МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів детектування в розумних системах паркування
(тема)

Виконав:
Студент 2 курсу, групи ІПЗздм-21-1
Чуприна С. О.
(прізвище, ініціали)

Спеціальність 121-Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Голян Н. В.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ Дудар З. В.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121-Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.кафедри _____
(підпис)
« ____ » _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Чуприни Сергія Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів детектування в розумних системах паркування

затверджена наказом університету від 03.04.2023р. № 83Ст

2. Термін подання роботи до екзаменаційної комісії 12 05 2023р.

3. Вихідні дані до роботи детектування в розумних системах паркування, детектування автомобілів, інтелектуальне паркування, пояснювальна записка.

4. Перелік питань, що потрібно опрацювати в роботі мета роботи, дослідження методів детектування, аналіз предметної галузі, постановка задачі, порівняння і вибір алгоритму детектування, розробка демонстраційного програмного додатку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Титульний слайд, Актуальність дослідження, Мета дослідження, Постановка

завдання, Огляд сучасних досліджень, Методи детектування, Аналіз предметної області, Огляд методів виявлення, Двоступеневі алгоритми виявлення, Одноступеневі алгоритми виявлення, Порівняння алгоритмів детектування, Похибки алгоритмів детектування, Розробка програмного продукту, Вимоги програмного програмного продукту, Взаємодія апаратного забезпечення, База даних програмного продукту, Інтерфейс користувача, Висновки.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спец. частина	доц. Голян Н. В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вивчення наявної літератури	24.01.2023 - 05.02.2023	виконано
2	Вивчення технології	06.02.2023 - 11.02.2023	виконано
3	Збір вхідних даних	12.02.2023 - 05.03.2023	виконано
4	Проведення експериментів	06.03.2023 - 31.03.2023	виконано
5	Підготовка пояснювальної записки	01.04.2023 - 05.04.2023	виконано
6	Підготовка презентації та доповіді	06.04.2023 - 30.04.2023	виконано
7	Нормоконтроль, рецензування	01.05.2023 - 04.05.2023	виконано
8	Занесення диплому в електронний		
	архів	05.05.2023 - 07.05.2023	виконано
9	Допуск до захисту у зав. кафедри	08.05.2023 - 09.05.2023	виконано

Дата видачі завдання 23 січня 2023р.

Студент _____ (підпис)

Керівник роботи _____ доц. Голян Н. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 81 с., 29 рис., 2 табл., 35 джер.

КОМП'ЮТЕРНИЙ ЗІР, РОЗУМНЕ ПАРКУВАННЯ, СИСТЕМИ ДЕТЕКТУВАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ВИЯВЛЕННЯ ОБ'ЄКТІВ, OPENCV, SSD, YOLO.

Об’єктом дослідження є алгоритми виявлення автомобілів на місцях для паркування за допомогою комп’ютерного зору на основі машинного навчання.

Метою роботи є визначення найбільш ефективного методу детектування в розумних системах паркування.

Методи розробки базуються на таких технологіях, як Python, OpenCV, NumPy.

В результаті роботи було досліджено методи детектування автомобілів на місцях для паркування. Проведений аналіз та порівняння алгоритмів які використовують комп’ютерний зір для виявлення предметів на зображенні. Та їх експериментальне дослідження. Запропоновано методи визначення наявності вільних місць на міських та приватних паркуваннях. Реалізовано програмний продукт для допомоги водіям у пошуку місця для паркування.

За темою кваліфікаційної роботи були опубліковані англомовні тези доповіді на 27-му Міжнародному молодіжному форумі «Радіоелектроніка і молодь у XXI столітті».

COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORK, DETECTION SYSTEMS, OBJECT DETECTION, OPENCV, SMART PARKING, SSD, YOLO.

The object of research is algorithms for detecting cars in parking spaces using computer vision based on machine learning.

The aim of the study is to determine the most effective detection method in smart parking systems.

The development methods are based on technologies such as Python, OpenCV, and NumPy.

As a result of the work, methods for detecting cars in parking spaces were investigated. The analysis and comparison of algorithms that use computer vision to detect objects in the image. And their experimental study. Methods for determining the availability of free spaces in public and private car parks are proposed. A software product was implemented to help drivers find a parking space.

On the topic of the qualification work, the English-language abstracts of the report were published at the 27th International Youth Forum "Radio Electronics and Youth in the XXI Century".

Умови публікації пояснювальної записки

Я, Чуприна Сергій Олександрович, студент гр. ІПЗм-21-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів детектування в розумних системах паркування», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Більше половини населення планети мешкає у містах (55%), за прогнозами ООН ця частка зросте до двох третин (68%) вже у середині сторіччя [1]. Сучасні мегаполіси дозволяють проживати мільйонам людей на порівняно малих за площею територіях. Коли рівень автомобілізації у розвинених країнах Європи складає від 500 до 600 автомобілів на 1000 жителів [2].

В Україні майже кожен четвертий має автомобіль і користується ним регулярно. Тому пошук вільного місця для паркування перетворився на щоденну рутину яка віднімає дорогоцінний час та сили. Постає питання де залишити автомобіль в місцях скупчення ділової та комерційної активності, у дворах багатоповерхівок і біля визначних пам’яток. Навіть пошук вільного місця у підземному або багаторівневому паркінгу може перетворитись на справжню пригоду.

Проблема лише посилюється якщо брати до уваги те, що близько 30% щоденних заторів спричинені тим, що водії шукають вільне місце для безкоштовного паркування [3]. Додаймо до цього очевидні негативні наслідки – нераціональне споживання палива, збільшення рівня викидів, підвищення аварійності, зменшення середньої швидкості трафіку.

Вирішити цю проблему допомагають системи інтелектуального паркування. На сьогоднішній день існує безліч комерційних та відкритих систем для цього. Всі вони базуються на різноманітних методах та принципах розпізнавання зайнятого та вільного місця. В переважній більшості вони інформують водія про наявність вільних місць безпосередньо перед в’їздом. Не дають повної інформації про наявні місця, наприклад на якому поверсі воно знаходиться та як оптимально туди потрапити.

Зростаючий попит на послуги паркування є однією з основних причин, чому розвиток інтелектуальних систем паркування має велике значення. Необхідно модернізувати існуючі та створювати нові методи виявлення, щоб задовольнити

зростаючий попит. Крім того, інтелектуальні системи паркування можуть зробити значний внесок у нормалізацію дорожнього руху у великому місті. Цього можна досягти, надаючи в режимі реального часу інформацію про вільні місця для паркування та програмні додатки для швидкого пошуку паркомісць.

Інтелектуальні системи паркування пропонують безліч переваг як для кінцевого користувача, так і для міст. З боку кінцевого користувача, програмні додатки для швидкого пошуку місця для паркування можуть зменшити стрес і допомогти більш ефективно організувати маршрути пересування. З боку міста, інтелектуальні системи паркування можуть зробити значний внесок у нормалізацію дорожнього руху у великому місті та покращити якість і ефективність послуг паркування.

Метою дослідження є вивчення існуючих методів виявлення, що використовуються в системах розумного паркування, визначення потенційних удосконалень і нових рішень для більш ефективного і надійного виявлення автомобілів в таких системах. Дослідження ефективності різних методів виявлення та їх сумісність з існуючими та новими технологіями. Актуалізація проблеми.

Предметом дослідження є методи виявлення, що використовуються в системах розумного паркування.

Об'єктом дослідження є вивчення існуючих методів виявлення, що використовуються в системах розумного паркування, а також виявлення потенційних поліпшень і нових рішень для більш ефективного і надійного виявлення автомобілів.

Наукова новизна цього дослідження полягає у вивченні потенційних удосконалень та нових рішень для більш ефективного та надійного виявлення автомобілів у системах розумного паркування.

Практична цінність дослідження полягає в кращому розумінні існуючих методів виявлення, що використовуються в системах розумного паркування, і в розробці більш ефективних та надійних рішень для виявлення автомобілів.

За темою дипломної роботи було опубліковано тези для участі у Міжнародному молодіжному форумі «Радіоелектроніка і молодь у XXI столітті».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд сучасних досліджень

Розглянемо дослідницькі роботи, пов’язані з інтелектуальними системами паркування автомобілів.

Зазвичай водії притримуються маршруту до свого місця призначення, а після курсують від однієї парковки до іншої або узбіччя вулиць в надії залишити там свій автомобіль витрачаючи час та інші ресурси. З розвитком технологій, таких як смартфони та камери відеоспостереження було запропоновано декілька автоматизованих методів, які допомагають пришвидшити пошук вільного місця для паркування.

Ольга Павлова та інші [4] запропонували метод розпізнавання зображень для розумного паркінгу на основі згорткової нейронної мережі. Запропонований прототип відповідає за отримання зображення з камери, визначення чи наявні на ньому вільні та зайняті місця для паркування та відправки цієї інформації до спеціальної платформи.

Камран Саттар Авайсі та інші [5] запропонував архітектурне рішення вирішення проблеми розумного паркування з використанням туманних технологій. Запропонована технологія складається з трьох рівнів: камери та мікроконтролери, туманний рівень, хмарний рівень. Камери в запропонованій архітектурі відповідають за захоплення зображення. Мікроконтролер є мостом між камерою та вузлом. Туманний вузол отримує зображення яке згодом передає на хмарний сервер та контролює інформаційні дисплеї. Також він має змогу спілкуватися з сусідніми вузлами з метою знаходження вільного місця у випадку коли всі місця які обробляє цей вузол зайняті. Всі розрахунки виконує хмарний сервер.

Новосад Марія-Руслана [6] розробила асистент паркування який є одним з модулей системи розумного міста. Ця система складається з трьох модулів: зберігання та обробка запитів, коректне відображення даних, обробка зображення. Зображення з камер обробляється за допомогою алгоритмів штучного інтелекту.

Перевагою цієї системи над іншими є веб інтерфейс зрозумілий для кінцевого користувача.

Леа Дуїц Родіч та інші [7] дослідили нову концепцію визначення наповненості місця для паркування на допомогою вимірювання сигналу з енергоефективних датчиків. В якості датчиків використовувалися автономні сенсори розроблені таким чином щоб передавати сигнал за допомогою технології LoraWAN який сприймають базові станції. Базові станції передають дані про рівень сигналу на сервер де і відбуваються розрахунки.

1.2 Огляд існуючих технологій з різними методами детектування

В цьому розділі порівняємо різні методи детектування автомобілів на місці для паркування. З розвитком технологій також розвивались технології для адміністрування та керування комерційними та муніципальними паркінгами. В контексті міста ці зміни були обумовлені також непоодинокими випадками самовільного захоплення парко-місць з метою незаконної їх реалізації [8].

Місця для паркінгу можна розділити на два типи:

– в межах вулиці (узбіччя доріг, майданчики позначені спеціальною розміткою та/або знаками в межах дороги або на тротуарі);
– поза межами вулиці (одно- або багаторівневі паркінги торгових центрів, житлових масивів тощо).

За методами детектування вільних та зайнятих місць на три [9]:

– наземні датчики (вбудовані в поверхню або накладні);
– надземні датчики (встановлюються на стелях, опорах вуличного освітлення або спеціальних конструкціях), до цього типу також віднесемо технологію засновану на камерах відеоспостереження;
– на основі лічильника.

Системи детектування з датчиками, відповідно, можна класифікувати ще на 12 типів [10]:

– з інфрачервоним випромінюванням;

– працюючі через стільникову мережу;

– магніторезистивні датчики;

– акустичні матриці;

– виявлення світла та вимірювання дальності (LiDAR);

– радіовиявлення та визначення дальності (RADAR);

– магнітометри;

– агенти або волонтери;

– радіочастотна ідентифікація (RFID);

– ультразвукові;

– на основі індуктивної петлі;

– відео камери.

1.2.1 Метод детектування на основі лічильника

Встановлені на в’їздах та виїздах паркомати підраховують скільки автомобілів знаходиться на території паркінгу. Ці лічильники зазвичай мають вбудований сенсор, що реагує на в'їзд та виїзд автомобілів зі стоянки. Коли автомобіль заїжджає на стоянку, лічильник реєструє це як одну прибуттєву операцію, а коли автомобіль виїжджає, лічильник реєструє це як одну вибуттєву операцію. Після чого порівнюють результат з загальною кількістю місць для паркування та за допомогою електронного табло повідомляють водієві кількість вільних. Попри свою простоту та невисокі фінансові витрати по впровадженню цю систему складно віднести до інтелектуальних та сучасних. Водій не отримує інформації про знаходження вільних місць для паркування та часто трапляється таке, що лічильник вказує на від’ємну кількість таких.

Рисунок 1.1 – Лічильник вакантних місць

Отже, хоча система лічильників та паркоматів на паркінгах є корисною, проте є можливість вдосконалювати та використовувати більш сучасні технології для забезпечення кращої організації та зручності для водіїв.

1.2.2 Метод детектування на основі датчиків

Такі системи дедалі частіше інтегруються у наявні та проектуються в нових комерційних паркінгах. Вони дозволяють спростити життя водіям та надають комплексні бізнес рішення для власників паркінгу. Розглянемо одну з таких систем детальніше.

Міжнародна компанія Smart Parking [11] постачальник систем для розумного паркування. Поставляє перевірені рішення як для клієнтів парковок так і для адміністраторів. Метод детектування в продукті компанії заснований на наземних, надземних та накладних автономних датчиках. Датчики з інфрачервоним випромінюванням поєднані в меш-мережу та передають інформацію про стан місця для паркування на шлюз який працює через стільникову мережу.

Шлюз керується і передає зібрану інформацію на хмарну платформу де і відбувається адміністрування паркінгу. Крім цього може керувати інформаційними дисплеями для допомоги водіям, паркоматами для здійснення оплати, розумним освітленням, визначати якість повітря, навіть роздавати публічний Wi-Fi.

Система використовує машинний зір для того щоб брати безконтактну дистанційну плату з клієнтів та для інспектування.

Також компанія надає зручну хмарну платформу та програмне забезпечення для адміністрування та розробки. Клієнтське програмне забезпечення доступне на IOS та Android та надає розгорнуту інформацію про стан місць для паркування, допомагає побудувати оптимальний маршрут до вільного місця та здійснити оплату чи бронування.

Система гарно себе зарекомендовала та вже більше 7 років працює на вулицях та комерційних паркінгах Сполученого Королівства, Німеччини, Австралії та Нової Зеландії. Легко інтегрується в систему сучасного розумного міста.

Рисунок 1.2 – Датчик у дорожньому полотні

Незважаючи на це система непозбавлена слабких сторін. Наземні датчики з інфрачервоним випромінюванням можуть хибно спрацьовувати через бруд або несподівані перешкоди. Термін служби датчиків до 10 років, що порівняно із сучасними камерами відеоспостереження, але кількість значно більша бо

необхідно встановити датчик на кожному місці для паркування. Також датчики вбудовані в поверхню дорожнього покриття пришвидшує її ерозію та скорочую час її експлуатації. Камери відеоспостереження встановлені лише для автоматичної детекції автомобільних номерних знаків та не задіяні у системах безпеки.

1.2.3 Метод детектування за допомогою комп'ютерного зору

На сьогодні системи засновані на розпізнаванні образів є лише додатками у більшості проектів. Так наприклад, автоматичне розпізнавання номерних знаків (ANPR) використовується в незначній кількості систем інтелектуального паркування. Переважна більшість паркінгів використовує паркомати з паперовими квитками або RFID картками. Але безліч концептів та приватних проектів вказують на те, що це може бути одна з епох розвитку систем інтелектуального паркування. Розглянемо одну з таких систем детальніше.

Адам Гейтгей запропонував метод пошуку вільного місця для паркування за допомогою глибокого навчання [12]. Цей проект працює наступним чином: відео-камера знімає вулицю на якій знаходяться місця для паркування, програмне забезпечення аналізує відео-потік і визначає де знаходяться місця для паркування, якщо місце звільняється відправляє повідомлення разом із зображенням на котрому позначене місце що звільнилося.

Для аналізу зображень і виділення на ньому автомобілей використовується згорткова нейронна мережа Mask R-CNN [13]. Програма знаходить на окремих кадрах відео автомобілі, визначає автомобілі які не рухаються певний час. Після робить висновок, що там знаходиться місце для паркінгу. Коли автомобіль зникає з сектору що був запам'ятований програма надсилає повідомлення про вільне місце і його зображення.

Прототип цікавий своїм алгоритмом визначення місця для паркування. В поєднані з високоефективною нейронною мережею він дозволяє з високою

точністю визначати таке місце навіть якщо розмітка відсутня або не може бути розпізнана.

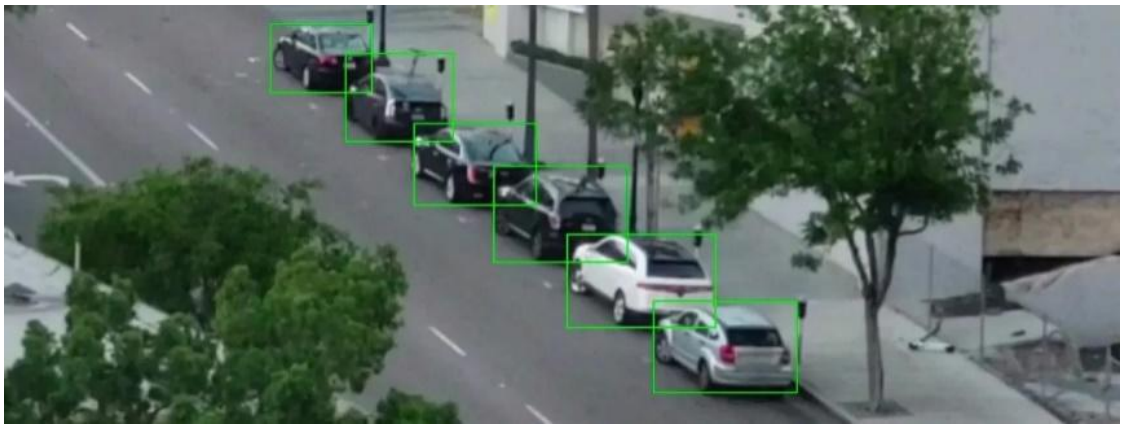

Рисунок 1.3 – Результат розпізнавання автомобілів

Не дивлячись на простоту і ефективність система має безліч недоліків. Одним з головних є саме алгоритм визначення місця для паркування. Автомобіль може бути припаркований у місці де це не дозволено, програмне забезпечення не знає що цей автомобіль систематично отримує штрафи за неправильне паркування або був евакуйований. Наступна проблема - великі затрати ресурсів на обробку кожного кадру відео. Сучасні камери відеоспостереження мають змогу повідомляти сервер коли спостерігається рух у певних областях кадру. Без оптимізації неможливе адекватне масштабування. Останнє – СМС повідомлення не кращий інструмент для інформування, хоча знімок вільного місця додає наочності. Сповіщення з GPS координатами місця мають бути зручніші.

1.3 Аналіз предметної області

EasyPark – find & pay parking – це додаток для смартфонів, який допомагає знайти та оплатити парковку [14]. Цей додаток був розроблений EasyPark Group, компанією, що спеціалізується на розробці рішень для автомобільної

промисловості. EasyPark є одним з найпопулярніших додатків для паркування в Європі і використовується в більш ніж 1700 містах.

EasyPark використовує GPS, та дозволяє користувачам знаходити парковку саме там, де їм потрібно. Додаток також використовує дані про наявність вільних місць для паркування, які отримуються з датчиків та камер встановлених на дорогах та автостоянках. Також датчики LiDAR або камери встановлюються на автомобілі волонтерів та дозволяють оперативно оновлювати інформацію про зайнятість паркінгів, виявляти правопорушення та зберігати дані для подальшої обробки.

Крім того, EasyPark дозволяє користувачам сплачувати за паркування безпосередньо через додаток, це забезпечує зручність та ефективність.

Рисунок 1.4 – Робоче вікно додатку EasyPark – find & pay parking

Однією з переваг EasyPark є широке покриття міста та автостоянок, що дозволяє користувачам знаходити парковку поблизу свого місця розташування в будь-якому місті. Крім того, додаток має простий та інтуїтивно зрозумілий інтерфейс, що робить його легким у використанні. Продовжити послугу

паркування автомобіля можливо прямо з додатку, не повертаючись до автомобіля. Також легко відслідковувати свої витрати.

Однак EasyPark має і певні недоліки. Наприклад, датчики або камери, встановлені на дорогах і паркінгах, працюють не у всіх містах, тому інформація про наявність вільних місць може бути неактуальною. Крім того, додаток не завжди правильно визначає місцезнаходження користувача, що може призвести до неправильного вибору парковки або помилки в оплаті послуги, що може призвести до штрафу.

EasyPark – find & pay parking є корисним додатком для тих, хто часто користується автомобілем і шукає швидкий та зручний спосіб знайти та оплатити парковку. Однак користувачам слід бути обережними і перевіряти інформацію про наявність вільних місць та правильність їх розташування.

Parkinto від Smartiple це паркувальна система яка здатна визначати зайнятість паркувального місця лише на основі зображення. Додаток використовує передові алгоритми штучного інтелекту та розпізнавання зображень. Parkinto – це хмарний сервіс де будь-яка камера може бути легко підключена до системи [15].

Компанія орієнтується в основному на власників паркувальних місць. Пропонуючи можливість спостерігати і контролювати паркувальні місця та надавати дані на веб-сторінки або інші системи, що належать клієнту. Типові випадки використання: муніципальні парковки, готелів, торгових центрів та офісів. Крім того для детектування доступні порти, що полегшує життя операторам та капітанам човнів.

Smartiple це молода "стартап"-компанія, яка прагне винаходити нові розумні рішення, що допомагають вирішувати "маленькі" проблеми сучасного світу.

До переваг системи належить легкість розгортування. Не усюди можливо встановити датчики або інші системи детектування транспортних засобів. Швидкість оновлення інформації, заявлений час – усього одна секунда. Легкість масштабування, одна камера здатна розрізняти до 90 транспортних засобів, вартість її встановлення порівняно мала. Індивідуальне налаштування – користувач

легко зможе змінити конфігурацію автостоянки чи додати інші програмні продукти.

До недоліків слід віднести залежність від якості обладнання, особливо роздільної здатності зображення. Ефективність буде сильно залежати від роздільної здатності камер та погодних умов. Значні ризики втратити контроль над значною частиною інфраструктури через вихід з ладу лише однією камери. Оскільки це хмарний сервіс, то є ризик втрати кофеденційної інформації через кібератаку.

У загальному, Parkinto від Smartiple – це цікавий інноваційний продукт, який вирішує більшість проблем, пов'язаних з паркуванням. Який може бути корисним для власників паркувальних місць, а також операторів порту. Проте, перед використанням системи, потрібно ретельно вивчити її переваги та недоліки, а також визначити, чи підходить вона для конкретної ситуації.

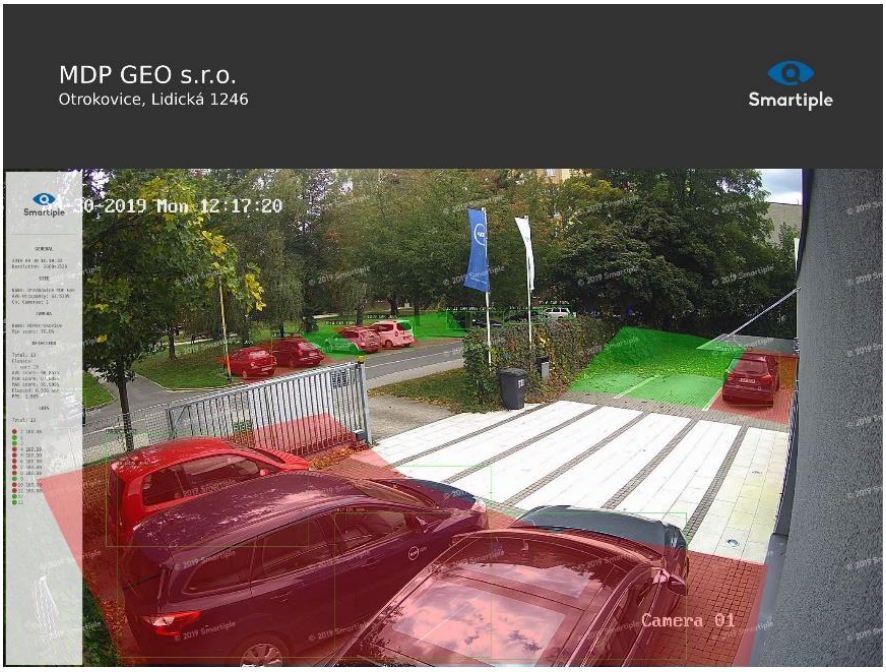

Рисунок 1.5 – Приклад роботи системи Parkinto

Cleverciti – це німецька компанія, що спеціалізується на розробці та виробництві інноваційних рішень для управління парковками [16]. Її продукти використовують технології машинного навчання, датчики, камери та хмарні

технології, щоб забезпечити точне та ефективне управління парковками у режимі реального часу.

Одним з ключових продуктів Cleverciti є Cleverciti Sensor – датчик, що встановлюється на стовпах або стінах парковок та здатний точно визначати наявність вільних місць для паркування [17]. Датчики збирають дані про стан парковки та передають ці дані до хмарної платформи Cleverciti. За допомогою машинного навчання та аналізу даних, Cleverciti може забезпечити точну та надійну інформацію про наявність вільних місць для паркування в реальному часі.

Рисунок 1.6 – Датчик Cleverciti Sensor на опорі вуличного освітлення

Подвійні оптичний датчик з високою роздільною здатністю 4K може охоплювати до 100 паркувальних місць і може бути легко встановлений на існуючій інфраструктурі, наприклад, на ліхтарних стовпах, будівлях або щоглах. Вони мають кут огляду до 2*128 градусів та вимірюють точне GPS-позиціонування та розмір відкритих паркувальних місць.

До переваг системи керування автостоянками Cleverciti слід віднести точність та ефективність. Датчики та технології компанії дозволяють точно обробляти велику кількість інформації. Швидкість детектування за допомогою

Cleverciti Sensor – 3 секунди. Конфіденційність – детектування відбувається "на периферії", і лише метадані залишають датчики. Економія – датчики встановлюються на вже існуючу інфраструктуру та потребують лише живлення.

Проте є і недоліки до яких можна віднести насамперед вартість. Незважаючи на необхідність лише встановити датчики та приєднати їх до хмари для малих за розміром стоянок вартість обладнання може бути занадто високою. До того ж датчики виконують лише функцію детектування, для інших цілей знадобиться окреме обладнання. Залежність від збоїв в інфраструктурі електропостачання та відсутності підключення до інтернету. Так як система використовує хмарні технології треба подбати про надійність живлення та мережі.

Узагалі Cleverciti є ефективним та інноваційним рішенням для бізнесу, яке використовує передові технології та аналітику для поліпшення управління парковками та зменшення трафіку в містах. Перед впровадженням цієї системи необхідно зважити всі переваги та недоліки, а також оцінити вона зможе відповідати потребам та ресурсам організації.

1.4 Постановка задачі та вибір оптимальної технології реалізації

Розумні системи паркування стають дедалі популярнішими завдяки своїй здатності зменшити трафік і покращити умови надання послуг. Однак існує потреба в підвищенні точності та ефективності методів виявлення, що використовуються в цих системах. Метою цього дослідження є вивчення існуючих методів виявлення та розробка нових методів для підвищення точності та ефективності систем розумного паркування.

У кожного методу є свої переваги та недоліки. Методи з використанням датчиків на сьогодні є найбільш досконалими та уживаними. Проте їх головним недоліком є складність з побудовою інфраструктури та її обслуговуванням. З економічної точки зору вони інколи занадто затратні. Натомість сучасне розумне

місто з кожним днем збільшує мережу камер відеоспостереження без яких неможливе забезпечення таких речей як безпека, організація дорожнього руху тощо. Тому наступною віхою у виборі методів детектування для розумних систем паркування можуть бути методи з використанням машинного навчання.

Трьома найважливішими завданнями машинного зору є класифікація, виявлення об’єктів та сегментація зображення [18].

Класифікація відповідає на запитання чи присутні на зображені певні об’єкти. В той самий час локалізація, яка є окремим завданням класифікації, відповідає на питання – де на зображенні знаходиться той чи інший об’єкт.

Виявлення об’єктів поєднує ці два завдання – класифікувати та локалізувати об’єкт, тобто знайти схожі об’єкти на зображенні чи відео та вказати на них.

Сегментація зображення виявляє та класифікує об’єкти, але на відміну від вищеперерахованих методів надає найбільш детальну інформацію про об’єкт окреслюючи його потенційно значущі області для подальшої обробки.

Вибір між виявленням об’єктів і сегментацією зображення залежить лише від мети роботи. У випадку коли необхідно лише виявити присутність об’єкта на зображенні можна скористатися будь-якою моделлю, обидві чудово впораються із завданням. Якщо необхідно мати уяву про форму, кривину, розміри об’єкта. Або потрібна попіксельна інформація про об’єкт та знання про конкретні його сегменти. Тоді необхідно обрати сегментацію зображення. У іншому випадку виявлення об’єктів чудово впорається із завданням.

1.5 Актуальність

Отже у цьому розділі були розглянуті розробки у сфері розумного паркування, та існуючи методи які використовуються по всьому світу. Наразі системи розумного паркування використовуються на значній кількості паркінгів у великих містах. Вони дозволяють поліпшити проблеми перевантаженості доріг та

значно скоротити час на пошук місця для паркування, пришвидшити виконання оплати та збільшити якість і швидкість обслуговування.

Не усюди є можливість створити нову інфраструктуру для розумних паркінгів, але є можливість використання існуючої. Тому питання поліпшення і модернізації систем детектування автомобілів на паркінгах за допомогою камер відеоспостереження є дуже актуальним.

Одним з підходів до модернізації існуючої інфраструктури паркінгів є використання камер відеоспостереження для детектування автомобілів на паркінгу. Для цього можна використовувати спеціальні алгоритми комп'ютерного зору та машинного навчання, які дозволяють точно визначати наявність автомобіля на паркінгу та його розташування. Розглянемо такі у наступному розділі.

2 МЕТОДИ ВИЯВЛЕННЯ ОБ’ЄКТІВ. ПОРІВНЯННЯ ТА ВИБІР

2.1 Огляд алгоритмів та методів які використовуються для виявлення об’єктів

Виявлення об’єктів — це техніка комп’ютерного зору, яка ідентифікує та визначає місцезнаходження об’єктів на зображеннях або відеокадрах. Він широко використовується в робототехніці, безпілотних автомобілях, системах спостереження та інших додатках. Алгоритми виявлення об’єктів використовуються для виявлення об’єктів на зображенні чи кадрі, призначення їх класу, а потім локалізації за допомогою обмежувальних рамок.

Алгоритми виявлення об’єктів зазвичай поділяються на дві категорії: класичні методи та методи на основі глибокого навчання.

Класичні методи виявлення об'єктів існують десятиліттями і використовуються в різних сферах. Ці методи зазвичай включають виділення ознак, сегментацію зображення та класифікацію. Виділення ознак включає вилучення певних характеристик або особливостей, таких як краї, кути або текстури, із зображення. Прикладами класичних методів є Oxford-MKL (HOG + Cascade SVM), DPM (HOG + Latent SVM), NLRP-HOGLBP (LBP/HOG + Latent SVM/Boosting), Selective Search(SIFT + SVM). Як бачимо переважна більшість цих методів базується на гістограмі напрямлених градієнтів (англ. histogram of oriented gradients, HOG).

Останнім часом для виявлення об'єктів все більшої популярності набувають методи глибокого навчання. Ці методи базуються на згорткових нейронних мережах (CNN), які є потужними моделями, що здатні навчатися на основі маркованих наборів даних, де об'єкти на зображеннях позначені рамкою та міткою класу. В яких метод вивчає ознаки, важливі для виявлення та локалізації об'єктів. Концептуально такі методи можна поділити на одноступеневі та двоступеневі детектори [19]. Прикладами методів виявлення об'єктів на основі глибокого навчання є R-CNN, Faster R-CNN, Mask R-CNN, YOLO, SSD.

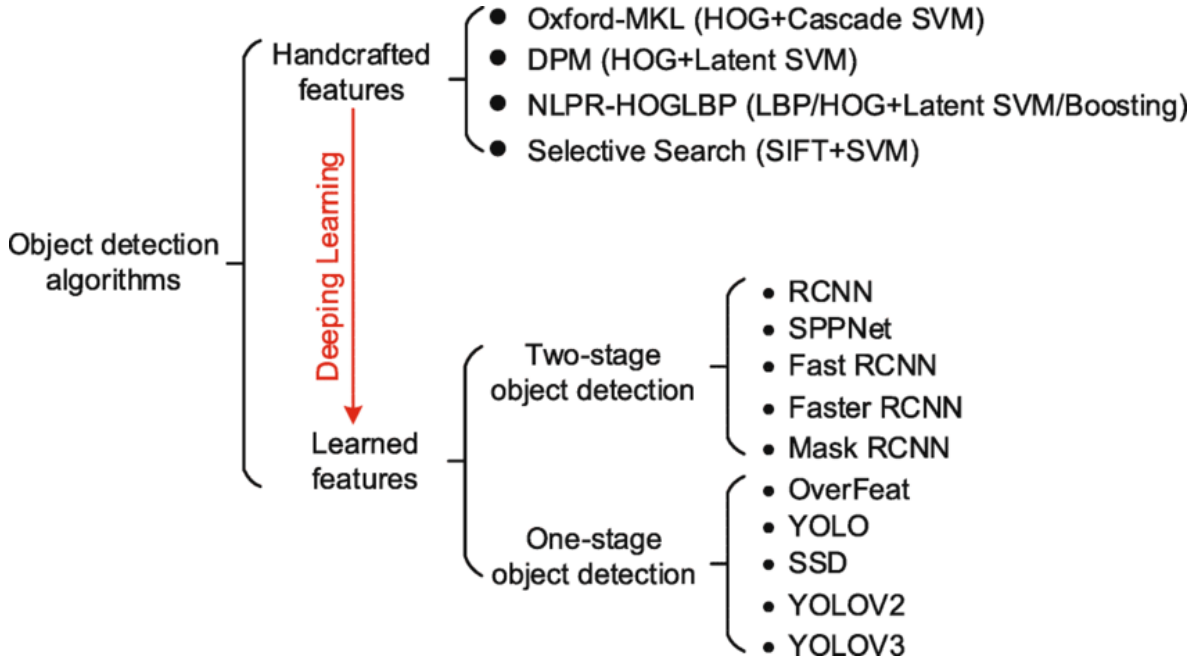

Рисунок 2.1 – Методи виявлення об’єктів

Розглянемо їх детальніше у наступних розділах.

2.1.1 Гістограма напрямлених градієнтів для виявлення об’єктів

Одним з найстаріших методів виявлення об’єктів є гістограма напрямлених градієнтів (HOG), перше він був представлений у 1986 році. HOG використовує екстрактор ознак для ідентифікації об’єктів на зображенні. Дескриптор функції є представленням частини зображення, де виявляється лише найнеобхідніша інформація, ігноруючи будь-що інше. Функція дескриптора ознак полягає в тому, щоб перетворити загальний розмір зображення у форму масиву або вектор ознак [20]. Навчання дескриптора можна організувати за допомогою класифікатора методом опорних векторів (SVM).

До переваг такого підходу можна віднести:

– унікальний дескриптор для ефективного виявлення об’єктів;

– висока точність виявлення об’єктів при вдалому навчанні;

– можливість застосування методу ковзного вікна для розрахунку позиції об’єкта.

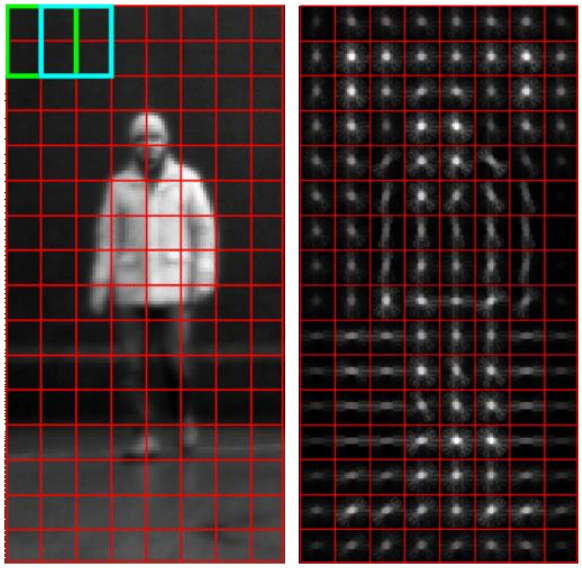

Рисунок 2.2 – Приклад роботи гістограми напрямлених градієнтів

Завдяки тому що метод шукає та використовує області зображення з високою інтенсивністю, та після обчислює їх орієнтацію, він виявляється стійким до зміни рівня освітленості та шуму у кадрі або на зображенні.

Попри все це метод має і недоліки. Не зважаючи на те що метод був досить революційним на ранніх етапах, на сьогоднішній день його продуктивність недостатня при наявності «складних» зображень. Також одиничні помилки при навчанні можуть значно негативно впливати на результати виявлення.

2.1.2 Двоступеневі методи виявлення об’єктів

Методи виявлення об’єктів постійно вдосконалюються і сьогодні крім HOG їх існує безліч.

Двоступеневі алгоритми виявлення – це алгоритми, які використовують два етапи для виявлення об'єктів на зображенні. Перший етап полягає в виявленні можливих областей інтересу на зображенні, а другий – в класифікації і локалізації об'єктів виходячи з цих областей.

До найпопулярніших двоступеневих алгоритмів, відносять R-CNN (Region-based Convolutional Neural Network), Faster R-CNN, Mask R-CNN.

Як можна побачити вони базуються на згортковій нейронній мережі (CNN). Архітектура мережі бере за основу біологічні процеси – окремі нейрони реагують лише у певній області зорового поля, таким чином не має необхідності в пов’язаній мережі нейронів для кожного пікселя зображення.

Проте таких областей може бути незліченна кількість, тому розвиток технологій призвів до появи допоміжного механізму який виділяє певну кількість таких регіонів. Цей механізм під назвою «Вибірковий пошук» використовується для визначення області інтересу (ROI). Після чого кожна область передається до нейронної мережі для створення вихідних характеристик. В кінці за допомогою набору машинних класифікаторів опорних векторів визначається тип об’єкта.

Саме так працює алгоритм R-CNN. Росс Гіршик та ін. запропонував метод який використовує вибірковий пошук та вилучає лише 2000 регіонів які називаються пропозиційними (region proposals) [21]. Ці регіони передаються до нейронної мережі, яка в якості вихідних даних дає 4096-вимірний вектор ознак. Ці ознаки передаються до SVM для визначення наявності об’єкта у запропонованому регіоні.

Попри гарну, для свого часу, точність розпізнавання алгоритм має низку недоліків. Серед яких: великий час навчання мережі, оскільки необхідно класифікувати 2000 регіонів; час вимірюється в десятках секунд тому класифікація у режимі реального часу неможлива; алгоритм вибіркового пошуку неможливо навчати через що можливе генерування невідповідних пропозицій регіонів.

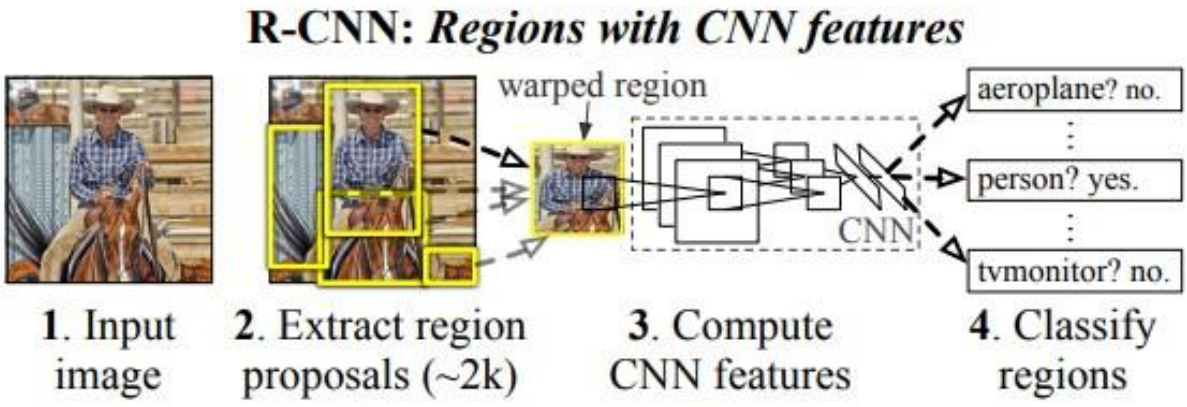

Рисунок 2.3 – Схема роботи алгоритму R-CNN

Частину з цих недоліків було вирішено у алгоритмі Fast R-CNN. Головним чином швидкість роботи вдалось поліпшити передаючи до нейронної мережі усе зображення замість окремих регіонів [22].

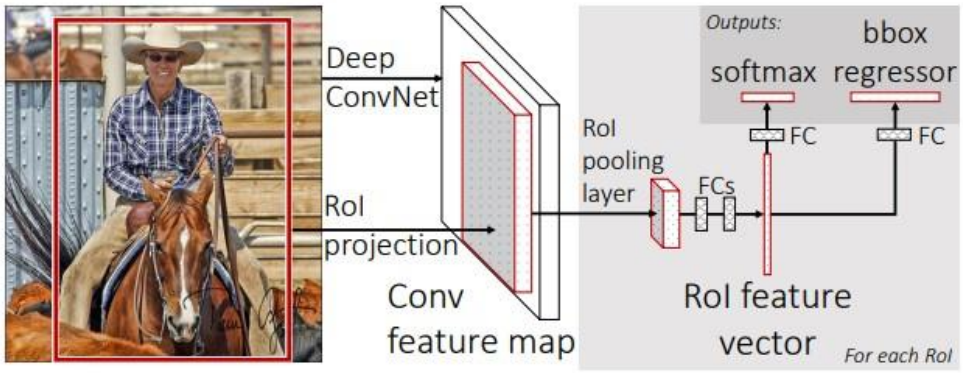

Рисунок 2.4 – Схема роботи алгоритму Fast R-CNN

Проте механізм вибіркового пошуку дуже трудомісткий процес, через що і повільний. Тому Шаоцін Рен та ін. запропонували алгоритм виявлення об’єктів під назвою Faster R-CNN у якому виключили цей механізм і дозволили безпосередньо нейронній мережі вивчати пропозиційні регіони [23]. Цю функцію виконує мережа пропозицій регіонів (region proposal network або RPN). RPN ділить більшу частину обчислень з мережею виявлення об'єктів. Якщо коротко, RPN ранжує області (так звані якорі) і пропонує ті з них, які, найімовірніше, містять об'єкти.

Алгоритм працює наступним чином:

– RPN генерує пропозиції регіонів;

– для всіх пропозицій областей на зображенні з кожної області витягується вектор ознак фіксованої довжини за допомогою шару ROI Pooling (Region of Interest Pooling – об'єднання областей інтересу);

– витягнуті вектори ознак потім класифікуються за допомогою швидкого R-CNN;

– на виході маємо оцінки класів виявлених об'єктів на додаток до їх обмежувальних рамок.

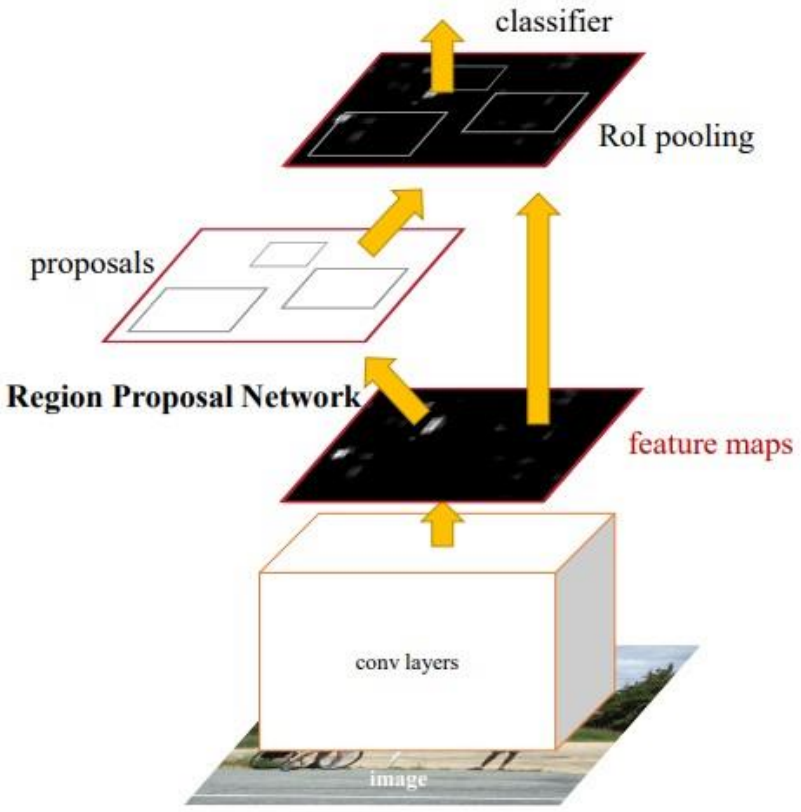

Рисунок 2.5 – Схема роботи алгоритму Faster R-CNN

Mask R-CNN – найбільш сучасний та потужний фреймворк для виявлення та сегментації об’єктів. Це розширення популярної моделі виявлення об'єктів Faster R-CNN. Алгоритм додає додаткову гілку для прогнозування масок сегментації на кожній області інтересу, таким чином дозволяючи прогнозувати маску для кожного

виявленого об'єкта. Гілка маски передбачає бінарні маски для кожного RoI. Це повністю згорткова мережа (FCN), яка складається з серії згорткових шарів, за якими слідує деконволюційний шар, що здійснює вибірку ознак до початкової роздільної здатності зображення. Результат роботи це попіксельна маска для кожного класу.

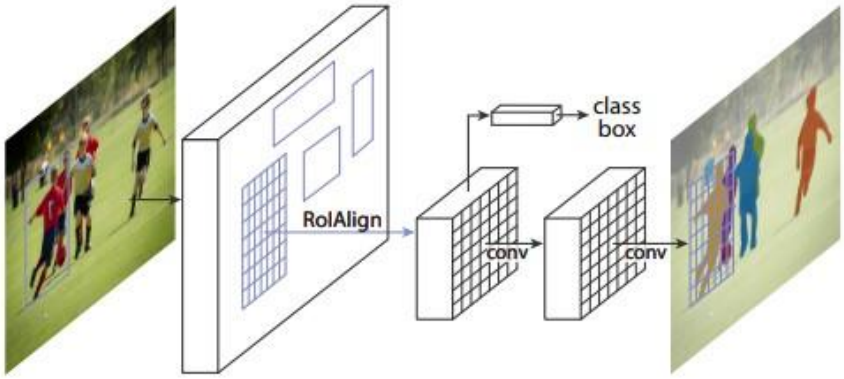

Рисунок 2.6 – Схема роботи алгоритму Mask R-CNN

Mask R-CNN працює наступним чином:

– відбувається виділення ознак за допомогою опорної мережі;
– генеруються пропозиції регіонів за допомогою RPN;
– застосовується RoIAlign до запропонованих регіонів;
– класифікується кожна RoI і уточняються координати обмежувальної рамки;
– для кожного виявленого об'єкта прогнозується маска сегментації;
– кінцевий результат складається з міток класів, уточнених обмежувальних рамок і масок сегментації для кожного виявленого об'єкта.

2.1.3 Одноступеневі методи виявлення об’єктів

Одноступеневі алгоритми виявлення – це алгоритми, які виконують виявлення об'єктів в один етап, без використання додаткових етапів, таких як

виявлення областей інтересу. Вони зазвичай використовують глибокі нейронні мережі, такі як CNN, для виявлення об'єктів.

Одноступеневі алгоритми мають деякі переваги порівняно з двоступеневими. Одна з головних переваг полягає в тому, що вони можуть працювати значно швидше, оскільки не потребують окремих етапів.

Однак, одноступеневі алгоритми можуть мати обмеження щодо точності виявлення об'єктів та можуть бути вразливими до різного роду шуму та перешкод на зображенні. Також, можуть бути менш точними в виявленні малих або фрагментованих об’єктів.

Одним із найпоширеніших одноступеневих алгоритмів є SSD: Single Shot MultiBox Detector. У 2016 році Вей Лю та ін. запропонував одноетапний алгоритм виявлення, який використовує архітектуру глибокої згорткової нейронної мережі (CNN) для виявлення об'єктів на зображеннях [24].

SSD складається з базової мережі, яка зазвичай є попередньо навченою CNN, такою як VGG або ResNet, і серії згорткових і об'єднуючих шарів, які додаються поверх базової мережі.

Алгоритм використовує підхід піраміди ознак, де згорнуті шари, додані поверх базової мережі, організовуються в декілька карт ознак з різною роздільною здатністю. Карти ознак з вищою роздільною здатністю фіксують дрібні деталі, тоді як карти ознак з нижчою роздільною здатністю фіксують більш глобальну інформацію.

Алгоритм SSD працює, розбиваючи вхідне зображення на сітку комірок і прогнозуючи наявність і розташування об'єктів у кожній комірці. Мережа прогнозує ймовірності класів та зміщення меж для кожної комірки і кожного заданого співвідношення сторін. Зміщення рамки використовуються для коригування стандартних координат рамки для кожного співвідношення сторін, щоб краще відповідати об'єкту в комірці.

За рахунок своєї архітектури алгоритм демонструє гарну швидкість роботи порівняно з двоступеневими алгоритмами. Також найкращі результати виявлення в багатьох відомих тестах, таких як PASCAL VOC і MS COCO.

Проте алгоритм має і свої недоліки. Одне з них – проблеми з виявленням об’єктів малих розмірів порівняно з зображенням, а також усічених та спотворених. Значна оклюзія також негативно впливає на результати. Крім того алгоритм вимагає вхідного зображення фіксованого розміру.

Розрізняють SSD300 та SSD512. Це один і той самий алгоритм проте перший навчений на зображеннях 300x300px, а другий – 512x512px. Збільшення розміру зображень частково допомагає вирішити проблему виявлення малих об’єктів. Проте збільшення розмірів зображення значно впливає на швидкість детектування.

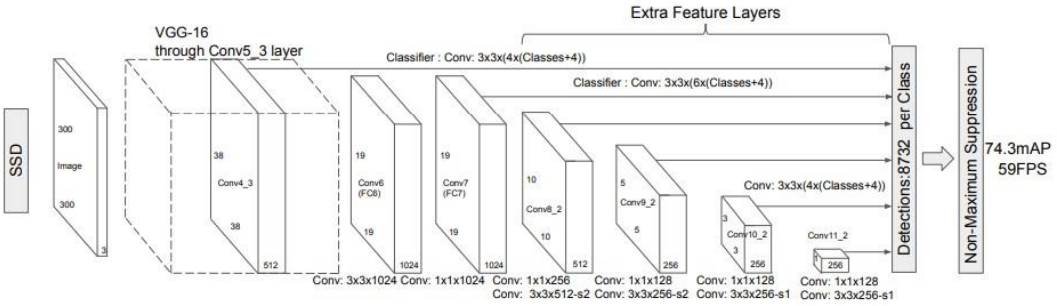

Рисунок 2.7 – Схема роботи алгоритму SSD

YOLO (You Only Look Once) – популярний алгоритм виявлення об'єктів, який використовує згорткові нейронні мережі (CNN) для виявлення об'єктів на зображеннях або відеокадрах. Джозеф Редмон та ін. вперше представив цей алгоритм ще у 2016 році [25]. Від тоді алгоритм пройшов кілька ітерацій де кожна покращувала попередню. У 2020 був представлений найпопулярніша версія YOLO v5. А на початку 2023 був представлений YOLO v8 – фреймворк який поєднує в собі переваги попередників та дозволяє класифікувати зображення, виявляти та сегментувати об’єкти[26].

Алгоритм YOLO працює, розбиваючи вхідне зображення на сітку фіксованого розміру. Прогнозує обмежувальні рамки, ймовірності класів та оцінки об'єктності для кожної клітинки. Оцінка об'єктності вказує на ймовірність присутності об'єкта в цій клітинці, тоді як ймовірність класу вказує на тип об'єкта, виявленого в цій клітинці. Після прогнозування алгоритм об’єднує результати і

отримує результат для усього зображення. Наприкінці відбувається фільтрація результатів.

Для рішення проблеми виявлення малих об’єктів алгоритм використовує пірамідальні мережі ознак (FPN). Вони поєднують ознаки з високою роздільною здатністю з попередніх шарів з ознаками з низькою роздільною здатністю з наступних шарів для створення багато-масштабної піраміди ознак.

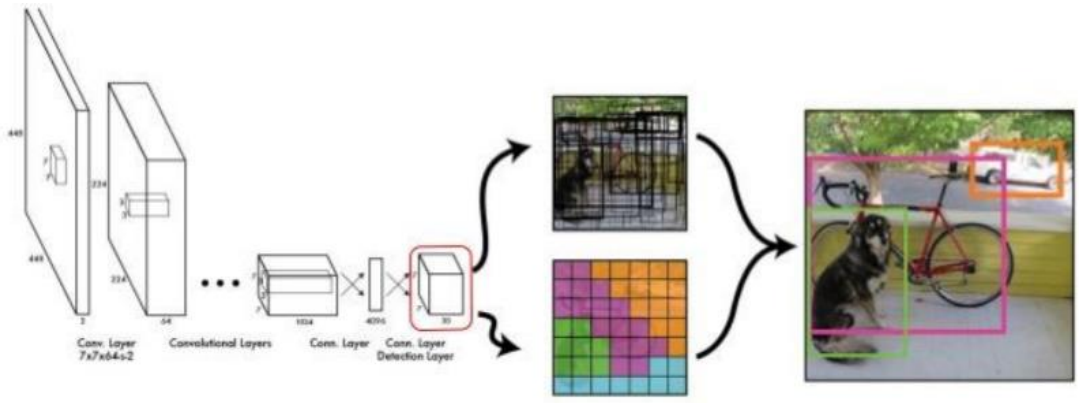

Рисунок 2.8 – Схема роботи алгоритму YOLO

Наразі алгоритми YOLO є одними з найкращих та найпоширеніших, проте вони мають деякі недоліки. Один з них – відсутність контексту. Алгоритм не враховує контекст зображення поза областями де відбувається виявлення, що може призвести до хибно позитивного результату. Ще один недолік – проблеми з виявленням об’єктів які змінюють форму з часом, або знаходяться під кутом до камери. Це пов’язано з використанням прямокутної описуючої сітки.

2.2 Порівняння алгоритмів детектування

Для порівняння методів детектування зображень використовуються різні параметри, які дозволяють оцінити їхню ефективність та придатність для конкретних завдань.

Точність є ключовим параметром, який вказує на те, наскільки точно метод виявляє об'єкти на зображеннях. Цей параметр може бути виміряний за допомогою таких метрик як Average Precision (AP) або Intersection over Union (IoU) [27].

Швидкість ще один найважливіший параметр, оскільки вона вказує на час, необхідний для обробки зображення та виявлення об'єктів. Він може бути виміряний за допомогою часу, потрібного для обробки кожного зображення або за кількістю зображень обробленими за одиницю часу.

Мінімальний розмір об'єктів є параметром, який вказує на найменший розмір об'єкта, який може бути виявлений.

Стійкість до змін – це параметр який вказує на здатність алгоритму стабільно працювати в умовах зміни освітлення, кута зйомки, зміни розмірів об’єкта, появи артефактів на зображенні тощо.

Об'єм даних - це параметр, який вказує на кількість даних, необхідних для навчання і роботи методу. Він важливий для додатків з обмеженими ресурсами, де обробка великого обсягу даних може бути проблемою.

Відтворюваність - це параметр, який вказує на те, наскільки легко можна відтворити результати методу на інших даних умовах.

Розглянуті вище алгоритми будемо порівнювати за першими двома параметрами. Оскільки інші можуть варіюватися через особливості архітектури або кількості та якості даних за якими проходило навчання.

Згідно роботи Саада Мохаммада Алкентар та ін. які досліджували швидкість та точність різних алгоритмів виявлення, таких як Faster RCNN, YOLOv3 та SSD в контексті виявлення дронів [28]. Наступні результати отримані на комп’ютері з процесором Intel® Core™i7-7700 CPU @3.60GHz. Об’ємом оперативної пам’яті 8GB. Графічним чипом NVIDIA GeForce GTX 1060 6GB. Тестові зображення розміром 1280 x 720px.

Число після назви алгоритму вказує на розмірність вхідних даних мережі. Вхідні зображення масштабуються відповідно до розмірів мережі. Аналізуючи отримані результати легко помітити, що величина середньої точності збільшується

із збільшенням розміру зображення. Проте збільшення розмірів значно негативно впливає на швидкість детектування.

Method	FPS	mAP%
Faster 1000	7.69	**6.05**
Faster 544	16.10	**1.78**
Faster 300	27.18	**0.78**
SSD 300	27.93	**0.84**
YOLO 832	45.10	**1.74**
YOLO 312	208.48	**0.71**
YOLO 632	72.67	**1.13**
YOLO 544	87.22	**1.19**

Рисунок 2.9 – Порівняння продуктивності роботи алгоритмів

У публікації щодо Single Shot MultiBox Detector Вей Лю та ін. наводять результати тестування свого алгоритму на різних датасетах та порівнюють з іншими популярними алгоритмами. Експериментальні результати з використанням даних PASCAL VOC, COCO і ILSVRC показали, що SSD не поступається в точності прогнозування навіть двоступеневим алгоритмам виявлення.

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Рисунок 2.10 – Порівняння продуктивності роботи алгоритмів

Для SSD300 з тестовими даними PASCAL VOC2007 середня точність досягає 74,3% при швидкості 59 FPS. Для SSD512 – 76,8% при швидкості 22 FPS. Тести проводились з використанням графічного чіпу Nvidia Titan X. Згідно з

результатами лише SSD300 зміг отримати результат більше 70% mAP при виявленні в режимі реального часу.

Трупті Махендракар у своїй роботі «Дослідження продуктивності YOLOv5 та Faster R-CNN для автономної навігації навколо некооперативних цілей» порівнював два концептуально різні алгоритми виявлення [29]. Для навчання та тестування детекторів було власноруч створено та марковано набір з 1231 зображення які містять принаймні один штучний супутник.

	YOLOv5			Faster R-CNN		
Class	**AR**	**mAP@ .5**	**mAP@ .5:.95**	**AR**	**mAP@ .5**	**mAP@ .5:.95**
All	50.925%	53.05%	31.725%	51.875%	63.575%	41.03%
Body			30.957%			57.03%
Solar			32.475%			30.69%
	Inference Rate:0.017s/img			Inference Rate: 0.17 s/img		

Рисунок 2.11 – Порівняння продуктивності роботи алгоритмів

Обчислення проводилося за допомогою процесору Intel® Core™i7-8550U. Результат середньої точності для алгоритму YOLOv5 становить 31,7% mAP при швидкості 58,8 FPS, для Faster R-CNN – 41% mAP при швидкості 5,8 FPS.

Отже, двоступеневі алгоритми зазвичай досягають вищих показників mAP. Проте їх швидкість нижча порівняно з одноступеневими алгоритмами. Слід зазначити, що точність та швидкість залежить від конкретної реалізації алгоритму та обладнання що використовується.

2.3 Експериментальне порівняння алгоритмів виявлення

Оскільки через велику варіативність реалізацій алгоритмів та тестових даних не можна впевнено відповісти на питання вибору «найкращого» алгоритму виявлення об’єктів. У цьому розділі проведемо експериментальне порівняння

декількох алгоритмів виявлення об'єктів. Проаналізуємо їх ефективність, швидкість та точність в різних умовах. Це допоможе зрозуміти, який алгоритм краще застосовувати у випадку детектування автомобілів на місцях для паркування.

В якості даних використаємо набір даних NDISPark [30]. Це анотований вручну датасет про автомобілі на автостоянках, що складається з близько 250 знімків. Зображення зроблені за різних погодних умов, кутів огляду, умов освітлення та перешкод, що описує більшість проблемних ситуацій, які можливо знайти в реальному сценарії.

Дослідження буде проводитися з використанням обчислення за допомогою процесору Intel® Core™i5-1035G1. Використовується бібліотека комп'ютерного зору з відкритим кодом OpenCV. За допомогою бібліотеки реалізовані такі алгоритми Faster R-CNN, Mask R-CNN, SSD, YOLOv3, YOLOv4, YOLOv8. Тренування за допомогою даних датасету не проводилось, були використані притреновані, за допомогою MS COCO, ваги моделей. Зображення були масштабовані до розміру 640x640 px.

Завданням для реалізованих алгоритмів було підрахунок автомобілів на автостоянці за допомогою малопотужного обладнання без використання графічного процесору. Тому параметрами для порівняння були обрані середня точність та швидкість виявлення.

Таблиця 2.1 - Порівняння продуктивності роботи алгоритмів

	FPS	mAP
Faster R-CNN	1,65	49,77
Mask R-CNN	1,65	44,92
SSD	10,11	31,78
YOLOv3	2,28	73,87
YOLOv4	1,7	64,79
YOLOv8	5,47	66,89

Представимо результат у графічному вигляді.

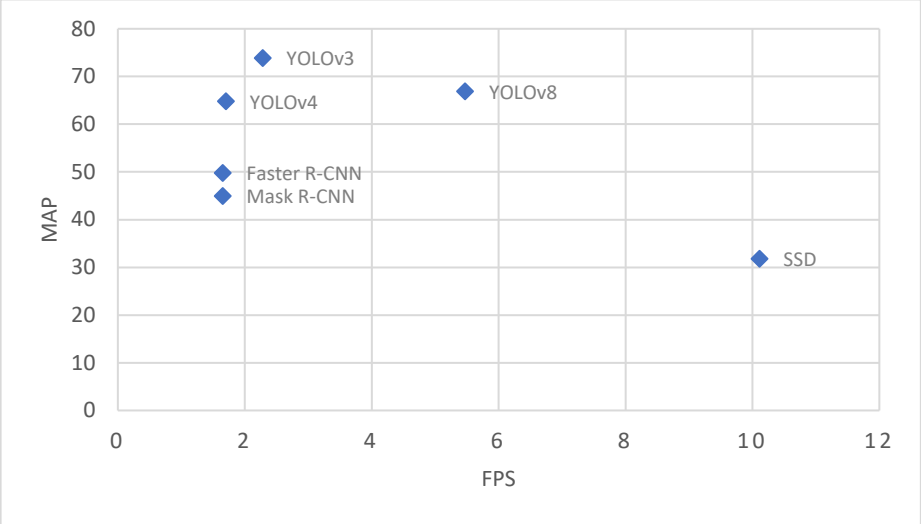

Рисунок 2.12 – Графік порівняння продуктивності роботи алгоритмів

Згідно графіка на рисунку 2.12 можна побачити, що системі не вистачає потужності для забезпечення роботи алгоритмів в режимі реального часу. З даних таблиці 2.1 – найбільш швидким виявився алгоритм SSD, найбільш точним - YOLOv3. З метою підвищення швидкості обробки зображення в алгоритмах SSD, YOLOv3 та v4 вхідні зображення були масштабовані до розмірів 416x416 px. Крім того YOLOv8 використовував модель “nano”, яка має найменшу кількість параметрів та відрізняється високою швидкістю та малою точністю.

Двоступеневі алгоритми Faster та Mask R-CNN не дивлячись на наявність додаткової гілки сегментації у останнього майже не відрізняються за швидкістю роботи та точністю.

Змінимо умови проведення експерименту та для детекторів SSD, YOLOv3 та v4 змінимо розмір вхідного зображення мережі до вихідного. Для YOLOv8 використаємо модель “large”.

Таблиця 2.2 – Порівняння продуктивності модифікованих алгоритмів

	FPS	mAP
SSD	5,08	34,15
YOLOv3	1,09	84,11
YOLOv4	0,82	76,07
YOLOv8	0,67	89,64

Додаймо отриманий результат отриманого на рисунку 2.12 і отримаємо наступне.

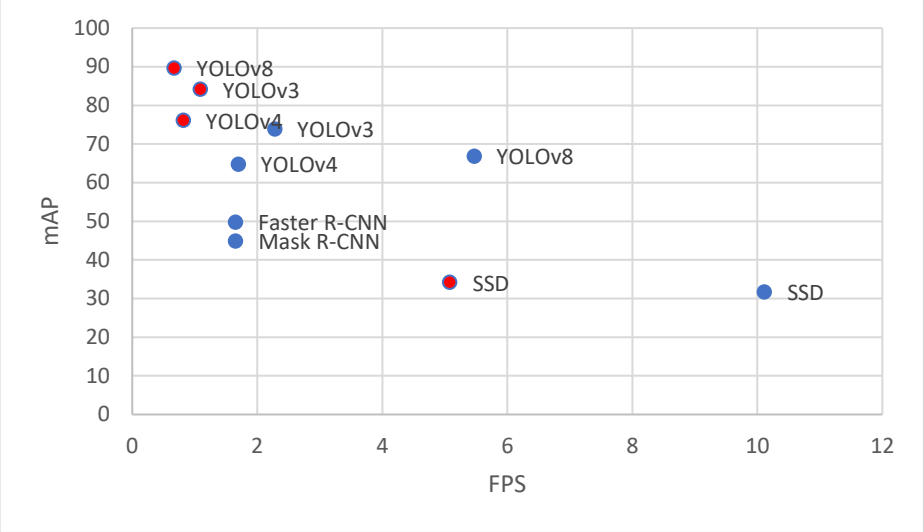

Рисунок 2.13 – Графік порівняння продуктивності модифікованих алгоритмів

Порівнюючи отримані результати, таблиця 2.2 та рисунок 2.13 з попереднім експериментом не складно помітити приріст точності для алгоритмів YOLO. YOLOv8 додав 22,75% mAP і став повільніше більше ніж у 8 разів. YOLOv4: +11,28% mAP, - 48% FPS. YOLOv3: +10,24% mAP, - 47% FPS. SSD: +2,37% mAP, - 50% FPS.

Найгірший результат для алгоритму SSD, збільшення розмірів зображення призвели лише до зниження швидкості виявлення. Найбільший приріст точності YOLOv8 призвів до значного зниження швидкості виявлення.

2.4 Похибки алгоритмів детектування об’єктів

Алгоритми детектування об'єктів є важливим інструментом в багатьох областях, таких як комп'ютерне зору, робототехніка, медична діагностика та інші.

Однак, як і будь-який алгоритм, вони можуть містити різні види помилок, які можуть призвести до неправильного розпізнавання або пропуску об'єктів.

Хибно позитивний результат роботи алгоритму детектування об'єктів – це помилкове визначення наявності об'єкта на зображенні, коли його там насправді немає. Це може статися через різноманітні причини, такі як шум на зображенні, схожість об'єкта з іншими об'єктами або помилки в самому алгоритмі.

Хибно негативний результат навпаки – це неспроможність алгоритму визначити наявність об’єкту на зображенні при його наявності. Це може статися через різноманітні причини, такі як недостатня чутливість алгоритму, недостатня роздільна здатність або якість зображення.

У обох випадках це може призвести до серйозних наслідків, таких як помилкове спрямування руху транспорту або некоректного аналізу медичного зображення.

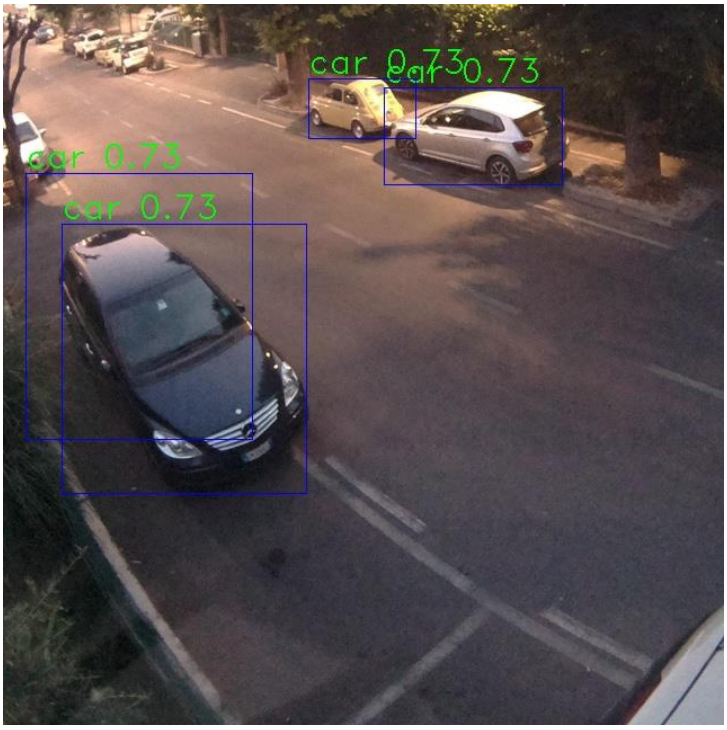

Рисунок 2.14 – Приклади хибно негативного та позитивного результату

Хибна локалізація об’єкта та помилка визначення розміру об’єкта – випадки коли алгоритм правильно помічає об’єкт проте обмежувальну рамку для

відповідного випадку малює не там та не відповідного розміру щодо визначеного об’єкту. Причина цих похибок - недостатня точність алгоритму, недостатня кількість даних для навчання або недостатній контраст між об'єктом та фоном, наявність багатьох об’єктів, які нагадують одне одного.

Рисунок 2.15 – Приклади хибного визначення розміру та локалізації

Похибка класифікації - це випадок, коли алгоритм помічає об'єкт правильно, але неправильно класифікує його. Це може статися через недостатню кількість даних для навчання алгоритму або через недостатню точність алгоритму.

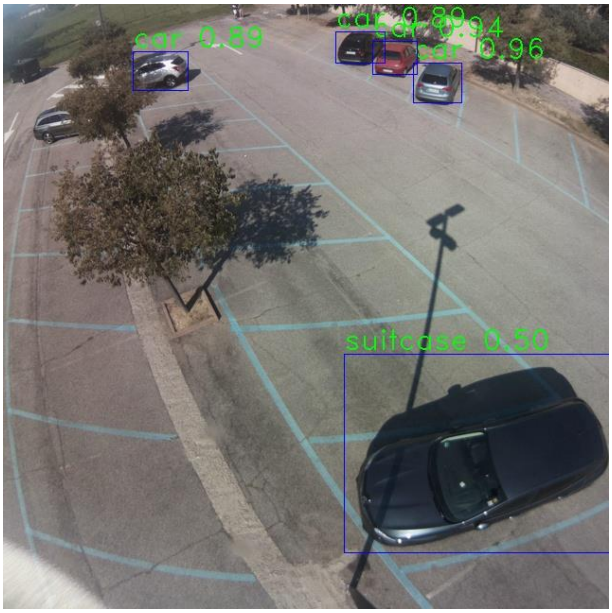

Рисунок 2.16 – Приклад похибки класифікації

Більшість помилок які роблять алгоритми детектування відбуваються через малу точність яка обумовлена недостатньою кількістю або якістю даних для навчання. Алгоритм має навчатися на даних які максимально наближені до тих з якими доведеться працювати.

Розглянемо попередньо навчену модель Сві Сяо Ци «Parking Lot Availability» [31]. Вона використовує лише 42 зображення для навчання проте має результат 88% mAP. Тренувальні зображення та сама модель націлені на те щоб визначати автомобілі та вільні місця на автостоянці.

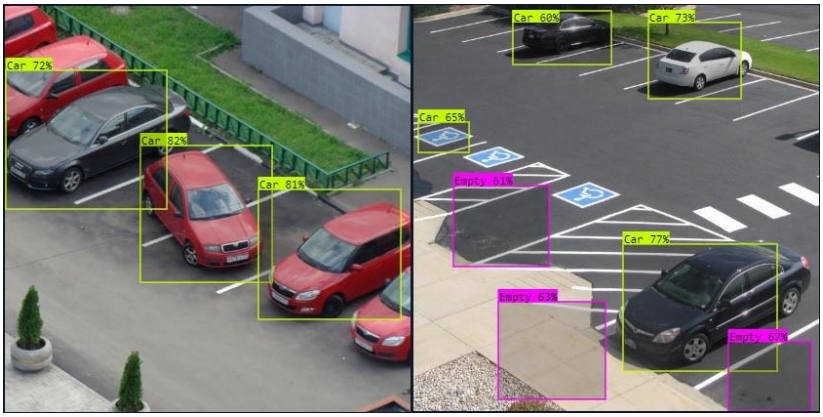

Рисунок 2.17 – Результат роботи Parking Lot Availability API

Попередньо навчена модель Бреда Дуайєра використовує набір даних PKLot містить 12 416 зображень парковок, отриманих з камер спостереження які анатовані вручну. Результат 99,4% mAP [32].

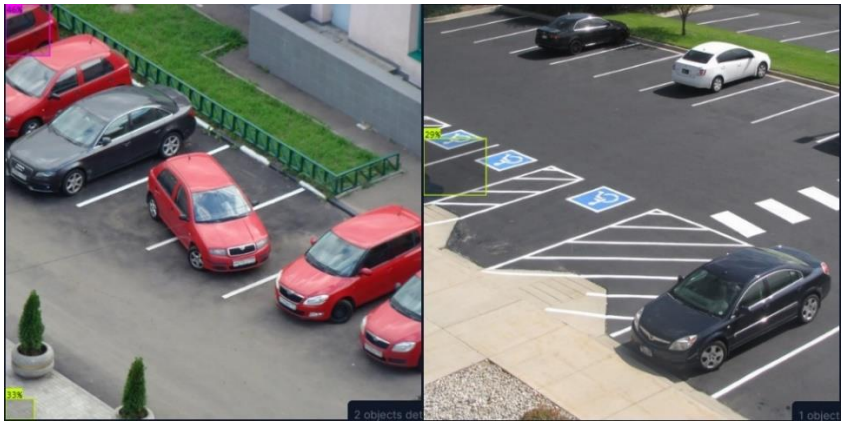

Рисунок 2.18 – Результат роботи PKLot Detection API

Як можна побачити результати роботи моделі на не релевантних даних містить усі можливі похибки описані вище.

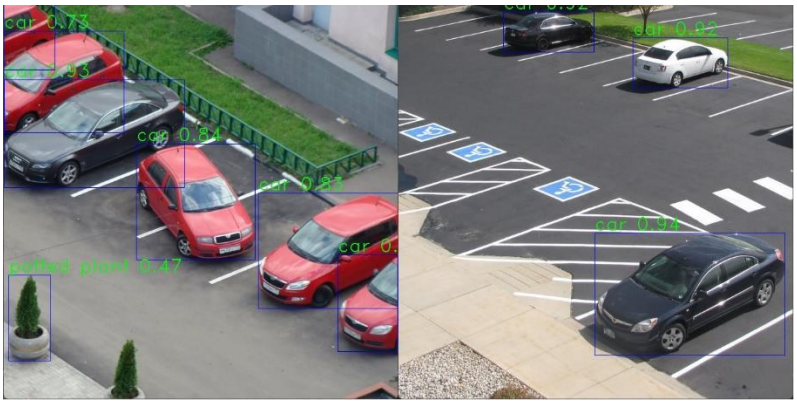

Рисунок 2.19 – Результат роботи YOLOv8

Проте модель YOLOv8 претренована лише на даних MS COCO та має результат 67% mAP, для детектування автомобілів, для тих самих зображень візуально має набагато кращі результати.

2.5 Аналіз результатів

У цьому розділі були розглянуті та порівняні існуючі алгоритми виявлення об’єктів. Порівняння детекторів відбувалося за допомогою огляду літератури та експериментального аналізу.

Експериментальний аналіз на основі даних які максимально наближені до реальних умов. Зйомка за допомогою камер відеоспостереження у різних погодних умовах, освітленні та наявності перешкод.

Під час аналізу було виявлено алгоритм з найкращими показниками точності, це YOLOv8 з моделлю “large”, який показав результат 89,64% mAP. Також

найшвидший – SSD, з результатом 10,11 FPS. Експеримент проводився у режимі обчислення за допомогою процесору Intel i5-1035G1.

Аналіз результатів показав що сучасні алгоритми виявлення мають достатній рівень точності для притренованих ваг. Проте кожен алгоритм має пройти точне донавчання на даних які безпосередньо відносяться до робочих.

Також було виявлено, що точність та швидкість виявлення об'єктів можуть залежати від різних чинників, таких як розмір зображення та складність об'єктів, що потрібно виявити.

До того ж було виявлено проблему помилкової класифікації виявлених об’єктів. Це може відбуватися з різних причин. Ці причини можуть бути пов'язані з контекстом зображення, недостатньою кількістю даних для навчання моделі, недостатньою розміром зображення, якістю зображення, ступенем деталізації виявлених об'єктів, а також з алгоритмами класифікації, які використовуються в методах виявлення об'єктів.

Оскільки кожен метод має свої переваги та недоліки, для вибору найкращого методу виявлення об'єктів необхідно враховувати багато чинників, таких як складність об'єктів, які потрібно виявити, розмір зображення та вимоги до точності та швидкості. При правильному виборі методу виявлення об'єктів можна досягти високої точності та швидкості виявлення об'єктів на зображеннях.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ДОПОМОГИ ВОДІЯМ У ПОШУКУ ВІЛЬНОГО МІСЦЯ ДЛЯ ПАРКУВАННЯ

3.1 Формування вимог до програмної системи

Першим етапом розробки прототипу системи розумного паркування є дослідження необхідних функцій та вимог, які є ключовими для цього типу системи. Це дослідження допоможе зрозуміти процес розробки всієї системи.

Програмна система має використовувати наявні камери відеоспостереження, що встановлені у місті. Також слід враховувати проблему конфіденційності при їх використанні.

Ще однією проблемою є використання мережі інтернет для відправки зображень які вимагають обробки на сервер. Слід використовувати методи шифрування даних для уникнення проблем безпеки. Для їх рішення також можливо обробляти зображення на місці.

З метою побудови динамічної розумної системи паркування слід вирішити проблему виявлення місць для паркування. Виявлення місць для паркування за допомогою комп’ютерного зору має декілька питань які необхідно вирішити. Приклад – прив’язка до певних об’єктів, або ліній які здатна виявити система комп’ютерного зору.

Підхід прив’язки до об’єктів, таких як лічильники або лінії розмітки пов’язаний з наявністю деяких недоліків – перешкоди будуть заважати процесу виявлення, лінії з часом зітруться або покриються снігом, роздільної здатності зображення може бути недостатньо для проведення виявлення.

Апаратне забезпечення яке зможе брати участь у роботі цієї системи складається з 3 різних частин: камери відеоспостереження, комп’ютер або сервер, інтерфейс користувача.

Камери відеоспостереження будуть використані в якості детектору. Вони будуть знімати стоянку та відправляти зображення на сервер.

Комп’ютер або сервер призначений для обробки зображень, запуску процесу виявлення та відправки повідомлень призначених для користувача.

Інтерфейс користувача буде мати вигляд мобільного додатку, або пуш повідомлень з усією необхідною інформацією.

Програмна система повинена мати наступні функціональні вимоги:

– збір та обробка даних. Додаток повинен збирати дані про наявність вільних місць для паркування з камер, які встановлені в місті. Дані повинні бути опрацьовані та відображатися в додатку в режимі реального часу;
– навігація. Додаток повинен мати функцію навігації, яка допоможе водіям знайти найближче вільне місце для паркування. Навігація повинна враховувати можливі обмеження на дорозі;
– інформаційні повідомлення. Додаток повинен мати функцію надсилання повідомлень користувачам про зміни у наявності вільних місць для паркування, наприклад, про те, що місце було зайнято;
– аналітика. Додаток повинен мати функцію збору та аналізу даних про використання парковок, щоб допомогти місту у плануванні та вдосконаленні інфраструктури паркування.

Програмна система повинена мати наступні нефункціональні вимоги:

– зручність. Система повинна мати інтуїтивно зрозумілий та зручний інтерфейс, що дозволить користувачам швидко та ефективно знаходити потрібну інформацію;
– конфіденційність. Система повинна забезпечувати безпеку даних користувачів та захищати їх від несанкціонованого доступу. Крім того, система повинна дотримуватися стандартів безпеки даних та захисту персональної інформації;
– надійність та стабільність: система повинна бути стабільною та безперебійною у роботі, навіть при великому навантаженні. Крім того, система повинна мати можливість відновлювати свою роботу у разі виникнення помилок або аварій;

– продуктивність. Система повинна працювати швидко та ефективно, забезпечуючи користувачам швидкий доступ до потрібної інформації;
– сумісність з іншими системами. Система повинна бути сумісною з іншими програмними продуктами та даними, що дозволить користувачам отримувати актуальну та точну інформацію про доступні місця для паркування;
– масштабованість. Система повинна мати можливість масштабування, щоб забезпечити свою ефективність та продуктивність при збільшенні кількості користувачів та обсягу даних.

Система повинна бути швидкою та ефективною у виконанні запитів користувачів. Система повинна мати можливість обробляти великі обсяги даних та забезпечувати безперебійну роботу, навіть при великому навантаженні.

Система повинна бути сумісною з різними операційними системами та браузерами, щоб користувачі могли використовувати її на різних пристроях.

Система повинна мати можливість оновлюватись та виправляти помилки, щоб забезпечити безперебійну та ефективну роботу. Крім того, система повинна мати можливість надавати технічну підтримку користувачам, які мають проблеми з використанням додатку.

Отже, формування вимог до програмної системи, призначеної для допомоги водіям в пошуку вільного місця для паркування, повинно враховувати функціональні, нефункціональні, вимоги до продуктивності, сумісності та технічної підтримки.

3.2 Архітектура та проектування програмного забезпечення

В системах орієнтованих на взаємодію з користувачем, таких як мультимедійна складова в автомобілі можлива розробка з використанням

методології BDD, коли сценарії користування можна використовувати як тести, керуючись якими і ведеться розробка [33].

Під час роботи візьмемо за основу водоспадну модель життєвого циклу програмного забезпечення. Водоспадна модель є статичною моделлю і підходить до розробки систем лінійно і послідовно, завершуючи одну дію перед іншою.

Проектування програмної системи призначеної для допомоги водіям в пошуку вільного місця для паркування повинно включати наступні етапи:

– аналіз вимог. Необхідно провести детальний аналіз вимог користувачів та потреб водіїв, щоб зрозуміти, які функції та можливості повинна містити система;
– проектування архітектури. На основі вимог необхідно розробити архітектуру системи та визначити, які компоненти та модулі будуть входити до системи;
– розробка бази даних. Необхідно розробити базу даних для зберігання інформації про вільні місця для паркування, користувачів та їхніх запитів;
– розробка інтерфейсу користувача. Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс, який дозволить користувачам швидко та ефективно знаходити вільні місця для паркування;
– розробка логіки програми. Необхідно розробити логіку програми, яка дозволить системі швидко та точно знаходити вільні місця для паркування на основі запитів користувачів;
– тестування та відлагодження. Після розробки програмної системи необхідно провести тестування та відлагодження, щоб переконатися у її правильності та ефективності;
– розгортання та підтримка. Після успішного тестування необхідно розгорнути систему та забезпечити її підтримку та оновлення в майбутньому.

Отже, проектування програмної системи призначеної для допомоги водіям в пошуку вільного місця для паркування повинно включати аналіз вимог,

проектування архітектури, розробку бази даних, розробку інтерфейсу користувача, розробку логіки програми, тестування та відлагодження, розгортання та підтримку.

Після формування вимог і вибору моделі життєвого циклу програмного забезпечення можна переходити до безпосередньо етапів розробки.

3.3 Аналіз вимог

Аналіз вимог є першим етапом проектування програмної системи призначеної для допомоги водіям в пошуку вільного місця для паркування. На цьому етапі необхідно зрозуміти, які функції та можливості повинна містити система, щоб задовольняти потреби користувачів.

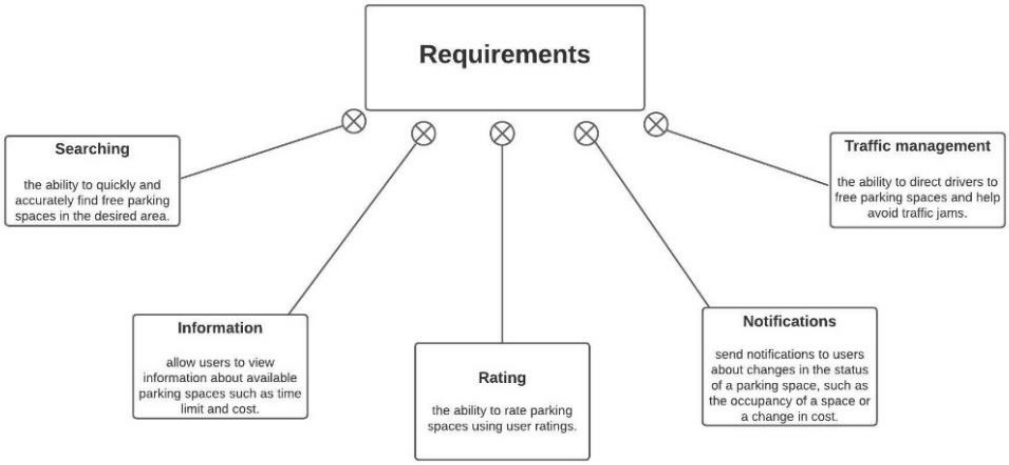

Рисунок 3.1 – Вимоги до програмної системи

Основні вимоги до програмної системи для допомоги водіям в пошуку вільного місця для паркування можуть бути наступними:

– пошук вільних місць для паркування. Система повинна мати можливість швидко та точно знаходити вільні місця для паркування в потрібній області;

– перегляд інформації про місце паркування. Система повинна дозволяти користувачам переглядати інформацію про вільні місця для паркування, таку як час обмеження та вартість;
– рейтинг місць для паркування. Система повинна мати можливість оцінювати місця для паркування за допомогою рейтингу користувачів;
– повідомлення про зміни у статусі місця для паркування. Система повинна надсилати повідомлення користувачам про зміни у статусі місця для паркування, такі як зайнятість місця чи зміна вартості;
– керування трафіком. Система повинна мати можливість направляти водіїв до вільних місць для паркування та допомагати уникнути заторів.

3.4 Проектування архітектури

Створення хорошого програмного забезпечення вимагає попереднього проектування його функцій, часових рамок і джерел даних. Завершення першого фундаментального етапу в розробці програмного забезпечення можна досягти шляхом планування цих частин [34].

На основі наведених вимог архітектура програмної системи для допомоги водіям у пошуку вільних місць для паркування може мати наступний вигляд:

Система може бути розділена на три основні компоненти:

– клієнт;
– серверна частина (API);
– база даних.

Клієнт.

Клієнт буде взаємодіяти з користувачами та відображати вільні місця для паркування. Він буде складатися з наступних модулів:

– авторизація: реєстрація користувачів та вхід до системи;

– пошук: пошук вільних місць для паркування за заданими критеріями (наприклад, відстань, ціна, час обмеження);

– інформація про паркування: відображення детальної інформації про обране місце паркування;

– рейтинги та відгуки: можливість оцінювати місця для паркування та залишати відгуки;

– сповіщення: отримання повідомлень про зміни у статусі місця для паркування та інших подій;

– навігація: інтеграція з системою навігації для автоматичного прокладання маршруту до обраного місця для паркування.

Серверна частина (API).

Буде обробляти запити від клієнту та надавати доступ до бази даних. Основні компоненти серверної частини:

– інтерфейс для спілкування з мобільним додатком;

– обробка запитів: виконання запитів до бази даних, обробка результатів та відправка відповідей до клієнту;

– аналіз даних: обробка даних (наприклад, аналіз рейтингів і відгуків);

– інтеграція з зовнішніми системами: забезпечення обміну даними зі сторонніми системами, такими як системи оплати, навігації, контролю паркування тощо.

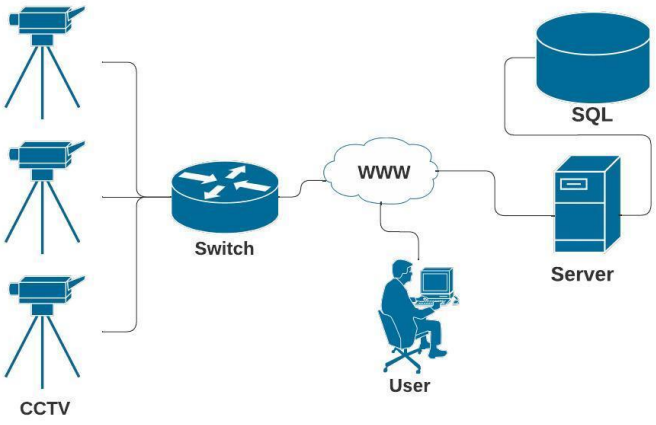

Рисунок 3.2 – Схема взаємодії апаратного забезпечення

База даних буде зберігати інформацію про місця паркування, користувачів, рейтинги, сповіщення, збирати інформацію для систем сучасного розумного міста, збирати зображення для донавчання алгоритмів виявлення.

3.5 Розробка бази даних

База даних має відповідати усі поставленим вище умовам. Для цього необхідно створити необхідні таблиці БД.

Таблиця "Місця паркування":

– id (INT, PRIMARY KEY): унікальний ідентифікатор місця паркування;

– назва (VARCHAR): назва місця паркування;

– адреса (VARCHAR): адреса місця паркування;

– координати (VARCHAR): географічні координати місця паркування;

– кількість місць (INT): загальна кількість місць для паркування;

– вільні місця (INT): кількість вільних місць для паркування;

– тип місця паркування (VARCHAR): тип місця паркування;

– опис (TEXT): додаткова інформація про місце паркування.

Таблиця "Користувачі":

– id (INT, PRIMARY KEY): унікальний ідентифікатор користувача;

– ім'я (VARCHAR): ім'я користувача;

– прізвище (VARCHAR): прізвище користувача;

– email (VARCHAR): email користувача;

– пароль (VARCHAR): пароль користувача;

– телефон (VARCHAR): телефон користувача;

– роль (VARCHAR): роль користувача.

Таблиця "Рейтинги":

– id (INT, PRIMARY KEY): унікальний ідентифікатор рейтингу;

– id_користувача (INT, FOREIGN KEY REFERENCES Користувачі(id)): ідентифікатор користувача, який залишив рейтинг;
– id_місця_паркування (INT, FOREIGN KEY REFERENCES Місця_паркування(id)): ідентифікатор місця паркування, якому був залишений рейтинг;
– оцінка (INT): оцінка місця паркування від 1 до 5;
– коментар (TEXT): коментар користувача про місце паркування;

Таблиця "Сповіщення":

– id (INT, PRIMARY KEY): унікальний ідентифікатор сповіщення;
– id_місця_паркування (INT, FOREIGN KEY REFERENCES Місця_паркування(id)): ідентифікатор місця паркування, на якому сталося подія;
– тип (VARCHAR): тип події (наприклад, вільні місця, заборонений припаркування, стан місця паркування);
– дата_і_час (DATETIME): дата та час сповіщення;
– опис (TEXT): додаткова інформація про подію.

Таблиця "Зображення":

– id (INT, PRIMARY KEY): унікальний ідентифікатор зображення;
– id_місця_паркування (INT, FOREIGN KEY REFERENCES Місця_паркування(id)): ідентифікатор місця паркування, на якому було зроблене зображення;
– шлях_до_файлу (VARCHAR): шлях до файлу зображення на сервері;
– дата_і_час_зйомки (DATETIME): дата та час зйомки зображення;
– опис (TEXT): додаткова інформація про зображення (наприклад, опис стану місця паркування на зображенні).

Програма отримуватиме зображення з камер відеоспостереження за необхідністю або після деякої події. До бази даних буде записана вся необхідна інформація на кшталт тої яка необхідна для донавчання алгоритму, або запису стану справ на автостоянці.

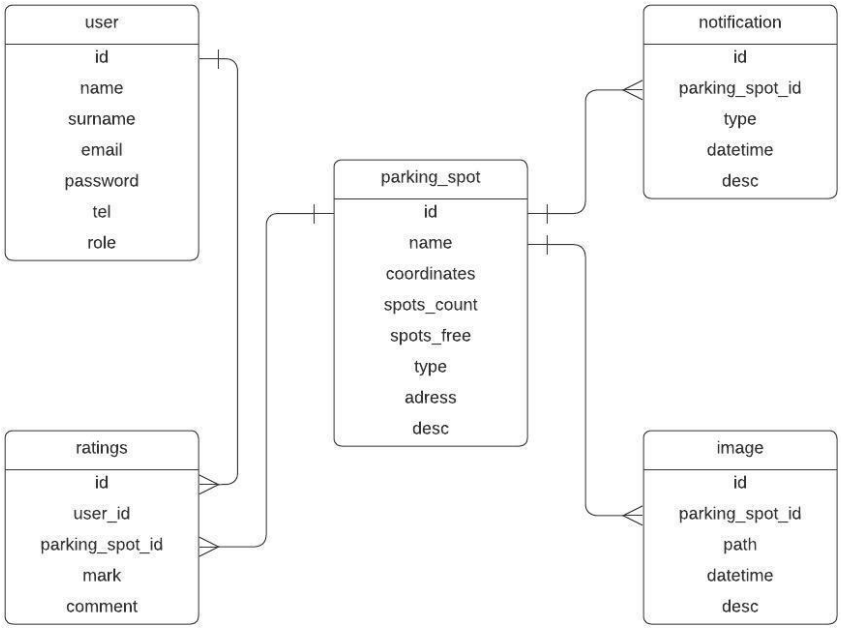

Рисунок 3.3 – Схема бази даних

За допомогою цієї бази даних можна відслідковувати кількість вільних місць для паркування на різних місцях, отримувати сповіщення про події на місцях паркування (наприклад, заборонений припаркування), а також збирати відгуки та рейтинги користувачів про місця паркування. Крім того, база даних дозволяє збирати зображення з місць паркування для донавчання алгоритмів виявлення та покращення роботи системи розумного міста.

3.6 Розробка інтерфейсу користувача

Оскільки в більшості країн користуватися за кермом заборонено, то можливо спростити розробку графічного інтерфейсу. Клієнт буде мати вигляд телеграм боту з усім необхідним функціоналом. Відповіддю на запит пошуку вільного місця на автостоянці бот буде відправляти фото місця для паркування та кількість вакантних місць.

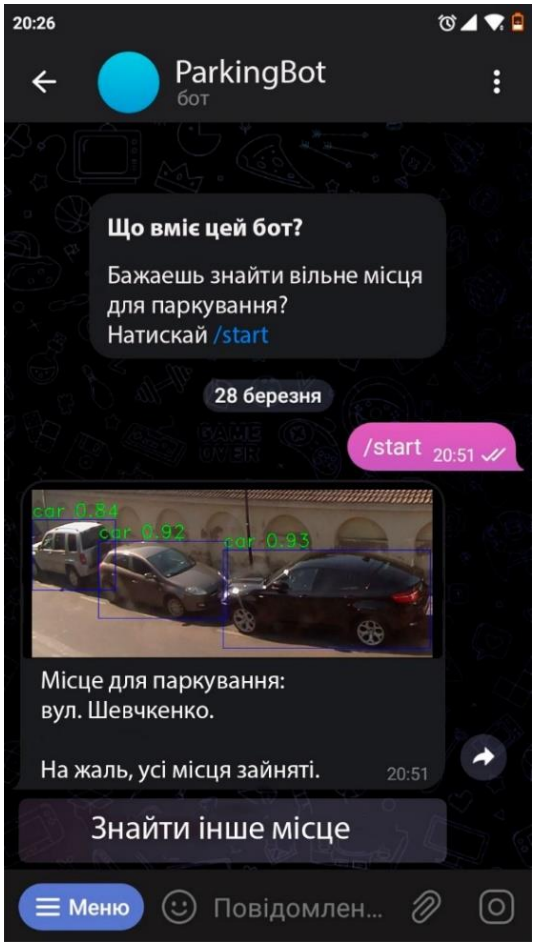

Рисунок 3.4 – Інтерфейс користувача

Фото автостоянки може не відповідати умовам конфіденційності, проте це обумовлене тим що на даний момент алгоритми виявлення не можуть забезпечити 100% точності, та потребують перевірки людиною. В нашому випадку це буде користувач. Проблему конфіденційності можливо вирішити редагувавши зображення і видаливши на ньому номера автівок.

3.7 Розробка логіки програми

Розробка логіки додатку буде включати кілька кроків на якому буде розроблено і імплементовано наступні блоки програми.

Збір інформації про місця паркування: програма збирає інформацію про місця паркування за допомогою камер відеоспостереження.

Перевірка доступності місць паркування: програма перевіряє наявність вільних місць паркування за допомогою засобів збору інформації.

Відображення інформації про вільні місця паркування: програма відображає інформацію про вільні місця паркування на мапі та фотознімку, щоб користувачі могли легко знайти доступні місця для паркування.

Фільтрація місць паркування: програма дозволяє користувачам фільтрувати місця паркування за різними критеріями, такими як розташування, тип місця паркування або доступність для інвалідів.

Навігація до місць паркування: програма надає користувачам можливість прокладати маршрут до місць паркування за допомогою навігаційних додатків.

Система оновлення даних: програма має систему оновлення даних, яка оновлює інформацію про місця паркування та доступність вільних місць паркування в реальному часі.

3.8 Тестування та відлагодження

Тестування та відлагодження є важливими етапами в розробці будь-якої програми, в тому числі і цієї. Додаток має пройти увесь цикл тестування перш ніж кінцевий користувач почне використовувати її. Всі види тестування є життєво важливими для створення якісного продукту. З одного боку, існує широкий спектр інструментів, які можуть бути використані для досягнення цієї мети, використовуючи найкращі рішення, що існують на сьогоднішній день, але з іншого боку, ці інструменти мають суттєві переваги та недоліки, які полягають у підході до них, тому завжди є мінуси та плюси використання того чи іншого інструменту [35]. Одним із найважливіших етапів буде тестування навантаження. Воно

обумовлене тим що алгоритми виявлення потребують значних обчислювальних ресурсів.

До процесу відлагодження будуть запрошені волонтери. З їх допомогою вдасться зібрати та анотувати дані для донавчання алгоритму виявлення.

3.9 Розгортання та підтримка

Останній етап – це розгортання та підтримка користувачів. Розгортання програми — це процес, коли вона стає доступною для користування. Цей етап може включати встановлення програми на сервер або на комп'ютер користувача, відповідну конфігурацію та налаштування. Після розгортання важливо забезпечити підтримку користувачів, щоб вони мали можливість звертатись за допомогою та отримувати необхідну підтримку. Програма збирає необхідну інформацію і здатна розширюватись та донавчати алгоритм виявлення.

ВИСНОВКИ

За результатами проведених теоретичних та практичних досліджень було вивчено та запропоновано вдосконалення яке відсутнє в існуючих методах детектування в системах інтелектуального паркування. Саме метод детектування автомобілів та вільних місць для паркування за допомогою підрахунку зайнятих місць з використанням комп’ютерного зору.

У першому розділі був проведений аналіз предметної галузі в якому досліджені існуючі методи детектування автомобілів на місцях для паркування. В процесі аналізу були виявлені проблемні місця цієї галузі. Це і послугувало стимулом для постановки задачі. Задача виглядає наступним чином – детектування автомобілів на місцях для паркування у місті та на паркінгах за допомогою камер відеоспостереження та комп’ютерного зору на основі алгоритмів машинного навчання.

У другому розділі було проведено огляд алгоритмів та методів для виявлення об'єктів. Розглянуто двоступеневі та одноступеневі методи виявлення об'єктів. Було проведено теоретичне порівняння алгоритмів детектування та експериментальне порівняння алгоритмів. Також було проаналізовано похибки алгоритмів детектування об'єктів. На основі отриманих результатів можна зробити висновок, що одноступеневі методи виявлення об'єктів мають більш високу точність та швидкодію порівняно з двоступеневими методами. Однак, враховуючи похибки детектування, можна стверджувати, що вибір конкретного алгоритму повинен залежати від конкретного завдання та вимог до точності та швидкодії. Також варто зазначити що попереднє навчання алгоритмів з використанням даних які не відносяться до безпосередніх є недоцільним.

У третьому розділі були сформовані вимоги до програмної системи яка допоможе водіям знаходити місця для паркування набагато швидше та ефективніше. Також був розроблений концепт-документ до розроблюваного програмного забезпечення. Проведена розробка архітектури програмного

забезпечення. Під час якої користуючись каскадною моделлю побудови багаторівневого процесу розробки сплановані етапи затвердження вимог, проектування, розробки, тестування, розгортання та підтримки продукту. В процесі роботи отриманий проект з попередньою назвою «ParkingBot» який допоможе водіям швидко та ефективно знаходити вільні місця на автостоянках. Програмний продукт використовує сучасний та найефективніший алгоритм розпізнавання об’єктів YOLOv8.

Подальше дослідження буде присвячене клієнтської та серверної частини програмного додатку які відповідають умовам сучасного ринку в області рішень інтелектуального паркування.

Практична значимість отриманих результатів полягає у дослідженні та розробці методу за допомогою якого можливо проводити підрахунок зайнятих та вакантних місць на автостоянці та рекомендувати користувачам певні дії або допомагати у прийнятті рішень.

За темою кваліфікаційної роботи були опубліковані англомовні тези доповіді на 27-му Міжнародному молодіжному форумі «Радіоелектроніка і молодь у XXI столітті».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Department of Economic and Social Affairs. Population Division. World Urbanization Prospects. The 2018 Revision (2019).

2. N. O. Semchenko. "Study of the influence of economic factors on the level of automobilization". Communal management of cities, 2020, volume 6, issue 159 (2020). doi:10.33042/2522-1809-2020-6-159-161-168.

3. T. Perković, P. Šolić, H. Zargariasl, D. Čoko, Joel J.P.C. Rodrigues. Smart Parking Sensors: State of the Art and Performance Evaluation, Journal of Cleaner Production, Volume 262, 20 July 2020, 121181, 2020.

4. O. Pavlova, V. Kovalenko, T. Hovorushchenko, V. Avsiyevych. Neural network based image recognition method for smart parking, Computer systems and information technologies (2021): doi: 10.31891/CSIT-2021-3-7.

5. Kamran Sattar Awaisi, Assad Abbas, Mahdi Zareei, Hasan Alikhattak, Muhammad U. S. Khan, Mazhar Ali,Ikram Ud Din, Sajid Shah. Towards a Fog Enabled Efficient Car Parking Architecture, IEEE Access (Volume: 7) (2019): pp. 159100 – 159111. doi: 10.1109/ACCESS.2019.2950950.

6. НОВОСАД Марія-Руслана. Асистент паркування як модуль системи розумного міста, Вісник Хмельницького національного університету, 2022, issue 5, volume 313 (2022): С. 56-60. doi: 10.31891/2307-5732-2022-313-5-56-60.

7. Lea Duji´c Rodi´c, Toni Perkovi´c, Tomislav Županovi´c and Petar Šoli´c. Sensing Occupancy through Software: Smart Parking Proof of Concept. Electronics 9(12):2207 (2020). doi: 10.3390/electronics9122207.

8. V. Masenko. A municipal parking lot will be built in the Passage, The Village (2021). Дата оновлення: 14.12.2021. URL: https://www.the-village.com.ua/village/city/city-news/320085-na-pasazhi-zroblyat-munitsipalniy-parking (дата звернення: 15.02.2023).

9. Smart Parking Technologies Market, Electronics and Semiconductors (2022). Дата оновлення: 04.2022. URL: https://www.transparencymarketresearch.com/smart-parking-technologies-market.html (дата звернення: 16.02.2023).

10. Abrar Fahim, Mehedi Hasan, Muhtasim Alam Chowdhury. Smart parking systems: comprehensive review based on various aspects, Heliyon, volume 7 issue 5 (2021). doi: 10.1016/j.heliyon.2021.e07050.

11. SmartPark System. URL: https://www.smartparking.com/smartpark-system (дата звернення: 17.02.2023).

12. Adam Geitgey. Snagging Parking Spaces with Mask R-CNN and Python, (2019). Дата оновлення: 22.01.2019. URL: https://medium.com/@ageitgey/snagging-parking-spaces-with-mask-r-cnn-and-python-955f2231c400 (дата звернення: 17.02.2023).

13. Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick. Mask R-CNN, Facebook AI Research (FAIR) (2018). doi: 10.48550/arXiv.1703.06870.

14. EasyPark - The EasyPark way of parking and charging. URL: https://www.easypark.com/en-it/how-it-works (дата звернення: 18.03.2023).

15. Smartiple | Camera car/boat parking occupancy detection. URL: https://smartiple.com/en/ (дата звернення: 19.03.2023).

16. Cleverciti | Smart Parking for Smart Cities. URL: https://www.cleverciti.com/ (дата звернення: 19.03.2023).

17. Cleverciti sensor. Cutting-edge technology for clever parking. URL: https://www.cleverciti.com/en/innovations/cleverciti-sensor (дата звернення: 19.03.2023).

18. Classification, Object Detection and Image Segmentation. URL: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation (дата звернення: 17.02.2023).

19. A. Rizzoli. The Ultimate Guide to Object Detection, V7, (2023). Дата оновлення: 03.02.2023. URL: https://www.v7labs.com/blog/object-detection-guide (дата звернення: 17.02.2023).

20. K. Bharath. Object Detection Algorithms and Libraries, MLOps Blog, (2023). Дата оновлення: 25.01.2023. URL: https://neptune.ai/blog/object-detection-algorithms-and-libraries (дата звернення: 17.02.2023).

21. R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. Tech report (v5), (2014). doi: 10.48550/arXiv.1311.2524.

22. R. Girshick. Fast R-CNN, (2015). doi: 10.48550/arXiv.1504.08083.

23. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, (2016). doi: 10.48550/arXiv.1506.01497.

24. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector, (2016). doi: 10.1007/978-3-319-46448-0_2.

25. J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection, (2016). doi: 10.48550/arXiv.1506.02640.

26. Ultralytics YOLOv8: The State-of-the-Art YOLO Model, (2023). URL: https://ultralytics.com/yolov8 (дата звернення: 18.03.2023).

27. Rafael Padilla, Sergio L. Netto, Eduardo A. B. da Silva. A Survey on Performance Metrics for Object-Detection Algorithms, (2020). doi: 10.1109/IWSSIP48289.2020.

28. Saad Mohammad Alkentar, B. Alsahwa, A. Assalem, D. Karakolla. Practical comparation of the accuracy and speed of YOLO, SSD and Faster RCNN for drone detection. Vol. 27 No. 8 (2021): Journal of Engineering. 2021-08-01. doi: 10.31026/j.eng.2021.08.02.

29. Trupti Mahendrakar, Andrew Ekblad, Nathan Fischer, Ryan White, Markus Wilde, Brian Kish, Isaac Silver. Performance Study of YOLOv5 and Faster R-CNN for Autonomous Navigation around Non-Cooperative Targets. 2022 IEEE Aerospace Conference (AERO), (2022). doi: 10.1109/AERO53065.2022.9843537.

30. NDISPark Dataset. Night and Day Instance Segmented Park dataset. Дата оновлення: 26.11.2021. URL: https://www.ai4europe.eu/research/ai-catalog/ndispark-dataset (дата звернення: 17.02.2023).

31. Swee Xiao Qi. Parking Lot Availability Computer Vision Project. Дата оновлення: 28.07.2022. URL: https://universe.roboflow.com/swee-xiao-qi/parking-lot-availability. (дата звернення: 22.03.2023).

32. Brad Dwyer. PKLot Computer Vision Project. Дата оновлення: 05.01.2021. URL: https://universe.roboflow.com/brad-dwyer/pklot-1tros (дата звернення: 22.03.2023).

33. Bezsmertnyi O., Golian N., Golian V., Afanasieva I. Behavior Driven Development Approach in the Modern Quality Control Process // Problem of Infocommunications. Science and Technology (PIC S&T′2020), (2020). doi: 10.1109/PICST51311.2020.9467891

34. Nikitin D., Golian V., Golian N., Dudar Z. Automated Software Development with Finite-State Machine Based Structures. Proceedings Book, (2022).

35. Golian N., Golian V., Afanasieva I. Black and white-box unit testing for webapplications. Bulletin of the National Technical University "KhPI". Series: System analysis, management and information technologies, 1 (7). (2022) pp 79 – 83. doi: 10.20998/2079-0023.2022.01.13