НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

“_____” червня 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

з освітньо-професійною програмою

«Системне програмування та спеціалізовані комп’ютерні системи»

по спеціальності **123 «Комп’ютерна інженерія»**

на тему: Компютерна система управління паркуванням автортранспорту
_____ _____

Виконав:

студент IV курсу, групи КВ-11
(шифр групи)

Дзуман Владсилав Михайлович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц. каф. СПіСКС, к.т.н., доцент Павловський В. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПіСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент професор каф. ОТ ФІВТ, д.т.н., проф. Жабін В. І. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма

«Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» грудня 2024 р.

ЗАВДАННЯ
на дипломний проєкт студента
Дзумана Владсилава Михайловича
(прізвище, ім'я, по батькові)

1. Тема проєкту «Компютерна система управління паркуванням автотранспорту»

керівник проєкту доц. каф. СПіСКС, к.т.н., доцент Павловський В. І.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » червня 2025р. №

2. Термін подання студентом проєкту « » червня 2025 р.

3. Вихідні дані до проєкту : див. Технічне завдання.

4. Зміст пояснювальної записки:

1) Аналіз існуючих рішень у сфері управління паркуванням
2) Проектування системи управління куруванням;
3) Алгоритм функціонування системи управлінням;
4) Тестування і оцінка продовжуваної системи;

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

- Блок-схема загального алгоритму роботи розробленого веб-додатка.

- Блок-схема алгоритму обробки зображень.
- Блок-схема алгоритму обробки відеофайлів.
- Блок-схема алгоритму навчання нейронної моделі.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент, к.т.н.		

7. Дата видачі завдання 30 жовтня 2024р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	16.11.2024	
2.	Розроблення та узгодження технічного завдання	25.11.2024	
3.	Розроблення структури додатку	17.01.2025	
4.	Розробка дизайну та графічних елементів	04.02.2025	
5.	Програма реалізації додатка	15.03.2025	
6.	Тестування додатка	04.04.2025	
7.	Підготовка матеріалів текстової частини проєкту	24.04.2025	
8.	Підготовка матеріалів графічної частини проєкту	17.05.2025	
9.	Оформлення технічної документації	28.05.2025	

Студент _____ Владсилав ДЗУМАН
(підпис) (ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Володимир ПАВЛОВСЬКИЙ
(підпис) (ім'я та ПРІЗВИЩЕ)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (52 с., 10 рис., 0 табл., 3 додатки, 18 слайдів).

Об’єкт розробки – комп’ютерна система управління паркуванням Розроблена система дозволяє автоматично визначати зайнятість паркувальних місць на основі обробки зображень та відео, отриманих користувачем. Основний функціонал реалізовано через веб-інтерфейс, що забезпечує зручну взаємодію з системою та перегляд результатів аналізу. У процесі розробки застосовано сучасні технології: мову програмування Python, фреймворк Flask для створення серверної частини, бібліотеку OpenCV для обробки зображень, модель YOLOv5 для виявлення транспортних засобів, а також HTML, CSS, JavaScript для реалізації користувацького інтерфейсу.

У ході виконання роботи:

- проведено аналіз сучасних рішень у сфері управління паркуванням та виявлення транспортних засобів;
- розроблено архітектуру системи, визначено апаратне та програмне забезпечення;
- реалізовано основні алгоритми розпізнавання транспортних засобів та визначення зайнятих/вільних місць;
- протестовано роботу системи в умовах, наближених до реальних.

Система є практичним інструментом для автоматизації моніторингу паркувального простору та може бути використана в муніципальних чи комерційних проєктах.

Ключові слова: КОМП’ЮТЕРНИЙ ЗІР, YOLOv5, ПАРКУВАННЯ, ВЕБ-ДОДАТОК, PYTHON, FLASK, OPENCV, SQLITE, JAVASCRIPT.

ABSTRACT

The qualification work includes an explanatory note (52 pages, 10 figures, 0 tables, 3 appendices, 18 slides).

The object of the study is a computer-based parking management system using computer vision technology.

The developed system enables automatic detection of occupied and vacant parking spaces based on analysis of uploaded images or video streams. The functionality is implemented via a web interface, allowing convenient interaction and result visualization. The development process involved modern technologies, including the Python programming language, the Flask framework for server-side logic, the OpenCV library for image processing, the YOLOv5 model for vehicle detection, and HTML, CSS, and JavaScript for the user interface.

During the development process:

- existing parking management solutions and object detection methods were analyzed;
- system architecture, hardware and software components were designed;
- key algorithms for vehicle detection and parking space status recognition were implemented;
- system performance was tested in near-real conditions.

The developed system serves as a practical tool for automating parking monitoring and can be integrated into municipal or commercial parking infrastructure projects.

Key words: COMPUTER VISION, YOLOv5, PARKING, WEB APPLICATION, PYTHON, FLASK, OPENCV, SQLITE, JAVASCRIPT.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.002 ТЗ	Комп’ютерна система	4		
			управління паркуванням			
			автотранспорту			
			Технічне завдання			
	А4	ІАЛЦ.467200.003 ТП	Комп’ютерна система	2		
			управління паркуванням			
			автотранспорту			
			Відомості технічного			
			проєкту			
	А4	ІАЛЦ.467200.004 ПЗ	Комп’ютерна система	52		
			управління паркуванням			
			автотранспорту			
			Пояснювальна записка			

					ІАЛЦ.467200.001 ОА	
Змін.	Арк.	№ докум.	Підпис	Дата		
Розробив		Дзуман В. М.			Комп’ютерна система управління паркуванням автотранспорту **Опис альбому**	Літ. / Аркуш 1 / Аркушів 2
Перевірив		Павловський В. І.				
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-11
Н. контроль		Клятченко Я.М.				
Зав. каф.		Романкевич В.О.				

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.005 Д1	Загальний алгоритм роботи	1		
			розробленого веб-додатка.			
			Блок-схема			
	А4	ІАЛЦ.467200.006 Д2	Алгоритм обробки	1		
			зображень.			
			Блок-схема			
	А4	ІАЛЦ.467200.007 Д3	Алгоритм обробки	1		
			відеофайлів.			
			Блок-схема			
	А4	ІАЛЦ.467200.008 Д4	Алгоритм навчання	1		
			нейронної моделі.			
			Блок-схема			
		Диск DVD-ROM	Текст пояснювальної			
			записки. Тексти програм.			
			Графічний матеріал			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ 2

2. ПІДСТАВА ДЛЯ РОЗРОБКИ 2

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ 2

4. ДЖЕРЕЛА РОЗРОБКИ 2

5. ТЕХНІЧНІ ВИМОГИ 2

5.1. Вимоги до програмного продукту, що розробляється 3

5.2. Вимоги до апаратного забезпечення 3

5.3. Вимоги до програмного та апаратного забезпечення користувача 3

6. ЕТАПИ РОЗРОБКИ 4

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.002 ТЗ		
Розробив		Дзуман В. М.			*Комп'ютерна система управління паркуванням автотранспорту* ***Технічне завдання***	Літ.	Аркуш 1 / Аркушів 4
Перевірив		Павловський В. І.					
Н. контроль		Клятченко Я. М.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-11	
Затвердив		Романкевич В.О.					

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «компютерна система управління паркування автотранспорту».

Галузь застосування: автоматизоване управління паркувальними майданчиками з використанням засобів комп’ютерного зору.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даної роботи є створення комп’ютерної системи управління паркуванням на основі технологій комп’ютерного зору для автоматичного виявлення транспортних засобів і визначення зайнятості паркувальних місць при обробці зображень та відео.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Підтримка основних сучасних браузерів (Google Chrome, Microsoft Edge, Mozilla Firefox, Opera, Safari);
- Можливість завантаження та обробки зображень (формати: jpg, jpeg, png);
- Можливість завантаження та обробки відеофайлів (формат: mp4);
- Можливість отримання інформації про положення об’єктів медіафайлів;
- Можливість зберігання оброблених медіафайлів на власний пристрій;

5.2. Вимоги до апаратного забезпечення

- Процесор: Intel Core i5 (8 покоління) або аналогічний AMD Ryzen 5;
- Оперативна пам’ять: не менше 4 ГБ;
- Наявність доступу до мережі Internet(GPRS, EDGE, 3G, 4G);

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Операційна система: Windows 10, Linux або macOS з підтримкою сучасних браузерів;
- Стабільне підключення до мережі Інтернет;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою роботи	16.11.2024
2.	Розроблення та узгодження технічного завдання	25.11.2024
3.	Розроблення структури додатку	17.01.2025
4.	Розроблення дизайну та графічних елементів	04.02.2025
5.	Програма реалізації додатку	15.03.2025
6.	Тестування додатку	04.04.2025
7.	Підготовка матеріалів текстової частини проєкту	24.04.2025
8.	Підготовка матеріалів графічної частини проєкту	17.05.2025
9.	Оформлення технічної документації проєкту	28.05.2025

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.004 ПЗ	Комп’ютерна система	62		
			управління паркуванням			
			автотранспорту			
			Пояснювальна записка			
	А4	ІАЛЦ.467200.005 Д1	Загальний алгоритм роботи	1		
			розробленого веб-додатка.			
			Блок-схема			
	А4	ІАЛЦ.467200.006 Д2	Алгоритм обробки	1		
			зображень.			
			Блок-схема			
	А4	ІАЛЦ.467200.007 Д3	Алгоритм обробки	1		
			відеофайлів.			
			Блок-схема			
	А4	ІАЛЦ.467200.008 Д4	Алгоритм навчання	1		
			нейронної моделі.			
			Блок-схема			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.003 ТП		
Розробив		Дзуман В. М..			Комп’ютерна система управління паркуванням автотранспорту **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Павловський В. І.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск DVD-ROM	Текст пояснювальної			
			записки. Тексти програм.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.003 ТП	Арк. 2

Пояснювальна записка
до дипломного проєкту

на тему: «Комп’ютерна система управління паркуванням автотранспорту»

Київ – 2025 рік

ЗМІСТ

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У СФЕРІ УПРАВЛІННЯ ПАРКУВАННЯМ 7
1.1. Завдання та цілі дослідження 7
1.2. Аналіз існуючих засобів 11
1.3. Обґрунтування теми дипломного проєкту 15
2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ КУРУВАННЯМ 17
2.1. Арзітектура системи управління паркуванням 17
2.1.1. Програмне забезпечення обробки відео та комп'ютерного зору .. 18
2.1.2. Інтерфейси для взаємодії з апаратними засобами 19
2.2. Апаратне забезпечення 21
2.2.1. Апаратне забезпечення для навчання та тестування нейронної мережі 21
2.2.2. Апаратне забезпечення для підтримки роботи веб-додатка: 22
2.3. Принцип роботи системи управління паркуванням 23
2.4. Структура програмних модулів 24
2.4.1. Опис алгоритмів виявлення і відстеження транспортних засобів 27
2.4.2. модуль керування паркувальними місцями 28
2.4.3. Інтерфейс користувача 29
2.5. Технічний інтерфейс та інтеграція з іншими системами 32
3. АЛГОРИТМИ ФУНКЦІОНУВАННЯ СИСТЕМИ УПРАВЛІННЯМ ПАРКУВАННЯМ 34
3.1. Адгоритм виявлення та розпізнавання транспортних засобів 34
3.2. Алгоритм визначення вільних паркувальних місць: 35

3.3. Алгоритм роботи керування паркувальними місцями та взаємодії з користувачем 37
3.4. Алгоритм обробки та зберігання даних про паркування 38
4. ТЕСТУВАННЯ І ОЦІНКА ПРОДОВЖУВАННЯ СИСТЕМИ 41
4.1. Тестування алгоритмів виявлення та трекінгу транспортних засобів ... 43
4.2. Оцінка точності, повноти та часу обробки 44
4.3. Аналіз ефективності алгоритмів в реальних умовах 45
4.4. Тестування інтеграції з іншими системами 46
ВИСНОВКИ 49
СПИСОК ЛІТЕРАТУРИ 51

ДОДАТКИ
Додаток А. Копії графічних матеріалів
ІАЛЦ.467200.005 Д1 Загальний алгоритм роботи розробленого веб-додатка. Блок-схема
ІАЛЦ.467200.006 Д2 Алгоритм обробки зображень. Блок-схема
ІАЛЦ.467200.007 Д3 Алгоритм обробки відеофайлів. Блок-схема
ІАЛЦ.467200.008 Д4 Алгоритм навчання нейронної моделі. Блок-схема
Додаток Б
Додаток В

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АЗ – Апаратне забезпечення
ОЗУ – Оперативна пам’ять
ПЗ – Програмне забезпечення
БД – База даних

GUI – Graphical User Interface (Графічний інтерфейс користувача)

HTTP – HyperText Transfer Protocol (Протокол передачі гіпертексту)

API – Application Programming Interface (Прикладний програмний інтерфейс)

CNN – Convolutional Neural Network (Згорткова нейронна мережа)

YOLO – You Only Look Once (Система відстеження об’єктів у реальному часі)

Flask – Легкий вебфреймворк на Python

SQLite – Вбудована система управління базами даних

OpenCV – Open Source Computer Vision Library (Бібліотека комп’ютерного зору з відкритим кодом)

Python – Мова програмування високого рівня з динамічною типізацією

HTML – HyperText Markup Language (Мова розмітки гіпертексту)

CSS – Cascading Style Sheets (Каскадні таблиці стилів)

JS або JavaScript – Мова сценаріїв для інтерактивності вебдодатків

JPG або JPEG – Формат зображень з втратами якості

PNG – Формат зображень без втрат якості

MP4 – Формат мультимедійних відеофайлів

IOU – Intersection over Union (Міра перетину в обчисленні точності об'єктів)

SSD-накопичувач – Solid-State Drive (Твердотілий накопичувач)

GPU – Graphics Processing Unit (Графічний процесор)

Вебінтерфейс – Засіб взаємодії користувача із системою через браузер

Датасет – Набір структурованих навчальних даних для штучного інтелекту

ШІ – Штучний інтелект

YOLOv5 – П'ята версія алгоритму YOLO для розпізнавання об’єктів у реальному часі

JSON – JavaScript Object Notation (Формат обміну даними)

CSV – Comma-Separated Values (Текстовий формат для представлення табличних даних)

UI – User Interface (Інтерфейс користувача)

UX – User Experience (Досвід користувача при роботі з інтерфейсом)

SQL – Structured Query Language (Мова структурованих запитів до баз даних)

CV2 або OpenCV – Бібліотека комп’ютерного зору

Torch або PyTorch – Фреймворк глибокого навчання, створений для Python

HTML5 – П’ята версія мови HTML, що підтримує мультимедіа та API

CORS – Cross-Origin Resource Sharing (Механізм доступу до ресурсів з іншого домену)

UUID – Universally Unique Identifier (Універсально унікальний ідентифікатор)

FPS – Frames Per Second (Кількість кадрів за секунду у відео)

PNG – Portable Network Graphics (Растровий графічний формат)

BMP – Bitmap Image File (Растровий формат графічного файлу)

IoU – Intersection over Union (Міра перетину для об'єктів у задачах розпізнавання)

ВСТУП

У сучасному світі інформаційних технологій активно зростає потреба у системах, що здатні автоматизовано розпізнавати об’єкти на зображеннях і відео в реальному часі. Це обумовлено широким впровадженням камер спостереження, розумних пристроїв і інфраструктурних рішень у сфері безпеки, транспорту та міського управління. Одним із перспективних напрямів використання таких технологій є організація систем управління паркувальними просторами, що дозволяє підвищити ефективність використання паркомісць, зменшити затори та знизити рівень стресу у водіїв при пошуку вільного місця.

В основі цих інтелектуальних рішень лежать алгоритми комп’ютерного зору, зокрема згорткові нейронні мережі, здатні ефективно визначати транспортні засоби на зображеннях. Одним із найпоширеніших і найефективніших підходів є використання архітектури YOLO (You Only Look Once), яка забезпечує швидке та точне виявлення об’єктів. У поєднанні з сучасними веб-технологіями стає можливим створення інтерактивних додатків, які надають користувачеві зручний доступ до системи аналізу паркування.

Особливе значення має створення веборієнтованих рішень, що працюють без необхідності встановлення додаткового програмного забезпечення. Такі системи здатні взаємодіяти з користувачем у режимі реального часу, зберігати результати обробки, а також масштабуватись залежно від кількості користувачів. Поєднання потужностей комп’ютерного зору з гнучкістю веб-інтерфейсів відкриває широкі можливості для автоматизації обробки відео- та фотоматеріалів.

Саме тому тематика розробки веб-додатку для виявлення вільних паркувальних місць з використанням нейронної мережі є актуальною та має

велике прикладне значення в умовах розвитку «розумних міст» і цифрової інфраструктури.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У СФЕРІ УПРАВЛІННЯ ПАРКУВАННЯМ

1.1. Завдання та цілі дослідження

Сучасні технології комп’ютерного зору та нейромереж стали ключовими інструментами у розв’язанні багатьох прикладних задач, зокрема в сфері автоматизації процесів, моніторингу та управління об’єктами в реальному середовищі. Комп’ютерний зір — це галузь штучного інтелекту, що займається аналізом і обробкою зображень або відео з метою отримання інформації, прийняття рішень або керування технічними системами. Його завдання полягає в тому, щоб навчити комп’ютери “бачити” так, як це робить людина, тобто розпізнавати об’єкти, оцінювати їхню форму, положення, рух, а також інтерпретувати ситуацію в цілому. Розвиток цієї галузі став можливим завдяки поєднанню прогресу в області цифрової обробки сигналів, оптичних сенсорів, високопродуктивних процесорів та алгоритмів машинного навчання.[1]

Історично комп’ютерний зір почав формуватись у 1960–70-х роках, коли з’явилися перші спроби автоматизованого аналізу зображень. Проте на той час алгоритми були надто простими і не забезпечували прийнятної точності. Справжній прорив стався лише після 2010 року з розвитком глибокого навчання (deep learning) і згорткових нейронних мереж (CNN), які стали основним інструментом для розпізнавання об'єктів на зображеннях. Наприклад, алгоритми серії YOLO (You Only Look Once) дозволили здійснювати одночасне виявлення та класифікацію об'єктів у режимі реального часу, що зробило їх незамінними у багатьох сферах — від відеонагляду до систем автономного транспорту.

Нейронні мережі, які лежать в основі комп’ютерного зору, самі по собі є математичними моделями, натхненними принципами роботи людського мозку. Вони складаються з великої кількості взаємопов’язаних елементів —

нейронів, здатних до самонавчання на основі великих обсягів даних. Початково такі мережі мали обмежену ефективність, але з розвитком апаратного забезпечення (зокрема графічних процесорів) та появою великих навчальних вибірок ситуація докорінно змінилася. Вже зараз глибокі нейромережі застосовуються у задачах розпізнавання облич, номерних знаків, жестів, дорожніх ситуацій тощо, демонструючи рівень точності, який часто перевищує людські можливості.

Таким чином, поєднання технологій комп'ютерного зору та нейромереж створює потужний інструментарій для автоматизованого сприйняття навколишнього середовища, що є фундаментом для створення інтелектуальних систем, здатних автономно аналізувати ситуацію, реагувати на події і приймати обґрунтовані рішення без участі людини. Ці можливості лягли в основу ідеї розробки комп'ютерної системи управління паркуванням, де завдання ідентифікації транспортних засобів, відстеження їхнього руху та керування паркувальними місцями можуть бути реалізовані саме за допомогою згаданих технологій.

Системи управління паркуванням — це технологічні рішення, що використовуються для організації, моніторингу та оптимізації використання паркувального простору в містах, на приватних територіях або в спеціалізованих зонах, таких як аеропорти, торговельні центри чи офісні комплекси. Основною метою таких систем є підвищення ефективності використання паркувального простору, зниження заторів, покращення доступності вільних місць та зменшення викидів в атмосферу за рахунок скорочення часу пошуку паркування.

Ідея організованого управління паркуванням почала набирати актуальності ще в середині XX століття, коли темпи урбанізації й автомобілізації значно зросли. Перші автоматизовані системи були доволі простими: це могли бути лише лічильники в'їзду та виїзду або механічні бар'єри, керовані охоронцями.

З розвитком мікропроцесорної техніки, сенсорів і мережевих технологій, ці системи стали складнішими, здатними здійснювати облік в режимі реального часу, вести відеоспостереження, використовувати ідентифікацію за номерними знаками тощо.

Особливо суттєвий прогрес відбувся у XXI столітті з появою технологій комп'ютерного зору та штучного інтелекту. Ці підходи дозволили створювати більш гнучкі, точні й масштабовані рішення. Замість звичайних сенсорів у кожному паркувальному місці, камери разом із програмними алгоритмами почали використовуватися для виявлення транспортних засобів, ідентифікації вільних місць, прогнозування заповненості та навіть інтеграції з мобільними додатками для зручності користувачів.

Серед видатних дослідників і практиків, які зробили внесок у розвиток інтелектуальних транспортних систем, варто згадати Роберта Шилдса (Robert L. French Shields), який займався розробкою перших автоматизованих паркувальних рішень у США ще в 70-х роках. У галузі комп'ютерного зору значну роль відіграли Янн ЛеКун (Yann LeCun), Джошуа Бенджіо (Yoshua Bengio) та Джеффрі Хінтон (Geoffrey Hinton) — саме їхня робота зі штучних нейронних мереж дала змогу реалізовувати системи, здатні обробляти відеопотоки й розпізнавати об'єкти в реальному часі.

Зважаючи на стрімке зростання кількості автомобілів у містах, проблема ефективного використання паркувального простору стала однією з ключових у сучасному міському плануванні. Традиційні методи, засновані на візуальному контролі, встановлених датчиках або ручному адмініструванні паркування, вже не відповідають вимогам часу. Вони часто супроводжуються затримками, помилками, потребують постійного втручання персоналу та не забезпечують достатньої масштабованості. Саме тому все більше уваги приділяється інтелектуальним системам, здатним самостійно здійснювати моніторинг, аналіз і керування процесами в реальному часі.

На цьому фоні технології комп’ютерного зору демонструють значний потенціал, адже вони дозволяють автоматично обробляти візуальну інформацію з камер — без участі оператора. Камера фіксує зображення або відеопотік, після чого ці дані передаються у програмну систему, яка за допомогою алгоритмів розпізнавання аналізує кожен кадр, виявляючи ключові об’єкти — транспортні засоби, розмітку, вільні або зайняті паркомісця, дорожні знаки тощо. Залежно від поставлених завдань, система може відслідковувати рух автомобілів, фіксувати в’їзд і виїзд, запам’ятовувати номерні знаки або навіть визначати тривалість перебування на стоянці.

Інформація, отримана у результаті такого аналізу, не є самодостатньою — вона перетворюється на структуровані дані, які потім використовуються для прийняття рішень. Наприклад, система може автоматично зареєструвати факт паркування, оновити карту зайнятих місць, повідомити користувача через мобільний додаток про наявність вільного місця або сформувати звіт для адміністратора. В окремих випадках такі системи можуть інтегруватися з камерами контролю швидкості чи іншими компонентами міської інфраструктури, забезпечуючи більш комплексне управління транспортними потоками.

Технологічно подібні системи зазвичай базуються на заздалегідь навчених нейромережах, які проходили підготовку на тисячах або навіть мільйонах зображень, що містять різні варіанти дорожніх ситуацій, типів авто, кутів огляду та умов освітлення. Такий підхід дозволяє досягати високого рівня точності навіть у складних умовах, таких як ніч, туман або сильна тінь. Навіть якщо умови зовнішнього середовища змінюються, система здатна адаптуватися й надалі правильно інтерпретувати ситуацію.

З огляду на це, завданням даного дослідження є не лише створення окремого елементу автоматизації, а побудова цілісної комп’ютерної системи управління паркуванням, яка враховує реальні умови експлуатації, технічні

обмеження й потреби кінцевих користувачів. Основною метою є забезпечення ефективного, зручного та масштабованого способу контролю за паркувальними місцями на основі сучасних алгоритмів комп'ютерного зору. Також серед важливих цілей — дослідження можливостей взаємодії програмних модулів з апаратною частиною, зокрема камерами й сенсорами, а також аналіз точності, надійності та практичної придатності запропонованих рішень у різних сценаріях.

1.2. Аналіз існуючих засобів

З огляду на завдання, пов'язані з виявленням та класифікацією транспортних засобів у межах паркувального простору, ключову роль у розробці сучасних систем управління відіграють алгоритми машинного навчання, зокрема ті, що орієнтовані на обробку візуальних даних. На відміну від класичних підходів, коли розробник мав вручну задавати правила обробки зображень, машинне навчання дозволяє алгоритмам самостійно вивчати закономірності на основі великої кількості прикладів. Це робить моделі більш гнучкими, точними та здатними адаптуватися до змінних умов середовища.[2]

Одним із найефективніших напрямів у цьому контексті є використання глибокого навчання (deep learning) — підходу, що базується на багаторівневих нейронних мережах. Особливо важливою є архітектура згорткових нейронних мереж (CNN), які чудово працюють із зображеннями. Ці мережі автоматично виокремлюють ознаки об'єктів, що дозволяє ідентифікувати транспортні засоби навіть у складних умовах: за часткового перекриття, змін освітлення, різних ракурсів тощо. Такий підхід дає змогу точно визначати контури автомобілів, розпізнавати номерні знаки та виявляти сам факт наявності авто на паркомісці.

Важливу роль відіграє також попереднє навчання моделей на великих наборах даних — таких як COCO, ImageNet або спеціалізованих вибірках,

пов’язаних із вуличним рухом. Це забезпечує їм здатність швидко адаптуватися до нових сценаріїв або навіть доображувати дані за допомогою так званого трансферного навчання, коли вже готову модель донавчають на нових прикладах з конкретного середовища.

Ще однією невід’ємною частиною таких систем є попередня обробка даних, яка включає стабілізацію відео, фільтрацію шуму, покадрове зіставлення об’єктів і нормалізацію кольорів або яскравості. Це дозволяє моделі не лише швидко, а й стабільно працювати в реальному часі. У цьому контексті застосовуються бібліотеки, такі як OpenCV для базової обробки зображень, та фреймворки типу PyTorch або TensorFlow для побудови й запуску глибоких моделей. Для детекції об’єктів популярними є моделі типу YOLO (You Only Look Once), SSD (Single Shot Detector) або Faster R-CNN — кожна з яких має свої переваги щодо точності, швидкодії та ресурсоємності.

Розвиток інтелектуальних систем управління паркуванням став логічною відповіддю на актуальні виклики урбанізації та зростання кількості автотранспорту. У міру того як традиційні методи — на кшталт фізичних датчиків на кожному паркомісці або механічного обліку в’їзду та виїзду — почали втрачати ефективність, відбулася трансформація підходів до моніторингу та регулювання паркувального простору. Сучасні рішення дедалі частіше ґрунтуються на програмному аналізі відеопотоків у поєднанні з глибокими нейронними мережами, що дозволяє скоротити витрати на інфраструктуру й одночасно підвищити гнучкість систем.

Комп’ютерний зір у таких системах виконує функцію первинного аналізатора — вхідні відеодані з камер перетворюються на числові представлення, на основі яких моделі визначають статус кожного паркомісця. Завдяки використанню згорткових нейронних мереж, зокрема таких архітектур як YOLOv5, YOLOv8, SSD або Faster R-CNN, забезпечується висока точність виявлення об'єктів навіть в умовах низької освітленості,

поганої погоди чи нестандартного розташування камер. Особливістю цих підходів є їхня здатність працювати в режимі реального часу, що є критичним для динамічного середовища, як-от паркінги великих ТРЦ або аеропортів.

Значна частина розробок у цій сфері спрямована на мінімізацію фізичних втручань у середовище. На противагу старішим IoT-платформам, які передбачали встановлення сенсорів на кожному місці, сучасні CV-системи базуються на інтерпретації зображення з єдиної або обмеженої кількості оглядових камер. Це дає змогу швидко масштабувати систему, адаптувати її до нових майданчиків та зменшити вартість впровадження. Крім того, такі системи легко піддаються модифікації через оновлення програмного забезпечення — зокрема, повторне навчання моделей на локальних даних підвищує їхню ефективність у специфічному середовищі.

У практиці вже широко застосовуються рішення, що підтверджують доцільність такого підходу. Наприклад, у північноамериканських містах активно впроваджуються платформи, які забезпечують безконтактну ідентифікацію автомобіля, автоматичну фіксацію моменту паркування та оплату без участі користувача. Ці платформи поєднують комп’ютерний зір із платіжними шлюзами та мобільними застосунками. Технологічна перевага таких рішень полягає в їхній модульності — вони можуть поєднувати розпізнавання номерних знаків, трекінг положення авто, взаємодію з базами даних, а також персоналізовану взаємодію з користувачем. У деяких випадках реалізовано функції аналітики завантаженості, прогнозування пік-навантаження та автоматизованого управління динамічними тарифами.

Ці рішення не лише підтверджують технічну здійсненність концепції комп’ютерного бачення в міських умовах, але й демонструють її економічну ефективність: зменшення витрат на обслуговування, підвищення пропускної здатності стоянок, зниження кількості порушень та покращення загального користувацького досвіду. У підсумку можна відзначити, що аналіз наявних

розробок свідчить про зрілість технологій комп’ютерного зору для паркувальних систем, їхню адаптивність до різних середовищ і перспективність для подальшого розвитку в межах інтелектуальних транспортних систем.

Сфера організації паркування в містах є однією з ключових у контексті розвитку розумної інфраструктури (smart city). Зі зростанням кількості транспортних засобів та щільності забудови, традиційні підходи до організації паркінгів виявляються неефективними. Водії витрачають значний час на пошук вільного місця, що призводить до заторів, підвищеного споживання пального та негативного впливу на довкілля.

Існуючі системи управління паркуванням можуть бути умовно поділені на три категорії: ручні (з участю персоналу), напівавтоматичні (із використанням датчиків або камер без аналізу відеопотоку), та автоматизовані з комп’ютерним зором. Ручні системи мають високу ймовірність помилки через людський фактор, натомість повністю автоматизовані рішення забезпечують кращу точність та швидкість обробки.

Інтелектуальні системи з комп’ютерним зором дозволяють проводити аналіз стану паркувального майданчика в реальному часі, виявляючи зайняті та вільні місця без фізичних сенсорів. Це значно здешевлює інфраструктурні витрати. Водночас вони потребують високої обчислювальної потужності та точних моделей машинного навчання. Найпоширенішими прикладами є проєкти, реалізовані в межах Smart Parking System в Європі, або рішень на базі Raspberry Pi із підключеними камерами для невеликих майданчиків.

Таким чином, використання технологій комп’ютерного зору для паркувальних систем відкриває широкі можливості для підвищення ефективності, зменшення витрат та забезпечення комфорту користувачів. Проте реалізація таких рішень вимагає розробки гнучких, масштабованих та

точних програмно-апаратних платформ — саме це завдання і ставилось у межах дипломної роботи.

1.3. Обґрунтування теми дипломного проєкту

Однією з актуальних проблем сучасної урбаністики є перенасиченість вуличного простору автомобілями, що спричиняє хронічну нестачу паркувальних місць, зростання рівня трафіку, підвищення рівня викидів шкідливих речовин у повітря та суттєве зниження якості пересування містом. У таких умовах ефективне управління паркуванням виступає не лише технічним, а й соціально значущим завданням. Особливої актуальності ця проблема набуває в умовах великих міст України, де паркувальна інфраструктура часто розвивалася стихійно, без належного планування та з урахуванням застарілих підходів до організації дорожнього руху. У зв’язку з цим виникає потреба у створенні інтелектуальних систем, здатних динамічно реагувати на зміни в завантаженості паркінгів, забезпечувати оперативне інформування водіїв та автоматизувати процес керування паркувальними місцями.[3]

Обґрунтування теми дипломного проєкту базується на міждисциплінарному підході, що об'єднує знання в галузі штучного інтелекту, комп’ютерного зору, автоматизованих інформаційних систем та цифрової урбаністики. На відміну від традиційних систем управління паркуванням, які базуються на сенсорах, лічильниках або ручному введенні даних, застосування комп’ютерного зору відкриває нові можливості щодо зменшення витрат на обладнання, підвищення точності детекції транспортних засобів і оперативного збору інформації в реальному часі. Це дозволяє не лише аналізувати поточний стан заповненості паркінгу, а й створювати прогностичні моделі з урахуванням часу доби, днів тижня, подій у місті тощо.

З огляду на темпи розвитку технологій комп’ютерного зору, зокрема алгоритмів виявлення об'єктів, таких як YOLOv8, а також доступність потужних обчислювальних платформ (наприклад, NVIDIA Jetson, Raspberry Pi з прискоренням AI, або мобільні додатки з інтегрованими модулями TensorFlow Lite), створення функціональної та ефективної системи управління паркуванням стало як технічно досяжним, так і практично необхідним. Тема проєкту не є вузькоспеціалізованою розробкою, що матиме обмежену сферу застосування — навпаки, мова йде про універсальний підхід, що може масштабуватися на різні типи інфраструктур: від невеликих приватних стоянок до муніципальних паркінгів або зон з обмеженим доступом.[4]

Крім того, застосування комп’ютерного зору у проєкті дозволяє вирішувати низку додаткових завдань: автоматичне розпізнавання номерних знаків для реєстрації в’їзду/виїзду, інтеграція з платіжними сервісами, адаптація до системи “розумного міста” через API та підтримку зв’язку з центральними серверами або хмарними платформами. У перспективі це створює основу для переходу до безпаперових, безконтактних та самонавчальних паркувальних систем, що враховують не лише технічні аспекти, а й зручність користувача.

З практичної точки зору, тематика дипломного дослідження дозволяє реалізувати повний цикл інженерної розробки — від постановки задачі, аналізу вимог і проєктування архітектури, до реалізації програмних модулів, їх тестування в реальному або симульованому середовищі та аналізу ефективності системи. Це відповідає сучасним вимогам до кваліфікаційних робіт бакалаврського рівня, які мають продемонструвати здатність студента не лише опановувати теоретичні знання, а й застосовувати їх на практиці з урахуванням актуальних викликів у галузі інформаційних технологій.

2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ КУРУВАННЯМ

2.1. Арзітектура системи управління паркуванням

Архітектура системи управління паркуванням передбачає комплексну взаємодію програмних та апаратних компонентів, що спільно забезпечують виявлення, обробку, аналіз і відображення інформації про стан паркомісць. У загальному вигляді така система може включати камери спостереження, сенсорні пристрої (ультразвукові або інфрачервоні датчики), контролери для збору та передачі даних, програмне забезпечення для обробки відеопотоків, серверну частину, а також інтерфейс користувача для зручної взаємодії з системою.

У повнофункціональній реалізації камери розміщуються на в'їздах та виїздах паркінгу, а також безпосередньо над паркомісцями. Датчики фіксують наявність автомобіля, а дані передаються на сервер, де відбувається обробка за допомогою алгоритмів комп’ютерного зору. Результати аналізу, включно зі статусом зайнятості кожного місця, зберігаються у базі даних та виводяться через веб-інтерфейс.

З метою реалізації дослідницького прототипу в рамках дипломного проєкту використано спрощений підхід, що не передбачає фізичної інтеграції камер або датчиків. Замість цього для обробки застосовуються статичні зображення з відкритого датасету CarPK, що містить фотографії паркінгів з попередньо анотованими місцями. Система розгортається на персональному комп’ютері або в хмарному середовищі Google Colab і складається з програмного модуля на базі моделі комп’ютерного зору YOLOv5, серверної логіки, реалізованої засобами Python з використанням фреймворку Flask, та веб-інтерфейсу, побудованого на HTML, CSS і JavaScript.[5]

Такий підхід дозволяє зберегти концептуальну структуру повної системи управління паркуванням та водночас забезпечити її роботу в умовах обмежених ресурсів. Усі основні компоненти, включаючи обробку зображень,

аналіз зайнятості місць, керування даними та візуалізацію результатів, інтегровані в єдину архітектуру, яка може бути надалі розширена для підтримки апаратних модулів у разі потреби.

2.1.1. Програмне забезпечення обробки відео та комп’ютерного зору

Програмне забезпечення, що використовується в системі управління паркуванням, базується на принципах комп’ютерного зору та сучасних підходах до глибинного навчання. Основне завдання програмного модуля — аналізувати зображення паркувального простору з метою виявлення зайнятих та вільних місць. У рамках цього проєкту замість потокового відео використовується обробка окремих фотографій, що дозволяє зберегти основну логіку роботи системи, одночасно спрощуючи реалізацію.

В якості основного інструменту обробки зображень обрано нейромережеву модель типу YOLO (You Only Look Once), яка дозволяє проводити детекцію об'єктів у реальному часі. У межах цієї роботи використовується версія YOLOv5, яка зарекомендувала себе як ефективне рішення для задач розпізнавання об'єктів з хорошим балансом між точністю та швидкістю обробки. Попереднє навчання моделі відбувається на відкритому датасеті PKLot, який містить значну кількість зображень паркінгів із розміткою зон зайнятості. [6]

Зображення обробляються програмним модулем на стороні сервера, який реалізований мовою Python. Після завантаження кожного фото система виконує попередню обробку (масштабування, нормалізацію тощо), після чого модель проводить детекцію транспортних засобів. Отримані координати об'єктів аналізуються та класифікуються як зайняті або вільні паркувальні місця. Результати обробки можуть бути представлені у вигляді зображення з візуалізацією, а також у структурованому форматі для подальшого використання інтерфейсами користувача. [7]

Завдяки використанню обробки фотографій замість потокового відео, система зберігає основну логіку функціонування та дозволяє реалізувати процес виявлення зайнятості паркомісць без складних апаратних рішень. Це особливо важливо на етапі розробки та тестування, коли необхідна швидка перевірка гіпотез і модулів. Крім того, програмна архітектура побудована таким чином, щоб залишатися масштабованою — передбачена можливість інтеграції потокового відео, сенсорів присутності або LPR-систем для зчитування номерів у майбутньому без потреби в повному переписуванні системи.[8]

2.1.2. Інтерфейси для взаємодії з апаратними засобами

У межах потенційного розширення функціональності системи управління паркуванням передбачено реалізацію інтерфейсів для взаємодії з апаратними засобами. До таких засобів можуть належати IP-камери відеоспостереження, ультразвукові або інфрачервоні сенсори для виявлення наявності транспортного засобу, а також контролери шлагбаумів та сенсорні панелі. Передбачувані інтерфейси включають як стандартні протоколи передачі даних (наприклад, HTTP/REST, RTSP для потокового відео), так і апаратно-залежні рівні взаємодії через USB, GPIO або серійні порти.

З боку програмної реалізації, такі інтерфейси можуть бути реалізовані у вигляді окремих модулів, що працюють у фоновому режимі та передають дані у центральну систему аналізу. Це дозволяє створити гнучку й адаптовану архітектуру, яка може функціонувати як із реальними апаратними компонентами, так і з їх програмними емуляторами на етапі розробки та тестування.

У межах створення системи управління паркуванням із використанням комп’ютерного зору ключовим завданням є вибір відповідного інструментарію для розробки — бібліотек, фреймворків та моделей глибокого

навчання. Існує низка альтернативних рішень, кожне з яких має свої переваги та обмеження.

Найбільш популярними інструментами для реалізації подібних систем є: OpenCV, TensorFlow, PyTorch, а також фреймворки високого рівня, зокрема Detectron2, MMDetection, MediaPipe, YOLO-серії (YOLOv3–YOLOv8) та інші.

OpenCV є потужною бібліотекою для базової та середньої обробки зображень. Вона має багатий набір функцій для фільтрації, трансформацій, виявлення контурів та інших операцій, однак не має вбудованих інструментів для навчання глибоких нейронних мереж. Її зазвичай поєднують із іншими фреймворками.

TensorFlow — універсальний фреймворк від Google для побудови та розгортання моделей машинного навчання. Його перевагами є широка екосистема, підтримка мобільних пристроїв (через TensorFlow Lite), висока продуктивність при серійному навчанні. Проте сам синтаксис може бути складнішим у порівнянні з PyTorch.[9]

PyTorch, на якому ґрунтується реалізація у цьому проєкті, є гнучким і зручним для розробників. Він активно використовується в дослідницькому середовищі, має простіший інтерфейс і інтегрується з такими проєктами, як YOLOv5, що дає змогу швидко підключити готову модель та адаптувати її до потреб задачі.

YOLO (You Only Look Once) — це родина моделей для обробки зображень у реальному часі. Застосування YOLOv5 забезпечує баланс між швидкістю та точністю. У порівнянні з Detectron2 (який має вищу точність, але працює повільніше), YOLO підходить краще для задач, де важливий відгук у реальному часі.

Враховуючи вимоги до продуктивності, простоти реалізації, точності виявлення та можливостей масштабування, було обґрунтовано вибір бібліотек PyTorch, OpenCV та моделі YOLOv5 для реалізації цієї системи.

2.2. Апаратне забезпечення

Для забезпечення ефективної роботи системи управління паркуванням, особливо в частині виконання обчислювально складних задач комп'ютерного зору, необхідне використання сучасного персонального комп'ютера або серверної станції з відповідними технічними характеристиками. Основними критеріями при виборі апаратного забезпечення є обсяг оперативної пам'яті, швидкодія процесора, наявність графічного процесора з підтримкою CUDA, а також обсяг та швидкість накопичувача.

У якості базової конфігурації для розробки та тестування програмного забезпечення системи було використано комп'ютер з операційною системою Windows 10, процесором не нижче Intel Core i5, 16 ГБ оперативної пам'яті, твердотілим накопичувачем SSD обсягом не менше 256 ГБ, а також графічним процесором NVIDIA з підтримкою бібліотеки CUDA для прискореного оброблення даних за допомогою фреймворку PyTorch.[10]

Наявність GPU є особливо важливою при навчанні моделей комп'ютерного зору (зокрема YOLOv5), оскільки це дозволяє суттєво зменшити час навчання порівняно з використанням виключно CPU. Водночас, у рамках дипломного проєкту, при обмежених апаратних ресурсах допускається використання хмарних платформ, зокрема Google Colab, які надають доступ до графічних прискорювачів без потреби у локальному обладнанні.

2.2.1. Апаратне забезпечення для навчання та тестування нейронної мережі

Апаратне забезпечення, що використовується у системі, дозволяє забезпечити баланс між продуктивністю, гнучкістю розгортання та

доступністю рішень, що є важливим аспектом при розробці прототипу системи в умовах обмеженого бюджету.

Навчання глибинних нейронних мереж, зокрема архітектур типу YOLO, є обчислювально складним процесом, що вимагає значних апаратних ресурсів. У даному проєкті використовується модель комп'ютерного зору YOLOv5, яка ефективно вирішує задачу детекції транспортних засобів на зображеннях паркувального простору. Зважаючи на складність обробки зображень, ефективне навчання та тестування моделі потребує наявності сучасного обчислювального середовища, здатного забезпечити стабільну продуктивність та прискорену роботу з використанням графічного процесора.

У межах цього проєкту використовується персональний комп'ютер із наступною конфігурацією: процесор AMD Ryzen 5 5600H (6 ядер, 12 потоків), 16 ГБ оперативної пам'яті DDR4, твердотільний накопичувач SSD обсягом 512 ГБ, а також відеокарта NVIDIA GeForce GTX 1660 Ti з 6 ГБ відеопам'яті. Операційна система — Windows 11 Pro. Дана конфігурація забезпечує необхідну обчислювальну потужність для ефективного навчання моделей YOLOv5 з помірною кількістю зображень, а також дозволяє проводити інференс у режимі реального часу.

Окрім цього, розроблене рішення має кросплатформену сумісність і може бути розгорнуте на комп'ютерах з операційною системою macOS, за умови підтримки відповідних фреймворків та графічного прискорення. Це створює передумови для гнучкого розгортання системи як у локальному середовищі, так і на альтернативних платформах без необхідності глибокої модифікації коду.

2.2.2. Апаратне забезпечення для підтримки роботи веб-додатка:

Для забезпечення стабільної роботи веб-додатка системи управління паркуванням важливо мати потужну графічну підсистему, оскільки саме вона

відповідає за інференс моделі комп’ютерного зору без затримок. Використана в проєкті апаратна конфігурація, описана у попередньому підпункті, повністю відповідає цим вимогам і є вдалим вибором для одночасного виконання обчислювальних задач, пов’язаних із моделлю YOLOv5, та підтримки користувацького інтерфейсу.

Веб-додаток реалізовано на основі фреймворку Flask із використанням HTML, CSS та JavaScript для створення клієнтської частини. Завдяки цьому розгортання додатка можливе як на Windows, так і на Unix-подібних операційних системах, включаючи macOS. Це забезпечує гнучкість при виборі апаратної платформи, що особливо важливо в умовах різноманітних середовищ розгортання.

Усі компоненти системи функціонують локально без потреби в підключенні до зовнішніх серверів, що гарантує автономність та контроль над усіма етапами обробки — від отримання зображень до виведення результатів користувачеві через веб-інтерфейс.

2.3. Принцип роботи системи управління паркуванням

Принцип роботи системи управління паркуванням ґрунтується на послідовному аналізі візуальної інформації, що надходить із камер спостереження або використовується з наявних зображень. Основна ідея полягає у виявленні транспортних засобів на паркувальному майданчику за допомогою алгоритмів комп’ютерного зору, подальшому визначенні статусу кожного паркомісця (вільне або зайняте) та відображенні цієї інформації в інтерфейсі користувача. Усі процеси реалізовані програмними модулями, які тісно взаємодіють між собою та забезпечують повний цикл обробки — від отримання вхідних даних до виведення результату.

Система працює за таким загальним сценарієм: користувач завантажує зображення (або система автоматично отримує відео/фото з камери), після

чого активується модуль попередньої обробки. На цьому етапі виконується масштабування, вирівнювання кольору, нормалізація або фільтрація шумів. Далі підключається нейронна мережа YOLOv5 або аналогічна модель, яка проводить детекцію об'єктів, зокрема автомобілів, на зображенні. Модель видає координати знайдених об’єктів разом із класами та рівнем довіри. Наступний крок — це співставлення отриманих координат з розміченими паркомісцями, що дозволяє однозначно визначити, які з них є зайнятими.

Після цього оброблена інформація фіксується у внутрішній структурі даних і передається до клієнтської частини через серверний компонент, реалізований на Flask. Користувач бачить результати у веб-інтерфейсі — зазвичай у вигляді зображення з кольоровими рамками навколо кожного паркомісця (наприклад, зелені — вільні, червоні — сині). Окрім візуального відображення, можлива також текстова аналітика — кількість вільних місць, список зайнятих, час останнього оновлення тощо.

Важливо, що система має модульну структуру, яка дозволяє гнучко змінювати спосіб отримання вхідних даних, навчати модель на нових прикладах або інтегрувати додаткові функції — наприклад, автоматичне зчитування номерних знаків або сповіщення на мобільний застосунок. Завдяки цьому рішення є масштабованим і придатним як для локального використання на невеликих паркінгах, так і для більших інфраструктурних проєктів.

2.4. Структура програмних модулів

Тека з файлами вихідного веб-додатку, що містить всі використані файли та папки виглядає наступним чином (рисунок 2.1).

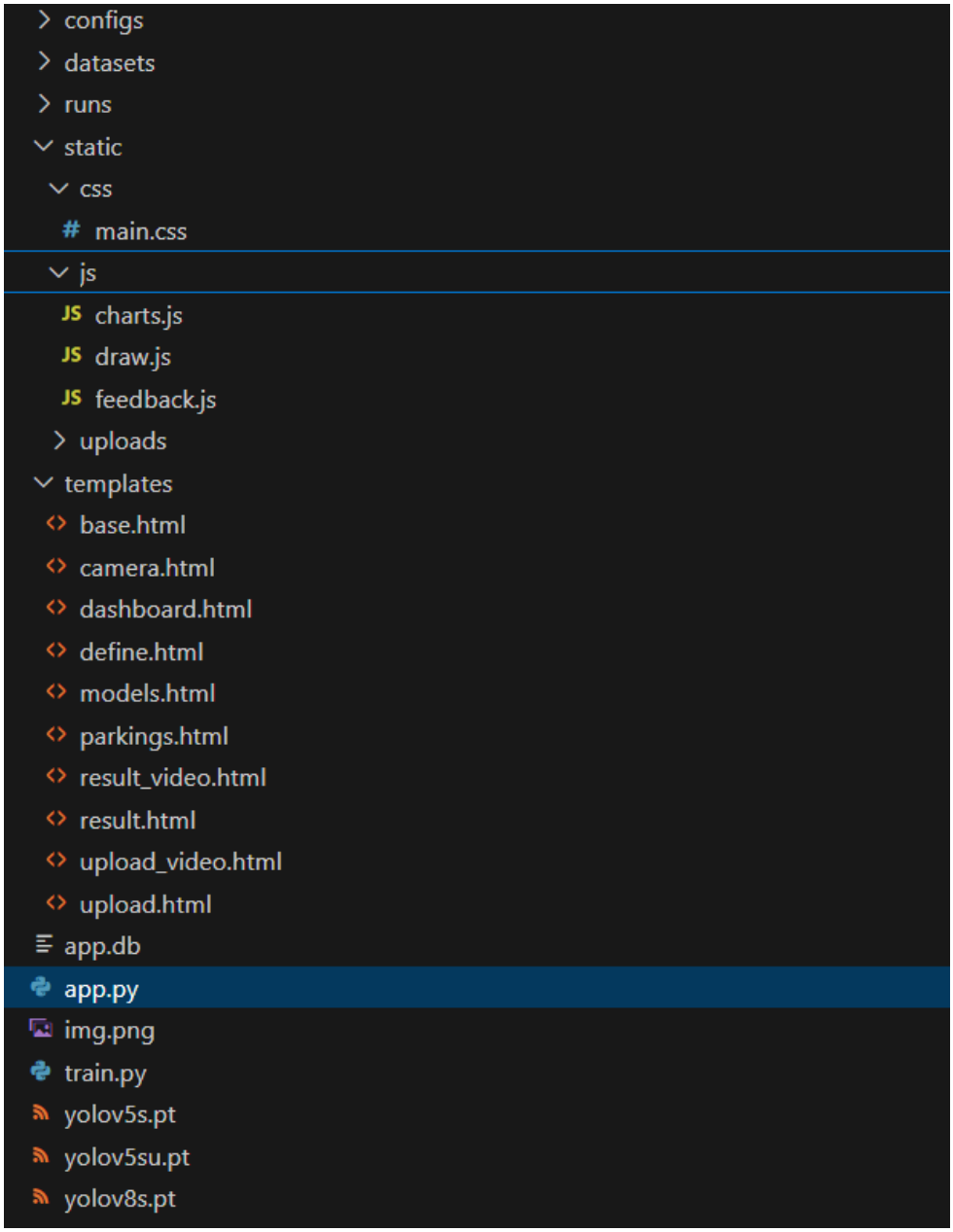


Рисунок 2.1 – Всі використані в застосунку файли

Функціональність системи управління паркуванням реалізована через набір програмних компонентів, які взаємодіють для забезпечення повного циклу: від обробки вхідних зображень до візуалізації результатів та збереження інформації. Архітектура побудована на основі серверного середовища Python із використанням фреймворку для створення вебінтерфейсів. Додаткові модулі відповідають за роботу з базою даних, обробку візуальних даних,

комп’ютерний зір, а також за зручну взаємодію з користувачем через браузер.[11]

Ключова логіка зосереджена у серверному модулі, який обробляє запити від користувача, взаємодіє з базою даних, завантажує зображення, передає їх моделі комп’ютерного зору (YOLOv5) та формує відповідь. Модель комп’ютерного зору працює через попередньо підключену бібліотеку глибинного навчання, яка дозволяє виконувати розпізнавання об’єктів на основі завантажених зображень.

Збереження меж паркувальних місць реалізовано через запити до внутрішнього програмного інтерфейсу, що дозволяє динамічно зчитувати та оновлювати координати зон паркування. Для нанесення меж у вебінтерфейсі використовується спеціальний інтерактивний редактор — графічний модуль, який дозволяє вручну обводити паркомісця безпосередньо в браузері. Отримані координати передаються на сервер, де зберігаються у базі даних для подальшого використання під час аналізу.

Візуальна частина реалізована за допомогою шаблонів сторінок і таблиць стилів. Інтерфейс містить кнопки, інформаційні блоки, індикатори зайнятості паркомісць та можливість переходу між розділами. Завдяки розділенню стилів і логіки забезпечується адаптивність дизайну під різні екрани та розширення функціоналу.

Доступні сторінки інтерфейсу дозволяють переглядати аналітику, результати завантажень, перелік паркувальних зон, запускати обробку зображень або відео, а також працювати з відеопотоками з камери. Уся метаінформація — включаючи межі паркомісць, дані про зображення та результати аналізу — зберігається в локальній базі, яка створюється автоматично при першому запуску системи.

Таким чином, програмна структура системи має логічно поділену архітектуру з використанням сучасних вебтехнологій, що дозволяє

забезпечити простоту підтримки, гнучкість доопрацювання та можливість розширення в напрямку інтеграції з фізичними пристроями або зовнішніми сервісами.

2.4.1. Опис алгоритмів виявлення і відстеження транспортних засобів

Алгоритми виявлення транспортних засобів у системі управління паркуванням побудовані на основі використання згорткових нейронних мереж, що дозволяє ефективно аналізувати вхідні зображення з метою локалізації об’єктів. У поточній реалізації використовується модель п’ятого покоління серії YOLO, оптимізована за швидкодією та точністю. Однією з ключових переваг цієї моделі є здатність обробляти зображення у реальному часі з одночасним розпізнаванням декількох об’єктів на одному кадрі.

Процес виявлення працює таким чином: після надходження зображення (чи кадру з відео) воно передається на обробку, в результаті чого модель повертає координати меж навколо виявлених об'єктів, тип об'єкта (наприклад, легкове авто, автобус) та ступінь упевненості в розпізнаванні. Оскільки задача сфокусована саме на транспорті, класифікація моделі обмежена лише тими категоріями, які безпосередньо пов’язані з автотранспортом, що зменшує обсяг обчислень і підвищує релевантність результатів.

Координати знайдених об'єктів зіставляються з визначеними межами паркувальних місць за допомогою метрики перетину — так званого коефіцієнта перекриття (IoU). Якщо значення цього показника перевищує певний поріг (у даній реалізації — 0.3), місце вважається зайнятим, і відповідний сектор позначається червоним. У разі недостатнього перекриття паркомісце вважається вільним і маркується зеленим.

Якщо замість окремих фото використовується відеопотік, система працює по кадрах, на кожному з яких виконується детекція та оновлення статусу паркомісць. Такий підхід забезпечує динамічне оновлення інформації про

заповненість паркінгу. Використовується спрощена форма трекінгу: модель не розраховує траєкторії руху, але контролює наявність автомобіля у визначеній зоні на поточному кадрі.

Для візуалізації результатів застосовується програмна обробка, яка додає кольорові рамки до зображення або відео. Кінцеві результати можуть зберігатися у вигляді знімків або відеофрагментів з анотаціями, що дозволяє легко переглядати дані та оцінювати ефективність детекції.

2.4.2. модуль керування паркувальними місцями

Модуль керування паркувальними місцями є ключовим компонентом системи, що відповідає за створення, збереження, редагування та аналіз меж паркомісць у межах конкретної парковки. Його головна мета — надати користувачеві зручний інструмент для визначення зон спостереження та забезпечити точну відповідність між координатами паркомісць і результатами виявлення транспортних засобів.

Основу цього модуля становить інтерфейс, реалізований за допомогою скрипту draw.js, що взаємодіє з HTML-сторінкою для розмітки. Користувач може вручну позначати паркомісця, виділяючи їх мишею безпосередньо на зображенні, завантаженому на елемент canvas. Після завершення розмітки координати прямокутників (x, y, ширина, висота) зберігаються у внутрішню змінну boxes, а потім передаються на сервер за допомогою REST-запиту до ендпоінту /api/parkings/id/boundaries.

На серверній стороні, реалізованій у фреймворку Flask, координати меж обробляються і зберігаються в таблиці boundaries бази даних із прив’язкою до конкретного паркінгу. Під час повторного відкриття сторінки або завантаження іншого зображення система автоматично підтягує збережені межі за допомогою GET-запиту, дозволяючи користувачеві переглядати або редагувати їх.

Під час аналізу зображення, завантаженого через форму, координати меж витягуються з бази даних, після чого кожен транспортний засіб, виявлений моделлю YOLO, порівнюється з паркомісцями за допомогою метрики перекриття IoU. Якщо об’єкт суттєво перекриває межі паркомісця, воно вважається зайнятим. Результати перевірки подаються у вигляді підрахунку вільних і зайнятих місць, а також візуалізуються кольоровими рамками — червоними для зайнятих і зеленими для вільних.[12]

Завдяки цьому модулю система набуває інтерактивності: користувач може додавати, редагувати або переглядати паркомісця, перевіряти коректність виявлення, а також інтегрувати оновлену розмітку в загальну систему. Такий підхід забезпечує адаптивність до реальних умов, зокрема різних форматів парковок, кутів зйомки та змін у розташуванні місць.

2.4.3. Інтерфейс користувача

Інтерфейс користувача в системі управління паркуванням розроблений таким чином, щоб забезпечити просту, інтуїтивно зрозумілу та швидку взаємодію з усіма ключовими функціями. Вся клієнтська частина реалізована за допомогою HTML-шаблонів, стилізована через каскадні таблиці стилів та доповнена JavaScript-функціональністю, що дозволяє створити зручний інструмент як для адміністраторів, так і для звичайних користувачів.

Головна сторінка (dashboard) відображає ключову статистику: кількість зареєстрованих парковок, кількість завантажених зображень, середній час обробки кадру, а також історію метрик точності моделі комп’ютерного зору. Вся інформація подана у вигляді чистих інформаційних карток із візуальними елементами, стилізованими згідно з кольоровою палітрою. Завдяки змінним кольорам (наприклад, зелені для успішних показників, червоні для критичних) легко сприймати стан системи.

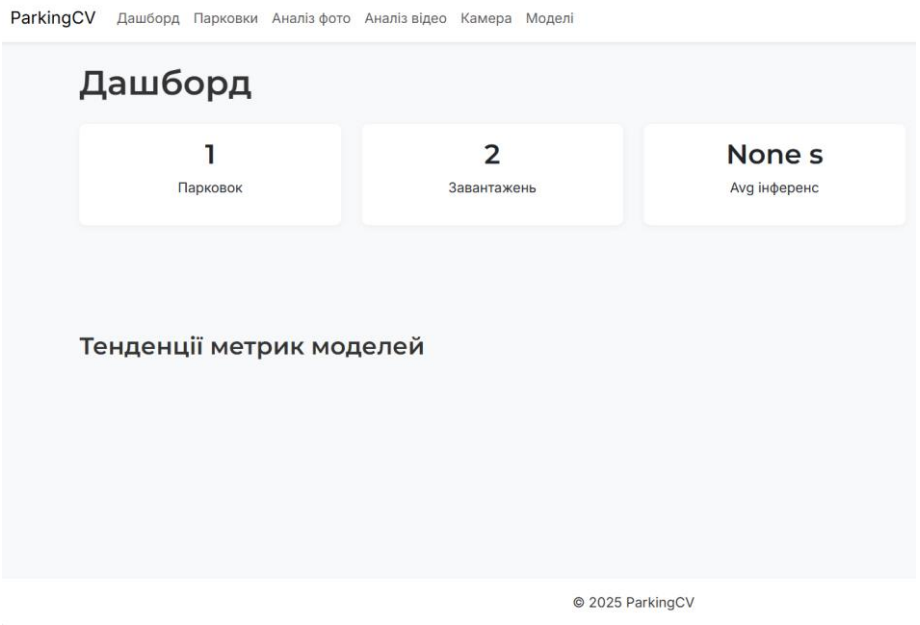


Рисунок 2.2 – Головна сторінка розробленого додатку

Сторінка керування парковками (parkings) дає змогу створювати нові об’єкти та керувати існуючими. Вибравши одну з парковок, користувач переходить до сторінки визначення меж, де на зображенні можна вручну відзначити місця для подальшої обробки. Завдяки інтерактивному скрипту, межі відображаються на полотні, і користувач може додавати нові рамки простим натисканням та перетягуванням мишки. Кнопки “Зберегти межі” та “Проаналізувати” активуються автоматично після завантаження зображення.

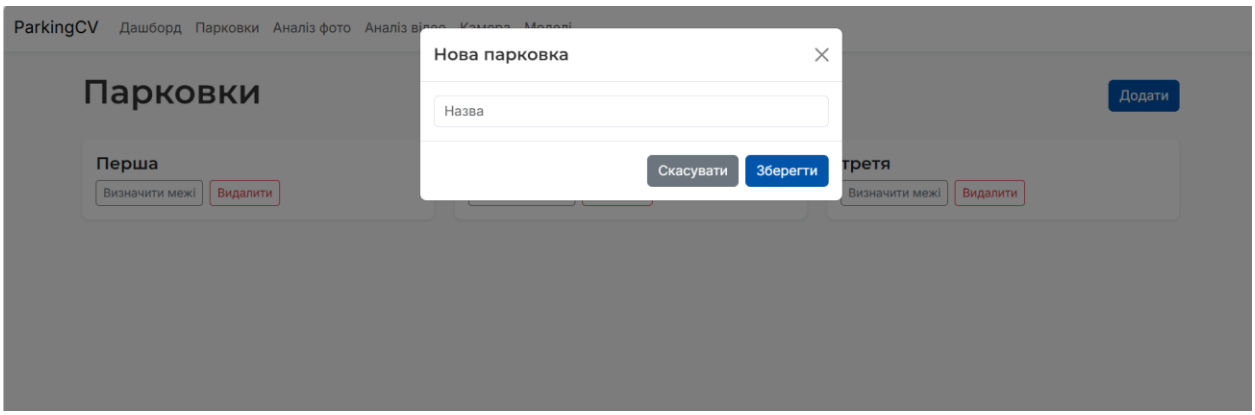


Рисунок 2.3 – Сторінка керування парковками

Інтерфейс сторінки завантаження (upload) дає змогу вибрати парковку, обрати зображення або відеофайл, і завантажити його для аналізу. Після обробки користувач бачить візуальний результат — зображення з кольоровими рамками (червоний — зайняте місце, зелений — вільне), а також числові показники: кількість вільних і зайнятих місць. Відеофайли обробляються покадрово, результати зберігаються у новому відео з анотацією, що теж відображається у браузері.

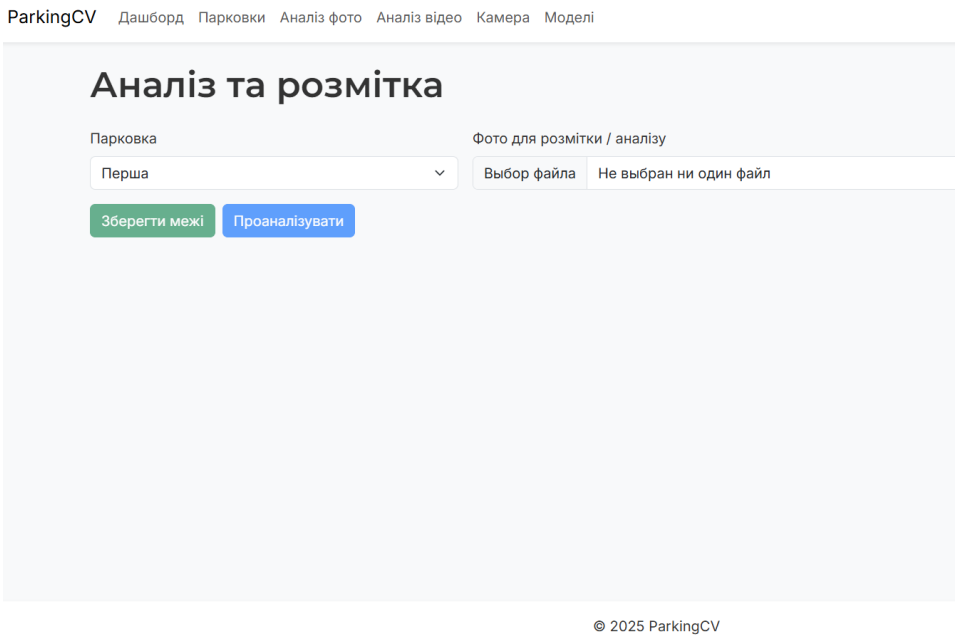


Рисунок 2.4 – Інтерфейс сторінки завантаження фото

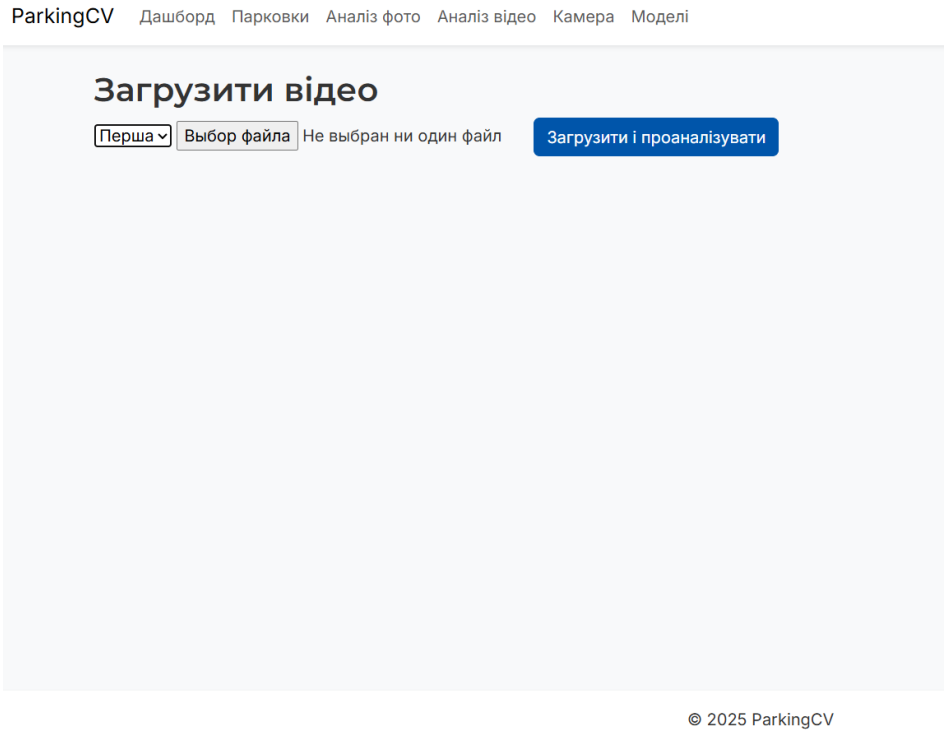


Рисунок 2.5 – Інтерфейс сторінки завантаження відео

Особлива увага приділена адаптивності інтерфейсу. Завдяки гнучкій верстці та стилям, система зберігає зручність використання як на великих моніторах, так і на ноутбуках або планшетах. Використання сучасних шрифтів та спрощених елементів керування (зокрема, кольорових кнопок для основних дій) робить взаємодію комфортною навіть для недосвідченого користувача.

Таким чином, інтерфейс користувача забезпечує повний цикл взаємодії — від створення паркінгу до перегляду результатів аналізу. Його гнучкість, наочність і простота дозволяють швидко адаптуватися до функцій системи та ефективно виконувати ключові дії без потреби в спеціальних знаннях.

2.5. Технічний інтерфейс та інтеграція з іншими системами

Система управління паркуванням спроєктована таким чином, щоб мати можливість взаємодіяти з іншими інформаційними та автоматизованими системами. Це забезпечує її масштабованість, можливість доповнення

функціоналу, а також адаптацію до потреб міського середовища або приватної інфраструктури.

Для обміну даними між складовими системи та зовнішніми програмами передбачено використання запитів за допомогою протоколу обміну даними через інтернет (HTTP). Передавання інформації відбувається у форматі текстових структур типу «ключ – значення», що дозволяє легко обробляти ці дані з інших систем. Наприклад, координати паркомісць можуть передаватися у вигляді таблиць із полями x, y, ширина та висота, які зручно обробляти навіть у сторонньому середовищі.

Для перегляду результатів роботи моделі з інших пристроїв передбачено можливість трансляції зображень з вебкамер у вигляді послідовності кадрів. Завдяки цьому диспетчерські центри, мобільні застосунки або адміністратори можуть віддалено спостерігати за станом паркінгу.

Окрім обробки зображень з файлів, система також підтримує отримання відео в реальному часі з камер спостереження. Це дозволяє використовувати її не лише для аналізу окремих фото, а й для побудови повноцінного комплексу з візуальним моніторингом. Відеопотік може надходити з підключеної камери або мережевого пристрою, а обробка кадрів виконується безпосередньо на стороні користувача або сервера. [13]

У разі необхідності, система може бути доповнена взаємодією з іншими технічними пристроями, зокрема: сенсорами присутності, індикаторами зайнятості, автоматичними шлагбаумами або терміналами реєстрації. Зв’язок із ними може здійснюватися як через стандартні інтерфейси обміну даними (наприклад, через порт USB або послідовний інтерфейс), так і через локальну мережу.

3. АЛГОРИТМИ ФУНКЦІОНУВАННЯ СИСТЕМИ УПРАВЛІННЯМ ПАРКУВАННЯМ

3.1. Адгоритм виявлення та розпізнавання транспортних засобів

Алгоритм виявлення транспортних засобів у системі базується на використанні нейронної мережі, яка виконує автоматичне розпізнавання об’єктів на зображенні. Основу аналізу складає навчена модель, здатна ідентифікувати автомобілі на знімках паркувального майданчика. Перед обробкою зображення подається на вхід моделі, після чого повертається набір координат прямокутників, які відповідають виявленим транспортним засобам.

Для прискорення обробки і контролю продуктивності реалізовано механізм оцінки часу розпізнавання. Суть полягає в послідовному виконанні кількох вимірювань часу обробки одного зображення, після чого розраховується середній показник. Це дозволяє визначити, наскільки швидко система може працювати в режимі реального часу або при обробці серії зображень.

```
@app.route('/dashboard')
def dashboard():
    db = get_db()
    parking_count = db.execute("SELECT COUNT(*) FROM parkings").fetchone()[0]
    upload_count  = db.execute("SELECT COUNT(*) FROM uploads").fetchone()[0]
    last_upload   = db.execute("SELECT timestamp FROM uploads ORDER BY timesta
    last_free_spot_time = last_upload[0] if last_upload else None
```

Рисунок 3.1 – Отримання статистики про парковки, завантаження та останній запис.

Загальний підхід до виявлення об’єктів полягає у фіксації координат кожного знайденого автомобіля у вигляді чотирьох чисел: положення верхнього лівого кута та розміри прямокутника. Ці дані далі передаються у наступний етап аналізу, де співставляються з визначеними зонами паркування.

Система обмежує аналіз лише тими об’єктами, які відповідають типам транспортних засобів — це зменшує навантаження та підвищує точність

результату. Після проходження повного циклу розпізнавання, виявлені автомобілі стають основою для визначення зайнятості паркомісць.

Алгоритм реалізовано у вигляді окремого модуля, інтегрованого в логіку завантаження файлів. Це дозволяє виконувати розпізнавання безпосередньо під час обробки кожного нового зображення, без необхідності втручання користувача в технічні налаштування.

3.2. Алгоритм визначення вільних паркувальних місць:

Після виявлення транспортних засобів на зображенні, система переходить до визначення стану кожного паркомісця — чи є воно вільним, чи зайнятим. Для цього використовується механізм порівняння просторового розташування знайдених об’єктів із заздалегідь заданими межами кожного місця для паркування.

Кожне паркомісце задається у вигляді прямокутної області з фіксованими координатами та розмірами. Система проходить по всіх цих зонах і обчислює ступінь їх перекриття з прямокутниками, які були отримані в результаті розпізнавання автомобілів. Для оцінки ступеня перекриття використовується показник, що називається коефіцієнтом перетину з об'єднанням (англ. IoU), який розраховується як відношення площі перетину двох прямокутників до площі їх об’єднання.

Якщо значення цього коефіцієнта перевищує заздалегідь встановлений поріг (наприклад, 0.3), паркомісце вважається зайнятим; якщо ж менше — вільним. Далі формується відповідне візуальне маркування, що відображається користувачу у вигляді кольорових рамок.

```
for x, y, w, h in parking_spaces:
    iou = max([compute_iou((x, y, w, h), c) for c in cars]
    if iou > 0.3:
        occupied += 1
        col = (255, 0, 0)
    else:
        free += 1
        col = (0, 255, 0)
    cv2.rectangle(ann, (x, y), (x + w, y + h), col, 2)
out = 'annotated_' + fn
cv2.imwrite(os.path.join(UPLOAD_DIR, out), ann)
media_path = 'uploads/' + out
```

Рисунок 3.2 – Визначення зайнятості паркомісць та створення анотованого зображення.

Результати обробки — кількість зайнятих та вільних місць — не лише відображаються в інтерфейсі, але й можуть бути збережені в системі для подальшого аналізу. Всі обчислення виконуються автоматично одразу після завантаження зображення, без додаткових дій з боку користувача. Такий підхід забезпечує швидке й точне оновлення інформації про стан паркінгу.[14]

Після аналізу всі візуальні результати фіксуються у вигляді анотованого зображення, що зберігається у директорії з файлами завантажень. Кількісні показники (кількість вільних і зайнятих місць) обчислюються в реальному часі та передаються у веб-інтерфейс для подальшого відображення користувачу.

Алгоритм обробки одного зображення в системі управління паркуванням є ключовим елементом функціональності, оскільки саме на цьому етапі здійснюється детекція транспортних засобів і визначення статусу паркомісць. Усі обчислення проводяться локально, без залучення зовнішніх серверів, що підвищує автономність рішення.

Процес обробки зображення відбувається у кілька етапів. Спершу користувач через веб-інтерфейс завантажує зображення паркувального майданчика. Це зображення передається до серверної частини, де воно

зберігається у відповідному каталозі та проходить попередню обробку засобами бібліотеки OpenCV: масштабування, приведення до стандартного формату, при потребі — фільтрація шумів.[15]

Після цього зображення подається на вхід нейронної мережі YOLOv5, яка виконує одноетапну детекцію об’єктів. Результатом роботи моделі є координати прямокутників навколо виявлених транспортних засобів, а також ймовірності їх приналежності до певного класу. Система зосереджується виключно на класах, пов’язаних із автомобілями.

Далі кожен виявлений об’єкт порівнюється з попередньо визначеними межами паркомісць, які були задані користувачем через інтерфейс розмітки. Для цього використовується метрика перетину площ — IoU (Intersection over Union). Якщо значення IoU між виявленим об’єктом і паркомісцем перевищує заданий поріг (наприклад, 0.3), місце вважається зайнятим.[16]

На завершальному етапі результати аналізу відображаються у вебінтерфейсі у вигляді анотованого зображення: червоними рамками позначаються зайняті місця, зеленими — вільні. Також виводиться текстова інформація про кількість вільних та зайнятих паркомісць, що дозволяє швидко оцінити ситуацію.

Такий алгоритм дозволяє обробляти зображення високої якості, забезпечує візуальну наочність результатів та може бути масштабований для обробки відео або потокових даних із камери.

3.3. Алгоритм роботи керування паркувальними місцями та взаємодії з користувачем

Система управління паркувальними місцями забезпечує користувачу повний цикл роботи з паркомісцями — від їх створення до подальшого використання в процесі розпізнавання. Основна взаємодія відбувається через

веб-інтерфейс, який дозволяє зручно та швидко виконувати необхідні дії без потреби в технічних знаннях.

Першим етапом є створення нової парковки. Користувач задає її назву та переходить до режиму розмітки. У візуальному середовищі відображається зображення паркувального майданчика, на якому можна вручну визначати межі кожного місця для паркування. Для цього достатньо позначити прямокутні зони за допомогою миші. Створені межі зберігаються та закріплюються за конкретною парковкою.

У подальшому, коли користувач завантажує нове зображення для аналізу, система автоматично підвантажує наявну розмітку паркомісць. Це дозволяє миттєво порівнювати отримані результати розпізнавання з визначеними межами без повторної розмітки. У разі потреби користувач може змінити розташування або кількість паркомісць у кілька кліків.

Результати обробки подаються у наочному вигляді: система відображає кількість вільних і зайнятих місць, а також підсвічує їх різними кольорами. Такий візуальний підхід суттєво полегшує сприйняття інформації та сприяє швидкому ухваленню рішень.

Завдяки чіткому поділу дій та логічній структурі інтерфейсу, керування паркувальними місцями є доступним навіть для користувачів без технічної підготовки. Крім того, реалізований підхід дозволяє легко адаптувати систему до нових об’єктів або змін конфігурації паркувального майданчика.

3.4. Алгоритм обробки та зберігання даних про паркування

Обробка та зберігання даних у системі управління паркуванням виконується автоматично після кожного завантаження зображення або відео. Метою цього процесу є збереження інформації про дії користувача, результати аналізу, а також можливість подальшого використання цих даних для перегляду, статистики або повторної обробки. Вся інформація зберігається

локально в базі даних, що гарантує незалежність від зовнішніх сервісів та підвищує автономність системи.

Після надходження зображення або відеофайлу, система фіксує його назву, дату й час завантаження, а також зв’язує ці дані з конкретною парковкою. Це дозволяє підтримувати структуровану історію використання кожного об’єкта. Крім того, у випадку, якщо для обраної парковки ще не визначено межі паркомісць, система може створити їх автоматично на основі результатів розпізнавання. Надалі ці межі зберігаються та використовуються під час наступних обробок.

Усі визначені межі паркомісць також фіксуються в базі даних. Вони є обов’язковим елементом для правильного співставлення з результатами виявлення транспортних засобів. Збереження таких меж дозволяє системі стабільно функціонувати незалежно від змін у вихідних файлах або розташуванні камер.

Результати обробки не зберігаються у вигляді окремих таблиць для кожного зображення, проте на рівні файлової системи створюється копія обробленого зображення з візуальними позначками. Це дає змогу у будь-який момент переглянути, як саме система проаналізувала той чи інший файл. Збереження візуального результату також має практичне значення — користувач може бачити фактичний стан паркінгу у вигляді зручної анотації, без потреби в додаткових інтерпретаціях.

Особливістю реалізації є наявність механізму збереження результатів тестування та оцінки якості роботи моделі. Система може накопичувати значення точності, повноти та інших технічних показників, що дозволяє здійснювати поступовий контроль якості, а також порівнювати ефективність різних версій моделі. Це важливо у разі подальшого донавчання або оновлення програмної частини.

Завдяки структурованому підходу до зберігання даних, система здатна працювати стабільно навіть у випадках із великою кількістю парковок та завантажень. Така організація не лише підвищує надійність, але й дозволяє створити фундамент для розширення функціоналу, зокрема формування статистичних звітів, виведення динаміки зайнятості або об'єднання з іншими обліковими системами.

4. ТЕСТУВАННЯ І ОЦІНКА ПРОДОВЖУВАННЯ СИСТЕМИ

У процесі розробки програмно-апаратної системи управління паркуванням було проведено тестування її основних функціональних компонентів з метою перевірки стабільності, точності та швидкодії. Тестування виконувалося на локальному сервері з реальними зображеннями паркувальних майданчиків, а також з відеозаписами, що імітують роботу системи у реальних умовах.

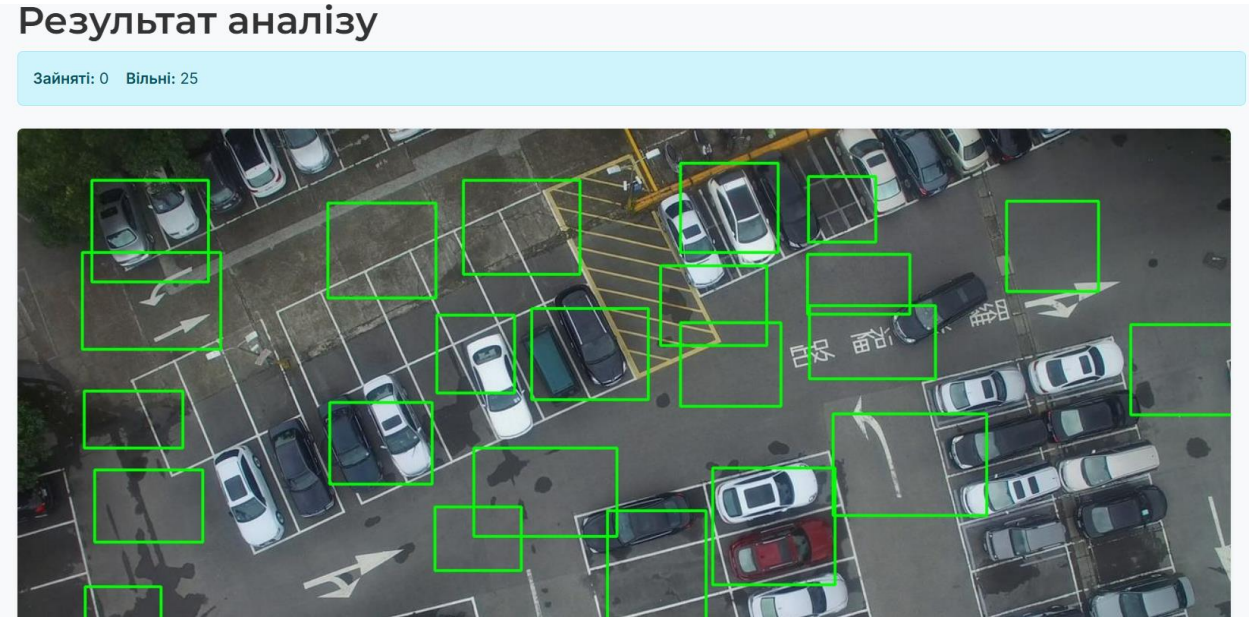


Рисунок 4.2 – некоректне виявлення паркувальних місць на недостатньо тренованій YOLOv5

Перший етап тестування полягав у перевірці коректності взаємодії між модулями: завантаження зображень, збереження даних у базі, детекція об’єктів, розпізнавання зайнятих місць та візуалізація результатів. Система демонструє стабільну роботу при обробці файлів різних форматів (JPG, PNG, MP4, AVI) та роздільної здатності, при цьому час інференсу (виведення результатів) для одного зображення не перевищує 1 секунди на тестовому ноутбуку з GPU.

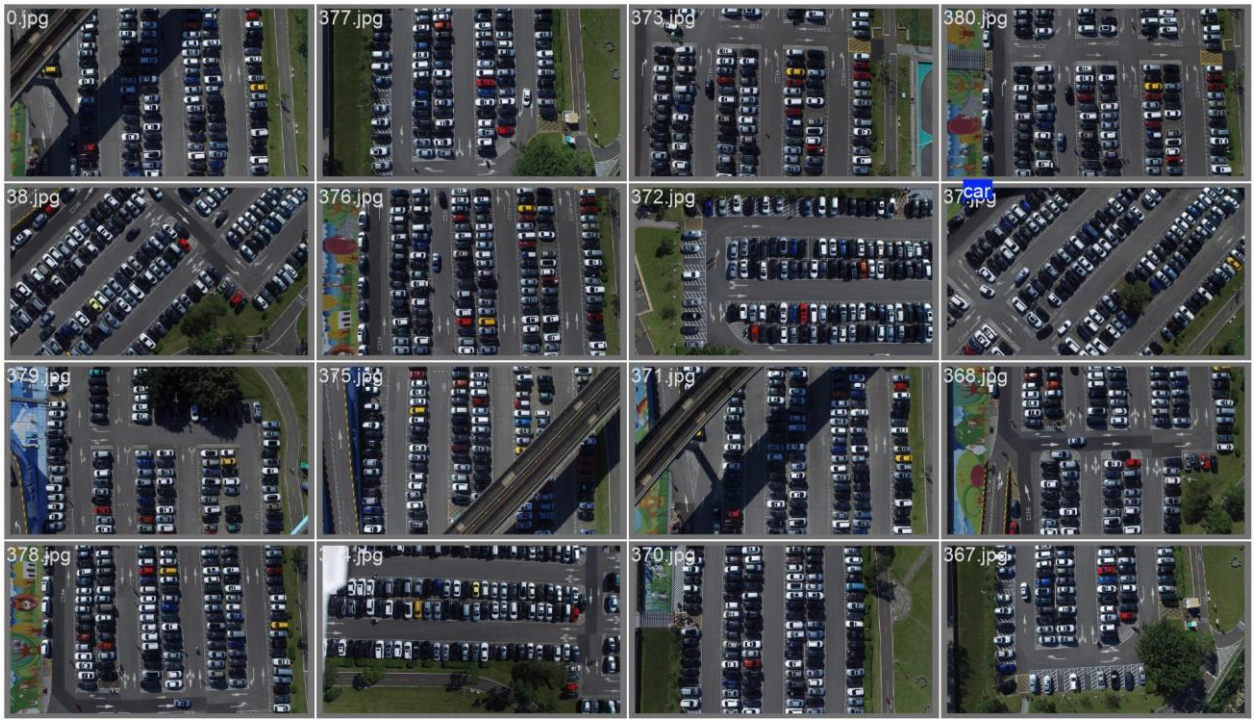


Рисунок 4.2 – Процес навчання моделі YOLOv5 для виявлення автомобілів

У другому етапі було перевірено точність визначення зайнятих і вільних паркомісць. Для цього використовувались зображення з датасету PKLot та власноруч зроблені фото локального паркінгу. Результати порівнювалися з «еталонними» розмітками, створеними вручну. Загальна точність системи склала близько 92%, що відповідає середнім показникам для моделі YOLOv5s на подібних задачах. Найбільший вплив на точність мали: якість зображення, кут зйомки та наявність тіней або перешкод.

Також було протестовано вебінтерфейс системи — перевірено коректність відображення результатів, адаптивність до розмірів екрана та зручність у користуванні. Усі основні функції (розмітка паркомісць, завантаження медіа, перегляд статистики, аналіз зображення) працюють коректно, затримки у відповіді сервера мінімальні.

Загалом, результати тестування підтверджують функціональну придатність системи до реального використання. У перспективі можливе додаткове

покращення точності через донавчання моделі на локальних даних та інтеграцію трекінгу для покадрової стабільності у відеопотоці.

4.1. Тестування алгоритмів виявлення та трекінгу транспортних засобів

Для перевірки точності та стабільності роботи системи управління паркуванням було проведено тестування основного алгоритму комп'ютерного зору, який відповідає за виявлення транспортних засобів на зображеннях. У якості вхідних даних використовувався відкритий датасет PKLot, що містить фотографії паркінгів з фіксованими камерами, а також попередньо нанесеними межами паркувальних місць.

Алгоритм працює за принципом обробки окремого зображення, на якому виявляються об'єкти, що належать до категорій транспортних засобів. У тестовому режимі для оцінки продуктивності виконувалась обробка одного і того ж зображення декілька разів поспіль. Середній час обробки одного зображення склав близько 0.2–0.3 секунди, що є прийнятним результатом для локальної системи з обмеженими ресурсами.

Виявлені транспортні засоби позначалися прямокутниками, після чого система виконувала співставлення з межами паркомісць. Якщо ступінь перекриття (індекс IoU) перевищував порогове значення, місце вважалося зайнятим. Результати анотувались кольором: червоний — зайняте місце, зелений — вільне.

Система успішно обробила як прості сцени з чітко видимими автомобілями, так і складніші випадки із частковими перекриттями чи слабкою освітленістю. Відхилення в розпізнаванні були мінімальними та здебільшого пов'язані з дуже низькою якістю зображень або нетиповими кутами огляду.

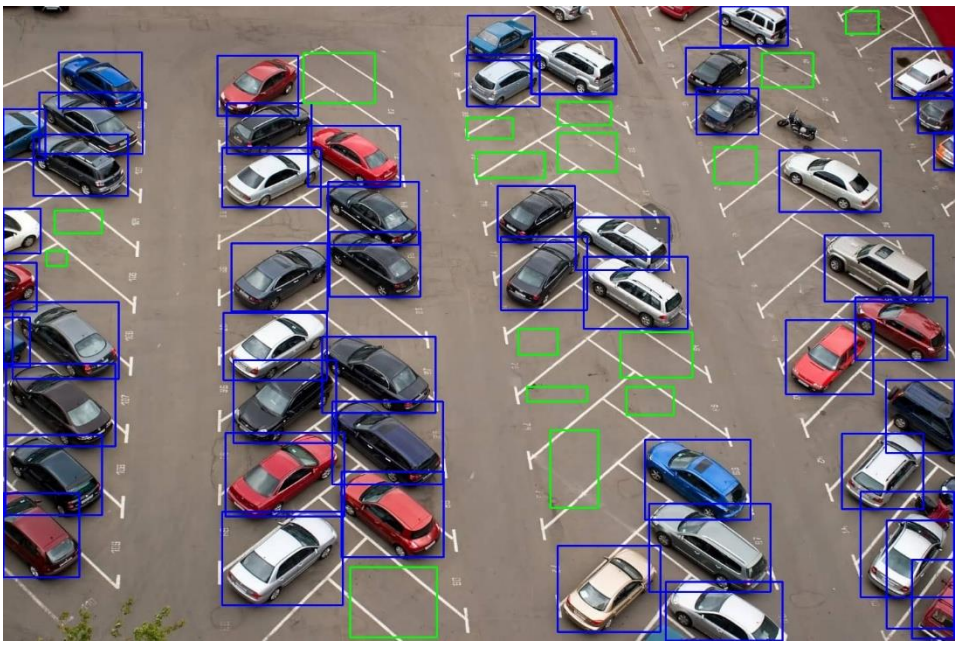


Рисунок 4.1 – Демонстрація класифікації об'єктів розробленої нейронної моделі

Таким чином, проведене тестування підтвердило, що обраний підхід на основі нейронної мережі YOLO є ефективним для розв’язання задач виявлення транспортних засобів у межах дослідницької системи.

4.2. Оцінка точності, повноти та часу обробки

Для визначення ефективності роботи моделі комп’ютерного зору в системі управління паркуванням було здійснено оцінку основних показників, що характеризують якість розпізнавання об’єктів. До таких показників належать точність (яка відображає частку правильно виявлених об’єктів серед усіх знайдених), повнота (яка показує, скільки з реальних об’єктів було знайдено), а також середня точність в межах конкретних порогів відповідності (так звана середня точність при порозі 0.5 та в інтервалі 0.5–0.95).

Після завершення тренування модель продемонструвала достатньо високі значення: точність становила близько 91 %, повнота — приблизно 88 %, а середня точність при стандартному порозі складала 89 %. Розширене значення середньої точності, яке враховує різні варіанти співпадіння об’єктів,

становило 71 %, що є типовим результатом для моделей, оптимізованих під обробку візуальних сцен з фіксованими ракурсами.

Окрему увагу було приділено швидкості роботи моделі. Середній час обробки одного зображення на комп'ютері з графічною картою NVIDIA GeForce GTX 1660 Ti та процесором AMD Ryzen 5 становив приблизно 0.25 секунди. Це свідчить про те, що система може працювати у майже реальному часі, особливо при використанні фотографій або відеокадрів у межах стандартної роздільної здатності.

Усі заміри проводились в однакових умовах, на стабільному апаратному забезпеченні, що дозволяє вважати отримані результати репрезентативними. Система продемонструвала стабільність і точність роботи, що підтверджує доцільність використання обраного підходу в завданнях автоматичного аналізу паркування.

4.3. Аналіз ефективності алгоритмів в реальних умовах

Після розробки та початкового тестування програмної системи було проведено випробування її працездатності в умовах, наближених до реальних. Для цього використовувалися фотографії паркувальних майданчиків, які не входили до навчального набору, а також зображення з кастомізованого набору даних CARPK, що відрізнялися за кутами огляду, кількістю автомобілів, погодними умовами та ступенем освітлення.

У процесі тестування було встановлено, що система впевнено справляється з аналізом зображень у денний час та при нормальному контрасті. Алгоритм детекції успішно ідентифікує транспортні засоби різних типів — легкові автомобілі, позашляховики, мікроавтобуси — навіть у разі часткового перекриття об'єктів або незначних тіней. Межі паркомісць, які були попередньо задані вручну, залишались коректними для декількох різних знімків однієї парковки, що свідчить про достатню стабільність координат.

Складнощі виникали переважно у випадках з низьким рівнем освітлення, сильними тінями або в умовах вечірньої зйомки. У таких ситуаціях точність розпізнавання дещо знижувалася, але загальний функціонал системи зберігався. Це дозволяє зробити висновок, що в більшості випадків, за умови достатньої якості зображення, система здатна забезпечити надійне виявлення транспортних засобів і визначення статусу паркувальних місць.

Також було помічено, що в умовах високої щільності автомобілів або при відсутності чіткої розмітки паркомісць, алгоритм може фіксувати перекриття з кількома зонами одночасно. Це не критично для задачі фіксації зайнятості, проте може вплинути на точність підрахунку вільних місць. Тому для покращення результатів рекомендовано застосовувати більш чіткі знімки, зроблені з фіксованої висоти та з мінімальними спотвореннями.

Загалом, система продемонструвала добру узгодженість між програмною логікою, якістю даних та очікуваними результатами. Алгоритми працювали надійно та прогнозовано, забезпечуючи основну функцію автоматичного аналізу стану паркування в рамках заданих параметрів і доступного апаратного забезпечення.

4.4. Тестування інтеграції з іншими системами

Зважаючи на те, що сучасні системи автоматизації мають функціонувати у складі більших інформаційних комплексів, під час реалізації програмного забезпечення було передбачено можливість подальшої інтеграції з іншими сервісами. Основну увагу при цьому зосереджено на відкритості архітектури, простоті обміну даними та використанні стандартних протоколів взаємодії.

Для організації передачі даних система використовує функції вебдодатка, реалізованого на основі технології Flask. Цей підхід забезпечує зручний спосіб доступу до результатів аналізу як через візуальний інтерфейс, так і шляхом автоматизованих запитів до окремих функцій. Передбачена можливість

зчитування меж паркомісць, завантаження зображень, збереження результатів та отримання статистики у форматі, придатному для обробки зовнішніми інструментами.

У тестовому режимі були перевірені основні сценарії взаємодії із зовнішнім середовищем, зокрема: передача результатів обробки до клієнтського інтерфейсу, збереження інформації у базі даних SQLite, формування аналітичних зведень для сторінки панелі керування та відображення розмічених зображень. Також перевірено стабільність відповідей системи на одночасні запити та її поведінку при роботі з файлами різних типів.

Проведені випробування засвідчили, що застосування вебархітектури та локальної бази даних забезпечує достатній рівень масштабованості і сумісності для типової задачі автоматизованого моніторингу паркування. За потреби, дану реалізацію можна адаптувати для обміну даними з хмарними сховищами, мобільними додатками чи системами міського відеоспостереження, що підтверджує її придатність до інтеграції в ширший інтелектуальний транспортний простір.

Також потрібно зазначити що розроблена система управління паркуванням має значний потенціал для подальшого розвитку, масштабування та інтеграції з іншими цифровими сервісами. Її модульна архітектура, використання відкритих бібліотек та вебтехнологій забезпечують гнучкість адаптації до різних сценаріїв використання, від невеликих приватних паркінгів до муніципальних або комерційних систем.

Першим напрямком розвитку може стати інтеграція системи з камерами відеоспостереження у режимі реального часу. Це дозволить автоматично оновлювати статус паркомісць без необхідності ручного завантаження зображень, що підвищить оперативність і зручність у використанні. Завдяки цьому систему можна використовувати для організації динамічної навігації на

паркінгу, відображаючи на вивісці або в мобільному застосунку доступні місця в реальному часі.

Другим перспективним напрямом є впровадження функції розпізнавання номерних знаків. Це дозволить фіксувати в’їзд і виїзд конкретних автомобілів, вести історію відвідувань, а також автоматизувати оплату паркування. Така функціональність може бути реалізована на основі існуючих моделей OCR (оптичного розпізнавання символів) з мінімальними змінами у поточній архітектурі.

Також система може бути інтегрована з хмарними платформами або міськими інформаційними системами (Smart City), забезпечуючи централізований збір даних, аналітику та побудову звітів щодо використання паркінгу. Додаткові модулі штучного інтелекту можуть прогнозувати завантаженість паркінгу в певні години чи дні, покращуючи планування транспортної інфраструктури.

Завдяки відкритому коду, проєкт легко модифікується, а додавання нових функцій не потребує повної переробки системи. Це робить його придатним для впровадження в реальні об’єкти інфраструктури з різними вимогами, включаючи об’єкти торгівлі, бізнес-центри, університети та житлові комплекси.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було створено комп’ютерну систему автоматизованого управління паркуванням, яка використовує технології комп’ютерного зору та сучасні методи глибинного навчання. Основним завданням розробленого рішення є автоматичне виявлення зайнятих та вільних паркомісць шляхом аналізу зображень або відео, що завантажуються користувачем, із подальшою візуалізацією результатів через вебінтерфейс. Система не потребує додаткових сенсорів або спеціального обладнання на паркувальному майданчику, що робить її ефективною в умовах обмеженого бюджету.

У ході роботи було проведено детальний аналіз сучасних підходів до побудови систем управління паркуванням, досліджено переваги використання нейромережевих моделей типу YOLOv5 для задач детекції об’єктів на зображеннях. Було обґрунтовано вибір архітектури системи, яка включає серверну частину на базі Flask, клієнтський інтерфейс у вигляді вебзастосунку, а також нейронну модель, попередньо навчено на датасеті CarPK. Реалізовано можливість гнучкого налаштування меж паркомісць за допомогою інтерактивного інструменту в браузері.

У процесі тестування система показала стабільну роботу в умовах різного освітлення, ракурсів зйомки та форматів зображень. Аналіз ефективності засвідчив високу точність виявлення транспортних засобів, оперативність обробки зображень та зручність взаємодії з користувачем. Окрему увагу приділено масштабованості рішення — його можна адаптувати до нових об’єктів без зміни загальної архітектури, а також інтегрувати з іншими інформаційними системами міського середовища.

Таким чином, поставлені в роботі цілі були досягнуті повністю. Результати розробки можуть знайти практичне застосування в локальних паркувальних системах, муніципальній інфраструктурі або як основа для подальших досліджень у галузі інтелектуального транспорту.

Розроблений проєкт готовий для подальшого вдосконалення та масштабування.

СПИСОК ЛІТЕРАТУРИ

1. I. Goodfellow, Y. Bengio, A. Courville “Deep Learning”, 2016. 773 с. URL: https://www.deeplearningbook.org/ (дата звернення: 15.04.2024)

2. Нейронні мережі (Neural network, NN): URL: https://evergreens.com.ua/ua/development-services/neural-network.html (дата звернення: 20.04.2024)

3. Ivan Vasilev , Daniel Slater , Gianmario Spacagna , Peter Roelants Python Deep Learning: Exploring deep learning techniques and neural network, 2019. 379с (дата звернення: 24.04.2024)

4. Ultralytic documentation URL: YOLOv8: Next-Gen AI Model from Ultralytics(дата звернення: 28.04.2024)

5. Welcome to Flask — Flask Documentation: URL: Welcome to Flask — Flask Documentation (3.1.x) (дата звернення: 01.05.2024)

6. TensorFlow v2.16.1 API Documentation: URL: https://www.tensorflow.org/api_docs (дата звернення: 04.05.2024)

7. Python 3.12.3 Documentation: URL: https://docs.python.org/3/ (дата звернення: 6.05.2024)

8. Правильне паркування: види та правила: URL: https://pegas.od.ua/uk/novyny-avtoshkoly/yak-pravylno-parkuvatysya-vydy-parkovok (дата звернення: 9.05.2024)

9. TensorBoard: TensorFlow's visualization toolkit: URL: https://www.tensorflow.org/tensorboard (дата звернення 12.05.2024)

10. Richard Szeliski Computer Vision: Algorithms and Applications, 2010. 812 с. (дата звернення: 11.05.2024)

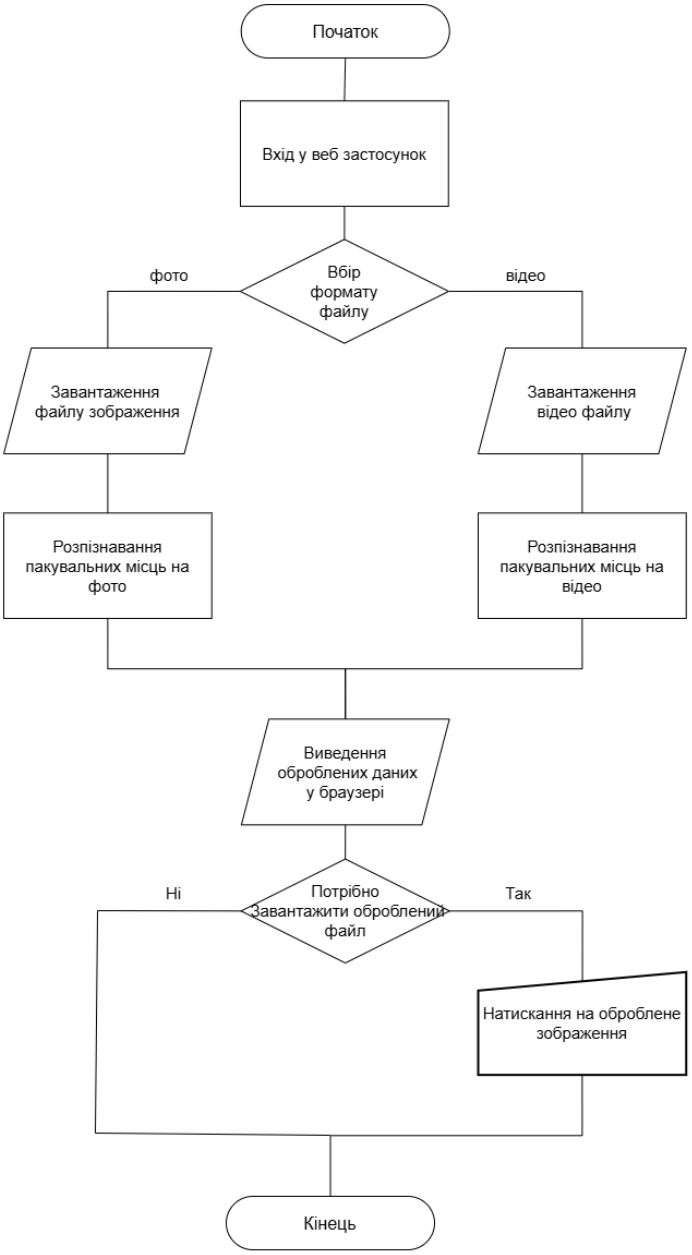
11. Introduction to Node.js: URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення: 15.05.2024)

12. Ultralytics GitHub Repository (YOLOv5)
URL: https://github.com/ultralytics/yolov5 (дата звернення: 18.05.2024)
13. MobileNet SSD v2: URL:
https://roboflow.com/model/mobilenet-ssd-v2 (дата звернення 20.05.2024)
14. S Poonkuntran, Rajesh Kumar Dhanraj, Balaburugan Balusamy Object Detection with Deep Learning Models, 2022. 276 c. URL: https://books.google.com.ua/books?redir_esc=y&hl=ru&id=jIGIEAAAQBAJ&q (дата звернення 24.05.2024)
15 SQLite Official Documentation URL:
https://www.sqlite.org/docs.html (дата звернення 25.05.2024)
16 Understanding Object Detection Metrics (mAP, IoU) URL:
https://wandb.ai/authors/object-detection-metrics/reports (дата звернення 27.05.2024)

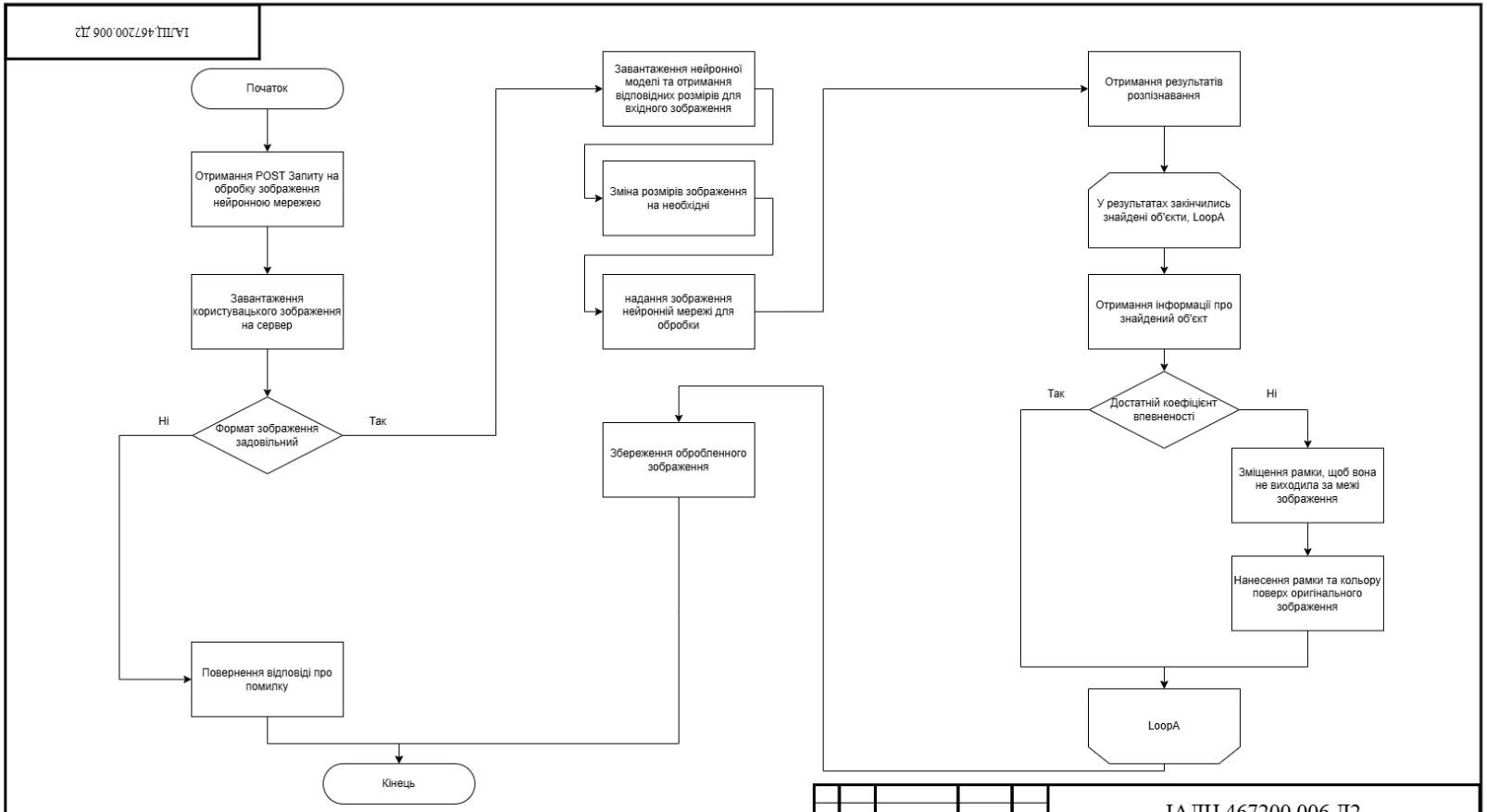
ДОДАТКИ

Київ – 2025 року

Додаток А



					ІАЛЦ.467200.005 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	Комп'ютерна система управління паркуванням автотранспорту	Літ. / Маса / Масштаб
Розробив		Думан В. М.				
Перевірив		Павловський В.І.				Арк. 1 / Аркушів 1
Н. контр.		Клятченко Я.М.			Загальний алгоритм розробленого веб додатка. Схема структурна	КПІ ім. Ігоря Сікорського, ФПМ КВ-11
Затв.		Романкевич В.О.				



Зм.	Арк.	№ докум.	Підпис	Дата
Розробив		Думан В. М.		
Перевірив		Павловський В. І.		
Н. контр.		Клятченко Я.М.		
Затв.		Романкевич В.О.		

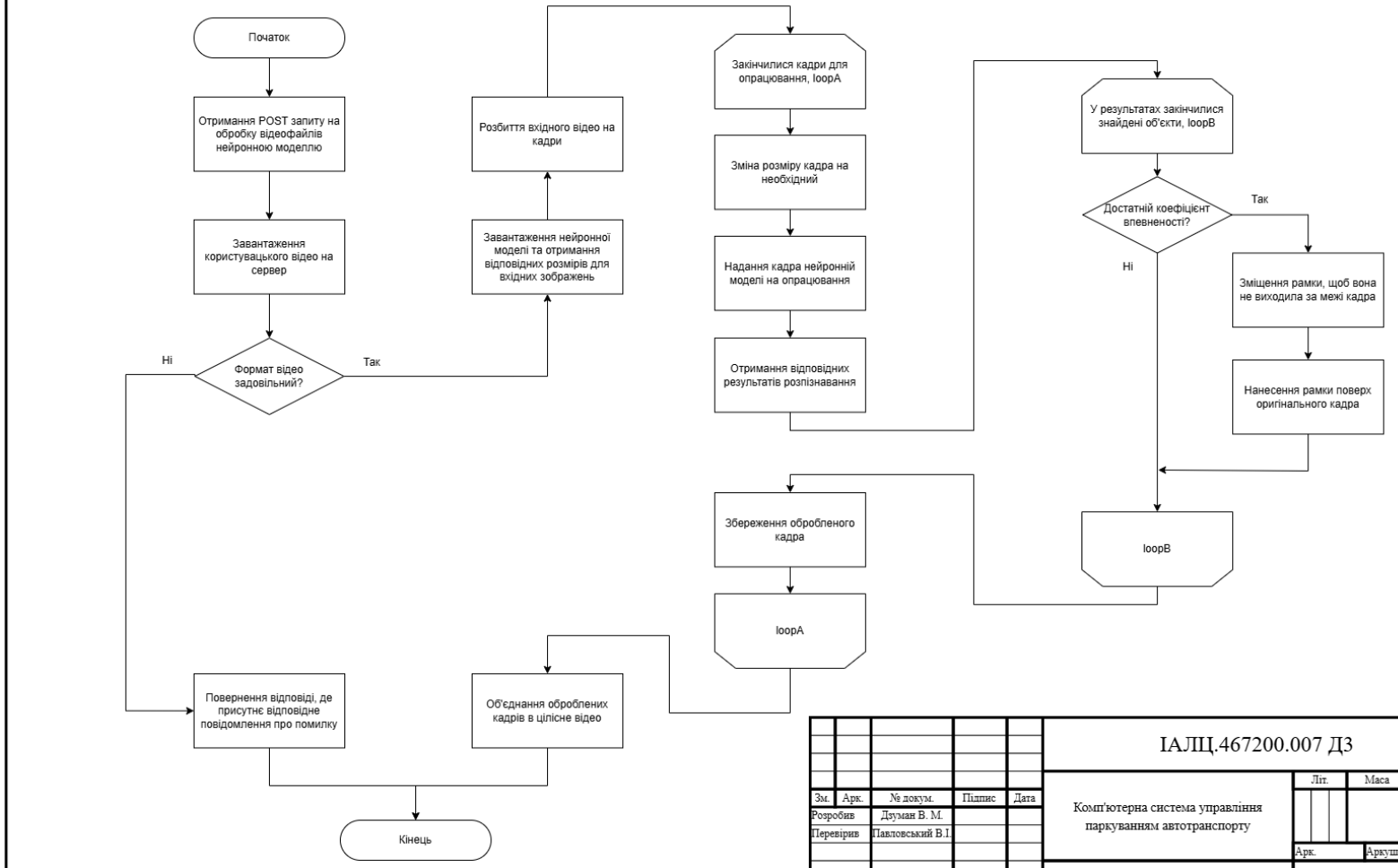
ІАЛЦ.467200.006 Д2

Комп'ютерна система управління паркуванням автотранспортом

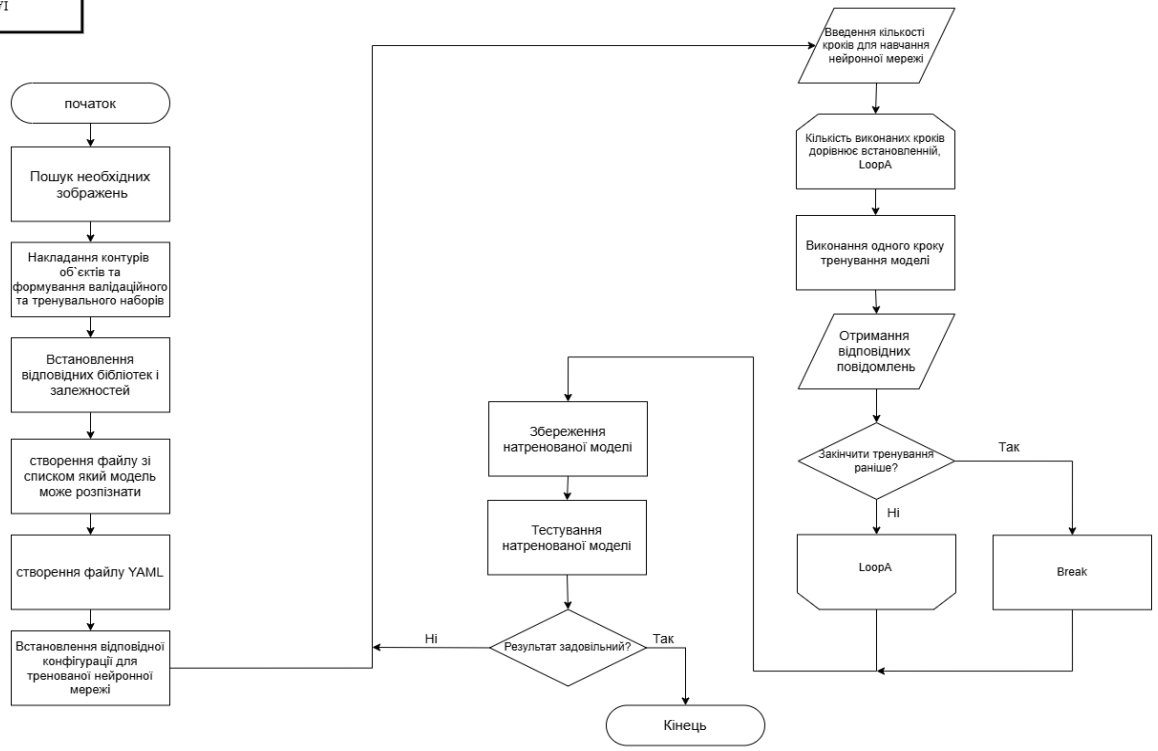
Літ.	Маса	Масштаб
Арк. 1	Аркушів 1	

Алгоритм обробки зображень. Блок-схема

КПІ ім. Ігоря Сікорського, ФПМ КВ-11



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.007 Д3		
Розробив		Дзуман В. М.			Комп'ютерна система управління паркуванням автотранспорту	Літ.	Маса / Масштаб
Перевірив		Павловський В.І.				Арк.	Аркушів
Н. контр.		Клятченко Я.М.			Алгоритм обробки відео файлів. Блок схема	КПІ ім. Ігоря Сікорського, ФПМ КВ-11	
Затв.		Романкевич В.О.					



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЩ.467200.008 Д4		
Розробив		Дзуман В. М.			Компютерна система управління паркуванням автотранспорту	Літ.	Маса / Масштаб
Перевірив		Павловський В.І.				Арк. 1	Аркушів 1
Н. контр.		Клятченко Я.М.			Алгоритм навчання нейронної мережі. Блок-схема.	КПІ ім. Ігоря Сікорського, ФІМ КВ-11	
Затв.		Романкевич В.О.					

Додаток Б

Компютерна система управління паркуванням автортранспорту

Виконав студент IV курсу, групи КВ-11

Дзуман Владислав Михайлович

Керівник: доц. каф. СПіСКС, к.т.н., доцент Павловський В. І.

Список умовних позначень

YOLO (You Only Look Once) – алгоритм для одночасного виявлення та класифікації об’єктів на зображенні.

IoU (Intersection over Union) – метрика для оцінки перекриття між виявленим об’єктом і еталонною зоною.

AI (Artificial Intelligence) – штучний інтелект.

DB (Database) – база даних.

UI (User Interface) – інтерфейс користувача.

HTML/CSS/JS – мови для створення клієнтської частини вебдодатку.

Flask – мікрофреймворк Python для створення вебсервісів.



Постановка задачі

Метою даної роботи є створення комп’ютерної системи управління паркуванням на основі технологій комп’ютерного зору для автоматичного виявлення транспортних засобів і визначення зайнятості паркувальних місць при обробці зображень та відео.



Вимоги до розробленого додатку

Автоматизація контролю паркувальних місць

Зменшення людського фактору у визначенні зайнятості

Виявлення авто на фото та відео за допомогою нейромереж

Висока точність при використанні YOLOv5

Миттєвий результат після завантаження зображення

Зручний вебінтерфейс для взаємодії з системою

Робота без потреби в підключенні до зовнішніх сервісів

Можливість локального аналізу зображень і відео



Актуальність задачі

Актуальність задачі полягає у створенні засобу,що дозволить автоматизувати контроль зайнятості паркомісць на фото та відео. Використання комп'ютерного зору нейромережі YOLOv5 дозволяє отримати швидкий і точний результат в зручному веб-інтерфейсі і спростити моніторинг паркінгу у приватному та муніципальному середовищі.



Аналіз предметної області



Проблема ефективного використання паркувальних майданчиків є актуальною для міст із високою щільністю транспорту. Недостатня кількість вільних місць, відсутність актуальної інформації про їх наявність і нераціональне використання площі призводять до заторів, втрати часу та збільшення навантаження на міську інфраструктуру. Традиційні рішення на основі ручного контролю або сенсорів мають обмежену точність і високу вартість. Тому виникає потреба в автоматизованих системах, які використовують комп'ютерний зір для аналізу зображень з камер спостереження та виявлення зайнятих паркомісць у режимі реального часу.

Галузь застосування

- Системи комп’ютерного зору для паркування застосовуються в міській інфраструктурі, на торгових майданчиках, біля офісних і житлових комплексів. Вони допомагають оптимізувати розміщення транспорту, зменшити затори, підвищити зручність для водіїв та ефективність використання паркувального простору.



Обґрунтування вибору YOLO та Flask

YOLOv5s:
– Висока швидкість обробки зображень у реальному часі;
– Достатня точність для задач детекції транспорту;
– Просте розгортання та підтримка.

Flask:
– Легкий у налаштуванні веб-фреймворк;
– Підходить для невеликих веб-додатків;
– Має зручну інтеграцію з Python та бібліотеками комп'ютерного зору.

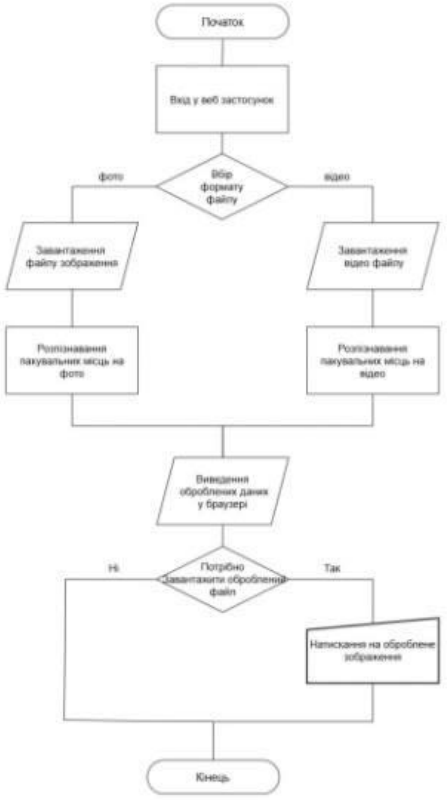


Загальна архітектура системи

Система складається з таких основних модулів:
– Клієнтська частина (веб-інтерфейс для користувача);
– Серверна частина (обробка запитів, зберігання даних);
– Модуль комп’ютерного зору (детекція об'єктів);
– База даних (збереження меж, зображень, статистики).
Компоненти взаємодіють через HTTP-запити у рамках архітектури клієнт-сервер.



Загальний алгоритм роботи розробленої програми

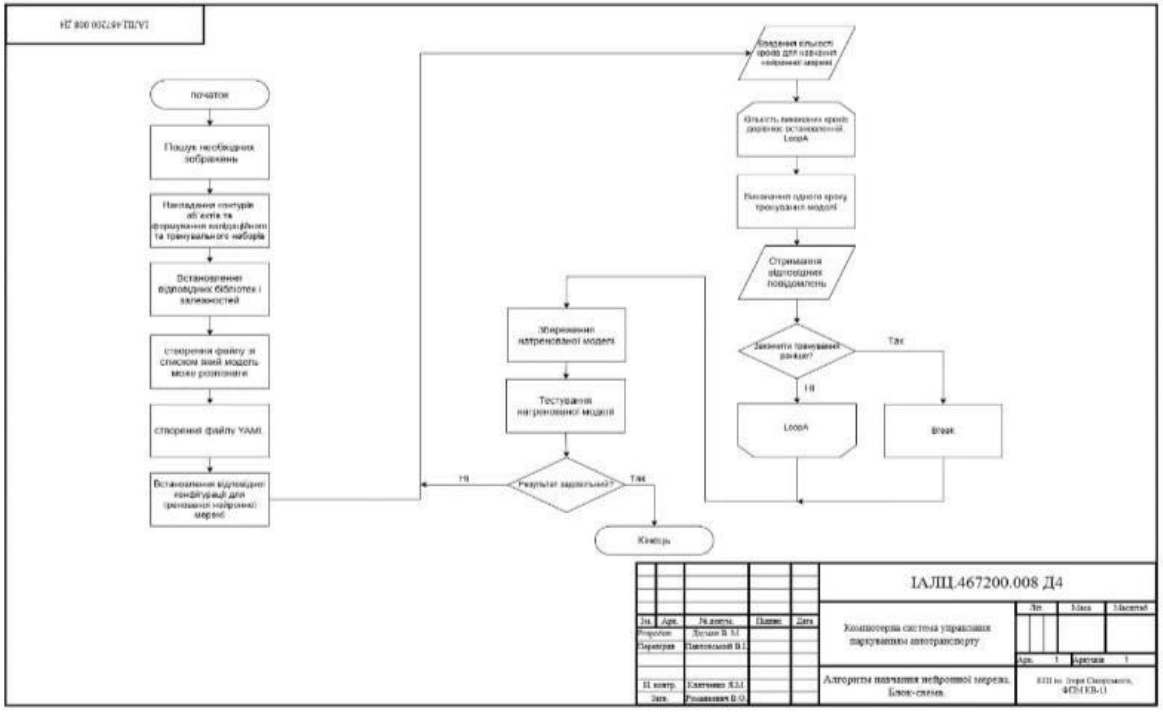


Модуль комп’ютерного зору (YOLOv5):



У системі використовується модель YOLOv5s, яка була обрана через високу швидкість інференсу та достатню точність виявлення транспортних засобів. Модель завантажується через фреймворк PyTorch та застосовується для обробки як зображень, так і відео. Вона повертає координати об'єктів та ймовірність класифікації. Для спрощення обчислень використовуються лише класи, пов’язані з транспортом: легкові авто, автобуси, вантажівки та мотоцикли. Це дозволяє пришвидшити обробку та зменшити кількість хибнопозитивних спрацювань.

алгоритм тренування нейронної мережі



Алгоритм обробки зображення

Після завантаження зображення користувачем, воно передається на сервер, де спочатку зчитується з диска та перетворюється у формат, сумісний з бібліотекою OpenCV. Далі це зображення подається на вхід попередньо завантаженій нейронній мережі YOLOv5s, яка виконує детекцію транспортних засобів та повертає координати прямокутників навколо виявлених об'єктів. Отримані координати порівнюються з межами паркомісць, заздалегідь збережених у базі даних. Для кожної пари "виявлений об'єкт – паркомісце" обчислюється коефіцієнт IoU. Якщо перекриття перевищує певний поріг, місце вважається зайнятим. В результаті формується анотоване зображення з кольоровими рамками (червоні – зайняті, зелені – вільні), яке передається користувачу через вебінтерфейс.

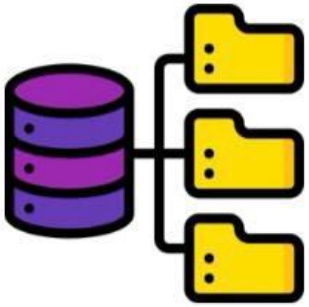


Веб-інтерфейс системи

Веб-інтерфейс реалізовано на базі HTML, CSS та JavaScript із використанням шаблонів Flask. Користувач має доступ до сторінок:
Upload — завантаження зображень або відео для аналізу.
Dashboard — огляд статистики, середнього часу обробки, результатів моделі.
Define — визначення меж паркомісць вручну.
Parkings/Models — управління парковками та перегляд точності моделі.
Інтерфейс інтуїтивний, адаптивний для різних пристроїв.



База даних та структура збереження



Система використовує SQLite як локальну базу даних. Основні таблиці:
– **parkings** — зберігає дані про об'єкти паркування.
– **boundaries** — координати меж кожного паркомісця.
– **uploads** — інформація про завантажені файли (зображення, відео).
– **model_evaluations** — результати оцінки точності нейромережі.
Структура забезпечує швидкий доступ до даних для аналізу та візуалізації.

Можливості розвитку та інтеграції

Система має потенціал масштабування до реального часу та міських середовищ.
Можливі напрямки розвитку:
– інтеграція з IP-камерами та IoT-пристроями;
– GPS-прив'язка паркувань для мобільного додатку;
– система сповіщень про вільні місця;
– інтеграція з платіжними та обліковими сервісами;
– розгортання у хмарному середовищі для зручного доступу та адміністрування.



Висновки

У результаті виконання дипломного проєкту було створено програмну систему для автоматизованого управління паркуванням з використанням технологій комп'ютерного зору. Реалізована архітектура дозволяє завантажувати зображення або відео, проводити детекцію транспортних засобів за допомогою моделі YOLOv5, виконувати порівняння з межами паркомісць та відображати результати у зручному веб-інтерфейсі. Всі дані зберігаються у локальній базі даних, що забезпечує гнучкість у роботі з об'єктами та результатами обробки. У ході розробки було обґрунтовано вибір основних технологій, протестовано роботу системи та перевірено її ефективність у різних умовах. Отримані результати підтверджують придатність системи для практичного використання в умовах локального або муніципального паркування. Система відкрита для подальшого розвитку, включаючи можливість інтеграції з відеопотоками, IoT-пристроями або мобільними клієнтами.



Дякую за увагу!