Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Ідентифікація людей за фото та відео засобами Computer Vision

Виконав: студент VI курсу, групи СБм-61
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

(підпис) *Гаврилов М. В.* (прізвище та ініціали)

Керівник (підпис) *Александер М. Б-А.* (прізвище та ініціали)

Нормоконтроль (підпис) *Лобур Т.Б.* (прізвище та ініціали)

Завідувач кафедри (підпис) *Загородна Н.В.* (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«____» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Гаврилову Миколі Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Ідентифікація людей за фото та відео засобами Computer Vision

Керівник роботи Александер Марек Богуслав Антонович, д.т.н., професор кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_25_» _листопада_ 2022 року № _4/7-966_

2. Термін подання студентом завершеної роботи 14 грудня 2022р.

3. Вихідні дані до роботи Наукові публікації про загрози хмарної безпеки та проблем безпеки хмарних середовищ

4. Зміст роботи (перелік питань, які потрібно розробити): Вступ, 1 Загальний огляд od та re-id в сфері ds, 1.1 Виявлення об'єктів у відкритому світі, 1.1.1 Порівняльний аналіз відомих Джерел, 1.2 Повторне виявлення об'єктів, 1.2.1 Порівняльний аналіз методів, 2 Обгрунтування вибору моделі та підходів, 2.1 Порівняльний аналіз архітектур моделей для Object Detection, 2.1.1 SSD, 2.1.2 Faster R-CNN, 2.1.3 YOLOv3, 2.1.4 Порівняння швидкодії моделей YOLO, 2.1.5 Обгрунтування вибору моделі від YOLO, 2.2 Порівняння підходів для ReID особи, 2.2.1 ReID особи, 2.2.2 Підсумок досліджених підходів, 2.2.3 Обгрунтування вибору підходу, 3 Практична реалізація, 3.1 Підготовка середовища, 3.2 Особливості встановлення Torchreid, 3.3 Налаштування YOLOv7, 3.4 Створення набору даних для тренування та оцінювання, 3.5 Тренування Torchreid, 3.6 Оцінка моделей, 4 Охорона праці та безпека в надзвичайних ситуаціях, 4.1 Охорона праці, 4.1.1 Обов'язки роботодавця, 4.1.2 Обов'язки працівників, 4.2 Вплив виробничого середовища на здоров'я та працездатність користувачів комп'ютерів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема. Мета. Об'єкт. Предмет дослідження. 3 Завдання дослідження. 4 Загальні відомості про Object Detection. 5 Загальні відомості про Re-ID. 6 Проблеми Object Detection. 7 Проблеми Re-ID. 8 Порівняння YOLO моделей на GPU та CPU. 9 Порівняння підходів Re-ID. 10 Принцип роботи інформаційної системи. 11 Результати оцінки Re-ID. 12 Висновки. 13 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 14 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.2022-15.11.2022	*Виконано*
2.	Підбір наукових джерел про засоби виявлення		
	та ідентифікації людей	16.11.2022-20.11.2022	*Виконано*
3.	Переклад та опрацювання наукових джерел про		
	дослідження методів повторної ідентифікації людей		
	в різних випадках	21.11.2022-23.11.2022	*Виконано*
4.	Виконання дослідження щодо аналіз інструментів для		
	розробки інформаційної системи виявлення та		
	ідентифікації людей	24.11.2022-27.11.2022	*Виконано*
5.	Оформлення розділу «Загальний огляд od та re-id		
	в сфері ds»	28.11.2022-30.11.2022	*Виконано*
6.	Оформлення розділу «Обгрунтування вибору моделі		
	та підходів»	01.12.2022-04.12.2022	*Виконано*
7.	Оформлення розділу «Практична реалізація»	05.12.2022-07.12.2022	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	08.12.2022-09.12.2022	*Виконано*
9.	Виконання завдання до підрозділу «Вплив		
	виробничого середовища на здоров'я та працездатність		
	користувачів комп'ютерів»	10.12.2022-11.12.2022	*Виконано*
10.	Оформлення кваліфікаційної роботи	12.12.2022-13.12.2022	*Виконано*
11.	Нормоконтроль	14.12.2022-15.12.2022	*Виконано*
12.	Перевірка на плагіат	9.12.2022	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.12.2022	*Виконано*
14.	Захист кваліфікаційної роботи	.12.2022	

Студент _____ Гаврилов М. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Александер М. Б-А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ідентифікація людей за фото та відео засобами Computer Vision // Кваліфікаційна робота освітнього рівня «Магістр» // Гаврилов Микола Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2022 // С. __, рис. – 34, табл. – 2, додат. – 10, бібліогр. – 53.

Ключові слова: НАУКА ПРО ДАНІ, АРХІТЕКТУРА, МОДЕЛЬ, ВИЯВЛЕННЯ ЛЮДИНИ, ІДЕНТИФІКАЦІЯ ЛЮДИНИ, ПОВТОРНА ІДЕНТИФІКАЦІЯ ЛЮДИНИ.

У кваліфікаційній роботі розроблено інструмент для знаходження конкретної людини на фото та відео з використанням інструментів Data Science.

Інформаційну систему, яка дає змогу обробляти фото чи відео, створювати свою базу даних, тренувати модель для конкретної людини і подальшого використання моделі для ідентифікації людини на різних камерах та локаціях.

У першому розділі було проведено загальний огляд проблем виявлення людини на фото та відео та виконано аналіз наявних методик для повторної ідентифікації людини.

В другому розділі було проведено дослідження наявних моделей для виявлення людини на фото та відео, проведена їх оцінка. Також, другий розділ містить більш детальний аналіз підходів для повторної ідентифікаціх людини.

У третьому розділі програмно реалізовано мовою програмування Python інформаційної системи, з використанням наявних рішень та архітектур для

виявлення та ідентифікації людини, таких репозиторіїв як YOLOv7 та Torchreid.

ANNOTATION

Identification of people by photo and video using Computer Vision tools // Qualification paper of the educational level "Master" // Mykola Havrylov // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security, SBm-61 group // Ternopil, 2022 // P. __, fig. – 34, tables – 5, annexes – 10, references – 53.

Key words: DATA SCIENCE, ARCHITECTURE, MODEL, PERSON DETECTION, PERSON IDENTIFICATION, PERSON RE-ID

In this qualification paper an information system that gives you an opportunity to find a specific person in a photo and video using Data Science tools was developed.

The information system allows to process a photo or video, create your own database, train a model for a specific person and then use the model to identify a person on different cameras and locations.

The first section includes a general overview of the problems of person detection in photos/videos and an analysis of existing methods for person re-identification.

In the second section, existing models for person detection in photos/videos and model's evaluation were researched. Also, the second section contains more detailed analysis of approaches for re-identification of a person.

In the third chapter, the information system was implemented using Python programming language, using existing solutions and architectures for person detection and identification, such repositories as YOLOv7 and Torchreid.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Networks

CBN – Camera-based Batch Normalization

DS – Data Science

IPT – image processing transformer

IoU – Intersection Over Union

mAP – Mean Average Precision

NLP – natural language processing

OD – Object Detection

PaaS – platform as a service

ReID – re-identification

RoI – Region of Interest

SIE – side information embedding

VSCode – Visual Studio Code

YOLO – You Only Look Once

YOLOR – You Only Learn One Representation

ЗМІСТ

ВСТУП

Як люди, ми зазвичай проводимо своє життя, спостерігаючи за навколишнім середовищем за допомогою зорових нервів, сітківки та зорової кори. Ми отримуємо контекст, щоб розрізняти об’єкти, вимірювати їхню відстань від нас та інших об’єктів, розраховувати швидкість їх руху та виявляти помилки. Подібним чином комп’ютерний зір дозволяє машинам на основі штучного інтелекту навчитися виконувати ці самі процеси. Ці машини використовують для цього комбінацію камер, алгоритмів і даних.

Однак, на відміну від людини, комп’ютер не втомлюється. Ви можете навчити машини з комп’ютерним зором аналізувати тисячі виробничих активів або продуктів за лічені хвилини. Це дозволяє виробничим підприємствам автоматизувати виявлення дефектів, непомітних для людського ока.

Проте, щоб комп’ютерний зір був справді ефективним, потрібна велика база даних. Пояснення цьому те, що ці рішення аналізують інформацію неодноразово, доки не отримають усі можливі відомості, необхідні для виконання поставленого завдання. Наприклад, комп’ютер, навчений розпізнавати здорові посіви, мав би «бачити» тисячі візуальних еталонних даних про посіви, сільськогосподарські угіддя, тварин та інші пов’язані об’єкти і лише тоді він зможе ефективно розпізнавати різні типи здорових культур, диференціювати їх від нездорових, оцінювати якість сільськогосподарських угідь, виявляти шкідників та інших тварин серед культур тощо.

Метою даного дослідження є створення інформаційної системи, яка дає змогу ідентифікувати конкретну людину з існуючої бази даних за фото-та/відео рядом.

З мети випливають настпуні завдання: аналіз наявних моделей виявлення об’єктів, аналіз підходів для повторної ідентифікації.

Проблемою даної роботи є вирішення різного роду специфікацій пов’язаних з повторною ідентифікацією особи: різне освітлення, ракурси, порівняння малого та великого розміру об’єктів.

Відносно проаналізованої інформації було обрано найкращий підхід для вирішення проблеми поставленої задачі та проаналізовано її результати.

Наукова новизна полягає в створенні інформаційної системи, яка має змогу виявляти людину та повторно її ідентифікувати в інший ракурсах та локаціях і має покращені метрики в порівнянні з моделями, які були натреновані на наборі даних Imagenet.

РОЗДІЛ 1. ЗАГАЛЬНИЙ ОГЛЯД OD ТА RE-ID В СФЕРІ DS

1.1 Виявлення об’єктів у відкритому світі

Глибоке навчання прискорило прогрес у дослідженні виявлення об’єктів [1], де моделі доручено ідентифікувати та локалізувати об’єкти на зображенні. Усі існуючі підходи працюють за припущенням, що всі класи, які мають бути виявлені, будуть доступні під час фази навчання. Коли ми послаблюємо це припущення, виникають два складні сценарії:

1) Тестове зображення може містити об’єкти з невідомих класів, які слід класифікувати як невідомі.

2) Коли інформація (мітки) про такі ідентифіковані невідомі стає доступною, модель повинна мати можливість поступово вивчати новий клас.

Дослідження в галузі психології розвитку [2] показують, що здатність ідентифікувати те, чого людина не знає, є ключем до найцікавішого.

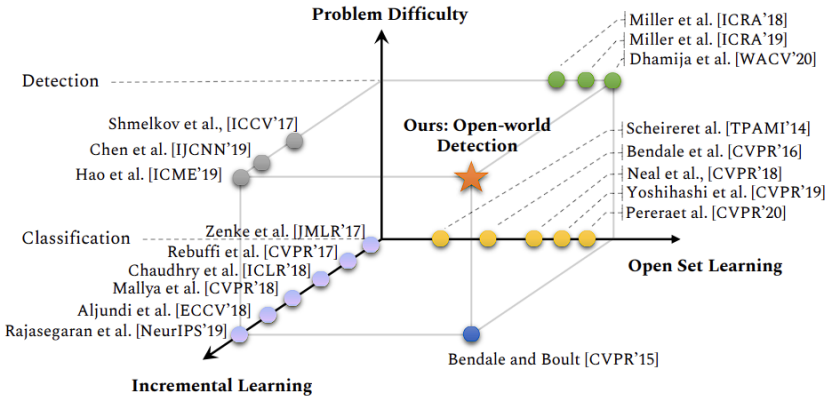


Рисунок 1.1 – Виявлення об’єктів у відкритому світі (⭐) є новою проблемою, яка досі не була офіційно визначена та не розглянута.

Така допитливість збільшує бажання пізнавати нове [3]. Нас це спонукає запропонувати нову проблему, коли модель повинна мати можливість ідентифікувати екземпляри невідомих об’єктів як невідомі, а потім навчитися

розпізнавати їх, коли поступово надходять навчальні дані, уніфікованим способом. Ми називаємо цю проблему налаштуванням Open World Object Detection (див. рис. 1.1).

Кількість класів, відмічених у стандартних наборах даних vision, таких як Pascal VOC і MS-COCO, дуже мала (20 і 80 відповідно) у порівнянні з нескінченною кількістю класів, які присутні у відкритому світі. Визнання невідомого невідомим вимагає сильного узагальнення. Scheirer разом з іншими [4] формалізують це як проблему Open Set класифікації. Відтоді для вирішення цієї складної ситуації було розроблено різні методології (з використанням 1-vs-rest SVM та моделей глибокого навчання). Bendale [5] розширює Open Set до налаштування Open World класифікації шляхом додаткового оновлення класифікатора зображень для розпізнавання ідентифікованих нових невідомих класів. Цікаво, що, як показано на рис. 1, виявлення об’єктів Open World є недослідженим через складність постановки проблеми.

Удосконалення класифікації зображень Open Set і Open World не можна тривіально адаптувати до виявлення об’єктів Open Set і Open World через фундаментальну різницю в постановці проблеми: детектор об’єктів навчений виявляти невідомі об’єкти як фон. Екземпляри багатьох невідомих класів уже були введені в детектор об’єктів разом із відомими об’єктами. Оскільки вони не позначені, ці невідомі екземпляри будуть явно засвоєні як фон під час навчання моделі виявлення. Dhamija та інші [13] виявили, що навіть із цим додатковим навчальним сигналом найсучасніші детектори об’єктів призводять до хибно позитивних виявлень, коли невідомі об’єкти в кінцевому підсумку класифікуються як один із відомих класів, часто з дуже високою ймовірністю. Інші [6] пропонують використовувати відбір вибірки для отримання оцінки невизначеності прогнозу виявлення об’єкта. Це єдина рецензована дослідницька робота в літературі з виявлення відкритих об’єктів. Запропонована технологія Open World Object Detection йде далі, щоб

поступово вивчати нові класи, коли вони виявляються як невідомі, а оракул надає мітки для об’єктів, що цікавлять, серед усіх невідомих. Наскільки відомо, це не було спробувано в літературі.

Open World Object Detection є набагато природнішим, ніж існуюче статичне навчання закритого світу. Світ різноманітний і динамічний за кількістю, типом і конфігураціями нових класів. Було б наївно припускати, що всі класи, які очікуються під час висновку, видно під час навчання. Практичне розгортання систем виявлення в робототехніці, безпілотних автомобілях, фенотипуванні рослин, охороні здоров’я та нагляді не може дозволити собі мати повне знання про те, яких класів очікувати під час висновків, під час навчання в компанії. Найбільш природною та реалістичною поведінкою, яку можна очікувати від алгоритму виявлення об’єктів, розгорнутого в таких обставинах, було б впевнене передбачення невідомого об’єкта як невідомого, а відомі об’єкти – у відповідні класи. Коли стане доступною додаткова інформація про ідентифіковані невідомі класи, система повинна мати можливість включити їх у свою існуючу базу знань і це визначило б розумну систему виявлення об’єктів.

В загальному є три найвідоміших джерела:

1) Open Set Classification враховує дані, отримані за допомогою навчального набору, вважаються неповними, тому під час тестування можна зустріти нові невідомі класи. Scheirer та ін. [7] розробили класифікатори відкритого набору в налаштуванні «one-vs-rest», щоб збалансувати продуктивність і ризик позначення зразка далеко від відомих навчальних прикладів (що називається ризиком відкритого простору). Подальші роботи [18, 19] розширили структуру відкритого набору до налаштувань класифікатора з кількома класами з ймовірнісними моделями для врахування зникнення достовірності класифікатора у випадку невідомих класів.

Bendale і Boult [8] визначили невідомі в просторі ознак глибоких мереж і використали розподіл Вейбулла для оцінки встановленого ризику (так званий

класифікатор OpenMAX). Основна версія OpenMAX була запропонована в [9] шляхом синтезу нових образів класів. В роботі [10] розглядають довгострокову систему визнання, де співіснують більшість, меншість і невідомі класи. Вони розробили метричну структуру навчання, щоб ідентифікувати невидимі класи як невідомі. У подібному дусі кілька спеціалізованих підходів спрямовані на виявлення зразків, які не розповсюджуються, або новинок. Нещодавно для розпізнавання відкритого набору досліджувалися самоконтрольоване навчання і неконтрольоване навчання з реконструкцією. Однак, незважаючи на те, що ці роботи можуть розпізнавати невідомі екземпляри, вони не можуть динамічно оновлюватися поетапно протягом кількох етапів навчання.

2) Open World Classification вперше запропонувала налаштування відкритого світу для розпізнавання зображень. Замість статичного класифікатора, навченого на фіксованому наборі класів, вони запропонували більш гнучке налаштування, де співіснують як відомі, так і невідомі. Модель може розпізнавати обидва типи об’єктів і адаптивно вдосконалюватися, коли надаються нові мітки для невідомих. Їхній підхід розширює класифікатор Nearest Class Mean для роботи у відкритому світі шляхом повторного калібрування ймовірностей класу, щоб збалансувати ризик відкритого простору. [11] вивчає навчання ідентифікації обличчя у відкритому світі, тоді як [12] запропонував використовувати типовий набір побачених класів, щоб зіставити їх із новим зразком та відхилив його у випадку низького збігу з усіма раніше відомими класами. Однак вони не перевіряють контрольні показники класифікації зображень і не вивчають класифікацію продуктів у програмах e-commerce.

3) У роботі [13] офіційно досліджували Open Set налаштування на популярні детектори об’єктів. Вони помітили, що найсучасніші детектори об’єктів часто класифікують невідомі класи з високою достовірністю до видимих класів. Це незважаючи на те, що детектори явно навчаються з

фоновим класом та/або застосовують класифікатори one-vs-rest для моделювання кожного класу. Спеціальна частина робіт [14] зосереджена на розробці заходів (просторової та семантичної) невизначеності в детекторах об’єктів для відхилення невідомих класів. Наприклад, [6] використовує вибірку за методом Монте-Карло Dropout [15] у детекторі SSD для отримання оцінок невизначеності. Проту ці методи не можуть поступово адаптувати свої знання в динамічному світі.

1.2 Повторне виявлення об’єктів

Повторна ідентифікація об’єкта (ReID) спрямована на пов’язування певного об’єкта між різними сценами та видами камери, наприклад у програмах ReID особи та ReID автомобіля. Вилучення надійних і дискримінаційних ознак є ключовим компонентом ReID, і протягом тривалого часу в ньому домінували методи, засновані на CNN (див. рис. 1.2, 1.3).

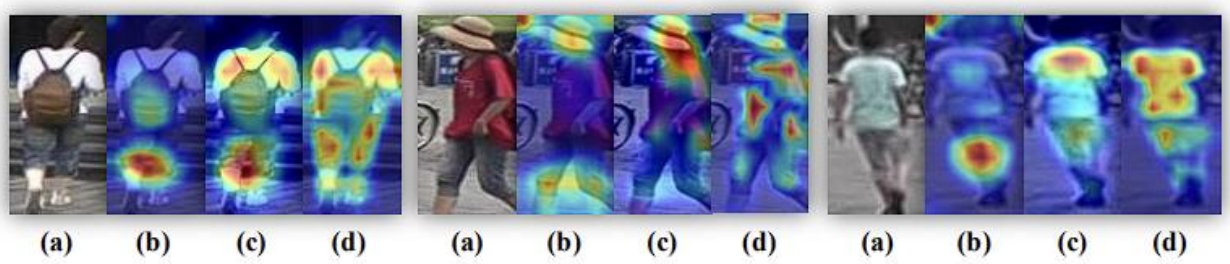


Рисунок 1.2 – Grad-CAM візуалізація карт уваги: (a) вихідні зображення, (b) методи на основі CNN, (c) методи CNN+увага, (d) методи на основі трансформаторів, які фіксують інформацію про глобальний контекст і більш розрізнювальні частини.

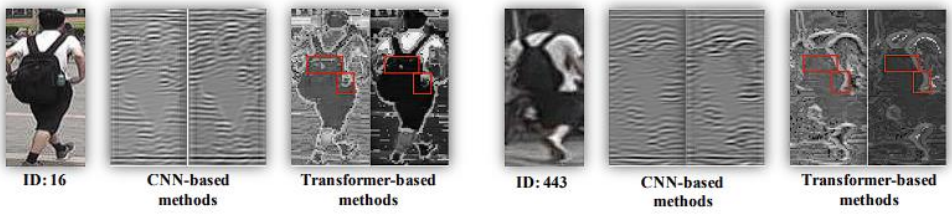


Рисунок 1.3 – Візуалізація вихідних карт характеристик для 2 зразків із подібним виглядом. Методи на основі трансформаторів зберігають деталі рюкзака на вихідних картах функцій на відміну від методів на основі CNN, як зазначено в червоних прямокутниках. Для кращої візуалізації вхідні зображення масштабуються до розміру 1024 × 512.

Переглядаючи методи, засновані на CNN, було виявили дві важливі проблеми, які недостатньо розглянуті в області ReID об’єкта:

1) Використання багатих структурних шаблонів у глобальному масштабі має вирішальне значення для ReID об’єкта. Однак методи, засновані на CNN, головним чином зосереджені на невеликих дискримінаційних областях через розподіл Гауса ефективних рецептивних полів. Нещодавно були введені модулі уваги для дослідження довгострокових залежностей, але більшість із них вбудовані в глибинні шари і не вирішують принципової проблеми CNN. Таким чином, методи на основі уваги все ще віддають перевагу великим безперервним областям і важко виділити кілька різноманітних дискримінаційних частин (див. Рис. 2).

2) Точні функції з детальною інформацією також важливі. Однак оператори зменшення дискретизації (наприклад, об’єднання та поступова згортка) CNN зменшують просторову роздільну здатність вихідних карт ознак, що значно впливає на здатність розрізняти об’єкти зі схожою зовнішністю [18]. Як показано на Рис. 3, деталі рюкзака втрачені на картах функцій на основі CNN, що ускладнює розрізнення двох людей.

Нещодавно Vision Transformer (ViT) [19] і Dataefficient image Transformers (DeiT) [20] показали, що чисті трансформатори можуть бути

такими ж ефективними, як і методи на основі CNN щодо виділення ознак для розпізнавання зображень. Із запровадженням multi-head attention modules, видаленням операторів згортки та зменшення дискретизації моделі на основі трансформаторів підходять для вирішення вищезазначених проблем у ReID на основі CNN з наступних причин:

1) Multi-head self-attention фіксує великі залежності та спонукає модель звертати увагу на різні частини людського тіла, ніж моделі CNN (наприклад, стегна, плечі, талія на Рис. 2).

2) Без операторів зменшення дискретизації трансформатори можуть зберігати більш детальну інформацію. Наприклад, можна помітити, що різниця в картах характеристик навколо рюкзаків (позначених червоними прямокутниками на Рис. 3) може допомогти моделі легко відрізнити двох людей. Такі переваги спонукають нас до впровадження чистих трансформаторів в об'єкт ReID.

Незважаючи на великі переваги, як обговорювалося вище, трансформатори все ще повинні бути розроблені спеціально для ReID об’єкта, щоб впоратися з унікальними проблемами, такими як великі варіації (наприклад, оклюзії, різноманітність поз, перспектива камери) на зображеннях. Значні зусилля були спрямовані на полегшення цієї проблеми за допомогою методів, заснованих на CNN. Серед них було доведено, що локальні елементи [21] і додаткова інформація (така як камери та точки огляду) [22] є важливими та ефективними для підвищення надійності функції. Вивчення агрегованих функцій частин/смуг робить його стійким до оклюзій і зміщень [23]. Однак розширення методів жорсткої смугової частини з методів на основі CNN на методи на основі чистих трансформаторів може пошкодити довгострокові залежності через розбиття глобальних послідовностей на кілька ізольованих підпослідовностей. Крім того, беручи до уваги додаткову інформацію, таку як інформація про камеру та точку огляду, можна сконструювати простір інваріантних ознак, щоб зменшити зміщення,

викликане варіаціями побічної інформації. Однак складні конструкції для додаткової інформації, побудовані на основі CNN, якщо безпосередньо застосувати до трансформаторів, не можуть повною мірою використовувати властиві трансформаторам можливості кодування. Як наслідок, спеціально розроблені модулі неминучі та необхідні для того, щоб чистий трансформатор успішно справлявся з цими викликами.

Для розширення довготривалих залежностей та підвищення надійності функцій було запропоновано модуль патчів Jigsaw (JPM) шляхом перегрупування вставок патчів за допомогою операцій зсуву та перемішування та їхнього перегрупування для подальшого вивчення функцій. JPM використовується на останньому рівні моделі для отримання надійних функцій паралельно з глобальною гілкою, яка не включає цю спеціальну операцію. Таким чином, мережа має тенденцію витягувати інваріантні до подразнювачів і надійні функції з глобальним контекстом.

Також, для подальшого покращення вивчення надійних функцій введено додаткову інформацію вектора (SIE). Замість спеціальних і складних розробок у методах на основі CNN для використання цих невізуальних підказок було запропоновано уніфіковану структуру, яка ефективно включає невізуальні підказки через вбудовування, яке можна вивчати, щоб зменшити зміщення даних, яке приносять камери або точки зору. Взявши, наприклад, камери, запропонований SIE допомагає усунути велику розбіжність подібності між звичайними камерами та камерами нічного бачення. SIE можна легко розширити, включивши будь-які невізуальні підказки, крім тих, які було згадано вище.

Два джерела, які було досліджено:

1) Дослідження ReID об’єкта були в основному зосереджені на ReID людини та транспортного засобу, при чому більшість найсучасніших методів базуються на структурі CNN. Популярним методом для ReID об’єкта є розробка відповідних функцій втрати для навчання CNN (наприклад, ResNet

[24]), яка використовується для вилучення особливостей зображень. Cross-entropy loss (ID loss) [25] і triplet loss [26] найбільш широко використовуються в глибокому ReID. В цій роботі [56] було запропоновано BNNeck для кращого поєднання ID loss та triplet loss.

Fine-grained властивості навчилися збирати інформацію з різних частин або регіонів. Fine-grained частини або автоматично генеруються за допомогою приблизно горизонтальних смуг, або шляхом семантичного аналізу. Такі методи, як PCB [16], MGN [17], AlignedReID++ [61], SAN[27] тощо, ділять зображення на кілька смуг і виділяють локальні особливості для кожної смуги. Використання аналізу або оцінки ключових точок для вирівнювання різних частин або двох об’єктів також було доведено ефективним для ReID як людини, так і автомобіля [28].

Для зображень, знятих системою перехресних камер, існують значні варіації щодо пози, орієнтації, освітлення, роздільної здатності тощо, спричинені різними налаштуваннями камери та точками огляду об’єктів. У деяких роботах для вивчення інваріантних властивостей використовується додаткова інформація, така як ідентифікатор камери або інформація про точку огляду. Наприклад, пакетна нормалізація на основі камери (CBN) змушує проектувати дані зображення з різних камер на той самий підпростір, таким чином розрив у розподілі між парами між камерами та парами камер значно зменшується. Вивчення властивостей, незмінних у точці огляду/орієнтації [29] також є важливим для ReID як людини, так і автомобіля.

2) Модель Transformer запропонована в [30] для обробки послідовних даних у сфері NLP. Багато досліджень також показують його ефективність для завдань комп’ютерного зору. В роботах [31] дослідили застосування Transformer в області комп'ютерного зору.

Моделі Pure Transformer стають все більш популярними. Наприклад, IPT [32] використовує переваги трансформаторів за допомогою широкомасштабного попереднього навчання та досягає найсучаснішої

продуктивності в кількох завданнях обробки зображень, таких як надроздільна здатність, усунення шумів і усунення дощу. Нещодавно запропоновано ViT [19], який застосовує чистий трансформатор безпосередньо до послідовностей патчів зображення. Однак ViT вимагає великомасштабного набору даних для попереднього навчання моделі. Для подолання цього недоліку, було запропоновано структуру під назвою DeiT [20], яка представляє стратегію «викладач-учень», специфічну для трансформаторів, щоб прискорити навчання ViT без потреби великомасштабних даних попереднього навчання.

РОЗДІЛ 2. ОБГРУНТУВАННЯ ВИБОРУ МОДЕЛІ ТА ПДІХОДІВ

2.1 Порівняльний аналіз архітектур моделей для Object Detection

Комп’ютер розглядає всі види візуальних засобів масової інформації як масив числових значень. Як наслідок цього підходу, їм потрібні алгоритми обробки зображень для перевірки вмісту зображень. Далі буде порівняно 3 основні алгоритми обробки зображень: Single Shot Detection (SSD), Faster Region based Convolutional Neural Networks (Faster R-CNN) і You Only Look Once (YOLO), щоб знайти найшвидший і найефективніший із трьох. У цьому порівняльному аналізі з використанням набору даних Microsoft COCO (Common Object in Context) оцінюється продуктивність цих трьох алгоритмів, а їхні сильні сторони й обмеження аналізуються на основі таких параметрів, як accuracy, precision and F1 score. За результатами аналізу можна зробити висновок, що придатність будь-якого з алгоритмів порівняно з двома іншими значною мірою залежить від варіантів використання, у яких вони використовуються. В ідентичному середовищі тестування YOLO-v3 перевершує SSD і Швидший R-CNN, що робить його найкращим із трьох алгоритмів.

Останнім часом промислова революція використовує комп’ютерний зір для своєї роботи. Сектори автоматизації, робототехніки, галузі медицини та спостереження широко використовують глибоке навчання. Глибоке навчання стало найбільш обговорюваною технологією завдяки її результатам, які в основному отримані в додатках, що включають мовну обробку, виявлення об’єктів і класифікацію зображень. Прогноз ринку передбачає значне зростання в найближчі роки. Основними причинами цього є доступність як потужних графічних процесорів (GPU), так і багатьох наборів даних. Останнім часом обидві ці вимоги легко доступні.

Класифікація зображень і виявлення є найважливішими стовпами виявлення об’єктів. Існує безліч доступних наборів даних. Microsoft COCO є одним із

таких широко використовуваних доменів класифікації зображень. Це еталонний набір даних для виявлення об’єктів. Він представляє великомасштабний набір даних, доступний для виявлення та класифікації зображень.

Метою цієї огляду є порівняльний аналіз SSD, Faster-RCNN і YOLO. Першим алгоритмом для порівняння в поточній роботі є SSD, який додає рівні кількох функцій до кінцевої мережі та полегшує виявлення [33]. Faster R-CNN — це уніфікований, швидший і точний метод виявлення об’єктів, який використовує згортову нейронну мережу. Тоді як YOLO був розроблений Джозефом Редмоном, який пропонує наскрізну мережу [33].

Набір даних Microsoft COCO було використано як загальний фактор аналізу та вимірювання тих самих показників у всіх згаданих реалізаціях, відповідні показники трьох вищезгаданих алгоритмів, які використовують різні архітектури, були порівняні між собою. Результати, отримані шляхом порівняння ефективності цих алгоритмів на одному наборі даних, можуть допомогти отримати уявлення про унікальні атрибути кожного алгоритму, зрозуміти, чим вони відрізняються один від одного, і визначити, який метод розпізнавання об’єктів є найефективнішим для будь-якого сценарію.

2.1.1 SSD

Інші моделі виявлення об’єктів, такі як YOLO або Faster R-CNN, виконують свої операції зі значно меншою швидкістю порівняно з SSD, що робить метод виявлення об’єктів набагато сприятливішим.

До розробки SSD було зроблено кілька спроб створити швидший детектор шляхом модифікації кожного етапу виявлення. Однак будь-яке значне збільшення швидкості за рахунок таких модифікацій призводило лише до зниження точності виявлення, і тому дослідники дійшли висновку, що замість того, щоб змінювати існуючу модель, їм доведеться розробити принципово іншу модель виявлення об’єктів, а отже, створити моделі SSD [34].

SSD не виконує повторну дискретизацію пікселів або властивостей для гіпотез обмежувальної рамки та є таким же точним, як і моделі, які це роблять. На додаток до цього, він досить простий порівняно з методами, які вимагають виділені регіони об’єктів, оскільки він повністю усуває етапи передискретизації функцій або створення пікселів і пропозицій, охоплюючи всі обчислення в одній мережі. Таким чином, SSD дуже простий у навчанні та може бути легко інтегрований у системи, які виконують виявлення як одну зі своїх функцій [34].

Його архітектура значною мірою залежить від створення обмежувальних рамок і вилучення карт функцій, які також відомі як рамки за замовчуванням. Втрати розраховуються мережею, використовуючи порівняння зсувів прогнозованих класів і обмежувальних рамок за замовчуванням із базовими значеннями створених навчальних вибірок, використовуючи різні фільтри для кожної ітерації. Усі параметри оновлюються, використовуючи алгоритм зворотного поширення та розраховане значення втрат. Таким чином, SSD може вивчати найбільш оптимальні структури фільтрів, які можуть точно ідентифікувати особливості об’єкта та узагальнювати задані навчальні зразки, щоб мінімізувати значення втрат, що призводить до високої точності на етапі оцінки [35].

2.1.2 Faster R-CNN

R-CNN це регіональні згорткові нейронні мережі. Цей метод поєднує регіональні пропозиції для сегментації об’єктів і CNN високої ємності для виявлення об’єктів [36].

Алгоритм оригінальної методики R-CNN такий [37]:

1) За допомогою алгоритму вибіркового пошуку кілька пропозицій регіону-кандидата витягуються з вхідного зображення. У цьому алгоритмі численні регіони-кандидати генеруються в початковій підсегментації. Потім подібні регіони об’єднуються для створення більших регіонів за допомогою жадібного алгоритму. Ці регіони складають остаточні пропозиції регіонів.

2) Компонент CNN деформує пропозиції та виділяє відмінні характеристики як векторний вихід.

3) Витягнуті характеристики передаються в SVM (Support Vector Machine) для розпізнавання об’єктів, що представляють інтерес у пропозиції.

Нижче наведено пояснення особливостей та робота R-CNN (див. рис. 2.1).

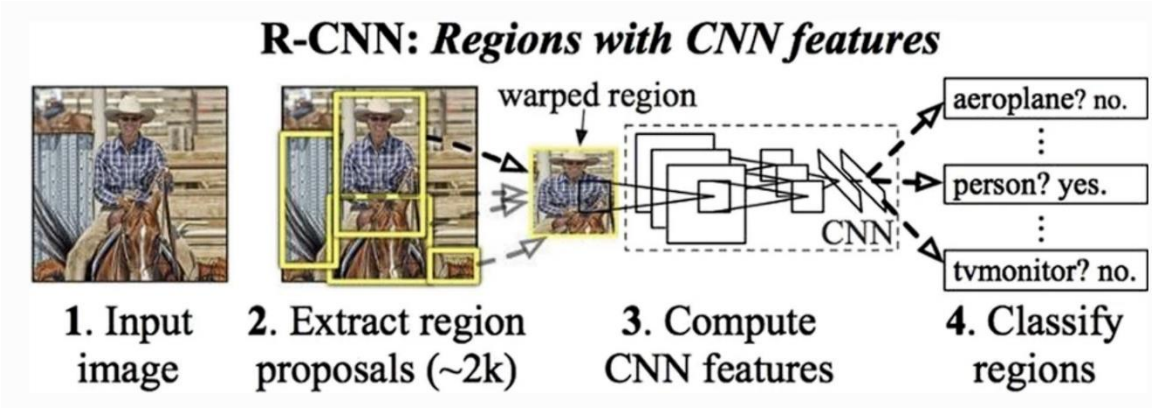


Рисунок 2.1 - Особливості та принцип роботи R-CNN.

Ця техніка мала масу недоліків. Вимога класифікувати ~2000 регіональних пропозицій робить навчання CNN дуже трудомістким процесом, тому це унеможливлює реалізацію в реальному часі, оскільки виконання кожного тестового зображення займає близько 47 секунд.

Крім того, машинне навчання не могло відбутися, оскільки алгоритм вибіркового пошуку є фіксованим алгоритмом, тому це може призвести до створення неідеальних пропозицій регіонів-кандидатів [37].

Fast R-CNN — це алгоритм виявлення об’єктів, який усуває деякі недоліки R-CNN. У ньому використовується підхід, подібний до підходу його попередника, але на відміну від використання пропозицій регіонів, CNN використовує саме зображення для створення згорткової карти ознак, після чого визначаються пропозиції регіонів і викривляються з неї. RoI (регіон інтересів) використовується для зміни форми викривлених квадратів відповідно до попередньо визначеного розміру, щоб повністю підключений

рівень міг їх прийняти, потім клас регіону прогнозується з вектора RoI за допомогою шару SoftMax.

Fast R-CNN є швидшим за свого попередника, оскільки не потрібно надсилати ~2000 пропозицій як вхідні дані до CNN за одне виконання. Операція згортки виконується для генерації карти функцій лише один раз на зображення. Нижче, описано функції та роботу Fast RCNN (див. рис. 2.2).

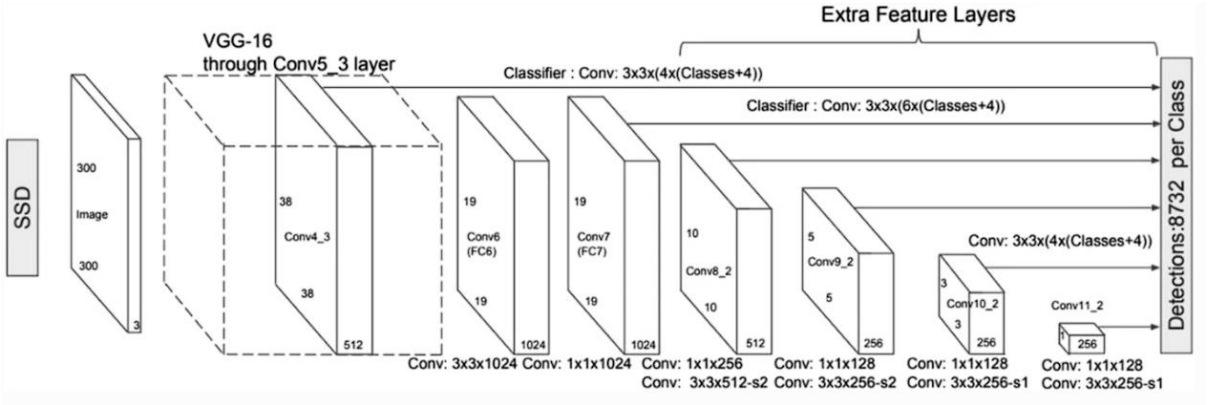


Рисунок 2.2 - Функції та робота Fast R-CNN.

Цей алгоритм демонструє значне скорочення часу, необхідного як для навчання, так і для тестування, порівняно з R-CNN. Але було помічено, що включення регіональних пропозицій значно ускладнює роботу алгоритму, знижуючи його продуктивність.

Fast R-CNN і його попередник використовували вибірковий пошук як алгоритм для визначення пропозицій регіону. Оскільки цей алгоритм дуже забирає час, Faster R-CNN усунув необхідність його впровадження, а натомість дозволив мережі вивчати пропозиції. Як і у випадку Fast R-CNN, згорточна карта виходить із зображення, але окрема мережа замінює алгоритм вибіркового пошуку для прогнозування пропозицій. Потім ці пропозиції змінюються та класифікуються за допомогою об’єднання RoI.

Fast R-CNN пропонує настільки значне покращення порівняно зі своїми попередниками, що тепер його можна використовувати для виявлення об’єктів у реальному часі.

2.1.3 YOLOv3

R-CNN це регіональні згорткові нейронні мережі. Цей метод поєднує регіональні пропозиції для сегментації об’єктів і CNN високої ємності для виявлення об’єктів [36].

У наш час YOLO (You Only Look Once) є одним із найточніших і найбільш влучних доступних алгоритмів виявлення об’єктів. Його було створено на основі нещодавно зміненої та налаштованої архітектури під назвою Darknet. Перша версія була на основі Google Net, яка використовувала тензор для вибірки зображення та передбачала його з максимальною точністю. Тензор генерується на основі подібної процедури та структури, яка також спостерігається в RoI, який об’єднується та компілюється, щоб зменшити кількість окремих обчислень і зробити аналіз швидшим, який використовується в мережі Faster R-CNN. Наступне покоління використовувало архітектуру лише з 30 згорткових шарів, які, у свою чергу, складалися з 19 шарів з DarkNet-19 і додаткових 11 для виявлення природних об’єктів або об’єктів у природному контексті, оскільки використовувався набір даних і метрики COCO. Він забезпечив більш точне виявлення та хорошу швидкість, хоча йому було важко із зображеннями дрібних об’єктів і дрібних пікселів, але версія 7 була найкращою та найточнішою версією YOLO, яка широко використовується завдяки своїй високій точності. Крім того, багаторівнева архітектура зробила виявлення більш точним.

YOLOv3 почала використовувати новітні властивості даркнету, такі як 53 шари, і пройшовши навчання з одним із найнадійніших наборів даних під назвою ImageNet. Використані шари взяті з архітектури Darnnet-53, яка є згортковою за своєю природою. Для виявлення вищезазначені 53 шари були доповнені замість існуючих 19, і ця вдосконалена архітектура була навчена та інструктована за допомогою PASCAL VOC. Після такої кількості додаткових рівнів архітектура підтримує один із найкращих часів відгуку із

запропонованою точністю. Він також дуже корисний для аналізу відео в реальному часі через його швидку дискретизацію даних і методи виявлення об’єктів. Можна помітити, що ця версія є найкращим удосконаленням ML (Machine Learning) з використанням нейронних мереж серед вищезгаданих структур. Ця версія погано працювала із зображеннями малих пікселів, але нещодавні оновлення цієї версії зробили її дуже корисною для аналізу супутникових зображень навіть для міністерств оборони деяких країн. Архітектура працює на 3 різних рівнях, що робить її ефективнішою, але процес трохи повільніший, але це найсучасніше в порівнянні з попередніми архітектурами (див. рис. 2.3).

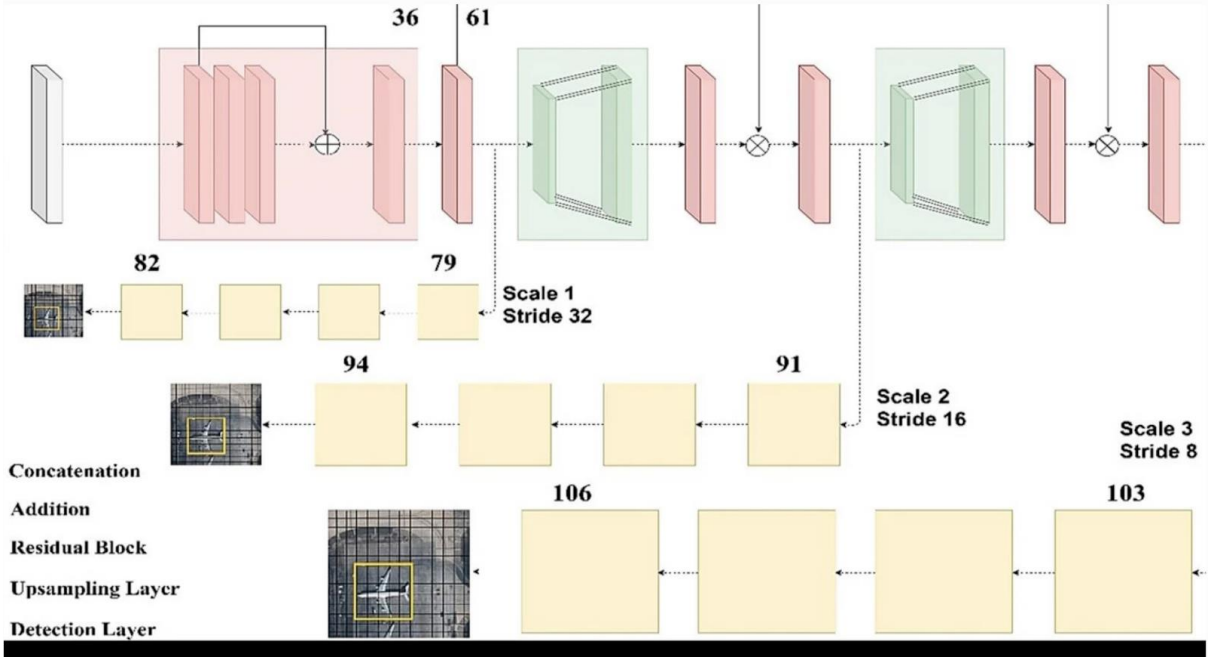


Рисунок 2.3 - Архітектура YOLO

2.1.4 Порівняння швидкодії моделей YOLO

Якщо ви берете участь у проекті виявлення об’єктів, висока ймовірність того, що ви виберете одну з багатьох моделей YOLO. З огляду на кількість доступних моделей виявлення об’єктів YOLO, вибрати найкращу з них складно.

Ось загальні питання над якими варто подумати:

- Яку модель YOLO вибрати для найкращого FPS.
- Як щодо швидкості аналізу на CPU проти GPU.
- Який GPU вибрати.
- Маленька, середня чи велика модель.
- Яка модель YOLO найточніша.

Ці питання стають ще більш актуальними під час створення реальних додатків.

Моя головна мета — відповісти на вищезазначені запитання шляхом ретельного порівняння продуктивності різних моделей виявлення об'єктів YOLO. Цей огляд надасть вам повне та всебічне уявлення про те, яка модель стоїть на місці з точки зору своїх сильних сторін, недоліків тощо.

Для проведення порівняльного аналізу було обрано сімейство моделей YOLOv5, YOLOv6 і YOLOv7. Я обрав ці моделі, тому що це найновіші й одні з найкращих моделей YOLO.

Критерії оцінки ефективності базуватимуться на 3 ключових моментах:

- Точність моделей з точки зору mAP.
- Швидкість висновку в кадрах за секунду (FPS).
- Тип використовуваного графічного процесора: ігровий або графічний процесор для задач пов'язаних із штучним інтелектом: GTX 1080 Ti, RTX 4090, Tesla V100 і Tesla P100.

Разом із вищесказаним я розкриваю як на FPS різних моделей YOLO впливає використання ігрового графічного процесора або графічного процесора для задач пов'язаних із штучним інтелектом.

Далі ми порівняємо продуктивності моделей YOLO на NVIDIA Tesla P100, V100, GTX 1080 Ti та RTX 4090 (див. рис. 2.4). Наша мета — знайти найшвидшу модель після тестування на таких графічних процесорах NVIDIA:

- TESLA P100 GPU.
- TESLA V100 GPU
- GTX 1080Ti GPU

- RTX 4090 GPU

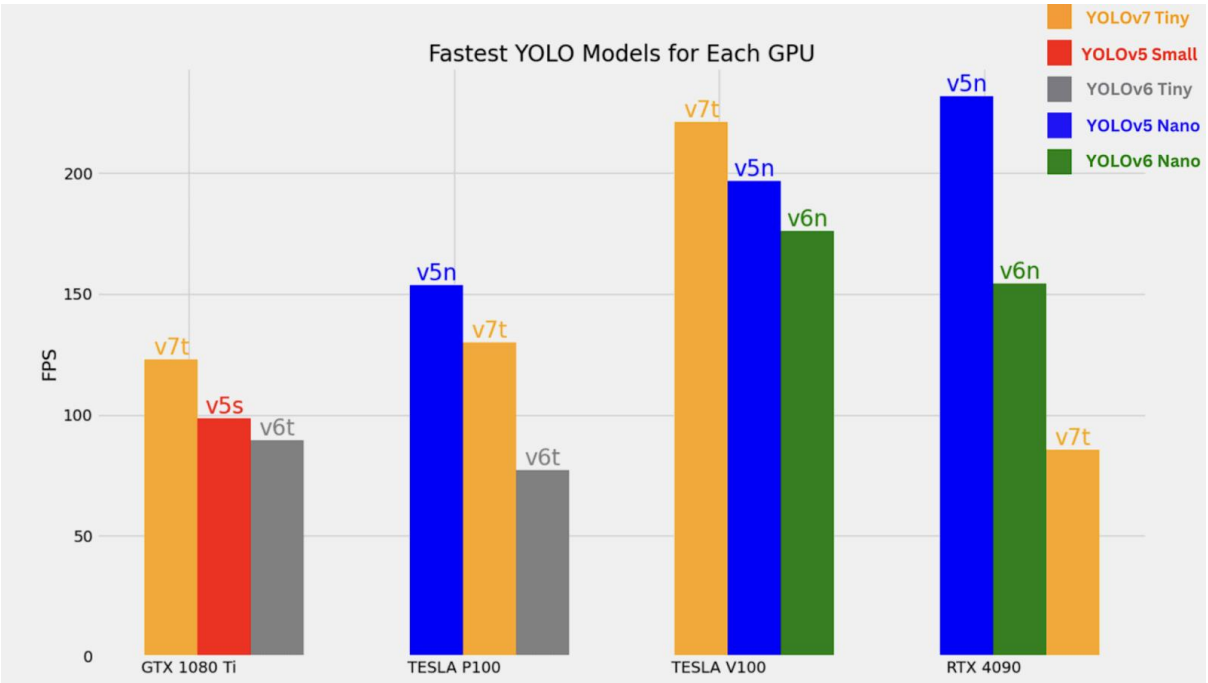


Рисунок 2.4 – Найшвидші моделі YOLO на кожному GPU – RTX, GTX і TESLA.

З наведеного вище графіка ми можемо спостерігати наступне:

- На GPU RTX 4090 і TESLA P100 YOLOv5 Nano є найшвидшим.
- YOLOv7 Tiny забезпечує найбільшу пропускну здатність на GTX 1080 Ti і TESLA V100.
- Моделі YOLOv6 Nano і Tiny не показують того самого FPS, як моделі YOLOv5 і YOLOv7, хоча вони не дуже повільні.

Також порівняємо моделі YOLO на процесорі i7 6850K. Різні моделі було протестовано на CPU і різних GPU відповідно до їх mAP і FPS. На наступних графіках усі результати mAP наведені в 0,50:0,95 IoU.

Почнемо з порівняння графіка mAP і FPS для пропускної CPU. Оскільки процесори не призначені для великих моделей, ми порівнюємо моделі YOLOv5 Nano (P5 і P6), YOLOv6 Nano та YOLOv7 Tiny.

Значення mAP для YOLOv7 Tiny та YOLOv5 Nano P6 було перераховано на зображеннях із роздільною здатністю 640 (див. рис. 2.5).

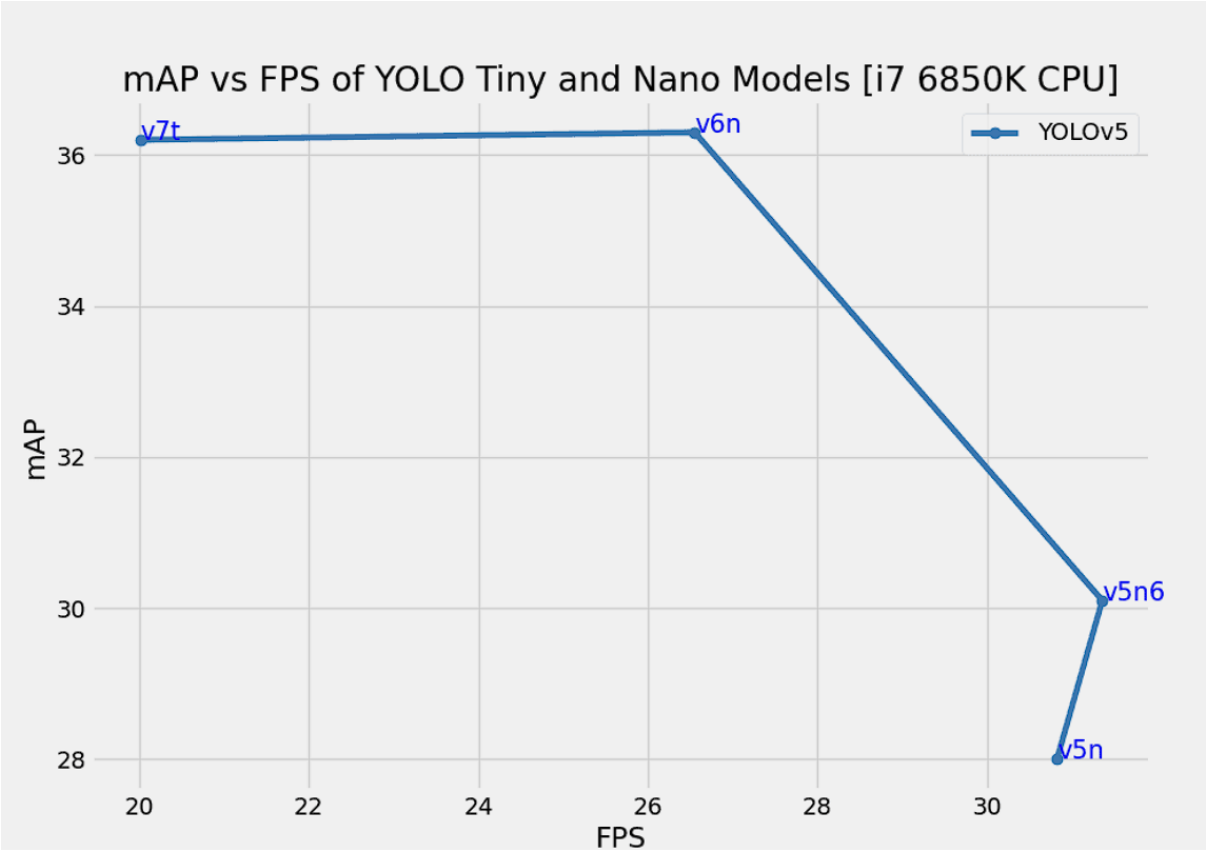


Рисунок 2.5 – Порівняння mAP і FPS для моделей YOLOv5, YOLOv6 і YOLOv7 на CPU.

Тепер давайте розберемося далі, порівнявши mAP і FPS на GPU. У деяких випадках, проводячи експерименти на графічних процесорах GTX (ігровий графічний процесор від NVIDIA), ми можемо помітити аномалії в FPS найменшої та другої найменших моделей у сімействі YOLO. Я припускаю, що це відбувається через тип реалізації шару.

Найменші моделі сімейства YOLO призначені для кінцевих пристроїв і зазвичай не використовують ті самі шари, що й більші моделі. Отже, вони дають трохи нижчий FPS порівняно з другою найменшою моделлю в цьому конкретному сімействі. Такі проблеми вирішуються при використанні новіших графічних процесорів (наприклад, серії RTX) або графічних

процесорів AI (наприклад, серії TESLA V100). Деякі зі старих графічних процесорів такі як TESLA P100, також демонструють цю аномалію.

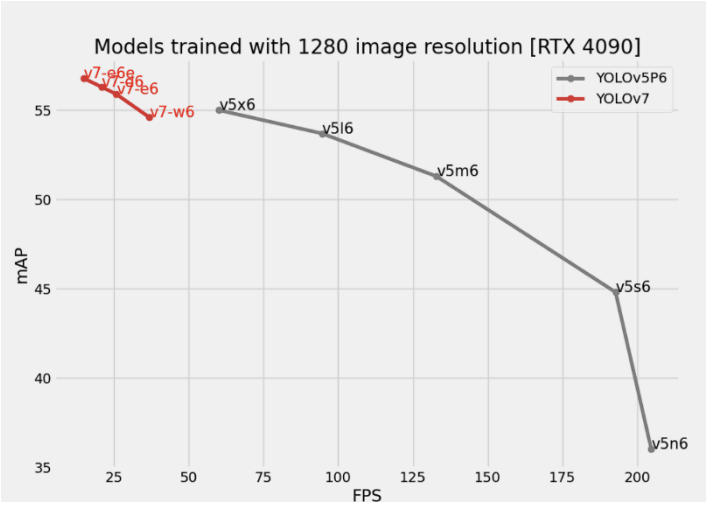
На наступних графіках показано моделі YOLOv5, YOLOv6 і YOLOv7, попередньо навчені на зображеннях із роздільною здатністю 640. Висновок також проводився на відео зі зміненим розміром кадрів до роздільної здатності 640. Я виключив модель YOLOv7-Tiny з експериментів GPU, оскільки вона була попередньо навчена на зображеннях із роздільною здатністю 416. Крім того, ми вже бачили, як модель YOLOv7 Tiny працює на зображеннях із роздільною здатністю 640 під час роботи на CPU.

Наступні експерименти проводилися на графічному процесорі NVIDIA RTX 4090 (див. рис. 2.6).

Рисунок 2.6 – mAP та FPS для попередньо навчених моделей із роздільною здатністю 640 на графічному процесорі RTX 4090

Середні та великі моделі сімейства YOLOv5 можуть працювати на RTX 4090 зі швидкістю понад 100 FPS.

На наступному графіку показано порівняння mAP і FPS попередньо підготовлених моделей з роздільною здатністю 1280. Висновок проводився на кадрах з роздільною здатністю 1280 (див. рис. 2.7).



Рисуснок 2.7 – mAP та FPS для попередньо навчених моделей із роздільною здатністю 1280 на графічному процесорі RTX 4090.

Модель YOLOv5m P6 працює зі швидкістю понад 100 FPS. Однак модель YOLOv5l P6 падає нижче 100 FPS.

Решта наступних графіків показує порівняння mAP і FPS попередньо навчених моделей з роздільною здатністю 640 на TESLA P100, TESLA V100 і GTX 1080 Ti GPU (див. рис. 2.8, 2.9, 2.10).

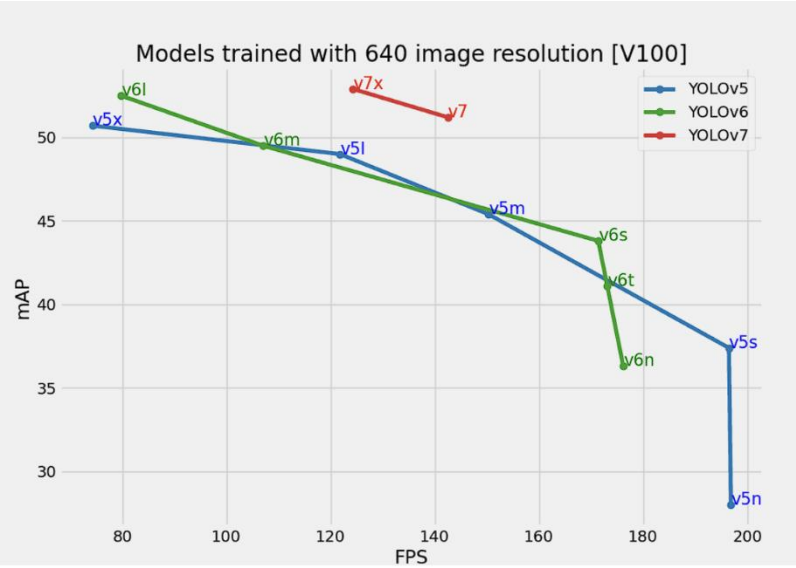


Рисунок 2.8 – mAP та FPS для попередньо навчених моделей із роздільною здатністю 640 на GPU TESLA V100.

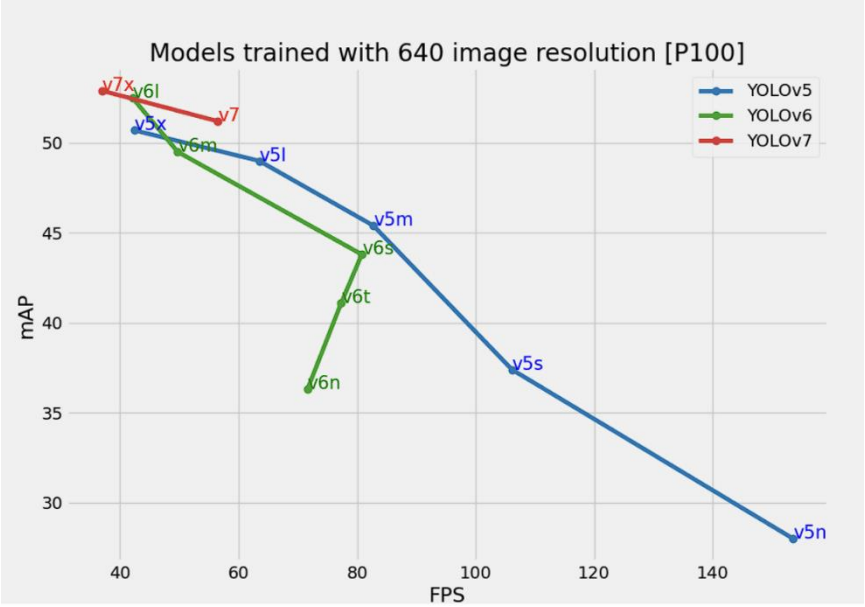


Рисунок 2.9 – mAP та FPS для попередньо навчених моделей з роздільною здатністю 640 на GPU TESLA P100.

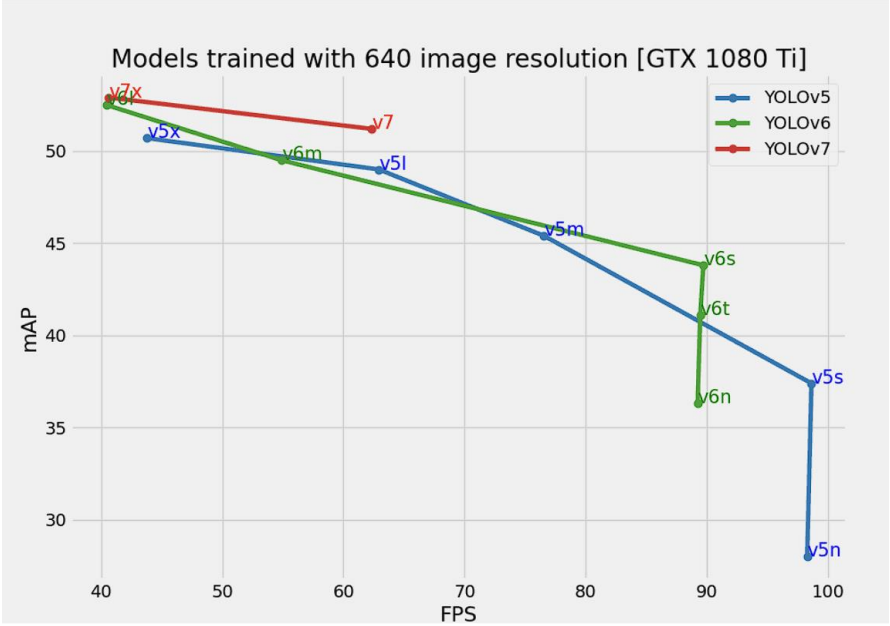


Рисунок 2.10 – mAP проти FPS для попередньо навчених моделей із роздільною здатністю 640 на графічному процесорі GTX 1080 Ti.

2.1.5 Обгрунтування вибору моделі від YOLO

Використання великих моделей, таких як YOLOv5x або x6, має бути останнім засобом. Із сімейства YOLOv5 дотренування YOLOv5m із

роздільною здатністю 640 дасть хороші результати. Він здатний працювати зі швидкістю понад 80 кадрів в секунду навіть на старішому графічному процесорі, такому як TESLA P100. У той же час показник mAP дорівнює 45,4.

YOLOv6m також є досить хорошою моделлю з 49,5 mAP і майже 50 FPS на графічному процесорі TESLA P100. Навчання спеціального набору даних моделі YOLOv6 Medium має дати одні з найкращих результатів.

Так само дотреновування YOLOv7 забезпечує хороший баланс між FPS і mAP. Вони можуть працювати зі швидкістю 56 FPS, даючи при цьому більше 51 mAP.

В подальшому в даній роботі я буду використовувати найновішу наявну модель YOLOv7 на графічному процесорі GEFORCE RTX 3060.

2.2 Порівняння підходів для ReID особи

Непостійне відстеження людей на великих локаціях, тобто пошук особи, яка вас цікавить, у різних місцях, що не збігаються, на різних камерах, є критично важливим завданням, відомим як повторна ідентифікація людей. Більш формальне визначення парадигми повторної ідентифікації можна підсумувати таким чином: заданий набір людей, отриманий у місці X у момент часу T0, повторна ідентифікація має на меті зіставити його предмети з суб’єктами в наборі галереї, зібраним у іншому місці Y у час T1. Біометрія здається дієвим для вирішення цієї проблеми, коли потрібно стежити за людьми. Загальновизнано, що біометричне розпізнавання в умовах контрольованого збору даних є відносно старою технологією, яка довела свою ефективність для учасників агенцій безпеки, громадського транспорту, урядів та в незалежних ініціативах з оцінки технологій (Phillips та ін., 2010).

Однак доцільність біометричних методів в умовах неконтрольованого збору даних все ще викликає значний і часто добре вмотивований скептицизм. Саме з цієї причини повторна ідентифікація людей, які переміщуються по

різних локаціях, покритих неперекриваючими камерами, може бути ідеальним сценарієм для тестування технології в умовах неконтрольованого збору даних.

Оскільки дані зображення (відео) є досить великими, завдання полягатиме в тому, щоб розробити швидкі стратегії для біометричного позначення людини.

Сценарії застосування забезпечують надзвичайно складне завдання аналізу відео. Відеопотоки зніматимуться як у приміщенні, так і на вулиці, а розмір людей на зображеннях (у пікселях на ціль) коливатиметься від кількох десятків до кількох сотень пікселів. Біометрія обличчя та ходи є особливо перспективною для повторної ідентифікації, оскільки вона може працювати на відстані та не потребує детального та/або зображення високої роздільної здатності суб'єкта та/або його біометричних ознак.

2.2.1 ReID особи

Однією з найбільш критичних проблем, пов'язаних із повторною ідентифікації людини, є розпізнавання тієї самої людини, яку спостерігають різні камери, які, можливо, не перекриваються у різні моменти часу та в різних місцях.

Ретельний опис основних проблем і відкритих викликів, пов'язаних з дистанційною ідентифікацією обличчя, а також повторною ідентифікацією обличчя, вже були раніше представлені роботи, які аналізують вплив кожного з основних аспектів, що характеризують необмежені прикладні сценарії, як-от відстань між камерою та об'єктом, роздільна здатність камери, освітлення на відкритому повітрі, зміна пози, розмиття, оклюзії та погодні артефакти.

Зовсім недавно запропонували дослідження, спеціально присвячене повторній ідентифікації. Техніка повторної ідентифікації може бути проаналізована відповідно до кількох аспектів:

1) Кількість відстежуваних суб'єктів.

2) Деталізація вилучених ознак (сегментовані області, краплі, ключові точки).

3) Можливе перекриття полів зору камери.

4) Використання контекстної інформації.

5) Ефективність і результативність.

Нижче буде згруповано низку репрезентативних методів відповідно до критеріїв, беручи до уваги, що одна система може відповідати більш ніж одному категорії.

Одні з найвідоміших підходів повторної ідентифікації:

1) Відстеження однієї та кількох цілей на непересічних камерах. Як метод, запропонований в [38], так і метод, запропонований в [39] базуються на відстеженні однієї людини, хоча використовують аналіз дуже різних характеристик. Модель зовнішнього вигляду, запропонована першим, заснована на інкрементній гістограмі основного колірного спектру (IMCHSR). Це досягається шляхом поєднання онлайн-алгоритму кластеризації кольорів k-середніх і поступового використання кадрів. Другий виконує сегментацію моделі суміші Гауса (GMM) на відеокадрах і включає лише дані переднього плану на наступних кроках. Постійність кольорів між камерами потім обробляється для покращення якості сегментації перед вилученням кількох дескрипторів осіб.

Вони базуються на зовнішньому вигляді (середній колір, коваріація, домінуючий колір MPEG-7) і на просторово-часових властивостях (на основі фільтра Калмана, топологічні) спостережень. Новітніші методи спрямовані на відстеження більшої кількості об’єктів одночасно, щоб значно підтримати функції відеоаналітики, наприклад, розпізнавання траєкторії, дії та взаємодії між людьми та об’єктами.

Особливо критичний аспект пов’язаний із відсутністю перекриття полів огляду камери, що є звичайним для більшості реальних випадків, де камери можуть навіть охоплювати велику географічну територію.

Як наслідок, такі характеристики як освітлення, колір або атмосферні умови, можуть суттєво відрізнятися для різних камер, тому це збільшує увагу

до таких елементів під час обробки відео. Зокрема, потреба в корекції/відображенні кольорів, у функціях, стійких до спотворень, для характеристики осіб/об'єктів, часто об'єднаних контекстною інформацією, і в оптимізації, спрямований на підвищення ефективності. Структура, запропонована в [40] ілюструє приклад типової структури таких архітектур.

Пропонована система складається з двох основних частин: перша реалізує автоматичний процес виділення силуету на основі адаптивного GMM у спільному просторово-колориметричному просторі ознак; друга частина реалізує техніку класифікації, яка використовує дискримінаційну характеристику розрідженого представлення сигналів для повторної ідентифікації людей. У [41] автори зосереджуються на проблемах, пов'язаних із системами кількох камер, які не перекриваються. Зокрема, на можливо дуже різному зовнішньому вигляді одного й того самого об'єкта через різні параметри камери чи умови освітлення. Вони показують, що функції передачі яскравості від даної камери до іншої камери лежать у низьковимірному підпросторі, який можна вивчити під час фази навчання, і демонструють, що цей підпростір можна використовувати для обчислення зовнішньої схожості. З цією метою вони використовують систему оцінки Maximum A Posteriori (MAP), використовуючи ознаки розташування та зовнішнього вигляду. У [41] автори далі розширюють цей підхід, спостерігаючи, що в більшості випадків люди або транспортні засоби, як правило, слідують однаковими шляхами, тому пропонують новий алгоритм, який використовує цю відповідність. Алгоритм фіксує взаємозв'язки між камерами у формі багатовимірної щільності ймовірності просторово-часових змінних за допомогою оцінки щільності ядра, використовуючи цю властивість. Моделі простору-часу та зовнішнього вигляду для відстеження об'єктів вивчаються на етапі навчання.

У роботі [42] оцінюють невідому функцію передачі кольору між парами непересічних камер за допомогою методу калібрування кольору. Розроблена

модель працює на зразках кольоровості для підвищення тимчасової стабільності передатної функції між будь-якою парою камер.

2) Постійність біометричних ознак. Важливим аспектом, який багато методів, не враховують – час. Вони ефективні, щоб повторно ідентифікувати ту саму особу на відеозаписах, знятих несумісними камерами, але у невеликому проміжку часу. Коли час збільшується (тобто більше днів), неможливо припустити, що людина носить однакові сукні, тому в цьому випадку багато методів ризикують бути абсолютно неефективними. Навпаки, методи, засновані на біометричних ознаках, також можуть бути застосовані в контекстах такого роду, тому такі методи швидко поширюються.

У контексті повторної ідентифікації особи обличчя пропонує найкращий компроміс між конкретною здійсненністю систем і точністю розпізнавання: це пов’язано з тим фактом, що поверхня обличчя безперечно більша за інші фізичні біометричні характеристики, наприклад, райдужної оболонки ока, і що для цього потрібне безконтактне отримання, на відміну від, відбитків пальців.

У той же час він гарантує прийнятну точність навіть у неконтрольованих умовах. У будь-якому випадку слід враховувати, що розпізнавання обличчя в необмежених умовах, включаючи нестабільні умови збору даних, є дуже складним завданням. Насправді багато алгоритмів працюють добре, коли було отримано обмежені зображення обличчя, але їх продуктивність значно погіршується, коли тестові зображення містять варіації, яких немає в навчальних зображеннях. Раніше висвітлювались деякі з ключових проблем дистанційного розпізнавання облич і представляють базу даних віддалених облич, яка була отримана в необмеженому зовнішньому середовищі.

Проблема повторної ідентифікації обличчя за допомогою непересічних камер глибоко досліджена в роботі [43]. Розпізнавання обличчя базується на модифікованому консенсусному перетворенні, і підмножина кадрів сканується для різних можливих орієнтацій обличчя з урахуванням можливої оклюзії між суб’єктами. Відгуки користувачів використовуються для

допомоги предметному класифікатору, вказуючи на найкращі та найгірші збіги кандидатів.

Обмежена роздільна здатність пристроїв збору даних, або велика відстань до цілі, м'які біометричні ознаки, такі як волосся, шкіра та тканинні плями, можуть суттєво сприяти процесу повторної ідентифікації обличчя, як запропоновано в [44]. У цій статті основна мета авторів полягає в розгляді варіацій поз, пов'язаних з мережею камер систем відеоспостереження, тому це одна з типових проблем розпізнавання обличчя, і в цьому контексті вона ускладнюється процедурою отримання.

Ідея полягає в тому, щоб імітувати спосіб, у який люди покращують точність розпізнавання frontal-to-side, використовуючи прості та очевидні ознаки, наприклад, блондини, коричневі, руді та чорні для ознаки кольору волосся. Запропонований алгоритм аналізує колір і текстуру вибраних патчів, потім створюється комбінований класифікатор, щоб посилити та об'єднати всі розглянуті ознаки. Результати не кваліфікують цей підхід як кандидата на надійну повторну ідентифікацію, а скоріше як систему скорочення для рішень високого рівня безпеки для контролю доступу або як частину багато-біометричних систем.

Іншою (поведінковою) біометрією, яка може бути використана для повторної ідентифікації людини, є аналіз ходи, оскільки він може працювати на відстані навіть при низькій роздільній здатності та без співпраці.

В роботі [45] пропонують ієрархічну структуру для повторної ідентифікації некооперативного суб'єкта шляхом поєднання ходи з фазою руху в просторово-часовій моделі. Вони використовують три ознаки: дві для динаміки руху об'єкта, а третя – похідна від просторово-часової моделі мережі камери. Метод, поєднавши їх, може відстежувати об'єкти, навіть якщо вони змінюють швидкість або зупиняються на деякий час у сліпій зоні.

3) Перекриття полів зору. Хоча це часто недосяжно з очевидних практичних міркувань економії, кілька камер, що перекриваються, можуть

забезпечити як більш надійне виявлення, так і 3D-оцінку розташування об’єктів, охоплюючи особливості об’єкта з усіх боків. Коли система обробляє модель перекриття, поля зору, що перекриваються, також можуть підтримувати плавне перемикання між камерами під час відстеження.

У роботі [46] відстеження триває з однієї камери, доки система не передбачить, що активна камера втратить хороший кут огляд об’єкта. У цей час відстеження перемикається на камеру, яка, за оцінками, забезпечує кращий огляд і потребує найменшого перемикання.

Задіяно три основні модулі:

1) Відстеження одного перегляду (SVT).
2) Відстеження переходу між кількома видами (MVTT).
3) Автоматичне перемикання камери (ACS).

Під час SVT байєсівський класифікатор знаходить найбільш вірогідний збіг суб’єкта в наступному кадрі. MVTT передбачає як просторову, так і часову оцінку руху. ACS покладається на алгоритм передбачення. В дослідженні [47] пропонують систему відстеження людей із кількома камерами, засновану на регіональному стереоалгоритмі, який знаходить 3D-точки всередині об’єкта з інформації про регіони, що належать об’єкту, отриманої з двох різних видів. Асоціація між кадрами використовує кольорові моделі горизонтальних ділянок людини, які також можна використовувати для повторної ідентифікації протягом більших інтервалів часу.

2.2.2 Підсумок досліджених підходів

Відкритою проблемою для повторної ідентифікації є обчислювальна вартість, особливо з мережами камер для програм реального часу. В статті [48] використовують структуру багатокомпонентної відмінності (MCD), де метод, заснований на зовнішньому вигляді, використовує відстані, засновані на відмінності. Оскільки дескриптори на основі несхожості є векторами дійсних чисел, вони набагато компактніші, а час на їх зіставлення значно скорочується.

Крім того, представлення на основі несхожості, яке використовується в MCD підході, дозволяє дуже швидко реалізувати повторну ідентифікацію.

Нижче наведено таблицю авторів, використаних ними наобрів даних та підхід (див. табл. 2.1).

Таблиця 2.1 – Роботи у сфері повторної ідентифікації особи.

№	Автор	Набір даних	Підхід
1	Bauml and Stiefelhagen	CAVIAR, Власний набір	Точки інтересу, локальні дескриптор-и
2	Bauml	Власний набір	Кілька детекторів на основі Modified Census трансформації
3	Cai and Aggarwal	Власний набір	Схеми байєсівської класифікації, багатовимірний нормальний розподіл
4	Colombo	Власний набір	Gaussian Mixture Model (GMM)
5	Dantcheva and Dugelay	FERET	AdaBoost використовується на волосся, шкіру та одяг патчами
6	Hu	NLPR	головна вісь тіла
7	Javed	Власний набір	Багатовимірна концентрація ймовірності просторово-часових змінних
8	Javed	Власний набір	На основі навчання, оцінка MAP локації та зовнішнього вигляду
9	Jeong and Jaynes	Власний набір(Terrascope)	Оцінка невідомої функції передачі кольору між парами камер

Продовження таблиці 2.1

№	Автор	Набір даних	Підхід
10	Madden	Власний набір	Major Colour Spectrum Histogram Representation (MCSHR)
11	Mittal and Davis	Власний набір	Регіональний стереоалгоритм, байєсова класифікація
12	Roy	Власний набір	хода, фаза руху, просторово-часова модель
13	Truong Cong	Власний набір	колірно-позиційна гістограма, спектральний аналіз, SVM

2.2.3 Обгрунтування вибору підходу

Ефективність системи повторної ідентифікації особи, очевидно, пов'язана з поставленою задачею. Обробка в реальному часі накладає серйозні обмеження. Алгоритми машинного навчання можуть вимагати свого роду навчання, яке може бути неможливим у дуже динамічних умовах. Використання однієї чи кількох камер, а також полів зору, що перекриваються або не перекриваються, також має свої наслідки. Задачі де є можливість використовувати тільки одну камерою зараз обмежені у виконанні. З іншого боку, хоча кути огляду, що накладаються, надають більш надійну інформацію та дозволяють більш «інтелектуальну» обробку, слід також враховувати, що можлива тріангуляція вимагає обчислень, тому вибір системи для використання прив'язаний до реальних експлуатаційних потреб.

При виникненні проблем повторної ідентифікації необхідно вирішити безліч питань, включаючи кластеризацію та вибір відповідних регіонів, розпізнавання по частинах, виявлення аномалій і змін, вибірку та відстеження, швидке індексування та пошук, аналіз чутливості та їх остаточну інтеграцію. Також є багато інших перешкод для теперішніх і майбутніх досліджень. Багато проблем запозичено з додаткових областей, таких як біометричне розпізнавання, так що рішення також можуть бути адаптовані та вдосконалені.

Завдання, яке вирішується, полягає в управлінні на основі фактичних даних для поступового збору та додавання корисної інформації до даних, щоб генерувати знання та відповідне ініціювання заздалегідь визначених дій.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Підготовка середовища

Для цієї роботи було використано середовище Visual Studio Code (див. рис. 3.1) [49] для розробки програми на WSL2 [50]. Мовою програмування було обрано python, так як вона є легкою у вивченні та легко використовується на практиці. На даний час, python вважається найкращою мовою програмування в сфері DS, а також вже існує дуже велика база готовий реалізації для вирішення різних задач на цій мові.

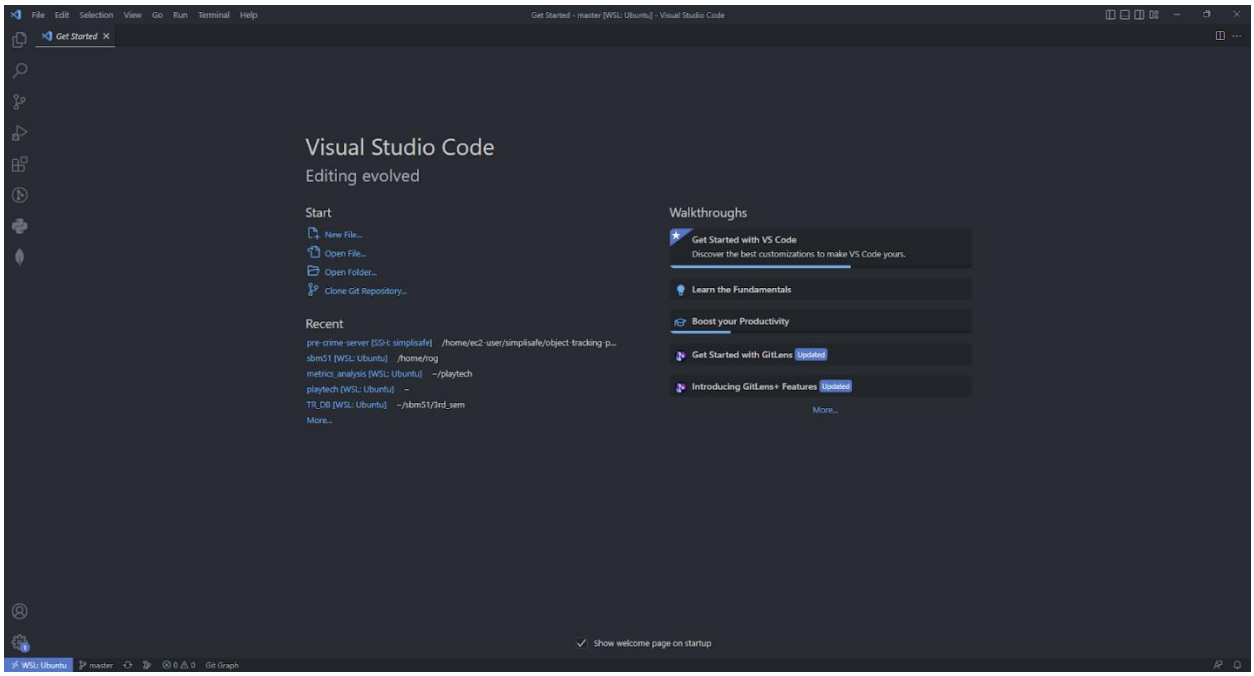


Рисунок 3.1 – Середовище розробки VSCode на WSL2.

Для контролю версійності python було використано програму простого керування версіями – pyenv (див. рис. 3.2) [51].

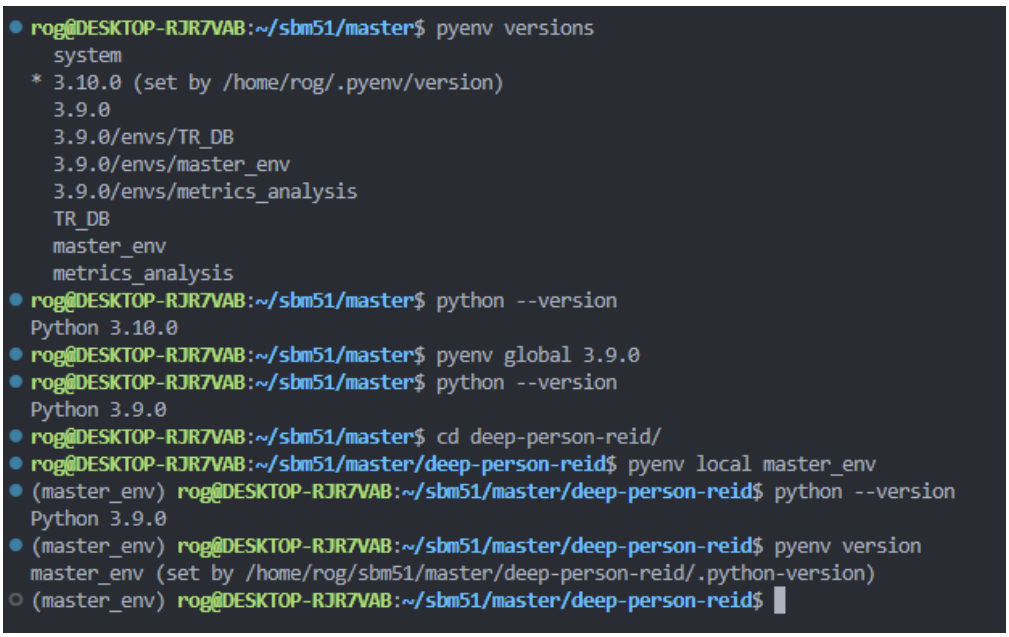


Рисунок 3.2 – Pyenv python versions manager

Можливості pyenv:

- Дозволяє змінювати глобальну версію Python для кожного користувача.
- Забезпечує підтримку версій Python для кожного проекту.
- Дозволяє замінити версію Python за допомогою змінної середовища.
- Шукає команди в кількох версіях Python одночасно.

Створено віртуальне середовище Python для версії 3.9.0 під назвою master_env, активовано та зроблено локальним для нашого проекту (див. рис. 3.3).

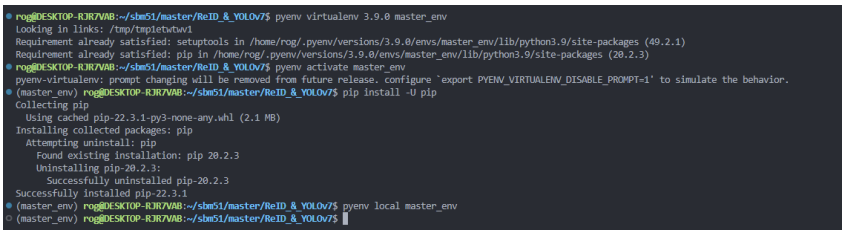


Рисунок 3.3 – Створення віртуального середовища Python версії 3.9.0

3.2 Особливості встановлення Torchreid

Склоновано Torchreid [52] репозиторій на локальну машину командою нижче (див. рис. 3.4):

git clone https://github.com/KaiyangZhou/deep-person-reid.git

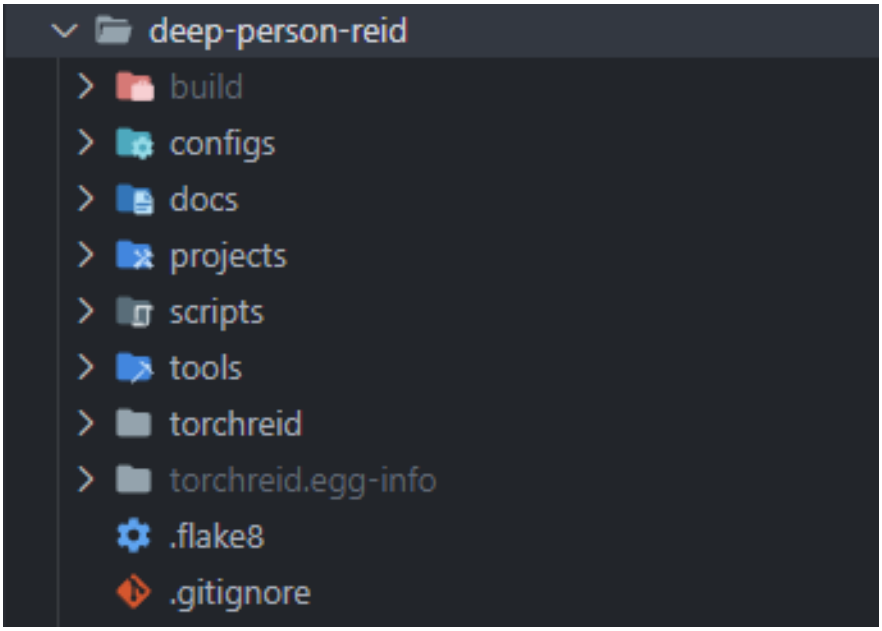


Рисунок 3.4 – Склонований репозиторій Torchreid

Встановлено необхідні пакети командою нижче:

pip install -r deep-person-reid/requirements.txt

Встановлено пакет torch для того, щоб з’явилась можливість використовувати CUDA ядра для швидшої обробки даних та отримання результатів.

https://pytorch.org/get-started/previous-versions/ – це посилання дозволить переглянути, які на даний момент є версії пакету torch та встановити відповідний для вашої системи. Для перевірки було використано nvidia-smi (див. рис. 3.5).OS: WSL, Linux):

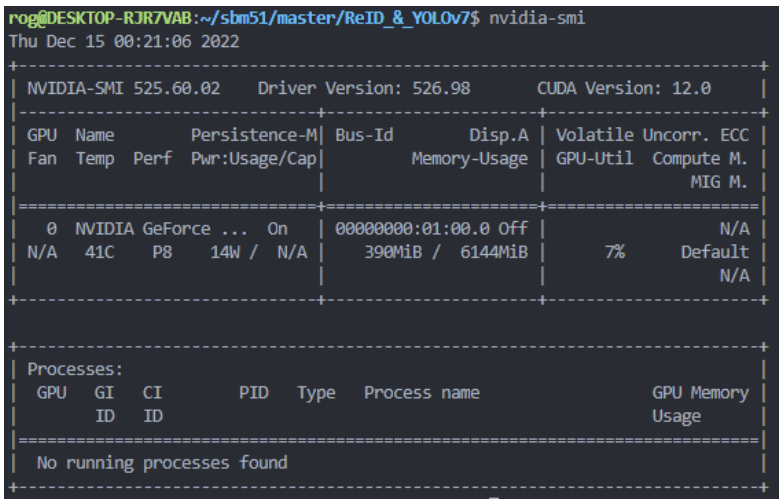


Рисунок 3.5 - Результат виводу команди nvidia-smi

За допомогою команди нижче встановлено потрібний пакет на локальну машину (див. рис. 3.6):

pip install torch==1.12.0+cu116 torchvision==0.13.0+cu116 torchaudio==0.12.0 --extra-index-url https://download.pytorch.org/whl/cu116

Проте, це може зайняти трохи часу, так як розмір загального пакет є досить великим.

Якщо виникають проблеми з доступністю CUDA, то є можливість встановити їх підтримку керуючись офіційним гайдом NVIDIA – https://docs.nvidia.com/cuda/wsl-user-guide/index.html

Рисунок 3.6 - Встановлення пакету torch для віртуального середовища master_env

Далі потрібно перейти безпосередньо в папку репозиторію Torchreid, щоб добавити його як пакет у віртуальне середовище за допомогою команд нижче (див. рис. 3.7, 3.8):

cd deep-person-reid/

python setup.py develop

```
Best match: importlib-metadata 5.1.0
Adding importlib-metadata 5.1.0 to easy-install.pth file

Using /home/rog/.pyenv/versions/3.9.0/envs/master_env/lib/python3.9/site-packages
Searching for oauthlib==3.2.2
Best match: oauthlib 3.2.2
Adding oauthlib 3.2.2 to easy-install.pth file

Using /home/rog/.pyenv/versions/3.9.0/envs/master_env/lib/python3.9/site-packages
Searching for pyasn1==0.4.8
Best match: pyasn1 0.4.8
Adding pyasn1 0.4.8 to easy-install.pth file

Using /home/rog/.pyenv/versions/3.9.0/envs/master_env/lib/python3.9/site-packages
Searching for zipp==3.11.0
Best match: zipp 3.11.0
Adding zipp 3.11.0 to easy-install.pth file

Using /home/rog/.pyenv/versions/3.9.0/envs/master_env/lib/python3.9/site-packages
Finished processing dependencies for torchreid==1.4.0
```

Рисунок 3.7 - Встановлення Torchreid

```
from torchreid.utils import FeatureExtractor
✓ 1.9s
/home/rog/.pyenv/versions/3.9.0/envs/master_env/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress
  from .autonotebook import tqdm as notebook_tqdm


extractor = FeatureExtractor(
    model_name="osnet_x0_75",
    device="cuda"
)

image_list = [
    "test_reid.png",
]

features = extractor(image_list)
print(features.shape) # output (5, 512)
✓ 4.5s
Successfully loaded imagenet pretrained weights from "/home/rog/.cache/torch/checkpoints/osnet_x0_75_imagenet.pth"
** The following layers are discarded due to unmatched keys or layer size: ['classifier.weight', 'classifier.bias']
Model: osnet_x0_75
- params: 1,299,224
- flops: 571,754,536
torch.Size([1, 512])
```

Рисунок 3.8 – Запуск отримання вектора для тестової картинки людини.

3.3 Налаштування YOLOv7

Склоновано Torchreid [52] репозиторій на локальну машину командою нижче (див. рис. 3.4):

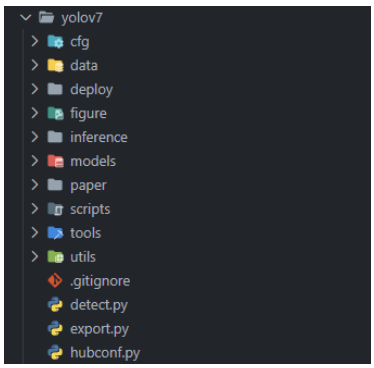


Рисунок 3.9 – Склонований репозиторій YOLOv7

Перед тим як встановити необхідні пакети потрібно закоментувати ті, які вже було встановлено раніше, тобто torch та torchvision у файлі yolov7/requirements.txt. Після цього використовуємо команду нижче:

pip install -r yolov7/requirements.txt

Завантажуємо відповідну модель для того, що отримати результати виявлення об’єктів. Для цього потрібно повернутись в репозиторій YOLOv7 та знайти секцію в README де знаходиться таблиця моделей та їхні метрики (див. рис. 3.10).

Performance

MS COCO

Model	Test Size	AP^{test}	AP_{50}^{test}	AP_{75}^{test}	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 *fps*	2.8 *ms*
YOLOv7-X	640	53.1%	71.2%	57.8%	114 *fps*	4.3 *ms*
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 *fps*	7.6 *ms*
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 *fps*	12.3 *ms*
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 *fps*	15.0 *ms*
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 *fps*	18.7 *ms*

Рисунок 3.10 – Таблиця наявних моделей YOLOv7

В цьому проекті було використано найкращу модель відносно метрик AP. Команди для завантаження моделі нижче (див. рис. 3.11):

cd yolov7

mkdir weights

wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7-e6e.pt -P weights/

Рисунок 3.11 – Завантаження моделі YOLOv7-E6E

Зроблено тестове виявлення людей (див. рис. 3.12, 3.13). Команда нижче:

python detect.py --weights yolov7.pt --conf 0.25 --img-size 640 --source inference/images/image2.jpg --weights weights/yolov7-e6e.pt

Рисунок 3.12 – Оригінальна картинка

Рисунок 3.13 – Проведено виявлення об’єктів за допомогою YOLOv7

Можна побачити, що модель для знаходження об’єктів на фото/відео визначила, що на картинці є Людина1 (впевненість передбачення дорівнює 0.96), Людина2 (впевненість передбачення дорівнює 0.95), Спортивний м’яч (впевненість передбачення дорівнює 0.95).

3.4 Створення набору даних для тренування та оцінювання

Встановлено додаткові пакети для подальшої роботи:

```
# additional requirements
scikit-learn==1.2.0
pymongo==4.3.3
openpyxl==3.0.10
```

pip install -r requirements.txt

Набір даних було створено з власних даних(відео) за допомогою написаного детектора об’єктів та отримання ознак (див. Додаток В) та збереження виявлення людей в окремі кластери (див. Додаток Б).

Створено початкову версію набору даних за допомогою YOLOv7 та Torchreid, потім вручну перевірено та перенесено неправильно розподілені об’єкти в інакші кластери (див. рис. 3.14).

Також показано результати згенерованих кластерів для подальшої обробки (див. рис. 3.15, 3.16, 3.17).

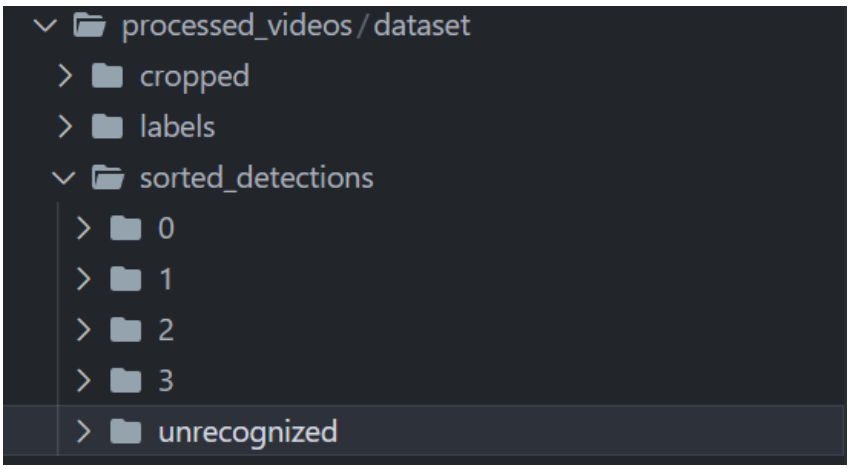


Рисунок 3.14 – Результат створення початкового датасету, папка “sorted_detections”

Рисунок 3.15 – Згенерований кластер 0, взято 10 перших людей

Рисунок 3.16 – Згенерований кластер 1, взято 10 перших людей

Рисунок 3.17 – Згенерований кластер «unrecognized», який не пройшов перевірку сумісності з іншими кластерами.

Зроблено перевірку кожного кластеру вручну і переміщено неправильно виявлених людей в правильні кластери. За допомогою скріпта (див. Додаток Г) кожна назва картинки була змінена відповідно до її кластеру та скопійовано до відповідної директорії train або query для подальшого тренування.

Створено тестовий датасет за схожою схемою. Використано скріпт (див. Додаток Б) та створено кластери, перевірено кластери вручну, переміщено знайдених людей в загальну папку тестового набору даних та розподілено відносно їх імен. Картинку кожної людини було відповідно пронумеровано {index_of_images_list}.jpg.

3.5 Тренування Torchreid

Скрипт для тренування було розроблено відповідно до інструкції поданій в get-started-30-seconds-to-torchreid та use-your-own-dataset. Створені файли можна переглянути нижче (див. Додаток Д, Е).

Після налаштування конфігу та створення допоміжних функцій (див. Додаток А, И) було запущено тренування (див. рис. 3.18, 3.19):

python train.py

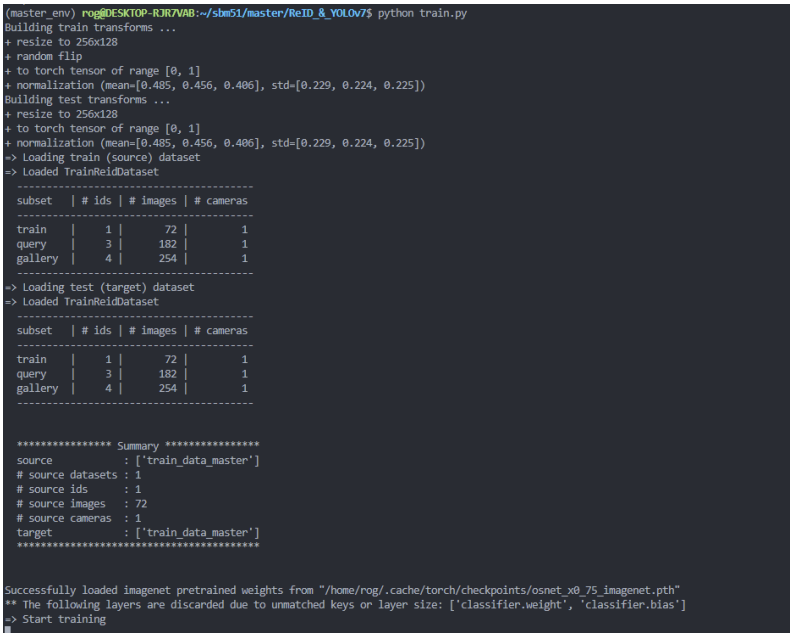


Рисунок 3.18 – Тренування OSNet_x0_75 моделі на наших даних

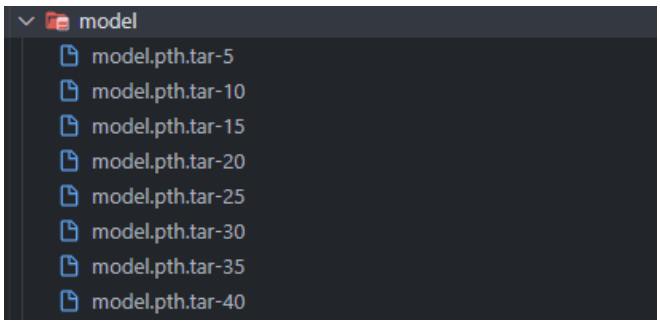


Рисунок 3.19 – Результати тренування відносно встановленої частоти епохи для зберігання моделі.

3.5 Оцінка моделей

З допомогою використання пакету pymongo в python було написано просте підключення до бази даних:

```
import pymongo
from pymongo.database import Database
mongodb = pymongo.MongoClient("127.0.0.1:27017",
uuidRepresentation="standard")
reid_db = Database(mongodb, "reid")
wanted_people_collection = reid_db["wanted_people"]
```

Добавлено тестову людину в базу даних MongoDB (див. рис. 3.20).

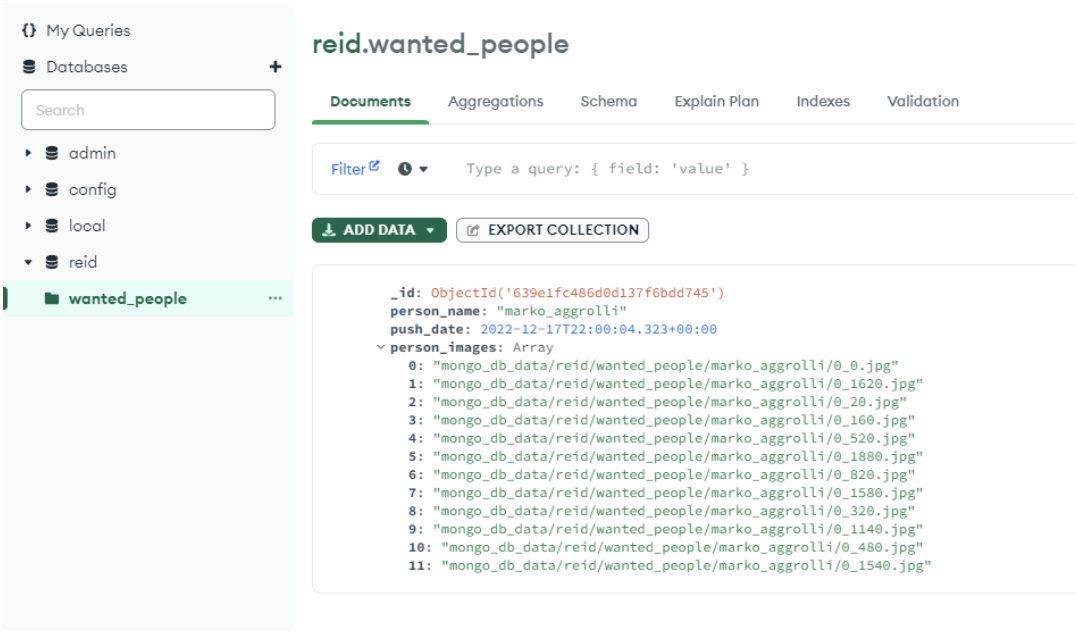


Рисунок 3.20 – Тестовий об’єкт людини в MongoDB

З іншого відео було обрано 10 радномних вирізаних фото людини за допомогою YOLOv7 (див. рис. 3.21).

Рисунок 3.21 – Набір кадрів людини для подальшого пошуку на інших відео.

Проведено оцінку моделей за допомогою розробленого скрипта (див. Додаток Є), отримано такі метрики для кожної моделі: accuracy, precision, recall, f1_score (див. табл. 3.1).

Таблиця 3.1 – Результати оцінки моделей

Model name	accuracy	precision	recall	f1_score	min cosine similarity (мінімальне значення схожості векторів)
osnet_x1_0	0.8	0.66	1.0	0.79	0.65
osnet_x0_75	0.86	0.76	1.0	0.87	0.65
osnet_x0_5	0.72	0.62	1.0	0.76	0.65
osnet_x0_25	0.78	0.63	1.0	0.77	0.65
натренована модель(основа – osnet_x0_75)	0.92	0.85	1.0	0.92	0.997

Перший висновок, який можна зробити відносно даної таблиці результатів, що наш метод тренування працює і показує кращі результати

ідентифікації конкретної людини на інакших відео, ніж інакші моделі, які було натреновано на глобальному наборі даних Imagenet.

Однак можна помітити, що після тренування потрібно заново обирати min_cosine_similarity для порівняння схожості векторів, тому що при кожному інакшому тренуванні він може змніюватись.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності. (Закон України «Про охорону праці» від 14.10.1992 №2694-12.)

Керівники підприємств організовують, забезпечують і контролюють трудову діяльність працівників у відповідності з вимогами Закону України «Про охорону праці» і забезпечують безпечні методи праці на кожному робочому місці.

Працівники під час прийняття на роботу і в процесі роботи повинні проходити за рахунок роботодавця інструктаж, навчання з питань охорони праці, з надання першої медичної допомоги потерпілим від нещасних випадків і правил поведінки у разі виникнення аварії.

Працівники, зайняті на роботах з підвищеною небезпекою або там, де є потреба у професійному доборі, повинні щороку проходити за рахунок роботодавця спеціальне навчання і перевірку знань відповідних нормативноправових актів з охорони праці.

Посадові особи, діяльність яких пов'язана з організацією безпечного ведення робіт, під час прийняття на роботу і періодично, один раз на три роки, проходять навчання, а також перевірку знань з питань охорони праці.

Порядок проведення навчання та перевірки знань посадових осіб з питань охорони праці визначається типовим положенням, що затверджується спеціально уповноваженим центральним органом виконавчої влади з нагляду за охороною праці.

Не допускаються до роботи працівники, у тому числі посадові особи, які не пройшли навчання, інструктаж і перевірку знань з охорони праці. У разі виявлення у працівників, у тому числі посадових осіб, незадовільних знань з питань охорони праці, вони повинні у місячний строк пройти повторне навчання і перевірку знань.

Відповідальність за організацію, здійснення навчання, перевірку знань працівників і проведення інструктажів з питань охорони праці покладається на керівника підприємства.

4.1.1 Обов'язки роботодавця

Організація охорони праці на підприємстві покладається на роботодавця. Завдання роботодавця також полягає у забезпеченні дотримання прав робітників, передбачених у нормативних та регуляторних актах з охорони праці.

Для створення безпечних і здорових умов праці роботодавець виконує, зокрема, такі функції:

- формує відповідні відділи і призначає уповноважених осіб для нагляду за дотриманням вимог охорони праці, затверджує внутрішні правила, технологічні карти та стандарти;
- затверджує колективний договір та вживає комплексні заходи для підтримання і підвищення рівня охорони праці;
- розробляє програму оптимізації виробництва, впроваджує новітні технології та наукові досягнення;
- відповідає за належний стан промислових будівель, приміщень, виробничого обладнання та машин;
- вживає невідкладних заходів для допомоги постраждалим, організовує виплату компенсації таким особам;
- ініціює проведення неупередженого та об'єктивного розслідування нещасних випадків, вивчає причини, що призвели до аварії та

затверджує перелік профілактичних заходів, спрямованих на усунення ризиків виникнення аналогічних причин в подальшому;

- несе персональну відповідальність за рівень охорони праці і порушення іншими особами її вимог;
- здійснює нагляд за додержанням робітниками технологічних процесів, установлених правил поведінки та режиму роботи.

Крім того, роботодавець зобов’язаний за свої кошти забезпечити фінансування та організувати проведення попереднього (під час прийняття на роботу) і періодичних (протягом трудової діяльності) медичних оглядів працівників, зайнятих на важких роботах, роботах із шкідливими чи небезпечними умовами праці або таких, де є потреба у професійному доборі, щорічного обов'язкового медичного огляду осіб віком до 21 року.

4.1.2 Обов’язки працівників

Статтею 14 ЗУ «Про охорону праці» передбачено такі обов’язки працівника щодо додержання вимог нормативно-правових актів з охорони праці:

- здійснює нагляд за додержанням робітниками технологічних процесів, установлених правил поведінки та режиму роботи.
- дбати про власну безпеку, а також про безпеку сторонніх людей при виконанні робіт чи під час перебування на території підприємства;
- користуватися засобами колективного та засобами індивідуального захисту;
- знати і виконувати вимоги нормативно-правових актів з охорони праці, правила поводження з машинами, механізмами, устаткуванням та іншими засобами виробництва;
- проходити періодичні медичні огляди, навчальні курси, інструктажі, атестацію знань з безпеки праці.

Також працівники під час прийняття на роботу і в процесі роботи повинні проходити за рахунок роботодавця інструктаж, навчання з питань охорони праці, з надання домедичної допомоги потерпілим від нещасних випадків і правил поведінки у разі виникнення аварії.

Працівник несе безпосередню відповідальність за порушення зазначених вимог.

4.2 Вплив виробничого середовища на здоров'я та працездатність користувачів комп'ютерів

Трудова діяльність користувачів комп'ютерів (ВДТ) відбувається у певному виробничому середовищі, яке впливає на їх функціональний стан. Найбільш значимі — фізичні фактори виробничого середовища, до яких належать електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ціла низка світлотехнічних показників. Вплив хімічних та, особливо, біологічних факторів виробничого середовища на користувачів комп'ютерів — значно менший.

Трудовий процес суттєво впливає на психофізіологічні можливості користувачів комп'ютерів, оскільки їх діяльність характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями.

Професійні якості та виробничий досвід, які визначають внутрішні засоби діяльності, обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях

Зовнішні засоби діяльності, які в основному визначаються ергономічними показниками щодо організації робочого місця, форми та параметрів його елементів, просторового розташування основного і допоміжного устаткування, можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів.

У професійних операторів частіше зустрічаються порушення органів зору, опорно-рухового апарату, центральної нервової, серцево-судинної, імунної та статевої систем, захворювання шкіри. Зафіксована значна кількість скарг операторського персоналу на загальне недомагання, передчасне стомлювання, головний біль, порушення функцій органів зору, які здійснювали несприятливий психофізіологічний вплив на самопочуття та працездатність операторів.

Сучасна професія користувача ВДТ належить до розумової праці, яка характеризується: високою напруженістю зорових функцій; одноманітною позою; великою кількістю стереотипних висококоординованих рухів, що виконуються лише м'язами кистей рук на фоні малої загальної рухової активності; значним нервовоемоційним компонентом, особливо в умовах дефіциту часу; роботою з великими масивами інформації, що викликає активізацію уваги та інших вищих психічних функцій. Крім того, при роботі з дисплеями на електронно-променевих трубках виникає вплив на користувача цілої низки факторів фізичної природи — електростатичні поля, радіочастотне та рентгенівське випромінювання тощо.

Діяльність професіоналів можна поділити на три групи:

1) Діяльність, яка пов'язана з виконанням нескладних багаторазово повторюваних операцій, що не вимагають великого розумового напруження. Наприклад, робота операторів комп'ютерного набору, працівників довідкових служб.

2) Діяльність, яка пов'язана із здійсненням логічних операцій, що постійно повторюються. Це робота інженера-економіста, інженера-проектувальника, оператора автоматизованого виробництва.

3) Діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму. Наприклад, робота інженера-програміста, диспетчерів руху залізничного транспорту, аеропортів тощо.

У користувачів, які інтенсивно використовують комп'ютер в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість тощо). Серед користувачів ВДТ в США і Європі значного поширення набуло специфічне захворювання, яке отримало назву синдром комп'ютерного стресу (СКС). СКС супроводжується головним болем, запаленням очей, алергією, роздратованістю, млявістю і депресією. Інформаційне перевантаження користувачів ВДТ супроводжується низкою специфічних захворювань, які називають інформаційними. Першим симптомом їх є головний біль. Дослідження, проведені в США, Німеччині, Швейцарії та інших країнах, показали, що робота з обслуговування ВДТ супроводжується підвищеним напруженням зору, інтенсивністю і монотонністю праці, збільшенням статичних навантажень, нервово-психічним напруженням, впливом різного виду випромінювань та ін. Внаслідок цього серед операторів ВДТ, як зазначають фахівці Всесвітньої організації охорони здоров'я, частіше, ніж в інших групах працюючих, трапляються такі професійні захворювання, як передчасна стомлюваність, погіршення зору, м'язові і головні болі, психічні й нервові розлади, хвороби серцево-судинної системи, онкологічні захворювання та ін. Вважається, що стан організму операторів ВДТ визначається комплексним впливом факторів трудового процесу і середовища, значення яких є неоднаковим. На операторів з малим стажем роботи на ВДТ домінуючий вплив чинять фактори середовища, а на операторів зі стажем понад 5 років - фактори трудового процесу.

ВИСНОВОК

Якщо потужність комп’ютерів використовувати розумно та своєчасно, можна запобігти злочинам та іншим незаконним діям, а правопорушників можна легко відстежити. Незважаючи на те, що розробка таких моделей все ще триває, проте їх вдосконалення вражає з кожним роком, а застосування можна знайти в багатьох сферах.

Повторна ідентифікація особи обговорюється та розробляється вже дуже давно. Саме зараз, коли розрахункові потужності ростуть, набуває великого інтересу в сучасній науковій спільноті.

По-перше, представлено коротку історію повторної ідентифікації особи та описано її подібності та відмінності від класифікації зображень та пошуку екземплярів. Потім розглядаються існуючі методи на основі зображень і відео, які класифікуються на системи створені вручну, і системи з глибоким вивченням.

Повторна ідентифікація особи, розташована між класифікацією зображень і пошуком екземплярів, далека від того, щоб стати точною та ефективною програмою. В цій роботі більше уваги приділяється недостатньо розвиненим, але критичним майбутнім можливостям, таким як наскрізні системи повторної ідентифікації, які об’єднують виявлення та відстеження пішоходів, а також повторну ідентифікацію людей у великих об’ємах даних, які, на мою думку, є необхідними кроками для практичних систем.

Також було розглянуто деякі важливі відкриті питання. Вони включають вирішення проблеми обсягу даних, методи повторного ранжування ідентифікатора та відкриті системи повторного ідентифікації. Загалом, інтеграція вивчення дискримінаційних ознак, оптимізація детектора/відстеження та ефективні структури даних призведуть до успішної системи повторної ідентифікації людини.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ross Girshick. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015. 1, 2

2. John A Meacham. Wisdom and the context of knowledge: Knowing that one doesn't know. On the development of developmental psychology, 8:111–134, 1983. 1

3. Susan Engel. Children's need to know: Curiosity in schools. Harvard educational review, 81(4):625–645, 2011. 1

4. Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. IEEE transactions on pattern analysis and machine intelligence, 35(7):1757–1772, 2012. 1

5. Abhijit Bendale and Terrance Boult. Towards open world recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1893–1902, 2015. 1, 2

6. Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sunderhauf. Dropout sampling for robust object detection in ¨ open-set conditions. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–7. IEEE, 2018. 2, 6, 8

7. Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 7(35):1757–1772, 2013. 2

8. Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1563–1572, 2016. 2

9. Zongyuan Ge, Sergey Demyanov, Zetao Chen, and Rahil Garnavi. Generative openmax for multi-class open set classification. In British Machine Vision Conference 2017. British Machine Vision Association and Society for Pattern Recognition, 2017. 2

10. Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2537–2546, 2019. 2

11. Federico Pernici, Federico Bartoli, Matteo Bruni, and Alberto Del Bimbo. Memory based online learning of deep representations from video streams. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2324–2334, 2018. 2

12. Hu Xu, Bing Liu, Lei Shu, and P Yu. Open-world learning and application to product classification. In The World Wide Web Conference, pages 3413–3419, 2019. 2

13. Akshay Dhamija, Manuel Gunther, Jonathan Ventura, and Terrance Boult. The overlooked elephant of object detection: Open set. In The IEEE Winter Conference on Applications of Computer Vision, pages 1021–1030, 2020. 2, 3, 6

14. David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Sunderhauf. Probabilistic object detection: Definition and evaluation. In The IEEE Winter Conference on Applications of Computer Vision, pages 1031–1040, 2020. 2

15. Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning, pages 1050–1059, 2016. 2

16. Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In ECCV, pages 480–496, 2018. 1, 2

17. Guanshuo Wang, Yufeng Yuan, Xiong Chen, Jiwei Li, and Xi Zhou. Learning discriminative features with multiple granularities for person re-identification. In ACMMM, pages 274–282, 2018. 1, 2, 3, 8

18. Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In CVPRW, pages 0–0, 2019. 1, 2, 3

19. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, and Xiaohua et al. Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR, 2021. 2, 3

20. Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve J ´ egou. Training ´ data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877, 2020. 2, 3

21. Hao Luo, Wei Jiang, Xuan Zhang, Xing Fan, Jingjing Qian, and Chi Zhang. Alignedreid++: Dynamically matching local information for person re-identification. Pattern Recognition, 94:53–61, 2019. 2

22. Dechao Meng, Liang Li, Xuejing Liu, Yadong Li, Shijie Yang, Zheng-Jun Zha, Xingyu Gao, Shuhui Wang, and Qingming Huang. Parsing-based view-aware embedding network for vehicle re-identification. In CVPR, pages 7103– 7112, 2020. 2, 3, 8

23. Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person reidentification: A survey and outlook. IEEE TPAMI, 2021. 2

24. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016. 2

25. Zhedong Zheng, Liang Zheng, and Yi Yang. A discriminatively learned cnn embedding for person reidentification. ACM TOMM, 14(1):13, 2018. 2

26. Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, and Shuicheng Yan. End-to-end comparative attention networks for person re-identification. IEEE TIP, 26(7):3492–3506, 2017. 2

27. Jingjing Qian, Wei Jiang, Hao Luo, and Hongyan Yu. Stripebased and attribute-aware network: A two-branch deep model for vehicle re-identification. Measurement Science and Technology, 31(9):095401, 2020. 3, 8

28. Jiaxu Miao, Yu Wu, Ping Liu, Yuhang Ding, and Yi Yang. Pose-guided feature alignment for occluded person re-identification. In ICCV, pages 542–551, 2019. 2, 3, 5, 8

29. Zhihui Zhu, Xinyang Jiang, Feng Zheng, Xiaowei Guo, Feiyue Huang, Xing Sun, and Weishi Zheng. Aware loss with angular regularization for person re-identification. In AAAI, volume 34, pages 13114–13121, 2020. 3

30. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, pages 6000–6010, 2017. 3

31. Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. arXiv preprint arXiv:2101.01169, 2021. 3

32. Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. CVPR, 2021. 3

33. Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2016;39(6):1137–49.

34. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. Ssd: single shot multibox detector. In: European conference on computer vision. Cham: Springer; 2016, p. 21–37.

35. Alganci U, Soydas M, Sertel E. Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. Remote Sensing. 2020;12(3):458.

36. Reza Z. N. (2019). Real-time automated weld quality analysis from ultrasonic B-scan using deep learning (Doctoral dissertation, University of Windsor (Canada)).

37. Shen X, Wu Y. A unified approach to salient object detection via low rank matrix recovery. In: 2012 IEEE conference on computer vision and pattern recognition. IEEE; 2012, p. 853–60.

38. Madden C., Cheng E. D., Piccardi M., 2007. Tracking people across disjoint camera views by an illumination tolerant appearance representation. Journal of Machine Vision and Applications 18(3), pp. 233–247.

39. Colombo A., Orwell J., and Velastin S., 2008. Colour constan-cy techniques for re-recognition of pedestrians from mul-tiple surveillance cameras. In Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications, 2008, Marseille, France

40. Truong Congl D.-N., Khoudour L., Achard C., Meurie C., Lezoray O., 2010. People re-identification by spectral classification of silhouettes. Signal Processing, vol. 90, no. 8, pp 2362-2374.

41. Javed O., Shafique K., Shah M., 2005. Appearance modeling for tracking in multiple non-overlapping cameras. In IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 26-33.

42. Jeong K., Jaynes C., 2008. Object matching in disjoint cameras using a colour transfer approach. Journal of Machine Vision and Applications, 19(5), pp. 88–96

43. Bäuml M., Bernardin K., Fischer M.,. Ekenel H. K, 2010. Multi-pose face recognition for person retrieval in camera networks. In IEEE 7 th Int. Conf. on Advanced Video and Signal Based Surveillance, pp. 441-447.

44. Dantcheva A.,. Dugelay J. L, 2011. Frontal-to-side face reidentification based on hair, skin and clothes patches. In IEEE Int. Conf. on Advanced Video and SignalBased Surveillance, pp. 309-313

45. Roy A., Sural S., Mukherjee J., 2012. A hierarchical method combining gait and phase of motion with spatiotemporal model for person re-identification. In Pattern Recognition Letters, 33(14), pp. 1891-1901.

46. Cai Q. and Aggarwal J.K., 1999. Tracking Human Motion Using a Distributed-Camera System. In IEEE Trans. On PAMI., 21(12), pp. 1241-1247

47. Mittal A., Davis, L. 2003. M2 tracker: a multi-view approach to segmenting and tracking people in a cluttered scene. Int. J. Comput. Vis. 51(3), pp.189–203

48. Satta R., Fumera G., Roli F., 2012. Fast person reidentification based on dissimilarity representations. In Pattern Recognition Letters, 33(14), pp. 1838-1848.

49. Visual Studio Code on Windows [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://code.visualstudio.com/docs/setup/windows.

50. How to Install WSL 2 on Windows 10 (Updated) [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.omgubuntu.co.uk/how-to-install-wsl2-on-windows-10.

51. Simple Python Version Management: pyenv [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/pyenv/pyenv.

52. Torchreid [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/KaiyangZhou/deep-person-reid.

53. Official YOLOv7 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/WongKinYiu/yolov7.

Додаток А – Лістинг файлу config.yaml

```
paths:
  model_training:
    output_folder: model_training/
  data:
    input_videos: videos_data
  process_clusters:
    input_folder: train_data_master/clusters

generate_detections:
  detector:
    weights: yolov7/weights/yolov7-e6e.pt
    conf_thres: 0.8
    device: cuda:0
    project: processed_videos
    name: dataset
    save_img: False
    clear_workspace: True
  frame_freq: 20
  video_index: 2

  feature_extractor:
    model_name: osnet_x1_0
    model_path: deep-person-
reid/weights/osnet_x1_0_market_256x128_amsgrad_ep150_stp60_lr0.0
015_b64_fb10_softmax_labelsmooth_flip.pth
    device: cpu
  cosine_similarity_sort_by: 0.65
  extraction_batch_size: 16

train:
  datamanager_build:
    root: ""
    sources: train_data_master
    targets: train_data_master
    height: 256
    width: 128
    batch_size_train: 8
    batch_size_test: 8
    use_gpu: true

  model_build:
    name: osnet_x0_75
    use_gpu: true
    loss: softmax
    pretrained: true

  optimizer_build:
    optim: adam
    lr: 0.0002
```

```
  lr_scheduler_build:
    lr_scheduler: single_step
    stepsize: 10

  engine_build:
    label_smooth: true

  engine_run:
    max_epoch: 300
    eval_freq: 25
    print_freq: 5
    test_only: false

evaluate:
  feature_extractor:
    model_name: osnet_x0_75
    model_path: deep-person-
reid/weights/osnet_x1_0_market_256x128_amsgrad_ep150_stp60_lr0.0
015_b64_fb10_softmax_labelsmooth_flip.pth
    device: cuda

  min_cosine_similarity: 0.9971
  db_data_path: mongo_db_data/reid/wanted_people
  ground_truth: test_data_master
  save_metrics_to: metrics_results.txt
```

Додаток Б – Лістинг файлу generate_detections.py

```
import os
import sys
from collections import namedtuple
from glob import glob
from pathlib import Path
from shutil import copy2

import numpy as np
import yaml
from sklearn.metrics.pairwise import cosine_similarity
from torchreid.utils import FeatureExtractor

from utils_reid import create_folders

if str(Path(os.getcwd(), "yolov7")) not in sys.path:
    sys.path.append(str(Path(os.getcwd(), "yolov7")))

from yolov7_detector import YOLOv7_Detector

with open("config.yaml", "r") as stream:
    try:
        config = yaml.safe_load(stream)
    except yaml.YAMLError as exc:
        print(exc)


video_info = namedtuple("video", ["video_path",
"video_size_in_bytes"])

DATA_PATH = Path(config["paths"]["data"]["input_videos"])
VIDEOS_DATA = [
    video_info(video_path=video_path,
video_size_in_bytes=os.stat(video_path).st_size)
    for video_path in glob(f"{str(DATA_PATH)}/*.mp4")
]

VIDEOS_DATA = sorted(VIDEOS_DATA, key=lambda video:
video.video_size_in_bytes)
print("Existing videos list:")
for video in VIDEOS_DATA:
    print(video)

print("Model defining.")
model_params = dict(
    source=VIDEOS_DATA[config["generate_detections"]["video_inde
x"]].video_path,
    **config["generate_detections"]["detector"],
)

print(f"Model params: {model_params}")
```

```
detector = YOLOv7_Detector(**model_params)

print("Extracting person objects from video.")
detector.run_inference_from_given_source(
    frame_processing_freq=config["generate_detections"]["frame_f
req"]
)

print("Sort detections with default ReID")

# create folders for sorting
images_folder = Path(
    config["generate_detections"]["detector"]["project"],
    config["generate_detections"]["detector"]["name"],
    "cropped",
)

images_paths = glob(f"{str(images_folder)}/*.jpg")

folder_names_to_create = sorted(
    list(set([Path(image_path).name.split("_")[0] for image_path
in images_paths]))
)
folder_names_to_create.append("unrecognized")

folders_full_path = [
    Path(
        config["generate_detections"]["detector"]["project"],
        config["generate_detections"]["detector"]["name"],
        "sorted_detections",
        folder_name,
    )
    for folder_name in folder_names_to_create
]

create_folders(path_or_list=folders_full_path, exist_ok=True)

# extract features for each person
extractor =
FeatureExtractor(**config["generate_detections"]["feature_extrac
tor"])

print("Extracting features.")


def batch(iterable, n=1):
    l = len(iterable)
    for ndx in range(0, l, n):
        yield iterable[ndx : min(ndx + n, l)]
```

```
features = [feature for images in batch(images_paths) for
feature in extractor(images)]

clusters = dict(
    {
        f"{str(cluster)}": []
        for cluster in folders_full_path
        if "unrecognized" not in str(cluster)
    }
)

print("Sorting process.")
# sort person detections
for image_path, feature in zip(images_paths, features):
    feature_numpy = feature.cpu().numpy()
    cluster_found = False

    for cluster_path in folders_full_path[:-1]:

        cluster_path = str(cluster_path)
        cluster_features = clusters[cluster_path]

        # cluster is empty
        if not cluster_features:
            clusters[cluster_path].append(feature_numpy)
            copy2(src=image_path, dst=cluster_path)
            cluster_found = True
            break

        # if not check embedding similarity to other clusters
        else:
            similarity = np.mean(
                cosine_similarity(clusters[cluster_path],
[feature_numpy])
            )
            if similarity > 0.75:
                clusters[cluster_path].append(feature_numpy)
                copy2(src=image_path, dst=cluster_path)
                cluster_found = True
                break

    # move to "unrecognized"
    if not cluster_found:
        copy2(src=image_path, dst=str(folders_full_path[-1]))

print("Process done.")
```

Додаток В – Лістинг файлу yolov7_detector.py

```
import time
from pathlib import Path

import cv2
import numpy as np
import torch
import torch.backends.cudnn as cudnn
from numpy import random
from shutil import rmtree

from yolov7.models.experimental import attempt_load
from yolov7.utils.datasets import LoadImages, LoadStreams,
letterbox
from yolov7.utils.general import (
    check_imshow,
    increment_path,
    non_max_suppression,
    scale_coords,
    xyxy2xywh,
)
from yolov7.utils.plots import plot_one_box
from yolov7.utils.torch_utils import (
    select_device,
    time_synchronized,
)

class YOLOv7_Detector:
    def __init__(
        self,
        source: str = "",
        weights: str = "weights/yolov7-e6e.pt",
        device: str = "cpu",
        img_size: int = 1280,
        augment: bool = False,
        conf_thres: float = 0.45,
        iou_thres: float = 0.5,
        agnostic_nms: bool = False,
        project: str = "runs/detect",
        name: str = "exp",
        exist_ok: bool = False,
        save_txt: bool = True,
        save_conf: bool = False,
        save_img: bool = True,
        save_crop: bool = True,
        view_img: bool = False,
        clear_workspace: bool = False,
    ) -> None:

        # general params
        self._project = project
```

```
        if clear_workspace:
            print(f"Cleaned workspace: {self._project}")
            rmtree(self._project, ignore_errors=True)

        self._name = name
        self._exist_ok = exist_ok
        self._save_txt = save_txt
        self._save_conf = save_conf
        self._save_img = save_img
        self._save_crop = save_crop
        self._view_img = view_img

        # model params
        self._img_size = img_size
        self._old_img_w = self._old_img_h = self._img_size
        self._old_img_b = 1
        self._device = select_device(device)
        self._half = self._device.type != "cpu"  # half
precision only supported on CUDA

        # model loading
        self._model = self._load_model(weights)
        self._stride = int(self._model.stride.max())  # model
stride
        if self._half:
            self._model.half()  # to FP16

        # dataset loading params
        self._source = source
        self._webcam = (
            self._source.isnumeric()
            or self._source.endswith(".txt")
            or self._source.lower().startswith(
                ("rtsp://", "rtmp://", "http://", "https://")
            )
        )

        # Directories
        self._save_dir = Path(
            increment_path(Path(self._project) / self._name,
exist_ok=self._exist_ok)
        )  # increment run
        self._save_txt = save_txt
        (self._save_dir / "labels" if self._save_txt else
self._save_dir).mkdir(
            parents=True, exist_ok=True
        )  # make dir

        if self._save_crop:
            self._save_crop_folder = self._save_dir / "cropped"
            self._save_crop_folder.mkdir(parents=True,
exist_ok=True)
```

```
        # Get names and colors
        self._names = (
            self._model.module.names
            if hasattr(self._model, "module")
            else self._model.names
        )
        self._colors = [[random.randint(0, 255) for _ in
range(3)] for _ in self._names]

        # detection params
        self._augment = augment
        self._conf_thres = conf_thres
        self._iou_thres = iou_thres
        self._agnostic_nms = agnostic_nms
        self._classes = [0]

    def detect(self, bgr_frame: np.ndarray):

        # do preprocessing
        img = self._image_preprocess(bgr_frame=bgr_frame)

        # Inference
        with torch.no_grad():  # Calculating gradients would
cause a GPU memory leak
            pred = self._model(img, augment=self._augment)[0]

        # Apply NMS
        pred = non_max_suppression(
            prediction=pred,
            conf_thres=self._conf_thres,
            iou_thres=self._iou_thres,
            classes=self._classes,
            agnostic=self._agnostic_nms,
        )[0]

        # transform predictions
        pred[:, :4] = scale_coords(img.shape[2:], pred[:, :4],
bgr_frame.shape).round()

        return pred.numpy()

    def run_inference_from_given_source(self,
frame_processing_freq: int = 25):
        vid_path, vid_writer = None, None
        dataset = self._load_source()

        t0 = time.time()
        for frame_num, (path, img, im0s, vid_cap) in
enumerate(dataset):

            if frame_num % frame_processing_freq != 0:
```

```
                continue

            img = torch.from_numpy(img).to(self._device)
            img = img.half() if self._half else img.float()  #
uint8 to fp16/32
            img /= 255.0  # 0 - 255 to 0.0 - 1.0
            if img.ndimension() == 3:
                img = img.unsqueeze(0)

            # Warmup
            if self._device.type != "cpu" and (
                self._old_img_b != img.shape[0]
                or self._old_img_h != img.shape[2]
                or self._old_img_w != img.shape[3]
            ):
                self._old_img_b = img.shape[0]
                self._old_img_h = img.shape[2]
                self._old_img_w = img.shape[3]
                for i in range(3):
                    self._model(img, augment=self._augment)[0]

            # Inference
            t1 = time_synchronized()
            with torch.no_grad():  # Calculating gradients would
cause a GPU memory leak
                pred = self._model(img,
augment=self._augment)[0]
            t2 = time_synchronized()

            # Apply NMS
            pred = non_max_suppression(
                prediction=pred,
                conf_thres=self._conf_thres,
                iou_thres=self._iou_thres,
                classes=self._classes,
                agnostic=self._agnostic_nms,
            )
            t3 = time_synchronized()

            # Process detections
            for i, det in enumerate(pred):  # detections per
image
                if self._webcam:  # batch_size >= 1
                    p, s, im0, frame = (
                        path[i],
                        "%g: " % i,
                        im0s[i].copy(),
                        dataset.count,
                    )
                else:
                    p, s, im0, frame = (
                        path,
```

```
                        "",
                        im0s,
                        getattr(dataset, "frame", 0),
                    )

                p = Path(p)  # to Path
                save_path = str(self._save_dir / p.name)  #
img.jpg
                txt_path = str(self._save_dir / "labels" /
p.stem) + (
                    "" if dataset.mode == "image" else
f"_{frame}"
                )  # img.txt
                gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  #
normalization gain whwh
                if len(det):
                    # Rescale boxes from img_size to im0 size
                    det[:, :4] = scale_coords(
                        img.shape[2:], det[:, :4], im0.shape
                    ).round()

                    # Print results
                    for c in det[:, -1].unique():
                        n = (det[:, -1] == c).sum()  #
detections per class
                        s += f"{n} {self._names[int(c)]}{'s' *
(n > 1)}, "  # add to string

                    # Write results
                    for detection, (*xyxy, conf, cls) in
enumerate(reversed(det)):

                        if self._save_crop:
                            xmin, ymin, xmax, ymax = np.array(
                                [
                                    coordinate.cpu().detach().nu
mpy()
                                    for coordinate in xyxy
                                ],
                                dtype=np.int32,
                            )
                            person = im0s[ymin:ymax,
xmin:xmax].copy()
                            cv2.imwrite(
                                f"{self._save_crop_folder}/{dete
ction}_{frame_num}.jpg",
                                person,
                            )

                        if self._save_txt:  # Write to file
                            xywh = (
```

```
                                    (xyxy2xywh(torch.tensor(xyxy).vi
ew(1, 4)) / gn)
                                    .view(-1)
                                    .tolist()
                                )  # normalized xywh
                                line = (
                                    (cls, *xywh, conf) if
self._save_conf else (cls, *xywh)
                                )  # label format
                                with open(txt_path + ".txt", "a") as
f:
                                    f.write(("%g " *
len(line)).rstrip() % line + "\n")

                        if self._save_img:  # Add bbox to image
                            label = f"{self._names[int(cls)]}
{conf:.2f}"
                            plot_one_box(
                                xyxy,
                                im0,
                                label=label,
                                color=self._colors[int(cls)],
                                line_thickness=1,
                            )

                # Print time (inference + NMS)
                print(
                    f"{s}Done. ({(1E3 * (t2 - t1)):.1f}ms)
Inference, ({(1E3 * (t3 - t2)):.1f}ms) NMS"
                )

                # Stream results
                if self._view_img:
                    cv2.imshow(str(p), im0)
                    cv2.waitKey(1)  # 1 millisecond

                # Save results (image with detections)
                if self._save_img:
                    if dataset.mode == "image":
                        cv2.imwrite(save_path, im0)
                        print(f" The image with the result is
saved in: {save_path}")
                    else:  # 'video' or 'stream'
                        if vid_path != save_path:  # new video
                            vid_path = save_path
                            if isinstance(vid_writer,
cv2.VideoWriter):
                                vid_writer.release()  # release
previous video writer
                            if vid_cap:  # video
                                fps =
vid_cap.get(cv2.CAP_PROP_FPS)
```

```
                                    w =
int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                                    h =
int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                                else:  # stream
                                    fps, w, h = 30, im0.shape[1],
im0.shape[0]
                                    save_path += ".mp4"
                                vid_writer = cv2.VideoWriter(
                                    save_path,
cv2.VideoWriter_fourcc(*"mp4v"), fps, (w, h)
                                )
                            vid_writer.write(im0)

        if self._save_txt or self._save_img:
            s = (
                f"\n{len(list(self._save_dir.glob('labels/*.txt'
)))} labels saved to {self._save_dir / 'labels'}"
                if self._save_txt
                else ""
            )
            print(f"Results saved to {self._save_dir}{s}")

        print(f"Done. ({time.time() - t0:.3f}s)")

    def _load_model(self, weights: str):
        model = attempt_load(weights,
map_location=self._device)  # load FP32 model
        return model

    def _load_source(self):
        if self._webcam:
            self._view_img = check_imshow()
            cudnn.benchmark = True  # set True to speed up
constant image size inference
            dataset = LoadStreams(
                self._source, img_size=self._img_size,
stride=self._stride
            )
        else:
            dataset = LoadImages(
                self._source, img_size=self._img_size,
stride=self._stride
            )

        return dataset

    def _image_preprocess(self, bgr_frame: np.ndarray) ->
np.ndarray:
        # Padded resize
        img = letterbox(img=bgr_frame, new_shape=self._img_size,
stride=self._stride)[0]
```

```
        # Convert
        img = img[:, :, ::-1].transpose(2, 0, 1)  # BGR to RGB,
to 3x416x416
        img = np.ascontiguousarray(img)

        img = torch.from_numpy(img).to(self._device)
        img = img.half() if self._half else img.float()  # uint8
to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        return img
```

Додаток Г – Лістинг файлу process_separated_cluster.py

```
import os
from pathlib import Path
from shutil import copy2

import yaml
from tqdm import tqdm

from utils_reid import create_folders

with open("config.yaml", "r") as stream:
    try:
        config = yaml.safe_load(stream)
    except yaml.YAMLError as exc:
        print(exc)

CLUSTERS_PATH =
Path(config["paths"]["process_clusters"]["input_folder"])
CLUSTERS_ID_SET = sorted(os.listdir(str(CLUSTERS_PATH)))

# fix person id according to cluster id
clusters = tqdm(CLUSTERS_ID_SET)
for cluster_id in clusters:
    clusters.set_description(f"Processing cluster:
{cluster_id}")

    cluster_images = tqdm((CLUSTERS_PATH /
cluster_id).glob(pattern="*.jpg"))
    for person_image in cluster_images:
        cluster_images.set_description(f"Processing image:
{person_image.name}")

        [person_id, frame_id], file_ext = (
            person_image.stem.split("_"),
            person_image.suffix,
        )

        if person_id != cluster_id:
            new_person_image =
f"{cluster_id}_{frame_id}{file_ext}"
            person_image.rename(person_image.with_name(new_perso
n_image))

# move cluster's data to [train] and [query] folders
dataset_folders = {
    "train": CLUSTERS_PATH.parent / "train",
    "query": CLUSTERS_PATH.parent / "query",
}

create_folders(list(dataset_folders.values()))
```

```
print("Moving images from clusters to [train] or [query] folders
dataset")
train_images = tqdm((CLUSTERS_PATH /
CLUSTERS_ID_SET[0]).glob(pattern="*.jpg"))
for train_image in train_images:
    train_images.set_description(f"Processing train image:
{train_image}")
    dst_path = dataset_folders["train"] / train_image.name
    copy2(src=train_image, dst=dst_path)

query_images = tqdm(
    sorted(
        Path("train_data_master/clusters/").glob(
            pattern=f"[{','.join(CLUSTERS_ID_SET[1:])}]/*.jpg"
        )
    )
)
for query_image in query_images:
    query_images.set_description(f"Processing query image:
{query_image}")
    dst_path = dataset_folders["query"] / query_image.name
    copy2(src=query_image, dst=dst_path)

print("Generated [train] and [query] dataset parts")
```

Додаток Д – Лістинг файлу train_dataset.py

```
from __future__ import absolute_import, division, print_function

import os
import os.path as osp
from glob import glob
from typing import Literal

import numpy as np
from torchreid.data import ImageDataset

class TrainReidDataset(ImageDataset):
    dataset_dir = "train_data_master"

    def __init__(self, root="", **kwargs):
        self.root = osp.abspath(osp.expanduser(root))
        self.dataset_dir = osp.join(self.root, self.dataset_dir)

        # All you need to do here is to generate three lists,
        # which are train, query and gallery.
        # Each list contains tuples of (img_path, pid, camid),
        # where
        # - img_path (str): absolute path to an image.
        # - pid (int): person ID, e.g. 0, 1.
        # - camid (int): camera ID, e.g. 0, 1.
        # Note that
        # - pid and camid should be 0-based.
        # - query and gallery should share the same pid scope
(e.g.
        #   pid=0 in query refers to the same person as pid=0 in
gallery).
        # - train, query and gallery share the same camid scope
(e.g.
        #   camid=0 in train refers to the same camera as
camid=0
        #   in query/gallery).
        # - query and gallery should have different from each
other camid

        # folder - [train]
        # train = [
        #           (
        #               train_data_master/train/0_0.jpg,
        #               0,
        #               18376454934
        #           ),
        #           (
        #               train_data_master/train/0_20.jpg,
        #               0,
        #               18376454934
        #           )
```

```
        # ]
        # query = [
        #             (
        #                 train_data_master/query/1_0.jpg,
        #                 1,
        #                 1
        #             ),
        #             (
        #                 train_data_master/query/1_20.jpg,
        #                 1,
        #                 1
        #             )
        # ]

        # define train, query and gallery data lists
        train = self.prepare_data("train")
        query = self.prepare_data("query")

        gallery = train + query

        # change camid
        gallery = [(image_path, person_id, 0) for image_path,
person_id, _ in gallery]

        super(TrainReidDataset, self).__init__(train, query,
gallery, **kwargs)

    def prepare_data(self, data_for: Literal["train", "query"]):
        image_paths = glob(os.path.join(self.dataset_dir,
data_for, "*.jpg"))
        result = []
        for image_path in image_paths:
            person_info = np.array(
                os.path.basename(image_path).split(".")[0].split
("_"), dtype=np.int64
            )

            person_id, cam_id = person_info[0], 1

            result.append((image_path, person_id, cam_id))

        return result
```

Додаток Е – Лістинг файлу train.py

```
from __future__ import absolute_import, division, print_function

import os

import torchreid

from train_dataset import TrainReidDataset
import yaml

with open("config.yaml", "r") as stream:
    try:
        config = yaml.safe_load(stream)
    except yaml.YAMLError as exc:
        print(exc)

# add custom dataset to the image data manager
torchreid.data.register_image_dataset("train_data_master",
TrainReidDataset)

# Load data manager
datamanager = torchreid.data.ImageDataManager(
    **config["train"]["datamanager_build"]
)

# Build model, optimizer and lr_scheduler
config["train"]["model_build"]["num_classes"] =
datamanager.num_train_pids
model = torchreid.models.build_model(
    **config["train"]["model_build"]
)

model = model.cuda()

config["train"]["optimizer_build"]["model"] = model
optimizer = torchreid.optim.build_optimizer(
    **config["train"]["optimizer_build"]
)

config["train"]["lr_scheduler_build"]["optimizer"] = optimizer
scheduler = torchreid.optim.build_lr_scheduler(
    **config["train"]["lr_scheduler_build"]
)

# Build engine
config["train"]["engine_build"]["datamanager"] = datamanager
config["train"]["engine_build"]["model"] = model
config["train"]["engine_build"]["optimizer"] = optimizer
config["train"]["engine_build"]["scheduler"] = scheduler

engine = torchreid.engine.ImageSoftmaxEngine(
```

```
    **config["train"]["engine_build"]
)

# Run training and test
config["train"]["engine_run"]["save_dir"] =
os.path.join(config["paths"]["model_training"]["output_folder"],
config["train"]["model_build"]["name"])
engine.run(
    **config["train"]["engine_run"]
)
```

Додаток Є – Лістинг файлу evaluate.py

```
from datetime import datetime
from pathlib import Path

import pandas as pd
import yaml
from torchreid.utils import FeatureExtractor
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

import warnings
warnings.filterwarnings('ignore')

from db_connection import wanted_people_collection

with open("config.yaml", "r") as stream:
    try:
        config = yaml.safe_load(stream)
    except yaml.YAMLError as exc:
        print(exc)

DB_DATA_PATH = Path(config["evaluate"]["db_data_path"])
MIN_COS_SIMILAR = config["evaluate"]["min_cosine_similarity"]
METRICS_SAVE_PATH = config["evaluate"]["save_metrics_to"]

def refresh_db():
    for wanted_person in DB_DATA_PATH.iterdir():
        person_images_list = [
            str(image_path) for image_path in
wanted_person.glob("*.jpg")
        ]

        person_db_data = wanted_people_collection.find_one(
            {"person_name": wanted_person.name}
        )

        if person_db_data:
            images_list_from_db =
person_db_data["person_images"]

            difference_list = list(set(person_images_list) -
set(images_list_from_db))
            if difference_list:
                wanted_people_collection.update_one(
                    {
                        "person_name": wanted_person.name,
                        "push_date": datetime.now(),
                        "person_images": person_images_list,
                    }
                )
```

```
        else:
            wanted_people_collection.insert_one(
                {
                    "person_name": wanted_person.name,
                    "push_date": datetime.now(),
                    "person_images": person_images_list,
                }
            )

refresh_db()

# define models
extractor =
FeatureExtractor(**config["evaluate"]["feature_extractor"])

# get and sort existing people in the MongoDB
features_for_people_from_db = {
    person["person_name"]: [feature.cpu().detach().numpy() for
feature in extractor(person["person_images"])]
    for person in wanted_people_collection.find()
}

# create dataframe to save results
results_df = pd.DataFrame(
    columns=["actual_person", "predicted_person", "pred_res"]
)

results_df = results_df.astype(
    {"actual_person": str, "predicted_person": str}
)

# calculate results
test_data = Path(config["evaluate"]["ground_truth"])
for test_person in test_data.iterdir():

    extracted_person_images = extractor(
        [str(image_path) for image_path in
test_person.glob("*.jpg")]
    )

    actual_person_name = test_person.name

    for extracted_person_feature in extracted_person_images:

        person_similarity = 0
        person_match_name = "unrecognized"

        for (
            person_from_db,
            person_from_db_features,
        ) in features_for_people_from_db.items():
```

```
            similarity = np.mean(
                cosine_similarity(
                    [extracted_person_feature.cpu().detach().num
py()],
                    person_from_db_features,
                )
            )

            if similarity > person_similarity:
                person_match_name = person_from_db
                person_similarity = similarity

        if person_similarity > MIN_COS_SIMILAR:
            # print(f"actual_person - {actual_person_name},
predicted_person - {person_match_name}, pred_res -
{person_similarity}")
            results_df = results_df.append(
                {
                    "actual_person": actual_person_name,
                    "predicted_person": person_match_name,
                    "pred_res": True if actual_person_name ==
person_match_name else False,
                    "similarity": person_similarity,
                },
                ignore_index=True
            )
        else:
            results_df = results_df.append(
                {
                    "actual_person": actual_person_name,
                    "predicted_person": "unrecognized",
                    "pred_res": False if actual_person_name ==
person_match_name else True,
                    "similarity": person_similarity,
                },
                ignore_index=True
            )

results_df.to_csv("preds.csv", index=False)

# metrics calculation
tp_tn = results_df[results_df["actual_person"] ==
results_df["predicted_person"]]
fp_fn = results_df[results_df["actual_person"] !=
results_df["predicted_person"]]

tp_fn_values = tp_tn["pred_res"].value_counts()
fp_tn_values = fp_fn["pred_res"].value_counts()

tp = tp_fn_values[True] if True in tp_fn_values.keys() else 0
fn = tp_fn_values[False] if False in tp_fn_values.keys() else 0
```

```
fp = fp_tn_values[False] if False in fp_tn_values.keys() else 0
tn = fp_tn_values[True] if True in fp_tn_values.keys() else 0

accuracy = (tp + tn) / (tp + tn + fp + fn)
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1_score = 2 * tp / ((2 * tp) + fp + fn)

metrics = {
    "accuracy": accuracy,
    "precision": precision,
    "recall": recall,
    "f1_score": f1_score,
}

with open("metrics_result.txt", mode="w", encoding="utf-8") as
f:
    for metric_name, metric_value in metrics.items():
        f.write(f"{metric_name} {round(metric_value, 2)}\n")
```

Додаток Ж – Лістинг файлу utils_reid.py

```
import os
from pathlib import Path
from typing import Union

def create_folders(path_or_list: Union[list[str], str,
list[Path], Path], exist_ok: bool = True):
    """Creates multiple folders using list of paths.
    Could also create a single folder.

    Args:
        path_or_list (Union[list[str], str, list[Path], Path]):
list of string paths of single string path.
        exist_ok (bool, optional): Ignore existing folders.
Defaults to True.
    """
    if isinstance(path_or_list, str):
        os.makedirs(name=path_or_list, exist_ok=exist_ok)
    elif isinstance(path_or_list, Path):
        path.mkdir(parents=True, exist_ok=exist_ok)
    elif isinstance(path_or_list, list):
        for path in path_or_list:
            if isinstance(path, str):
                os.makedirs(name=path, exist_ok=exist_ok)
            elif isinstance(path, Path):
                path.mkdir(parents=True, exist_ok=exist_ok)
            else:
                raise Exception(f"Given path has to be [str] or
[Path], processed path - {path}")
    else:
        raise Exception(f"Input has to be in set [str,
list[str], Path, list[Path]], processed input - {path_or_list}")
```

Додаток З – Тези наукової конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ
2022

УДК 004.056
М. Гаврилов
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПОВТОРНА ІДЕНТИФІКАЦІЯ ЛЮДЕЙ ЗА ФОТО ТА ВІДЕО ЗАСОБАМИ COMPUTER VISION

UDC 004.056
M. Havrylov

RE-IDENTIFICATION OF PEOPLE FROM PHOTOS AND VIDEOS BY MEANS OF COMPUTER VISION

Повторна ідентифікація особи (Re-ID) широко вивчалася як проблема пошуку конкретної особи через камери, що не перекриваються. Мета Re-ID полягає в тому, щоб визначити, чи з'являлася конкретна особа в іншому місці в інший час, зафіксоване іншою камерою. Особа може бути представлена зображенням, відеорядом і навіть текстовим описом. У зв'язку з гострою потребою громадської безпеки та збільшенням кількості камер спостереження в університетах, тематичних парках, на вулицях і тому подібне, повторна ідентифікація особи є обов'язковою у розробці інтелектуальних систем відеоспостереження. Re-ID напрямок виник не так давно, але враховуючи його дослідницький вплив та практичну важливість, він продовжує стрімко розвиватись.

Повторна ідентифікація особи є складним завданням через наявність різних точок зору, різну роздільну здатність зображення, зміну освітленості, невимушені пози, оклюзії, тощо. Ранні дослідницькі зусилля зосереджені в основному на ручній конструкції елементів зі структурами тіла або дистанційне метричне навчання. З розвитком глибокого навчання Re-ID досяг надихаючої продуктивності за широко використовуваними тестами. Проте все ще існує великий розрив між дослідницькими сценаріями та практичним застосуванням. Це спонукає багатьох осіб, які зацікавлені в даному напрямку шукати методи удосконалення Re-ID.[1]

Під час повторної ідентифікації особи порівнюють досліджувану особу з набором осіб із існуючої бази картинок для створення ранжованого списку відповідно до їх подібності, зазвичай припускаючи, що правильний збіг призначається одному з вищих рангів, в ідеалі рангу 1. Оскільки люди з галереї часто знімаються з пари камер, які не накладаються одна на одну, у різний час, візуальні варіації зовнішнього вигляду можуть бути значними. Повторна ідентифікація шляхом візуального зіставлення за своєю суттю є складною. Сучасні методи виконують це завдання здебільшого шляхом зіставлення просторових характеристик зовнішнього вигляду (наприклад, гістограми градієнта кольору та інтенсивності) за допомогою пари однознімкових зображень людини. Однак функції зовнішнього вигляду одного кадру внутрішньо обмежені через притаманну візуальну неоднозначність, спричинену подібністю одягу людей у громадських місцях, а зовнішній вигляд змінюється через зміну освітленості в поперечному ракурсі, різницю в точках огляду, захаращений фон і оклюзії. Бажано досліджувати просторово-часову інформацію з послідовностей зображень людей для повторної ідентифікації в громадських місцях [2].

В подальших дослідженнях планується покращити наявне рішення Re-ID під конкретно встановлені задачі цієї роботи, а саме для повторного виявлення конкретних людей на інших відео або фото матеріалах.

Література

1. Ye, Mang, et al. «Deep learning for person re-identification: A survey and outlook.» IEEE transactions on pattern analysis and machine intelligence 44.6 (2021): 2872–2893.
2. Wang et al. (2014) Wang, T., Gong, S., Zhu, X., and Wang, S. (2014). Person re-identification by video ranking. In ECCV.