КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

ПОШУК ТА ВІДСТЕЖУВАННЯ ОБ'ЄКТІВ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент 4-го курсу
Владислав МАСЕХНОВИЧ


(підпис)

Науковий керівник:
професор кафедри теоретичної кібернетики
доктор фіз.-мат. наук, професор
Анатолій ПАШКО

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент


(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики
«01» червня 2022 р.,
протокол № 11
Завідувач кафедри,
доктор фіз.-мат. наук, професор

Юрій КРАК

(підпис)

КИЇВ – 2022

РЕФЕРАТ

Обсяг роботи 51 сторінка, 26 ілюстрацій, 34 джерел посилань.

КОМП’ЮТЕРНИЙ ЗІР, МАШИНЕ НАВЧАННЯ, НЕЙРОНА МЕРЕЖА, ПОШУК ТА ВІДСТЕЖУВАННЯ, СИСТЕМИ СТЕЖЕН

Об’єктом роботи є процес пошуку та відстежування об’єктів за допомогою штучного інтелекту. Предметом роботи є програмний засіб для є пошук та відстежування об’єктів на зображенні та відеопотоці.

Метою роботи є аналіз алгоритмів пошуку та відстежування та створення програмного засобу, який шукає та відстежує об’єкти на зображенні та відеопотоці.

Інструменти розробки: безкоштовне, вільно поширюване інтегроване середовище розробки PyCharm IDE Community Edition 2022.1.1, мова програмування Python 3.7.6.

Результати роботи: виконано загальний огляд історії штучного інтелекту та його напрямків, проаналізовано алгоритми пошуку та відстежування об’єктів, які використовуються для роботи зі штучним інтелектом, розглянуто існуючі програмні рішення з пошуку та відстежування об’єктів, розроблено додаток, в якому реалізовано робота алгоритмів пошуку та відстежування об’єктів на зображенні та відеопотоці.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ 5
ВСТУП 6
РОЗДІЛ 1. ПОНЯТТЯ ШТУЧНОГО ІНТЕЛЕКТУ ТА ОБЛАСТІ ЙОГО ЗАСТОСУВАННЯ 8
1.1 Розпізнавання мови 11
1.2 Обробка природної мови 13
1.3 Ігри 13
1.4 Робототехніка 14
1.5 Експертні системи 15
1.6 Комп’ютерний зір 16
РОЗДІЛ 2. ПОШУК ТА ВІДСТЕЖУВАННЯ ОБ’ЄКТІВ 17
2.1 Аналіз алгоритмів пошуку об’єктів 17
2.1.1 Метод Віоли-Джонса 18
2.1.2 R-CNN 21
2.1.3 YOLO 23
2.1.4 SSD 24
2.2 Аналіз алгоритмів відстежування об’єктів 25
2.2.1 Predator 26
2.2.2 GOTURN 28
РОЗДІЛ 3. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ 31
3.1 ART 32
3.2 SOLOSHOT3 33
3.3 Vision4ce 34
РОЗДІЛ 4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ПОШУКУ ТА ВІДСТЕЖУВАННЯ ОБ’ЄКТІВ 35
4.1 Підготовка програмного середовища 36
4.2 Обробка зображення за допомогою ImageAI та RetinaNet 37

4.3 Обробка відеопотоку за допомогою YOLO 43
ВИСНОВКИ 47
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 48

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API – application programming interface, прикладний програмний інтерфейс;

IDE – integrated development environment, інтегроване середовище розробки;

НМ – нейрона мережа;

ШІ – штучний інтелект.

ВСТУП

Оцінка сучасного стану об’єкта розробки. На даний момент штучний інтелект є невід’ємним атрибутом сучасного світу, який керується за допомогою технологій та даних. Він інтенсивно застосовується в таких областях, як пошукові системи, розпізнавання образів, робототехніка і, звісно ж, пошук та відстеження об’єктів на фото та відеопотоці в реальному часі.

Завдання відстеження об'єктів на відео є одним із найцікавіших завдань в інформаційних технологіях. На перший погляд, відеопотік можна розглядати як послідовність окремих кадрів, тому застосовуються багато алгоритмів, що використовуються для обробки звичайних зображень.

Актуальність роботи та підстави для її виконання. Останнім часом демонструється стрімкий ріст обсягів цифрової інформації та поширення систем, які зчитують та аналізують цю інформацію. Створення нових систем розпізнавання та відстеження об’єктів є актуальною. Наприклад, у випадку безпеки та контролю об’єктів, що охороняються, через систему відеокамер. Технології, які при цьому використовуються, розвиваються дуже швидко, адже області їх застосування не обмежуються лише вище наведеним прикладом – практика використання технологій розпізнавання перманентно розширюється.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення програмного засобу для пошуку та відстеження об’єктів на фото та відеопотоці. Для досягнення цієї мети поставлено такі завдання.

- Дослідити поняття штучного інтелекту.
- Дослідити існуючі алгоритми пошуку та відстеження об’єктів.
- Дослідити існуючі програмні рішення для вирішення задачі.
- Реалізувати роботу алгоритму для пошуку та відстеження об’єктів на фото та відеопотоці.

Об’єкт, методи й засоби розроблення. Об’єктом роботи у нас виступають зображення та відео. В якості методів розроблення було застосовано загальні алгоритми пошуку та відстежування об’єктів. У якості інструменту створення додатку було обрано PyCharm - інтегроване середовище розробки мовою Python. Також були використані наступні бібліотеки мови програмування Python:

- OpenCV;
- TensorFlow;
- NumPy;
- SciPy;
- Pillow;
- Matplotlib;
- H5py;
- Keras;
- ImageAI.

Можливі сфери застосування. Розроблений додаток є більше експериментальною моделлю для обробки зображення та відеопотоків, але при подальшому удосконаленні може застосовуватись у системах стеження.

РОЗДІЛ 1. ПОНЯТТЯ ШТУЧНОГО ІНТЕЛЕКТУ ТА ОБЛАСТІ ЙОГО ЗАСТОСУВАННЯ

Штучний інтелект — це інтелект, який демонструють машини, на відміну від природного інтелекту, який демонструють люди та тварини. ШІ дозволяє наділяти машинам можливості, які імітують інтелектуальну поведінку людини та його здатність міркування. ШІ має багато спільного з інтелектуальними програмами, які контролюють поведінку машин. Наука про штучний інтелект розробляє теорії та методи, які дозволяють машинам аналізувати навколишнє середовище та реагувати на різні ситуації так, як на них реагувала би людина [1].

Історія штучного інтелекту почалася з одного припущення від французького математика та філософа Рене Декарта, яке він зробив на початку XVII століття. Він допустив, що тварина являє собою деякий складний механізм. На основі цього твердження, він сформулював механічну теорію, який дав старт для створення перших механічних цифрових обчислювальних машин. У 1623 році німецький математик Вільгельм Шиккард став першим, хто створив обчислювальну машину. Згодом було створено машини Блеза Паскаля у 1643 році та Готфріда Лейбніца у 1671 році.

В 1943 році Воррен Маккалох і Вальтер Піттс опублікували статтю A Logical Calculus of the Ideas Immanent in Nervous Activity, тим самим поклавши основи нейроних мереж.

Вперше алгоритми ШІ з'явилися в 1960-х роках. Пристрої, попередньо запрограмовані для найпростіших міркувань, породили ранні платформи для створення цілих експертних і кваліфікованих прогностичних систем. І, не дивлячись на те, що на початкових етапах роботи з такими системами вчені зіштовхнулися з низкою проблем, які, на перший погляд, було неможливо вирішити, — результати численних досліджень принесли свої плоди [2].

Як показує детальний аналіз направлень розвитку науки про ШІ, в своїх спробах дати визначення штучному інтелекту вченні застосовували різні методи та підходи. В сучасності штучний інтелект застосовується в багатьох областях, приймаючи різні форми.

Праці в області ШІ тісно пов’язані з вивченням властивостей людського мозку. Дослідники припускають, що розуміння принципів роботи мозку зробить створення ШІ цілком здійсненним завданням. Імітуючи процеси, які відбуваються в людському мозку в процесі навчання, мислення й прийняття рішень, можна створити машину, яка буде здатна повторити ці процеси. Така машина стане платформою для створення інтелектуальних систем, здатних до навчання.

Успіхи в створенні ШІ здатні вплинути на всі аспекти людського життя. Дослідження в цій області направленні на вивчення властивостей та закономірностей поведінки об’єктів.

На рис 1.1 можна побачити процес обробки інформації нашим мозком.

Рисунок 1.1 – Процес обробки інформації людським мозком.

В порівняні з іншими областями науки, такими як математика або фізика, які існують століттями, наука про ШІ відносно молода. За останні два десятиліття вона продемонструвала значні досягнення, прикладами яких можуть бути безпілотні автомобілі та роботи. Вже досягнуті результати в області ШІ здатні докорінно змінити наше життя.

Можна лише дивуватися тому, як наш мозок здатний справлятися з обробкою великої кількості інформації з мінімальними зусиллями: здатність розуміти мову, розпізнавати образи, вчитися новому та виконувати різноманітні складні задачі. Прагнучи зробити те саме за допомогою машин, можна зрозуміти, що за рівнем розвитку вони залишаються далеко позаду.

На рис 1.2 зображено окремі етапи обробки вихідних даних в знання у вигляді схеми.

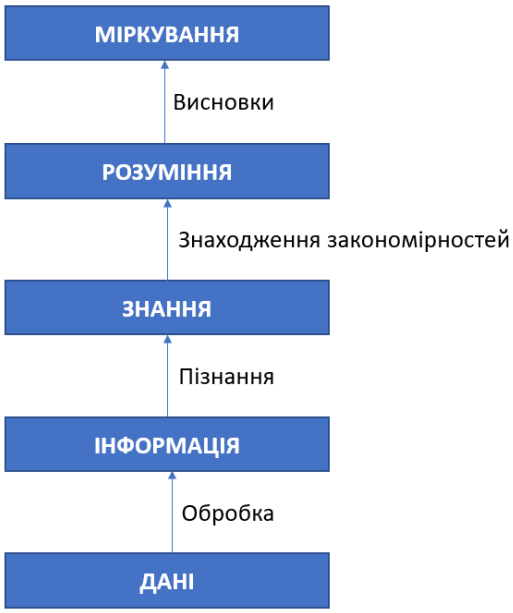

Рисунок 1.2 – Процес обробки вихідних даних в знання.

Одна з основних причин нашого прагнення до вивчення штучного інтелекту – можливість автоматизації багатьох процесів, що мають поліпшити життя людей.

Не дивлячись на те, що людський мозок виявляє чудові здібності в відношенні аналізу середовища, його здібностей недостатньо для вирішення таких проблем, як структуризація хаотичних даних, аналіз величезних об’ємів даних, з обробкою яких людський мозок просто не в змозі впоратись та т.п. Відповідно, вченні змушені розробляти інтелектуальні системи, які дозволять здолати ці обмеження. Людям потрібні системи ШІ, які могли би:

а) ефективно обробляти величезні об’єми даних;
б) отримувати дані одночасно з декількох джерел без затримок;
в) навчатися на основі отриманих даних та постійно оновлювати отримані знання;
г) приймати рішення на основі обставин, що виникають в реальному часі.

Методики ШІ активно використовуються для удосконалення існуючих машин з інтелектуальними системами, щоб вони могли більш ефективніше виконувати покладені на них завдання.

Тепер, коли відомо, як оброблюється інформація, можна розглянути області, в яких застосовується ШІ в реальному житті. Штучний інтелект може проявляти себе у різних формах, тому важливо розуміти, як саме він може бути корисним у тій чи іншій сфері людської діяльності. Розглянемо найбільш популярні сфери застосування, до яких відносяться:

а) Розпізнавання мови;
б) Обробка природної мови;
в) Ігри;
г) Робототехніка;
д) Експертні системи;
е) Комп’ютерний зір.

1.1 Розпізнавання мови

В наше життя технологія розпізнавання мови прийшла зовсім недавно. Ці системи здатні сприймати звукову інформацію й розуміти слова, що вимовляються. Наприклад, в наші смартфони та колонки вбудовані інтелектуальні персональні помічники, які розуміють наші голосові команди та реагують на них.

Ця система тримала ряд гарних відгуків від того, що пристрій може не просто зрозуміти, про що людина говорить, але і відповісти їй.

За статистикою, в середньому людина говоримо в 4-5 разів швидше, ніж пишемо. І звідси виникає питання: чому ж технологічні компанії тільки нещодавно почали створювати системи голосового керування пристроями? Насправді ідея створення машини для розпізнавання мови виникла приблизно в XVIII ст. Тоді-то і відбулися перші зрушення в напрямку розвитку технології. Саме в цей період відбулося головне досягнення – створення машини Фабера, яка базувалася на людській анатомії відтворення звуку. Ідея створення машини первинно належала австрійському вченому Вольфгангу фон Кемпелену. З початку машина здатна була промовляти ряд голосних та приголосних звуків, але не всіх. Потім професор Іосиф Фебер удосконалив та видозмінив механізм Кемпелена. Звісно ж, темпи технологічного розвитку тієї епохи бажали більшого, тому піти далі, ніж створення механічної машини було майже неможливо. Але, це стало основою для відкриттів, яким судилося з’явитися набагато пізніше. До цих відкриттів відноситься диктофон, який був створений Александром Грем Беллом та його асистентами на основі станіольного фонографа Томаса Едісона, яких змінили так, щоб він міг відтворювати звук з воску замість станіолі.

Технологія розпізнавання вперше стала реальністю тільки в 1952 році. Винахідники з компанії Bell Labs створили машину Audrey, яка розпізнавала цифри від 0 до 9.

У 1997 році була створена комп’ютерна програма Dragon's NaturallySpeaking. Вона відрізнялася тим, що могла безперервно розпізнавати до 100 слів за хвилину. Тобто, диктору не було необхідності робити паузи між словами.

Система машинного навчання зробила прорив в технології розпізнавання. Це привело нас до нового етапу розпізнавання мови, який зараз набирає все

більшої популярності: голосовим ботам, які стали фундаментом для створення повноцінних голосових помічників: Google Assistant, Siri, Cortana та ін [3].

1.2 Обробка природної мови

Системи цього типу призначенні для розпізнавання текстів, написаних на природніх мовах. Можна взаємодіяти з машиною, передаючи їй команди у вигляді текстових даних. Пошукові системи інтенсивно використовують цю технологію для доставки релевантних результатів пошуку користувачам.

В наш час значну роль у вирішенні задач з обробки природномовних даних відіграють онтології, наприклад, WordNet, UWN. У процесі дослідження обробки природної мови було досягнуто значних результатів, серед яких розробка потужних лексикографічних систем, програм для машинного перекладу, електронних словників та ін [4].

1.3 Ігри

Штучний інтелект широко використовується в індустрії ігор. Він використовується для проектування інтелектуальних агентів, які здатні змагатися в майстерності гри з людиною.

В якості прикладу можна привести AlphaGo – комп’ютерну програму, яка здатна грати в китайську стратегічну гру Go. В AlphaGo застосовано алгоритм пошуку Монте-Карло, керований за допомогою технології поглибленої НМ для оцінки позиції та пошуку найбільш вдалих ходів. Спочатку AlphaGo вчилась грати на записаних партіях професійних гравців, з яких вона вибирала свої ходи. Згодом, після досягнення певного рівня, вона почала грати проти себе самої, для подальшого вдосконалення.

У випадку індустрії комп’ютерних ігор реалізація ШІ сильно впливає на ігровий процес під час гри, системні вимоги та бюджет гри, і розробники балансують між цими вимогами, намагаючись зробити цікавий і невимогливий до ресурсів ШІ ігровий процес. Тому підхід до ігрового ШІ серйозно відрізняється від підходу до традиційного ШІ — широко застосовуються різного роду спрощення, обмани й маніпуляції. Наприклад, з одного боку, маємо гру в жанрі шутеру від першої особи, в якому безпомилковий рух та миттєве прицілювання, властиве ботам під керуванням штучного інтелекту, не залишає жодного шансу гравцеві, так що ці здатності штучно знижуються. З іншого боку – боти повинні робити засідки, діяти командою й т.д, інакше гра перетворюється на звичайний тир. Простіше кажучи, головне завдання ШІ в комп’ютерній відеогрі – не виграти в гравця, а красиво йому піддатися. Але це також залежить від рівня складності штучного інтелекту [5].

1.4 Робототехніка

Робототехнічні системи в дійсності об’єднують в собі багато концепцій штучного інтелекту. Ці системи здатні виконувати багато найрізноманітніших задач. В залежності від ситуації, роботи можуть бути обладнані датчиками та приводними елементами, що забезпечують виконання всіляких дій. Датчики можуть розпізнавати предмети, що знаходяться в їх полі зору, вимірювати температуру, реагувати на вчинені ними дії і т.п. Вмонтовані в електронні плати процесори виконують необхідні розрахунки в режимі реального часу. Окрім того, роботи можуть адаптувати свою поведінку до змін зовнішніх чинників.

Сьогодні широко розповсюджені комерційні і промислові роботи, що використовуються для виконання різної праці дешевше, точніше та надійніше за людей. Вони також застосовуються у деяких роботах, які занадто брудні, небезпечні або марудні, щоб бути придатними для людей.

1.5 Експертні системи

В експертних системах методики ШІ використовуються для прийняття рішень або представлення відповідних рекомендацій. Як правило, в таких областях, як фінанси, медицина, маркетинг та інші. Для цієї цілі використовують бази знань. На рис 1.3 ілюструється, що являє собою експертна система і як вона взаємодіє з користувачем.

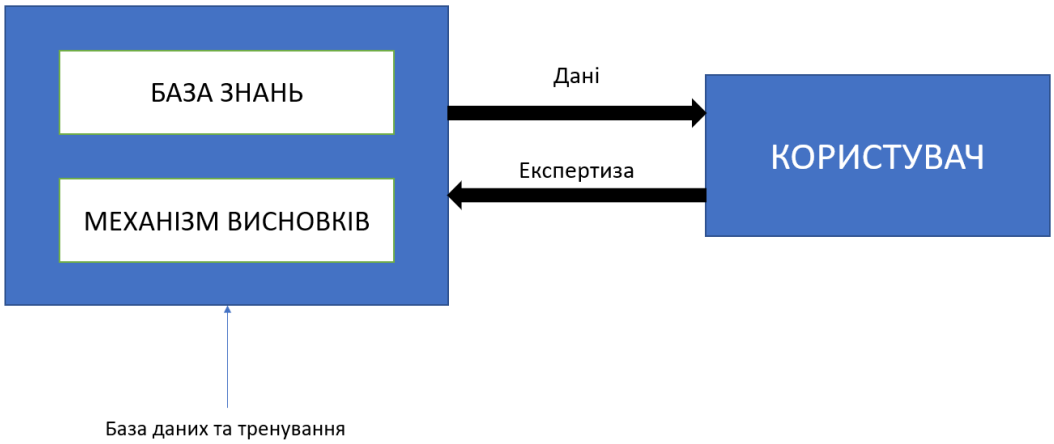

Рисунок 1.3 – Експертна система та її взаємодія з користувачем.

Однією з основних характеристик експертної системи є її швидкодія, тобто швидкість отримання результату та його достовірність. Дослідницькі програми штучного інтелекту можуть бути і не дуже швидкими, натомість, експертна система повинна за прийнятний час знайти розв'язок, що був би не гіршим за розв'язок, що може запропонувати фахівець в цій предметній області.

Експертна система повинна мати можливість пояснити, чому запропоновано саме цей розв'язок і довести його обґрунтованість. Користувач повинен отримати всю інформацію, необхідну йому для того, аби переконатись в обґрунтованості запропонованого розв'язку [6].

До відомих експертних систем можна віднести мови програмування, що використовуються для створення експертних систем CLIPS та Prolog, систему діагностики інфекційних хвороб крові Mycin, систему аналізу даних мас-спектрометрії Dendral та ін.

1.6 Комп’ютерний зір

Комп’ютерний зір можна охарактеризовати як молоду та різноманітну область, в якій застосовується ШІ. І, хоча існують більш ранні роботи, можна сказати, що тільки з кінця 1970-х почалось інтенсивне вивчення цієї проблеми, коли комп'ютери змогли керувати обробкою великих наборів даних, таких як зображення та відео.

Системи комп’ютерного зору аналізують вміст зображень та вилучають корисну інформацію на основі представлених типових зразків. Наприклад, Google використовує технологію реверсивного пошуку зображень для знаходження візуально подібних зображень в Інтернеті.

Як технологічна дисципліна комп'ютерний зір прагне застосувати теорії та моделі комп'ютерного зору до створення систем комп'ютерного зору. Прикладами таких систем можуть бути [7]:

а) системи керування процесами;

б) системи відеоспостереження;

в) системи організації інформації;

г) системи моделювання об'єктів або навколишнього середовища;

д) системи взаємодії.

РОЗДІЛ 2. ПОШУК ТА ВІДСТЕЖУВАННЯ ОБ’ЄКТІВ

Пошук та відстежування візуальних об'єктів (рисунок 2.1) є важливою темою дослідження комп'ютерного зору, розуміння зображень і розпізнавання образів. Враховуючи початковий стан (розташування центру та масштаб) цілі в першому кадрі відеоряду, метою відстеження візуального об’єкта є автоматичне отримання станів об’єкта в наступних відеокадрах.

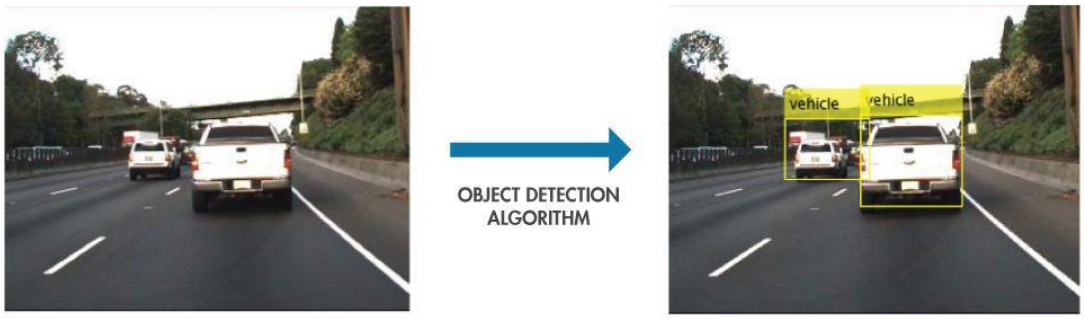

Рисунок 2.1 – Приклад пошуку об’єктів на зображенні.

У цьому розділі буде розглянуто відомі алгоритми пошуку та відстежування об’єктів.

2.1 Аналіз алгоритмів пошуку об’єктів

Якщо згадувати усі можливі алгоритми пошук об’єктів, то перше, що спадає на думку – це метод Віоли-Джонса. Цей алгоритм відрізняється своєю простотою, завдяки чому він до сих пір активно використовується для розпізнавання об’єктів на зображенні, але головним недоліком є відсутність стійкості до зовнішніх умов. Також великого успіху в цій області досягли згорткові НМ. До алгоритмів згорткових НМ, що використовуються для пошуку обвідносяться:

a) R-CNN (Region-based Convolutional Neural Networks);

1) Fast R-CNN;

2) Faster R-CNN;

б) HOG (Histogram of Oriented Gradients);

в) R-FCN (Region-based Fully Convolutional Network);

г) SSD (Single Shot Detector);

д) SPP-Net (Spartan Pyramid Pooling);

е) YOLO (You Only Look Once).

Більш детально буде розглянуто YOLO, SSD та R-CNN, але спочатку повернемось до методу Віоли-Джонса.

2.1.1 Метод Віоли-Джонса

Не зважаючи на те, що метод був розроблений та представлений Паулом Віолою та Майклом Джонсом в 2001 році, він до сих пір активно використовується для пошуку об’єктів на зображенні у реальному часі. Хоча алгоритм може розпізнавати об’єкти, основною його задачею при створенні було розпізнавання облич. Слід також зазначити, що цей алгоритм має вкрай низьку вірогідність помилкового знаходження обличчя. Метод також гарно знаходить обличчя навіть при спостереженні об’єкта під невеликим кутом, приблизно 30°. Якщо ж кут буде більшим за 30°, то ефективність різко знижується. Вказана особливість методу не дозволяє знайти обличчя людини, повернуте під довільним кутом, що ускладнює використання алгоритму в сучасних системах [8].

Алгоритм має чотири етапи:

а) Вибір ознаки Хаара;

б) Створення інтегрального зображення;

в) Машинне навчання з використанням алгоритму AdaBoost;

г) Каскадні класифікатори.

Ознаки Хаара — ознаки цифрового зображення, які використовуються для розпізнавання образів. Своєю назвою вони зобов'язані інтуїтивною подібністю до вейвлетів Хаара. Ознаки Хаара використовувалися у першому детекторі осіб, що

працював у реальному часі [9]. Однак ознаки, які схожі на ознаки Хаара та запропоновані Віолою та Джонсом, містять більше однієї прямокутної області та дещо складніші. На рис. 2.2 показано 4 різні типи ознак.

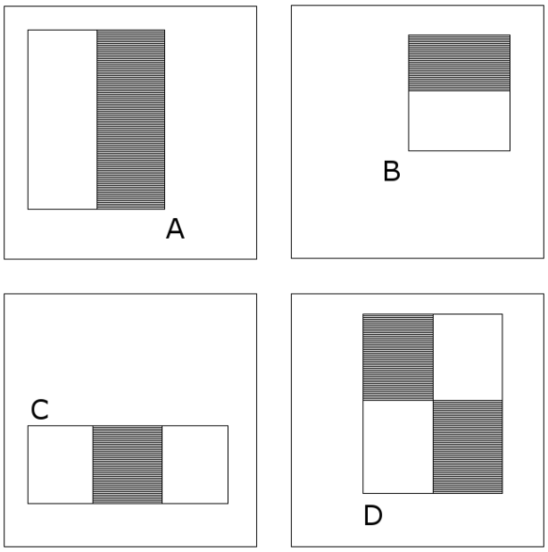

Рисунок 2.2 – Прямокутні ознаки Хаара.

Величина кожної ознаки обчислюється як сума пікселів у білих прямокутниках, з якої віднімається сума пікселів у чорних областях. Прямокутні ознаки примітивні, і, незважаючи на те, що вони чутливі до вертикальних і горизонтальних особливостей зображень, результат їх пошуку більш грубий.

Однак при зберіганні зображення в інтегральному форматі (рисунок 2.3) перевірка прямокутної ознаки на конкретній позиції проводиться за константний час, що є їх перевагою порівняно з більш точними варіантами.

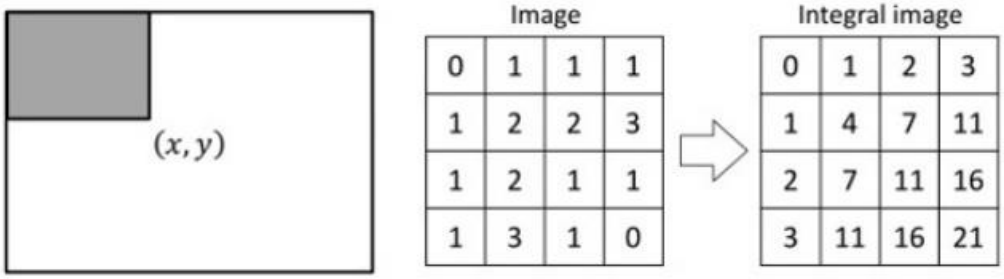

Рисунок 2.3 – Інтегральне представлення зображення.

Кожна прямокутна область в ознаках завжди суміжна з іншим прямокутником, тому розрахунок ознаки з 2 прямокутниками складається з 6

звернень в інтегральний масив, для ознаки з 3 прямокутниками - з 8, з 4 прямокутниками - з 9.

Висока швидкість розрахунку ознаки не компенсує значну кількість різних можливих ознак. Наприклад, при стандартному розмірі ознаки 24×24 пікселя можливо 162 тисячі різних ознак, та його розрахунок може зайняти багато часу. Тому в алгоритмі Віоли-Джонса використовується варіація алгоритму навчання AdaBoost як для вибору ознак, так і для налаштування класифікаторів шляхом створення каскаду (рисунок 2.4).

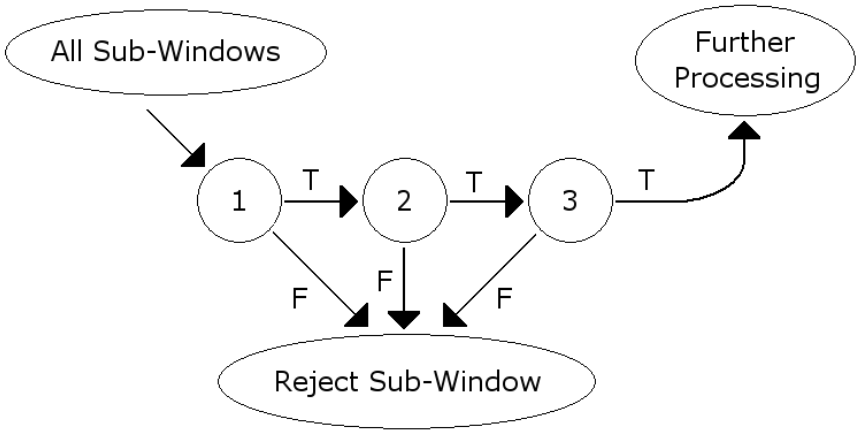

Рисунок 2.4 – Архітектура каскада.

Отже, алгоритм AdaBoost має багато переваг:

а) Гарна узагальнююча здатність, яка може поліпшуватись у міру збільшення числа базових алгоритмів;
б) Простота реалізації;
в) Ідентифікація шумових викидів;
г) Невеликі витрати при виконані процедури послідовної побудови композиції алгоритмів машинного навчання.

2.1.2 R-CNN

Одним з перших підходів, що застосовуються для визначення розташування об'єкта на зображенні, є R-CNN. Ця архітектура була створенна Россом Гіршиком, Джеффом Донахью, Тревором Даррелом та Джитендрою Маліком [10]. Вона складається з декількох кроків, що послідовно виконуються, та проілюстрована на рис 2.5:

1) Визначення набору гіпотез;
2) Вилучення з передбачуваних регіонів ознак за допомогою згорткової НМ та їх кодування у вектор;
3) Класифікація об'єкта всередині гіпотези на основі вектора з кроку 2;
4) Поліпшення (корегування) координат гіпотези;
5) Все повторюється, починаючи з кроку 2, доки не будуть оброблені всі гіпотези з кроку 1.

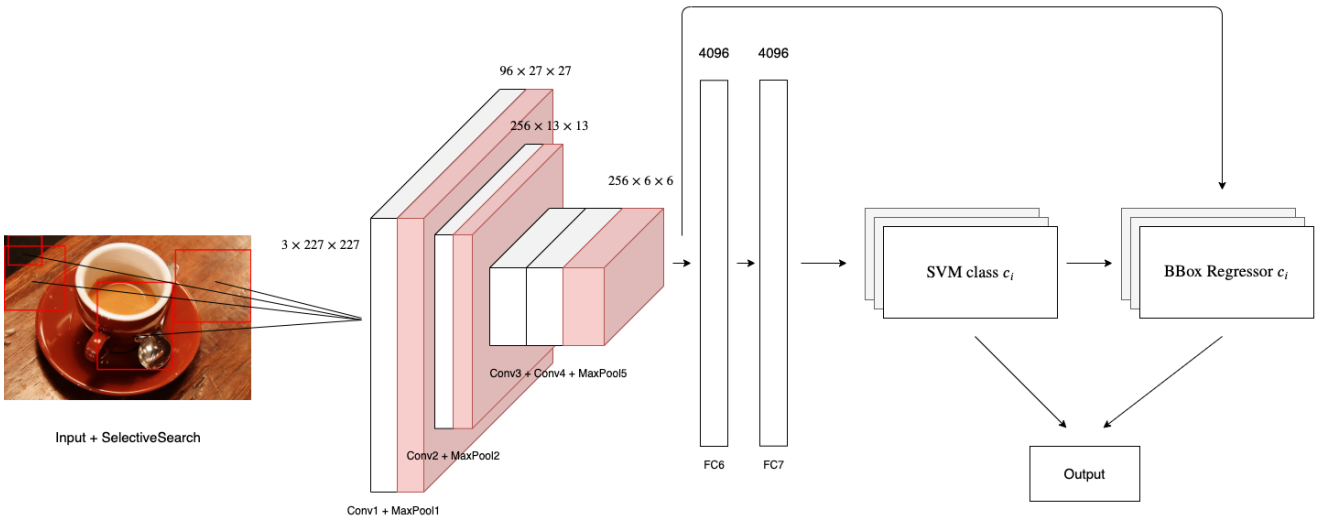

Рисунок 2.5 – Архітектура R-CNN.

Але ця архітектура має певні недоліки:

а) Гіпотези, запропоновані на першому кроці, можуть частково дублювати один-одного – різні гіпотези можуть складатися з однакових частин, а кожна така гіпотеза може окремо оброблятися

нейронною мережею. Виходить, що більша частина запусків мережі більш-менш дублює один-одного без необхідності;

б) Архітектуру не можна використовувати в режимі реального часу;

в) Алгоритм виводу гіпотез ніяк не може навчатися. Тому удосконалення якості є майже неможливим.

Оскільки R-CNN є повільною та не дуже ефективною системою, досить швидко тими самими авторами було запропоновано покращення у вигляді мережі Fast R-CNN [11]. Процес обробки зображення змінився в порівняні з оригінальним алгоритмом і виглядає таким чином:

1) Вилучення мапи ознак зображення (не для кожної гіпотези окремо, а для всього зображення);
2) Пошук гіпотез (аналогічно R-CNN);
3) Порівняння кожної гіпотези з місцем на мапі ознак. Тобто використання єдиного набору виділених ознак для кожної гіпотези;
4) Класифікація кожної гіпотези та виправлення координат, що обмежують рамки.

Обидві вищенаведені моделі використовують алгоритм вибіркового пошуку (selective search) та були залежні від нього. Він дозволяє скласти набір та на основі сегментації визначити межі об'єктів за інтенсивністю пікселів, перепаду кольорів, контрасту та текстур. Вибірковий пошук є повільним та трудомістким процесом, який впливає на продуктивність мережі. Тому Шаокінг Рен розробив алгоритм виявлення об’єктів, який усуває алгоритм вибіркового пошуку та дозволяє мережі вивчати пропозиції регіонів (region proposal network, RPN). Цей алгоритм став основою для створення Faster R-CNN [12].

На рис 2.6 можна побачити порівняння швидкості тестування вищенаведених алгоритмів пошуку об’єктів разом з алгоритмом SPP-Net.

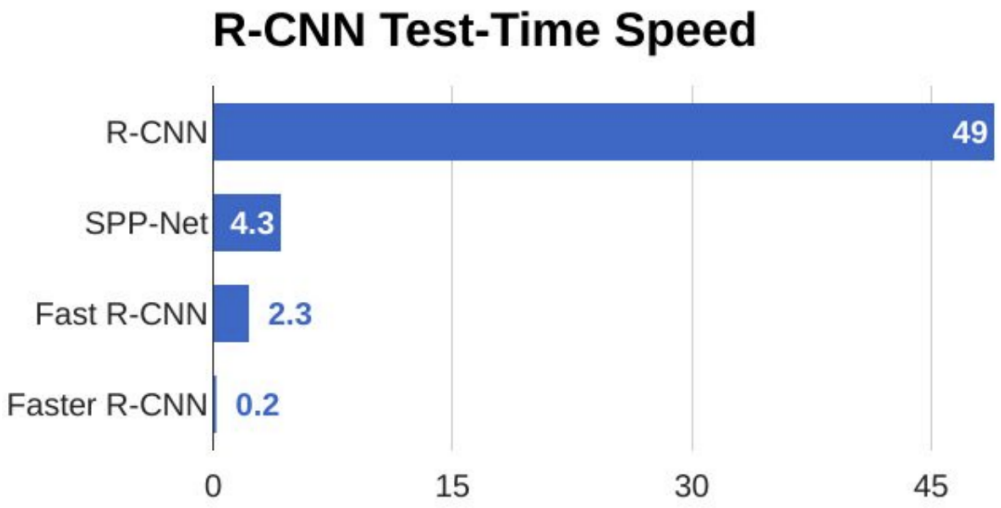

Рисунок 2.5 – Архітектура R-CNN.

Можна зрозуміти, що покращення алгоритму R-CNN до Faster R-CNN зменшило час виконання обробки зображення майже в 250 разів. Також, судячи з останніх відгуків можна сказати, що Faster-CNN є однією з найточніших і досі.

2.1.3 YOLO

YOLO – сучасний алгоритм, що широко використовується для виявлення об'єктів. Він був розроблений Джозефом Редмоном та Алі Фархаді у 2016 році.

Основна відмінність YOLO від інших алгоритмів згорткової нейронної мережі, що використовуються для виявлення об'єктів, полягає в тому, що він дуже швидко розпізнає об'єкти в режимі реального часу. Принцип роботи YOLO має на увазі введення відразу всього зображення, яке проходить через згорткову нейронну мережу лише один раз. В інших алгоритмах цей процес відбувається багаторазово, тобто зображення проходить через мережу знову і знову. Так що YOLO має перевагу високошвидкісного виявлення об'єктів (рисунок 2.6).

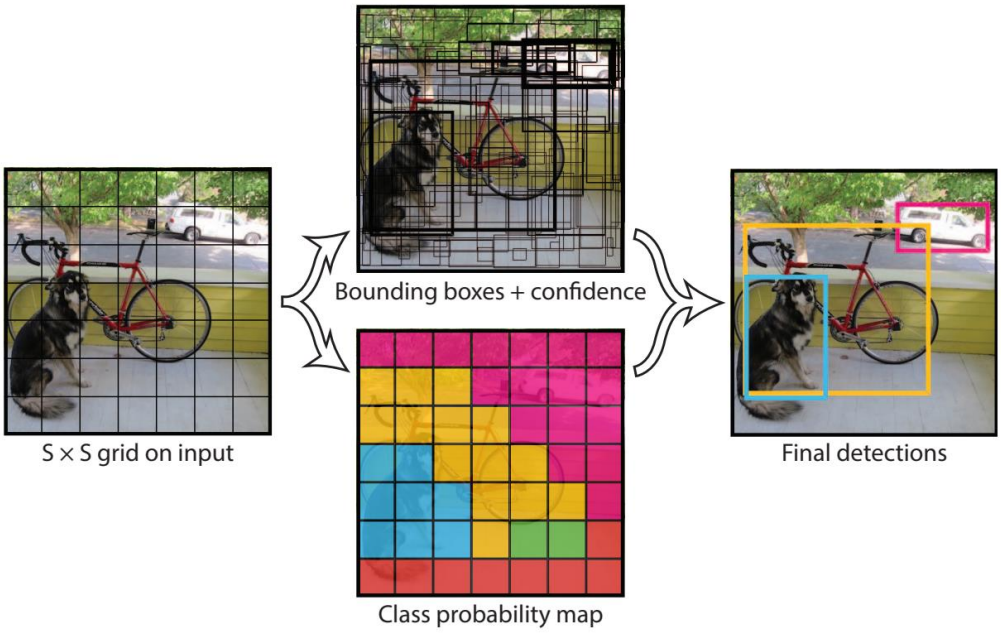

Рисунок 2.6 – Алгоритм YOLO.

Алгоритм YOLO був навчений на певному типі набору даних, що складається з 80 різних типів класів. Алгоритм YOLO здатний виявляти всі ці 90 видів об'єктів на зображенні. Він також може бути навчений, щоб легко знаходити нові об'єкти.

Про роботу цього алгоритму більш конкретно поговоримо трохи пізніше, під час створення програми на його основі.

2.1.4 SSD

Модель SSD (рисунок 2.7) використовує ідею використання пірамідальної ієрархії виходів згорткової мережі для виявлення об'єктів різних розмірів. Зображення послідовно передається на шари згорткової мережі, які зменшуються в розмірах. Вихід із останнього шару кожної розмірності бере участь у прийнятті рішення щодо розпізнавання об'єктів, таким чином, складається "пірамідальна характеристика" зображення. Це дозволяє виявляти об'єкти різних масштабів,

оскільки розмірність виходів перших шарів сильно корелює з рамками, що обмежують, для маленьких об'єктів, а останніх — для великих.

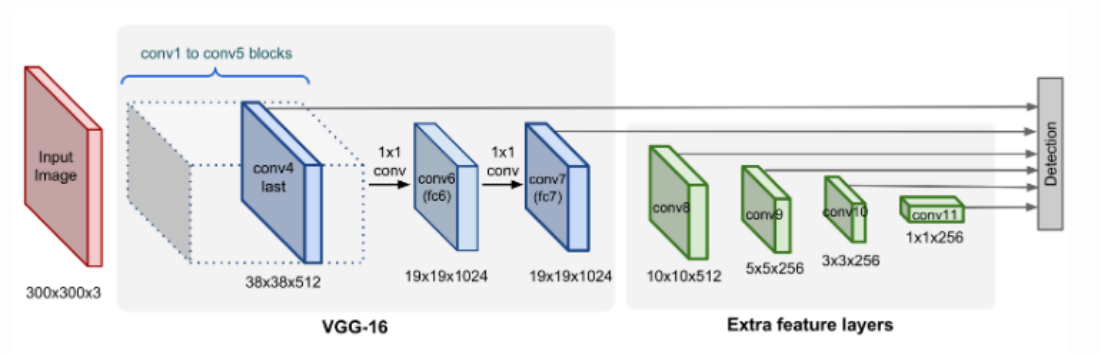

Рисунок 2.7 – Архітектура нейронної мережі для алгоритму SSD

SSD передбачає зміщення ключових рамок. Ключові рамки на різних рівнях масштабуються так, що одна розмірність вихідного шару відповідає за об'єкти свого масштабу. Таким чином, великі об'єкти можуть бути виявлені лише на вищому рівні, а маленькі об'єкти – на низьких рівнях [13].

2.2 Аналіз алгоритмів для відстежування об’єктів

Алгоритми пошуку об’єктів підходять по більшій частині для пошуку на зображеннях, оскільки ці алгоритми знаходять лише одноразово. Тому щоб знайти потрібний об’єкт на наборі кадрів чи відеопотоці потрібно застосовувати цей алгоритм на кожному кадрі, що не завжди є ефективним.

Тому для вирішення цієї проблеми було розроблено алгоритми трекінгу (відстежування) об’єктів. Як саме це працює? Спочатку завантажується алгоритм пошуку для розрахунку координат об’єкта з початкового кадру. Далі, на наступному кадрі по координатам з попереднього кадру робиться обчислення координат об’єкта, і так ми просуваємось до останнього кадру.

Перед тим як перейти до аналізу існуючих алгоритмів відстежування, слід згадати про одну з найбільших бібліотек функцій та алгоритмів комп’ютерного зору – OpenCV.

Проект OpenCV офіційно був запущений компанією Intel Research у 1999 році. На перших етапах розвитку бібліотеки головними цілями були:

а) Розвивати дослідження у напрямку комп'ютерного зору;
б) Поширювати знання у сфері комп'ютерного зору, забезпечуючи загальну інфраструктуру, яку б могли розвивати розробники;
в) Розвивати засновані на роботі з комп'ютерним зором комерційні додатки.

На даний момент бібліотека OpenCV містить в собі 2500 оптимізованих алгоритмів, серед яких є алгоритми відстежування об’єктів [14]. Бібліотека OpenCV містить реалізацію 8 основних алгоритмів для відстеження об’єктів:

а) MEDIAFLOW;
б) MIL;
в) BOOSTING;
г) TLD;
д) MOSSE;
е) Predator;
ж) GOTURN;

Більш детально поговоримо про останні два алгоритми – Predator та GOTURN, оскільки, судячи з останніх відгуків, вони є найбільш ефективними для виконання завдань пов’язаних з трекінгом.

2.2.1 Predator

TLD (Tracking-Learning Detection) – це нова архітектура відстежування, яка чітко розкладає довгострокове завдання відстежування на відстеження, навчання та виявлення. Трекер слідкує за об’єктом від кадру до кадру. Детектор відстежує усі появи об’єкта, які спостерігалися на даний момент, і при необхідності коригує трекер. Завдяки навчанню алгоритм оцінює помилку детектора та оновлює його, щоб уникнути цих помилок у майбутньому. Алгоритм також використовує метод

машинного навчання P-N (мета якого – підвищення продуктивності детектора об'єкту, при онлайн обробці відеопотоку) для виявлення помилок детекторів і навчання на них (рисунок 2.8).

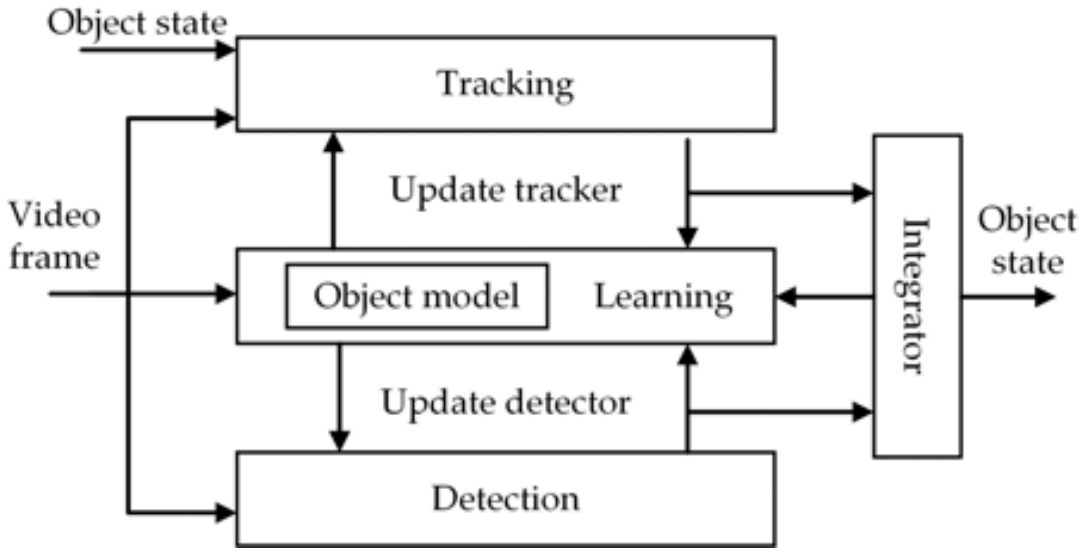

Рисунок 2.8 – Блок-схема трекера на основі TLD.

Завдяки своїм здібностям до навчання TLD рекламується під назвою Predator. TLD був розроблений Зденеком Калалом під час його докторської дисертації під керівництвом Крістіана Миколайчика та Іржі Матаса в університеті Суррея.

До алгоритму TLD найсучасніші методи відстеження об’єктів виконували відстеження за виявленням, тобто детектор прогнозує положення об’єкта та одночасно адаптує його параметр до зовнішнього вигляду об’єкта. Цей метод добре працює, якщо немає оклюзій, тобто ситуацій, коли два об'єкти розташовані приблизно на одній лінії і один об'єкт, розташований ближче до камери, частково або повністю закриває видимість іншого об'єкта. Але нажаль таке дуже часто відбувається.

TLD долає цю проблему, навчаючи детектор за допомогою прикладів, знайдених на траєкторії трекера, яка не залежить від детектора. Завдяки розділенню задач відстеження та виявлення досягається висока надійність і ефективність в порівняні з попередніми методи відстеження за виявленням.

Детектор використовує каскадний підхід для прискорення обчислень і використовує прості функції для виявлення об’єктів [15].

2.2.2 GOTURN

GOTURN (Generic Object Tracking Using Regression Networks) – це алгоритм відстежування об’єктів на основі глибинного навчання. Трекер GOTURN - єдиний з запропонованих алгоритмів, який працює на нейронних мережах, тому він має вищу точність ніж у конкурентів, а проста архітектура забезпечує високу швидкість виконання алгоритму.

Більшість алгоритмів відстеження навчаються онлайн. Іншими словами, алгоритм відстеження вивчає зовнішній вигляд об’єкта, який він відстежує під час виконання.

Тому багато трекерів у реальному часі покладаються на алгоритми онлайн-навчання, які зазвичай набагато швидші, ніж рішення на основі глибинного навчання.

GOTURN змінив спосіб застосування глибокого навчання до проблеми відстеження, вивчаючи рух об’єкта в автономному режимі. Модель GOTURN навчається на тисячах відеопослідовностей і не потребує навчання під час виконання.

GOTURN був представлений Девідом Хелдом, Себастьяном Трун та Сільвіо Саварезе у своїй статті під назвою “ Learning to Track at 100 FPS with Deep Regression Networks”.

Як показано на рис 2.9, GOTURN навчається за допомогою пари обрізаних кадрів із тисяч відео.

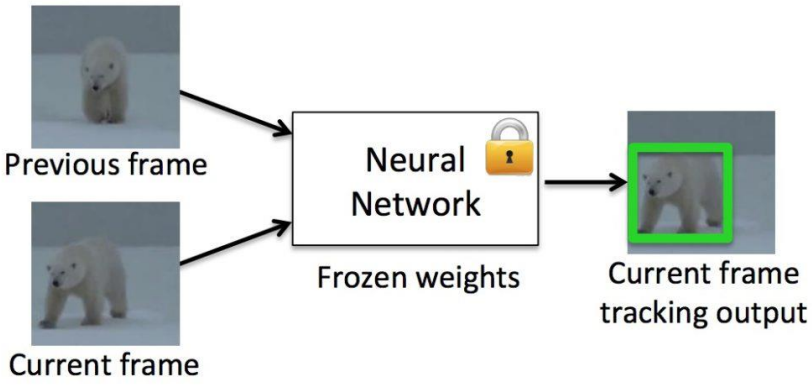

Рисунок 2.9 – GOTURN приймає два обрізаних кадри в якості вхідних даних та виводить обмежувальну рамку навколо об’єкта в другому кадрі.

У першому кадрі (який ще називається попереднім кадром) місце розташування об’єкта відоме, і кадр обрізається до розміру, який вдвічі більший за розмір обмежувальної рамки навколо об’єкта. Об’єкт у першому обрізаному кадрі завжди відцентрований.

Розташування об’єкта у другому кадрі (який ще називається поточним кадром) необхідно передбачити. Обмежувальна рамка, яка використовується для обрізання першого кадру, також використовується для обрізання другого кадру. Оскільки об’єкт міг рухатися, об’єкт не відцентрований у другому кадрі.

Ці дані подаються на вхід згортковій нейронній мережі, яка навчається за попереднім кадром передбачати місцезнаходження об’єкту на наступному кадрі.

Для простішого розуміння можна уявити згорткову нейрону мережу як чорну скриньку з багатьма ручками, які можна встановити на різні значення. Коли налаштування регуляторів (ручок) правильні, згорткова нейрона мережа створює правильний обмежувальний прямокутник. Спочатку налаштування ручок є випадковими. Під час навчання показуємо пару кадрів нейронній мережі, для яких відомо розташування об’єкта (тобто обмежувальні рамки). Якщо мережа робить помилку, налаштування ручок принципово змінюються за допомогою алгоритму зворотного поширення помилки. Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи багатошарового перцептрону та отримання бажаного виходу. Коли зміна налаштувань регуляторів перестає покращувати результати, говоримо, що модель навчена.

Тепер, коли було показано згорткову нейрону мережу як чорну скриньку, давайте подивимося, що всередині цієї скриньки.

На рис. 2.10 показана архітектура GOTURN. Як згадувалося раніше, він приймає два обрізаних кадри в якості вхідних даних. Зверніть увагу, попередній кадр, показаний внизу, відцентрований, і наша мета — знайти обмежувальну рамку для поточного кадру, показаного зверху.

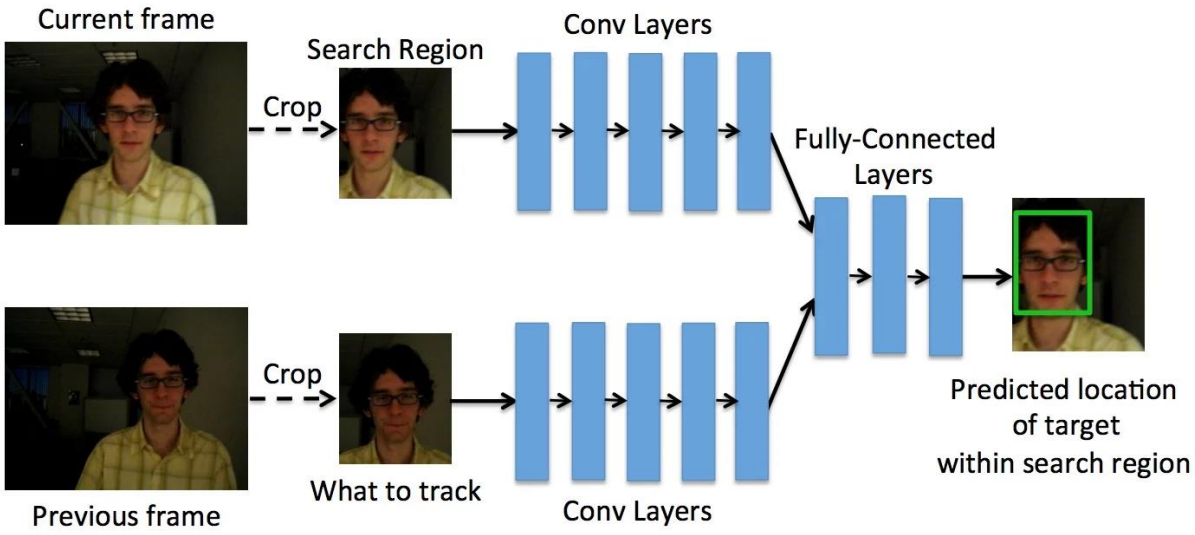

Рисунок 2.10 – Архітектура GOTURN.

Обидві рамки проходять через сховище згорткових шарів. Шари - це просто перші п'ять згорткових шарів архітектури CaffeNet. Вихідні дані цих згорткових шарів об’єднуються в один вектор довжиною 4096. Цей вектор вводиться в три повнозв’язних шари. Останній повністю підключений шар нарешті з’єднується з вихідним шаром, що містить 4 вузли, що представляють верхню і нижню точки обмежуючого прямокутника [16].

РОЗДІЛ 3. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Кожного дня розробники технічних компаній намагаються створювати або удосконалювати уже існуючи системи пошуку та відстежування, або створювати технічні продукти, робота яких базується на використані алгоритмів цих систем, оскільки вони є дуже затребуваними у сферах безпеки, дослідження космосу та ін.

Першість з використання подібних систем належить Китаю. Сьогодні за громадянами Китаю спостерігають 170 мільйонів вуличних відеокамер, і найближчими роками їх число зросте більш ніж втричі. Ці камери є частиною система тотального стеження (рисунок 3.1). Дані з камер використовують системи соціального кредиту. Система безперервно відстежуватиме та оцінюватиме діяльність громадян, додаючи бали до рейтингу за схвальну поведінку та віднімаючи за порушення. У результаті власники високих балів отримають соціальні та економічні пільги, тоді як низький рейтинг спричинятиме заборони та обмеження [17]. І ця система дає свої плоди в області тотального контролю.

Рисунок 3.1 – Китайська система стеження.

На жаль, більш конкретно не можна розібрати цю систему відстежування, оскільки вона є закритою, та не для використання в широкому доступі. Але можна розібрати три системи для пошуку та відстежування об’єктів у відеопотоці:

a) ART (Advanced Realtime Tracking);

1) SMARTTRACK;
2) TRACKPACK;
3) ARTTRACK;

б) SOLOSHOT3;

в) Vision4ce.

Слід також зазначити, що не всі ці системи розв’язують задачі пошуку та відстеження повністю. Вони сконцентровані лише на відстежуванні об’єктів в кадрі, а також не всі вони мають зручний інтерфейс для користувача.

3.1 ART

ART є одним з провідних виробників преміальних оптичних систем відстежування руху для віртуальної та доповненої реальності [18]. Типовими сферами застосування є візуалізація та оптимізація на етапах проектування та виробництва, захоплення руху для вивчення ергономіки, а також навчальні та дослідницькі проекти в академічних установах у яких використовуються Powerwall, CAVE та HMD.

Компанія, в свою чергу, пропонує одразу три системи відстежування об’єктів для використання: SMARTTRACK, TRACKPACK та ARTTRACK.

Система SMARTTRACK була розроблена та оптимізована під мобільні пристрої. Незважаючи на високу продуктивність та надійність, право на використання системи можна придбати має помірну ціну. Вона здатна знаходити об’єкти на кадрі та відстежувати їх місцезнаходження. Також система використовується в транспортних засобах для таких застосувань, як перевірка систем спостереження водіїв.

Система TRACKPACK розширює функціонал SMARTTRACK тим, що може знаходити невеликі об’єкти та класифікувати чи є об’єкт рухомим чи нерухомим. Також ця система вірекомендована себе для розпізнавання обличь та жестів.

Система ARTTRACK, як найбільш розвинена, має кращу якість розпізнавання та є стійкою до швидких рухомих об’єктів та поганих світлових умов на кадрі.

3.2 SOLOSHOT3

Продукт SOLOSHOT3 – камера, що має вбудовані алгоритми відстежування об’єктів [19].

Основними перевагами цієї камери є:

а) якісний та оптичний зум, що дозволяє збільшити зображення у 65 разів і при цьому не втрачати якість;
б) записування відео у високій якості;
в) автоматичне відстежування об’єкта у кадрі.

Для того, щоб отримати дані з камери, використовується зручний інтерфейс, написаний на мові програмування Python. Проте, немає цілісної системи для роботи з даними з камер та їх відображення. Крім того, виробник не рекомендує використовувати цю камеру для зйомок в приміщеннях та місцях великого скупчення людей, що є суттєвим обмеженням в можливих сферах використання.

3.3 Vision4ce

Vision4ce є провідним постачальником обладнання для обробки відео та зображень у режимі реального часу для електрооптичних систем. Компанія також займається постачанням надійного обладнання для важких умов на землі, на морі та в повітрі, а також розроблює інтегровані програмні рішення, що включають складні алгоритми обробки зображень для пошуку та відстежування об’єктів [20].

Vision4ce пропонує свою камеру разом зі своїм програмним забезпеченням для задач відстежування об’єктів. Їх програмний продукт працює на операційній системі Windows або Linux та дозволяє розпізнавати одночасно багато об’єктів на кадрі та відстежувати їх місцезнаходження, а також класифікувати об’єкти на рухомі та нерухомі (рисунок 3.2).

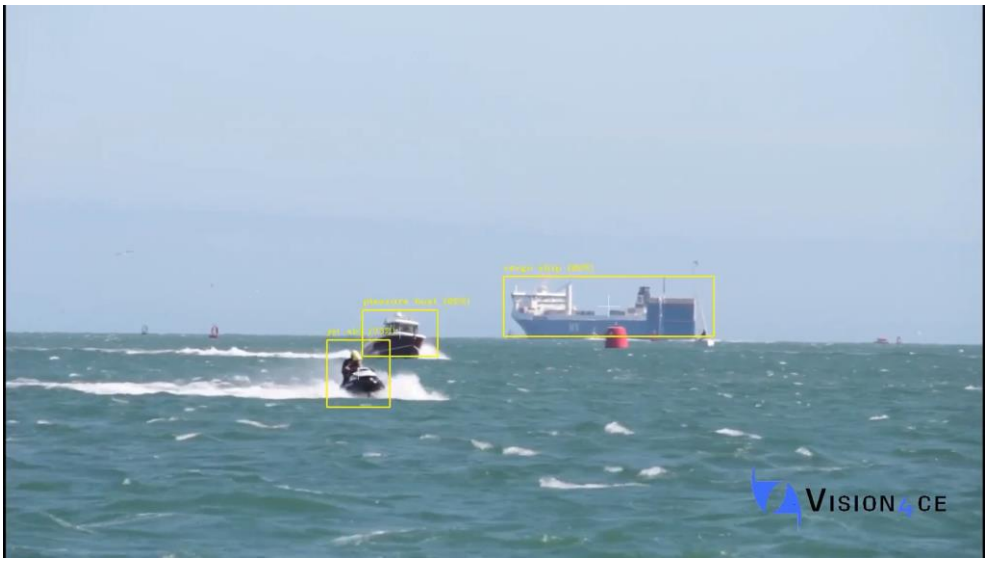

Рисунок 3.2 – Приклад відстежування об’єкту за допомогою Vision4ce.

РОЗДІЛ 4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ПОШУКУ ТА ВІДЕСТЕЖУВАННЯ ОБ'ЄКТІВ

Тепер, коли було проаналізувано алгоритми пошуку та відстежування об'єктів, а також розглянули існуючі програмні рішення, нарешті можна перейти до створення власного проекту.

Проект, який пов'язаний з використанням штучного інтелекту, відрізняється від традиційного програмного проекту. Відмінності полягають у стеці технологій, необхідності глибоких досліджень алгоритмів, а також в використанні потрібних навичок, пов'язаних з використанням нейронних мереж. Щоб реалізувати власний проект, потрібно використовувати таку мову програмування, яка матиме доступні та популярні інструменти для використання алгоритмів ШІ. Під це описання гарно підходить мова програмування Python.

Python – це одна з найпопулярніших, на даний момент, мов програмування, яка має багато сфер застосування. На графіку можна побачити рейтинги пошуку навчальних посібників з мов програмування за даними сайту Darly Solutions за 2020 рік [21] (рисунок 4.1).

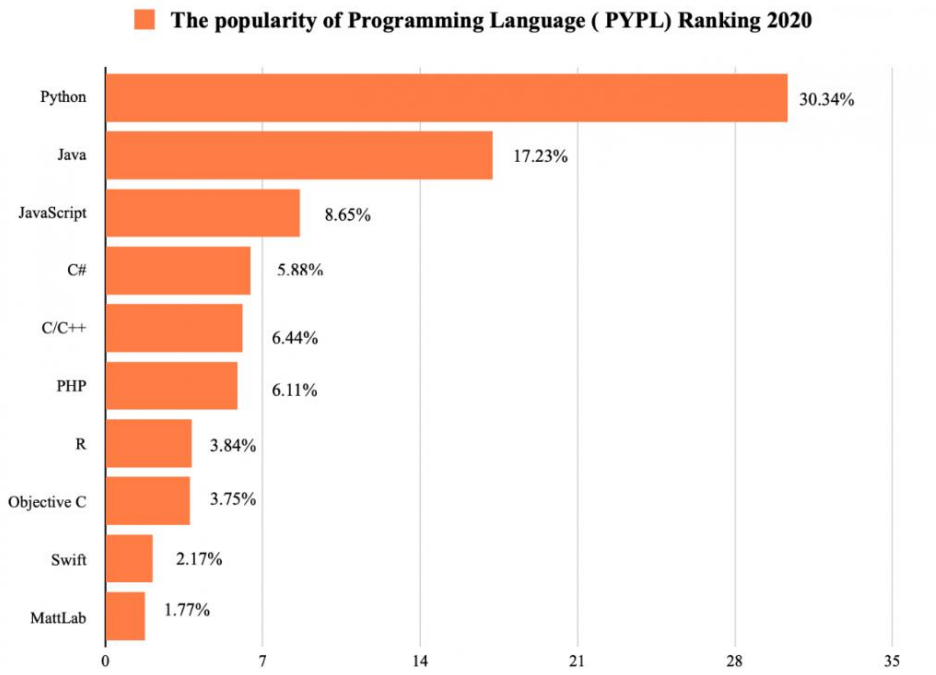

Рисунок 4.1 – Популярні мови програмування за 2020 рік.

Ця мова має багато бібліотек для машинного навчання, за допомогою яких можна будувати складні моделі для різних цілей. Вважається, що Python найкраще підходить для різних проектів пов’язаних з використанням штучного інтелекту. Дана мова є простою в використанні та дозволяє розробникам легко створювати бажані архітектури, а потім так само легко впроваджувати їх в виробництво. Окрім всього Python є кросплатформною мовою програмування, що дозволяє розробникам переносити проект з однієї машини на іншу без будь-яких змін [22].

4.1 Підготовка програмного середовища

Перед тим як почати розробку проекту, необхідно завантажити та встановити мову програмування Python (рисунок 4.2). Це можна зробити на офіційному сайті розробника мови.

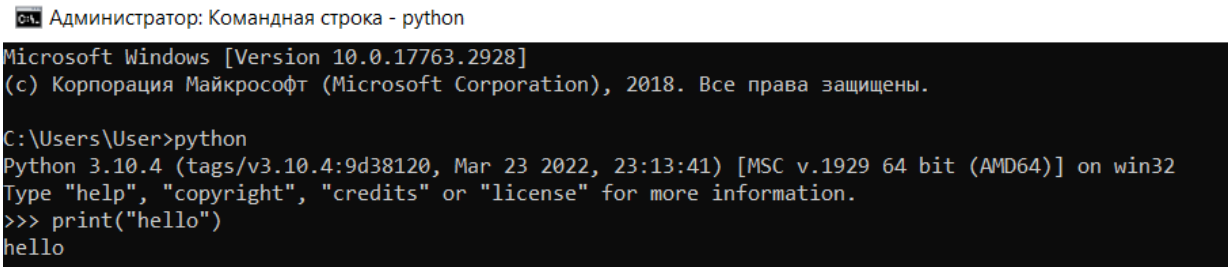

Рисунок 4.2 – Перевірка працездатності Python.

А також необхідно встановити систему керування бібліотеками під назвою pip. Вона буде використовуватись для встановлення різних додаткових бібліотек в наше середовище розробки. Але, якщо у вас версія Python 2.7.9, Python 3.4, чи більш новіша версія, то встановлення системи не є обов’язковим, оскільки вони вже містять pip за замовчуванням (або pip3 для Python 3) [23].

Також для створення проекту будемо використовувати текстовий редактор – PyCharm IDE. PyCharm – це інтегроване середовище розробки для мови програмування Python. Він був випущений на ринок інтегрованих середовищ

розробки для Python компанією JetBrains у 2010 році, щоб створити конкуренцію з PyDev і поширенішим середовищем розробки Komodo IDE [24]. Тестовий редактор також можна завантажити з офіційного сайту розробника. Для проекту можна обрати інше інтегроване середовище розробки, серед яких Atom, Visual Studio та ін. PyCharm було обрано через його ручний редактор коду з усіма корисними функціями: підсвіткою синтаксису, автоматичним форматуванням, доповненням та відступами.

4.2 Обробка зображення за допомогою ImageAI та RetinaNet

ImageAI - це проект, розроблений Мозесом Олафенва та Джоном Олафенва, командою з DeepQuest AI. Вона являє собою бібліотеку, яка створена, щоб надати розробникам можливість створювати програми та системи з автономними можливостями глибинного навчання та комп’ютерного зору [25].

ImageAI використовує декілька API, які працюють в автономному режимі – у нього є API виявлення об'єктів та їх відстеження, які можна викликати без доступу до інтернету. ImageAI використовує заздалегідь навчену модель і її можна легко налаштовати.

Настав час створювати проект. Після цього, за допомогою pip, потрібно встановити необхідні бібліотеки через термінал під проектом. Окрім ImageAI ще знадобляться такі бібліотеки:

- OpenCV;
- TensorFlow;
- NumPy;
- SciPy;
- Pillow;
- Matplotlib;

- H5py;
- Keras.

Про бібліотеку OpenCV та її можливості детально говорилося в розділі 2.2.

TensorFlow - це відкрита програмна бібліотека для машинного навчання, розроблена компанією Google. Він має комплексну та гнучку екосистему інструментів, бібліотеку та ресурси спільноти, які дозволяють дослідникам внедрять найсучасніші технології машинного навчання, а розробникам легко створювати та розвертати додатки на основі машинного навчання [26].

NumPy – це бібліотека, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами [27].

SciPy надає модулі для оптимізації, інтегрування, спеціальних функцій, обробки сигналів, обробки зображень та інших задач, які розв'язуються при інженерній розробці [28].

Pillow забезпечує широку підтримку форматів файлів, ефективне внутрішнє представлення та досить потужні можливості обробки зображень [29].

Matplotlib — це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python [30].

Бібліотека H5py — це інтерфейс до формату двійкових даних HDF5. HDF5 дозволяє зберігати величезну кількість числових даних і легко маніпулювати цими даними з бібліотеки NumPy [31].

Keras — це бібліотека, ціллю якого є дотримання найкращих практик щодо зниження користувацького навантаження. Він пропонує послідовні та прості API, мінімізує кількість дій користувача, необхідних для поширених випадків використання, а також надає чіткі й ефективні повідомлення про помилки [32].

Усі вищеперелічені бібліотеки можна встановити за допомогою спеціальних команд, які можна знайти в документації до бібліотеки ImageAI (рисунок 4.3).

- **Tensorflow** 2.4.0

```
pip install tensorflow==2.4.0
```

or Tensorflow-GPU if you have a NVIDIA GPU with CUDA and cuDNN installed

```
pip install tensorflow-gpu==2.4.0
```

- **Other Dependencies**

```
pip install keras==2.4.3 numpy==1.19.3 pillow==7.0.0 scipy==1.4.1 h5py==2.10.0 matplotlib==3.3.2 ope
```

- **ImageAI**

```
pip install imageai --upgrade
```

Рисунок 4.3 – Команди для встановлення необхідних бібліотек.

Існує також інший спосіб встановлення бібліотек. Завдяки PyCharm можна скористатись інтегрованим менеджером бібліотек та через пошуковий рядок знайти та встановити потрібні елементи з вибором потрібної версії (рисунок 4.4).

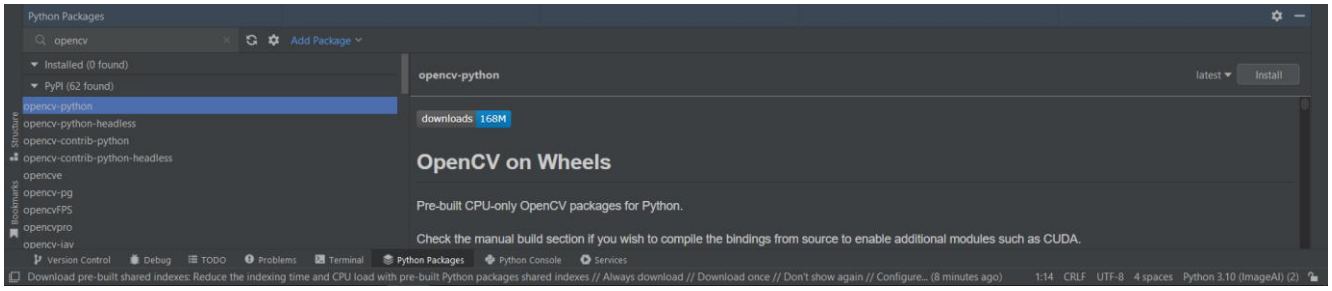

Рисунок 4.4 – Інтегрований менеджер бібліотек PyCharm.

Єдиним, що залишається зробити перед повноцінним написанням коду є завантаження моделі RetinaNet з офіційної сторінки ImageAI на GitHub та її додавання в проект.

Дана модель дозволить зрозуміти, що конкретно знаходиться на зображенні чи відеопотоці. Тобто, в даній моделі описані різні об’єкти та за допомогою опису

цих об’єктів можна відображати користувачу, що на кадрі знаходиться, наприклад, машина чи автобус. Так само як і модель YOLO, ця модель була навчена розпізнавати 90 різних типів об’єктів. В цьому їм допоміг набір даних COCO [33].

На рис 4.5 можна побачити блок-схему роботи майбутньої програми.

Рисунок 4.5 – Блок-схема роботи програми для пошуку об’єктів на зображенні.

Судячи з цієї блок-схеми, створення програми для застосування алгоритму пошуку об’єктів на зображенні не уявляє з себе нічого складного. Просто потрібно знати певні функції та їх функціонал для правильного застосування моделі RetinaNet.

Для повноцінної роботи програми потрібно імпортувати з бібліотеки ImageAI об’єкт ObjectDetection. Цей клас дозволить знаходити різні об’єкти на зображенні. Далі, додатково імпортуємо бібліотеку OS – це інтегрована бібліотека в Python, яка дозволить працювати з операційною системою. Створюємо нову зміну під назвою exec_path та присвоюємо їй функцію os.getcwd(). Ця функція дозволить вказати шлях до цього проекту, щоб виявити різні об’єкти, наприклад,

файл з моделлю RetinaNet або зображення, яке потрібно проаналізувати (рисунок 4.5).

```
from imageai.Detection import ObjectDetection
import os

exec_path = os.getcwd()
```

Рисунок 4.6 – Перша частина коду.

На основі об’єкту ObjectDetection створюється декілька властивостей. Створюємо змінну detector та присвоюємо їй клас ObjectDetection(). Далі, через detector звертаємося до трьох функцій: setModelTypeAsRetinaNet(), setModelPath(), та detector.loadModel(). Через першу функцію встановлюємо тип моделі RetinaNet, через другу – встановлюємо шлях до файлу з моделлю, в якій вказуємо назву цього самого файлу, через третю – завантажуємо модель всередині класу для пошуку.

Далі, створюємо змінну list, в якій будуть міститися список знайдених об’єктів. Цій змінній присвоюємо значення detector.detectObjectsFromImage(), в якій вказуємо назву та формат вхідного зображення через функцію os.path.join(). В цій самій змінній вказуємо вихідне зображення після процесу обробки.

Основна частина програми готова, але також можна прописати в код додатковий функціонал. В функцію detectObjectsFromImage() додається кілька змінних (Рисунок 4.6):

а) minimum_percentage_probability (мінімальний процент точності роботи алгоритму пошуку, чим менший процент, тим більше об’єктів буде відображено);
б) display_percentage_probability (відображення процентів точності над об’єктом);
в) display_object_name (відображення типу об’єкта).

```
detector = ObjectDetection()
detector.setModelTypeAsRetinaNet()
detector.setModelPath(os.path.join(exec_path, "resnet50_coco_best_v2.1.0.h5"))
detector.loadModel()

list = detector.detectObjectsFromImage(
    input_image=os.path.join(exec_path, "image_before.jpg"),
    output_image_path=os.path.join(exec_path, "image_after.jpg"),
    minimum_percentage_probability=70,
    display_percentage_probability=True,
    display_object_name=True
```

Рисунок 4.7 – Перша частина коду.

Для прикладу, можна завантажити будь-яке зображення з людьми та додати її в проект. На рис. 4.7 можна побачити зображення до та після роботи алгоритму пошуку.

Рисунок 4.8 – Результат роботи алгоритму: а – до обробки алгоритмом;
б – після обробки.

За бажанням, можна також скорегувати список об’єктів, які необхідно знайти на зображенні, через функцію CustomObjects().

4.3 Обробка відеопотоку за допомогою YOLO

В розділі 2.1.3 було лише згадано про існування алгоритму YOLO. Тепер перейдемо до принципу роботи цього алгоритму.

Спочатку зображення, яке вводиться в мережу, розбивається на секції, наприклад, на матрицю-сітку 3 на 3, тобто маємо 9 секторів. Кожна з цих секцій має певні параметри. Якщо допустити, що загальна кількість класів, які шукають на зображенні, дорівнює 3 (наприклад, автомобіль, рюкзак та кіт), то кожний сектор матиме в загальному по 8 параметрів. Тому, що кожний сектор має по 5 параметрів та 3 параметри класу.

Щоб коментувати ці параметри, необхідно знати, що таке обмежувальні рамки. При підготовці навчальних даних потрібно виділити об’єкт, який хочемо знайти на зображенні. А робиться це за допомогою цих самих рамок, які виділяють певно область зображення, на якій зображено потрібний об’єкт.

Для прикладу, необхідно знайти автомобілі, тому розміщуються обмежувальні рамки навколо усіх автомобілей, що є на зображенні. Тепер необхідно дізнатися значення 5 параметрів в кожному секторі на зображенні (рисунок 4.9).

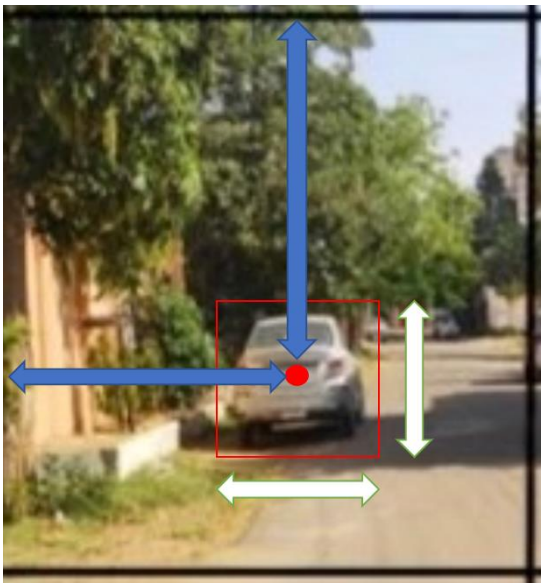

Рисунок 4.9 – Сектор з потрібним об’єктом.

Червнона точка – центр обмежувальної рамки. Горизонтальна синя стрілка – це параметр “tx”, який дорівнює відстані між центром рамки та лівим краєм сектора. Вертикальна синя стрілка – це “ty”, і він дорівнює відстані між центром рамки та верхнім краєм. Білі стрілки відповідають за ширину та висоту обмежувальної рамки.

Параметр “p” - це індекс об’єктності, який виражає ймовірність успішного знаходження об’єкта в обмежувальній рамці. Індекси p1, p2, p3 відповідають за ймовірність того, що об’єкт всередині рамки буде автомобіль, рюкзак, або кіт. Усі 9 секторів мають ці 8 параметрів, і вони допомагають алгоритму YOLO точно знаходити об’єкт [34].

Існують п’ять версій цього алгоритму, з них гарно зарекомендували останні три. Для нашої програми скористаємось YOLOv3.

Інтерпретуємо цей алгоритм в нашу програму. Для цього завантажуємо з цієї ж самої сторінки ImageAI на GitHub файл з моделлю алгоритму та додаємо до проекту.

Для застосування алгоритмів пошуку та відстежування на відеопотоці необхідно змінити декілька рядків коду. Замість об’єкту ObjectDetection будемо

використовувати VideoObjectDetection. Також потрібно встановити тип моделі через функцію setModelTypeAsYOLOv3(). А для виклику алгоритму пошуку та відстеження об’єктів на відеопотоці можна скористатись функцією detectObjectsFromVideo().

На рис 4.10 можна побачити блок-схему роботи програми для пошуку та відстежування об’єктів на відеопотоці.

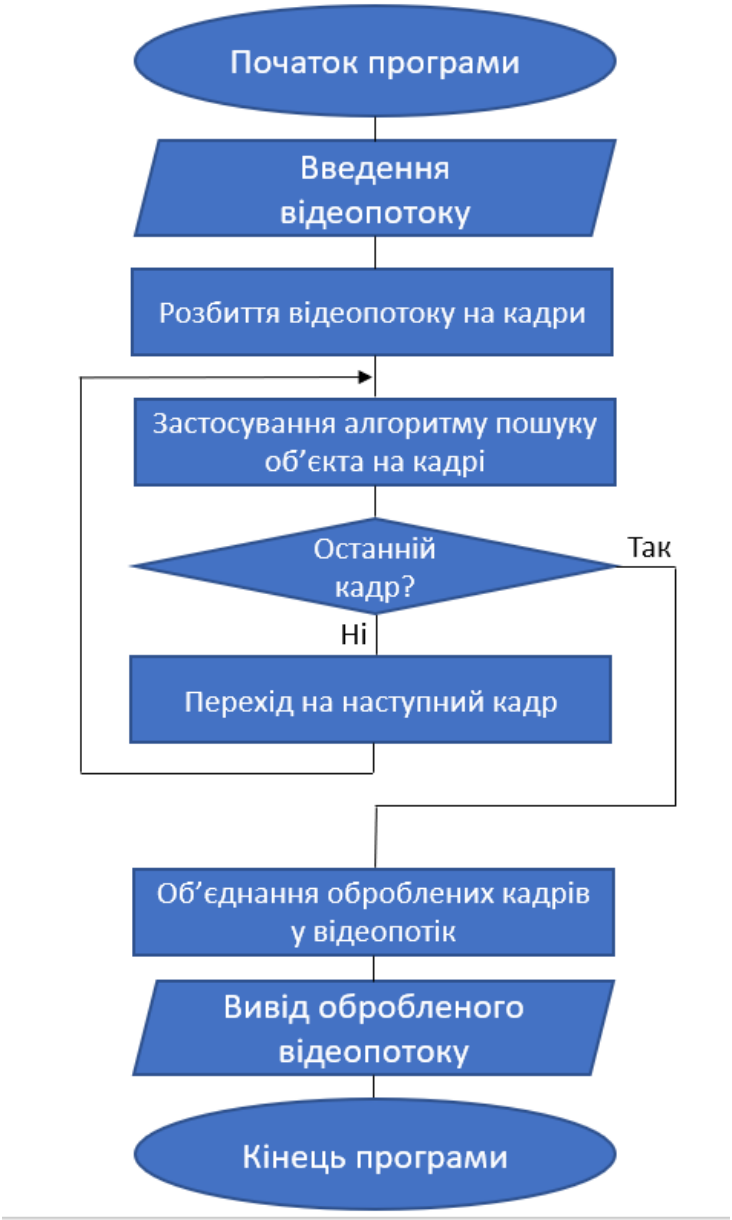

Рисунок 4.10 - Блок-схема роботи програми для пошуку об’єктів на відеопотоці.

Для прикладу, візьмемо відео з трафіком транспорту. На рис 4.11 можна побачити результат роботи.

Рисунок 4.11 – Результат роботи обробки відеопотоку.

Незважаючи на те, що алгоритм моделі YOLO є одним з найшвидших, швидкодія алгоритму сильно залежить від обчислювальної потужності комп'ютера. На обробку одного кадру може витрачатись 5 секунд. З урахуванням того, що в середньому відео може йти з 20-25 кадрами в секунду, то на обробку відео тривалістю в 30 секунд, може витрачатись приблизно одна година.

Звісно, запустити програму можна й на більш сучаснішому та потужнішому комп’ютері. Також можна для більш швидкої обробки кадрів збільшити процент точності роботи через змінну minimum_percentage_probability, щоб алгоритм не чіпав об’єкти, які не підпадають під цей процент. Або обмежити кількість кадрів в секунду для обробки, щоб алгоритм швидше пройшовся по відеопотоку, за допомогою числової змінної frames_per_second в функції detectObjectsFromVideo().

ВИСНОВКИ

В рамках цієї кваліфікаційної роботи було проаналізовано основні відомості про штучний інтелект та його напрямки досліджень, серед яких комп’ютерний зір. Також було проаналізовано існуючі алгоритми з пошуку та відстежування об’єктів, а також існуючі програмні рішення.

В результаті роботи було реалізовано працездатність алгоритмів пошуку та відстежування об’єктів.

Для підтвердження був розроблений програмний додаток на мові програмування Python, за допомогою інтегрованого середовища розробки PyCharm IDE, що призначений для пошуку та відстежування об’єктів на зображеннях та відеопотоках з виводом результатів.

Під час створення програми було також обговорено використання функцій бібліотеки ImageAI, що необхідні для створення програми, що може використовувати моделі алгоритмів пошуку та відстежування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Джоши Пратик – Artificial Intelligence with Python [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://mazz.keybase.pub/ebooks/ai/9781786464392-ARTIFICIAL_INTELLIGENCE_WITH_PYTHON.pdf
2. Штучний інтелект: історія виникнення та перспективи розвитку [Електронний ресурс]. – Режим доступу до ресурсу: https://futurum.today/shtuchnyi-intelekt-istoriia-vynyknennia-ta-perspektyvy-rozvytku/
3. Історія розвитку розпізнавання мови [Електронний ресурс]. – Режим доступу до ресурсу: https://dut.edu.ua/ua/news-1-576-9187-istoriya-rozvitku-rozpiznavannya-movi_kafedra-shtuchnogo-intelektu
4. Обработка естественного языка [Електронний ресурс]. – Режим доступу до ресурсу: https://web.archive.org/web/20151208163941/http://chernykh.net/content/view/1105/1189/
5. Игровой искусственный интеллект [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B8%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82
6. Пітер Джексон – Введение в экспертные системы [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: https://lib.nsu.ru/xmlui/bitstream/handle/nsu/9053/Jackson-2001.pdf?sequence=1&isAllowed=y

7. Комп’ютерний зір [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B8%D0%B9_%D0%B7%D1%96%D1%80
8. Ганнинг Жу, Томас С. Хуанг – Tracking Articulated Hand Motion with Eigen Dynamics Analysis [Електронний ресурс]. – 2003. – Режим доступу до ресурсу: http://lear.inrialpes.fr/people/triggs/events/iccv03/cdrom/iccv03/1102_zhou.pdf
9. Признаки Хаара [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8_%D0%A5%D0%B0%D0%B0%D1%80%D0%B0
10. Р. Гіршик, Дж. Донахью, Т. Даррел, Дж. Малік – Rich feature hierarchies for accurate object detection and semantic segmentation [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/pdf/1311.2524.pdf
11. Р. Гіршик – Fast R-CNN [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/pdf/1504.08083.pdf
12. Ш. Рен, К. Ге, Р. Гіршик, И. Сун – Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/pdf/1506.01497.pdf
13. В. Лю, Д. Ангелов, Д. Эрхан, К. Сегеди, С. Рид, Ч. Фу, А.С. Берг – SSD: Single Shot MultiBox Detector [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/pdf/1512.02325.pdf
14. OpenCV [Електронний ресурс]. – Режим доступу до ресурсу: https://opencv.org/about/
15. Evaluation TLD/Predator algorithm [Електронний ресурс]. – Режим доступу до ресурсу: http://vision.stanford.edu/teaching/cs231b_spring1415/papers/Kalal-PAMI.pdf
16. GOTURN: Deep Learning based Object Tracking [Електронний ресурс]. – Режим доступу до ресурсу: https://learnopencv.com/goturn-deep-learning-based-object-tracking/

17. Як працює система тотального стеження за громадянами в Китаї [Електронний ресурс]. – Режим доступу до ресурсу: https://nv.ua/ukr/world/countries/pobachiti-kozhnogo-yak-pracyuye-sistema-totalnogo-stezhennya-za-gromadyanami-v-kitaji-50005116.html
18. Advanced Realtime Tracking [Електронний ресурс]. – Режим доступу до ресурсу: https://ar-tracking.com/en/tracking-systems
19. SOLOSHOT3 [Електронний ресурс]. – Режим доступу до ресурсу: https://soloshot.com/collections/soloshot3
20. Vision4ce [Електронний ресурс]. – Режим доступу до ресурсу: https://www.vision4ce.com/
21. The most popular programming languages in 2021 [Електронний ресурс]. – Режим доступу до ресурсу: https://darly.solutions/the-most-popular-programming-languages-in-2021/
22. Python [Електронний ресурс]. – Режим доступу до ресурсу: https://www.python.org/about/
23. Installation – pip 7.1.2 documentation [Електронний ресурс]. – Режим доступу до ресурсу: https://web.archive.org/web/20150907184553/https://pip.pypa.io//en/latest/installing.html
24. PyCharm: the Python IDE for Professional Developers by JetBrains [Електронний ресурс]. – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/
25. Official English Documentation for ImageAI [Електронний ресурс]. – Режим доступу до ресурсу: https://imageai.readthedocs.io/en/latest/
26. TensorFlow [Електронний ресурс]. – Режим доступу до ресурсу: https://www.tensorflow.org/
27. NumPy [Електронний ресурс]. – Режим доступу до ресурсу: https://numpy.org/
28. SciPy [Електронний ресурс]. – Режим доступу до ресурсу: https://scipy.org/

29. Pillow [Електронний ресурс]. – Режим доступу до ресурсу: https://pillow.readthedocs.io/en/stable/
30. Matplotlib [Електронний ресурс]. – Режим доступу до ресурсу: https://matplotlib.org/stable/index.html
31. HDF5 for Python [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.h5py.org/en/stable/
32. Keras: the Python deep learning API [Електронний ресурс]. – Режим доступу до ресурсу: https://keras.io/
33. What Object Categories / Labels Are In COCO Dataset? [Електронний ресурс]. – Режим доступу до ресурсу: https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/
34. Дж. Редмон, С. Діввала, Р. Гіршик, А. Фархаді – You Only Look Once: Unified, Real-Time Object Detection [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/pdf/1506.02640.pdf