Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра математичної інформатики

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

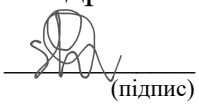
за спеціальністю 122 «Комп’ютерні науки»

на тему:

Система адаптивного трекінгу об’єктів в реальному часі

Студента 4-го курсу

Шевченка Олексія Олександровича



(підпис)

Науковий керівник:

Терещенко Ярослав Васильович

(підпис)

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

Студент 

(підпис)

Робота заслухана на засіданні кафедри математичної інформатики та рекомендована до захисту в ЕК, протокол № …… від …….2022р.

Завідувач кафедри Терещенко В.М.

Київ – 2022

РЕФЕРАТ

Робота складається зі вступу, 5 розділів, висновків, списку використаних джерел (14 найменувань). Робота містить 24 рисунки. Загальний обсяг становить 42 сторінок, основний текст роботи викладено на 30 сторінках.

Ключові слова: МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ЗБІР ІНФОРМАЦІЇ, ОБРОБКА ВІДЕОПОТОКУ, КОМП'ЮТЕРНИЙ ЗІР, ВИЗНАЧЕННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННІ, ВІДСЛІДКОВУВАННЯ ОБ'ЄКТІВ НА ВІДЕО, АРХІТЕКТУРА НЕЙРОННИХ МЕРЕЖ.

Об'єктом роботи є розв'язування задачі визначення та подальше відстежування собак на відеозаписі у реальному часі, використовуючи засоби машинного навчання. Предметом роботи є дослідження існуючих систем, проектування власної моделі машинного навчання з використання згорткової нейронної мережі, її тестування та подальше покращення роботи з розпізнавання та трекінгу в реальному часі.

Метою роботи є аналіз існуючих системи штучного інтелекту та проектування, розробка і реалізація власних засобів на основі вже існуючих для її використання пошуковими службами та службами з відлову тварин.

Методи розроблення: проектування моделей машинного навчання, розробка згорткових нейронних мереж, методи оптимізації навчання нейронних мереж, методи покращення результатів нейронної мережі.

Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки PyCharm, мова програмування Python.

Результати роботи: спроектовано та розроблено згорткову нейронну мережу для визначення собак на зображенні, мережу було натреновано на великому наборі вхідних даних для специфікації алгоритму на розпізнаванні собак, розроблено алгоритм, який за допомогою нейронної мережі відстежує шукані

об’єкти між двома кадрами на відеозаписі у реальному часі застосовуючи фільтр Калмана та відстань Евкліда, як метрики. Написано допоміжні засоби для більш зручного застосунку існуючої системи, які надають можливість як визначати собак на окремих зображеннях, так і відслідковувати їх на відеопотоці з файлу або пристрою підключеного до машини, на якій розгорнута система.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ML – Machine learning

DL – Deep learning

CNN – Convolution neural network

R-CNN - Region Convolution Neural Network

ReLU – Rectified Linier Unit

ВСТУП

Оцінка сучасного стану об’єкта розробки. Кожного року інформаційний та технологічний прогрес має все більший і більший вплив на буденне життя кожної людини. Останніми роками все ширше і ширше розповсюджуються технології машинного навчання та великі корпорації розробляють свої нейронні мережі для різних цілей: системи розумного пропонування реклами, розпізнавання обличчя, пошукові алгоритми, покращення фотографій та аудіофайлів, автономні засоби пересування, розумні автоматизовані перекладачі. Вони використовуються у медицині, машинобудуванні та навіть подекуди у творчості. Основні технічні ідеї, що стоять за глибоким навчанням, були розроблені це кілька десятиліть тому, то чому саме у останні роки звідусіль почало доноситись про глибоке машинне навчання. Причинами такого швидкого розвитку індустрії машинного навчання стало різке збільшення інформації, яка зберігається на електронних носіях. Інформація про дії користувачів інтернету, фінансова інформація та звіти, зображення, відео- та аудіо- записи будь-яка інша інформація, може бути матеріалом для навчання нейронних мереж, а чим більше цих даних, тип загалом більшу мережу можливо натренувати, звідси слідує покращення точності виконання цільових задач нейронних мереж. Наразі важко собі уявити життя без сервісів пошуку, рекомендацій та автоматизації, які основані на алгоритмах глибокого машинного навчання, це стало невід’ємною частиною життя людства.

Актуальність роботи та підстави для її виконання. Хаотична та тяжка ситуація в сучасній Україні стала причиною появи великої кількості тварин на вулиці. Люди вимушені змінювати своє місце проживання та тікати від війни та, не маючи змогу забрати своїх домашніх улюбленців з собою, вимушені випускати їх на вулицю. Це в свою чергу значно підвищує кількість вуличних

тварин та вулицях міст України, що після закінчення війни стане нагальною проблемою.

Запропоноване рішення пошуку собак за допомогою глибокого навчання допоможе вирішити декілька проблем: пошук та відлов вуличних собак притулками та пошук домашніх улюбленців їхніми господарями після повернення людей на їхнє постійне місце проживання після завершення бойових дій на певних територіях. Зазначений інструмент потребує подальшого розповсюдження та широкої інтеграції для використання в широких колах людей.

Мета й завдання роботи. Метою роботи є проектування та реалізація моделі глибокого машинного навчання для відслідковування собак на відео у режимі реального часу, тренування моделі на наборі даних зображень собак за допомогою бібліотек мови програмування Python, ітеративне тренування та покращення моделі, імплементація алгоритмів оптимізації як архітектурних, так і пов’язаних з обчисленням. Надання можливості розширення моделі та додаткового тренування на даних специфікованих для джерел збору відео, які будуть використовуватися в реальності, а також забезпечення цілісності та інтегрованості усіх компонентів системи.

Виконання роботи можна умовно поділити на такі етапи:

- Дослідження існуючих рішень машинного навчання з машинного зору, знаходження об’єктів;
- Проектування моделі пошуку об’єкту на зображенні на основі існуючих рішень, необхідної для коректного функціонування системи;
- Розробка функціоналу з можливістю зручного налаштування та подальшого тренування нашої моделі пошуку певних об’єктів на зображенні;
- Тренування моделі на основі набору зображень, потрібних для цілі нашого застосунку;

- Специфікація моделі виявлення собак на зображеннях в залежності від результатів ближче до потрібного нам результату;
- Дослідження існуючих вільно поширюваних рішень з трекінгу об’єктів на зображенні;
- Побудова свого функціоналу з передбачення на основі минулої картинки положення цікавого нам об’єкту на наступній картинці;
- Модифікація алгоритму передбачення шляхом вибору найкращого для нас способу обрахування відстані від положення об’єкту на попередньому зображенні в відеопотоці до передбачених варіантів положення на наступних кадрах;
- Створення інструментарію для тестування результатів системи як на окремих зображеннях с використання тільки натренованої моделі для розпізнавання .

Перераховані вище етапи мають під собою систематизацію існуючого досвіду створення прикладних програм та моделей комп’ютерного зору з використанням актуальних підходів у розробці подібних по цілям програмних засобів.

Об'єкт, методи й засоби розроблення. Для реалізації засобів та моделей машинного навчання системи трекінгу у реальному часі було обрано мову програмування Python та її зовнішні бібліотеки tensorflow, використаної для розробки нейронної мережі, що визначає об’єкт на окремому зображенні, opencv для реалізації алгоритму передбачення та зв’язування визначених першою моделлю об’єктів на різних кадрах у відеопотоці та numpy для проведення обчислень та представлення даних на будь-яких етапах розробки та використання програми. За основу розробки згорткової нейронної мережі, яка визначає об’єкти на зображенні, було взято архітектурну методологію побудови моделі Yolo, а для побудови міжкадрового трекеру була взята ідея алгоритму Sort з використанням

фільтру Калмана та відстані Евкліда, як метрики. Для тренування моделі було узято набір даних COCO (Common Objects in Context) від Microsoft

РОЗДІЛ 1. АНАЛІЗ РИНКУ

1.1 Загальний огляд ситуації

На ринку продуктів машинного навчання не так багато існує готових продуктів з трекінгу, особливо в області трекінгу об’єктів у реальному часі для кожної ситуації та задачі потрібні свої специфіковані рішення. Але все ж таки на ринку можливо знайти рішення для відслідковування об’єктів на відео, частіше за все людей та машин, але зустрічаються і моделі заточені на трекінг тварин, але майже всі такі системи використовуються виключно на фермах і налаштовані на домашню худобу: корови, вівці, свині, такі як Cattle-Care та Cows.ai . Готових продуктів для визначення та відслідковування собак на відеозаписах немає, але слід зазначити що багато продуктів розробники викладають в спільний доступ, тим самим намагаючись розвивати індустрію даючи своїм колегам готові приклади для рішення певних задач, а новачкам в цій сфері дивитись і вчитися на архітектурах та рішеннях готових продуктів.

Повертаючись к відслідковуванню собак на зображенні слід зазначити, що ця поставлена задача складається з двох менших, перша – це визначити об’єкт на зображенні, на припустимо першому кадрі відеозапису, а друга – порівнюючи об’єкти з першого і другого зображення передбачити рух предмету і відслідковувати його постійно. Тому розберемо методи вирішення двох окремих задач: визначення об’єкту на зображенні та алгоритмів відслідковування крізь кадри відеозапису.

1.2 Огляд можливих рішень для задачі знаходження об’єктів на зображенні

Раніше технології визначення певних об’єктів на зображенні розроблялись шляхом класичного машинного навчання, що було потребувало великих обчислювальних потужностей та давали незадовільні результати. Але,

починаючи з 2014, індустрія зазнала значного прориву після того, як було представлено нову методологію роботи із визначення об’єктів на зображенні – R-CNN модель. На теперешній час існують рішення основані на R-CNN шляхом покращення її архітектури та введення додаткових засобів облегшуючих обчислення або покращуючих результати. Найвідоміші та найуспішніші з них це Yolo, Fast R-CNN, Faster R-CNN, SSD.

Якщо порівнювати не сильно вдаючись у архітектурні особливості, R-CNN, Fast R-CNN та Faster R-CNN це послідовні покращення першої с них, ці моделі відносяться к так званим двоетапним методам, бо поділяються на перший етап, в якому визначаються регіонали в яких нейронна мережа вирішила, що вірогідно знайде щось важливе, та другий етап, в якому вже мережа зосереджується на знайдених регіонах і вже в них шукає потрібні об’єкти, як це зображено на рис. 1.1. Вони мають гарну точність, але потребують доволі значну кількість часу на тренування та обробку зображення, порівнюючи з іншими рішеннями задачі знаходження об’єктів на зображенні.

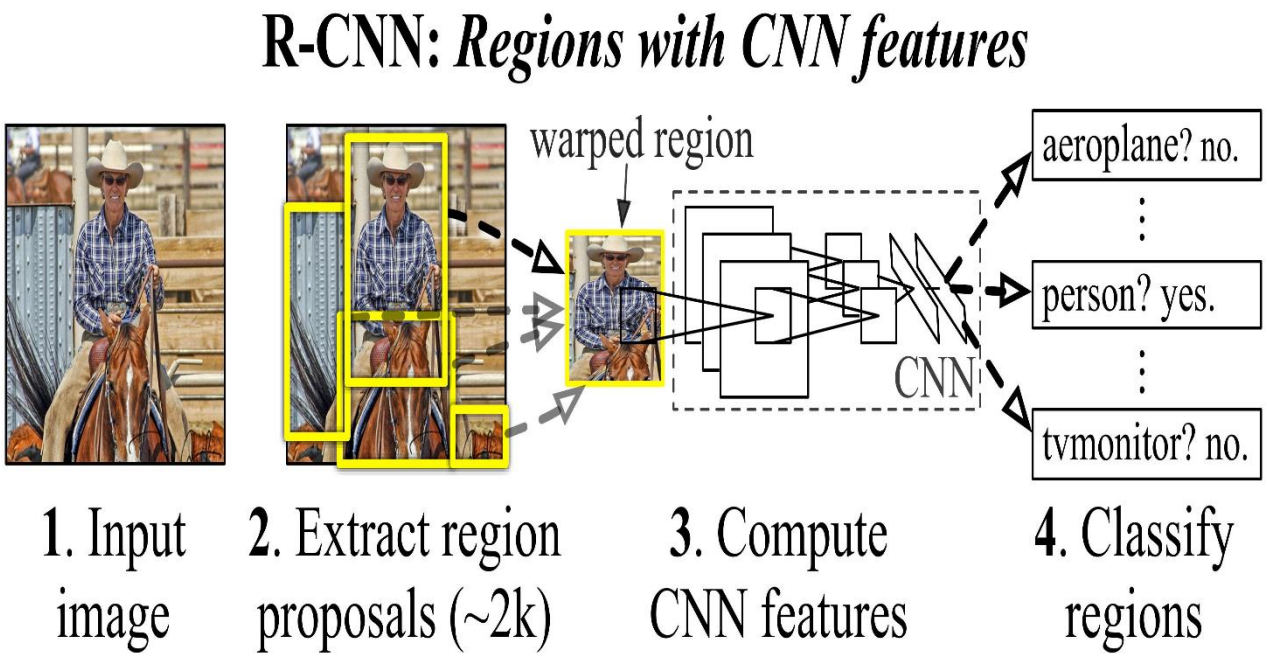


Рис. 1.1 R-CNN

З наведених вище моделей одноетапними є Yolo, SSD. Вони не мають спеціального етапу для припущення важливості того чи іншого регіону і виконуються в один етап. Вони використовують підхід, в якому передбачають певну кількість рамок з різними характеристиками, обмежуючих шуканий об'єкт, вираховуючи його клас, якщо в моделі декілька типів шуканих об'єктів, та вірогідність, тобто впевненість системи у своєму рішенні, а потім обирають найвірогідніші рамки та класи, надалі коректуючи положення обмежуючих рамок. Загальну схему обробки показано на рис. 1.2. На відміну від двоетапних такі моделі мають меншу потребу у часі, але дають менш точні результати.

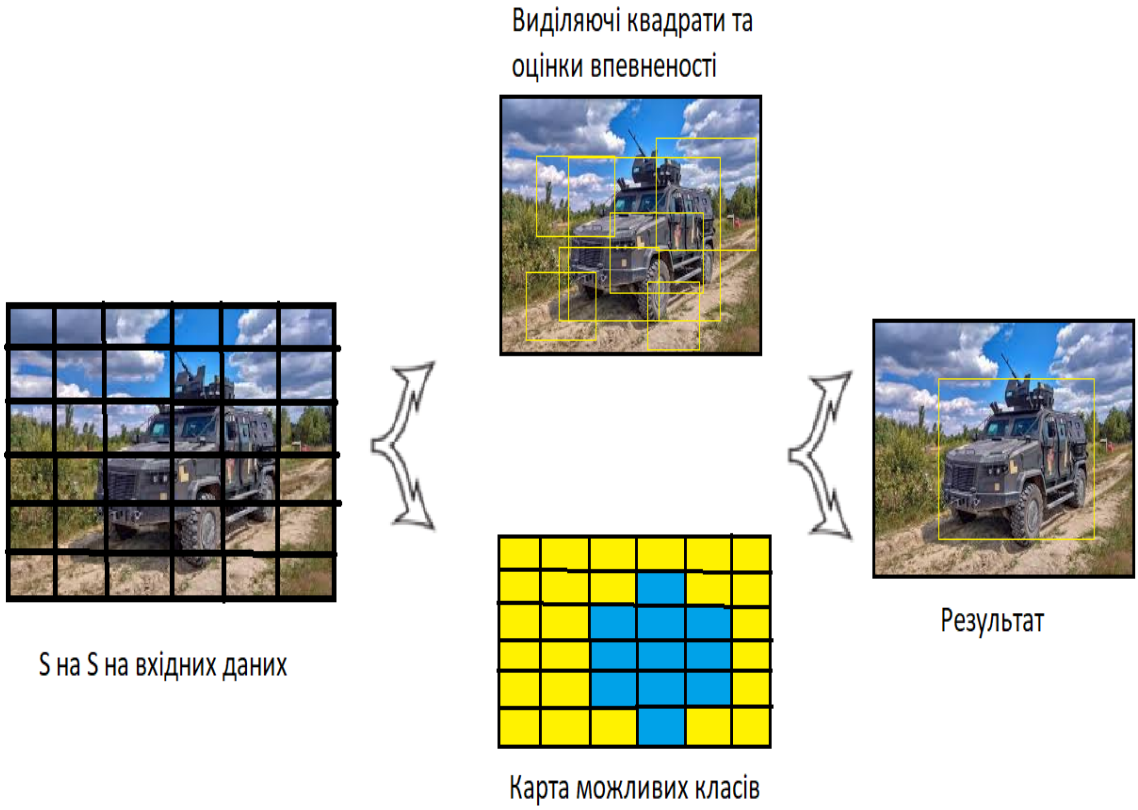


Рис. 1.2 Схема обробки зображення Yolo

Порівнюючи наведені вище типи моделей для визначення у реальному часі підходять більше одноетапні алгоритми, тому що мають більшу швидкодію.

1.3 Різниця між знаходження об’єктів на кадрах в реальному часі та відслідковуванням об’єкту в реальному часі

Інколи люди не розуміють різницю між двома методами комп’ютерного зору: визначення об’єктів на кадрах в реальному часі і відслідковуванням об’єкту на відео в реальному часі. Слід зазначити що перший з методів, це визначення наявності об’єктів на кожному зображенні (кадрі) в відео один за одним, в цього є свої плюси: простіша реалізація, тому що не потрібно реалізовувати алгоритм трекінгу, а просто постійно застосовувати до зображення нашу нейронну мережу, яка визначає об’єкти на зображенні. Але в свою чергу є мінуси, по-перше це потребує більше ресурсів комп’ютера, по-друге при різкій зміні форми (повороті чи падінні) алгоритм визначення об’єкту його втратить і скоріше за все не впізнає або можливо визначить неправильно. В свою чергу, алгоритм відслідковування об’єктів може передбачати зміни, використовуючи інформацію з минулих кадрів, тому він краще справляється з різкими змінами.

1.4 Порівняння існуючих алгоритмів відслідковування об’єктів на відеозаписах та відеопотоках

Наступним етапом обробки відеофайлу буде передбачення для визначеного об’єкту на першому зображенні наступного положення на наступному кадрі. Найпоширенішими алгоритмами для цієї задачі є Goturn, FCSiam, Sort, DeepSort, SiamRPN.

Sort алгоритм використовує фільтр Калмана та угорський алгоритм для визначення відношення між елементами і по цьому значенню пов’язує об’єкти на різних кадрах. DeepSort в свою чергу пішов ще далі і покращив алгоритм Sort додавши ReID. ReID – це алгоритм, який навчений запам’ятовувати і ідентифікувати об’єкти по їх зовнішнім рисам рисам. Готові рішення найчастіше зустрічаються для розпізнавання людини по обличчю та зовнішньому вигляду

(форма, розміри, одежа, тощо) або для розпізнавання інших розповсюджених речей у комп’ютерному зорі: марки машин, домашня худоба.

В свою чергу Goturn, FCSiam, SiamRPN оптичні трекери, які є насправді натренованими нейронними мережам, як це показано на рис. 1.3, використовуючи ці варіанти ви отримаєте гарну швидкість, але якщо в вас є можливість і дані аби натренувати свою власну мережу спеціалізовану під вашу задачу, то краще обрати цей варіант. В загальному використанні готових продуктів ви можете зіткнутися з проблемами при перепаді світла чи при різкій зміні форми об’єкту. Також не враховуючи направлення та швидкість, а тільки спираючись на нейронну мережу, інколи виникають проблеми з розпізнаванням схожих об’єктів.

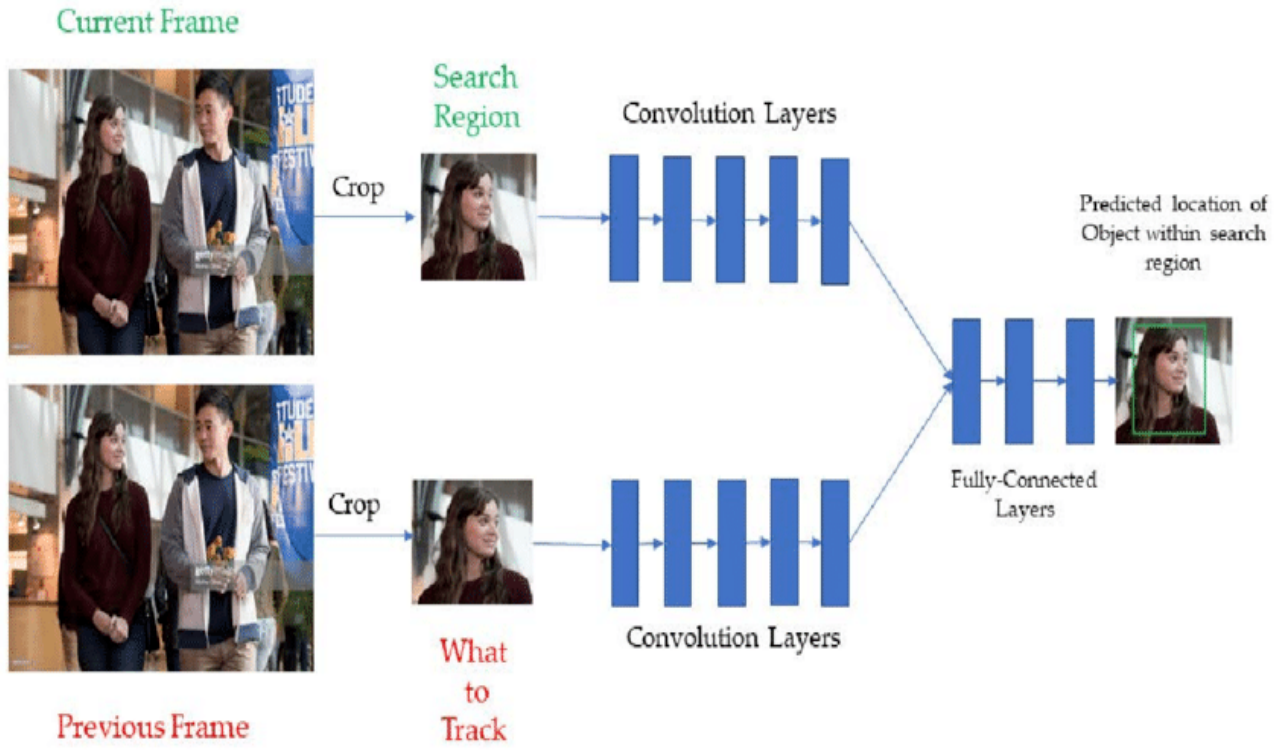


Рис. 1.3 Goturn

РОЗДІЛ 2. ОГЛЯД ФУНКЦІОНАЛУ СИСТЕМИ

2.1 Загальний огляд функціоналу системи

В цьому розділі зосередимось на функціональних можливостях моделі,

Реалізована система адаптивного трекінгу собак в реальному часі надає можливості як прямі, такі як визначати і відслідковувати собак на пристрої підключеному до комп’ютера, так і відслідковування собак на збереженому відеофайлі, який можна подати на вхід до системи, та визначення собак на фотографіях (статичних зображеннях).

2.2 Функціонал знаходження об’єкту на окремому зображенні

Аналізуючи функціонал почнемо з простішого, а саме з визначення та виділення об’єкту на окремому зображенні. Для запуску цієї частини

функціоналу потрібно вказати шлях до зображення, яке ми плануємо оцінювати

в файлі *detection_demo.py*, для цього потрібно задати значення змінної image_path рівної відносному шляху від кореневого каталогу проекту до самого файлу із зображенням. Після цього нам потрібно перейти в командному рядку у кореневу директорію проекту і звідти виконати *python detection_demo.py* після чого результат буде виведено на екран як це показано на Рис. 2.1.

На зображенні Рис. 2.1 ми можем побачити дві ключові функціональні особливості. Перше – це рамку, яка знаходиться навколо знайденого алгоритмом об’єкту, а друге – це вірогідність, впевненість алгоритму, того факту, що знайденим об’єктом є собака.

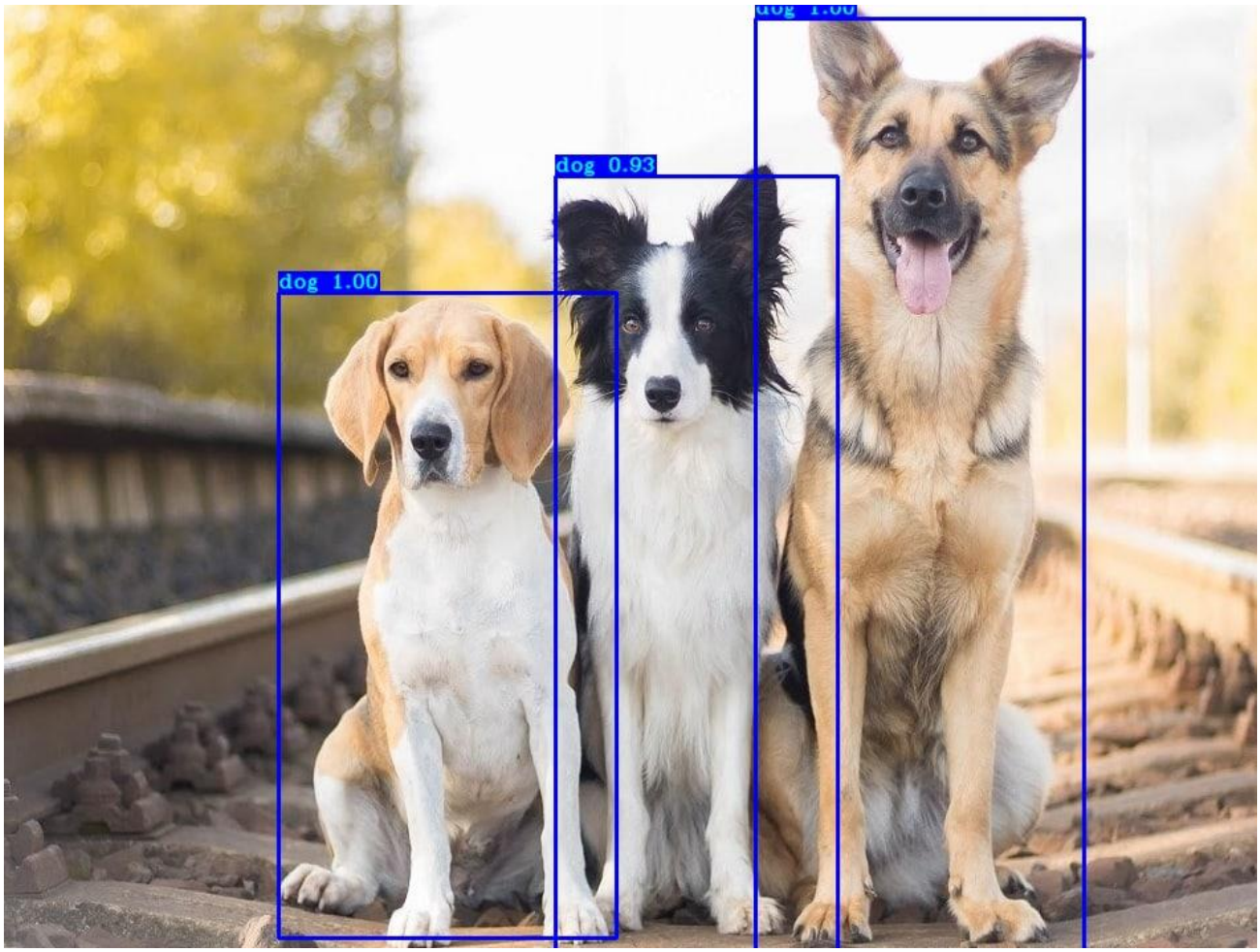


Рис 2.2 Декілька собак

На Рис. 2.2 зображена робота алгоритму визначення об’єктів на зображенні, якщо на фотографії буде не одна собака. Алгоритм виділяє та прораховує вірогідності для кожної з собак окремо.

Цей фунціонал є не основним в нашій системі, але є побічним продуктом способу реалізації основної задачі систему. Він реалізується першою частиною алгоритму системи, а саме алгоритмом визначення об’єкту на зображенні моделлю глибокого навчання на основі архітектури darknet53.

2.3 Функціонал трекінгу об’єктів на відеозаписі з файлу

Перейдемо до використання другого етапу роботи системи, а саме відслідковування собак на зображенні на відео із файлу. Для початку обробки системою відеофайлу потрібно вказати відносний шлях від кореневої директорії проекту до потрібного файлу, який ми запланували проаналізувати, присвоївши змінній video_path у файлі *object_tracking.py* в основній директорії потрібне значення шляху.

Після зміни файлу ми повинні перейти у командному рядку до директорії проекту і виконати команду *python object_detection.py*, це викличе нове вікно в якому буде програватися задане відео з помітками, як це показано на Рис. 2.3. Відео буде програватися з швидкістю обробки кадрів системою, наприклад на машині, на якій відбувалася розробка, кількість кадрів коливається від 10 до 20 при використанні ресурсів процесора, тобто без використання потужностей дискретної відеокарти.

На Рис. 2.3 видно, що в кожному з наведених кадрів, як і в цілому на відео, собака виділяється рамкою, якщо собак декілька, то вони всі виділяються і підписуються відповідним номером, як це можна побачити зліва зверху від рамки об’єкту.

Рис 2.3 Набір кадрів з вихідного відео

2.4 Відслідковування об’єктів відеопотоку з підключеного до комп’ютера камери

Третє можливе використання системи це аналіз відеопотоку, який в реальному часі знімається на записуючий пристрій, який підключений до

машини, на якій розгорнутий проект. Для використання цього функціоналу ми повинні присвоїти змінній *video_path* значення пустого текстового рядку або None, а потім в командному рядку з директорії проекту ми повинні виконати команду *python object_detection.py.*

Виконання відбувається аналогічно до минулого прикладу з відслідковуванням на відео з файлу, але швидкість зміни кадрів в секунду не залежить від власних потужностей машини, на якій розгорнута система.

2.5 Можливість натренувати на власних вхідних даних

Якщо ми хочем змінити модель, наприклад, для покращення точності визначення об’єктів на зображенні або для знаходження інших класів об’єктів ми можемо надати моделі визначення об’єктів на зображенні власні потрібні нам значення. Це реалізовано в декілька кроків. По-перше ми повинні завантажити власний набір зображень до папки data. Потім ми повинні згенерувати файл, в який ми запишемо шляхи до зображень з набору за допомогою виконання директиви з командного рядка python make_new_data.py, налаштувавши шлях до папки, в якій ми зберігли зображення.

Також потрібно провести таку саму операцію над тестовими даними, щоб їх також використовувати в тренуванні моделі. Тепер коли ми підготували дані для використання моделлю нам потрібно змінити шляхи в файлі config.py. В ньому ми вказуємо шляхи до згенерованих на минулому кроці файлів з шляхами до тренувальних і тестових зображень. Також в цьому файлі ми можемо вказати кількість епох, яку буде тренуватися наша модель. Для покращення точності слід виставляти значення більше 50, аби модель мала час та дані аби достатньо натренувати себе і підібрати потрібні для визначення ваги. Потім переходимо у кореневу директорію проекту і виконуємо команду *python train.py.* Після цього залишається тільки чекати. Для пришвидшення тренування алгоритм буде

намагатися використовувати ваші графічні потужності, якщо вони є, або буде використовувати потужності процесору, якщо графічних нема.

РОЗДІЛ 3. ТЕОРЕТИЧНЕ ПІДҐРУНТЯ РЕАЛІЗАЦІЇ СИСТЕМИ

3.1. Загальне представлення нейронних мереж

У сучасному світі нейронні мережі виконують багато різноманітних задач, але ці всі цілі можливо узагальнити у такі направлення: передбачення, можливість вгадати подальший розвиток якоїсь ситуації по вхідним даним попередніх змін, класифікація, вгадування певного класу шуканого об’єкту по вхідним даним, та розпізнавання, на якому ми зупинимося більш детально у наступних пунктах. Найменшою складовою нейронної мережі є нейрон, він отримує на вхід певні дані проводить прості обрахування і видає деякий результат. З нейронів складаються шари, вони поділяються на три типи, вхідний (синій), які отримують інформацію та передають її на нейрони прихованих шарів, прихований (червоний), які проводять деякі обчислення та передають інформацію далі на наступні приховані шари або на вихідні нейрони, та вихідний (зелений), який отримує результати останнього прихованого шару та перетворює її в результат. У мережі може бути необмежена кількість прихованих шарів, але вхідних та вихідних по одному.

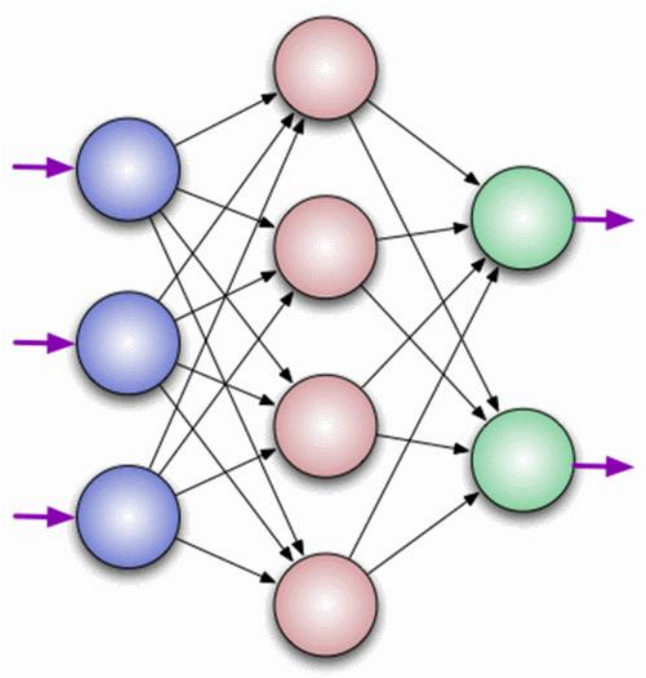


Рис. 3.1

Детальніше роздивимось, які обчислення проводяться на кожному прихованому шарі:

$$Z^{[i]} = W^{[i]T} A^{[i-1]} + b^{[i]}$$

$$A^{[i]} = g(Z^{[i]})$$

Де $A^{[i]}$ це вихідні дані з і-го шару ($A^{[0]}$= X)

$W^{[i]}$ – це матриця ваг для і-го шару,

$b^{[i]}$ – вектор зміщення для і-го шару,

g() – функція активації, це може бути sigmoid, ReLU, tanh, Leaky ReLU, тощо.

При тренуванні моделі після проходження одного набору тренувальних даних підраховується функція втрат, яка задається при побудові архітектури моделі, яка вказує на величину розбіжності між значеннями, які вирахувала нейронна мережа і які вказані у тренувальних даних. Основна задача моделі при тренуванні змінювати матрицю ваг та вектор зміщення таким чином, щоб функція втрат ставала все меншою. Таким чином тренуючи нейронну мережу ми збільшуємо шанс на те, що вона буде мати більшу точність на тренувальних даних, і подібних на них за характеристиками.

3.2 Комп’ютерний зір

Комп’ютерний зір це одна з областей, яка швидше за все розвивається завдяки глибокому навчанню. Воно використовується в автомобільній промисловості для створення засобів для розпізнавання інших автомобілів, пішоходів, дорожніх знаків і інших об’єктів, що можуть траплятися на дорозі, в розпізнаванні обличь для захисту персональних даних чи корпоративних об’єктів, в різного виду застосунків підбору фотографій чи розумного аналізу зображень та багато ще де. Основною проблемою розвитку комп’ютерного зору по методології класичного машинного навчання є велика кількість вхідних параметрів, особливо для зображень великої роздільної здатності, потрібно враховувати кожний піксель кожного з трьох основних кольорів, а потім обраховувати і зберігати значення параметрів нейронів, що призводить до дуже великих потреб в пам’яті та обчислювальних можливостях. Для вирішення

спеціалісти з машинного навчання запропонували новий підхід до створення архітектури моделей глибокого навчання – згорткові нейронні мережі, вони же CNN.

3.3 Згорткові нейронні мережі

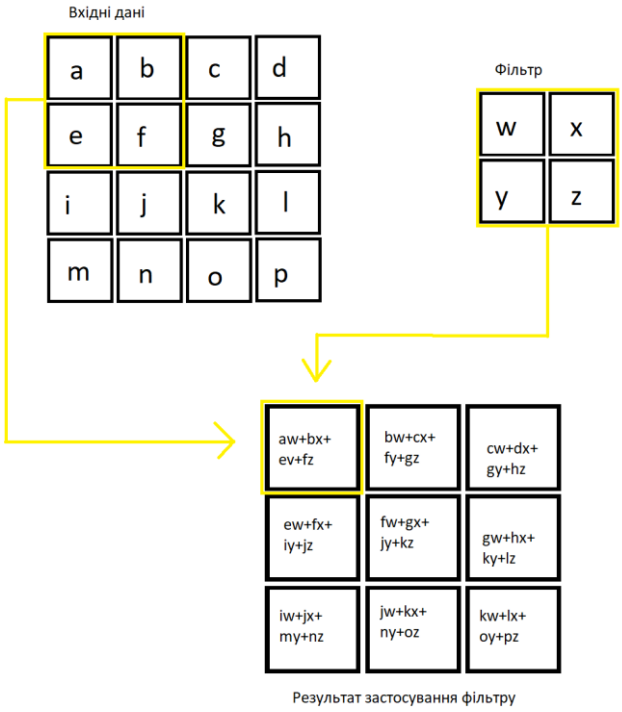


Рис.3.2

Згорткові нейронні мережі – це клас нейронних мереж, які використовують операцію згортування (Convolutional layer), замість класичного перемноження аргументів з матрицею ваг. Як показано на рисунку 3.2 певна матриця, яка називається фільтром, також в загальному випадку вона може бути довільних розмірів перших двох вимірів і мати третій, який буде рівним кількості каналів (основних трьох кольорів для зображень наприклад) вхідних даних. Третій вимір вихідної матриці залежить від кількості фільтрів застосованих до вхідної матриці. Параметри одного з таких фільтрів це розмір в трьох вимірах, значення елементів матриці фільтру, а також відступ та крок, перший – це значення, на яке потрібно збільшити матрицю вхідних даних по кожній з чотирьох сторін двох основних вимірів, нові клітинки заповнюються за замовчанням 0, другий – значення кроку на який потрібно буде посувати фільтр по матриці вхідних даних. Використовуючи ці параметри ми можем записати формулу розмірів вихідної матриці, якщо початково розмір матриці вхідних даних це $(X_h,\ X_w, n_C)$ і n_f

фільтрів розмірами ($f_h,\ f_w, n_C$), нехай результуюча матриця після перетворень з відступом p та кроком s буде мати розміри ($Y_h,\ Y_w, n_y$), де

$$Y_h = \lfloor (X_h - f + 2p)/s + 1 \rfloor$$

$$Y_w = \lfloor (X_w - f + 2p)/s + 1 \rfloor$$

$$n_r = n_f$$

Також в згорткових існує інший основний тип шарів, підвибірковий шар (Polling layer), які бувають двух підтипів максимальний та середній, на відміну від згорткового, вони не перемножують елементи матриці і фільтру, а просто, накладаючи фільтр на матрицю, обирають найбільше або рахують середнє значення у межах накладення фільтру, як це показано на Рис 3.3 для вхідних даних 4x4 і фільтру 2x2. Такий тип зазвичай відбувається після одного чи декількох згорткових шарів. В деяких колах цю дію не вважають за окремий шар, а поєднують разом з попереднім згортковим шаром, тому що в цьому фільтрі не має власних ваг і він лише перетворює вхідні дані. Формули результуючих розмірів для нього такі самі, як і в згорткового шару.

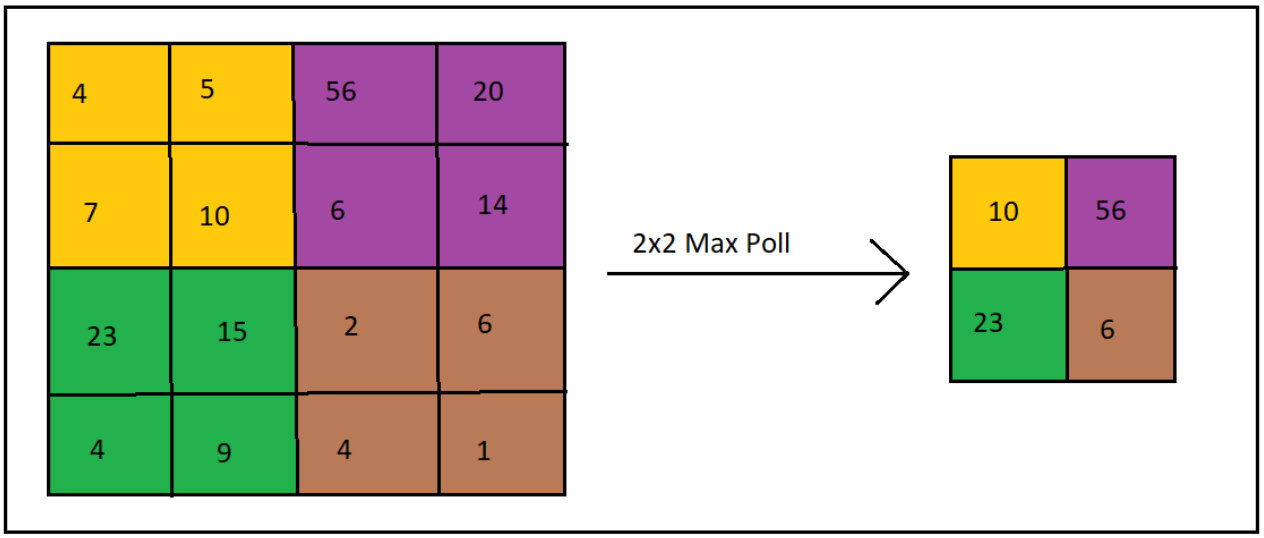


Рис 3.3

І останній тип повнозв'язний шар, це шар використовується як класифікатор, підготовлюючи дані для вихідного шару. По архітектурі він схожий на прихований шар з класичного машинного навчання.

РОЗДІЛ 4. ОПИС АРХІТЕКТУРИ СИСТЕМИ

4.1 Загальні положення архітектури

Система складається з двох основних складових частин. Перша частина є алгоритмом, який ми тренуємо на набору даних, для визначення об’єктів на зображенні та класифікації їх за певними параметрами. Вона аналізує кадри відеофайлу, визначаючи там наявні на зображеннях об’єкти і визначає приналежність їх до певних визначених раніше класів. Друга частина це алгоритм, який пов’язує об’єкти на різних зображеннях і пов’язує їх, як єдине ціле.

4.2 Архітектура моделі, яка визначає об’єкти на зображенні

За основу для проектування було взято модель darknet53, яка розроблена і використовується авторами рішення для визначення об’єктів на зображенні

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

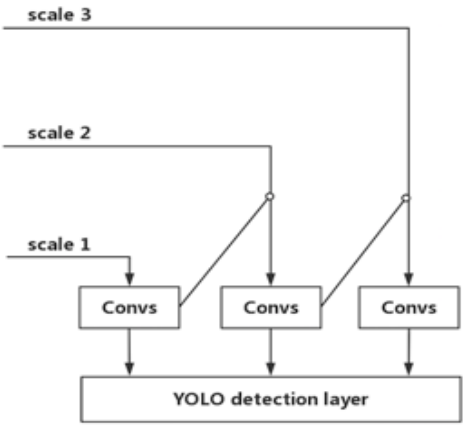


Рис. 4.1

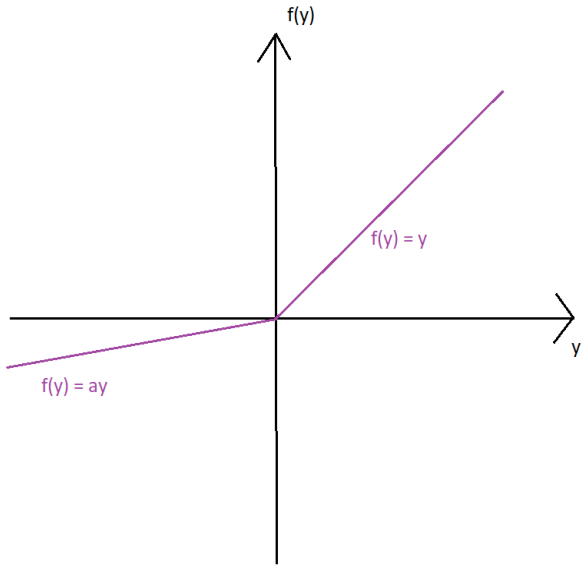


Рис. 4.2

YOLO, а саме структуру, яка зображена на Рис. 4.1. На ньому ми бачимо архітектуру моделі глибокого навчання згорткової нейронної мережі. Спочатку розберемо, що знаходиться всередині таблиці. Блок Convolutional виконує наступні дії, застосовує операцію згортки, для першого випадку це фільтри 3x3x3 кількістю 32, потім проводить нормалізацію вихідних після цієї операції даних та застосовує до них функцію активації Leaky ReLU, яка зображена на Рис. 4.2, зазвичай значення параметру a(з Рис. 4.2) для Leaky ReLU встановлюється як 0.01.

Інший блок, який зустрічається на Рис 4.1 це Residual. В свою він складається з декількох інших шарів, як показано на Рис. 4.3, а саме дублювання даних і до однієї копії ми застосовуємо операцію згортки, нормалізацію, функцію активації і потім ще раз згортку і нормалізацію, а потім зливаємо з іншою незміненою копією даних і потім застосовуємо функцію активації до цих злитих даних. Цей механізм дозволяє робити глибші нейронні мережі без втрати точності тренування, запобігаючи ситуації, в якій модель показує дуже гарні результати на

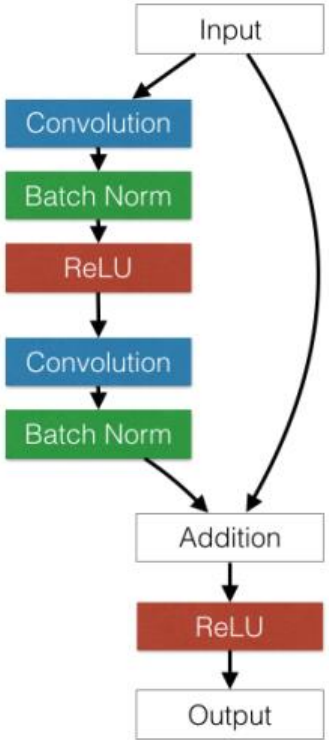


Рис. 4.3

даних для тренування, але на всіх інших показує незадовільні результати, які сильно відрізняються від точності на тренувальному наборі.

Під час проходження по всім шарам на Рис. 4.1 ми бачимо, що з нашої моделі тричі надаються вихідні дані (scale 1, scale 2, scale 3), це допомогає моделі краще розпізнавати об’єкти різних розмірів, scale 1 для малих об’єктів, scale 2 для середніх, scale 3 для великих. Вони також ще проходять декілька операції згортки в блоках Convs з Рис. 4.1.

Ці всі кроки підготовлюють вхідні дані до входу в класифікатор, надалі ми вираховуємо бокси, якими виділяються квадрати, тут ми застосовуємо декілька способів оптимізації, а саме Non-max Suppression та Anchor Boxes.

Однією з проблем алгоритмів визначення об’єктів на зображенні є неодноразове визначення одного і того самого об’єкту. Для того аби позбутися цього ми можемо застосувати метод Non-max Suppression, він аналізує бокси побудовані навколо одного і того самого об’єкту і вибирає тільки один той, якому алгоритм визначення надав найбільшу вірогідність. Це дозволяє нам вирішити проблему декількох передбачених моделлю боксів для одного і того самого об’єкту.

4.3 Архітектура алгоритму, який пов’язує об’єкти на різних зображеннях між собою для відслідковування

Нам не потрібні ReID алгоритми, бо вони за замовчуванням натреновані в основному на людей, а спеціалізованих для собак у відкритому доступі немає, тому за основу цього етапу розробки було взято алгоритм Sort, представлений у 2016 році.

Двома основними складовими алгоритму SORT є фільтр Калмана та угорський алгоритм, які і є архітектурною базою і ідеєю, на якій побудована реалізація алгоритму.

Фільтр Калмана – це лінійний фільтр, загальна ідея, якого буде в тому аби знайти баланс між врахуванням ваг нашого передбачення і нашого нового виміру на результат та зменшити вплив менш надійних джерел і збільшити вплив другого фактору. Цей фільтр має найменший середній квадрат помилки серед інших лінійних фільтрів.

І так ми розібралися, що фільтр Калмана допомогає нам фільтрувати наші вхідні дані, тепер ми можем перейти до угорського алгоритму. Це алгоритм оптимізації, який вирішує задачу про призначення. Суть задачі в тому аби оптимально розподілити завдання між виконавцями, маючи матрицю вартостей цих завдань для кожного з виконавців. Дві основні ідеї, а саме: по-перше, якщо у всіх елементів якогось рядку або колонки матриці вартостей вирахувати деяке однакове значення, то результуюча вартість зменшиться на це число, а оптимальне рішення залишиться тим самим, по-друге, якщо є рішення нульової вартості, то воно буде оптимальним.

Отже, його виконання доволі просте, спочатку проводимо редукцію по рядкам, вибираючи мінімальне значення в кожному з рядків і віднімаючи його від усіх елементів, далі проводимо редукцію по стовпчикам. Тепер намагаємося вибрати оптимальне рішення виділяючи по одному нулю в кожному стовпчику і рядочку. Якщо на цьому етапі це зробити неможливо, ми обираємо рядки і стовпчики так, аби викреслити всі нулі з найменшою кількістю викреслювань стовпчиків чи рядків, перекреслюємо ці рядки чи стовпчики, серед невикреслених клітинок обираємо найменше значення і віднімаємо від всіх невикреслених і додаємо до тих, які викреслені двічі. Повторюємо спробу обрати оптимальний розв’язок для кожного виконавця і кожної задачі.

РОЗДІЛ 5. ОЦІНКА РЕЗУЛЬТАТІВ РОБОТИ

5.1 Оцінка для зображень під різним ракурсом

Однією з основних проблем розпізнавання об’єктів на зображенні чи відео є незручний або неправильний ракурс, з якого знімається відео чи робиться фотографія. Інколи навіть людина не правильно визначити вид тварини дивлячись наприклад зверху.

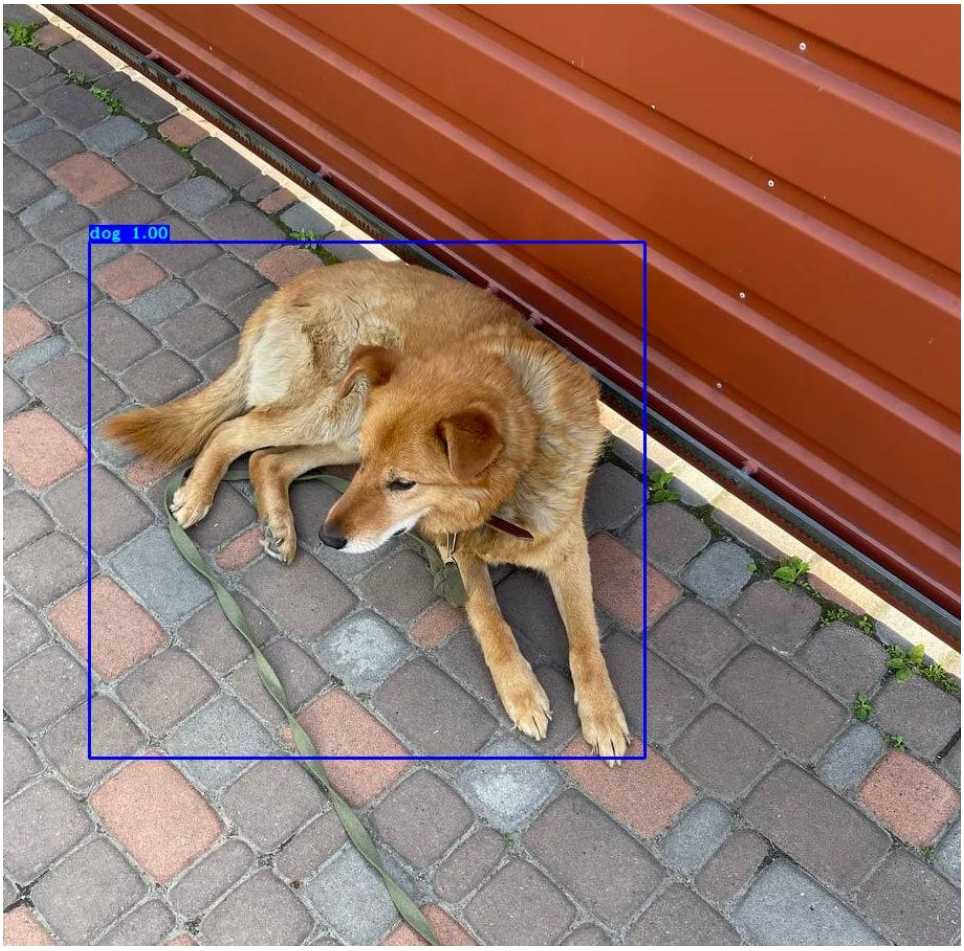


Рис. 5.1 Вид збоку

Ознайомившись з рисунками 5.1, 5.2 та 5.3, можемо порівняти значення впевненості алгоритму в визначенні об’єкту, що зображено в верхній лівій частині рамки об’єкту, як ми можемо побачити з першого зображення при

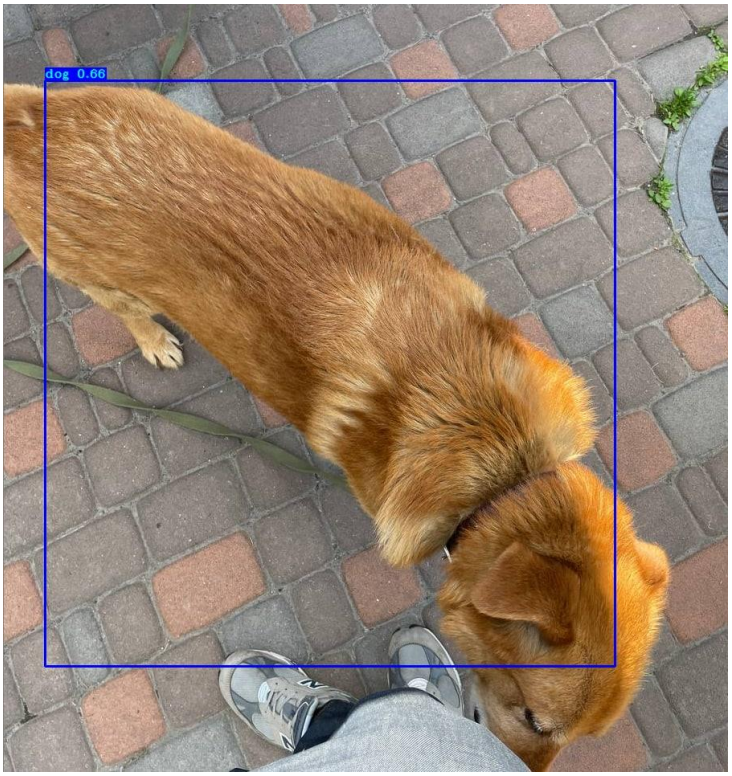


Рис 5.2 Вид зверху під кутом

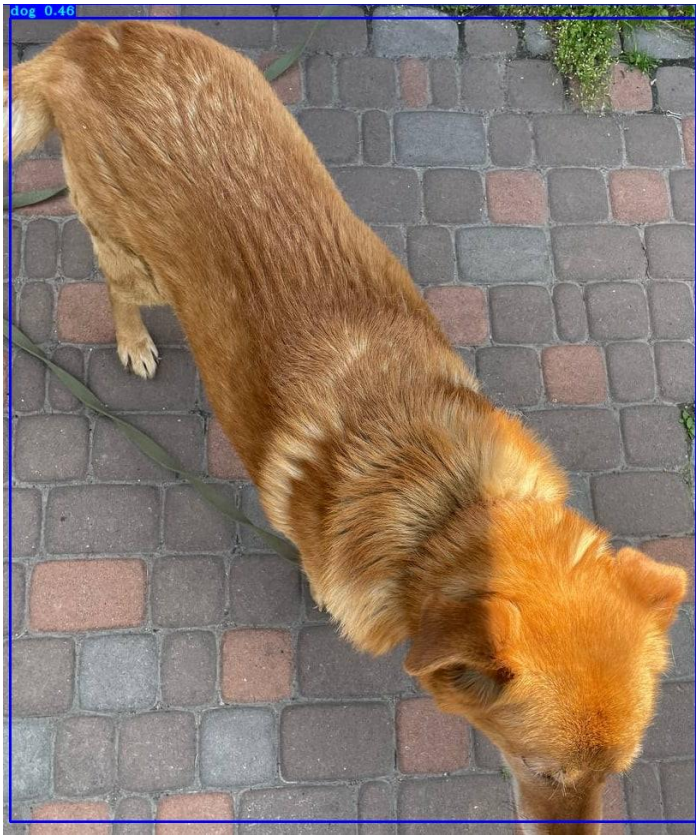


Рис 5.3 Вид зверху

фотографуванні збоку алгоритм гарно розпізнає клас об’єкту і впевнений на сто відсотків, що на фотографії зображена собака. На другому зображення собака зображена зверху під кутом, в цьому випадку алгоритм визначення зображення вже має меншу впевненість, а саме 66 відсотків. При фотографії зверху, як показано на третьому зображенні алгоритм показує доволі погані результати в 46 відсотків. При використанні його для аналізу відеофайлу в випадку зображення собаки зверху чи під іншими незручними кутами можливі погані результати відслідковування.

В цілому, ми бачимо, що найкраще алгоритм визначає собак збоку при гарному ракурсі на голову тварини.

5.2 Оцінка для зображень з декількома собаками

На рисунках 5.4 та 5.5 зображено роботу алгоритму визначення об’єктів на зображенні. Як ми можемо з рисунків на першому собаки сидять поруч, але не закривають один одного, тому алгоритм нормально розпізнав обох тварин с впевненістю в 100 відсотків, а на другому собаки знаходяться одна позаду іншої, що стало причиною неправильної оцінки зображення нейронною мережею і визначення лише однієї собаки з впевненістю 99 відсотків.

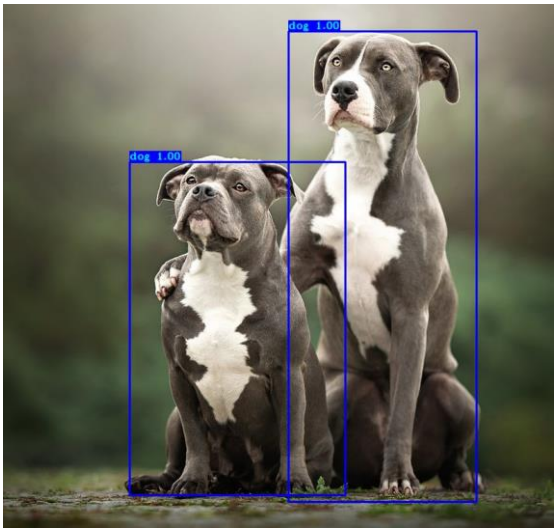


Рис. 5.4 Дві собаки поряд

Рис. 5.5 Дві собаки з іграшкою

На рисунках 5.6 та 5.7 показані знімки екрану при виведені результату відслідковування собак на відео після обробки системою відеофайлу. На рисунку 5.6 зображено роботу алгоритму з неправильним визначенням кількості об’єктів, що є наслідком того що, як і на рисунку 5.5 одна з собак знаходиться позаду іншої. В свою чергу на рисунку 5.7, хоча і дальня собака знаходиться майже повністю позаду іншої, система правильно визначає наявність двох собак. Це пов’язано с тим, що на відміну від ситуацією з двох нерухомими собака, ці собаки переміщаються і спочатку не затуляли одна одну, передбачаючи рух двох об’єктів алгоритм запам’ятав наявність дальньої собаки і не загубив її навіть після того як вона зайшла за ближчу. Що показує гарну роботу на рухомих об’єктах другої частини алгоритму з відслідковування, але погані результати роботи першої частини системи з визначення об’єктів на зображенні в таких випадках.

Рис. 5.6 Трекінг двух нерухомих собак

Рис. 5.7 Трекінг двух рухомих собак

5.3 Оцінка визначень собаки на проблемних для визначення породах або інших схожих тваринах

Третя основна проблема роботи системи, це обробка зображень чи відеофайлів, на яких зображенні собаки маленьких по розміру собак, які разюче відрізняються зовнішньо від переважної більшості інших порід або зображення інших тварин, які схожі на собак: вовки, гієни, тощо.

Як ми бачимо на рисунках 5.8 та 5.9 система гарно виконує аналіз собак з проблемними для визначення породами, такими як чіхуахуа на першому рисунку і йоршрським терьером на другій, не дивлячись на те що вони суттєво відрізняються від інших порід собак. В обох випадках алгоритм має впевненість у 99 відсотків, що є хорошим результатом.

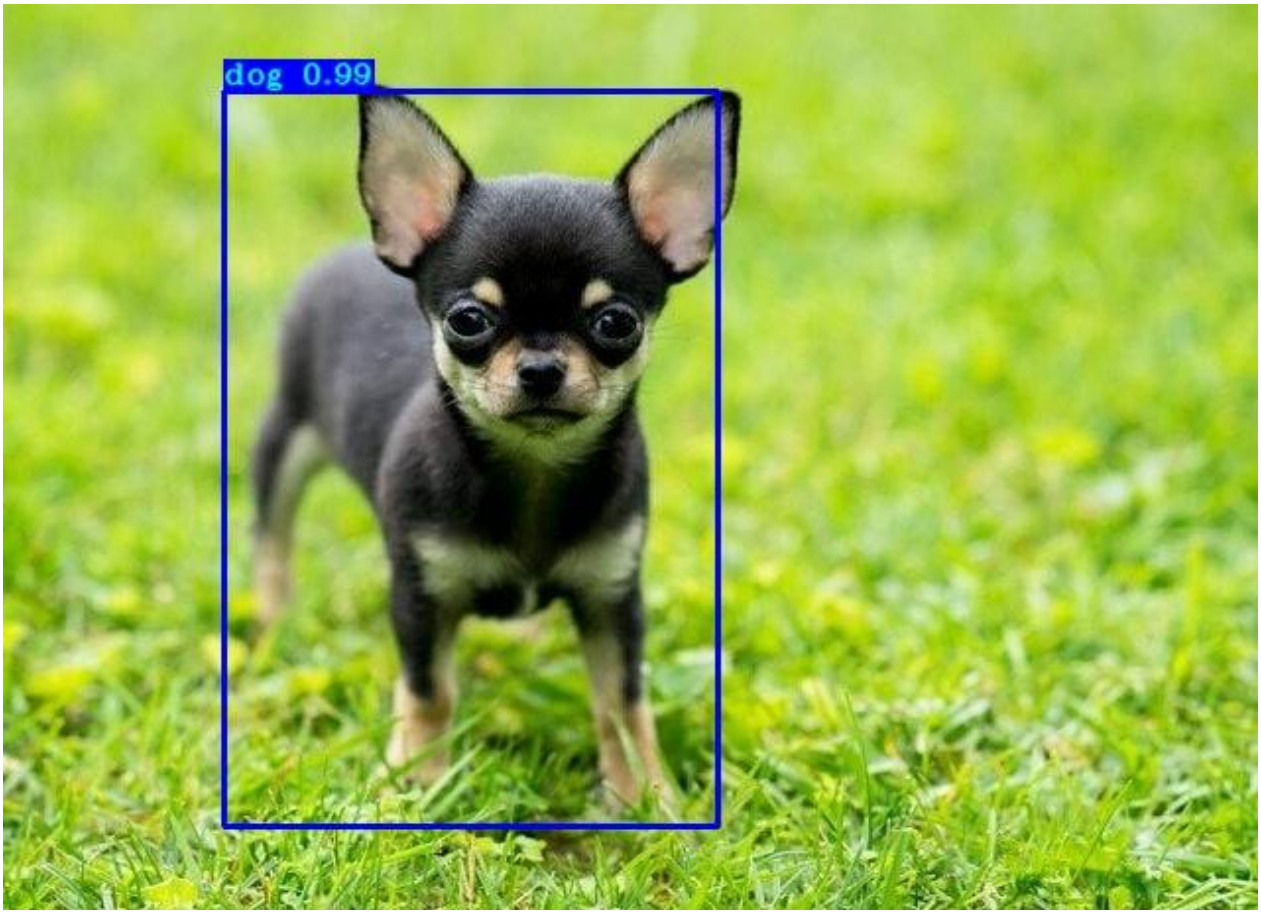


Рис. 5.8 Чіхуахуа

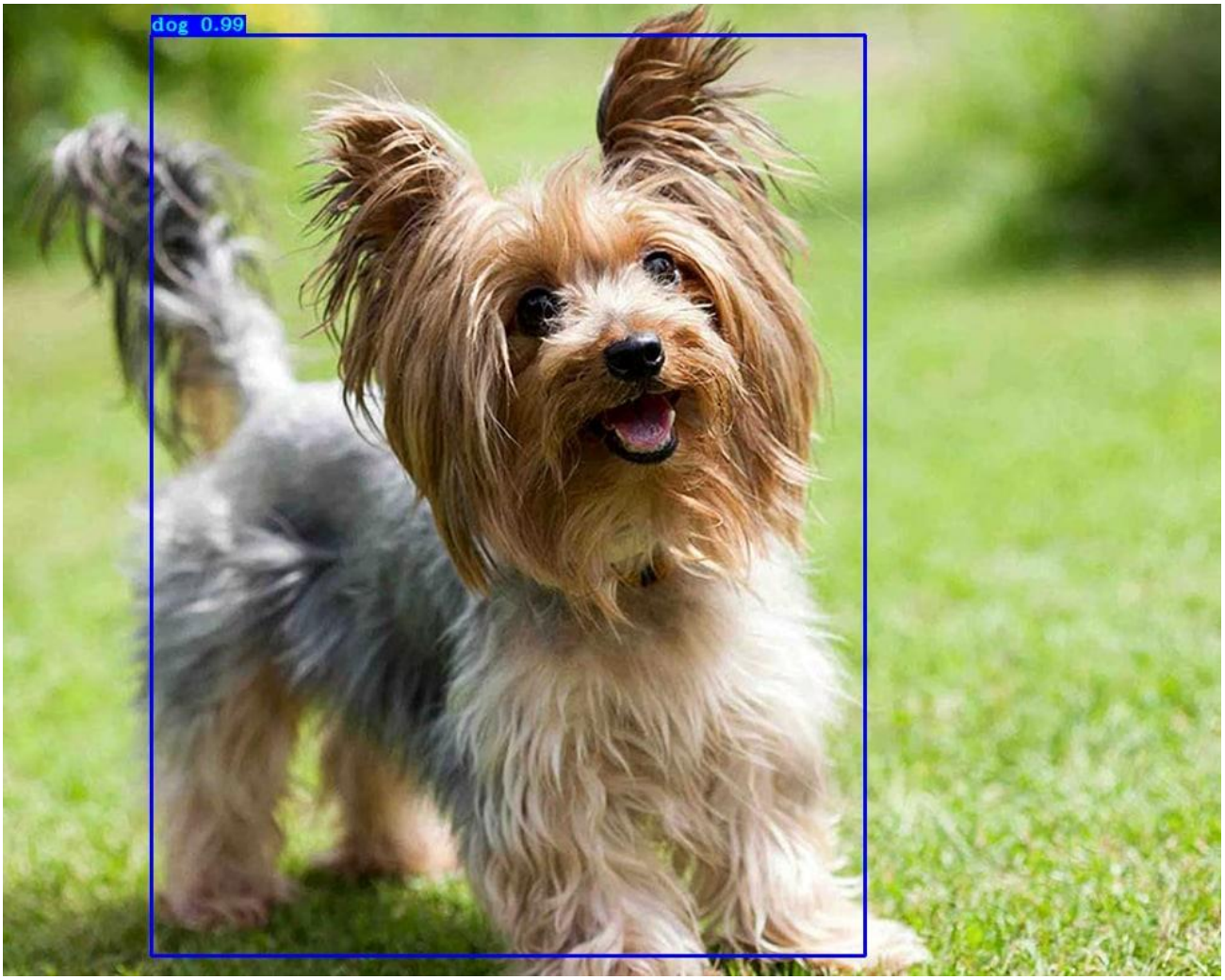


Рис. 5.9 Йоршірський терьер

Іншою проблемною ситуацією є визначення собак серед інших схожих на них тварин, наприклад на рисунках 5.10 та 5.11 зображено результат обробки в реальному часі відеофайлу з вовком. Як ми бачимо на рисунку 5.10 алгоритм спочатку не визначає вовка як собаку. Проте після того як він повернув голову до камери алгоритм перевизначив його як собаку. Насправді хоча і присутня помилка на другому зображенні, це вже доволі непоганим результатом, майже на рівні людини.

Рис. 5.10 Вовк в профіль

Рис. 5.11 Вовк в анфас

5.4 Трекінг собак на зображенні с іншого екрану

Інший цікавий випадок це аналіз системою адаптивного трекінгу у реальному часі відеопотоку, на якому знімається екран телефону, з відкритим зображенням, на якому є шуканий об’єкт.

В нашому випадку на рисунку 5.12 ми бачимо, що система знайшла та відслідковує собаку на зображенні з телефону, це звичайно помилковий результат, проте, це показує, що алгоритм гарно відпрацьовує навіть на зображеннях поганої якості або чіткості.

На рисунку ми бачимо, що всередині системи ця собака, як об’єкт, має 19 нумер, що пов’язано с тим, що це було знято на веб-камеру підключену до комп’ютера і номер знайденої собаки збільшувався кожний раз, коли в процесі зйомки була відведена камера або перегорнуто зображення в галереї, тому це не є помилкою алгоритму, а скоріше специфікою експерименту.

Рис. 5.12 Зображення екрану телефону

5.5 Загальна оцінка результатів

Загальна точність нейронної мережі визначення собаки на зображення з тестового набору COCO datasets була обраховано системою в 86.9 відсотків, що є доволі гарним результатом для алгоритмів комп’ютерного зору, але звісно є простір для покращення результату.

На складних для визначення об’єкту, як собаки, алгоритм показав себе добре, проте має проблеми з зображенням декількох собак, які візуально перекривають одна іншу.

ВИСНОВКИ

На сьогодні багато домашніх собак потрапляє на вулицю. Люди вимушені відпускати своїх улюбленців в зв’язку з великою кількістю випадків вимушених переїздів. Після вирішення глобальних проблем Україна може зіткнутися з проблемою великої кількості собак на вулицях міст, а також люди, які повернуться, будуть намагатися повернути собі свої собак. Система адаптивного трекінгу собак в реальному часі може допомогти відслідковувати собак та відповідно реагувати точніше і швидше.

Було розглянуто та проаналізовано існуючі системи машинного навчання доступних для загалу, які виконують визначення або відслідковування об’єктів на зображенні чи відео. Підсумовуючи аналіз, не було знайдено аналогів програмним засобам, які спеціалізуються на знаходженні собак на відео. Наведено загальний огляд та порівняння для існуючих технологій глибокого навчання і їх архітектурних особливостей.

Було спроектовано і реалізовано власну систему для відслідковування собак у реальному часі, використовуючи останні практики цієї сфери. Система показала гарну точність визначення собак на відео та їхнє подальше точне відслідковування, також вона має гарні показники точності в незручних для розпізнавання ситуаціях.

Для подальшого розвитку можлива розробка зручного інтерфейсу взаємодії, сайт чи додаток на смартфон. Також можливе вдосконалення буде програма, яка буде порівнювати зображення, надані людьми, що шукають своїх собак, з визначеними на відео собаками і знаходити співпадіння.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The deep learning specialization [Електронний ресурс] – Режим доступу до ресурсу: https://www.deeplearning.ai/program/deep-learning-specialization/
2. Neural networks and Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://cs230.stanford.edu/syllabus/
3. Simon, J. A New Approach for Filtering Nonlinear Systems / J. Simon // Proc. of the American Control Conference, Seattle, USA, Jun, 1997 — pp. 808 - 815.
4. The complete guide to object tracking. [Електронний ресурс] – Режим доступу до ресурсу: https://www.v7labs.com/blog/object-tracking-guide
5. Long, J.; Shelhamer, E. & Darrell, T. (2014), Fully convolutional networks for semantic segmentation
6. Everithing you ever wanted to know about computer vision. [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e
7. Bradski G. Learning OpenCV. Computer Vision with the OpenCV Library / G. Bradski, A. Kaehler. — O'Reilly Media, Inc., 2008. — 580 с.
8. Kalman Filter Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.kalmanfilter.net/default.aspx
9. Kálmán R.E. A New Approach to Linear Filtering and Prediction Problems / Rudolf Emil Kálmán — Journal of Basic Engineering, 1960. — pp. 35 - 45.
10. Anchor Boxes [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9
11. Yilmaz A., Javed O., Shah M. Object Tracking: A Survey // ACM Comput. Surv, 2006. V. 38 P. 13.
12. Comaniciu D., Ramesh V., Andmeer P. Kernel-based object tracking // IEEE Trans. Patt. Analy. Mach. Intell. 25, 2003. P. 564-575.
13. M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In Computer Vision– ECCV 2008, pages 2–15. Springer, 2008.
14. You Only Look Once: Unified, Real-Time Object Detection [Електронний ресурс] – Режим доступу до ресурсу: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf