Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Кафедра інформатики факультету інформатики

Знаходження рухомих об'єктів у відеопотоці

Кваліфікаційна робота

за спеціальністю «Комп’ютерні науки» 122

Керівник кваліфікаційної роботи
ст. викл. Бучко О.А.
_____*(підпис)*
«____» _____ 2023 р.
Виконав студент Федоров Д.М.
«____» _____ 2023 р.

Київ - 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Завідуючий кафедри інформатики,

доцент, кандидат наук

Гороховський С.С.

_____

(підпис)

«____» _____ 2023р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студента 4 курсу факультету інформатики

Федорова Данііла Максимовича

Тема: Знаходження рухомих об'єктів у відеопотоці

Зміст кваліфікаційної роботи:

- Вступ
- Розділ 1. Аналіз предметної області
- Розділ 2. Теоретичні аспекти алгоритму знаходження рухомих об’єктів у відеопотоці
- Розділ 3. Практичні аспекти алгоритму знаходження рухомих об’єктів у відеопотоці
- Розділ 4. Реалізація програмного застосунку
- Висновки
- Список використаних джерел
- Додатки

Дата видачі «____» _____ 2023 р.

Керівник_____*(підпис)*

Завдання отримав _____*(підпис)*

Календарний план виконання кваліфікаційної роботи:

Назва етапу кваліфікаційної роботи	Термін виконання етапу	Примітка
Отримання завдання на кваліфікаційну роботу	10.11.2022	
Пошук джерел за темою роботи	05.01.2023	
Ознайомлення з джерелами за темою роботи	15.01.2023	
Складання плану кваліфікаційної роботи	15.02.2023	
Написання кваліфікаційної роботи в цілому	20.04.2023	
Попередній захист кваліфікаційної роботи	13.05.2023	
Повне завершення написання й оформлення роботи	20.05.2023	
Публічний захист кваліфікаційної роботи	30.05.2023	

Вступ

Знаходження рухомих об’єктів – це ключове завдання у сфері комп’ютерного зору, яке дозволяє потім ідентифікувати об’єкти на зображенні або у відеопотоці. У даній кваліфікаційній роботі буде розглянуто питання знаходження та класифікації рухомих об’єктів у відеопотоці. У той час, як Штучний Інтелект надає можливість комп’ютеру «думати», Комп’ютерний Зір дозволяє машині бачити, аналізувати та розуміти об’єкти та рух[1].

Поняття комп’ютерного зору в сучасному світі невпинно розвивається, актуальність питання зростає разом із прогресом автоматизації в багатьох сферах виробництва: від розробки машин автоуправління до відстеження етапів розвитку сільськогосподарських рослин. Точність та ефективність в ідентифікації об’єктів у реальному часі є вирішальними чинниками для застосування комп’ютерного зору в даних та інших сферах. Очікується, що ринок комп’ютерного зору зросте до 20.88$ млрд до 2030 року[2].

Одним з найбільш складних аспектів знаходження та класифікації об’єктів є робота з відеопотоками. На відміну від статичних зображень, відео містить більшу кількість інформації, що створює додаткові проблеми з визначенням об’єктів, зокрема, у реальному часі.

На фоні прогресивного розвитку сфери, зростає необхідність у детальному аналізі сучасного стану комп’ютерного зору та підтримці в розробці нових революційних методів. У даній роботі буде досліджено новітні методи імплементації комп’ютерного зору для використання з метою знаходження та класифікації об’єктів у відеопотоці: використання глибинного навчання та нейронних мереж.

Метою роботи є дослідити сучасний стан та різноманітні аплікації знаходження об’єктів, проаналізувавши предметну область застосувань у найбільш задіяних у цьому сферах. Буде розглянуто використання комп’ютерного зору в медицині, промисловості та будівництві, торговельній та військовій справах та інших.

У наступних розділах буде розглянуто теоретичні аспекти знаходження об’єктів у відеопотоці: варіативні нейронні мережі та алгоритми, створені для ефективної роботи; проблеми, ініційовані аналізом великої кількості даних у відеоматеріалах, оглянуто сучасні дослідження та рішення цих викликів у сфері. Буде проаналізовано структуру та застосування технологій машинного навчання та нейронних мереж на прикладі алгоритму знаходження та класифікації об’єктів. У 2 розділі продемонстровано варіації нейронних мереж та їхні відмінності у використанні в умовах обмежених потужностей, а також загальний вплив машинного навчання на розвиток сфери.

Після цього, буде досліджено практичні аспекти знаходження об’єктів: підбір доцільної для завдання сучасної моделі – нейронної мережі, також опрацювання та підготовка зібраних даних для передачі, враховуючи її особливості та потреби розробника. Насамкінець, для проаналізованої інформації буде сформульовано методи для збереження та візуалізації для цільової аудиторії.

Наостанок, у роботі надано реалізацію програмного застосунку з використанням сучасних технологій для аналізу об’єктів у відеопотоці. Детально буде розібрано практичний підхід до вибору моделі з Tensorflow Hub, підготовка даних з використанням Numpy та OpenCV на конкретному прикладі та оптимізація застосунку відповідно до варіативних вимог.

У цій роботі використано джерела з провідних досліджень у даній сфері, а також статті та онлайн-ресурси від сучасних компаній-розробників продуктів для комп’ютерного зору й конкретно – аналізу зображень та відео для виявлення об’єктів.

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості

Як було зазначено у вступі, комп’ютерний зір дозволяє машинам «бачити», а також аналізувати та досліджувати світ навколо себе. Така можливість, безумовно, зацікавлює багато бізнесів та варіативних сфер до використання комп’ютерів у автоматизації та полегшенні процесу, а розробників – для розвитку та майбутньої покращеної імплементації застосунків.

Автоматичне знаходження та ідентифікація об’єктів допомагають індустріям розвиватися швидше, а точність та ефективність таких алгоритмів дозволяють бути впевненими в мінімальній похибці машин, особливо в сферах, де помилки є критичними, зокрема, використання автоматизованого процесу зменшує «людський фактор» в виробництві.

1.2 Застосування комп’ютерного зору у знаходженні рухомих об’єктів в індустріях

1.2.1 Транспорт

Зростаючі потреби в секторі транспорту викликали бурхливий технологічний розвиток індустрії, разом з використанням комп’ютерного зору.

Від вищезгаданих машин автоуправління – до аналізу зайнятості місць на паркінгу, сектор «розумного транспорту» зростає, підвищуючи ефективність, зручність та безпечність транспортування[3].

Під час розробки машин автоуправління, застосовуються алгоритми знаходження об’єктів та аналізу руху, що допомагає виявленню пішоходів, світлофорів, інших машин, а також перешкод, у реальному часі. Використовуючи дані, зібрані з камер, згадані вище застосунки здатні проаналізувати ситуацію навколо та передати зібрану інформацію на інші пристрої, що можуть приймати подальші автономні рішення, уникаючи безпосереднього впливу людини на процес.

Окрім цього, алгоритми класифікації об’єктів використовуються для виявлення порушень на дорозі, зокрема, невдало припаркованих автомобілів, або задля аналізу трафіку на шосе[4]. Також, доцільно навчені моделі здатні розпізнати на зображенні, або у відеопотоці, номери машини, що допомагає виявленню правопорушників на дорозі та автоматичного випису штрафів.

До того ж, системи з імплементованими продуктами комп’ютерного зору здатні виявляти та аналізувати порушення в стані покрівлі дороги, що може допомогти дорожнім робітникам вчасно дізнаватись про пошкодження.

1.2.2 Медицина

Кількість медичних даних, що збережена в форматі зображень, є однією з найбільш розповсюджених та багатих у світі[4]. Однак, проблема використання автоматизованих систем у медицині полягає в тому, що навіть на сучасному етапі розвитку комп’ютерного бачення, лікарі в більшості все одно схиляються до мануального аналізу знімків, зокрема рентгенівських.

Проте, комп’ютерний зір здатний допомогти автоматизувати та полегшити процес аналізу для лікарів. Зокрема, на прикладі рентгенівських знімків, комп’ютер здатний проаналізувати та «побачити» дані, непомітні для людського

ока, відстеживши та порівнявши побачене з виявленими заздалегідь патернами, на яких модель була навчена. Для цього може використовуватися алгоритм виявлення та класифікації об’єктів.

Як додаток до вищезгаданого, комп’ютерний зір здатний ідентифікувати об’єкти під час МРТ та КТ технологій, приклад надано на рисунку 1.

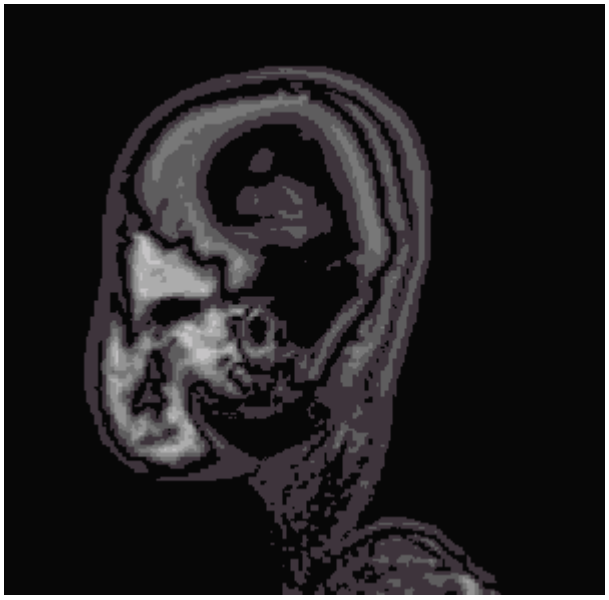


Рисунок 1. Анотація головного та спинного мозку за допомогою Машинного Навчання

Також, алгоритми ідентифікації об’єктів допомагають медичним працівникам діагностувати рак за аномаліями клітин на зображеннях, а також шляхом аналізу даних після МРТ-сканувань.

На додачу, як згадано в попередньому розділі, застосунки, що здатні ідентифікувати рух, допомагають запобігати інцидентам, пов’язаним з падінням чи іншими порушеннями руху пацієнтів у лікарнях або в будинках для людей похилого віку.

На фоні пандемії Covid-19 було створено та натреновано нейронні мережі, здатні діагностувати наявність вірусу в людини шляхом аналізу цифрових рентгенівських знімків грудної клітини. А також, було створено моделі, що ідентифікують наявність маски на обличчі в людини. Такі алгоритми допомагають, зокрема, компанії Uber визначати, чи їхні пасажири вдягають маски, і запобігти більшому розповсюдженню вірусу.

1.2.3 Промисловість і будівництво

У промисловості автоматизація з використанням комп'ютерного зору допомагає контролювати якість, мінімізувати ризики, викликані порушеннями процесу розробки, а також підвищити ефективність виготовлення продукту. Також, знаходження й аналіз об'єктів на відео допомагає розробникам слідкувати за процесом виготовлення продукту дистанційно.

Одним з основних надбань залишається виявлення дефектів у продуктів, що виготовляються. Моделі, що натреновані на зображеннях правильних продуктів, можуть ідентифікувати пошкодження або відсутність деталей у продукті, зіставляючи правильні дані – з отриманими, таким чином порівнюючи характерні риси, що відсутні на вхідному зображенні, і маючи можливість припустити похибки в нових продуктах.

Рисунок 2. Ідентифіковане порушення в виробництві ліків

Така функціональність зменшує кількість та рівень помилок під час виготовлення, а відповідно зменшується й вартість процесу, що допомагає виробництву.

До того ж, комп’ютерний зір допомагає ідентифікувати та проводити аналіз продуктів, що зберігаються на складах. Моделі, які навчені зіставляти штрих-, або інакше форматовані коди продуктів з їхніми назвами, допомагають користувачам програм перевіряти наявність продуктів на складі та слідкувати за їхньою кількістю без необхідності проводити мануальний аудит.

Також, навчені моделі здатні використовувати відео мануальної роботи робітника для того, аби слідкувати за його продуктивністю та діями. Дані, отримані таким шляхом, допомагають аналізувати управління часом робітника, а також взаємодією людей на роботі. Збір такої інформації, зокрема, може допомогти розробникам сформувати алгоритмізацію дій робітників для майбутньої автоматизації процесу з залученням роботів. Як було сказано вище

про ідентифікацію носіння масок, імплементовану Uber, ідентифікація об’єктів також допомагає слідкувати за використанням необхідного захисного обладнання у людей під час роботи.

У будівництві комп’ютерний зір допомагає ідентифікувати корозію та дефекти матеріалів, такий підхід допомагає людям та відповідним машинам заздалегідь ініціювати заміну пошкоджених елементів та уникнути майбутніх проблем.

1.2.4 Сільське господарство

Сільське господарство вирізняється зручністю автоматизації процесу, тому використання багатьох комп’ютерних систем у цьому секторі є показовим та розповсюдженим.

Зокрема, комп’ютерний зір допомагає в моніторингу етапів розвитку урожаю, автоматизованому зборі культур, аналізі погодних умов, слідкуванням за здоров’ям свійської худоби та виявленні захворювань у рослин.

Зокрема, натреновані моделі з використанням камер, облаштованих на фермах, дозволяють керувати дистанційними фермами, обмежуючи людську роботу фізично. Наприклад, відслідковуючи насадження за допомогою дронів, користувач може проаналізувати етап зрілості урожаю, оскільки натреновані моделі дозволяють використовувати алгоритми, що вміють розпізнавати цикл розвитку угідь. Також, системи можуть допомогти фермерам ідентифікувати кількість комах та шкідників, а отже передбачити використання хімікатів для збереження урожаю.

Рисунок 3. Моніторинг етапу розвитку сільськогосподарських рослин, а також наявність шкідливих насаджень

Правильно навчені системи також можуть автоматизувати розпилення хімікатів в тих місцях, де ростуть шкідливі, не сільськогосподарські, культури за допомогою візуального аналізу та розрізнення рослин, що проростають на території.

Окрім цього, системи з використанням комп’ютерного зору можуть ідентифікувати стан ґрунту й ініціювати його автоматичне зрошення, таким чином підвищуючи максимальну родючість культури.

1.2.5 Торговельна справа

У торговельній справі також зустрічається використання натренованих моделей для розпізнавання та знаходження об’єктів та людей. Наприклад,

отримуваний відеопотік з камер може бути проаналізований за допомогою алгоритмів комп’ютерного зору на рухи покупців, аби відстежувати та, за потреби, переміщувати елементи магазину в найбільш заохочувальний для користувача й вигідний для бізнесу структурний план. Окрім цього, кількість проаналізованих людей може допомогти торговельному бізнесу зрозуміти статистику його популярності, або лімітувати кількість людей в відповідних ситуаціях.

Як у випадку з Covid-19, моделі мають можливість ідентифікувати, чи покупці дотримуються соціальної дистанції на території закладу, таким чином, менеджери можуть запобігати поширенню хвороби та зберігати безпечний маршрут для покупців.

Одним з популярних наразі засобів використання комп’ютерного зору є система аналізу взаємодії покупця з касами самообслуговування з метою покращення дизайну та адаптації таких кас до потреб користувачів. Окрім цього, Приватбанк створив унікальну FacePay24 – можливість оплачувати покупки в магазині обличчям[5]. Такий спосіб можливо налаштувати в додатку банку, а на касі обрати варіант оплатити через FacePay24. Застосунок аналізує фотографію користувача та конвертує її в цифровий код, який потім порівнює з конвертованим зображенням людини, яку бачить камера на касі.

Окрім цього, комп’ютерний зір дозволяє ідентифікувати відсутність елементів на позиціях у магазині, і може ініціювати автоматичне наповнення полиць з урахуванням налаштованої системи.

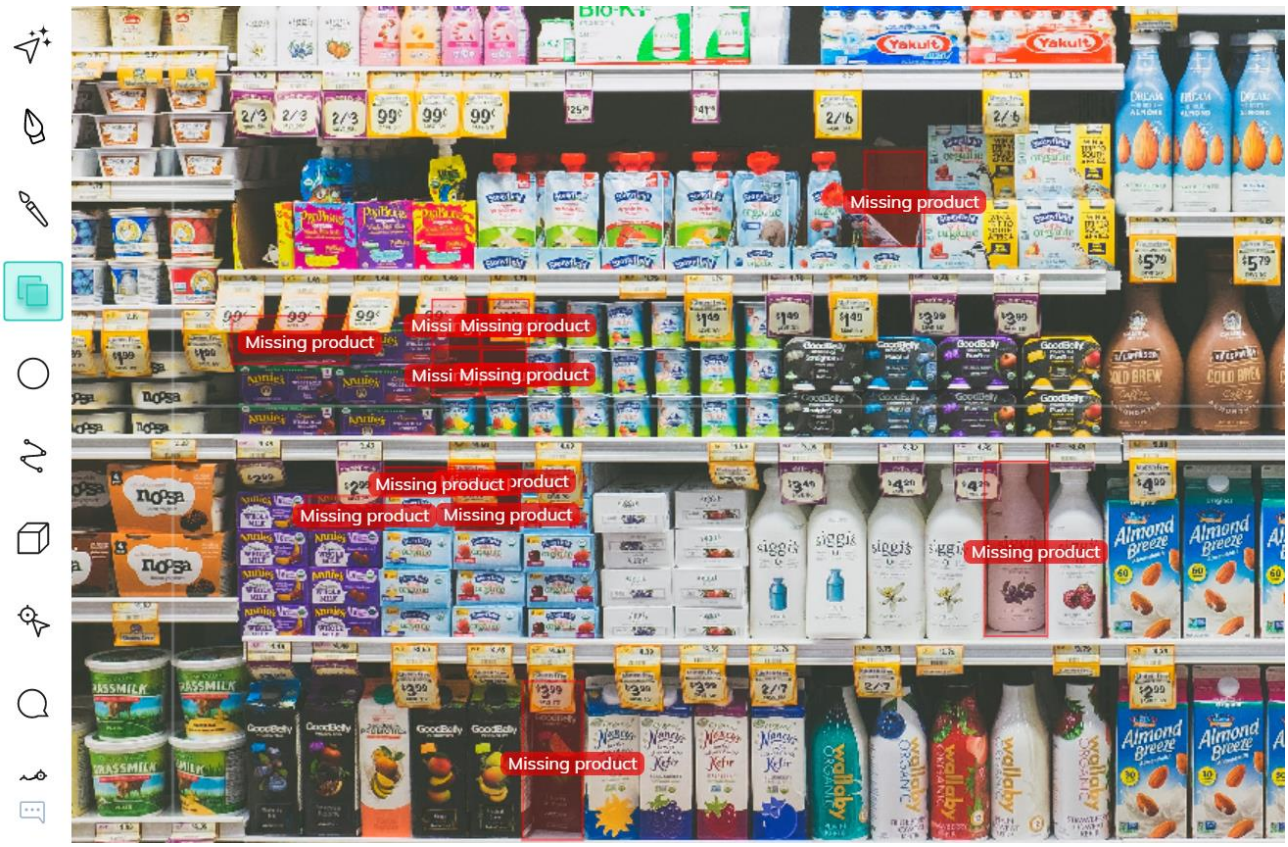


Рисунок 4. Відстеження недостатніх продуктів на полицях магазину

1.2.6 Спорт

Індустрія спорту постійно розвивається й підвищується рівень атлетів, а тому організації шукають можливостей покращити власні показники результатів. У цьому їм частково допомагає використання комп’ютерного зору[6].

Зокрема, для плавців, використовується технологія відстеження пози атлета під час руху. Таким чином, аналізуючи відео з плаваючою людиною, модель здатна автоматично запропонувати покращення в техніці, а також сформулювати дані щодо руху всього тіла людини без необхідності окремо

аналізувати кожну частину тіла, що здатне допомогти в визначенні якості підготовки та технічності атлета.

Окрім цього, комп'ютерний зір здатний ідентифікувати положення тіла людини в просторі без використання додаткових маркерів, в умовах, коли атлету використання додаткових фізичних елементів стає обмеженням. Наприклад, під час скелелазіння, для ідентифікації руху людини та мінімізації майбутніх ризиків від неправильних рішень атлета.

Комп'ютерний зір допомагає аналізувати продуктивність кожного атлета в груповому спорті, а також здатний ідентифікувати та покращувати позиції, специфічні для кожного виду спорту, аналізуючи та порівнюючи відео спортсменів з наданою базою. Також, такі застосунки здатні повертати зрозумілі для людей дані, аби згодом атлети змогли проводити статистику, порівнювати індивідуальні досягнення та покращувати рівень командної взаємодії.

Також, системи комп'ютерного зору здатні надавати автоматичні системи тренінгу в реальному часі, які б покращували позицію та форму атлета. Наприклад, автоматичний тренер з йоги може допомогти користувачу правильно стати в необхідну позицію.

Наостанок, знаходження об'єктів допомагає ідентифікувати гол під час гри, тим самим зменшуючи фактор людини в прийнятті рішення й створення нечесної партії. Як результат, правильно організовані системи можуть виконувати обов'язки тренерів чи суддів на змаганнях.

1.2.7 Військова справа

Військова справа завжди спиралася на найсучасніші технології[7], а отже точність та ефективність, яку надають можливості комп’ютерного зору, виявились використовувані.

Зокрема, системи використовуються в контролі та стеженні за територією. Моделі, як зазначалося вище, здатні ідентифікувати необхідні автомобілі, людей та інші об’єкти в зоні військової операції. Це може допомогти ідентифікувати потенційні загрози та підвищити загальне розуміння ситуації для війська.

Системи відстеження об’єктів допомагають слідкувати за загрозами та допомагають персоналу взаємодіяти з ними більш точно. Зокрема, виявлення та класифікація повітряних та наземних цілей для завдання вчасного удару по ним.

Також системи комп’ютерного зору використовуються в зборі розвідних даних: зокрема аналіз поверхні землі дронами та іншими засобами автоуправління.

Окрім цього, алгоритми знаходження об’єктів допомагають самонаведенню ракет. Аналіз поверхні та об’єктів зумовлює точне наведення ракети відповідно до патернів руху цілі та навколишніх об’єктів, і їхньої класифікації задля вдалого розпізнавання цілі. Ця технологія також використовується для розробки автономних збройних систем з точним наведенням для мінімізації впливу людини на систему. Системи з використанням комп’ютерного зору вирізняються надзвичайною точністю та ефективністю виготовлення, оскільки камери з такого роду системами можуть виявитися дешевшими за схожі датчики руху. Наразі така технологія застосовується разом з іншими варіаціями датчиків для більш точного використання.

Розділ 2. ТЕОРЕТИЧНІ АСПЕКТИ АЛГОРИТМУ ЗНАХОДЖЕННЯ РУХОМИХ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

2.1 Загальні відомості

Алгоритм знаходження об'єктів зазнає революційних змін у сфері комп'ютерного зору[8]. Основним завданням є комбінація визначення об'єктів на зображенні, а також їхня класифікація, що робить створення та адаптацію цього функціоналу складним завданням. Вдалим та сучасним рішенням проблеми є використання машинного навчання, а саме, зокрема, популярної аплікації Convolutional Neural Network (CNN), однак, не обмежуючись виключно нею. CNN ефективно використовуються в роботі з обробки зображень та відео.

Програма із використанням CNN зазвичай поділяється на два етапи: стадію визначення характеристичних рис об'єктів та стадію класифікації об'єктів на основі визначених характеристик, зіставляючи отримані дані відносно заданих класів, описів чи назв. Під час першого етапу, CNN приймає зображення або кадр відео та накладає на нього серію згорткових фільтрів, які допомагають визначати особливості об'єктів, серед яких можна виокремити знаходження границь та кутів об'єкта. Для цього використовується, зокрема, формула згорткового шару для визначення характеристик:

$$C_{i,j} = \sum_{k,l} I_{i+k,k+l} * K_{k,l},$$

де $C_{i,j}$ – це результуючий список характеристик, $I_{i+k,k+l}$ – це вхідне зображення, $K_{k,l}$ – ядро згорткового шару

Після цього визначені характеристики передаються до наступного етапу – класифікації, де CNN використовує набір повноз’єднаних шарів аби класифікувати об’єкти на зображенні чи відеокадрі.

Особливості об’єктів, які були виявлені першим етапом у використанні нейронних мереж, - це репрезентація зображення в різних рівнях абстракції. Тобто, кожен з елементів отриманого списку відображає регіон першочергового зображення, і значення, якими ці елементи наповнені, відповідають певним характеристикам, притаманним зображенню, і відображають наскільки дані якості репрезентативні в конкретному регіоні.

Таким чином, у другому етапі класифікації, перш за все, отримані списки перероблюються в більш «сплющені», тобто менш рівневі, які можуть бути ефективніше використані для класифікації об’єктів. Отже, отримуються одновимірні списки характеристик, які репрезентують кожен з регіонів, замість багатовимірних, що були сформовані під час першого етапу.

Наступним етапом, отримані списки характеристик передаються повноз’єднаним шарам нейронної мережі, що проводять серію лінійних трансформацій. Дані перетворення формуються під час процесу навчання нейронної мережі та призначені для доцільного формування одновимірних списків, що будуть правильно репрезентувати класифікацію об’єктів у заданому регіоні, враховуючи визначені характеристики.

Наостанок, отримані списки нормалізуються відповідно розподіленню всіх характеристик цього зображення, а отже формується належність заданого списку характеристик відповідно до класів, які можуть бути визначені на зображенні. Як результат, сформовано вектор, кожен елемент якого відповідає відсотку впевненості мережі щодо належності об’єкту, розташованого на

даному регіоні, до кожного з класів[9]. Нормалізація проводиться за допомогою використання функції «м'якої максимізації»:

$$y_i = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}},$$

де y_i – це передбачена алгоритмом належність об'єкту до класу j, z_j – це оцінка класу j, а K – загальна кількість класів.

Аналіз зображення проводиться відповідно до згаданого вище процесу виконання завдань поетапно, у той час, як для роботи з відео, комп'ютеру необхідно розбивати матеріал на кадри й обробляти кожний окремий як зображення, при цьому логічно поєднуючи зображуване в різних кадрах між собою. Даний функціонал здійснюється за допомогою використання технології комп'ютерного зору – відстеження об'єктів.

Стратегія відстеження допомагає комп'ютеру визначати позицію об'єктів на поточному фреймі та передбачати їхнє переміщення відносно наступного. Це допомагає алгоритму чіткіше локалізувати та класифікувати об'єкти на кадрі, аналізуючи їхнє переміщення на відео.

2.2 Варіації нейронних мереж та алгоритмів

Використання нейронних мереж стало революційним у сфері комп'ютерного зору та, зокрема, визначенні об'єктів. Основною перевагою їхнього використання є здатність згаданих мереж до самостійного навчання та

узагальнення характеристик з великого об’єму інформації. Моделі з використанням нейронних мереж здатні визначати складні патерни в зображеннях й ефективно працювати з великою кількістю даних, які іншим алгоритмам комп’ютерного бачення є заважкими.

Також, конкретні нейронні мережі є спеціально адаптованими для ефективного використання в реальному часі, що дозволяє їм бути імплементованими в машинах автоуправління, відеоспостереження, будівництві та експлуатації роботів та в інших застосуваннях.

Окрім цього, нейронні мережі є точними та надійними системами. Вони здатні ідентифікувати об’єкти з надзвичайною точністю навіть в умовах обмеженої видимості. Наприклад, коли фігура частково схована за перешкодами для камери, алгоритм здатний проаналізувати та зіставити ті зовнішні характеристики, що наявні, з існуючими в базі даних описами[9].

В завданні знаходження об’єктів можуть бути використані численні нейронні мережі та архітектури, які дозволяють обробляти дані в конкретизованих для завдання умовах, враховуючи бажаний результат та зазначені апаратні характеристики. Правильний підбір моделі може надати значні переваги в використанні програми в специфічних умовах.

2.2.1 CNN (Convolutional Neural Network)

Як було зазначено на початку розділу, CNN – одна з широко розповсюджених мереж у використанні в алгоритмі знаходження та класифікації об’єктів. Структура мережі дозволяє досягати надзвичайної точності та ефективності у варіативності ситуацій.

Для CNN існує розширення R-CNN (Region-based CNN). R-CNN складається з двох етапів аналізу зображення: розбиття зображення на регіони для знаходження об’єктів серед даних, а потім передачі проаналізованих елементів до CNN для класифікації.

Алгоритм спершу генерує велику кількість регіонів на зображенні, які можуть містити об’єкти, користуючись сформованою вірогідністю наявності об’єкту в регіоні та Selective Search способом, який формує надані моделі регіони в залежності від ідентичності суміжних пікселів, за формулою:

$$RPNscore = \Pr(object) * IoU,$$

де RPNscore – визначена оцінка даного регіону, Pr(Object) – обрахована попереднім етапом CNN вірогідність наявності об’єкту в регіоні, IoU – Intersection over Union, який обраховується за формулою:

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union},$$

де Area of Intersection – це перетин передбачуваного регіону та заздалегідь наданого й підписаного регіону, Area of Union – це сума площі передбачуваного регіону та заздалегідь наданого й підписаного, від якої слід відняти Area of Intersection.

Після цього обирається підсписок з усіх запропонованих регіонів, які більш вірогідно містять шукані об’єкти. Після знаходження цього списку, він передається CNN для класифікації[10].

Основною перевагою використання цієї мережі є підвищена точність в виявленні об’єкту, яка досягається за рахунок використання потужних можливостей CNN в знаходженні характеристик в наданих регіонах. Однак, недоліком R-CNN є сповільне виконання коду, яке спровоковано розбиттям виконання на більшу кількість етапів (слід врахувати розбиття зображення на регіони, потім фільтрацію отриманих регіонів для виявлення тих, що є більш значущими для алгоритму). За рахунок цього, R-CNN сповільнює роботу застосунку, а отже не може бути використане в умовах, де швидкість є ключовим параметром, зокрема, в аплікаціях, орієнтованих на роботу у реальному часі.

На покращення R-CNN був створений алгоритм Fast R-CNN. Удосконалення полягає в обмеженні кількості характеристик, які розглядає мережа. Таким чином, якщо R-CNN сприймає повне зображення як список характеристик, в Fast адаптації використовується Rol pooling, який дозволяє визначати точну кількість характеристик для всіх отриманих регіонів, чим зменшує та уточнює кількість обчислень для мережі відповідно до, наприклад, формули максимізації Pool:

$$P_{i,j} = max\ (I_{2i,2j}, I_{2i,2j+1}, I_{2i+1,2j}, I_{2i+1,2j+1}),$$

де $P_{i,j}$ – це результат даної операції, а $I_{2i,2j}, I_{2i,2j+1}, I_{2i+1,2j}, I_{2i+1,2j+1}$ – репрезентація вхідних елементів списку характеристик.

Слід зазначити, що окрім використання Max Pooling функції, також використовуються варіації Average функції, що замість максимального значення, обирає середнє; а також L2-norm pooling, що обраховує квадратний корінь суми квадратів значень в кожному з регіонів.

Проаналізовані дані після цього, як і в R-CNN передаються на класифікацію об’єктів та вдосконалення обраних локацій. Також, така варіація алгоритму дозволяє обмежити пошук регіонів інтересу для мережі, що зменшує кількість параметрів, які розглядаються в цілому, суттєво покращуючи ефективність обрахувань[11].

Після цього також був представлений Faster R-CNN. Основною відмінністю цієї адаптації - від Fast є метод генерації регіонів для знаходження об’єктів.

На відміну від Fast R-CNN, де може використовуватися окремий підхід для визначення регіонів, адаптація Faster дозволяє використовувати Region Proposal Network (RPN) для цієї цілі, накладаючи спеціальний метод, умову, на пошук областей на зображенні. RPN отримує список характеристик від CNN, які використовує для знаходження відповідних об’єктів на зображенні.

Faster R-CNN, так само як і Fast-алгоритм, отримує список характеристик від CNN, який потім аналізує, та здатний на його базі сформувати більш точні пропозиції регіонів, які він надсилає в Rol pooling [12] шляхом накладання додаткового шару CNN до того, як класифікувати об’єкти в визначених регіонах.

Підбиваючи підсумки щодо R-CNN:

- R-CNN розширяє звичайну CNN, впроваджуючи Region-based технологію для більш точного визначення об’єктів, але залишається повільним алгоритмом за рахунок використання додаткових етапів під час обрахування;
- Fast R-CNN покращує R-CNN, обмеживши кількість характеристик, за якими аналізуються всі регіони зображення;

- Faster R-CNN імплементує додатковий шар CNN для пришвидшеного визначення кількості необхідних регіонів на зображенні. Така процедура допомагає мережі формувати точніші й обмеженіші в кількості представлення щодо зображення, пришвидшуючи виконання обрахувань.

2.2.2 FPN (Feature Pyramid Network)

FPN – нейронна мережа, що також використовується у виявленні об’єктів на зображенні. Основною метою та передумовою створення даного типу мереж є проблема знаходження та класифікації об’єктів різних розмірів на зображенні або на відео фреймі.

Багато мереж, як, наприклад, SSD (Single Shot Detector), працюють за архітектурою обрахувань списків характеристик «знизу-вверх», у той час як FPN впроваджує також і систему «зверху-вниз», аби зручно будувати відповідні поєднання об’єктів на різних етапах обрахування[13].

Рухаючись знизу вгору, модель зменшує просторову роздільну здатність, збільшуючи семантичне значення кожного шару обрахунків. Використовуючи таку архітектуру, наприклад, алгоритм SSD використовує лише верхні шари, оскільки семантична важливість нижніх, перших, шарів не є суттєвою. У такому випадку ігноруються менші за розмірами елементами. Під час такого руху використовується формула:

$$C_i = relu(W_i * C_{i-1} + b_i), \text{ де}$$

C_i – це отриманий список характеристик після накладання i-того шару CNN на зображення, relu – rectified linear unit activation function, W_i – вага i-того згорткового шару, b_i – зсув i-того згорткового шару.

Слід зазначити, що формула ReLU – це формула, що застосовується в нейронних мережах, особливо в CNN, і визначається як:

$$f(x) = \max(0, x), \text{ де}$$

x – вираз, що передається даній функції.

Дана функція повертає значення, якщо воно є додатнім, або 0 – в іншому випадку. Функція ReLU має переваги в порівнянні з іншими нормалізуючими функціями, як сігмоїдною та tanh, серед яких:

- Підвищена конвергенція під час тренування;
- Зменшена подібність в проблемах зникаючого градієнту;
- Ефективність в обрахуваннях, оскільки дана функція є простою операцією порогового значення.

FPN створена для того, аби вирішити цю проблему. Таким чином, впроваджена мережа дозволяє, рухаючись зверху-вниз, додавати більш високоякісні шари роздільної здатності з семантично багатшими шарами. Оскільки внаслідок руху вгору та вниз, позиція об’єкта не є стабільною, використовуються додаткові з’єднання для визначення позиції об’єкта на шарі. Під час руху зверху-вниз використовується дана формула:

$$G_i = relu(W_i * P_i + b_i + Upsample(G_i + 1)), \text{ де}$$

P_i – результат роботи і-того згорткового шару під час проходження алгоритму знизу-вгору, G_i – результат роботи (і+1)-ого згорткового шару під час проходження алгоритму зверху-вниз, W_i – вага згорткового шару, b_i – зсув згорткового шару, Upsample – операція підвищення просторової роздільної здатності шара в 2 рази.

У результаті проходження алгоритму в обидва боки, необхідно поєднати результати накладання відповідних згорткових шарів на зображення, для цього використовується формула «бічних з'єднань»:

$$F_i = relu(W_i * P_i + G_i + b_i), \text{ де}$$

P_i – це результат роботи і-того згорткового шару під час руху знизу-вгору, G_i - результат роботи і-того згорткового шару під час руху згори-вниз, W_i – вага згорткового шару, b_i – зсув згорткового шару.

Наостанок, для покращення отриманих результатів, використовується функція зворотних з'єднань, яка проходить згори-вниз:

$$H_i = relu(W_i * F_i + b_i + Upsample(H_i - 1)), \text{ де}$$

F_i – результат поєднаної роботи і-того згорткового шару в обидва боки, H_i – результат роботи (і-1)-ого згорткового шару після ф-ї зворотних з'єднань, W_i – вага згорткового шару, b_i – зсув згорткового шару, Upsample – операція підвищення просторової роздільної здатності шара в 2 рази.[14]

Прикладом використання FPN є сім’я EfficientDet. Дані моделі представлена у 2019 році є поєднанням архітектури EfficientNet, запропонованої у [15] та FPN-мережі.

EfficientNet використовує спільне поєднання розширень характеристик зображення (глибини, ширини та якості) для мережі збалансовано. Це допомагає створенню менших та точніших моделей за ті, які індивідуально працюють з кожним параметром.

EfficientDet використовують FPN для генерування списків характеристик на різних масштабах задля більш точної локалізації об’єктів. Завдяки запропонованому FPN алгоритму аналізу шарів в обидва боки (зверху-вниз і навпаки), моделі здатні охоплювати як характеристики вищих рівнів шарів (більш семантично важливі), так і якості дрібніших об’єктів, що розташовані на нижчих шарах семантики й резолюції.

Прикладом використання FPN на рівні сім’ї нейронних мереж можна назвати RetinaNet – запропоновану в 2018 році архітектуру [16]. Згадана мережа була розроблена для вирішення проблеми локалізації менших за розміром об’єктів на зображенні. Ключовою відмінністю моделей сім’ї є їхня одноетапність, тобто опрацювання щільного набору ймовірних локацій об’єктів, на відміну від двоетапних представників, які розріджують (фільтрують) локації перед передачею їх на класифікаційний етап. Одноетапні є більш простими та швидшими, але менш точними в порівнянні з двоетапними.

RetinaNet використовує FPN для отримання характеристик з зображення, які потім формують багаторівневу піраміду характеристик, що описують об’єкти в різних масштабах. Після цього, дана піраміда передається підмережі, що класифікує класи об’єктів та визначає їхні межі.

Окрім цього, RetinaNet розширена функцією з «вагами», яка допомагає моделі уникнути незбалансованості класів, що представлені серед набору даних. Таким чином, класи, які легше ідентифікувати, отримують меншу вагомість, аніж складніші до ідентифікації класи. Таким чином, модель не перенавантажується легкими прикладами та об’єктами, і покращує власну ефективність в визначенні менших об’єктів. Дана функція ініційована аби покращити точність в порівнянні з іншими одноетапними представниками та конкурувати в точності з двоетапними варіаціями.

2.2.3 YOLO (You Only Look Once)

YOLO, You Only Look Once, - інший алгоритм для знаходження об’єктів на зображенні та у відеопотоці. Це нейронна мережа, що відома своєю точністю та швидкістю, а тому нерідко застосовується в аплікаціях, що орієнтуються на визначення та класифікацію об’єктів в режимі реального часу.

YOLO працює в один етап, на відміну від вищезгаданих R-CNN, це й надає йому перевагу в обробці даних в пришвидшеному варіанті. Двоетапні алгоритми, як було згадано вище, спершу виділяють об’єкти, або запропоновані регіони, що їх містять, а потім класифікують їх за класами; You Only Look Once робить це одночасно[17].

Алгоритм розділяє зображення на мережу(grid) клітин, і для кожного з розділених елементів визначає розташування об’єктів у ньому й класифікує їх за класами. Розбиття відбувається за формулою:

$$w_{cell} = \frac{w_{image}}{S}$$

$$h_{cell} = \frac{h_{image}}{S},$$

Де w_{cell}, h_{cell} – ширина та висота отримуваної клітини, w_{image}, h_{image} – ширина та висота зображення, S – розмір Grid в клітинах.

Після цього, результати формуються на основі передбачених значень точності для місця розташування та ідентифікованого класу.

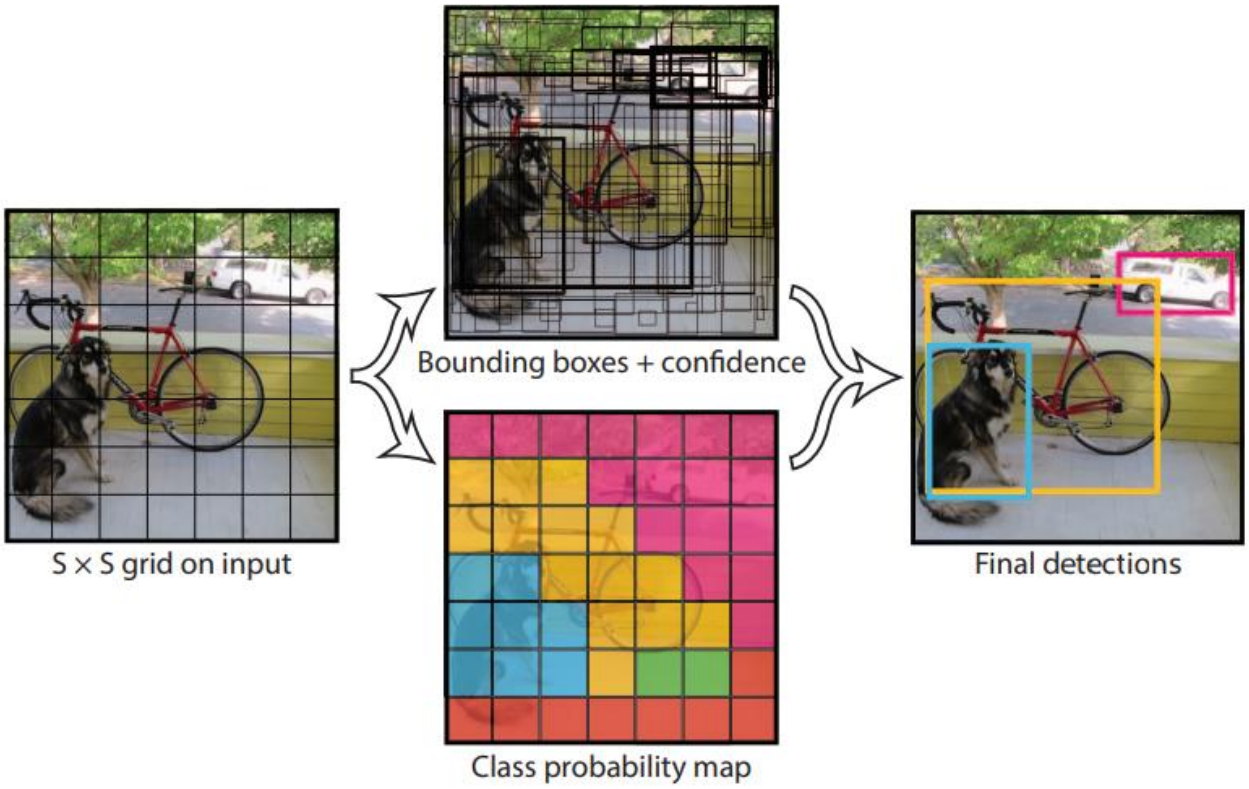


Рисунок 5. Процес виявлення та класифікації об’єктів за допомогою алгоритму YOLO[17]

Згідно з дослідниками штучного інтелекту в компанії Фейсбук, YOLO вирізняється своєю швидкістю: звичайна модель алгоритму здатна

опрацьовувати 45 кадрів у секунду, у той час як швидка версія має змогу витримувати навантаження до 155 кадрів у секунду, при цьому зберігаючи подвійну точність у порівнянні з іншими детекторами, що застосовуються в аплікаціях, орієнтованих на роботу в реальному часі[19].

2.2.4 SSD (Single-Shot Detector)

SSD – Single-Shot Detector – це архітектура на базі нейронної мережі, що також широко використовується в застосунках з метою знаходження об’єктів. Даний підхід розширює CNN додатковими компонентами, що дозволяють визначати об’єкти.

Основною перевагою використання цього методу є можливість проводити визначення та ідентифікацію об’єкту на зображенні за один прохід обрахувань. Іншими словами, після проходження даних через мережу, немає необхідності повторно її використовувати для підвищеної точності, або використовувати інші додаткові методи для досягання бажаного результату.

SSD використовує серію згорткових та шарів агрегування для визначення характеристик, що описують зображення або кадр з відео. Після цього отримані дані пропускаються через серію шарів, що визначають передбачене положення та клас об’єктів[20].

Особливість визначення локації та ідентифікації об’єктів у SSD полягає в використанні різних за масштабом та співвідношенням сторін регіонів, які задані за замовчуванням. Після кожного етапу, вони вдосконалюються відповідно функціям, які впливають на розмір та позицію даних регіонів, відносно характеристик, що були попередньо визначені з зображення.

Таким чином, Single-Shot Detector зручно визначає об’єкти різного розміру, використовуючи варіації регіонів різних розмірів, адаптуючись до характеристик. Також, алгоритм легко тренувати й інтегрувати в системи, що потребують визначення об’єктів. SSD розділяє достатню точність, порівняно з іншими варіаціями, а також є значно швидшим.

Згадані вище характеристики алгоритму забезпечують його ефективне використання в застосунках реального часу, таких як керування машинами автоуправління.

Порівнюючи SSD і вищезгаданий YOLO:

SSD є більш корисним та використовуваним у застосунках реального часу, де необхідна точність, яка досягається за рахунок використання згорткових шарів, генеруючи список характеристик для зображення. Однак, SSD має проблеми з ідентифікацією доволі маленьких об’єктів, та може гірше працювати з моделями великих розмірів.

YOLO, з іншого боку, вирізняється дуже швидкою роботою, але за рахунок цього зменшується точність та виникають проблеми з ідентифікацією об’єктів, що знаходяться поруч.

Підсумовуючи, SSD вирізняється точністю в порівнянні з YOLO та може бути застосований у тих областях, де це впливає більше: виявлення орієнтирів, правових розслідуваннях та інших. YOLO ж може бути варіантом у застосунках, де надмірною точністю можна знівелювати: наприклад, відслідковування фермерських угідь, слідкування за станом фруктів та овочів[21].

2.3 Робота алгоритму в умовах обмежених потужностей

У цьому розділі розглянемо роботу алгоритму в умовах обмежених потужностей. Оскільки завдання визначення об’єктів на зображенні та відеокадрах можуть бути використані в багатьох варіаціях застосунків: від аналізу даних в режимі реального часу до розгляду й аналізу масивних блоків інформації чи відео, необхідно враховувати можливості та ресурси для таких аплікацій.

Алгоритми для визначення об’єктів можуть бути використані, як на GPU (Graphics Processing Unit), так і на CPU (Central Processing Unit). Очевидним є факт, що швидкість та продуктивність програми залежить від використовуваного апаратного забезпечення. Однак, виконання коду на GPU зазвичай є швидшим завдяки можливості паралельних обчислень: тобто, розділяючи велику кількість даних, GPU може обраховувати їх одночасно, що є критичним у виконанні задачі визначення об’єктів.

З іншого боку, використання GPU є більш вимогливим, як за вартістю, так і за споживанням енергії. Тому, у специфічних випадках, зокрема таких як виконання алгоритму в застосунках, що потребують аналізу в реальному часі, або на техніці, що живиться акумуляторами: смартфонах, планшетах чи роботах. У таких випадках, більш класичним є виконання застосунку на CPU.

Для виконання алгоритмів на CPU поширеною є практика використання спеціальних оптимізованих бібліотек, як OpenCV або TensorFlow Lite, які дозволяють делегувати виконання коду на CPU, а також адаптувати розподіл та використання обмежених ресурсів.

До того ж, заохочується використання «легких» моделей, які вирізняються меншою кількістю обрахувань та використанням меншої кількості параметрів.

Прикладом таких представників є сім’я MobileNet, що була оптимізована для мобільних девайсів і може бути використана в застосунках визначення об’єктів у реальному часі[22].

MobileNet як тип архітектури нейронних мереж був розроблений дослідниками в компанії Google, задля «легкого» та ефективного використання нейронних мереж на пристроях з обмеженими ресурсами.

Відкриття в розробці MobileNet полягає в використанні «depthwise» відокремлених згорток, що зменшує складність системи, при цьому зберігаючи високу точність. У звичайній аплікації CNN, кожен фільтр, тобто згортковий шар, накладається на всі вхідні канали даних, що передаються, такий спосіб викликає велику кількість обрахувань. На відміну від згаданого способу, втілений у MobileNet підхід розділяє процес на дві частини: «depthwise» згортки накладають окремий фільтр на кожен з вхідних каналів, а потім використовуються точкові згортки, які, використовуючи фільтр розміру 1x1, поєднують результати з тими, що були після «depthwise» згортки.

У цьому випадку, «depthwise» згортка створює список характеристик для кожного каналу вхідних даних, а точкова згортка допомагає поєднати сформовані «depthwise» згорткою списки, формуючи результуючий список характеристик та кількість каналів з інформацією, що передаються наступному етапу[20].

Таким чином, зменшується кількість необхідних для обрахунку параметрів, оскільки накладається лише 1 фільтр на кожен канал даних, замість використання всіх фільтрів на кожен канал. Також, використання точкової згортки допомагає більш точно відображати характеристики вхідної інформації, компілюючи зібрані «depthwise» згорткою списки. Отже, такий підхід підвищує

швидкість виконання алгоритму, а також зменшує кількість задіяної пам’яті, що є зручним для використання на приладах з обмеженими ресурсами.

Згадані в попередньому розділі моделі сім’ї EfficientDet також є прикладом, що широко використовується у світі аплікації завдання визначення об’єктів в умовах обмежених ресурсів.

Дані моделі досягають високої ефективності алгоритму, при цьому зменшуючи кількість використовуваних параметрів та оброблюваних розрахунків, аніж інші звичні представники. Окрім цього, перевагою EfficientDet є можливість використання моделі, заздалегідь натренованої на специфічних лімітованих даних, у результатах використання в такий спосіб, ця сім’я показує перспективні результати. Таким чином, використання EfficientDet також є прикладом застосування алгоритму для девайсів з обмеженими потужностями.

2.4 Аналіз роботи нейронних мереж та візуалізація даних

Підсумовуючи даний розділ, слід вказати, що знаходження та класифікація об’єктів є ключовим завданням в задачах комп’ютерного зору. Нейронні мережі, у свою чергу, є революційним рішенням і покращенням роботи алгоритму та підвищили його ефективність. Застосування мереж є ключовим фактором, що зумовлює високу точність в аплікаціях реального часу, навіть під час аналізу об’ємних даних. Нейронні мережі здатні навчитися аналізувати велику кількість розподіленої за типами інформації та виявляти патерни схожості серед неї.

CNN – найбільш розповсюджений тип нейронних мереж, що використовуються в застосунках, які орієнтуються на виявлення та класифікацію об’єктів. Вони працюють, накладаючи серію фільтрів на

зображення чи кадр відео, та натреновані на визначення характеристик серед отриманих даних. Після цього, отримані списки характеристик передаються додатковим шарам мережі, що здатні класифікувати об’єкти за отриманими характеристиками.

У цьому розділі було розглянуто серію алгоритмів з використанням CNN: R-CNN, Fast R-CNN та Faster R-CNN. Дані варіації орієнтуються на виявлення характеристик об’єктів, після чого, використовуючи метод пропозиції регіонів, формують потенційні локації елементів. Після чого, кожен з алгоритмів має свої окремі умови та покращення, що вирізняють його від попередників, і додаткові шари мережі, які аналізують запропоновані регіони та покращують сформульовані припущення.

Після цього, було проаналізовано відмінності, що пропонує FPN. Ідея даної мережі полягає в уточненні виявлення та класифікації об’єктів, що мають маленький розмір, за рахунок проходження мережі «знизу-вверх», зменшуючи просторову роздільну здатність шарів, але при цьому збільшуючи семантичне значення; а потім «зверху-вниз», що за поєднанням двох проходжень дозволяє поєднати результати роботи рівнів і створити систему, яка за рахунок детекторів, що визначають об’єкти більших розмірів, і детекторів, що визначають менші об’єкти, підвищує точність виявлення об’єктів різних розмірів.

Після цього було розглянуто алгоритми YOLO і SSD, що орієнтовані на використанні однієї нейронної мережі для виявлення об’єктів та їхньої класифікації за один крок виконання, що робить їх швидшими за згадані вище методи. Було визначено відмінності між даними двома підходами та зазначено, що їхнє використання є зручним в застосунках, що орієнтовані на роботу в реальному часі.

Наостанок, було розглянуто роботу алгоритму в умовах обмежених потужностей.

Спершу було проаналізовано виконання застосунку на CPU та GPU: код на Graphical Unit'і може бути більш затратним у фінансовому та ресурсному плані, а тому в умовах, як зокрема мобільних девайсів, більш доцільним є проведення аналізу на Central Unit, жертвуючи багатопроцесорністю GPU, але підтримуючи ефективність роботи на апаратах з батарейним живленням або менш затратним апаратним забезпеченням.

MobileNet – один з типів нейронних мереж, що спеціально розроблений для використання на мобільних девайсах. Його відмінність полягає в використанні технології, що названа способом «depthwise» відокремлених згорток, яка допомагає зменшити кількість розрахунків, зменшивши кількість накладених на дані фільтрів, при цьому зберігши точність алгоритму.

EfficientNet і EfficientDet – технології, що також використовуються в обмежених ресурсних умовах. EfficientNet використовує комбінацію можливостей нейронних мереж в розширеннях та зміні характеристик зображення для підвищення точності та ефективності алгоритму. EfficientDet – сім'я моделей, що використовує мережу EfficientNet як основну архітектуру.

Останнім етапом після виконання обробки зображення чи відео залишається відображення даних для цільового користувача. Результатом роботи алгоритму є список регіонів визначених об'єктів разом з їхніми класами, а також рівень впевненості в точності його передбачення.

Для візуалізації отриманих результатів, регіони зазвичай наносяться на зображення або покадрово – на відео, разом з відповідним ідентифікованим класом об'єкта та оціночною точністю алгоритму на даному елементі поблизу.

Це допомагає для інтуїтивно зрозумілого людині дизайну та подальшого легкого аналізу отриманих даних у будь-якій іншій методиці.

Існує багато способів візуалізації: наприклад, використання вбудованого в багато сучасних ідентичних алгоритмів функціоналу, що наявні в бібліотеках OpenCV або TensorFlow Object Detection API. Також розробник може створити власні методи для відображення й аналізу отриманих даних, зокрема проаналізувати середній рівень впевненості застосунку у власних передбаченнях, встановлювати мінімальний рівень впевненості для відсіювання неточностей тощо.

РОЗДІЛ 3. ПРАКТИЧНІ АСПЕКТИ АЛГОРИТМУ ЗНАХОДЖЕННЯ РУХОМИХ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

3.1 Загальні відомості

Для успішного виконання алгоритму знаходження об'єктів на зображенні чи в відеопотоці на практиці слід дотримуватися відповідних правил та умов, що пов'язані з необхідною функціональністю.

Зокрема, слід приймати до уваги апаратне забезпечення девайсу, на якому буде виконуватися код, швидкість, якої має досягати аналіз зображення чи відео: наприклад, якщо робота проводиться в застосунку реального часу, слід вдало підібрати модель, що буде задовольняти необхідні швидкісні рамки.

Окрім цього, слід вважати на отримувану інформацію та правильно, якщо існує така можливість, підбирати дані, на яких алгоритм буде тренуватися та тестувати власні передбачення. Для цього мають бути надані відповідні об'єкти, які відповідають за складністю та варіативністю тим об'єктам, що будуть визначатися програмою.

На додачу, зображення чи кадри відео, що містять необхідну інформацію, мають бути правильно організовані та оброблені: зменшено показники «шуму», максимально підвищена якість зображуваних об'єктів, покращені, за можливістю, визначні характеристики об'єктів та інше. Для цього можуть бути застосовані практики зміни зображення: обрізання, нормалізація, зміна розмірів й інші.

3.2 Підбір моделі

Як було зазначено в розділах вище, ключовим елементом функціонування алгоритму є використання нейронної мережі. Обрану мережу на практиці представляє відповідна модель, яка може наслідувати та\або використовувати функціональність та підхід тієї нейронної мережі, яка є підходящою для заданих умов.

Перш за все, при підборі слід зважати на технічні ресурси, які є доступними для виконання алгоритму. Наприклад, обмеженість мобільного девайсу в потужності, заряді чи наявності необхідної кількості пам’яті може бути перешкодою для використання великих, важких моделей, і призвести до затримок у виконанні застосунку. У таких випадках, слід вивчити наявні варіанти рішень для наданих умов і, зокрема, обрати меншу та більш ефективну модель може бути кращим рішенням.

Окрім цього, слід орієнтуватися на розмір даних, що отримуються на опрацювання алгоритмом: на менших об’ємах може бути більш правильним звернутися до легших, простіших моделей, у той час як важчі об’єми можуть потребувати більш складних.

У залежності від семантичного наповнення задачі та цілей, які мають бути досягнені після виконання алгоритму, має бути дотриманий баланс між складністю моделі та її розміром. Зокрема, якщо завдання потребує вищої точності, більш комплексні приклади можуть бути рішенням; у той час, коли можна знехтувати точністю, наприклад, у аналізі стану фермерських угідь, легші моделі можуть підійти краще, оскільки окрім того, що вони швидші, вони також можуть бути використані на менш потужних девайсах, оскільки використовують менше пам’яті та ресурсів у цілому[23].

Наостанок, обрану модель необхідно оптимізувати. TensorFlow пропонує[23]:

- У моделі можна змінювати кількість потоків задля пришвидшення виконання алгоритму;
- Правильно обробляти вхідні дані: зокрема, видаляти непотрібні копії оброблених елементів, передавати їх в форматах, що швидше обробляються (наприклад, для Java API використання ByteBuffers призводить до швидшого виконання);
- Аналіз використання пришвидшувачів, доступних на обладнанні: аплікування Neural Networks API від Android, GPU-делегатів на Android та IOS й інші.

Із зазначеного вище, можна підсумувати, що підбір моделі є ключовим для вдалого виконання алгоритму. Правильно обраний представник має бути розрахований та виважений, враховуючи отримувані дані, бажану швидкість та точність аналізу та наявні потужності девайсу. Для порівняння та використання необхідних варіантів, одним із популярних варіантів підходу є аналіз рішень, представлених на TensorFlow Hub. Розробники пропонують уже створені та адаптовані моделі під різноманітні варіації можливостей та умов, яких треба досягти, а також надають достатню кількість інструкцій для підбору й адаптації її до специфічних особливостей.

3.3 Підготовка та обробка даних

Як було зазначено вище, іншим ключовим етапом для успішного виконання алгоритму, є правильна підготовка та обробка даних. Вдале навчання та

тренування моделі допоможе в майбутньому уникнути зайвих проблем із перенавчанням та повторним написанням застосунку.

Перш за все, необхідно визначити об’єкти інтересу – ті самі елементи, які цікавлять користувача для аналізу: наприклад, якщо модель буде відстежувати рух на вулиці, слід тренувати її на зображеннях та відео з людьми, автомобілями – тобто, цільовими об’єктами та елементами. Тому, мають бути підібрані різноманітні та репрезентативні набори даних, що складаються з зображень або відео, які містять об’єкти інтересу. Для точності, об’єкти мають бути варіативними у розмірі, позиції, орієнтації, з різним джерелом світла та заднім фоном й іншими оточуючими умовами.

Після цього, об’єкти для аналізу мають бути визначені та підписані на зібраних медіа. Зокрема, для кожного з них має бути назначено клас та місце розташування. Зручним варіантом рішення цього етапу є використання Roboflow – платформи для керування, підписування та інших можливих трансформацій зібраних даней для варіативних алгоритмів комп’ютерного зору.

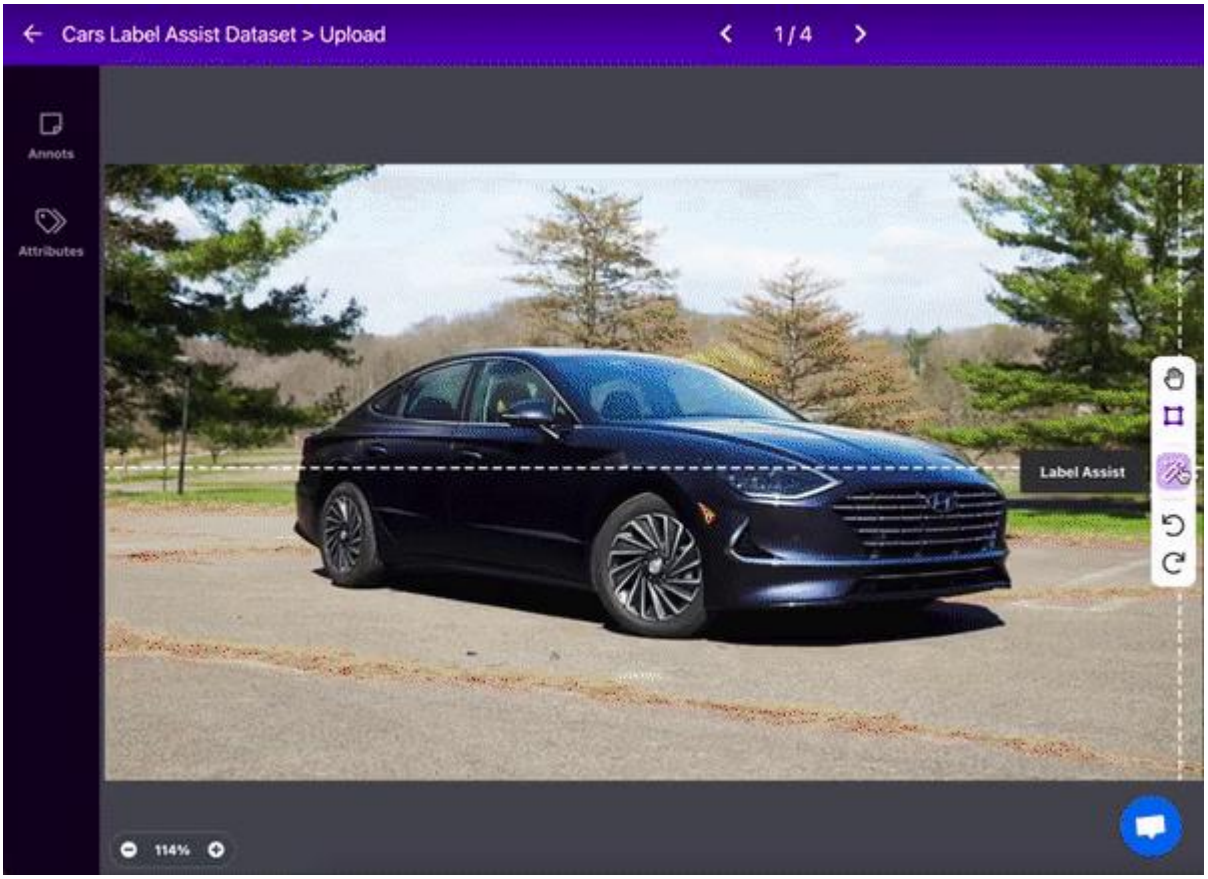


Рисунок 6. Приклад використання Label Assist для підписання елементів на додаткових зображеннях з використанням моделі користувача

Наступним кроком є очищення зібраних даних, аби прибрати непідписані елементи або дублікати. Таким чином, модель буде тренуватися на елементах вищої якості без повторень та зайвого «шуму». Також, на отриманих елементах можна використовувати техніки аугментації даних – тобто, зміна зображення: розміру, відповідності сторін, фізичні зміни положення, зміна кольору та контрасту – для кращого тренування моделі в різних ситуаціях із зображенням, що передається.

Наостанок, підписані та оброблені дані мають бути розбиті на три підсписки: для тренування, валідації та тестування. Це класичний процес для машинного навчання, який може бути виконано будь-яким із існуючих

алгоритмів: наприклад, K-fold cross validation. Спершу модель має тренуватися, або ж вчитися, на даних тренування, потім перевіряти власні параметри та запобігати перенавчанню за рахунок підсписку валідації, і фінальним етапом – перевіряти точність та оцінювати продуктивність моделі на невідомій для неї інформації.

Останнім кроком, зібрані дані мають бути відформатовані відносно формату та обраного алгоритму моделі: наприклад, YOLO і SSD формати потребують спеціальних форматів: YOLOv3, COCO(Common Objects in Context) або VOC(Visual Object Classes).

Окрім цього, як було зазначено вище, дані в коді слід зберігати в найбільш ефективних структурах та типах для більш швидкого та менш ресурсномісткого виконання.

Як підсумок, збір та організація, обробка даних є одним з ключових факторів для успішного виконання програми, який потребує детального та трудомісткого процесу визначення деталей. Однак, така прискіпливість здатна значно підвищити точність та продуктивність алгоритму визначення та класифікації об’єктів.

3.4 Збереження та візуалізаціях проаналізованих даних

Після проходження етапу детекції, визначені об’єкти та їхні регіони й класи зазвичай зберігаються як проаналізовані дані в спеціальному форматі, як, зокрема, зазначені вище COCO або VOC.

Останнім кроком в роботі алгоритму зазвичай є відображення проаналізованих даних для зручнішого розуміння результатів роботи моделі. Як зазначено в попередньому розділі, найбільш розповсюдженим варіантом візуалізації є накладання визначених регіонів об’єктів на оригінальне

зображення чи кадр відео. Також, на таких регіонах підписуються класи об’єктів для чіткішого розуміння приналежності.

Іншим варіантом для візуалізації є генерування схем щільності або «heat maps», які можуть відображати частоту або щільність визначених об’єктів в різних регіонах зображення чи відео. Такий підхід може бути корисним для виявлення зон, які мають високу концентрацію об’єктів, або також відстеження регіонів, де об’єкти можуть бути пропущені алгоритмом чи відсутні.

Окрім цього, існують варіанти кластеризації об’єктів по типу або локації, зокрема, звичайне обрахування кількості об’єктів, що зустрічаються, відповідно до зазначеного класу та інше. Також, опцією є генерації 3D point clouds[24], які відповідають за класифікацію та визначення позиції об’єкта в 3D-просторі.

Підсумовуючи, збереження, аналіз та визначення підходу до відображення даних відповідно до необхідних умов користувача є останнім, але не менш ключовим етапом в алгоритмі визначення локації та ідентифікації об’єкту. Візуалізація, зокрема, дозволяє дослідникам та іншим цільовим користувачам краще розуміти та аналізувати результати виконання й ідентифікувати потенційні проблеми з роботою моделі або використовуваними обмеженнями для її роботи.

Розділ 4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Загальні відомості

Як було зазначено впродовж роботи, виявлення об’єктів – це складне завдання, що є комбінацією передових технологій та алгоритмів.

У цьому розділі буде розглянуто реалізацію застосунку на мові програмування Python, що здатний використовувати моделі з TensorFlow Hub для виконання алгоритму знаходження та класифікації об’єктів у відеопотоці. Буде розібрано підхід до вибору моделей, що розглядатимуться, засоби підготовки вхідних даних та оптимізацію застосунку.

Одним із найбільш популярних підходів до написання такого алгоритму є використання завчасно тренованих представників з TensorFlow Hub. Платформа надає доступ до багатої вибірки моделей, які можуть бути використані для власної системи детекції об’єктів за потребами дослідника.

У цьому розділі буде сфокусовано на практичному підході до вибору моделі, її завантаженню в застосунок та використанню у відеопотоці. Окрім цього, розглянуто можливості в підготовці зібраних даних, отриманні класів та категорій для класифікації. Також, буде розібрано варіанти відображення результатів у реальному часі та збереження проаналізованих відео в систему.

4.2 Підхід до вибору моделі

Перш за все, необхідно визначитися з моделлю для використання в застосунку. Для цього можна скористатися наявними на TensorFlow Hub варіантами в розділі «Object Detection». На ресурсі надано моделі, представлені розробниками з Google, IREE, TensorFlow та іншими, які розрізняються нейронними мережами, що використовуються, а також деякі екземпляри можуть орієнтуватися на конкретне завдання, наприклад, на розпізнавання обличь.

TensorFlow надає зручний перелік моделей, їхніх швидкостей для аналізу одного зображення чи кадру відео, також точність, перевірену на COCO 2017 наборі даних, і вихідні результати, які надаються після виконання роботи [25].

Обрано дві моделі:

- EfficientDet D7 1536x1536 – модель вищезгаданої сім'ї EfficientDet, що використовує підхід SSD, разом з BiFPN, яка натренована на COCO 2017 наборі даних[26];
- EfficientDet-Lite4 – модель з тієї ж сім'ї, але полегшену версію, яка також використовує BiFPN, але оптимізована під TFLite, спеціально розроблена TensorFlow для використання на мобільних CPU, GPU та EdgeTPU[27].

Перший представник має наступні показники відповідно до розробників[28]: 153мс швидкості, 55.1% середньої точності, що є найвищим показником серед цієї сім'ї, та Boxes як вихідними даними[25]: тобто, після аналізу моделлю зображення чи кадру відео, можливо отримати відповідні регіони з визначеними об'єктами, а також їхні класи та оцінку точності алгоритму.

Друга модель має такі показники відповідно до розробників[29]: 260мс швидкості, 43.18% середньої точності.

Незважаючи на те, що першочергово швидкість другої моделі, яка є адаптованою під мобільні девайси, здається меншою, аніж першої, слід зважати, що тест проводився на різних апаратних характеристиках, що впливає на відмінні показники. До того ж, чітко продемонстровано, за рахунок точності, що більш комплексна модель на потужніших апаратних характеристиках

середовища показує більшу точність, аніж навіть повільніша й легша варіація – на мобільних девайсах.

Однак, для аналізу алгоритму в даній роботі використано обидві моделі на одному комп’ютері, що ставить їх в однакові умови відносно характеристик девайсу.

Спершу слід встановити в застосунок пакет розроблений TensorFlow з допоміжним функціоналом для візуалізації[30].

Для використання моделей у коді необхідно скористатися функціональністю TensorFlow Hub і завантажити їх зі згаданого веб-сайту за допомогою hub.load(website), де website – посилання на адресу обраного варіанту.

Однак, моделі також можна заздалегідь завантажити в систему, якщо використання інтернету не передбачено застосунком, тому hub.load() дозволяє завантажувати також з системи. Наприклад:

```
model_path = "efficientdet_lite4_detection_2"
model = hub.load(model_path)
```

Таким чином, отримано модель, яку можливо використовувати далі в застосунку.

4.3 Підготовка вхідних даних та використання моделі

Наступним кроком слід обрати та підготувати вхідні дані для використання. Оскільки обрані моделі є завчасно тренованими, тренування та тестування не проводиться.

Використовуючи бібліотеку OpenCV, що широко розповсюджена в застосунках комп'ютерного зору, слід завантажити попередньо створений файл із відео:

```
video_path = "resources/videos/street.mp4"
cap = cv2.VideoCapture(video_path)
```

Якщо застосунок працює в реальному часі, переданим має бути не файл, а відео з пристрою, яке має бути проаналізованим. Наприклад, 0 – для зчитування відео з відеокамери.

Окрім цього, слід ідентифікувати класи, які модель буде надавати об'єктам на відео. Для цього доцільним є використання заздалегідь завантажених файлів від TensorFlow. Спершу необхідно вказати на шлях до файлу, у якому містяться підписані класи відносно списку даних COCO, що створено розробниками. Після цього, використовуючи функціональність утиліт TensorFlow, слід адаптувати файл у список класів, що необхідні для даної моделі:

```
label_map_path =
"models/research/object_detection/data/mscoco_complete_label_map.pbtxt"
label_map = label_map_util.load_labelmap(label_map_path)
```

Наступним кроком слід конвертувати отримані класи в формат Eval, який буде використовуватися моделлю для категоризації об'єктів:

```
categories = label_map_util.convert_label_map_to_categories(
    label_map,
    max_num_classes=label_map_util.get_max_label_map_index(label_map),
    use_display_name=True)

category_index = label_map_util.create_category_index(categories)
```

На цьому етапі наявним є все необхідне для використання моделі у відеопотоці. Оскільки робота відбувається на відео, необхідно використовувати цикл, що проаналізує кожен з кадрів відео, для цього використано умовний цикл while cap.isOpened().

Необхідний кадр зчитується за допомогою cap.read() та додається перевірка на наявність кадру:

```
ret, image_np = cap.read()

if not ret:
    break
```

Після цього необхідно адаптувати отриманий кадр в необхідну структуру даних, що буде ефективно використовуватися. Надані вище моделі потребують масивів у форматі Numpy, які конвертуватимуться в об’єкти типу Tensor. Серед утиліт TensorFlow використаними є такі:

```
# Load image into numpy array and convert to tensor
image_np = load_image_into_numpy_array(image_np)
image_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.uint8)
```

Після цього модель може бути використана на вже адаптованих та підготовлених даних. Для цього необхідно чітко розуміти, які результати

обрахувань надає кожна з моделей. У випадку представленого застосунку, вихідні дані різняться за ключами, тому створеним має бути покажчик, що ідентифікує поточний варіант, наприклад:

```
model_type = "LITE"  # OTHER or LITE
```

Тоді, в тілі циклу:

```
if model_type == "LITE":
    boxes, scores, classes, num_detections = model(image_tensor)
    boxes = boxes[0].numpy()
    classes = classes[0].numpy()
    scores = scores[0].numpy()

else:
    outputs = model(image_tensor)

    boxes = outputs["detection_boxes"][0].numpy()
    classes = outputs["detection_classes"][0].numpy()
    scores = outputs["detection_scores"][0].numpy()
```

Отже, у залежності від типу наданої моделі, відрізняються вихідні дані й їхнє вилучення. Для наступної візуалізації, як було зазначено в розділах вище, необхідно отримати регіони виявлених об’єктів – boxes, класи та оцінку впевненості у власному передбаченні. Як було зазначено вище, функціонал використовує масиви типу Numpy, тому необхідно отримані дані конвертувати до необхідного типу.

Останнім етапом, як зазначалося в попередніх розділах, є відображення результатів. Для цього TensorFlow утиліти надають зручну функцію:

```
viz_utils.visualize_boxes_and_labels_on_image_array()
```

Яка отримує:

- image_np – оригінальне зображення;
- boxes;
- classes.astype(int);
- scores
- category_index
- use_normalized_coordinates
- min_score_thresh
- agnostic_mode

У цій функції новими залишаються use_normalized_coordinates - вказує на те, чи мають визначені регіони бути розпізнаваними за нормалізованими координатами або ні, min_score_thresh – задання нижнього порогу для відображення об’єкту на зображенні відносно впевненості алгоритму в точності; agnostic_mode – змінна типу Boolean, яка вказує, чи має функція ігнорувати клас об’єкта й лише відображати рівень впевненості ідентифікації, такий параметр може бути корисним, коли необхідно виявити об’єкти на відео, незважаючи на їхній клас.

Згадана вище функція наносить отримані регіони, класи та оцінки на оригінальне зображення, змінюючи його. Тому, результат зберігається в image_np.

Останнім етапом залишається питання відображення даного зображення користувачу. У залежності від потреб користувача, можливо зберігати отримані кадри в відео або одразу відображати їх на екран, що може бути корисним в застосунках, що орієнтовані на роботу в реальному часі. Для цього використовується функціональність OpenCV.

Для збереження проаналізованого відео, необхідно створити екземпляр класу VideoWriter з OpenCV, який за необхідними параметрами зберігатиме отримане відео. Наприклад:

```
# Get frame size and initialize video writer object
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
frame_size = (frame_width, frame_height)

out = cv2.VideoWriter('output_lite4.avi', cv2.VideoWriter_fourcc('M', 'J', 'P',
'G'), 20.0,
                      frame_size)
```

Тут спершу проаналізовано розмір вхідного відео для цільового передавання параметрів для збереження, після чого створено екземпляр VideoWriter.

Після аналізу кадру моделлю, слід використати створений об’єкт у тому ж циклі:

```
# Write the image with detected objects to video file and store it in list
out.write(image_np)
```

Іншим підходом є відображення отриманого кадру – одразу користувачу, наприклад, за допомогою:

```
# Show detected objects in real-time
cv2.imshow('frame', image_np)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Після закінчення циклу, а отже, після проходження алгоритму через всі кадри зображення, необхідно звільнити використовувані ресурси через OpenCV:

```
# Release video capture and video writer objects
cap.release()
out.release()
cv2.destroyAllWindows()
```

4.4 Оптимізація застосунку

Серед оптимізаційних функцій цього застосунку можна виділити використання OpenCV замість звичайної бібліотеки Python для обробки зображень та роботи з вхідними\вихідними потоками даних. OpenCV допомагає зручно та ефективно використовувати ресурси та оптимізовано їх контролювати за допомогою вбудованих функцій. Зокрема, release().

Також використовується модель з TensorFlow Lite: EfficientDet Lite4. Це легша варіація з TensorFlow, які розроблені для використання на мобільних та обмежених у ресурсах девайсах.

Слід додати, що за допомогою Boolean-змінної model_type можливим є використання різних за вихідними даними, змінюючи лише шлях завантаження моделі. Це допомагає розширенню цього додатку для майбутніх варіацій використання застосунку в різних ситуаціях. Зокрема, якщо спершу застосунок буде аналізувати вхідні дані: у залежності від об’єму та складності інформації, що необхідно буде проаналізувати, автоматизований підбір моделі може бути застосовано.

Окрім цього, оптимізацію можна провести за допомогою обмеження використаних моделей. Наприклад, зменшення використовуваних шарів. Також, на більш складних застосунках можливо оптимізувати TensorFlow Graph – «невидимий» граф, на якому обраховується та виконується модель, що була завантажена з TensorFlow Hub.

Також, Tensorflow може бути налаштований для виконання на GPU, керуючи девайсами за допомогою tf.config.set_visible_devices. Окрім цього, під час важких процесів, можна виділити використання пакетної обробки – обробку кількох кадрів відео одночасно замість індивідуального. Це може підвищити ефективність моделі, зменшивши кількість вхідних даних.

Застосунок протестовано на відео активного руху на вулиці, дані після аналізу всього відео (393 відео кадри):

Назва моделі / Отримані показники	EfficientDet D7 1536x1536	EfficientDet-Lite4
Середній час виконання обробки 1 кадру (с)	11,04	0,59
Середня оцінка точності виконання алгоритму (%)	0.1340	0.0999
Середня оцінка точності виконання алгоритму на	0.7502	0.7268

показниках, вищих за мінімальний поріг - 0.5 (%)		
Загальний час на виконання обробки (хв)	74,21	5,583
Висновок	Значно повільніше виконання алгоритму; підвищена загальна точність	Швидке виконання алгоритму; гірша загальна точність

З отриманих даних слід зробити висновок, що EfficientDet-Lite4 модель є значно швидшою за обробкою кадрів живого руху в реальному часі. Однак, EfficientDet D7 1536x1536 показує набагато вищі показники точності, як серед загальної кількості елементів, так і серед показників, що визначаються вищими за мінімальний встановлений поріг.

Таким чином можна підбити підсумок, що використання першої моделі є доцільним у ситуації, коли швидкістю виконання алгоритму можна знехтувати заради підвищеної точності; у той час, як друга модель нехтує точністю, але працює набагато швидше, що дозволяє її використання в застосунках реального часу, зокрема з відеокамерами.

Висновки

У цій кваліфікаційній роботі було проаналізовано сферу застосування комп'ютерного зору, а конкретно – алгоритму для знаходження та класифікації об'єктів у відеопотоці та на зображенні. Було розібрано та досліджено аплікації використання такого алгоритму в галузях сучасного світу:

- Транспорті;
- Медицині;
- Промисловості та будівництві;
- Сільському господарстві та інших.

Детально було розібрано та вивчено теоретичні аспекти роботи алгоритму знаходження та класифікації об'єктів. Проаналізовано загальну структуру методики детекції об'єктів, а також сучасний варіант використання нейронних мереж.

Глибинно визначено та продемонстровано способи використання варіативних нейронних мереж та алгоритмів, що їх використовують: CNN, разом з R-CNN, Fast R-CNN, Faster R-CNN; FPN; YOLO; SSD. Окрім цього, було оглянуто сучасні варіанти розроблених моделей, що наслідують використання відповідних нейронних мереж. Також, було надано порівняльні характеристики мереж: розібрано відмінності одноетапних та двоетапних, переваги та недоліки кожної з мереж.

Наступним пунктом детально досліджено роботу алгоритму в умовах обмежених потужностей. Надано теоретичні рекомендації і варіанти рішень для таких систем: розподіл виконання коду між GPU та CPU, використання легших моделей з прикладами: MobileNet та EfficientDet.

Після цього було підбито підсумки розділу й проведено аналіз використання нейронних мереж в визначенні та класифікації об’єктів. Зазначено, що нейронні мережі є революційним у точності та швидкості рішенням, що підвищує ефективність алгоритму. Наостанок, зазначено варіанти для аналізу вихідних даних, їхнє відображення та теоретичну важливість для дослідника.

У наступному розділі розглянуто практичні аспекти алгоритму знаходження та класифікації об’єктів у відеопотоці. Підкреслено на важливості точності та продуманості підходу до визначення моделі, обробці та підготовці вхідних даних, а також варіативності в практичній роботі з проаналізованими результатами.

Першим етапом детально розібрано підхід до вибору моделі залежно від наданих апаратних можливостей та умов, яких необхідно досягти. Сформовано та запропоновано варіанти з використанням представників, адаптації їх в умовах обмежених потужностей, врахуванні очікуваних точності та швидкості виконання, що мають впливати на обрання моделі.

Наступним кроком розглянуто підбір та обробку даних. Чітко зазначеною є важливість та трудомісткість процесу, оскільки він критично впливає на правильність використання обраної моделі, її тренування та майбутньої адаптації до роботи.

Насамкінець, у розділі зазначено практичні варіанти використання проаналізованих моделлю даних. Надано варіанти відображення об’єктів на оригінальному зображенні, що є комфортним для людського аналізу варіантом; окрім цього, зазначено про використання мап щільності та 3D point clouds для відстеження руху об’єкта в 3D-просторі.

В останньому розділі роботи продемонстровано варіант реалізації програмного застосунку мовою програмування Python, що імплементує алгоритм знаходження та класифікації об’єктів у відеопотоці. Використано моделі з TensorFlow Hub, а також функціональність OpenCv та Numpy. Чітко описано передумови до використання обраних представників та надано інформацію щодо їхніх характеристик.

Описано стрічки коду, що виконують завантаження та підготовку моделі до використання. Окрім цього, розібрано варіанти вхідних даних для аналізу, а також підготовку підписів та класів до визначення для об’єктів у відеопотоці.

Після цього, надано оптимізаційні заходи, що використовуються в застосунку. Також, перелічено додаткові варіанти покращення для практичного використання алгоритму в коді, які є доцільними для більш складних застосунків, що можуть використовувати важчі моделі, або більш об’ємні набори даних.

Наостанок, коротко проаналізовано роботу моделей з попередньо завантаженим відео, а також – в умовах реального часу – під час запису з камери. Відображено швидкості, що набуваються на аналізі даних, та сформульовано підсумок відносно використання застосунків, базуючись на отриманих спостереженнях, користуючись теоретичним підґрунтям з попередніх розділів.

Метою роботи було розглянути й проаналізувати сучасні засоби й техніки у аплікаціях алгоритму знаходження та класифікації об’єктів у відеопотоці. Розглянуто основні сфери використання в індустріях. Детально проаналізовано теоретичні аспекти завдання та революційність використання варіативних нейронних мереж у ньому. Наступним кроком досліджено практичне використання алгоритму та його адаптацію до необхідних умов. Наостанок,

продемонстровано роботу програмного застосунку, що використовуються під час знаходження та класифікації об’єктів у відеопотоці. Як результат, доведено, що розглянутий алгоритм є ключовим використанням комп’ютерного зору в світі, стрімко розвивається та має потенціал до зростання в майбутньому. Підтверджено, що використання нейронних мереж надає покращену точність та швидкість опрацювання вхідних даних, однак досі залишається відкритим плацдарм для покращень існуючих мереж.

Список літератури

Основні джерела

[1] - https://www.ibm.com/uk-en/topics/computer-vision

[2] - https://www.grandviewresearch.com/industry-analysis/computer-vision-market

[3] - https://www.v7labs.com/blog/computer-vision-applications#h1

[4] - https://viso.ai/applications/computer-vision-applications/

[5] - https://privatbank.ua/facepay24

[6] - https://viso.ai/applications/visual-ai-in-sports/

[7] - https://link.springer.com/article/10.1007/s00500-021-05912-0

[8] - https://analyticsindiamag.com/top-8-algorithms-for-object-detection/#h-1-fast-r-cnn

[9] - https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[10] - T. A. C. M. Yadav and J. S. Suri, International Journal of Computer Science and Information Security (IJCSIS), Vol. 17, No. 6, June 2019. «Object Detection with Deep Learning: A Review»

[11] - https://arxiv.org/pdf/1311.2524.pdf

[12] - https://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

[13] - https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/

[14] - https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610

[15] - https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

[16] - https://arxiv.org/pdf/1911.09070.pdf

[17] - https://arxiv.org/pdf/1708.02002.pdf

[18] - https://arxiv.org/pdf/1506.02640v5.pdf

[19] - https://medium.com/mlearning-ai/object-detection-with-yolov7-a74fa1f03c7e

[20] - https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

[21] - https://arxiv.org/pdf/1512.02325.pdf%EF%BC%89

[22] - https://technostacks.com/blog/yolo-vs-ssd/#:~:text=YOLO%20vs%20SSD%20%E2%80%93%20Which%20Are,and%20computes%20a%20feature%20map.

[23] - https://arxiv.org/pdf/1704.04861.pdf

[24] - https://www.tensorflow.org/lite/performance/best_practices

[25] - https://paperswithcode.com/task/3d-object-detection

[26] - https://tfhub.dev/tensorflow/collections/object_detection/1

[27] - https://tfhub.dev/tensorflow/efficientdet/d7/1

[28] - https://tfhub.dev/tensorflow/efficientdet/lite4/detection/2

[29] - https://github.com/google/automl/blob/master/efficientdet/README.md

[30] - https://github.com/tensorflow/models/tree/master/research/object_detection

Додаткові джерела

https://tfhub.dev/

https://roboflow.com/

https://blog.tensorflow.org/2022/07/how-roboflow-enables-thousands-of-developers-to-use-computer-vision-with-TensorFlow.js.html

Додаток А (обов’язковий) Програма алгоритму знаходження рухомих об’єктів у відеопотоці

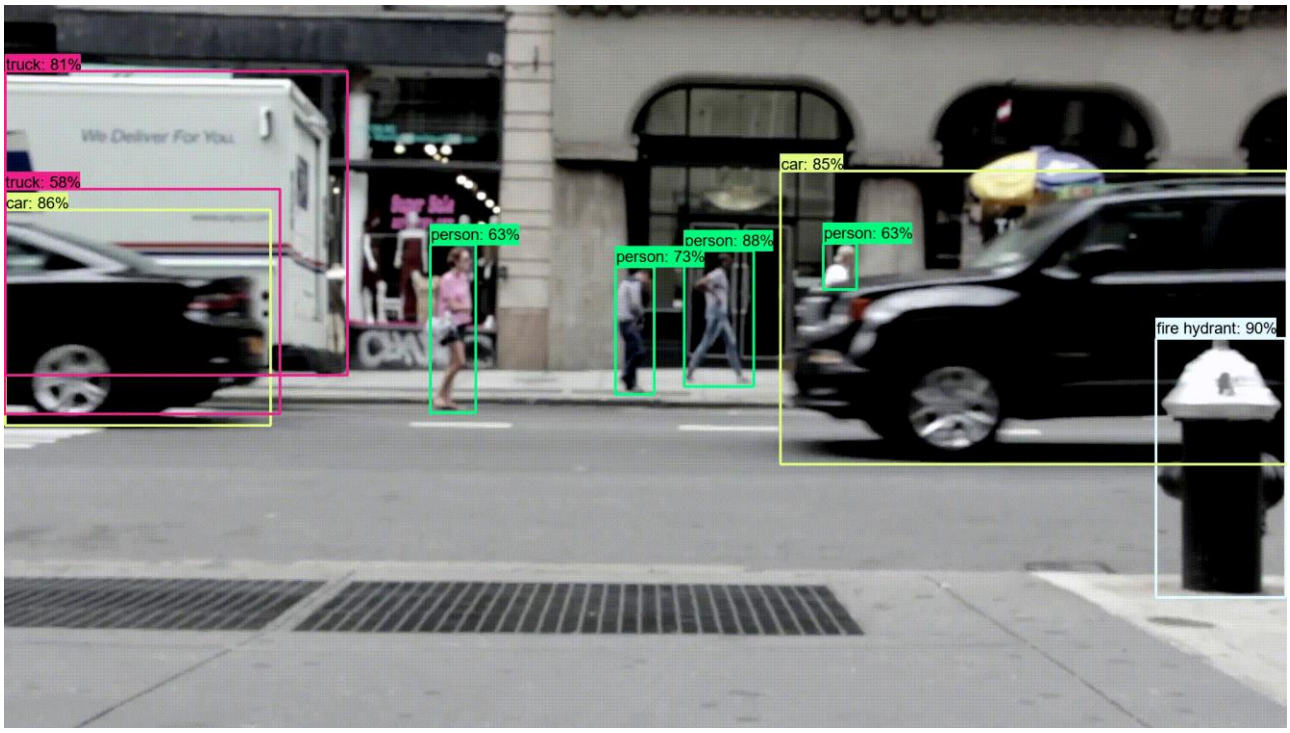
```
# Import necessary libraries
import time

import cv2
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from object_detection.utils import visualization_utils as viz_utils

from models.research.object_detection.utils import label_map_util

from statistics import mean


# Define function to load image into a numpy array
def load_image_into_numpy_array(image):
    return image.astype(np.uint8)


# Start timer
start_time = time.time()

# Load label map and create category index
label_map_path =
"models/research/object_detection/data/mscoco_complete_label_map.pbtxt"
label_map = label_map_util.load_labelmap(label_map_path)

categories = label_map_util.convert_label_map_to_categories(
    label_map,
    max_num_classes=label_map_util.get_max_label_map_index(label_map),
    use_display_name=True)

category_index = label_map_util.create_category_index(categories)

# Define input video path and initialize video capture object
video_path = "resources/videos/street.mp4"
cap = cv2.VideoCapture(video_path)
```

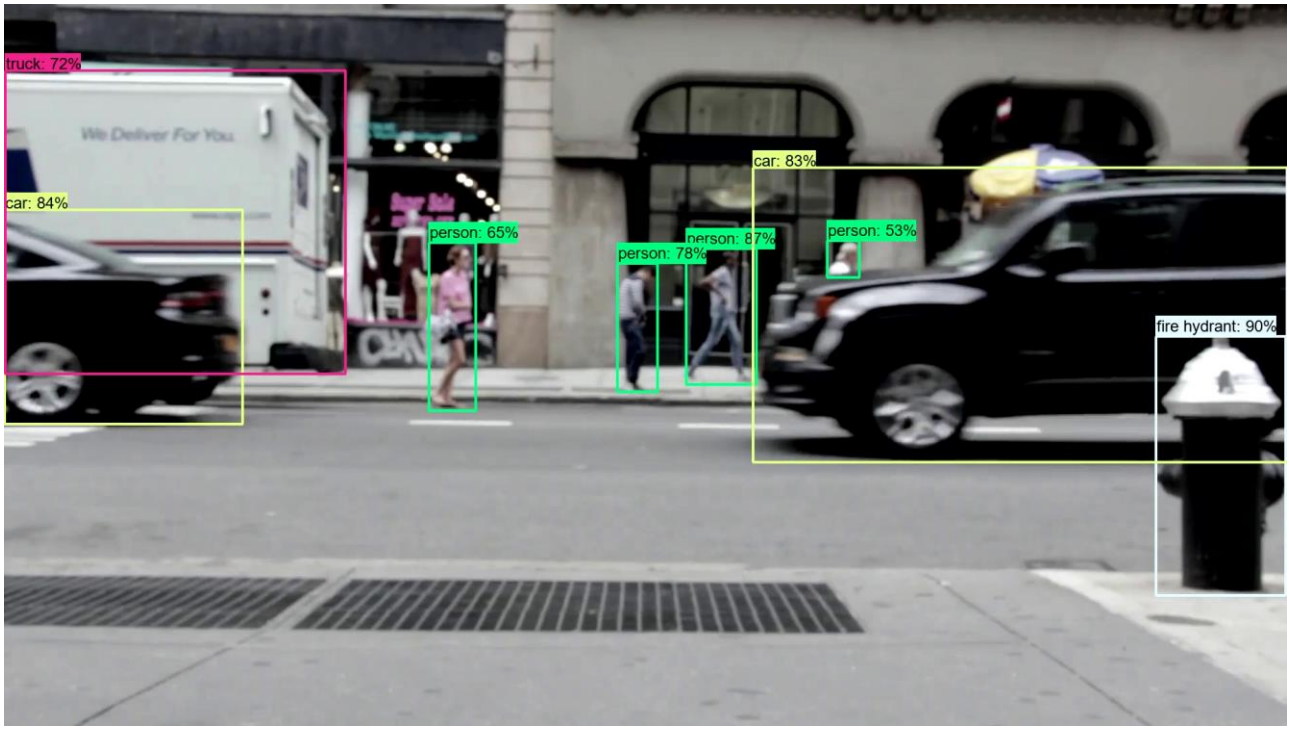
```
# Get frame size and initialize video writer object
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
frame_size = (frame_width, frame_height)

# Load Tensorflow model using Tensorflow Hub
model_type = "LITE"  # OTHER or LITE
model_path = "efficientdet_lite4_detection_2"

# model_type = "OTHER"
# model_path = "efficientdet_d7_1"

model = hub.load(model_path)

out = cv2.VideoWriter(f'output_{model_path}.avi', cv2.VideoWriter_fourcc('M',
'J', 'P', 'G'), 20.0,
                      frame_size)

# Process video frames and perform object detection
print("Starting processing video")

frame_counter = 1
frame_times = []
all_scores = []
all_scores_more_than_threshold = []

min_threshold = .5

while cap.isOpened():

    # Read a frame from the video
    ret, image_np = cap.read()

    if not ret:
        break

    # Load image into numpy array and convert to tensor
    image_np = load_image_into_numpy_array(image_np)
    image_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
```

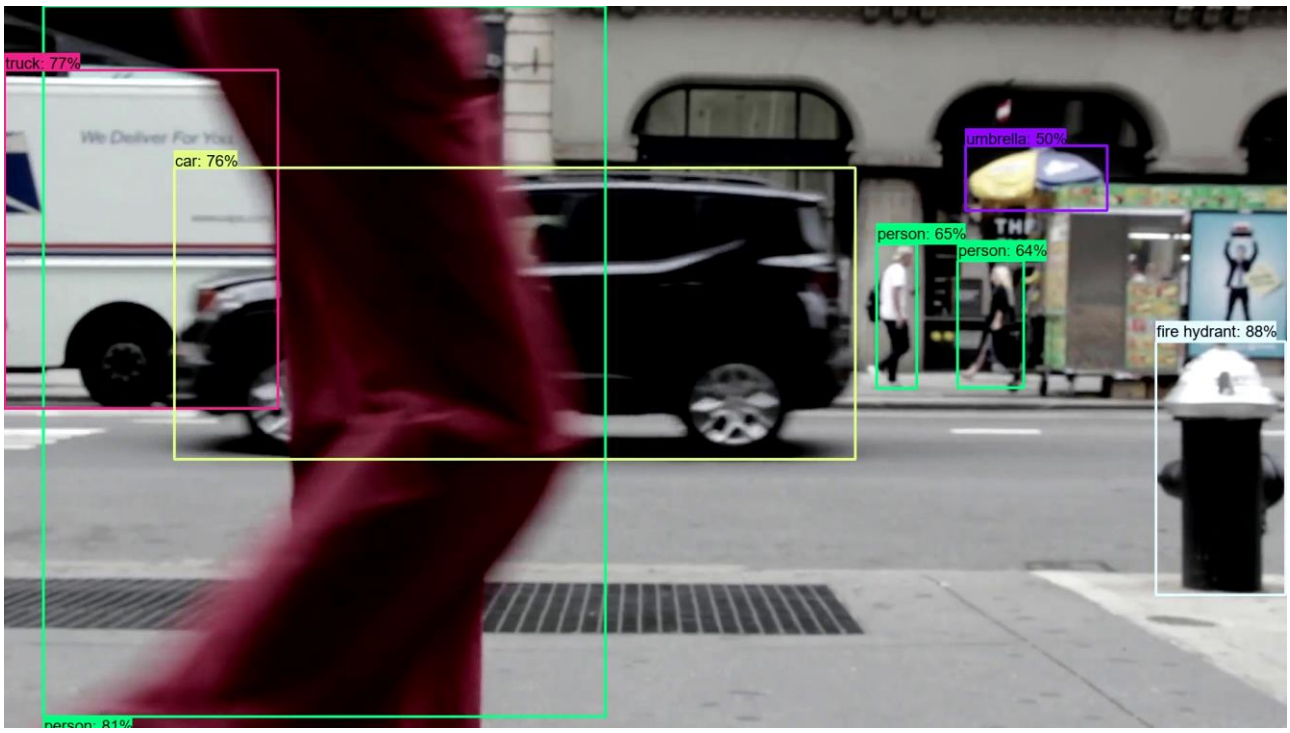
```
dtype=tf.uint8)

    frame_start_time = time.time()
    # Perform object detection using Tensorflow model
    if model_type == "LITE":
        boxes, scores, classes, num_detections = model(image_tensor)
        boxes = boxes[0].numpy()
        classes = classes[0].numpy()
        scores = scores[0].numpy()

    else:
        outputs = model(image_tensor)

        boxes = outputs["detection_boxes"][0].numpy()
        classes = outputs["detection_classes"][0].numpy()
        scores = outputs["detection_scores"][0].numpy()

    time_for_frame = time.time() - frame_start_time

    all_scores.extend(scores)
    all_scores_more_than_threshold.extend([element for element in scores if
element > min_threshold])

    # Visualize bounding boxes and labels on the image
    viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np,
        boxes,
        classes.astype(int),
        scores,
        category_index,
        use_normalized_coordinates=model_type != "LITE",
        min_score_thresh=min_threshold,
        agnostic_mode=False,
    )
    # Write the image with detected objects to video file and store it in list
    out.write(image_np)

    frame_times.append(time_for_frame)

    frame_counter += 1
```

```
    # Show detected objects in real-time
    # cv2.imshow('frame', image_np)
    # if cv2.waitKey(1) & 0xFF == ord('q'):
    #     break

print("AVERAGE TIME:")
print(mean(frame_times))

print("AVERAGE SCORE:")
print(f"Mean score: {mean(all_scores)}")
print(f"Mean score of those, more than the threshold:
{mean(all_scores_more_than_threshold)}")
# Release video capture and video writer objects
cap.release()
out.release()
cv2.destroyAllWindows()
print(f"Time for execution: {time.time() - start_time}")
```

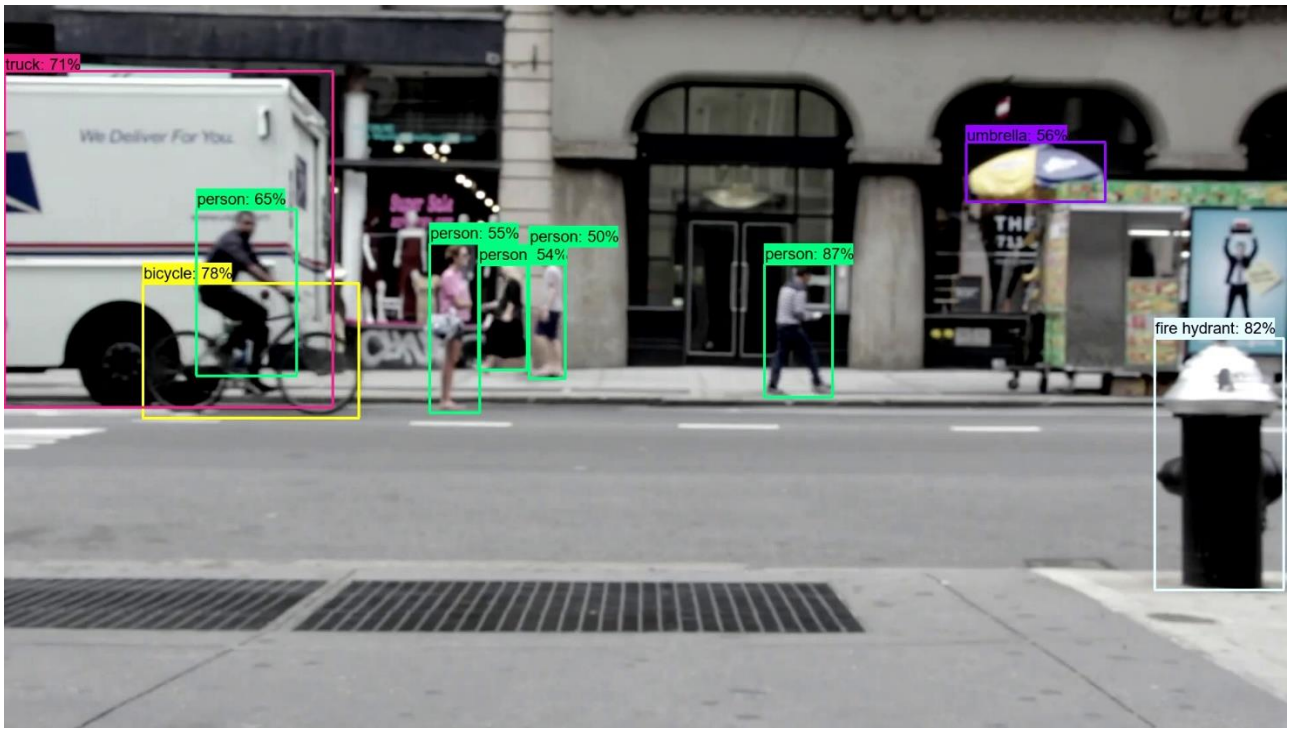
Додаток Б (обов’язковий) Приклад роботи програми: попередньо завантажене відео, EfficientDet D7 модель

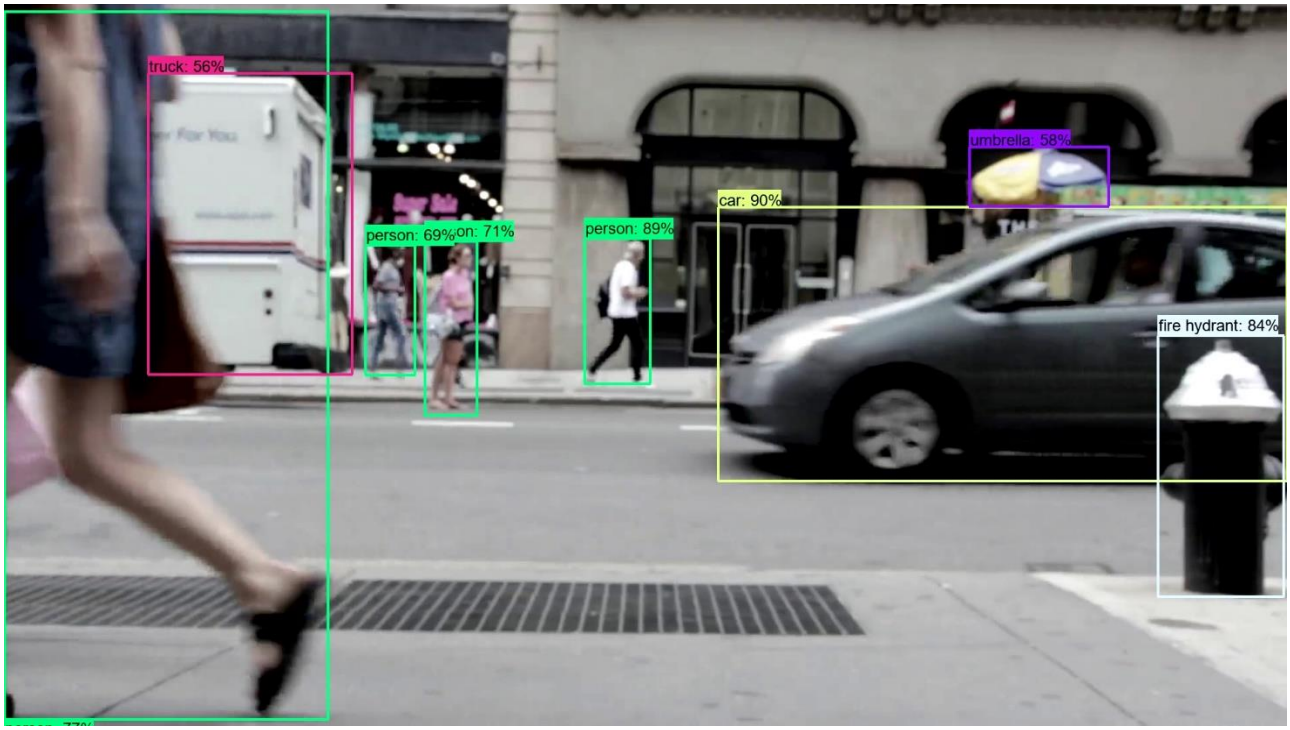


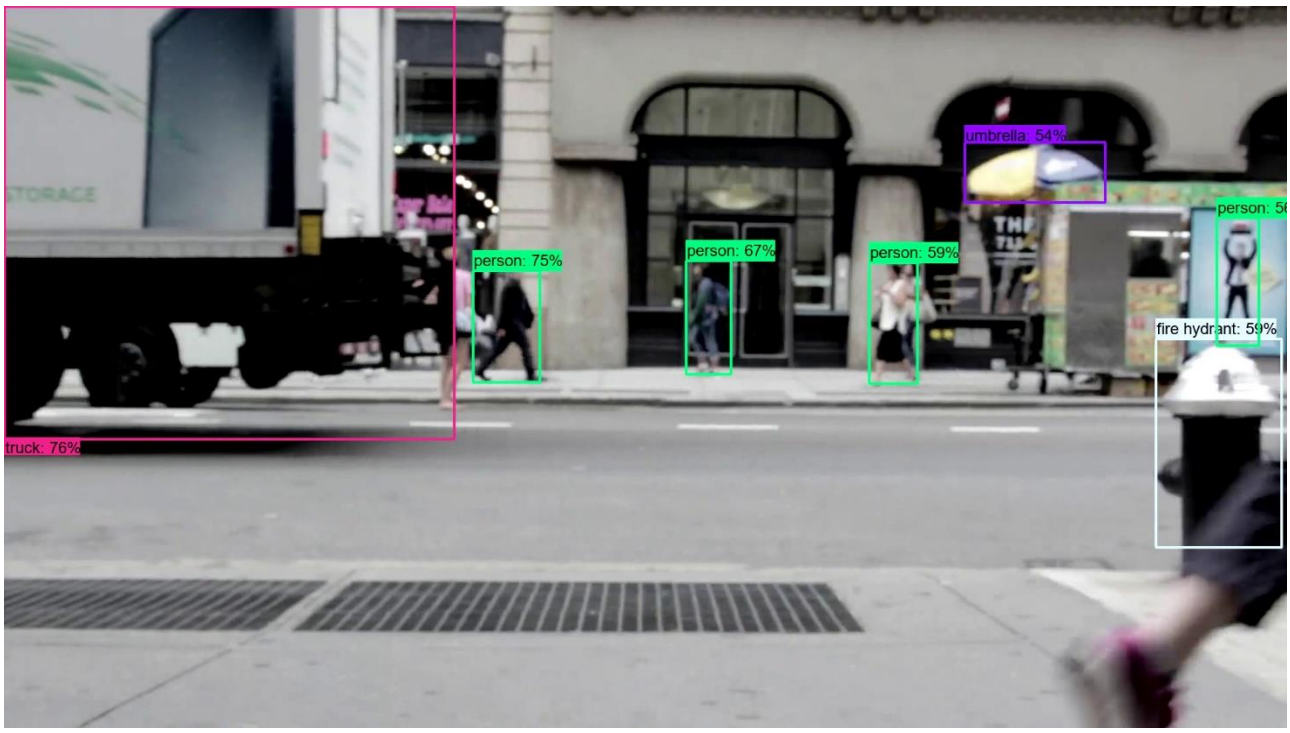



truck: 77%
car: 76%
umbrella: 50%
person: 65%
person: 64%
fire hydrant: 88%

Додаток В (обов’язковий) Приклад роботи програми: попередньо завантажене відео, EfficientDet-Lite4 модель






umbrella: 54%
person: 5
person: 75%
person: 67%
person: 59%
fire hydrant: 59%
truck: 76%