МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра комп'ютерних наук

СУДЕЙЧЕНКО Денис Владиславович

Математичне та програмне забезпечення для паркування автомобілів / Mathematical Tools and Software for Car Parking

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗзм-21
Д. В. Судейченко

_____

Науковий керівник:
к.т.н., доцент Р. П. Шевчук

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 11
1.1. Особливості організації процесу паркування. 11
1.2. Аналіз відомих підходів для пошуку вільних місць на парковках..... 16
1.3. Огляд готових рішень щодо створення нейронних мереж..... 19
Висновки до першого розділу 23
РОЗДІЛ 2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПАРКУВАННЯ АВТОМОБІЛІВ 24
2.1. Просторово-точкова імітаційна модель організації паркінгу автомобілів 24
2.2. Метод динамічного управління навігацією 33
2.3. Концепція та сценарії використання VDA..... 37
Висновки до другого розділу..... 40
РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПАРКУВАННЯ АВТОМОБІЛІВ 41
3.1. Проектування системи 41
3.2. Програмна реалізація системи..... 48
3.3. Експериментальні дослідження та тестування нейронної мережі 56
3.4. Програмна реалізація клієнтського додатку для пошуку вільних місць на парковках..... 58
Висновки до третього розділу 60
ВИСНОВКИ..... 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63
ДОДАТОК А ЛІСТИНГ PYTHON МОДУЛІВ НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ ПРИ АНАЛІЗІ ЗОБРАЖЕНЬ З КАМЕР СПОСТЕРЕЖЕННЯ ..... **Помилка! Закладку не визначено.**

ВСТУП

Актуальність теми. На сьогоднішній день пошук вільного місця на парковці у великому завантаженому місті є досить складним завданням для власників автомобілів. Особливо гостро ця проблема проявляється у парковок біля великих торгово-розважальних центрів, магазинів, лікарень та інших громадських місць. Багато водіїв змушені об'їжджати паркування кілька разів, щоб знайти вільне місце для паркування, хоча його може і не бути в цій паркувальній зоні. Як наслідок, витрачається багато часу, водії створюють перешкоди на дорозі в процесі пошуку місця для автомобіля, що тягне за собою пробки на дорогах, також відбувається зайва витрата палива і відповідно збільшення викиду вихлопних газів, що призводить до погіршення екологічної ситуації у місті.

Таким чином, пошук оптимального вирішення даної проблеми може дозволити не тільки заощадити час і гроші водіїв, а й сприяти зниженню рівня забруднення навколишнього середовища.

Можливим вирішенням цього питання може стати встановлення камер відеоспостереження на кожному паркуванні. За допомогою цієї системи можна буде отримати зображення з паркувальної зони і із застосуванням нейромережевих технологій дізнатися кількість вільних місць на парковці та їх розташування.

Таким чином, наразі актуальним стає питання регулювання організації місць для паркувань. Відомо, що кошти від оплати за паркування надходять до місцевого бюджету і найчастіше використовуються муніципалітетами на підтримку та розвиток інфраструктури в галузі організації парковок автомобілів.

Оплата за паркування, крім того, є одним із інструментів регулювання політики організації паркування. Для того щоб удосконалювати політику організації паркувань крім економічних факторів необхідний інструмент,

який дасть можливість особам, що приймають рішення оцінювати та вибирати ефективний варіант організації парковок автомобілів в умовах обмеженої кількості місць для паркування в центральній частині міської території.

Зв'язок роботи з науковими програмами, планами, темами

Напрямок виконаних досліджень безпосередньо пов'язаний з науково-дослідним напрямком кафедри “комп’ютерних наук” Західноукраїнського національного університету.

Мета і задачі дослідження

Метою роботи є розробка методів та засобів для ефективного паркування автомобілів у центральній частині великого міста.

Відповідно до мети у магістерській роботі необхідно вирішити такі завдання:

1. Проаналізувати предметну область дослідження, а саме особливості функціонування парковок та см процес паркування.

2. Дослідити відомі програмно-апаратні рішення пошуку вільних паркувальних місць.

3. Дослідити особливості застосування нейронних мереж для пошуку вільних паркувальних місць.

4. Розробити просторово-точну математичну модель для орагнізації процесу паркування в центральній частині великого обласного центру.

5. Розробити метод динамічного управління навігацією на основі згорткових нейронних мереж

5. Реалізувати запропоновані методи у вигляді програмної системи та мобільного додатку та провести відповідні експериментальні дослідження.

Об’єкт дослідження – процес організації паркування.

Предмет дослідження – методи та програмні засоби для пошуку парковок та вільних паркомісць.

Методи дослідження

В роботі використовувалися методи на основі штучних нейронних мереж, математичної статистики , об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів. Розроблена просторово-точна модель, яка дозволяє визначати вплив різних просторових сценаріїв на організацію паркувань в межах великого міста. На відміну від відомих, традиційних моделей цей підхід імітує дії кожного водія у просторово-точному міському середовищі, що значно підвищує значущість результатів моделювання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1. Особливості організації процесу паркування.

З кожним роком рівень автомобілізації стає дедалі вищим. Наприклад, в Україні за останні 20 років він виріс більш ніж у 2 рази (зі 132 авто на 1000 осіб до 330 авто на 1000 людина) [1]. За даними державної статистики [1-4], на початок 2022 р. зростання кількості автомобілів за останні десять років збільшилася у півтора рази, а кількість паркувальних місць практично не змінилася. На початок 2022 р. кількість автомобілів у середньому становила 319 на 1000 чоловік, що відповідає 1 автомобіль на сім'ю. Водночас міська інфраструктура ускладнюється і виникає проблема - збільшений транспортний потік і сильно завантажені місця для паркування. Виникає необхідність правильної логістики процесу паркінгу.

Найбільш відомою методикою оцінки рівня розвитку логістичної інфраструктури слід знати індекс LPI Світового банку, заснований на розрахунку інтегрального показника, що включає шість комплексних складових. Кількісне значення індексу ефективності логістики дозволяє оцінити рівень розвитку логістичної системи країни, в тому числі і транспортної інформації структури. Нестача паркувальних місць, так само як і велика кількість «пробок» на дорогах, на сьогоднішній день у більшості мегаполісів входять до списку найбільш гострих проблем. В великих містах проблема пошуку паркувального місця є однією з найважливіших.

Паркування – це невіддільна частина офісного, житлового та адміністративного комплексів, а також великих об'єктів культурного відпочинку людей, торгових та торгово-розважальних центрів. Одним з елементів комплексної проблеми забезпечення городян паркувальними

місцями є планування та керування паркуванням біля великих торгово-розважальних центрів. Найчастіше, приїжджаючи туди водій не може знайти місце на відкритих парковках, і йому доводиться тривалий час шукати його в іншому місці, тим самим відбувається збільшення забруднення навколишнього середовища та зростання кількості ДТП. Особливо яскраво це спостерігається в густонаселених містах, де людина може тривалий час шукати місце для паркування, коли всі місця на парковках зайняті. Отже, існують побоювання з приводу заторів на дорогах, а також витрати часу, нераціональної витрати палива та погіршення якості повітря через викиди дрібного пилу в житлових та торгових районах [4-7].

Віртуальний асистент – система автоматизації взаємодії з користувачем, реалізована на основі штучного інтелекту у діалоговому форматі. Сервіс здійснює текстові та голосові консультації, обробку заявок та підтримку за напрямами діяльності компанії. Користувач надсилає запит у чаті у вільній текстовій або голосовій формі, а система надає швидку відповідь [8-10].

Все різноманіття віртуальних помічників можна розділити на побутові, що використовуються в особистих цілях, для дозвілля, і не пов'язані з робочим процесом і функціонуванням бізнес-систем, та робітничі, що використовуються для допомоги на робочому місці та у вирішенні бізнес-завдань, додатки [11-13].

Побутові віртуальні помічники діляться на три великі групи:

- для керування пристроєм, універсальні. Програми для щоденного побутового використання, що дозволяє здійснювати інтернет-пошук, виконувати нескладні доручення (наприклад, замовлення таксі) та приймаючі команди природною мовою у вигляді тексту або мови;
- для взаємодії з бізнес-додатками. Помічники, інтегровані в мобільні або веб-програми різних компаній, що спрощують взаємодії клієнтів з компанією та автоматизують процес цієї взаємодії;

- предметно-орієнтовані послуги. Програми, спрямовані на вирішення специфічних особистих завдань, пов'язаних із проведенням різних форм дозвілля, а також ділового планування.

Основна мета VDA – допомогти водіям знайти підходящі паркувальні місця, в режимі онлайн контролювати доступність автостоянок та перенаправити водіїв, коли кількість вільних місць впаде до критичного рівня [14].

На даний момент часу є ряд мобільних додатків, які надають водіям можливість знаходження вільного місця на парковці. Кожне з них має свої особливості, переваги та недоліки, у зв'язку з цим проведемо аналіз трьох найпопулярніших додатків в офіційному магазині ОС Android. Додаток «Київ Цифровий» (рисунок 1.1). містить базу паркувань при торгових центрах, вокзалах та аеропортах, а також платних та безкоштовних стоянках Києва.

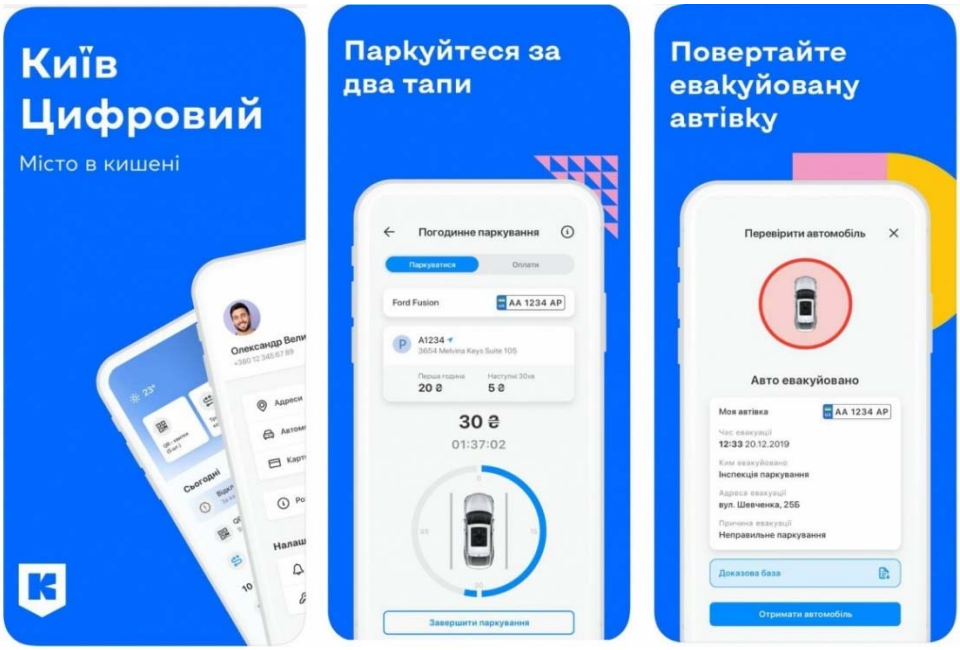

Рис.1.1. Мобільний додаток «Київ Цифровий»

На головному екрані представлена карта міста з доступними паркуванням. Щоб зайняти місце, слід вибрати парковку, зручну за розташуванням та прийнятну за вартістю, та натиснути «додати автомобіль».

Далі з'являється вікно, де потрібно ввести дані про машину і потім оплатити паркування. У програмі можна прив'язати банківську картку для швидкої та зручної оплати, а також змінити або завершити поточне паркування. У бічному меню програми є можливість подивитися історію паркувань, а також штрафи та евакуації.

У додатку «Parkopedia» (рисунок 1.2) [17], як і в інших аналогах, використовується база місць для паркування, що дає можливість прокласти маршрут до будь-якого з них. Від попереднього додатка це відрізняється тим, що працює не в конкретному місті, а по всьому світу. На головному екрані розташована картка з доступними парковками, які можна відсортувати за ціною. Після вибору паркування показується її вартість, розташування, місткість та відстань. У програмі є список всіх доступних парковок, а також у преміум-версії доступний перегляд вільних місць. Як запевняє розробник, це додаток допоможе знайти паркувальні місця навіть у найекзотичніших країнах, однак у деяких містах паркування взагалі не показуються, навіть якщо вони там є.

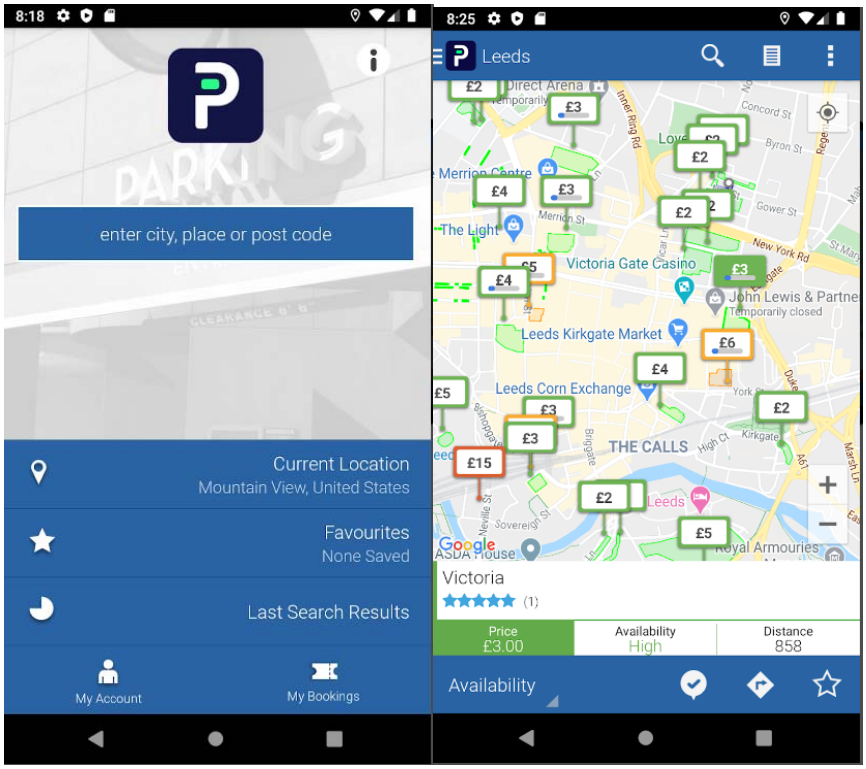

Рис.1.2. Мобільний додаток «Parkopedia»

Додаток «Parking» (рисунок 1.3) відрізняється від попередніх аналогів тим, що запам'ятовує місце, де водій залишив свою машину.

За допомогою цієї програми можна на карті поставити позначку, де знаходиться автомобіль, і виставити час паркування. На головному екрані, як і у всіх аналогах, представлена карта, причому не тільки конкретного міста. Внизу карти розташована панель управління, за допомогою якої можна змінити обране місце для паркування, видалити його, прокласти до нього маршрут тощо.

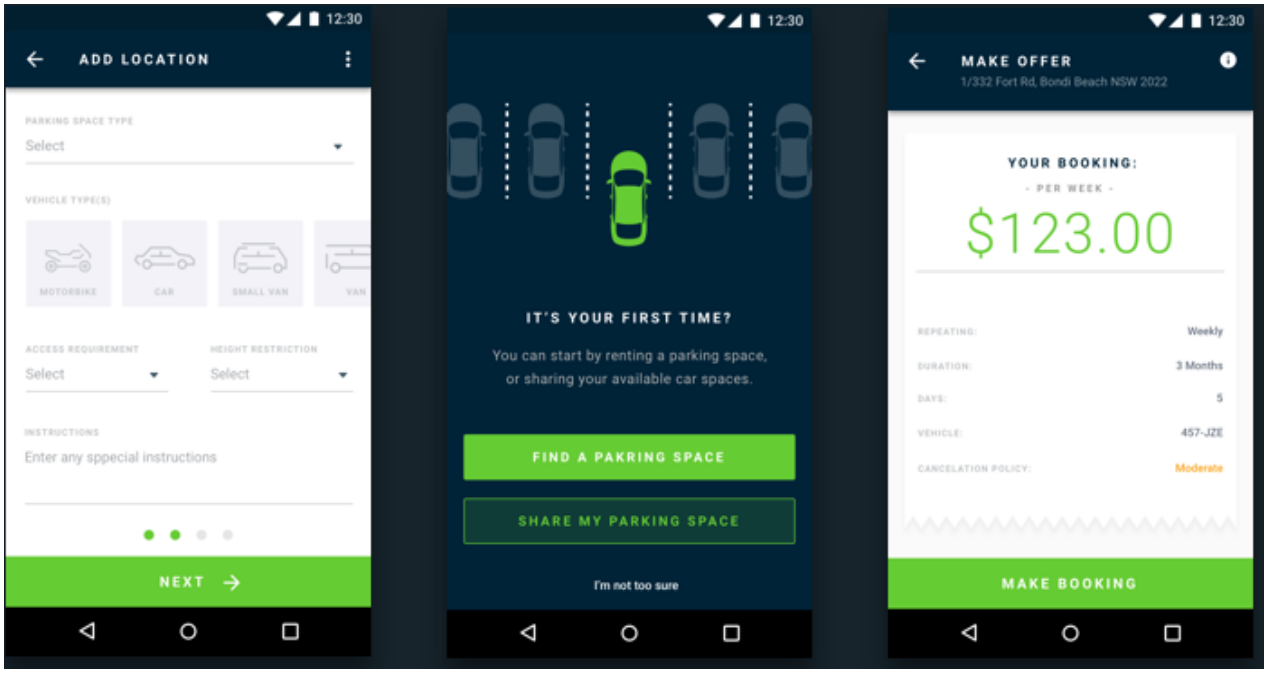

Рис.1.3. Мобільний додаток «Parkopedia»

При виборі місця на карті відкриється панель, де можна залишити нотатки, додати фотографію та виставити час паркування. У програмі працює функція навігації, достатньо натиснути на карту, поставити мітку і відкриється програма картки, встановлена на телефоні, за допомогою яких виставляється потрібний маршрут.

Докладно розглянувши програми, можна виділити наступні особливості, представлені в таблиці 1.1.

Таблиця 1.1.

Порівняння додатків для пошуку паркувальних місць

Додаток	Переваги	Недоліки
Додаток «Київ Цифровий»	З'являються всі доступні паркування та докладна інформація про них. Через програму можна відразу ж сплатити вартість паркування, подивитися штрафи та евакуації. Можна прокласти маршрут до обраного паркування.	Працює лише в одному конкретному місті України, не показуються вільні місця на парковці.
Додаток «Parkopedia»	Зручний фільтр на кшталт паркування, часу та інших характеристик. Можна прокласти маршрут до обраного паркування.	Показується вся карта, а не конкретного міста, але в деяких містах не видно паркування взагалі. Є інформація про кількість всіх місць на паркуванні, але інформацію про вільні місця можна отримати лише в преміум-версії.
Додаток «Parking»	Можна вибрати будь-яку точку на карті, щоб паркувати автомобіль, а також прокласти маршрут до обраного місця	Не показуються безпосередньо паркування, немає інформації про них (про вільні місця і т.д.). З'являється вся карта, а не конкретного міста

1.2. Аналіз відомих підходів для пошуку вільних місць на парковках

На сьогоднішній день існує багато підходів до вирішення задачі визначення вільних місць на парковках.

В одній із робіт було запропоновано алгоритм автоматичного підрахунку машин [16]. За допомогою спеціальних сенсорів, встановлених на в'їзді паркування, виходило досить точно встановлювати кількість вільних місць на ній. Недоліком цього підходу було те, що не можна було визначити їх розташування, а також вмонтування даних пристроїв пошкоджувало дорогу.

Інший підхід полягав у встановленні спеціальних датчиків на кожному паркувальному місці [17]. Вони дозволяли ефективно визначати кількість та розташування вільних паркувальних місць. Проте впровадження таких механізмів на кожному паркуванні міста буде досить дорогою операцією, а часом неефективною, коли автомобіль займає відразу кілька місць на паркуванні.

Інший запропонований метод для детектування вільних місць на паркування ґрунтується на машинному навчанні [18]. Використовуючи комбінацію з детектора на основі гістограм спрямованих градієнтів (Histogram of Oriented Gradients, HOG [19]) та ключових ознак автомобілів (крайові точки) вдалося ефективно розпізнавати вільні місця, але у випадках частково зайнятого місця ефективність падала. Даний підхід заснований на фіксованій розмітці, і буде не придатний на парковках без неї.

Підхід із фіксованою розміткою. Спочатку, в ході вирішення даної проблеми, використовувався підхід з фіксованою розміткою та застосуванням згорткових нейронних мереж. Таким чином, завдання розбивалося на дві підзадачі:

1. Необхідно було зробити детальну розмітку паркувальної зони, таким чином визначити розташування всіх доступних місць для паркування.

2. Застосувати згорткову нейронну мережу до кожного місця на парковці, щоб можна було встановити, чи воно є вільним або зайнятим.

Оскільки камера на парковці статична, то було досить одного разу визначити становище всіх місць на парковці та згодом використовувати цю карту при детектуванні місць згортковою нейронною мережею, яка була попередньо навчена на зображеннях з автомобілем і без.

В якості нейронної мережі було вирішено використати модель нейронної мережі VGG16. Архітектура нейронної мережі представлена нижче на рисунку 1.4.

Рис. 1.4. Архітектура нейронної мережі VGG16

Оскільки дана мережа має бути застосована для конкретного завдання і відповідно на іншому наборі даних, то необхідно прибрати останній шар з моделі, що відповідає за класифікацію. Таким чином, використовуватиметься лише згорткова частина мережі (рисунок 1.5).

Рис. 1.5. Згорткова частина мережі

Потім до згорткової частини мережі необхідно додати новий клас специфікатор. Його архітектура представлена нижче на рисунку 1.6.

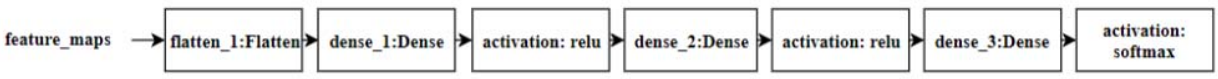

Рис. 1.6. Архітектура класифікатора

У систему завантажувалося зображення з паркування та координати паркувального місця. Далі кожне місце подавалося на вхід згорткової нейронної мережі, яка визначала, чи є місце вільним або зайнятим.

На жаль, даний підхід показав свою неефективність на парковках без розмітки і у випадках некоректно зайнятого місця, а саме коли одним

автомобілем було зайнято частину місця, а іншим частину наступного місця (рисунок 1.7).

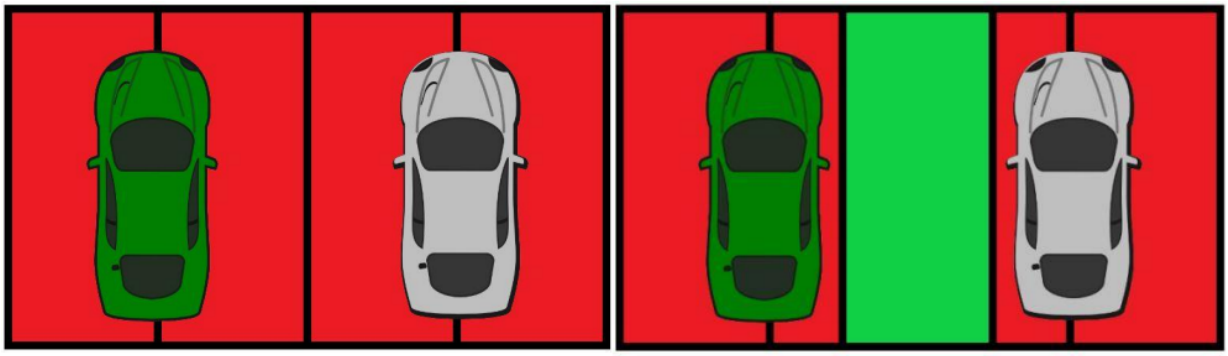

Рис. 1.7. Втрата вільного місця

Оскільки місця були фіксовані, згорткова нейронна мережа не змогла побачити це вільне місце.

1.3. Огляд готових рішень щодо створення нейронних мереж

NumPy (рисунок 1.8). Ця бібліотека мови Python призначена для наукових обчислень. Вона підтримує великі багатовимірні масиви та матриці, а також включає великий набір високорівневих математичних функцій, призначених для операцій з багатовимірними матрицями та масивами. Ця бібліотека підвищує продуктивність та прискорює виконання операцій з допомогою механізму векторизації.

SciPy (рисунок 1.9). Бібліотека, що містить у собі модулі для лінійної алгебри, аналізу зображень, оптимізації, інтеграції та статистики. Основною структурою даних у SciPy є багатовимірний масив, реалізований за допомогою NumPy. Також для даної бібліотеки присутня докладна документація на кожну функцію.

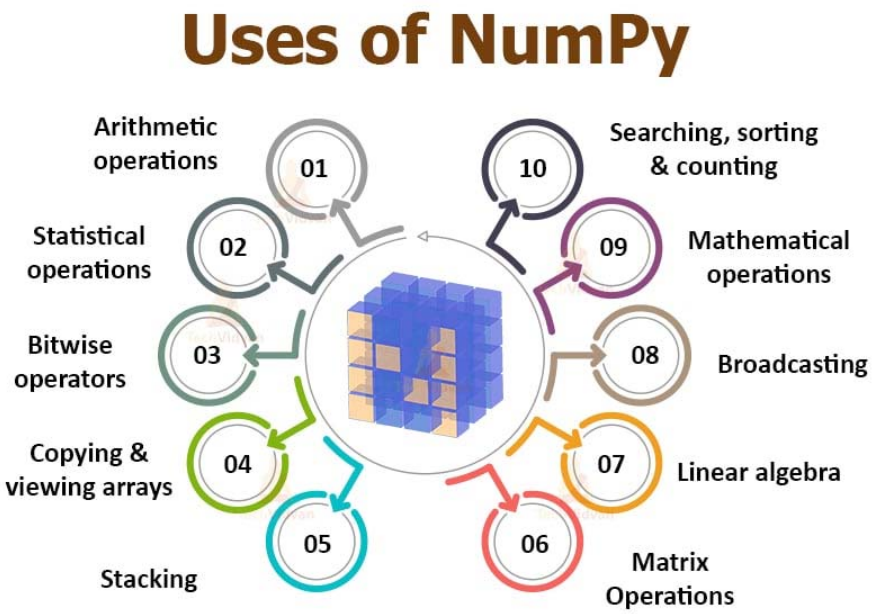

Рис. 1.8. NumPy

Рис. 1.9. SciPy

Mathplotlib (рисунок 1.10). Ця бібліотека є основним інструментом для візуалізації даних на мові Python. Включає можливість створення лінійних графіків, діаграм, спектрограм, гістограм, графіків розсіювання, а також інших форм візуалізації даних.

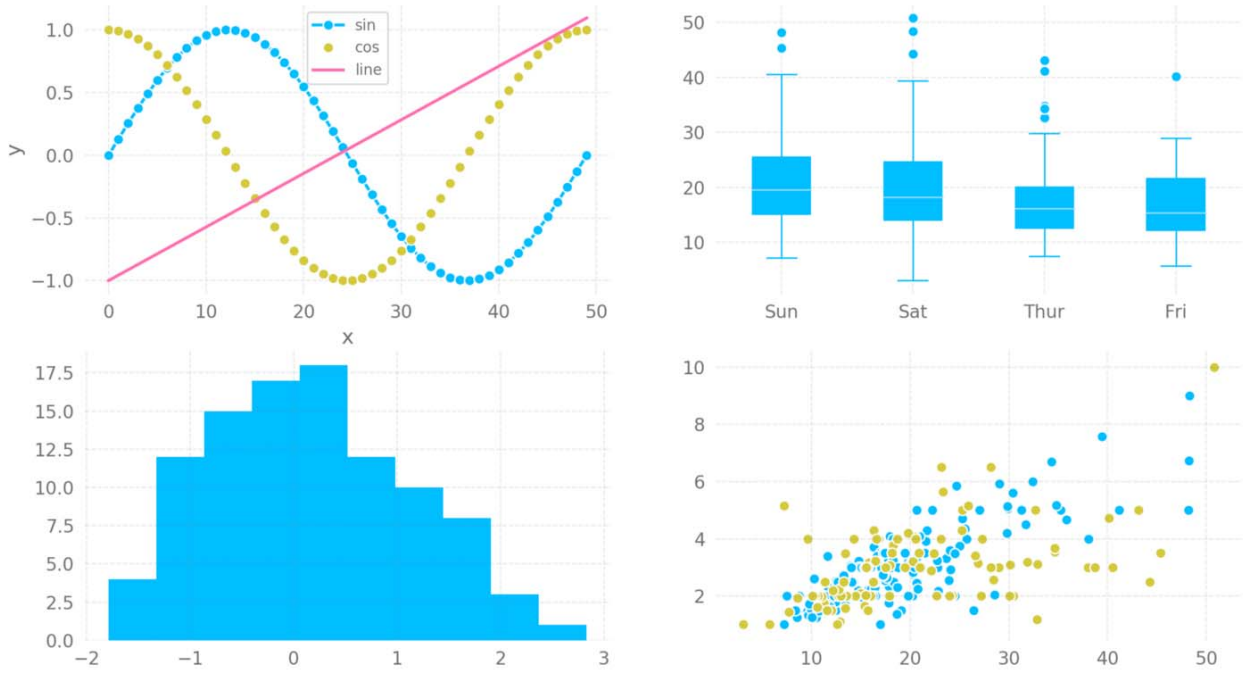

Рис. 1.10. Mathplotlib

TensorFlow (рисунок 1.11). Бібліотека від Google була розроблена спеціально для навчання нейронних мереж. Для зберігання даних у цій бібліотеці використовуються багатовимірні масиви – тензори, а для представлення нейронної мережі використовуються графи.

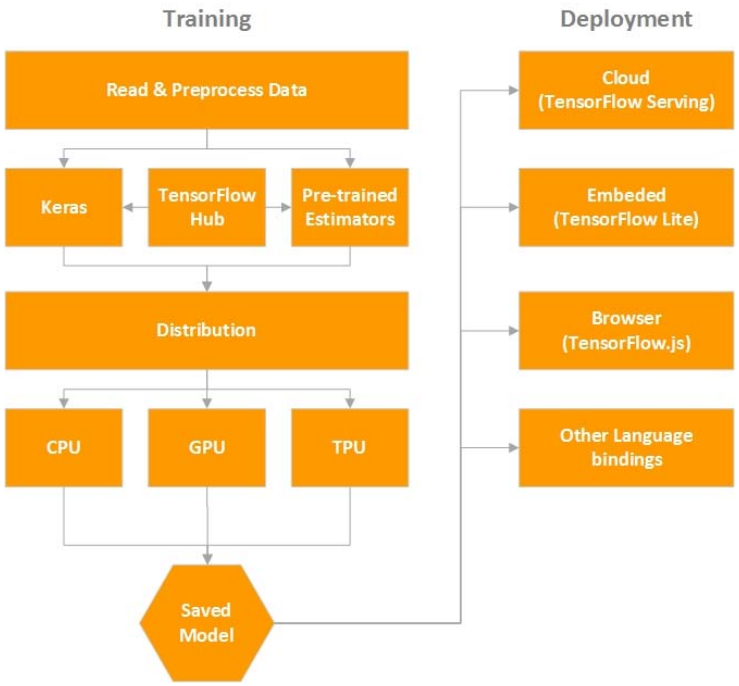

Рис. 1.11. TensorFlow

Keras (рисунок 1.12). Відкрита програмна бібліотека для глибокого навчання, написана мовою Python. Є надбудовою над Tensorflow. Бібліотека інтуїтивно проста і зрозуміла у використанні. Існує хороша документація та величезна кількість прикладів використання, у тому числі для завдань класифікації.

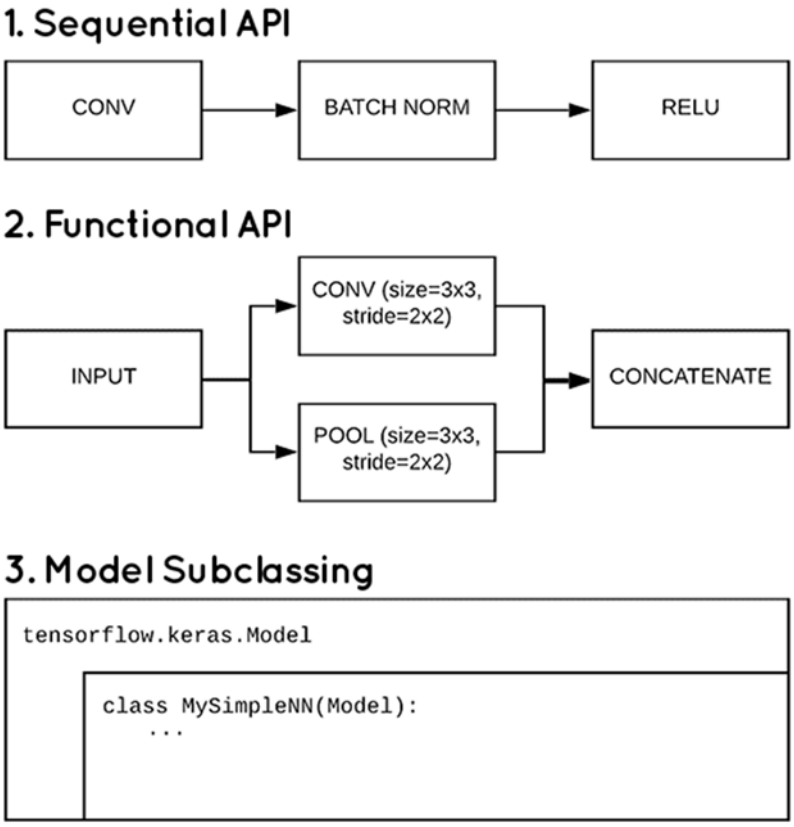

Рис. 1.12. Keras

Огляд існуючих рішень показав, що завдання визначення вільних місць на парковках є актуальним на сьогоднішній день. На даний момент існує багато реалізованих проектів з детектування вільних місць на парковках. Однак точність детектування в готових продуктах відбувається не завжди коректно. Також, варто відзначити існування широкого ряду бібліотек для створення та навчання нейронних мереж. Використання таких бібліотек може значно полегшити завдання реалізації. Зважаючи на всі описані вище

фактори, можна зробити висновок, що дослідження в даній галузі є необхідним, а також існує потреба у реалізації прикладних рішень.

Висновки до першого розділу

1. Огляд існуючих рішень показав, що завдання визначення вільних місць на парковках є актуальним на сьогоднішній день. На даний момент існує багато реалізованих проектів з детектування вільних місць на парковках. Однак точність детектування в готових продуктах відбувається не завжди коректно. Також, варто відзначити існування широкого ряду бібліотек для створення та навчання нейронних мереж.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПАРКУВАННЯ АВТОМОБІЛІВ

2.1. Просторово-точкова імітаційна модель організації паркінгу автомобілів

Нестача місць для паркування легкового автотранспорту відчувається на всіх територіях великих міст, а особливо в їх центральних районах, що породжує низку гострих проблем: зниження пропускної спроможності вулично-транспортної мережі міста, погіршення умов безпеки руху транспорту, ускладнення проїзду громадського транспорту та екстрених служб (аварійних, рятувальних, медичних, пожежних), труднощі у проведенні механізованого прибирання вулиць, дискомфорт та підвищена небезпека руху пішоходів, погіршення екологічної обстановки та ін.

Значне перевищення попиту на місця паркувань над кількістю паркувальних місць, що надається, є характерною рисою центрів великих міст. Крім того, негативний вплив на умови та безпеку дорожнього руху у містах надає процес неорганізованих парковок легкових автомобілів на вулично-дорожній мережі з порушенням правил дорожнього руху.

За наявними нечисленними даними вітчизняних та зарубіжних досліджень, частка дорожньо-транспортних пригод, пов'язаних із процесом паркування легкових автомобілів у великих містах, становить від 5 до 15%. Як правило, такі ДТП виникають при маневруванні автомобілів, що під'їжджають до краю проїжджої частини для зупинки або вбудовуються в транспортний потік. Разом з тим, в умовах гострого дефіциту машиномісць на позавуличних стоянках, проїжджа частина надає практично єдину можливість здійснити стоянку при внутрішньоміських поїздках автомобілем.

Таким чином, наразі актуальним стає питання регулювання організації місць для паркувань. Відомо, що кошти від оплати за паркування надходять до місцевого бюджету і найчастіше використовуються муніципалітетами на підтримку та розвиток інфраструктури в галузі організації парковок автомобілів.

Оплата за паркування, крім того, є одним із інструментів регулювання політики організації паркування. Для того щоб удосконалювати політику організації паркувань крім економічних факторів необхідний інструмент, який дасть можливість особам, що приймають рішення оцінювати та вибирати ефективний варіант організації парковок автомобілів в умовах обмеженої кількості місць для паркування в центральній частині міської території.

В рамках магістерського дослідження побудовано просторово-точну імітаційну модель організації паркінгу автомобілів у центральній частині великого міста. Модель організації паркінгу заснована на моделюванні дій кожного водія, що бере участь у процесі паркування свого автомобіля та охоплює основні етапи цього процесу: рух до місця призначення, пошук та залишення паркувального машиномісця.

У просторово-точній моделі організації паркінгу зроблено спробу врахувати дії кожного водія під час паркування, яке залежить від кількості доступних машиномісць, але найголовніше - те, що автомобілі знаходяться у просторі, що моделюється реальними верствами геоінформаційної системи.

Деталізоване міське середовище є основою для процесу аналізу часу пошуку місця для паркування, часу на пересування від місця паркування до місця призначення. Просторово-точна модель дозволяє вивчити потребу в паркувальних машиномісцях у центральній частині міської території у денний час.

Просторово-точна модель розроблялося відповідно до двох основних принципів:

- Перший принцип полягає в тому, що модель представляє просторово-точну модель, яка побудована на основі геоінформаційної системи, що містить тематичні шари, найбільш важливі для дослідження процесу паркування. Як такі елементи були використані елементи вулично-дорожньої мережі міста - сегменти вулиць, паркувальні місця на головних вулицях, паркувальні місця поза головними вулицями, будівлі, точки громадського транспорту на головних вулицях.

- Другий принцип полягає в тому, що просторово-точна модель розроблялася як агентно-орієнтована модель, яка дозволяє моделювати пересування кожного автомобіля, який рухається до пункту призначення, визначає місце для паркування та залишає паркувальне місце. Основним елементом моделі є опис дій об'єкта, тобто автомобіля. Просторово-точна модель містить правила, які визначають для кожного об'єкта моделі порядок руху до пункту призначення, умови пошуку машиномісця, умови паркування об'єкта та умови залишення паркування. Крім того правила визначають поведінку об'єкта у разі нестачі паркувальних місць на або поза головними вулицями моделюваної міської території.

Просторово-точна модель функціонує на основі верств міської території, отриманих у середовищі ArcGIS і може розглядатися як зовнішній додаток до ArcGIS, розроблений в середовищі Microsoft Visual Studio. Інтерфейс просторово-точної моделі містить набір інструментів для вибору області моделювання, встановлення сценаріїв моделювання, так і зберігання отриманих результатів. Обмін даними між ArcGIS та просторово-точною моделлю здійснюється файлами у форматі XML.

Просторова база даних у моделі складається з просторових шарів високої роздільної здатності (для створення шарів використовувався масштаб 1:2000) та непросторових таблиць. Такими просторовими шарами є:

- вулично-дорожня мережа, кожен сегмент якої характеризується ємністю машиномісць для паркувань;

- дозволу чи заборони поворотів;

- будинки на міській території, які визначають місця призначення для водіїв.

На основі вихідних шарів будуються два додаткові шари. Шар "ліній" для моделювання вулиць з двостороннім рухом, які розташовані по обидва боки від центральної лінії вулиць та ліній, що моделюють односторонній рух, що показано на рисунку 2.1.

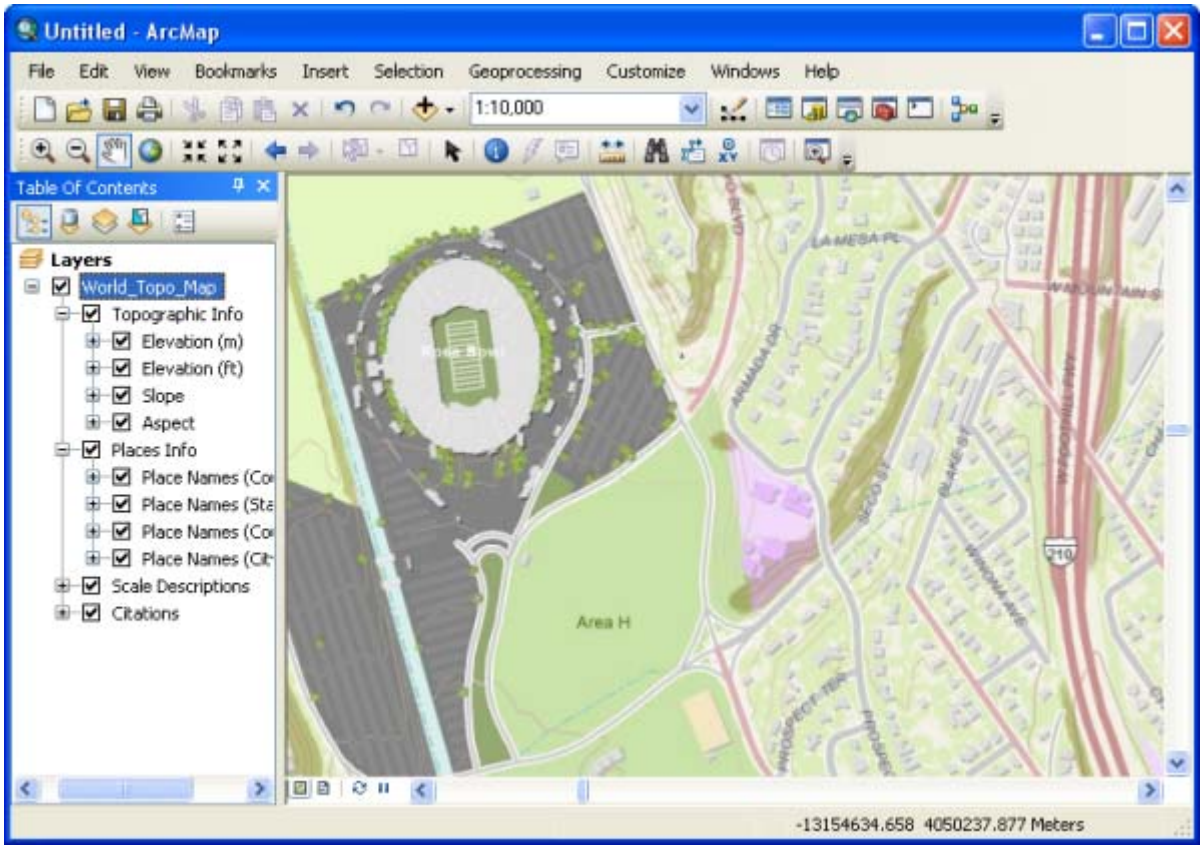

Рис. 2.1. Основні та вторинні шари просторово-точної моделі у середовищі

Паркувальні місця, розташовані на головних вулицях, моделюються у вигляді точок, які збудовані по обидва боки від центральної лінії сегмента вулиці. Середня дистанція між паркувальними місцями обрана 5 метрів, на підставі натурних спостережень.

Шар паркувальних машиномісць включає всі фізично існуючі паркувальні місця, в тому числі і ті, де парковка не дозволена, але технічно можлива.

Модель реалізується в дискретному просторі та часі: за кожну ітерацію об’єкт (автомобіль) змінює своє становище з урахуванням швидкості руху.

Часова частота моделювання залежить від довжини паркувального машиномісця, яке визначено як 5 метрів. Відповідно, інтервал часу обраний $\Delta t = 1$сек., за умови що автомобіль рухається зі швидкістю $V = 18$ км/год, щоб транспортний засіб міг за один крок ітерації просунутися на 5 метрів.

У моделі швидкість руху V_s(км/год) перераховується швидкість V_m , яка вимірюється в довжинах машиномісць для кожного інтервалу модельного часу. Модельна швидкість транспортного засобу V_m у моделі представляється як:

$$V_m = V_{m,int} + V_{m,dec}\,, \tag{2.1}$$

де $V_{m,int}$ – ціла частина V_m; $V_{m,dec}$ - десяткова частина V_m.

Як приклад, якщо швидкість 20 км/год, довжина місця для паркування 5 метрів, за умови, що час ітерації 1 сек., то швидкість транспортного засобу в моделі $V_m = 1.11$ довжини місця паркування за один крок модельного часу, тобто $V_{m,int} = 1.0$, $V_{m,dec} = 0.11$.

Щоб моделювати рух "нецілої", десяткової складової швидкості V_m, в моделі генерується випадкове число r_j, яке рівномірно розподілено на інтервалі [0,1]. У такому випадку автомобіль просувається на відстань $D = V_{m,int} + 1$ довжин місць паркування в напрямку до місця призначення при виконанні умови $V_{m,dec} > \eta$ або $D = V_{m,int}$ якщо $V_{m,dec} < r_j$, що можна записати у вигляді :

$$D = \begin{cases} V_{m,int} + 1, \text{якщо } V_{m,dec} > \eta \\ V_{m,int}, \text{якщо } V_{m,dec} < \eta \end{cases} \tag{2.2}$$

Вищерозглянута математична модель застосовується до кожного транспортного засобу, рух якого моделюється. Необхідно відзначити, що перед початком подолання інтервалу D перевіряється вільний черговий інтервал чи ні, в останньому випадку рух припиняється. Порядок руху транспортних засобів встановлюється наново, випадково, у кожному циклі роботи алгоритму.

Коли транспортний засіб наближається до перехрестя, водій приймає рішення про те, в якому напрямку йому рухатися, щоб послідовно рухатися у напрямку призначення. У просторово-точній МОП рішення ґрунтується на порівнянні дистанції до місця призначення від поточного перехрестя. У моделі розглядаються місця призначення віддалені лише на 3-5 сегментів вулично-дорожньої мережі.

Всі транспортні засоби розглядаються в моделі на відстані початку пошуку потрібного місця, що відповідає 100 метрам просторово-точної моделі, тобто ця відстань, на якій водій стає обізнаним про необхідність початку пошуку паркування. Місця призначення - це множина доступних точок, що потрапляють у коло радіусу 100 метрів, встановлений для даної моделі. Щоб почати моделювання руху транспортного засобу, одна з точок вибирається за випадковою схемою.

Правила поведінки водія залежить від етапу процесу паркування. У просторово-точній моделі розглядаються чотири основні етапи процесу паркування:

- етап 1: рух у напрямку місця призначення, яке обрано випадковим чином;
- етап 2: оцінка частки незайнятих (вільних) машиномісць;
- етап 3: зупинка у знайденому місці паркування;
- етап 4: залишення місця паркування та залишення середовища моделювання.

Проаналізуємо етапи реалізації запропонованої моделі процесу паркування. На етапі 1 рух до місця призначення виконується згідно з

правилом руху в модельованому середовищі (1). На етапі 2 виконується оцінка частки не зайнятих машиномісць. Другий етап виконується тоді, коли транспортний засіб знаходиться між відстанню пошуку місця для паркування та відстанню прийняття рішення на паркування (у моделі така відстань встановлено відповідно 100 та 50 метрів). На кожному кроці модельного часу оцінюється частка вільних місць для паркування $\Delta P_{freedom}$

$$\Delta P_{freedom} = N_{freedom}/(N_{freedom}+N_{busy}) \quad (2.3)$$

де $N_{freedom}$ – кількість вільних машиномісць, N_{busy} – кількість зайнятих машиномісць.

Починаючи з відстані від початку пошуку місця паркування до відстані ухвалення рішення на паркування водій рухається з оцінкою $\Delta P_{freedom}$.

Наступні два етапи процесу паркування в моделі залежать від накопиченого часу пошуку паркування τ. Якщо час пошуку паркування τ моделі перевищує встановлений τ_{search}, то транспортний засіб паркується на будь-якому машиномісці. У моделі час τ_{search} може встановлюватись в залежності від вимог до часу пошуку вільного машиномісця. Водій паркується з інтервалом часу, який приписується кожному водієві випадково. Після закінчення часу, відведеного на паркування, він зникає з моделі організації паркувань.

На підставі просторово-точної моделі оцінимо паркувальні можливості на головній вулиці Тернополя. Кількісна характеристика району, що вивчається, представлена в таблиці 2.1.

Оцінка загального попиту на паркування на вул. Руській заснована на кількості точок суспільного тяжіння, отриманих на основі польових досліджень.

Таблиця 2.1.

Вхідні параметри для моделювання

Характеристика	Значення
Кількість будинків у центральному районі	1589
Загальна довжина вулиць у центральному районі	32565,051
Кількість вуличних сегментів у центральному районі	177
Кількість машин на основі супутникових знімків у денний час	1249
Кількість будинків	161
Загальна довжина вулиці Руська	2766, 920

На підставі польових досліджень встановлено, що в одній третині будинків є можливість паркуватися поза головною вулицею. Таким чином, максимальна місткість паркінгу на вулиці може бути визначена виходячи з виразу:

$$P_{max} = \frac{2L}{l_p} - l_p k - \frac{n_{\text{буд}}}{3} \quad (2.4)$$

де P_{max} – максимальна місткість паркінгу на головній вулиці;

L – довжина вулиці Руська; l_p - довжина машиномісця для паркінгу;

k – кількість сегментів вулиці;

$n_{\text{буд}}$ - кількість будівель на головній вулиці.

Оцінюючи зазначені параметри на основі просторово-точної моделі, максимальна місткість паркінгу на вул. Руська складає 948,101 машиномісця. Загальне співвідношення попиту та пропозиції становитиме 1249/948,101 = 1,31. Враховуючи, що паркування в одній третині будинків в основному зайняті місцевими жителями і доступні лише в короткий час середини дня, то цими парковками можна знехтувати, тоді співвідношення попиту та пропозиції становитиме 1,4.

З урахуванням подальшого зростання кількості автотранспортних засобів співвідношення попиту та пропозиції на паркування постійно зростатиме.

Вирішення питання про додавання машиномісць до існуючих паркувальних можливостей на головній вулиці та чи покращить це ситуацію з попитом та пропозицією має забезпечити розроблена просторово-точна модель.

У моделі розглянута гіпотетична можливість збільшення паркувальних місць на досліджуваній території шляхом додавання 500 вільно доступних місць для паркування. Було розглянуто два сценарії розподілу місткості місць для паркування по території.

Перший сценарій передбачав 500 машиномісць, розміщені централізовано на досліджуваній території та другий сценарій, який передбачає 4 місця розміщення парковок по 125 машиномісць відповідно.

Результати проведеного моделювання показують, що для сценарію з 4 паркуванням при часі пошуку паркування в 10 хвилин кількість водіїв, час пошуку яких перевищує 10 хвилинний бар’єр, становить 280-320. У разі централізованої організації паркінгу кількість водіїв, чий час пошуку перевищує 10 хвилинний бар'єр, коливається між 400 і 450.

Розроблена просторово-точна модель дозволяє визначати вплив різних просторових сценаріїв на організацію парковок біля великого міста. Можна припустити, що зменшення часу пошуку паркувань позитивно вплине як на якість проживання мешканців центральних районів, так і на якість навколишнього середовища, що виражається у зменшенні забруднення повітря, скупчення автотранспорту, засноване на машинах, які знаходяться у виборі машиномісця для паркування.

2.2. Метод динамічного управління навігацією

Для підвищення ефективності використання паркувальних площ за рахунок скорочення кількості автомобілів, що перебувають у русі, в рамках дослідження використовується метод динамічного управління навігацією на відкритих паркувальних територіях. Незважаючи на широкий розвиток та використання картографічних сервісів, було прийнято рішення використовувати зображення з реальних потокових камер відеоспостереження. Метод полягає в наданні в реальному часі автовласникам, що в'їжджають на територію паркування, інформації про зайнятість паркувальних місць на основі обробки потокового відео з камер відеоспостереження, встановлених над територією паркінгу. Відмінністю запропонованого методу від застосовуваних в даний час є можливість його використання на відкритих парковках, які, як правило, не оснащуються спеціальним сигналізуючим обладнанням, детекторами зайнятості та контрольно-пропускними системами, значної площі.

Основні кроки методу динамічного керування навігацією:

1) побудова інформаційно-графічної моделі (ІГМ) відкритого паркування;

2) актуалізація даних ІГМ відкритого паркування;

3) візуалізація ІГМ відкритого паркування;

4) інтерактивна взаємодія та обробка запитів користувачів у ході навігації.

Структурно-функціональна модель пропонованого VDA-підходу в рамках методу динамічного управління навігацією зображено на рисунку 2.2.

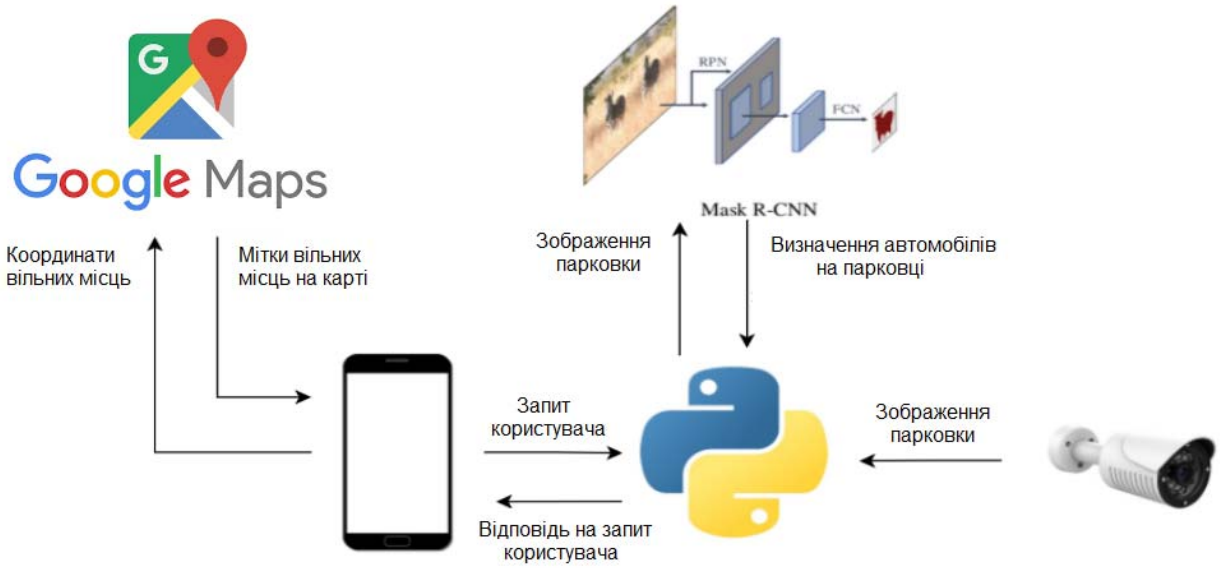

Рис. 2.2. Структурно-функціональна модель VDA

Алгоритм процесу додавання нового місця моніторингу. Складальник зображень (рисунок 2.3) виконує збір медіафайлів з камер, зареєстрованих у системі. Це можуть бути як потокові камери, так і пристрої користувача. У разі потокових камер модуль з певним часовим інтервалом підключається до зареєстрованих пристроїв і робить знімок поточного стану місця для паркування, потім відправляє його в систему зберігання.

Основна послідовність збирання зображень:

1. Складальник зображень опитує камери зі списку доступних.

2. Складальник зображень отримує зображення.

3. Складальник зображень надсилає зображення до файлового сховища.

Альтернативна послідовність збирання зображень:

1. Складальник зображень опитує камери зі списку доступних.

2. Камери недоступні.

У разі користування пристроєм, користувачеві необхідно буде зареєструвати камеру в системі. Для реєстрації необхідно в мобільному додатку заповнити форму із зазначенням адреси, розташування та додаткових коментарів. Потім встановити на свій комп'ютер сервіс,

дистрибутив якого буде згенеровано після заповнення форми. Після встановлення сервіс підключиться до пристрою введення відео, встановленого в системі за промовчанням, і надішле метадані, зібрані при заповненні форми, а також IP-адреса пристрою для реєстрації в системі. Після всіх цих дій пристрій буде надсилати знімки з камери в модуль складання зображень.

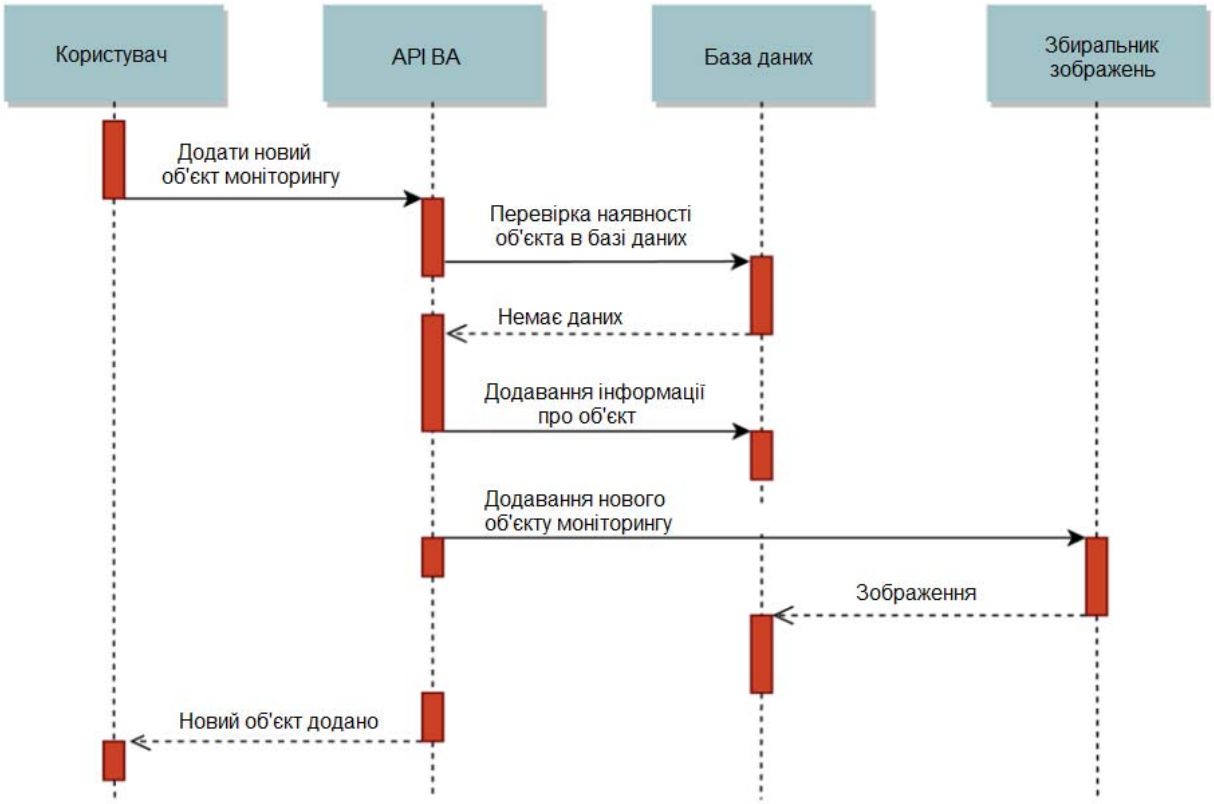

Рис. 2.3. Процес додавання нового об’єкту моніторингу

Основна послідовність додавання нового об'єкта:

1. Користувач заповнює форму та надсилає запит на додавання нового об'єкта моніторингу.

2. Запускає на комп'ютері згенерований скрипт.

3. Скрипт виявляє пристрій і додає планувальник для регулярного надсилання зображень до системи.

4. При першому запуску скрипта збирач зображень додає пристрій до списку доступних пристроїв.

5. Пристрій додано.

Альтернативна послідовність додавання нового об'єкта:

1. Користувач заповнює форму.

2. Мобільний додаток видає попередження про те, що дане місцезнаходження вже існує в системі.

Алгоритм процесу запиту місця для паркування. Деталізація процесу відправки запиту для перевірки паркувального місця зображена на рисунку 2.4.

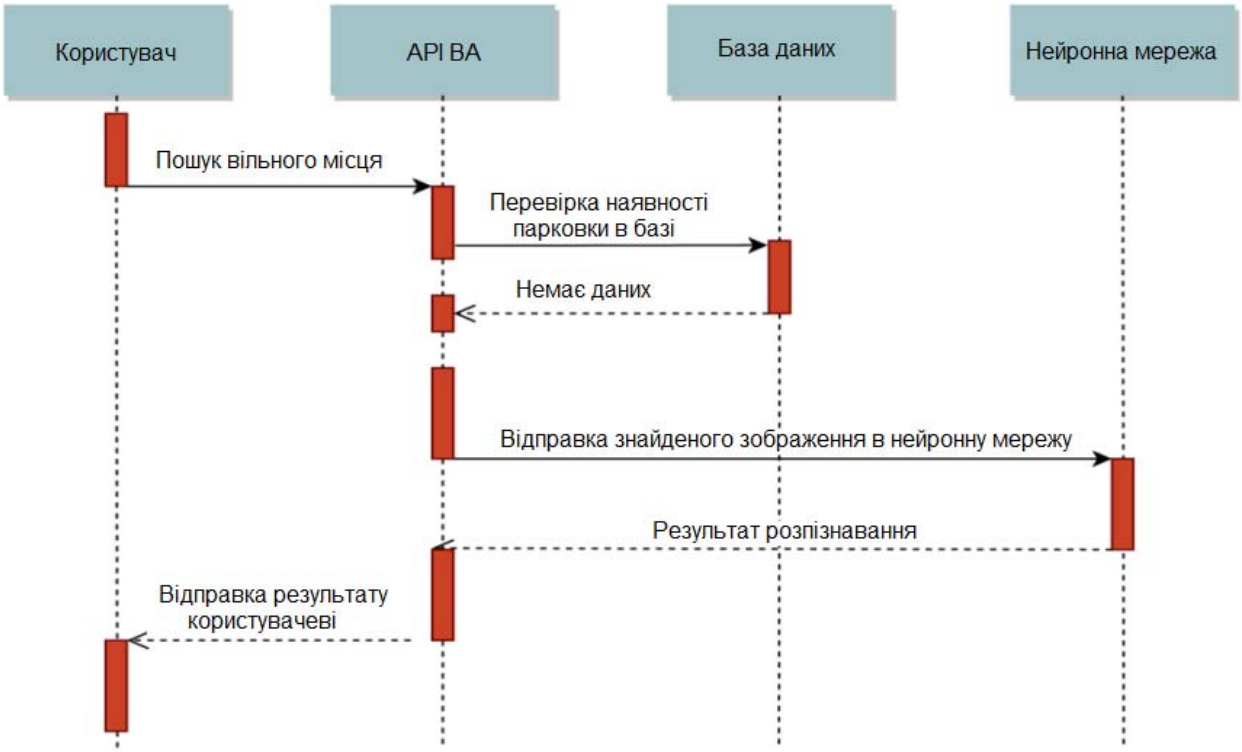

Рис. 2.4. Процес запиту паркувального місця

Користувач у мобільному додатку надсилає запит на перевірку наявності вільних парковок за заданою адресою. Звернення приймає API BA та перевіряє наявність даної адреси в системі зберігання медіафайлів. Якщо така адреса є, то зображення з камери за цією адресою відправляється в модуль обробки зображень і відбувається пошук вільних місць для паркування. Якщо заданої адреси немає в системі зберігання, з'явиться відповідне повідомлення про те, що немає даних за заданою адресою.

Основна послідовність запиту паркувальних місць:

1. Користувач надсилає запит для перевірки вільного місця для паркування.

2. Система надає інформацію про наявність вільних місць для паркування.

Альтернативна послідовність запиту паркувальних місць:

1. Користувач надсилає запит для перевірки вільного місця для паркування.

2. Система повідомляє про те, що немає даних щодо зазначеного місця розташування.

Основна послідовність визначення вільних місць:

1. API віртуального асистента надсилає запит для перевірки вільних паркувальних місць за даним місцезнаходженням.

2. У нейронну мережу розпізнавання передається посилання файл зображення.

3. Модуль розпізнавання отримує зображення та шукає вільні місця для паркування. 4. Інформація про наявність вільних місць для паркування відправляється в API ВА.

2.3. Концепція та сценарії використання VDA

Концепція системи дозволяє організувати сервер, де будуть присутні модулі обробника зображень, збирача зображень та віртуального асистенту, реалізованих мовою Python та з використанням його технологій машинного навчання (рисунок 2.5).

Мобільний додаток буде реалізовано з використанням технологій Flutter мовою програмування Dart. MySQL буде обрана як база даних, оскільки системі важлива швидкість і більша частина запитів буде

направлена на читання. Файлове сховище буде загальним мережевим ресурсом, розміщеним на твердотільних накопичувачах SSD.

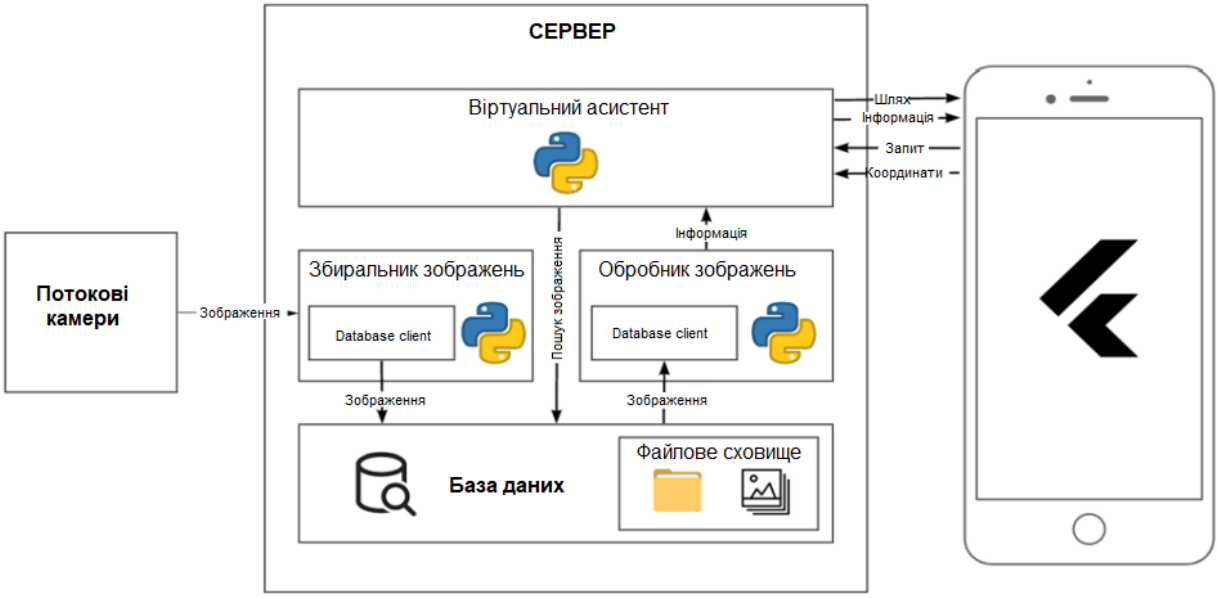

Рис. 2.5. Концептуальна модель системи з мобільним додатком

Для підвищення ефективності роботи програми використовуватимуться відеокамери. Для визначення вільних місць порівнюватимуть еталонні зображення території паркування, зроблені в редакторі контурів, та поточний стан паркувальної зони, взятий з камер.

Взаємодія між компонентами системи реалізується через API віртуального помічника. Система розділена на 4 основні компоненти:

1. Віртуальний помічник.
2. Обробник зображень.
3. Складальник зображень.
4. Мобільний додаток.

Віртуальний помічник забезпечує взаємодію з юзером мобільного додатка. У його функції входить обробка та валідація запиту, а також надсилання відповіді користувачеві.

Складальник зображень у певні проміжки часу опитує доступні камери, список яких знаходиться в БД, і завантажує знімки у файлове сховище. Крім цього, дозволяє додати нові камери до списку доступних камер.

Обробник зображень, отримавши від API віртуального помічника посилання на зображення перевіряє наявність вільних місць для паркування на зображенні і відправляє отриману інформацію назад.

За допомогою мобільного додатка користувач звертається до системи для пошуку вільного місця для паркування за заданою адресою. Крім цього, якщо користувач хоче додати нову камеру для моніторингу свого паркувального місця, він може заповнити форму і отримати скрипт, який допоможе підключити його пристрій до системи. У свою чергу, скрипт звернеться до модуля збирача зображень, що дозволить додати пристрій до БД до списку доступних камер.

Для актуалізації поточного стану паркувальних зон необхідно обробляти зображення з камер потокового відео. На зображеннях необхідно знаходити автомобілі для вирішення цього завдання відео необхідно визначати автомобілі на зображенні. Це завдання вирішує нейромережа Mask R-CNN.

Mask R-CNN – покращення алгоритму Faster R-CNN, що забезпечує здійснення можливості сегментації об'єктів. На вхід нейромережі Mask R-CNN подавали зображення паркувань і код шуканих об'єктів (у разі автомобілів). В результаті роботи алгоритму можна було отримати кількість знайдених автомобілів, а також їх піксельні координати та координати вільних місць.

Висновки до другого розділу

1. Розроблена просторово-точна модель, яка дозволяє визначати вплив різних просторових сценаріїв на організацію паркувань в межах великого міста. На відміну від відомих, традиційних моделей цей підхід імітує дії кожного водія у просторово-точному міському середовищі, що значно підвищує значущість результатів моделювання.

2. Спираючись на нестачу реалізації додатків з фіксованою розміткою, було прийнято рішення використати підхід із сегментацією об'єктів на зображенні, щоб можна було визначати місця на паркування відштовхуючись від розташування автомобілів на зображенні.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПАРКУВАННЯ АВТОМОБІЛІВ

3.1. Проектування системи

При проектуванні системи було прийнято рішення використати нейронну мережу Mask R-CNN для сегментації автомобілів на зображенні. Mask R-CNN дозволяє досить точно та ефективно знаходити об'єкти на зображенні. Для створення Mask R-CNN було вирішено використати відкриту реалізацію Matterport, яка досить проста у використанні та має докладну інструкцію та приклади щодо застосування даної збірки на власному наборі даних.

Модуль додавання паркування до системи. Для визначення доступних місць на парковці необхідно встановити їхнє місцезнаходження на зображенні. Для цього оператор системи повинен намалювати розмітку паркування на зображенні. Вимоги до розмітки паркування:

1. Потрібно забезпечити кадр із повністю коректно заповненою зоною паркуванням і дати йому унікальну назву, яка надалі використовуватиметься в системі як ідентифікатор паркування.

2. Розмір кадру в пікселях, на якому будуватиметься розмітка, повинен збігатися з розміром кадру, який завантажуватиметься в систему при детектуванні вільних місць на парковці.

3. На зображенні в першу чергу необхідно виділити паркувальні ряди, у разі наявності перешкоди в ряду паркування (стовпа, проїзду, виходу з двору і т.д.) потрібно розбити ряд паркування і дати кожному назву виду row [індекс ряду]. Розмітку паркувального ряду слід починати малювати зліва-направо або зверху-вниз у випадках горизонтального або вертикального ряду відповідно, попередньо намалювавши стартовий відрізок ряду.

4. Після закінчення процесу розмітки рядів паркування, необхідно намалювати маску паркування, яка охоплюватиме всі ряди паркування, та позначити її як mask.

5. Файл з усіма координатами потрібно зберегти в форматі JSON і помістити разом із кадром паркування в окрему папку.

6. Як інструмент з розмітки паркування слід використовувати Програму VGG Image Annotator.

Приклад розміченого місця для паркування представлено на рисунку 3.1.

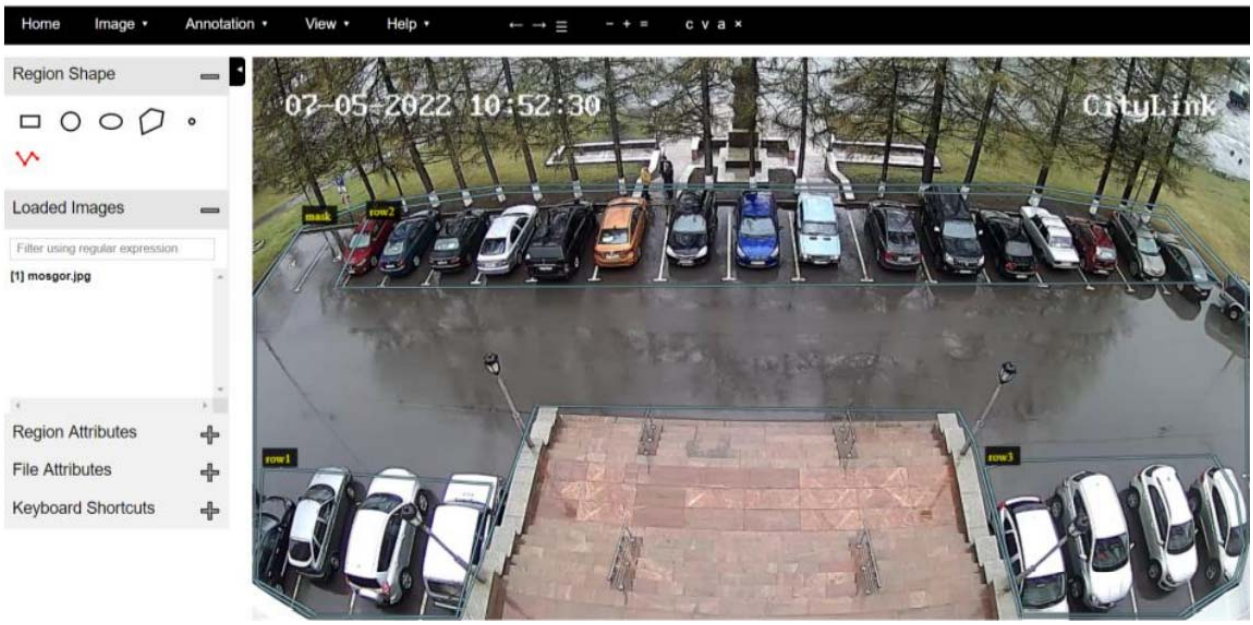

Рис. 3.1. Приклад розміченого місця для паркування

Після завантаження даних у модуль, буде розраховуватися детальна розмітка паркування, необхідна для визначення наявності вільних місць на ній.

Модуль детектування вільних місць на парковках. Даний модуль буде зчитувати отримані дані про паркування, і детальну розмітку з JSON файлу, отриману при додаванні паркування в систему. Далі будуть визначати вільні місця для паркування, і візуалізувати їх на зображенні (рисунок 3.2).

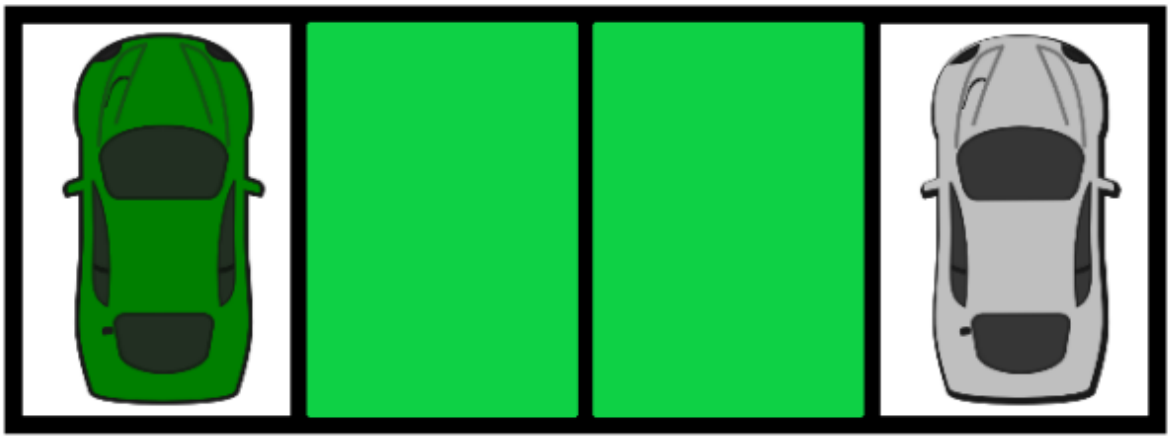

Рис. 3.2. Приклад роботи системи

Вимоги до системи. Для вирішення задачі визначення наявності вільних місць на парковках по фото та відео необхідно створити систему, яка повинна вміти детектувати вільні місця по завантаженому кадру.

Функціональні вимоги до проектованої системи. Система, що розробляється, повинна задовольняти наступним функціональним вимогам:

1. Система повинна вміти детектувати вільні місця на парковці, за переданими даними (зображення або відео).

2. Система повинна мати можливість завантаження нового паркування. Нефункціональні вимоги до проектованої системи. Додаток, що розробляється, повинен відповідати наступним нефункціональним вимогам:

1. Система має бути написана мовою Python.

2. Система може використовувати відкриту програмну бібліотеку для створення нейронних мереж Tensorflow.

Варіанти використання системи. Для проектування системи була використана мова графічного опису для об'єктного моделювання UML. Була побудована модель взаємодії зовнішніх акторів із програмною системою у вигляді діаграми варіантів використання (use-case diagram). Отримана діаграма зображена на рисунку 3.3.

Основні актори, що взаємодіють із системою. Інший додаток використовує систему для детектування вільних місць на парковках, які є в системі.

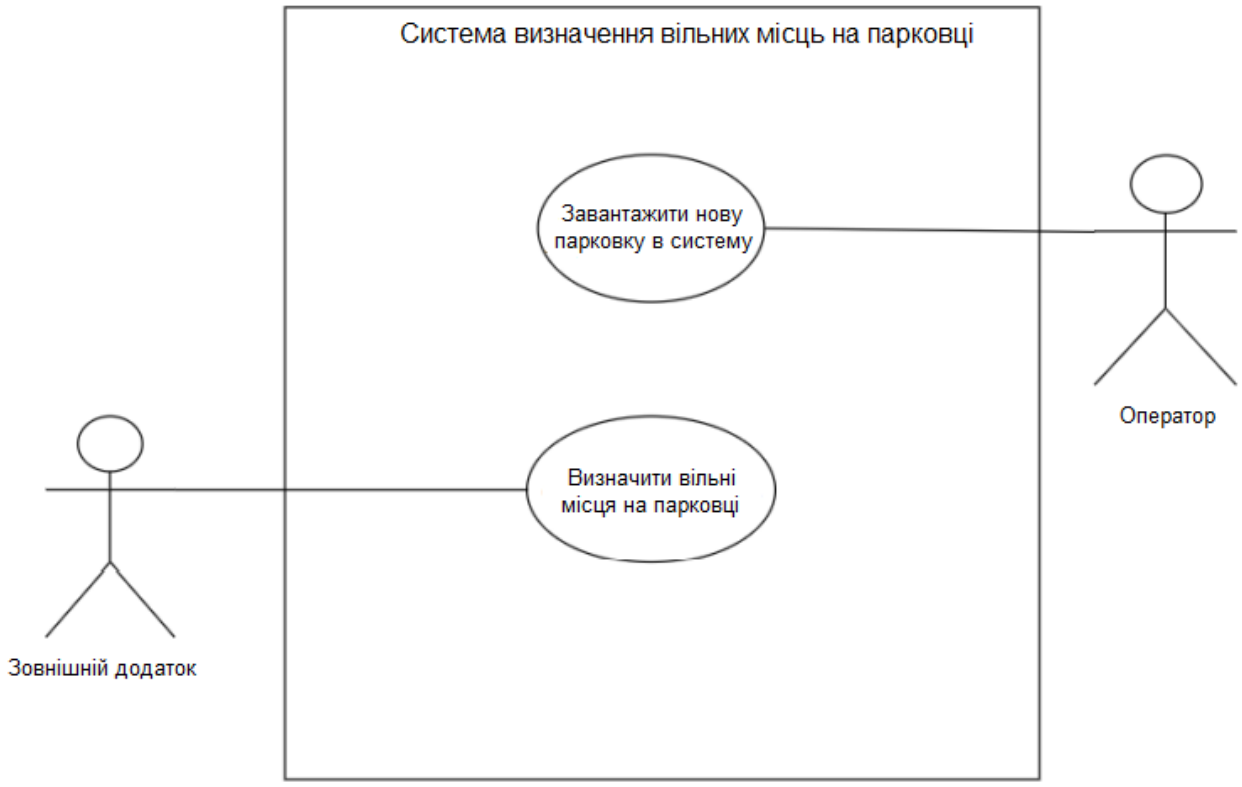

Рис. 3.3. Діаграма варіантів використання

Оператор використовує систему для додавання нових паркувальних місць.

Короткий опис варіантів використання. Інший додаток може:

-Визначити вільні місця на парковках по фото або відео.

Оператор може:

-завантажити нове паркування в систему.

Специфікація основних варіантів використання. У таблиці 3.1 наведено опис варіанта використання «Визначити вільні місця на паркуванні».

Таблиця 3.1.

"Визначити вільні місця"

UseCase: Визначити вільні місця на парковці
ID: 1
Анотація: Сторонній додаток надсилає запит на визначення вільних місць на парковках
Головні актори: Сторонній додаток
Інші актори: Ні
Передумови: система «знає» потрібне паркування
Основний потік: 1. Варіант використання починається, коли сторонній додаток викликає функцію пошуку місця на паркуванні, визначаючи необхідне паркування в системі та зображення (відео) для розпізнавання. 2. Здійснюється завантаження даних у систему. 3. Відбувається передобробка даних. 4. Відбувається визначення вільних місць для паркування. 5. Здійснюється вивід результату розпізнавання, де вільні місця виділено.
Постумова: Результат відображено
Альтернативні потоки: У разі некоректних даних відбувається вихід

У таблиці 3.2 наведено опис варіанту використання «Завантажити нове паркування в систему».

Таблиця 3.2.

" Завантажити нове паркування в систему "

UseCase: Завантажити нове паркування в систему
ID: 2
Анотація: Оператор завантажує нові дані про паркування
Головні актори: Оператор
Інші актори: Ні
Передумови: Ні
Основний потік: 1. Варіант використання починається, коли оператор викликає функцію додавання паркування, визначаючи зображення з паркуванням та її розміткою в формат JSON файлу. 2. Здійснюється завантаження даних у систему. 3. Відбувається передобробка даних. 4. Відбувається створення детальної розмітки паркування. 5. Здійснюється збереження розмітки паркування.
Постумова: Детальна розмітка паркування збережена.
Альтернативні потоки: У разі некоректних даних відбувається вихід

Для того щоб продемонструвати особливості фізичної побудови системи, була побудована діаграма компонентів (рисунок 3.4).

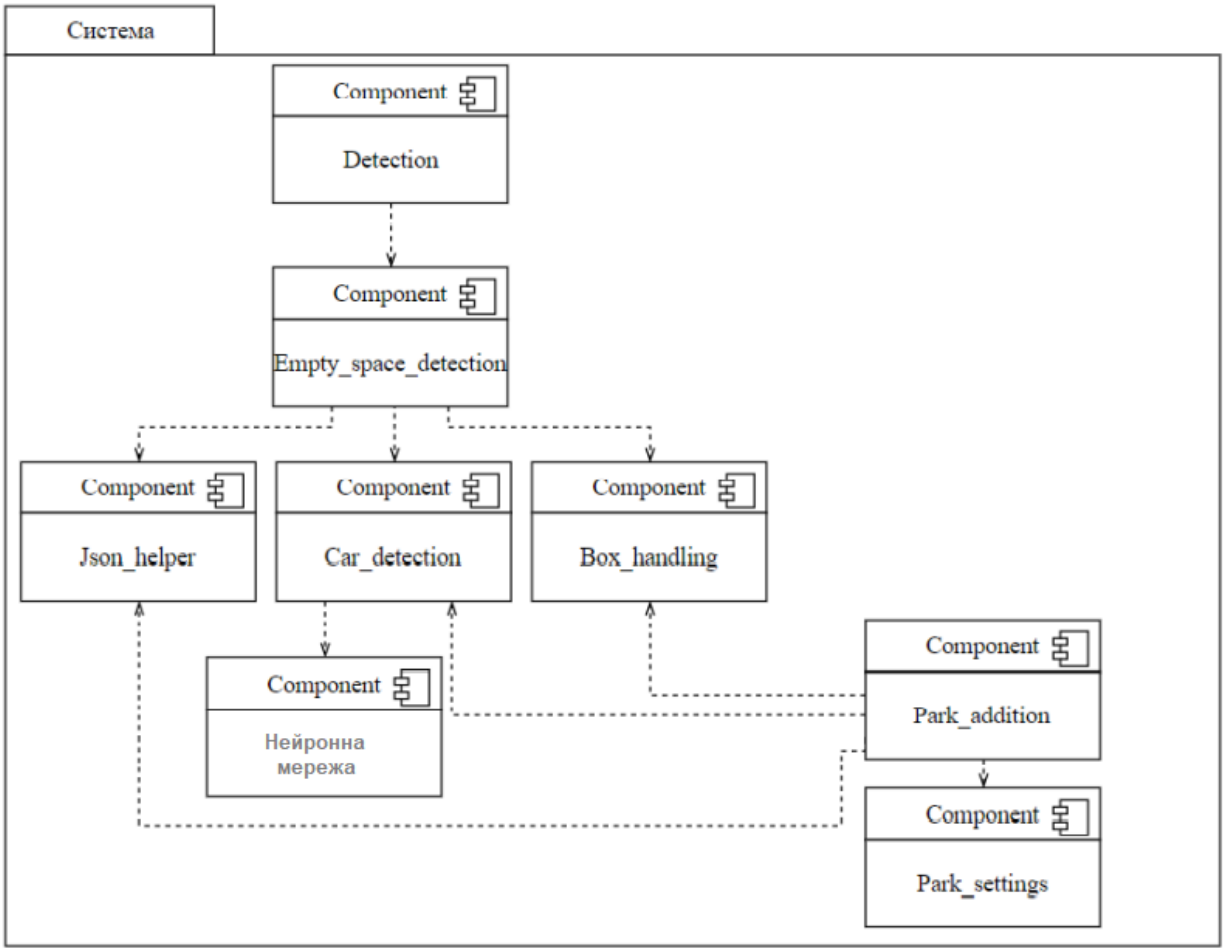

Рис. 3.4. Діаграма компонентів

Система складається з наступних компонентів:

1. Detection – головний компонент системи, який включає у собі всі методи, необхідні для обробки вхідних даних, їх подальшого розпізнавання та візуалізації результату.

2. Empty space detection – компонент, який відповідає за детектування вільних місць для паркування.

3. Json helper – компонент, який служить для завантаження та вивантаження даних про паркування з файлу JSON.

4. Car detection – компонент, який служить для детектування автомобілів на парковках.

5. Нейронна мережа – компонент, що представляє собою модель навченої штучної нейронної мережі, що використовується для класифікації зображення.

6. Box handling – компонент, в якому містяться всі необхідні методи для попередньої обробки розпізнаних автомобілів, після якої можна визначити вільні місця для паркування.

7. Park addition – компонент, який включає в себе методи додавання нового паркування.

8. Park settings – компонент, який служить для завантаження даних про паркування.

Діаграма діяльності системи, що описує процес визначення вільного місця на парковці представлено на рисунку 3.5

Рис. 3.5. Діаграма діяльності системи

Діаграма діяльності системи, що описує процес додавання нового паркування в систему представлено на рисунку 3.6.

Рис. 3.6. Діаграма діяльності додавання нової парковки

3.2. Програмна реалізація системи

Для розробки програмної частини був використаний високорівнева мова Python версії 3.6.5. Розробка проводилася у середовищі розробки Py Charm Community та на операційній системі Windows.

Налаштування та навчання нейронної мережі. Для навчання мережі Mask R-CNN необхідно підготувати великий набір анотованих зображень з різних парковок, у тому числі з паркувань на яких проводитиметься розпізнавання. На кожному зображенні необхідно намалювати маску (контур) кожного автомобіля. Як інструмент з розмітки зображень було обрано програму VGG Image Annotator. Ця програма проста у використанні та легко взаємодіє із платформою Matterport. За допомогою інструменту «polyline region shape» можна намалювати маски кожного об'єкта на зображення, які потрібно буде розпізнавати. Приклад роботи програми представлений на рисунку 3.7.

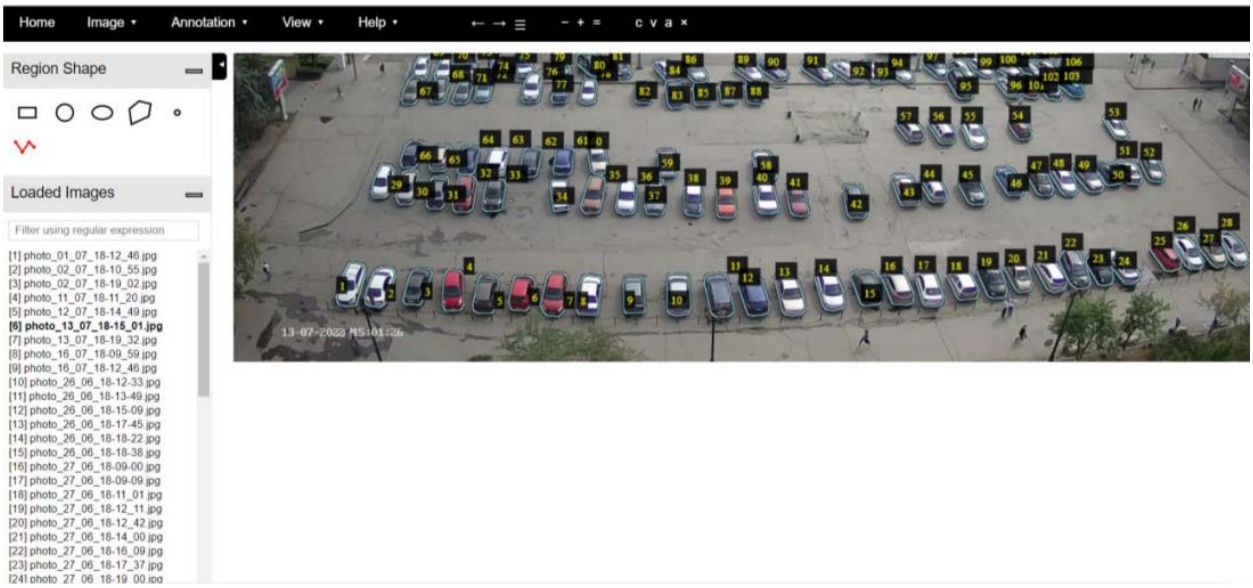

Рис. 3.7. VGG Annotator

Таким чином, було розмічено близько 300 зображень. Далі необхідно поділити набір даних на набір для навчання (training set) набір для перевірки (validation set), 70 відсотків на training та 30 відсотків на validation. Кожен набір повинен включати json файл (via_region_data.json), в якому знаходяться координати масок кожного автомобіля на зображенні.

Налаштування Mask R-CNN та навчання. Необхідно встановити репозиторій «Matterport Mask R-CNN'pepository». У даному репозиторії міститься файл, з реалізацією завантаження датасета з повітряними кульками

та навчанням нейронної мережі. Потрібно не багато видозмінити цей код для завантаження набору даних з автомобілями. Лістинг представлений на рисунку 3.8.

```
def train(model):
    """Train the model."""
    # Training dataset.
    dataset_train = carDataset()
    dataset_train.load_car(args.dataset, "train")
    dataset_train.prepare()

    # Validation dataset
    dataset_val = carDataset()
    dataset_val.load_car(args.dataset, "val")
    dataset_val.prepare()

    # *** This training schedule is an example. Update to your needs ***
    # Since we're using a very small dataset, and starting from
    # COCO trained weights, we don't need to train too long. Also,
    # no need to train all layers, just the heads should do it.
    print("Training network heads")
    model.train(dataset_train, dataset_val,
                learning_rate=config.LEARNING_RATE,
                epochs=70,
                layers='heads')
```

Рис. 3.8. Лістинг навчання нейронної мережі

Далі необхідно зробити налаштування конфігурації Mask R-CNN. Оскільки максимальна кількість розпізнаних об'єктів на зображеннях з паркування може досягати значення 300, необхідно змінити параметр «DETECTION_MAX_INSTANCES», надавши дане максимальне значення. Як «backbone» було прийнято рішення використати нейронну мережу resnet50. Реалізація навчання представлена на рисунк 3.9.

Модуль з додавання нового паркування до системи. Відбувається завантаження кадру та розмітки паркування, зробленого оператором, у систему. Компонент "park_addition" використовуючи функцію "add_park" виконує передобробку отриманих даних та заносить дані про паркування в систему. Реалізація add_park наведена на рисунку 3.10.

```
def add_park(path_):
    path_to_image=path_+'/*.jpg'
    im_path = next(iter(glob.glob(path_to_image)))
    base = os.path.basename(im_path)
    park_id=os.path.splitext(base)[0]
    path_to_json=path_+'/via_region_data.json'
    image= plt.imread(im_path)
    rows,mask=jhelper.load_park_info(path_to_json)
    car_boxes=car_detect.detect_cars(image,mask)
    s_boxes=box_handle.sort_boxes(car_boxes,rows)
    sorted_points,sides,distances,intersects=park_set.define_park_settings(
    s_boxes)
    jhelper.save_park_settings(park_id,sorted_points,
    sides,distances,intersects,rows,mask)
```

Рис. 3.9. Лістинг додавання нової парковки

Першим етапом відбувається обробка отриманих даних, потім береться маска паркування з JSON файлу і саме зображення. Маска накладається на кадр з паркування і запускається компонент розпізнавання автомобілів на зображенні "car_detection". Приклад розпізнавання подано на рисунку 3.9.

Далі необхідно визначити машини, які розташовані лише в рядах паркування. Для цього необхідно визначити середину кожної рамки автомобіля за формулою 3.1:

$$x, y = \left(\frac{x_2 - x_1}{2}\right)\left(\frac{y_2 - y_1}{2}\right) \tag{3.1}$$

Потім створимо об'єкти класу Polygon і Point із пакету Shapely. У конструктор класу Polygon передамо отримані ряди паркування, в конструктор класу Point, отримані серединні точки рамок автомобілів.

Визначимо наявність точки у полігоні за допомогою методу класу Polygon contains, у параметрах якого передамо точку. Таким чином, отримаємо тільки ті автомобілі, які знаходяться всередині ряду паркування.

Відсортуємо за зростанням отримані серединні точки в кожному ряду паркування. Результат подано на рисунку 3.10.

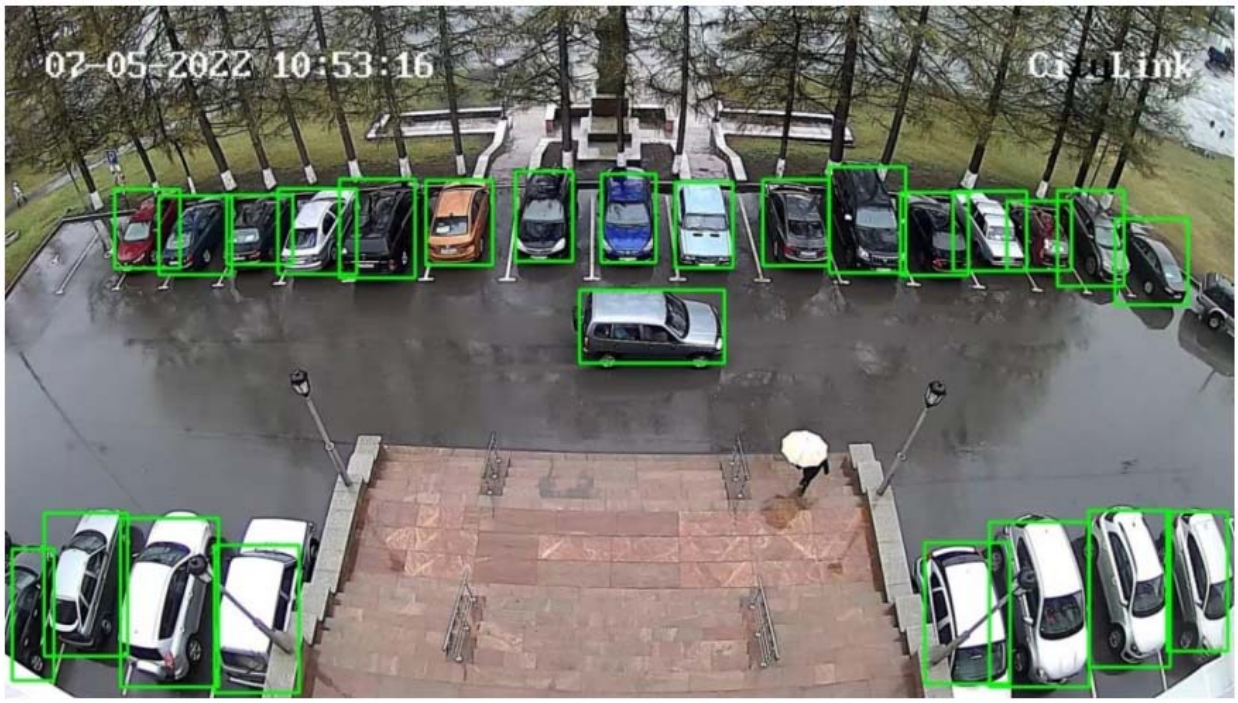

Рис. 3.10. Детектування автомобілів

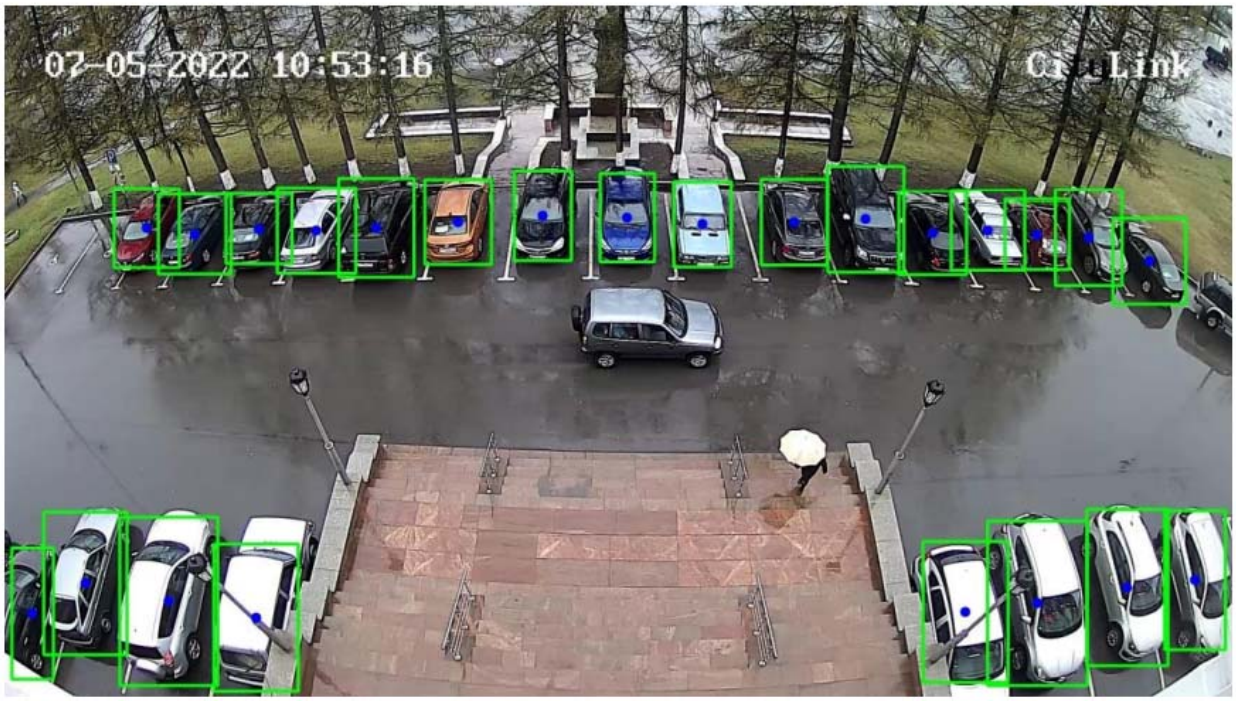

Рис. 3.11. Автомобілі в рядах паркування

Щоб встановити горизонтальний або вертикальний ряд паркування, необхідно знайти яку зі сторін прямокутної рамки автомобіля пересікає

відрізок, що з'єднує серединну точку першого і наступного автомобіля (рисунок 3.12).

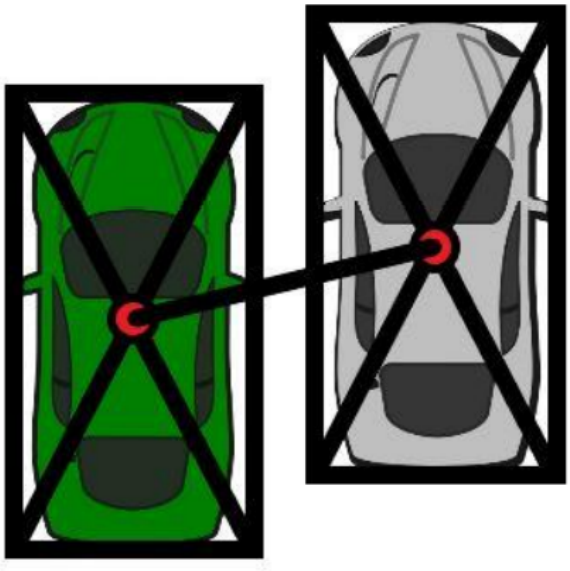

Рис. 3.12. Визначення перетнутої сторони

Розіб’ємо прямокутну рамку на чотири сторони, пройдемося по всім сторонам і визначимо за допомогою методу intersects класу LineString з пакета Shapely сторону, яку перетинає відрізок, що з'єднує серединні точки. Таким чином, якщо x координата першої та другої вершини пересіченої сторони збігаються, то відстань буде братися по x координаті, інакше по y координаті.

Далі кожному відрізку, що з'єднує центри двох сусідніх рамок автомобілів, зіставимо необхідну відстань для вільного місця. Для визначення необхідної відстані для вільного i-го місця, за пам'ятаємо дистанцію між x координатою правого краю рамки i -1 автомобіля x координати лівого краю $i + 1$ для горизонтального ряду. У разі вертикального ряду зафіксуємо дистанцію між y координатою верхнього краю рамки $i - 1$ автомобіля та у координатою нижнього краю $i + 1$.

Також необхідно визначити випадки накладання однієї рамки на іншу, таким чином врахувати ширину рамки. Для цього потрібно запам'ятати відстань між $x(y)$ координатою правого (верхнього) краю рамки і автомобіля і $x(y)$ координатою лівого (нижнього) краю $i+1$ для горизонтального (вертикального) ряду. Візуалізація роботи алгоритму представлена на рисунку 3.13.

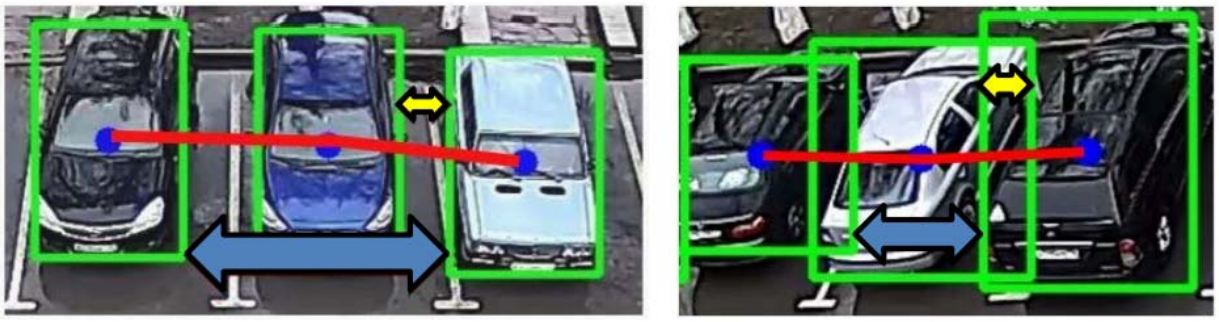

Рис. 3.13. Визначення перетнутої сторони

Червоними лініями відображені відрізки, що з'єднують центри рамок автомобілів (блакитні точки), синьою стрілкою показано необхідну відстань для вільного місця, жовтою відображаються випадки перетину рамок (ширина рамки автомобіля для даного вільного місця).

Збережемо ідентифікатор паркування, серединні точки кожного автомобіля, відстань для вільного місця, бік перетину з відрізком, ступеня накладання, паркувальні ряди та маску в JSON файл для подальшого використання.

Модуль детектування вільних місць на парковках. Завантаження паркування в систему стороннім додатком. Після обробки отриманих даних відбувається пошук вільних місць на парковці та візуалізація результату.

Першим етапом відбувається завантаження та обробка отриманих даних. Інший додаток передає ідентифікатор і шлях до зображення або відео паркування, яке необхідно розпізнати. У разі передачі відео сторонній додаток додатково надає третьому параметру в функції "isVideo" значення

дорівнює "True", яке сигналізує про те, що було передано відео, та його необхідно обробити іншим способом.

Далі ідентифікатор та кадр з паркування передаються функції detect_empty_box з компонента empty_space_detection. Ця функція, використовуючи переданий ідентифікатор, отримує необхідну інформацію про паркування (детальну розмітку). Накладається маска на паркування і запускається компонент розпізнавання автомобілів на зображенні car_detection.

Потім визначаються ті автомобілі, які розташовані в рядах паркування та відсортуються в кожному ряду за зростанням. Визначивши координати кожної рамки автомобіля на зображенні, необхідно пройтися рядами паркування і встановити, чи є потрібна відстань для вільного місця в паркувальному ряду.

Встановивши стартову точку рівної середині відрізка початку ряду, будемо проходити за сортованим списком з рамок автомобілів, що знаходяться в ряду.

Визначимо, чи є необхідна відстань між стартовою точкою та початковою гранню обмежуючої рамки першого елемента у списку. Відстань визначатиметься за допомогою відрізків, що з'єднують центри рамок автомобілів, отриманих з детальної розмітки паркування. Встановимо, в який з відрізків потрапила стартова точка, і візьмемо відповідну відстань для нього та випадки перетину рамок (ширина рамки для даного від різання).

Якщо дана відстань є, занесемо координати вільного місця до списку з вільних місць. Нову стартову точку визначимо використовуючи значення перетину, зіставленого відрізку. При від'ємному значенні перетину, нова стартова точка визначатиметься як старе значення плюс відстань необхідна для вільного місця. При позитивному значені визначатиметься як старе значення плюс відстань та значення перетину.

Якщо відстані немає, нова стартова точка буде визначатися як значення крайньої грані рамки поточного автомобіля. Візуалізація роботи алгоритму представлена на рисунку 3.14.

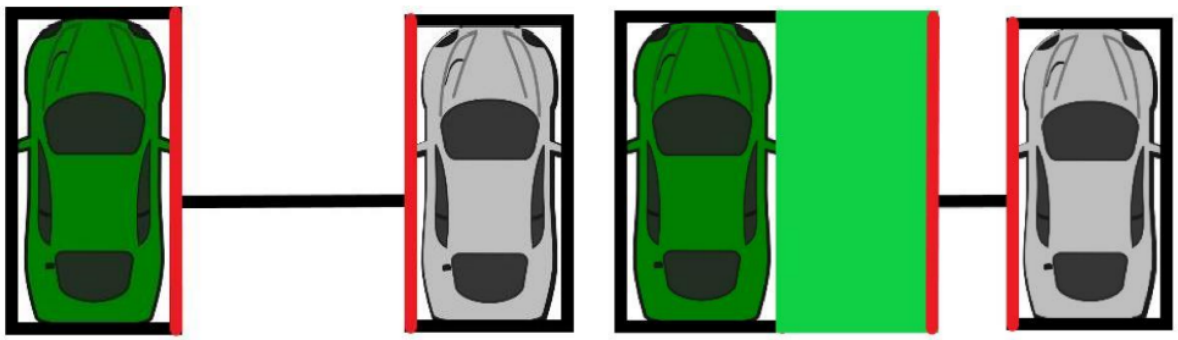

Рис. 3.14. Візуалізація пошуку вільних місць

Якщо при проходженні по всіх рамках автомобілів у рядку паркування, кінець ряду не досягнуто (кінець ряду визначається як крайня грань рамки останнього автомобіля в ряду паркування, отриманого з детальної розмітки), необхідно пройти до його закінчення.

Коли значення стартової точки стає більше або дорівнює кінцевій (значення кінця ряду), то алгоритм завершує свою роботу. Таким чином, пройшовши по всіх рядах паркування, визначимо список з вільних місць. Візуалізуємо отримані рамки кожного порожнього місця на зображенні з паркування, використовуючи засоби OpenCV.

3.3. Експериментальні дослідження та тестування нейронної мережі

Було проведено тестування нейронної мережі на 20 різних зображеннях з паркування. У ході тестування було встановлено, що нейронною мережею коректно детектуються близько 84 відсотків автомобілів. Помилки в

основному виникають у випадках, якщо машина не повністю видно на зображення (більша частина машини знаходиться поза межами зображення або одна з машин закриває іншу). Цю проблему можна вирішити, встановивши камеру на парковці вище, таким чином збільшити охоплення паркування. Приклад розпізнавання представлений на рисунку 3.15.

Рис. 3.15. Результат розпізнавання машин

Тестування детектування вільних місць. Проведено тестування модуля детектування вільних місць. У ході проведеного тестування було встановлено, що вільні місця визначаються коректно у 80 відсотків випадків. Помилки в основному пов'язані з місцями, розташованими по краях зображення. Можливим рішенням може стати, як і в минулому пункті, збільшення охоплення паркування, шляхом встановлення камери вище. Приклад розпізнаних вільних місць представлений на рисунку 3.16.

Рис. 3.16. Результат розпізнавання вільного місця

3.4. Програмна реалізація клієнтського додатку для пошуку вільних місць на парковках

Для зручності роботи користувачів із системою було реалізовано мобільний додаток. На рисунку 3.17 представлено копію головного вікна мобільного додатку, яка містить інформацію про вибір міста та зареєстровані відповідні парковки.

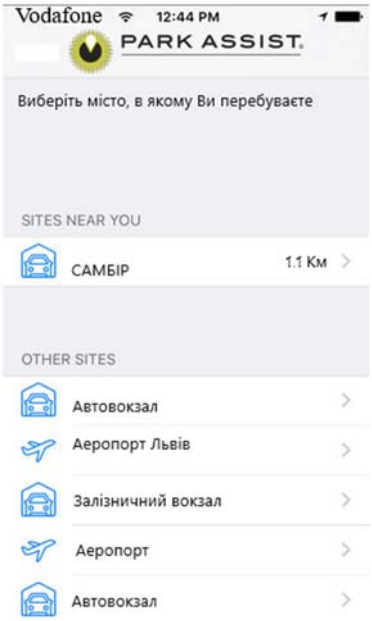

Рис. 3.17. Копія форми для вибору міста та парковки

На рисунку 3.18 наведено форму для авторизації в системі пошуку парковок та відповідно вільних місць.

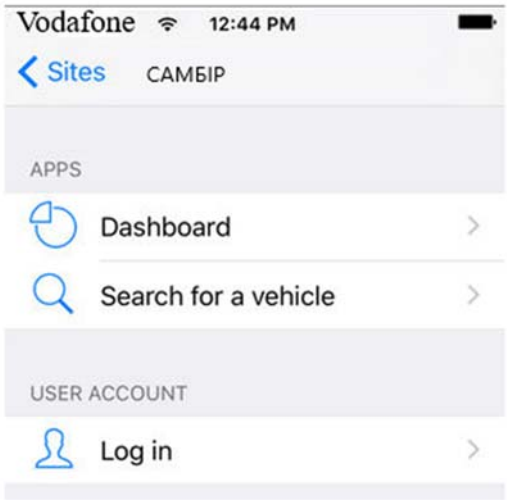

Рис. 3.18. Копія форми авторизації в системі

На рисунку 3.19 представлено копію форми із списком зареєстрованих парковок в системі для вибраного міста із загальною завантаженістю парковок.

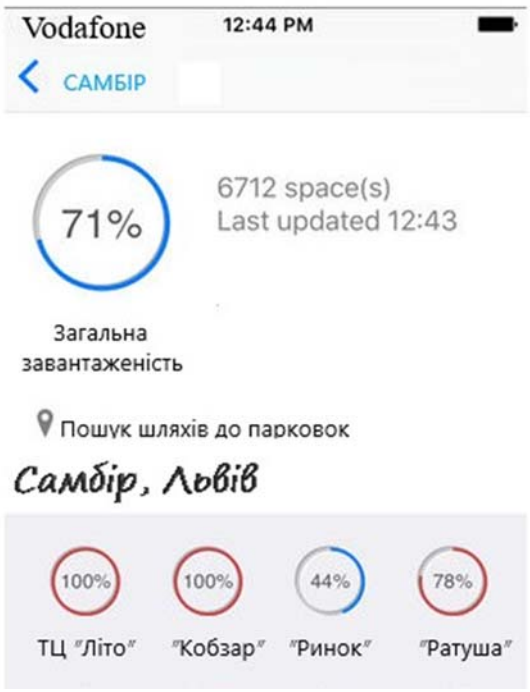

Рис. 3.19. Копія форми із інформацію про парковки для вибраного міста

На рисунку 3.20. представлено копію форми із пошуком вільних місць на відповідних парковках.

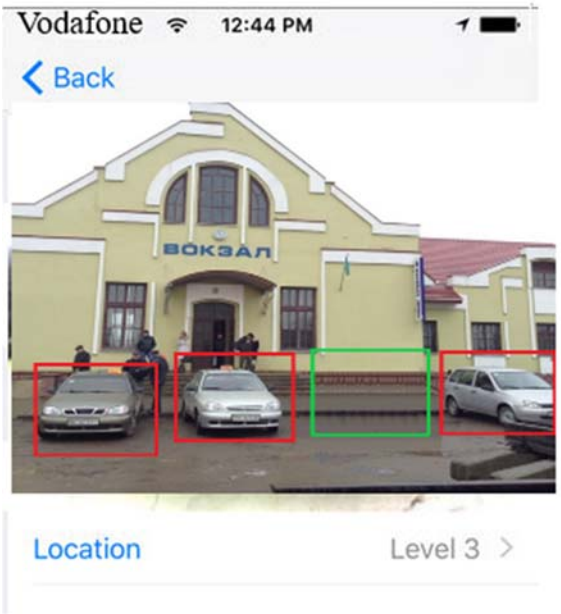

Рис. 3.20. Копія форми із інформацію про наявність вільних місць

При практичному використанні додаток отримав багато скхвальних відгуків від клієнтів.

Висновки до третього розділу

1. Було обрано архітектуру нейронної мережі. Були спроектовані модуль додавання паркування в систему і модуль детектування вільних місць. Описані функціональні та нефункціональні вимоги до системі, сформовано діаграму варіантів використання, описано акторів та специфікацію. Були розроблені діаграма компонентів та діаграми діяльності.

2. У цьому розділі наведено результати тестування нейронної мережі, а також подано протокол функціонального тестування. Нейронна мережа показала досить високу ефективність, а система успішно пройшла функціональне тестування.

ВИСНОВКИ

В магістерському дослідженні отримано наступні наукові та практично-орієнтовані результати:

1. Огляд існуючих рішень показав, що завдання визначення вільних місць на парковках є актуальним на сьогоднішній день. На даний момент існує багато реалізованих проектів з детектування вільних місць на парковках. Однак точність детектування в готових продуктах відбувається не завжди коректно. Також, варто відзначити існування широкого ряду бібліотек для створення та навчання нейронних мереж.

2. Розроблена просторово-точна модель, яка дозволяє визначати вплив різних просторових сценаріїв на організацію паркувань в межах великого міста. На відміну від відомих, традиційних моделей цей підхід імітує дії кожного водія у просторово-точному міському середовищі, що значно підвищує значущість результатів моделювання.

3. Спираючись на нестачу реалізації додатків з фіксованою розміткою, було прийнято рішення використати підхід із сегментацією об'єктів на зображенні, щоб можна було визначати місця на паркування відштовхуючись від розташування автомобілів на зображенні.

4. Було обрано архітектуру нейронної мережі. Були спроектовані модуль додавання паркування в систему і модуль детектування вільних місць. Описані функціональні та нефункціональні вимоги до системі, сформовано діаграму варіантів використання, описано акторів та специфікацію. Були розроблені діаграма компонентів та діаграми діяльності.

5. Наведено результати тестування нейронної мережі, а також подано протокол функціонального тестування. Нейронна мережа показала досить високу ефективність, а система успішно пройшла функціональне тестування.

6. Програмно реалізовано мобільний додаток для пошуку вільних паркувальних місць, який є клієнтським засобом для взаємодії із реалізованим програмним комплексом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Benenson I. Entity-based modeling of urban residential dynamics: the case of Yaffo, Tel-Aviv. Environment and Planning B: Planning and Design. / Benenson I., Omer I., Hatna E. - 2019. - V. 29. - P. 491-512.

2. Bandman O. Computation properties of spatial dynamics simulation by probabilistic cellular automata / O. Bandman // Future Generation Computer Systems. - 2019. - V.21. - P. 633-664.

3. Patrakeev I.M. Foundation of the parking lots in major cities on the basic of the space-accurate simulation / I.M. Patrakeev, V.E. Zhukov, O.G. Leontyeva // Scientific Notes of Taurida National V. Vemadsky University. - Series: Geography. - 2010. - Vol. 23 (62). - № 2 - P. 222-231.

4. Kayal, P.; Perros, H. A comparison of IoT application layer protocols through a smart parking implementation. In Proceedings of the 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, France, 7–9 March 2017. [Google Scholar].

5. Soaibuzzaman, A.S.; Rahman, M.S.; Rahaman, M. A Blockchain-Based Architecture for Integrated Smart Parking Systems. In Proceedings of the International Conference on Pervasive Computing and Communications Workshops, Kyoto, Japan, 11–15 March 2019. [Google Scholar].

6. Yang, J.; Portilla, J.; Riesgo, T. Smart parking service based on Wireless Sensor Networks. In Proceedings of the 38th Annual Conference on IEEE Industrial Electronics Society, Montreal, QC, Canada, 25–28 October 2012. [Google Scholar].

7. Allam, Z. On Smart Contracts and Organisational Performance: A Review of Smart Contracts through the Blockchain Technology. Rev. Econ. Bus. Stud. 2018, 11, 137–156. [Google Scholar] [CrossRef][Green Version].