss_giou	0.07198711484670639
test_cardinality_error	0.2539682686328888

Таблиця 1. Метрики на тестовому наборі даних

Де test/loss – загальне значення обраховних втрат на тестовому наборі даних, test_loss_ce – cross entropy loss, test_loss_bbox – значення втрат по обмежувальних коробках, L1 regression loss, test_loss_giou – значення втрат обраховане за функцією giou, test_cardinality_error – абсолютна похибка в наборі передбачених не порожніх обмежувальних коробок [18].

Висновки

У ході виконання роботи було досліджено можливі методи пришвидшення анотування для виявлення об’єктів. З-поміж них виокремлено кілька категорій, що базуються на важливості певних ознак. Створено застосунок, що дозволяє пришвидшити виконання розмітки даних за рахунок використання претренованих моделей та мульти-модальних претренованих моделей. Відповідно додано підтримку використання yolov8 та GroundingDINO. Також проведено дослідження побудови цих моделей та подано їх описи із окремими характеристиками елементів архітектури.

Перелік використаних у роботі термінів та їх відповідників

Анотування або розмітка даних - data annotation

Відстежування об’єктів - object tracking

Тонке настроювання - fine-tuning

Ознаки - features

Обмежувальна коробка або обмежувальна рамка - bounding box

Набір даних - dataset

Претренована модель - pretrained model

Мульти-модальна модель – multi-modal model

Список використаних джерел

1. Data Collection and Labelling [Електронний ресурс] – Режим доступу до ресурсу: https://www.globaldata.com/store/report/data-collection-and-labelling-market-analysis/#:~:text=The%20data%20collection%20and%20labelling%20ma

rket%20will%20be%20valued%20at,for%20high-quality%20training%20data.

2. Active Finetuning: Exploiting Annotation Budget in the Pretraining-Finetuning Paradigm / Yichen Xie, Han Lu, Junchi Yan, Xiaokang Yang, Masayoshi Tomizuka, Wei Zhan – Режим доступу до ресурсу: https://arxiv.org/abs/2303.14382.

3. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection / Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang – Режим доступу до ресурсу: https://arxiv.org/abs/2303.05499.

4. GroundingDINO [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/IDEA-Research/GroundingDINO.

5. Qt for Python [Електронний ресурс] – Режим доступу до ресурсу: https://doc.qt.io/qtforpython-6/.

6. Ultralytics YOLOv8 Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ultralytics.com.

7. COCO dataset [Електронний ресурс] – Режим доступу до ресурсу: https://cocodataset.org/#home.

8. Ultralytics [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ultralytics/ultralytics?ref=blog.roboflow.com.

9. You Only Look Once: Unified, Real-Time Object Detection [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://arxiv.org/pdf/1506.02640.

10. Real-Time Flying Object Detection with YOLOv8 [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://arxiv.org/pdf/2305.09972.

11. YOLOv3: An Incremental Improvement [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/pdf/1804.02767v1.

12. RangeKing yolov8 diagram [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/RangeKing.

13. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. – 2023. https://arxiv.org/pdf/2303.05499.

14. Attention Is All You Need [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1706.03762.

15. Rethinking Transformer-based Set Prediction for Object Detection / Zhiqing Sun, Shengcao Cao, Yiming Yang, Kris Kitani– 2021. – Режим доступу до ресурсу: https://arxiv.org/pdf/2011.10881.

16. Traffic Signs Detection (Signs Detection For Self-Driving Cars) / Roboflow user PARISA KARIMI DARABI [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/pkdarabi/cardetection

17. Traffic Signs Detection / Radu Oprea [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/raduoprea/traffic-signs

18. Detr [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/facebookresearch/detr/tree/master

19. End-to-End Object Detection with Transformers [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/2005.12872.

20. What is YOLOv8? The Ultimate Guide. [2024] [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://blog.roboflow.com/whats-new-in-yolov8/.

Додатки

Додаток А

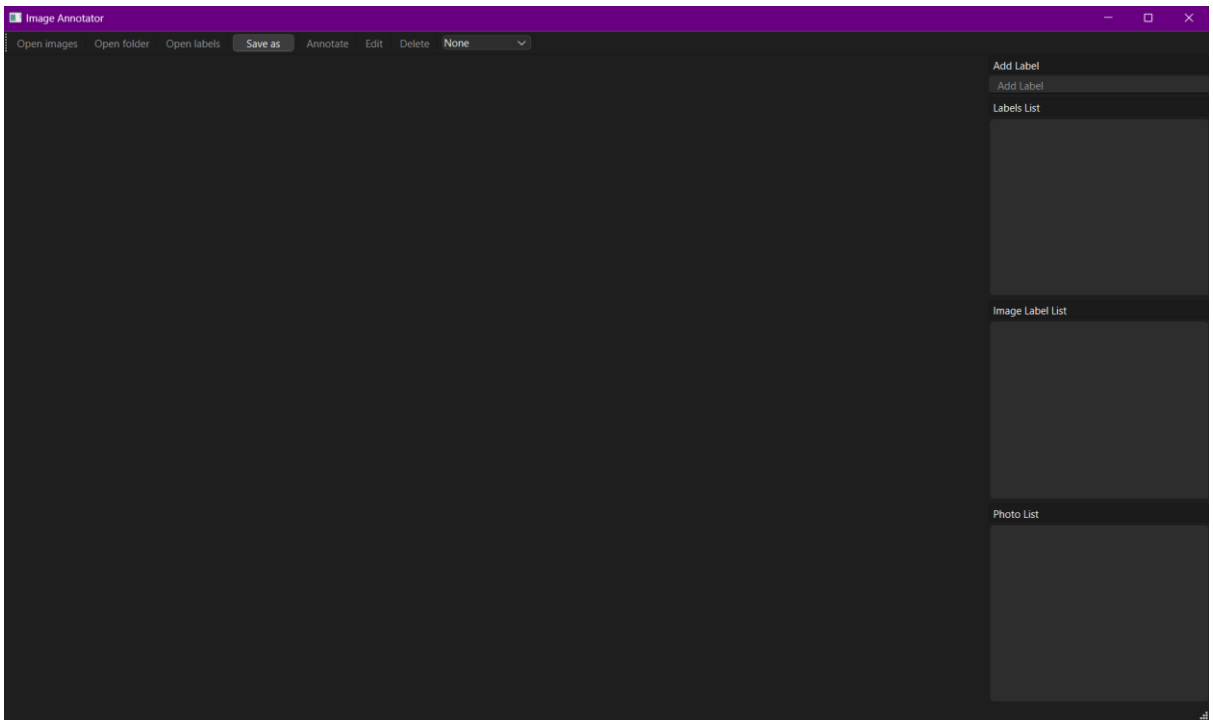

Рисунок 1. Основний інтерфейс застосунку

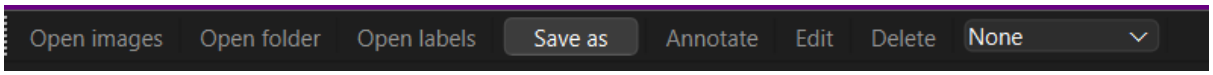

Рисунок 2. Верхня панель застосунку

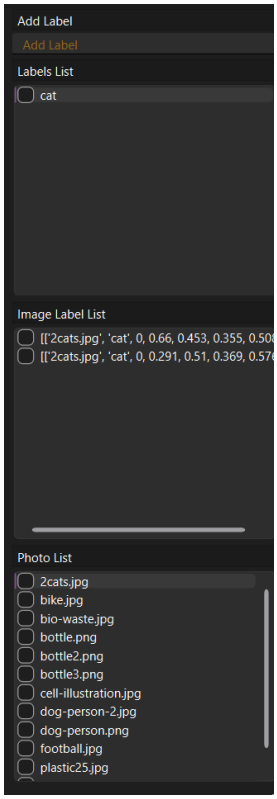

Рисунок 3. Права панель застосунку

Додаток Б

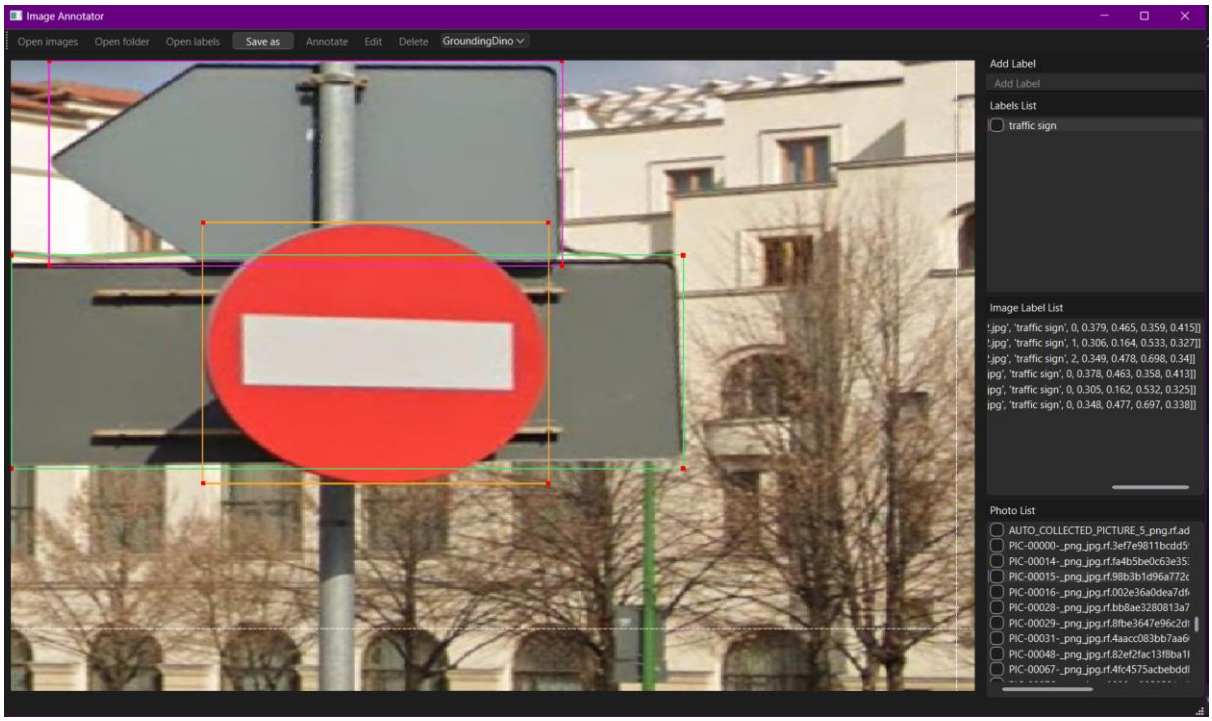

Рисунок 1. Обмежувальні рамки створені з використанням GroundingDINO, кількість виявлених об’єктів більша

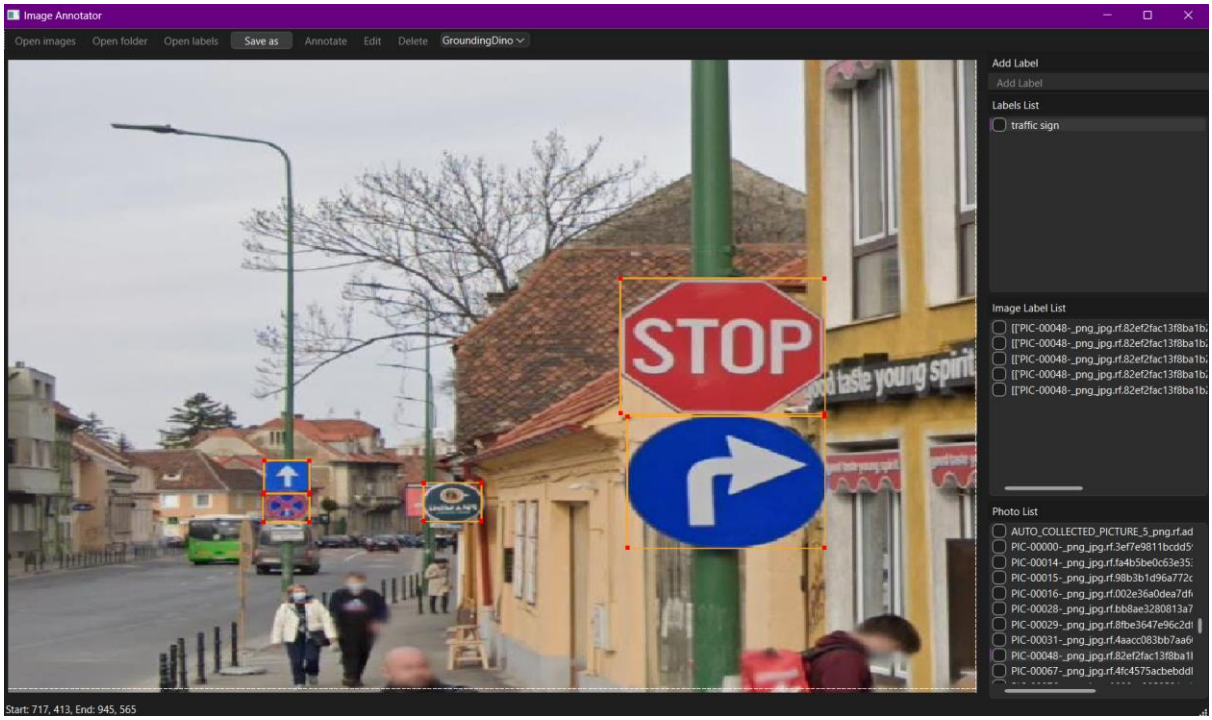

Рисунок 2. Помилковість окремих обмежувальних коробок