ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»
Факультет автоматизації виробництва та цифрових технологій
Кафедра цифрових технологій та проєктно-аналітичних рішень

«Допущено до захисту»
Гарант ОПП

Павло САГАЙДА

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістра

за підсумками виконання
освітньо-професійної програми
«Комп'ютерні науки та цифровий інтелект»
за спеціальністю 122 Комп`ютерні науки

на тему «**Дослідження методів, моделей та інформаційних технологій розпізнавання об'єктів роботоавтомобілем при його русі**»

Керівник роботи — Олександр ШМАТКО

Консультант від бази практики — Віталій НЕДОТОПА

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач — Юрій ЯКИМОВ

Підсумкова оцінка за атестацію			

Голова ЕК — Олена ПАВЛЕНКО

КРИВИЙ РІГ 2024

ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»

Факультет	автоматизації виробництва та цифрових технологій
Кафедра	цифрових технологій та проектно-аналітичних рішень
Ступінь вищої освіти	магістр
Спеціальність	122 Комп'ютерні науки
ОПП	Комп'ютерні науки та цифровий інтелект

ЗАТВЕРДЖУЮ
Гарант ОПП

_____ Павло САГАЙДА

«06» листопада 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Якимову Юрію Миколайовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи: *Дослідження методів, моделей та інформаційних технологій розпізнавання об'єктів роботоавтомобілем при його русі*
керівник роботи *Шматко О.В., доцент кафедри ЦТПАР*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету від 29.08. 2023 р. №137.1/29.08.2023
2. Термін подання роботи *10.01.2024 р.*
3. Вихідні дані до роботи *Навчальна література, державні стандарти, методична література з спеціальних дисциплін та дипломування, науково-дослідницькі роботи з тематики автоматизації обробки й аналізу даних та методів цифрового інтелекту, літературні джерела, результати власних експериментів та досліджень, технологічні інструкції тощо*
4. Зміст пояснювальної записки (перелік питань) *Реферат. Зміст. Вступ. 1. Аналіз стану питання, предметної області, концепцій з проблеми, що розглядається (літературний огляд, недоліки існуючих систем, сучасні тенденції). 2. Огляд методів інтелектуального аналізу в задачах управління роботизованими автомобілями 3. Проєктування та розробка програмних компонентів 4. Проведення та аналіз результатів теоретичних та експериментальних досліджень за індивідуальним завданням. 5. Економічне обґрунтування запропонованих технічних рішень. Висновки. Перелік використаних джерел. Додатки.*
5. Перелік графічного (демонстраційного) матеріалу (з точним зазначенням обов'язкових креслень): *Актуальність, мета, об'єкт, предмет та завдання дослідження; розроблені або удосконалені математичні моделі, методика дослідження; діаграми проекту програмно-методичного комплексу в нотації UML (діаграми прецедентів, класів, послідовностей, діяльності); результати розробки та експериментальних досліджень; результати економічних розрахунків; висновки до роботи; публікація результатів дослідження.*

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх.

Розділ	Прізвище, ініціали та посада консультанта
1	Шматко О.В., доцент кафедри ЦТПАР
2	Шматко О.В., доцент кафедри ЦТПАР
3	Шматко О.В., доцент кафедри ЦТПАР
4	Шматко О.В., доцент кафедри ЦТПАР
5	Гетьман І.А., доц. каф. ЦТПАР

7. Дата видачі завдання *06.11.2023*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1	Розділ 1. Огляд проблеми використання методів інтелектуального аналізу в задачах розпізнавання об’єктів	25.12.2023 - 30.12.2023
2	Розділ 2. Огляд методів інтелектуального аналізу в задачах управління роботизованими автомобілями	25.12.2023 - 30.12.2023
3	Розділ 3. Проєктування та розробка програмних компонентів	25.12.2023 – 02.01.2024
4	Розділ 4. Розробка та аналіз тестових наборів для розпізнавання автомобілів за допомогою YOLO	03.01.2024 - 07.01.2024
5	Розділ 5. Економічні розрахунки	03.01.2024 - 07.01.2024
6	Висновки, перелік посилань, вступ, зміст, реферат	07.01.2024 – 08.01.2024
7	Подання завершеної роботи. Перевірка на академічний плагіат	10.01.204 – 16.01.2024
8	Остаточне оформлення роботи, презентаційного матеріалу, автореферату	17.01.2024 – 19.01.2024
9	Рецензування завершеної роботи. Захист	19.01.2024 – 22.01.2024

Здобувач Юрій Якимов

Керівник роботи Олександр Шматко

РЕФЕРАТ

Кваліфікаційна робота: 125 с., 47 рис., 13 табл., 5 додатків, 116 літературних джерел.

Мета дослідження полягає у підвищенні точності розпізнавання рухомих об'єктів роботизованим автомобілем для точного визначення та прогнозування руху інших об'єктів на дорозі, безпечної та плавної їзди за рахунок дослідження методів та проєктування та розробки програмних компоненті для системи розпізнавання рухомих об’єктів.

Об'єкт дослідження – системи розпізнавання об’єктів у роботизованих автомобілях.

Предмет дослідження: методи та засоби побудови систем розпізнавання рухомих об’єктів для управління роботизованим автомобілем.

Метод дослідження – комплексний аналіз існуючих алгоритмів виявлення об'єктів, розробка та тренування моделей глибокого навчання на базі архітектури YOLO, експериментальна верифікація та порівняння ефективності різних моделей.

Робота включає аналіз існуючих досліджень, розробку програмних компонентів, тренування моделей глибокого навчання, аналіз роботи системи.

Наукова новизна полягає у впровадженні удосконалень в алгоритми обробки зображень та в реалізації адаптивних методів для підвищення точності і швидкодії системи розпізнавання в різноманітних умовах дорожнього середовища.

Практичне значення отриманих результатів виявляється у можливості використання розробленої системи для підвищення рівня безпеки та оптимізації управління в автономних транспортних засобах.

Публікації: представлено результати дослідження у вигляді тез доповідей на міжнародних наукових конференціях (Додаток Д).

Ключові слова: інтелектуальна система, розпізнавання об'єктів, роботизований автомобіль, YOLO, комп'ютерний зір, штучний інтелект, безпека дорожнього руху.

SUMMARY

Master's thesis: 124 pages, 47 figures, 13 tables, 5 appendices, 116 references.

The aim of the study is to improve the accuracy of moving object detection by an autonomous vehicle for precise identification and prediction of other objects' movement on the road, ensuring safe and smooth driving through the research, design, and development of software components for a moving object detection system.

The object of research - object detection systems in autonomous vehicles.

The subject of research: methods and tools for constructing object detection systems for the management of an autonomous vehicle.

Research method – a comprehensive analysis of existing object detection algorithms, development and training of deep learning models based on YOLO architecture, experimental verification, and comparison of the effectiveness of different models.

The work includes an analysis of existing research, the development of software components, training of deep learning models, and an analysis of the system's operation.

The scientific novelty consists in the introduction of enhancements to image processing algorithms and the implementation of adaptive methods to increase the accuracy and speed of the recognition system under various road conditions.

The practical significance of the obtained results is demonstrated by the potential use of the developed system to enhance safety levels and optimize control in autonomous vehicles.

Publications: research findings have been presented in the form of abstracts at international scientific conferences.

Keywords: intelligent system, object detection, autonomous vehicle, YOLO, computer vision, artificial intelligence, road safety.

ЗМІСТ

ВСТУП

З розвитком сучасних технологій тема розпізнавання об’єктів стає все більш актуальною. Розпізнавання об'єктів широко використовується в різних галузях промисловості, від особистої безпеки до підвищення продуктивності на робочому місці. Розпізнавання та виявлення об'єктів застосовується в багатьох областях комп'ютерного зору, включаючи пошук зображень, безпеку, нагляд, системи автоматичного керування транспортом та машинний огляд.

Розпізнавання об'єктів (англ. object detection) – це комп'ютерна технологія, пов'язана з комп'ютерним зором та обробкою зображень, яка має справу з виявлянням примірників семантичних об'єктів певного класу (таких як люди, будівлі чи автомобілі) у цифрових зображеннях та відео [1]. Ця технологія має здатність класифікувати один або декілька об'єктів на цифровому зображенні або відео одночасно.

Актуальною ця технологія є і при розробці безпілотних автомобілів. Щоб автомобіль міг вирішити, що робити на наступному кроці – прискоритися, натиснути на гальма або повернути – він повинен знати, де знаходяться всі об'єкти навколо автомобіля і які ці об'єкти (автомобілі, пішоходи, світлофори, дорожні знаки, велосипеди, мотоцикли та інше). Цю проблему дозволяє вирішити техніка розпізнавання образів.

Сучасний підхід до розпізнавання рухомих транспортних об'єктів полягає у використанні різних інтелектуальних технологій, заснованих на штучних нейронних мережах. Ці технології включають рекурентні та згорткові мережі, які дозволяють вирішувати широкий спектр задач. Завдяки цим інноваційним підходам, розпізнавання рухомих транспортних об'єктів стає більш точним та ефективним. Нейронні мережі (НМ) – це « це обчислювальні системи, натхнені біологічними

нейронними мережами, що складають мозок тварин. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу» [3].

Згідно [2], розпізнавання транспортних об’єктів (transport object detection) на фото та відео відбувається на сьогодні за допомогою нейронних мереж та застосовується в безпілотному транспорті, відеоспостереженні, системах контролю стану на дорогах, системах «розумного дому» тощо [4]. Також часто використовують системи розпізнавання автомобільних номерів [5], ідентифікації людини за зображенням особи [6], аналізу сцен з метою безпеки, технічного зору в робототехніці [7]. Розпізнавання в транспортному потоці об’єктів, що рухаються, є важливим етапом в обробці відеоматеріалу. Є кілька підходів, заснованих на різних методах, таких як віднімання фону [8], застосування ймовірнісних методів [9], використання НМ [10]. Віднімання фону широко застосовують у процесі розв’язання завдань відеоспостереження [11].

Задача розпізнавання об'єктів під час руху безпілотними транспортними засобами є складною, оскільки вимагає точного та швидкого виявлення різних об'єктів у реальному часі. Для цього потрібні додаткові дослідження та розвиток нових методів, які забезпечать ефективну роботу автономних транспортних засобів у різних умовах.

Дослідження в галузі розпізнавання об'єктів має великий потенціал для розвитку та впровадження нових технологій у сфері автономної транспортної системи. Постійний прогрес у цій галузі допомагає зробити безпілотні транспортні засоби більш безпечними та ефективними для суспільства.

Отже, розпізнавання об'єктів є одним з головних компонентів автономного керування автомобілями. Ця технологія дозволяє автомобілю отримувати детальну інформацію про навколишнє

середовище та приймати обґрунтовані рішення на основі цієї інформації. Використанням розпізнавання об'єктів дозволить створювати більш безпечні та ефективні системи автономного керування, які сприятимуть розвитку автономного управління та покращенню якості життя.

Об'єкт дослідження – системи розпізнавання об’єктів у роботизованих автомобілях.

Предмет дослідження: методи та засоби побудови систем розпізнавання рухомих об’єктів для управління роботизованим автомобілем.

Мета дослідження полягає у підвищенні точності розпізнавання рухомих об'єктів роботизованим автомобілем для точного визначення та прогнозування руху інших об'єктів на дорозі, безпечної та плавної їзди за рахунок дослідження методів та проєктування та розробки програмних компоненті для системи розпізнавання рухомих об’єктів.

Практична цінність: результати дослідження можуть бути використані для покращення систем автономного управління автомобілями, що забезпечить більшу безпеку та комфорт для пасажирів та інших учасників дорожнього руху. Розроблені методи та засоби також можуть бути використані в інших галузях, де потрібно розпізнавання рухомих об'єктів, наприклад, в медицині, промисловості, робототехніці та інших сферах, де важлива точність та швидкість розпізнавання.

Теоретична цінність: дослідження дозволить розширити знання про методи розпізнавання рухомих об'єктів та їх застосування в управлінні роботизованими автомобілями. Розробки в цій галузі можуть стати основою для подальшого наукового дослідження та розвитку нових технологій. Результати дослідження можуть бути використані в навчальних цілях, під час викладання курсів зі штучного інтелекту, машинного навчання та автономних систем.

РОЗДІЛ 1. ОГЛЯД ПРОБЛЕМИ ВИКОРИСТАННЯ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ В ЗАДАЧАХ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ

1.1 Огляд методів інтелектуального аналізу даних в задачах розпізнавання об’єктів

Інтелектуальний аналіз даних, також відомий як Data Mining, є процесом виявлення закономірностей у великих наборах даних за допомогою методів з галузей статистики, штучного інтелекту та аналізу даних. Основна мета інтелектуального аналізу даних – перетворення сирих даних на корисну інформацію.

На рис. 1.1 показана діаграма Венна, яка ілюструє взаємозв'язки між різними областями штучного інтелекту (ШІ). Для детального опису термінології, будь ласка, зверніться до Глосарію (Додатку В).

Рисунок 1.1 – Ієрархія штучного інтелекту

Найбільше коло, що охоплює всі інші, позначає штучний інтелект як найширшу область. Всередині цього кола є менше коло, що позначає машинне навчання як підмножину ШІ, яке включає алгоритми та техніки, дозволяючи системам навчатися та покращувати з досвідом. Ще менше коло всередині машинного навчання представляє нейронні мережі, які є особливим класом моделей машинного навчання, натхненних структурою та функціями мозку. І, нарешті, у центрі знаходиться глибоке навчання, яке є підмножиною нейронних мереж які мають багато шарів обчислень для представлення складних патернів у даних.

У глибокому навчанні мета полягає в оновленні параметрів нейронної мережі під час тренування таким чином, щоб модель навчилася представляти корисну функцію для свого завдання. Існує багато алгоритмів навчання, але більшість алгоритмів, можна класифікувати як навчання з вчителем або навчання з підкріпленням.

Навчання з вчителем використовує марковані дані, де експерт демонструє виконання обраного завдання. Кожна точка даних у наборі включає пару «спостереження-дія», яку нейронна мережа потім навчається моделювати. Під час навчання мережа наближає свою власну дію для кожного спостереження і порівнює помилку з маркованою дією експерта. Перевагою навчання під наглядом є швидкість збігу тренувань і відсутність потреби вказувати, як завдання має бути виконане. Хоча простота підходу під наглядом є привабливою, у підходу є деякі недоліки. Спочатку, під час тренування мережа робить прогнози щодо контрольної дії в офлайн-структурі, де прогнози мережі не впливають на стани, які спостерігаються під час тренування. Отже, дії мережі впливатимуть на майбутні стани, порушуючи припущення про незалежність і однакову розподільчу здатність [12–14]. Це призводить до зміни розподілу між тренуванням та експлуатацією, що може призвести до помилок мережі через незнайомі розподіли станів, що спостерігаються під час експлуатації. По-друге, навчання поведінки на

основі демонстрацій залишає мережу вразливою до упереджень у наборі даних. Для складних задач, таких як автономне водіння, має значення різноманіття набору даних, якщо метою є навчання моделі, яка може керувати в різних середовищах [14, 15].

Навчання з підкріпленням є методом, який дозволяє моделі навчатися шляхом взаємодії з середовищем і вчиться приймати рішення на основі отриманих винагород. Цей процес може бути формалізований як процес прийняття рішень Маркова, описаний кортежем (S, A, P, R), де:

– S представляє собою простір станів;

– A - простір можливих дій;

– P - модель ймовірності переходу між станами;

– R - функція винагороди.

На кожному кроці часу агент спостерігає певний стан s_t, вибирає дію a_t з множини можливих дій A, і потім середовище реагує і переходить в новий стан s_t+1, при цьому агент отримує винагороду r_t.

Метою агента є вивчення політики $\pi(s_t, a_t)$, яка відображає спостереження в дії так, щоб максимізувати накопичену винагороду. Основною перевагою цього підходу є те, що для навчання агенту не потрібні марковані набори даних, і він може навчатися на нових сценаріях через взаємодію з середовищем. Однак недоліком є низька ефективність використання вибірки, що може призвести до повільного збігання [16] і може вимагати дорогоцінних обчислювальних ресурсів та тривалого навчання в реальному середовищі [17].

Алгоритми навчання з підкріпленням можуть бути розділені на три основні класи: алгоритми, засновані на значенні, градієнт стратегії та алгоритми актор-критик [18].

Алгоритми, засновані на значенні (наприклад, Q-навчання [19]) оцінюють функцію значення V(s), яка представляє собою очікувану винагороду за перебування в даному стані. Ця оцінка дозволяє агенту

вибирати дії, які максимізують очікувані винагороди, якщо відома динаміка переходу між станами P. Проте у більшості випадків модель середовища не відома, і тому агент використовує функцію якості Q(s, a), яка оцінює значення дії в конкретному стані. Оптимальна стратегія визначається через жадібну максимізацію цієї функції, але не завжди можна гарантувати збіжність до оптимальної стратегії [20, 21].

Алгоритми з градієнтом стратегії (наприклад, REINFORCE [22]) не оцінюють функцію значення, а отримують параметризовану стратегію і максимізують очікувані винагороди через оновлення параметрів мережі, засноване на градієнтах. Цей підхід полягає в побудові функції втрат і обчисленні градієнта функції втрат щодо параметрів мережі. Параметри мережі оновлюються в напрямку градієнта стратегії під час навчання. Головним недоліком цього підходу є висока варіативність у оцінюваних градієнтах стратегії [23–25].

Алгоритми актор-критик (наприклад [26]) є гібридними методами, які комбінують функцію значення з параметризованою функцією стратегії. Це створює компроміс між недоліками високої варіативності градієнтів політики і недоліками методів, заснованих на значенні [27-29].

Ще однією ключовою характеристикою алгоритмів навчання з підкріпленням є тип функції винагороди. Функція винагороди може бути розрідженою або густою. У розрідженій функції винагороди агент отримує винагороду лише після конкретних подій, таких як успіхи або невдачі у виконанні завдань. Перевагою цього підходу є легкість визначення успіху чи невдачі для багатьох завдань. Однак це може призвести до проблеми складності вибірки, оскільки агент рідко отримує винагороду, що може сповільнити процес навчання. У густій функції винагороди агент отримує винагороду на кожному кроці часу на основі поточного стану, в якому він знаходиться. Це дозволяє агенту отримувати неперерваний сигнал зворотного зв'язку і оцінювати, наскільки корисними були обрані дії в кожному стані.

Вибір методу інтелектуального аналізу даних для розпізнавання об'єктів роботоавтомобілем істотно залежить від сфери його практичного застосування. Ключовими факторами є тип та різноманітність об'єктів, які потрібно ідентифікувати, умови оточення, в яких автомобіль повинен функціонувати, та динаміка дорожнього руху.

Точність та швидкість розпізнавання, які потрібні для безпечного автономного водіння, визначають вибір алгоритмів і моделей машинного навчання. Якість та обсяг наявних даних для тренування і моделей впливають на здатність алгоритму до узагальнення і адаптації до реальних ситуацій на дорозі. В результаті, ефективний вибір методу інтелектуального аналізу даних для системи розпізнавання об'єктів роботоавтомобіля вимагає комплексного підходу, що враховує усі ці аспекти.

1.2 Використання методів інтелектуального аналізу даних в задачах управління автомобілем

Керування рухом транспортного засобу можна поділити на два основні аспекти: бічний рух і повздовжній рух. Кожен з них має свої власні завдання і методи керування.

Бічний рух включає в себе управління позицією транспортного засобу на дорозі та виконання бічних дій, таких як зміна смуги або уникнення зіткнень. Для досягнення цього завдання зазвичай використовуються зображення з бортових камер як вхідні дані для нейронних мереж. Основними викликами в бічному керуванні є точне визначення положення транспортного засобу на дорозі, виявлення інших транспортних засобів та об'єктів, а також прийняття рішень щодо руху на дорозі.

Повздовжній рух включає в себе керування прискоренням транспортного засобу з метою підтримки бажаної швидкості, збереження безпечної відстані від інших транспортних засобів та уникнення зіткнень ззаду. Для досягнення цього завдання зазвичай використовуються датчики, які вимірюють відносну швидкість і відстань до інших транспортних засобів, такі як Radar або Lidar.

Багато дослідницьких проектів у сфері автономного водіння обирають фокусуватися на одному з цих аспектів, оскільки кожен з них представляє велику складність і вимагає різних технологій та рішень. Проте, існують також спроби поєднати бічне та повздовжнє керування для більшого рівня автономності транспортного засобу.

Різні типи систем керування мають різні виклики та відрізняються в аспектах реалізації. Також важливо враховувати, що різні транспортні засоби та середовища можуть вимагати різних підходів до керування.

Першим, хто вперше висловив ідею використання навчання з підкріпленням для керування транспортними засобами, був дослідник під ім'ям Yu [30]. Він пропонував систему слідування за дорогою, яка базувалася на роботі Pomerleau і використовувала навчання з підкріпленням для створення контролера. Однією з переваг цього підходу була можливість навчання на підставі попереднього досвіду керування в різних умовах та постійне вдосконалення навичок слідування за дорогою завдяки онлайн-навчанню. Інший дослідник Moriarty та інші [31], поєднали навчання з вчителем та навчання з підкріпленням для створення стратегії вибору смуги руху. Результати експерименту показали, що автотранспортні засоби з навченими контролерами могли підтримувати швидкість, близьку до бажаної, і рідше змінювали смугу руху. Більш того, навчена стратегія керування забезпечила кращий рух транспортного потоку, порівняно з контролерами, створеними вручну.

Слід відзначити, що розміри нейронних мереж, які використовувались у ранніх дослідженнях, значно менші порівняно з сучасними можливостями [32]. Зростаюча обчислювальна потужність, особливо завдяки паралельним графічним процесорам (GPU), відкрила нові перспективи для дослідження та впровадження глибокого навчання в різних сферах. Така обчислювальна потужність може значно зменшити час навчання та покращити продуктивність. Також важливою є доступність великих наборів даних та оптимізованих апаратних рішень, спеціально розроблених для глибокого навчання. Це робить процес навчання та перевірки нейронних мереж більш простим та результативним. Загалом, останні досягнення дозволяють досягти високого рівня продуктивності завдяки більш складним системам з більшим обсягом тренувальних даних та епізодів.

Використовуючи глибокі моделі з конволюційними нейронними мережами (CNN), дослідники Muller та інші [33] здійснили навчання моделі масштабованого радіокерованого автомобіля для подолання перешкод за межами дороги, в рамках проекту DARPA Autonomous VEhicle (DAVE). Модель була навчена на даних, отриманих з двох камер, спрямованих вперед, під час управління автомобілем людиною. З використанням 6-шарової CNN, ця модель досягла здатності подолати перешкоди при швидкості 2 м/с.

Розширюючи підхід DAVE, компанія NVIDIA також використала CNN для створення заключної системи керування рульовим управлінням транспортного засобу через навчання під наглядом. Ця система може автоматично оптимізувати свою продуктивність і виявляти корисні особливості навколишнього середовища, такі як визначення доріг і смуг руху. Використовуючи CNN (рис. 1.2) ця система може навчитися стратегії керування без явного ручного аналізу особливостей навколишнього середовища, планування маршруту або

визначення керувальних команд, і це вдалося здійснити за допомогою обмеженої кількості навчальних даних.

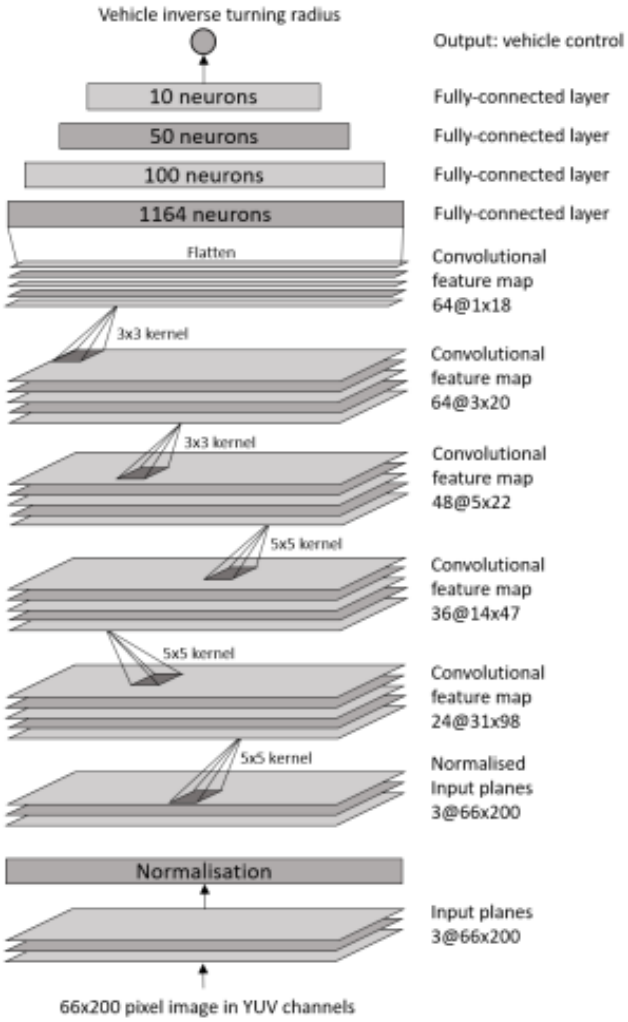

Рисунок 1.2 – Конволюційна нейронна мережа, що використовувалася у системі керування NVIDIA [32]

Набір тренувальних даних для цього дослідження включав в себе записи відео з камери і відомості про керування, надані людиною, яка керувала автомобілем. CNN мала 9 шарів, зокрема шар нормалізації, 5 конволюційних шарів і 3 повністю з'єднаних шарів, з загальною кількістю 27 мільйонів з'єднань і 250 000 параметрів. Цей метод досягнув 98% автономії на початкових тестах і 100% автономії під час 10-мильного

тесту на шосе, оціненого за кількістю втручань, необхідних протягом тесту. Проте важливо відзначити, що ця оцінка не враховує зміни смуги руху або повороти, і вона відображає лише здатність системи залишатися в поточній смузі руху.

Іншим прикладом використання навчання під наглядом для керування автономним транспортним засобом є робота, проведена Rausch та його співавторами [34], в якій навчання під наглядом використовувалося для створення системи бокового керування транспортним засобом. Для цього дослідження використовувалася глибока мережа CNN з чотирма прихованими шарами, трьома конволюційними шарами і одним повністю з'єднаним шаром. Для тренування цієї мережі використовували дані, які включали в себе кут керма та записи відео з камери, спрямованої вперед, зібрані під час управління автомобілем людиною в симуляторі CarSim [35]. Відео було записано з частотою 12 кадрів на секунду (FPS) і роздільною здатністю 1912x1036 протягом 15-хвилинної симуляції, що склалося з 10 800 кадрів. Після збору даних було вручну вилучено непотрібні кадри, такі, що виникли через некоректне керування або графічні помилки в симуляторі.

Проміжним етапом розвитку в напрямку бокового управління стало дослідження Eraqi та ін. [36] – використовували CNN з довготривалою та короткотривалою пам'яттю (C-LSTM) для навчання керування транспортним засобом на основі візуальних та динамічних часових залежностей. Мережу тренували так, щоб вона могла прогнозувати кути керування на основі вхідних зображень, і порівнювали її з простою архітектурою CNN, використаною у попередній роботі [37]. Експерименти показали, що модель C-LSTM виявилася точнішою і забезпечила більш плавні зміни керування порівняно зі звичайною CNN.

Проте важливо відзначити, що модель була оцінена лише в автономному режимі, порівнюючи передбачені дії управління з

реальними даними. Для отримання точної оцінки якості водіння слід використовувати живе тестування, де модель може керувати транспортним засобом в реальному середовищі та в реальному часі.

Крім того, були представлені техніки бокового керування для маневрів зміни смуги руху. Wang [38] використовували навчання з підкріпленням для тренування агента на виконання маневрів зміни смуги руху за допомогою глибокої Q-мережі (DQN). Мережа враховувала різні вхідні параметри, такі як швидкість транспортного засобу, поздовжнє прискорення, позиція, кут повороту, цільова смуга руху, ширина смуги та кривизна дороги, для надання безперервного значення бажаного кутового прискорення.

Результати досліджень технік бокового управління наведені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця технік бокового управління

Дослід.	**Стратегія навчання**	**Тип Мережі**	**Вхідні дані**	**Результати**	**Переваги**	**Недоліки**	**Тестування**
[39], [40]	навчання з вчителем	Пряме розповсюдження з 1 прихов. шаром	Зображення з камери	Дискретизовані кути керма	Перші позитивні результати на базі нейр.мереж	Проста мережа знижує продуктивність	Реальне та симуляція
[30]	з підкріпленням	Пряме розповсюдження з 1 прихов. шаром	Зображення з камери	Дискретизовані кути керма	Підтримка онлайн навчання	Зниження продуктивності	Симуляція
[33]	навчання з вчителем	6-шарова CNN	Зображення з камери	Кути керма	Стійкість до різноманітності навколишнього середовища	Великі помилки	Реальний світ

Продовження таблиці 1.1

Дослід.	**Стратегія навчання**	**Тип Мережі**	**Вхідні дані**	**Результати**	**Переваги**	**Недоліки**	**Тестування**
[32]	навчання з вчителем	9-шарова CNN	Зображення з камери	Значення кутів керма	Високий рівень автономної роботи	Розгляд тільки утримання у смузі, потребує втручання водія	Реальний світ
[34]	навчання з вчителем	8-шарова CNN	Зображення з камери	Значення кутів керма	Навчання з мін. тренувальними даними	Шумова поведінка сигналу керування	Симуляція
[36]	навчання з вчителем	C-LSTM	Зображення з камери	Значення кутів керма	Врахування часових залежностей	RNN важко тренувати, відсутність живого тестування	Тестування на прикладах
[38]	навчання з підкріпленням	3 прямі мережі	Стан трансп. засобу та геометрія дороги	Прискорення повороту транспортного засобу	Успішне виконання маневрів зміни смуги	Обмежене тестування, відсутність порівнянь з іншими алгоритмами зміни смуги	Симуляція

Важливо відзначити, що більшість згаданих робіт були проведені в симульованих середовищах, що створює необхідність подальших реальних тестів для підтвердження їхньої продуктивності в реальних умовах. Майбутні дослідження можуть розглядати розвиток більш глибоких моделей з більшою кількістю тренувальних даних та можливістю виконання більш широкого спектру дій. Живе тестування в реальних умовах залишається важливою частиною розвитку автономних транспортних засобів.

На відміну від систем бічного керування, контроль за відстанню, зазвичай, не використовує візуальні вхідні дані. Замість цього, основною інформацією для нього слугують дані, отримані від датчиків, таких як Radar або Lidar, які вимірюють відстань між транспортними засобами і їхнім станом. Ці дані мають меншу розмірність, наприклад, часовий інтервал або відносну відстань, і можуть бути ефективно використані для формування функції винагороди в контексті навчання з підкріпленням. У таблиці 1.2 представлено зведену інформацію про методи контролю за відстанню.

Таблиця 1.2 – Зведена інформація про методи контролю за відстанню

Дослід.	**Стратегія навчання**	**Тип Мережі**	**Вхідні дані**	**Результати**	**Переваги**	**Недоліки**	**Тестування**
[53]	Fuzzy Reinforcement Learning	Мережа прямого поширення з 1 прихованим шаром	Відносна відстань, відносна швидкість, попередній контрольний вхід	Кут дросельної заслонки, момент гальмування	Безперервні значення дій	Функція винагороди з одним терміном	Симуляція
[54]	Reinforcement Learning	Мережа прямого поширення з 1 прихованим шаром	Часовий проміжок, похідна проміжку	Прискорення, гальмування, або без дій	Підтримує безпечну відстань	Осцилятора поведінка прискорення, винагорода не враховує комфорт	Симуляція
[55]	Reinforcement Learning	Актор-Критик Мережа з мережами прямого поширення	Швидкість, помилка відстеження швидкості	Команди газу і гальма	Навчається з мінімальних даних навчання	Шумна поведінка сигналу прискорення	Реальний світ
[56]	Reinforcement Learning	Мережа пр.поширення з 5 прих.шарами	Швидкість транспортного засобу	Дискретизовані дії гальмування	Надійно уникає зіткнень	Розглядає уникнення зіткнень з пішоход.	Симуляція

Продовження таблиці 1.2

Дослід.	Стратегія навчання	Тип Мережі	Вхідні дані	Результати	Переваги	Недоліки	Тестування
[57]	Reinforcement Learning	Мережа прямого поширення з 1 прихованим шаром	Відносна відстань, відносна швидкість)	Бажане прискорення	Забезпечує плавний стиль водіння, навчається особистих стилях водіння	Немає методів для запобігання навчання поганих звичок від водіїв	Симуляція
[58]	Reinforcement Learning	Актор-Критик Мережа з мережами прямого поширення	Відносна відстань, відносна швидкість, прискорення господаря	Бажане прискорення	Враховує безпеку та комфорт, навчається	Пристосування небезпечних звичок водіїв може знизити безпеку	Симуляція
[59]	Supervised Reinforcement Learning	Актор-Критик Мережа з мережами прямого поширення	Відносна відстань, відносна швидкість	Бажане прискорення	Попереднє навчання за допомогою навчання з вчителем	Потребує навчання з учителем для збіжності, комфорт водіння не враховуєтьс я	Симуляція

Ще однією ключовою відмінністю між підходами бічного та повздовжнього керування є вибір стратегій навчання. Техніки бічного керування спираються на методи навчання з вчителем на основі розмічених наборів даних, тоді як алгоритми повздовжнього керування більше уваги приділяється методам навчання з підкріпленням, які навчаються через взаємодію з оточенням. Однак, слід відзначити, що для успішного застосування навчання з підкріпленням дуже важливо ретельно працювати над визначенням функції винагороди. Вона повинна враховувати безпеку, ефективність та комфорт водіння, оскільки погано спроектована функція винагороди може призвести до низької ефективності під час навчання моделі.

Додатково, важливим викликом для алгоритмів навчання з підкріпленням є баланс між дослідженням та використанням. Під час навчання агент повинен випробовувати випадкові дії, щоб досліджувати своє оточення, але в той же час використовувати здобуті знання для ефективного виконання завдань. Існують різні стратегії для досягнення цього балансу, такі як стратегія ε-жадібності, яка включає випадковий вибір дій з імовірністю ε, яка з часом зменшується, і алгоритм верхньої межі довіри, який сприяє дослідженню у станах з високою невизначеністю, використовуючи впевненість агента в станах з низькою невизначеністю. Таким чином, в системі реалізована внутрішня мотивація, яка спонукає агента досліджувати своє навколишнє середовище. У той час як використання найбільше підходить для станів, які вже були вивчені агентом [27, 42–44]. Інші підходи розглядали можливість використання навчання з вчителем як переднього етапу навчання, для отримання переваг як від навчання з підкріпленням, так і від навчання з вчителем.

Усі ці аспекти грають важливу роль у розвитку автономних систем керування та водіння, і подальші дослідження в цій області обов'язково допоможуть поліпшити ефективність та безпеку автономних транспортних засобів.

Попередні системи показали, що нейромережі можуть бути навчені для поздовжнього або бічного управління автомобілем. Однак для автономного водіння транспортного засобу необхідно здатність контролювати як керування, так і прискорення одночасно. Крім того, що використання глибоких нейронних мереж (DNN) для управління автомобілем може бути ефективним, але просте навчання автомобіля слідувати дорозі або залишатися в своїй смузі без врахування зовнішнього контексту не є достатнім для створення повністю автономних транспортних засобів. Люди водять автомобілі з метою досягнення певного пункту призначення, і навчання автомобіля за

допомогою камерних зображень для імітації людської поведінки водія не забезпечує повного розуміння контексту дій людського водія. Наприклад, при наближенні до розгалуження на дорозі стиль водіння може коливатися між двома можливими напрямками руху [39].

Для надання автономним транспортним засобам контекстної обізнаності, дослідники, такі як Hecker та інші [45], створили набір даних із 360-градусним оглядом, отриманим з 8 камер та водія, який слідував маршрутному плану. Цей набір даних використовувався для навчання глибоких нейронних мереж передбачати кут керма та швидкість на основі вхідних зображень та маршрутних планів з цього набору даних. Якісне тестування підтвердило, що модель навчилася імітувати поведінку людського водія. Однак живе тестування для підтвердження реальної продуктивності ще не було завершено.

З аналогічною метою Codevilla та інші [46] розробили алгоритм навчання з учителем, який використовує як зображення, так і високорівневі навігаційні команди для формування політики водіння. Мережа була навчена за допомогою методу навчання з учителем, де високорівневі команди включали такі інструкції, як "слідуйте за дорогою," "йдіть прямо," "поверніть ліворуч" або "поверніть праворуч." Автори тестували дві архітектури мережі, які можуть враховувати навігаційну команду: одна, де команда була додатковим входом у мережу, та інша, де мережа розгалужувалася в кінці на кілька підмодулів (послідовні шари) для кожної можливої команди. Автори зауважили, що остання архітектура працювала краще. Отриману мережу спочатку протестували в симуляції CARLA [47], а потім в реальних умовах на маштабованій до 1/5 автомобіль. Цей підхід в подальшому розширили в [48], додавши додатковий модуль для прогнозування швидкості, що допомагає мережі у деяких ситуаціях, наприклад, коли транспортний засіб під’їжджає до світлофору, передбачити очікувану швидкість з візуальних підказок та запобігти стану, коли транспортний засіб повністю

зупиняється. Подальші вдосконалення моделі включали глибший архітектурний дизайн мережі та більший набір даних для зменшення дисперсії під час навчання. Трохи інший підхід був досліджений за допомогою підсиленого навчання від Paxton et al. [49], де високорівнева команда надавалася за допомогою іншої мережі, відповідальної за прийняття рішень. Система складалася з мережі DDPG для низькорівневого керування та мережі DQN для стохастичної вищорівневої політики. Метою транспортного засобу була навігація на перехрестях, де деякі смуги мали зупинені транспортні засоби, тому транспортний засіб повинен був успішно пересуватися між смугами. Систему протестували на 100 симульованих перехрестях з зупиненими автомобілями перед смугами та без них, загалом 200 тестів. Без зупинених автомобілів агент перемагав у кожному тесті, але з зупиненими автомобілями відбулося 3 зіткнення.

Перехід від підходів "з одного кінця в інший" відбувся з роботи дослідників з Waymo, які недавно представили ChauffeurNet [50]. ChauffeurNet використовує "з середини до середини" навчання для створення стратегії керування, де вхідними даними є передпереробблений вид оточуючого середовища зверху, що представляє корисні функції, такі як дорожня карта, світлофори, маршрут, динамічні об'єкти та попередні положення транспортного засобу. Потім агент обробляє ці дані за допомогою рекурентної нейромережі (RNN), щоб забезпечити задану швидкість та точку шляху, які потім реалізуються за допомогою контролера нижнього рівня. Це має перевагу у тому, що передпереробблені дані можна отримати як з симуляції, так і з реальних даних, що полегшує перенесення стратегій керування з симуляції до реального світу [51, 52]. Крім того, створення певних змін для моделі, що відтворюють відновлення з неправильного положення в своєму ряду або навіть сценарії, такі як зіткнення або виїзд

за межі дороги, забезпечує моделі стійкість до помилок та дозволяє їй уникати таких сценаріїв.

Огляд підходів до повного керування автомобілем наведені в таблиці 1.3.

Таблиця 1.3 – Огляд підходів для повного управління автомобілем

Дослід.	**Стратегія навчання**	**Тип Мережі**	**Вхідні дані**	**Результати**	**Переваги**	**Недоліки**	**Тестування**
[60]	Навчання з підкріпленням під наглядом	Мережа прямого поширення з 2 прихованими шарами	Не зазначено	Керування, прискорення, гальмування	Швидке навчання	Нестабільне (може з'їхати з дороги)	Симуляція
[61]	Навчання з підкріпленням	Повністю з'єднана / Актор-Критик з прямими мережами	Позиція в смузі, швидкість	Керування, передача, гальма, прискорення (дискретизовано для DQN)	Неперервна політика забезпечує плавне керування	Просте симуляційне середовище	Симуляція
[62]	Навчання з вчителем	CNN / Прямий розповсюдження	Симульоване зображення з камери	Кут керування, двійкове рішення про гальмування	Оцінює безпеку політики в будь-якому стані, забезпечує стійкість до накопичуваних помилок	Просте симуляційне середовище, спрощений поздовжній вихід	Симуляція
[63]	Навчання з вчителем	CNN	Зображення з камери	Керування та газ	Швидке водіння, навчання на камерах низької вартості, стійкість DAgger до накопичуваних помилок	Навчений лише для еліптичних гоночних трас без інших транспортних засобів	Реальний світ (модель масштабного транспортного засобу) та Симуляція
[64]	Навчання з вчителем	CNN	Зображення	9 дискретних дій для руху	Увага до важливих об'єктів	спрощений простір дій	Симуляція

Продовження таблиці 1.3

Дослід.	**Стратегія навчання**	**Тип Мережі**	**Вхідні дані**	**Результати**	**Переваги**	**Недоліки**	**Тестування**
[65]	Навчання з підкріпленням	VAE-RNN	Семантично сегментоване зображення	Керування, прискорення	Покращує показники зіткнень порівняно з політиками, що включають лише гальмування	Враховує лише небезпеку безпосереднього зіткнення	Симуляція
[66]	Інверсне Навчання з Підкріпленням	CNN	LIDAR точкові хмари на сітчастій карті	Дискретні рухи	Стійке до шуму, уникає ручного формування функції вартості	Збільшене обчислювальне навантаження IRL, відсутність гарантії оптимальності функції вартості	Без живого тестування
[67]	Навчання з вчителем	CNN	360-градусне зображення з камери	Кут керування, швидкість	Враховує план маршруту	Брак живого тестування	Без живого тестування
[68]	Навчання з вчителем	CNN	Зображення з камери, навігаційна команда	Кут керування, прискорення	Враховує навігаційні команди, узагальнюється на нові середовища	Часом не вдається здійснити правильний поворот з першої спроби	Реальний (модель масштабного транспортного засобу) та Симуляція
[69]	Навчання з підкріпленням	Прямий розповсюдження з 1 приховани м шаром	набір характеристик для кожного найближчого транспортного засобу	Кут керування, прискорення	Враховує прийняття рішень, наданих іншою DNN	Велика кількість входів, які складно вилучити в реальності	Симуляція
[70]	Навчання з вчителем	CNN-RNN	Поперед. Обробл.зображення зверху околиць	Напрямок, швидкість	Легкість переносу	Може виводити контрольні точки	Реальний світ та Симуляція

Отже, навчання з вчителем є найбільш популярним підходом. Однак, потрібно пам’ятати, що стабільні та високопродуктивні моделі все ще залишаються недосяжними. Наприклад, техніки, які впроваджують повне керування автомобілем, мають гірші результати в керуванні кермом, ніж техніки, які розглядають тільки керування кермом. Це пояснюється значним збільшенням складності завдання, яке нейронна мережа навчається виконувати. З цієї причини нейронні мережі були навчені та оцінені в спрощених симульованих середовищах. Хоча повне керування автомобілем має бути кінцевою метою технік автономного керування, сучасні підходи ще не досягли достатньої продуктивності в складних і динамічних середовищах. Тому для подальшого покращення керування автономними транспортними засобами, потрібні нові дослідження.

Швидкий прогрес у впровадженні систем глибокого навчання у автономні транспортні засоби призвів до доступності різноманітних наборів даних глибокого навчання для автономного водіння та сприйняття. Одним з найвідоміших наборів даних для автономного водіння є комплект бенчмарків KITTI (рис. 1.3) [71, 72], який включає кілька наборів даних для оцінки стерео зору, оптичного потоку, потоку сцени, одночасної локалізації та картографування, виявлення та відстеження об'єктів, виявлення доріг та семантичної сегментації. Інші корисні набори даних включають Waymo Open [73], Oxford Robotcar [74], ApolloScape [75], Udacity [76], ETH Pedestrian [77] та Caltech Pedestrian [78]. Для більш повного огляду доступних наборів даних для автономного водіння дивіться дослідження Yin & Berger [79].

Рисунок 1.3 – Приклад набору даних KITTI

Окрім публічних наборів даних, також існує ряд інших інструментів, доступних для розробки глибокого навчання в автономних транспортних засобах. Провідною на сьогодні платформою штучного інтелекту (AI) для автономного водіння є NVIDIA Drive PX2 [80], яка забезпечує два системи на кристалі (SoC) Tegra та два графічних процесори Pascal з окремою пам'яттю та спеціалізованою підтримкою для розрахунків DNN. Для більш різноманітних завдань MobilEye EyeQ5 [81] забезпечує чотири повністю програмовані акселератори, кожен з яких оптимізований для різних сімейств алгоритмів машинного навчання. Це різноманіття може бути корисним у системах, де використовуються різні сімейства алгоритмів глибокого навчання. З іншого боку, SoC Cyclone V [81] від Altera забезпечує рішення для водіння, оптимізоване для сенсорної фузії. Для більш глибокого огляду апаратних платформ автономного водіння дивіться обговорення Liu et al. [83].

1.3 Огляд методів розпізнавання об’єктів

Виявлення об'єктів за допомогою глибокого навчання та комп'ютерного зору для роботи з потоками відео та відеофайлами надає можливість ідентифікувати різні типи об'єктів. Розпізнавання об'єктів є важливим завданням у сфері обробки зображень та комп'ютерного зору, і його метою є визначення та ідентифікація об'єктів на зображенні. Люди можуть легко розпізнавати об'єкти у реальному світі без особливих зусиль, тоді як комп'ютери самостійно не здатні для цього. Виявлення об'єктів є ключовим завданням у візуальному сприйнятті та однією з основних областей застосування комп'ютерного зору. Суть його полягає у виявленні та локалізації певних об'єктів на цифровому зображенні.

Розпізнавання об'єктів є однією з фундаментальних задач у галузі комп'ютерного зору. Цей процес включає в себе пошук та ідентифікацію окремих екземплярів об'єктів на цифрових зображеннях, які можуть бути збережені у відео або в режимі реального часу.

Виявлення об'єктів є передовою технологією у галузі комп'ютерних наук, що пов'язана з аналізом зображень і відео за допомогою комп'ютерного зору. Ця інноваційна технологія дозволяє виявляти та ідентифікувати об'єкти, такі як люди, автомобілі та тварини, на цифрових зображеннях і відео. Один з головних плюсів цієї технології полягає в її здатності класифікувати одночасно кілька об'єктів на зображенні або відео. Незважаючи на те, що виявлення об'єктів розвивалося протягом багатьох років, зараз воно стає більш розповсюдженим у різних галузях промисловості, ніж коли-небудь раніше.

Одним із важливих аспектів систем виявлення об'єктів є різноманітність методів, які використовуються для їх побудови. Хоча

існує безліч підходів, використання глибокого навчання надає велику точність та ефективність в розпізнаванні об'єктів.

Відстеження об'єктів, це процес локалізації рухомих об'єктів у часі з використанням відеокамери. Основною метою відстеження є асоціація об'єктів на послідовних кадрах відео. Це вимагає визначення положення та форми об'єктів на кожному кадрі. Таким чином, виявлення та класифікація об'єктів становлять важливі попередні кроки у відстеженні об'єктів з використанням комп'ютерного зору. Після виявлення об'єкта його можна класифікувати, наприклад, як транспортний засіб, людину, птаха або інший рухомий об'єкт.

Важливо зазначити, що відстеження об'єктів у послідовних кадрах є вкрай складним завданням обробки зображень. Це пов'язано з різними викликами, такими як складний рух об'єкта, неправильна форма об'єкта, можливе перекриття об'єктів один одним або об'єктом і фоном, а також вимоги до обробки в реальному часі.

Виявлення та відстеження об'єктів – це ключові складові сучасних систем комп'ютерного зору, які знаходять широке застосування в різних сферах, від автономних автомобілів і систем безпеки до медичної діагностики та анімації. Ці технології постійно розвиваються і надають нові можливості для автоматизації та вдосконалення різних аспектів нашого життя і промисловості.

Рисунок 1.4 – Процес виявлення об'єктів

Виявлення об'єктів – це технологія, яка знаходить широке використання в різних галузях, від забезпечення особистої безпеки до підвищення продуктивності на робочому місці.

Оптичне розпізнавання символів або оптичний читач символів, часто скорочено як OCR – це механічне або електронне перетворення зображень друкованого, рукописного або надрукованого тексту у машинно-кодований текст, незалежно від того, чи йдеться про сканований документ, фотографію документа, фотографію сцени (наприклад, текст на вивісках та бігбордах у ландшафтній фотографії) чи про текст субтитрів, накладений на зображення, ми видобуваємо символи з зображення чи відео.

Рисунок 1.5 – Оптичне розпізнавання символів

Широко використовується як спосіб введення інформації з друкованих паперових даних – чи то паспортні документи, рахунки-фактури, виписки з банків, комп'ютеризовані чеки, візитні картки, пошта, роздруківки статичних даних або будь-яка відповідна документація, це поширений метод цифрового перетворення друкованих текстів, щоб вони могли бути електронно редаговані, швидко знайдені, зберігатися більш компактніше, відображатися в мережі, та використовуватися у машинних процесах, таких як когнітивні обчислення, машинний переклад, (видобутий) текст у мову.

Одним з найкращих прикладів, для чого потрібне визначення об’єктів – автономне водіння. Для того, щоб автомобіль міг вирішити, що робити в наступний момент: прискорюватися, гальмувати, чи повертати, він повинен знати, де знаходяться всі об'єкти навколо себе, і що це за об'єкти. Це вимагає розпізнати об’єкти, і ми по суті навчаємо автомобіль виявляти відомий набір об'єктів, таких як автомобілі, пішоходи, світлофори, дорожні знаки, велосипеди, мотоцикли тощо.

Рисунок 1.6 – Відстеження транспортних засобів

Рисунок 1.7 – Самокеровані транспортні засоби

Виявлення пішоходів є важливим та головним завданням у будь-якій інтелектуальній системі відеоспостереження, оскільки воно

забезпечує основну інформацію для семантичного розуміння відеоматеріалів. Це має очевидне доповнення для автомобільних застосувань завдяки потенціалу покращення систем безпеки.

Рисунок 1.8 – Виявлення пішоходів

Автономні допоміжні роботи повинні бути оснащені можливістю обробляти візуальні дані в реальному часі, щоб вони могли адекватно реагувати та швидко адаптуватися до змін у навколишньому середовищі. Надійне виявлення та розпізнавання об'єктів зазвичай є необхідним початковим кроком для досягнення цієї мети.

Рисунок 1.9 – Робототехніка

Відеоспостереження є невід'ємною частиною безпеки та патрулювання. Останні досягнення в галузі комп'ютерного зору привели до розробки різних автоматизованих систем спостереження, однак їх

ефективність негативно впливає на багато факторів, і вони не є повністю надійними.

Розпізнавання образів є однією з найбільш розповсюджених і важливих технологій у сфері штучного інтелекту, яке відіграє ключову роль у задачах автоматичного керування роботизованими транспортними засобами. Використання глибоких нейронних мереж забезпечує здатність точно ідентифікувати та класифікувати різноманітні об'єкти у візуальних даних, що є критично важливим для безпеки і надійності автономних автомобілів. Ці системи дозволяють автомобілям "бачити" та "розуміти" дорожнє середовище, включно з іншими учасниками дорожнього руху, пішоходами, дорожніми знаками та лініями розмітки, що є вирішальним для ефективного прийняття рішень в реальному часі.

1.4 Висновки до розділу 1

Розробка роботоавтомобілів, здатних до автономного керування, є однією з ключових технологічних тенденцій у сучасному світі. Однак для досягнення цієї мети необхідно вирішити багато викликів, включаючи розпізнавання об'єктів.

Головною метою системи розпізнавання об'єктів для роботоавтомобіля є підвищення безпеки та підтримка автономного керування. Ця система повинна бути здатною аналізувати навколишнє середовище та надавати інформацію, яка допоможе водію або автономному пілоту приймати обґрунтовані рішення на дорозі.

Метою дослідження є підвищення точності розпізнавання рухомих об'єктів, яка дозволить роботизованому автомобілю точно визначати та прогнозувати рух інших об'єктів на дорозі, забезпечуючи безпечну та

плавну їзду за рахунок дослідження методів та проєктування та розробки програмних компоненті для системи розпізнавання рухомих об’єктів

Для досягнення цієї мети необхідно вирішити наступні задачі:

– виконати огляд проблеми використання методів інтелектуального аналізу в задачах розпізнавання об’єктів;

– виконати огляд методів інтелектуального аналізу в задачах управління роботизованими автомобілями;

– виконати проектування та розробку програмних компонентів для системи розпізнавання об’єктів в системі управління роботизованими автомобілями;

– виконати дослідження запропонованої системи розпізнавання об’єктів в системі управління роботизованими автомобілями.

– Розв'язання цих завдань дозволить створити ефективні програмні компоненти для аналізу відеопотоків, які знадобляться для інтеграції в інтелектуальні системи комп'ютерного зору.

РОЗДІЛ 2. ОГЛЯД МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ В ЗАДАЧАХ УПРАВЛІННЯ РОБОТИЗОВАНИМИ АВТОМОБІЛЯМИ

2.1 Розвиток роботизованих автомобілів

В останні роки розвиток роботоавтомобілів визначає новий етап у трансформації транспортної індустрії. Автономні транспортні засоби обіцяють не лише змінити парадигму пересування, але й вирішити проблеми мобільності в сучасному суспільстві. Одним із ключових напрямків в цьому контексті є використання передових технологій, зокрема глибокого підсиленого навчання, для розвитку ефективних та безпечних систем автоматизованого керування.

Розглянемо сучасні підходи до навчання нейронних мереж-агентів, спрямовані на досягнення оптимальних стратегій управління реальними роботоавтомобілями.

Нейронна мережа-агент навчається відображати свій оцінений стан в команди прискорення та керування з урахуванням мети досягнення конкретного цільового стану та виявлених перешкод. Навчання виконується за допомогою передової оптимізації наближених політик у поєднанні зі симульованим середовищем. Тривалість навчання з нуля становить від п'яти до дев'яти годин. Отриманий агент оцінюється у симуляції та подальше застосовується для керування дослідницьким транспортним засобом повного розміру. В рамках цього розглядається автономне дослідження парковки, включаючи маневри повороту та уникнення перешкод. В цілому ця робота є одним з перших прикладів успішного застосування глибокого підсиленого навчання до реального транспортного засобу.

Автономні автомобілі мають потенціал стало змінити сучасне суспільство, яке сильно залежить від мобільності. Переваги такої технології охоплюють самостійне спільне використання автомобілів та підходи платунінгу (від анг. Platoon – рух колонами або скупчення, а також це спосіб керування групою транспортних засобів. Мається на увазі підвищення пропускної спроможності доріг за допомогою автоматизованої системи автомобільних доріг [84]), що призводить до набагато ефективнішого використання транспортних засобів та доріг [85]. Останніми роками були зроблені великі досягнення у розробці цих систем, і головним чинником успіху є результати, досягнуті за допомогою методів глибокого навчання. Один з прикладів – нейронна мережа PILOTNET, яка була навчена керувати автомобілем виключно за допомогою зображень з камери [86–87].

Більш класичні методи розділяють обробку даних сенсорів та обчислення керування автомобілем на окремі задачі. Останнє можна досягти за допомогою різних модельно-заснованих підходів до керування, одним з яких є лінійний квадратичний контролер [88], також відомий як Riccati-контролер. Він мінімізує квадратичну цільову функцію у відхиленні стану та енергії керування, враховуючи лінійну модель підпорядкованої системи.

Хоча Riccati-контролер є відносно швидким, він не дозволяє прямо враховувати обмеження, такі як перешкоди, або більш розширені цільові функції у межах оптимізації. Такі вимоги відповідають загальному підходу нелінійного модельного передбаченого керування (MPC), що базується на вирішенні оптимальної задачі керування на кожному кроці часу. Хоча потрібні обчислення значно складніші, такі методи були успішно реалізовані для автономних транспортних засобів [89 - 90], використовуючи ефективні рішення, такі як TRANSWORHP [91] на основі WORHP [92].

Поєднання переваг швидкості Riccati-контролера та універсальності MPC можливе за допомогою знаходження функції, яка відображає значення стану на керуючі змінні, за допомогою навчання глибокої нейронної мережі. Таку модель можна навчити навчанням з учителем, як це зроблено для PILOTNET, або за допомогою підсиленого навчання. Останнє, зокрема, призвело до відмінних результатів у навчанні таких агентів для керування реальними системами, такими як роботи [93] чи вертольоти [94]. Останні роботи також показують перспективні застосування підсиленого для автономного водіння для прийняття стратегічних рішень [95–96] або обчислення керувальних команд [97–99]. Хоча ці результати демонструють успіх такого підходу в симульованих середовищах, дуже мало прикладів оцінки на реальних автомобілях. Один з них був представлений командою дослідників WAYVE, де було навчено політику слідування за смугою на основі відповідних зображень з камери. Навчання проводиться на борту автомобіля, і єдиний зворотний зв'язок для покращення надходить від втручання водія-інструктора. Хоча цей метод працює, коли навчання відбувається без наближення реальних перешкод і при низьких швидкостях, може бути складно застосувати цей підхід для більш загальних ситуацій.

2.2 Набори даних для побудови систем управління роботизованними автомобілями

Сьогодні загальновизнаним фактом являється те, що найважливішим інгредієнтом для побудови відмінної моделі машинного навчання є навчальні (тренувальні) дані. Ця практика систематичного створення наборів даних для досягнення кращої продуктивності моделі

часто називається дата-центричним штучним інтелектом. Проте спочатку більшість користувачів не мають великої кількості власних даних. З чого ж тоді почати? Відповідь на це питання одна - використати великий публічний набір даних.

Але проблема полягає в тому, що наразі доступно тисячі публічних наборів даних. Наше завдання полягає в тому, щоб скомпонувати найкращі публічні набори даних для найпоширеніших проблем і завдань у машинному навчанні.

Далі представлено список де охоплено кращі набори даних для виявлення об'єктів. Виявлення об'єктів є одним із найпоширеніших типів завдань у комп'ютерному зорі і використовується у різних сферах від розпізнавання обличь до автономного водіння.

Набори даних для розпізнавання образів:

COCO. Microsoft створила COCO (Common Objects in Context), великий набір даних для розпізнавання зображень. Він широко використовується в дослідженнях комп’ютерного зору та ідентифікації об’єктів і вважається одним із найкращих наборів даних виявлення об’єктів.

Набір даних MS COCO оцінює системи виявлення об’єктів і розпізнавання зображень. Зображення в наборі даних мають обмежувальні рамки навколо речей, позначених на них, забезпечуючи ретельний набір для навчання алгоритмам виявлення об’єктів. Крім того, набір даних містить маски сегментації екземплярів, які надають інформацію про форму об’єктів на зображенні.

Параметр	Значення
Загальна кількість зображень	330,000
Загальна кількість класів	91
Екземпляри об’єктів	1,5 млн.
Роздільна здатність	до 640 x 480 пікселів
Типи класів	Тварини, транспорті засоби, меблі, предмети побуту
Метрики оцінки	Середня точність, Recall, F1-Score

Рисунок 2.1 – Основні характеристики набору даних

1. Pascal VOC. Набір даних Pascal Visual Object Classes (VOC) є еталоном для виявлення та класифікації об’єктів у комп’ютерному зорі. Він був створений проектом Visual Object Classes (VOC) в Оксфордському університеті та став стандартним набором даних для оцінки алгоритмів виявлення об’єктів. Він містить зображення цих об’єктів у різних позах і на фоні, що робить його різноманітним і складним набором даних для алгоритмів виявлення об’єктів.

Параметр	Значення
Загальна кількість зображень	20 000
Загальна кількість класів	20
Екземпляри об’єктів	27 000+
Роздільна здатність	500×375 пікселів
Типи класів	Тварини, транспорт, предмети побуту, люди
Метрики оцінки	Середня середня точність (mAP), середня кількість правильних виявлень (ANCD)

Рисунок 2.2 – Основні характеристики набору даних Pascal VOC

2. ImageNet. ImageNet – це один із найвідоміших загальнодоступних наборів даних для візуального розпізнавання

об'єктів. Професор Фей-Фей Лі з Університету Стенфорда розпочала роботу над ImageNet у 2007 році.

Набір даних містить понад 14 мільйонів зображень, які були вручну позначені у понад 20 000 категоріях, що представляє одну з найбагатших таксономій серед будь-яких наборів даних з комп'ютерного зору.

Параметр	**Значення**
Загальна кількість зображень	14,197122
Загальна кількість класів	1000
Екземпляри об’єктів	>14 млн.
Роздільна здатність	256 x 256 пікселів
Типи класів	загальні об'єкти, абстрактні концепції
Метрики оцінки	Top-1 accuracy, Top-5 accuracy

Рисунок 2.3 – Основні характеристики набору даних ImageNet

3. CIFAR 100. Набір даних розпізнавання зображень CIFAR-100 широко використовується в дослідженнях машинного навчання. У ньому 100 класів із 600 зображеннями кожен, загальна кількість яких становить 60 000. Дрібнозернисті класи включають тварин, автомобілі та побутові предмети, тоді як крупнозернисті класи включають птахів і ссавців. CIFAR-100 — це жорсткий набір даних через невеликий розмір зображення та велику кількість класів, що робить його чудовим тестом для систем розпізнавання об’єктів.

Параметр	Значення
Загальна кількість зображень	60 000
Загальна кількість класів	100
Екземпляри об’єктів	-
Роздільна здатність	32×32 пікселів
Типи класів	Тварини, транспорт, предмети побуту, люди
Метрики оцінки	Top-1 accuracy, Top-5 accuracy

Рисунок 2.4 – Основні характеристики набору даних CIFAR-100

4. Open Images V6. Набір даних, який можна використовувати для розпізнавання, сегментації та виявлення об’єктів. Випущений у лютому 2020 року набір даних включає анотовані зображення з мітками та сегментацію пікселів. Зображення в наборі даних були зібрані з різних джерел, включаючи Flickr, Wikipedia та веб-сайт Open Images.

Обмежувальні рамки на зображеннях у наборі даних вказують на положення та розмір об’єктів всередині зображення. Колекція також містить анотації візуального зв’язку, які показують асоціації між речами на зображенні, як-от «людина верхи на коні» або «собака, що грає з м’ячем».

Параметр	Значення
Загальна кількість зображень	60 000 1 743 042 зображення (навчання), 41 620 зображень (перевірка) і 125 436 зображень (тест)
Загальна кількість класів	600
Екземпляри об’єктів	600 000
Роздільна здатність	від 256×256 до 2048×2048 пікселів
Типи класів	Тварини, транспортні засоби, меблі, кухонне приладдя, спортивне обладнання тощо.
Метрики оцінки	mean Average Precision (mAP) and mean Precision @ Overlap (mP@O)

Рисунок 2.5 – Основні характеристики набору даних Open Images V6

5. KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute). KITTI є одним із найпопулярніших наборів даних мобільної робототехніки та автономного водіння. Він складається з годин сценаріїв дорожнього руху, записаних за допомогою різних датчиків, включаючи RGB високої роздільної здатності, стереокамери у градаціях сірого та 3D-лазерний сканер. Технологічний інститут Карлсруе та Технологічний інститут Toyota розробили цей набір даних, який включає зображення з різними контекстами, такими як міські та сільські місця, шосе та перехрестя.

Параметр	**Значення**
Загальна кількість зображень	7481 навчальних зображень та 7518 тестових зображень
Загальна кількість класів	8
Екземпляри об’єктів	80 256
Роздільна здатність	1248×384 пікселів
Типи класів	Тварини, транспортні засоби, меблі, пішохід, спортивне обладнання тощо.
Метрики оцінки	Середня точність (AP), крива точності/відклику, швидкість виявлення

Рисунок 2.6 – Основні характеристики набору даних KITTI

6. Labelbox Open Images. Open Images від Labelbox – це набір даних для розпізнавання зображень. OpenAI розробив його в партнерстві з Labelbox, платформою анотації даних. Зображення в наборі даних отримано з різних джерел, які відрізняються за якістю, масштабом об’єкта та складністю сцени.

Параметр	Значення
Загальна кількість зображень	9 млн.
Загальна кількість класів	6000
Екземпляри об’єктів	60 000
Роздільна здатність	Різні, в середньому 1024×1024 пікселів
Типи класів	Люди, тварини, транспортні засоби, предмети побуту
Метрики оцінки	Середня середня точність (mAP)

Рисунок 2.7 – Основні характеристики набору даних Labelbox Open Images

7. GluonCV. Набір даних GluonCV Object Detection — це набір даних виявлення об’єктів, створений GluonCV, бібліотекою комп’ютерного зору з відкритим кодом на основі архітектури глибокого навчання MXNet. Цей набір даних пропонує фахівцям-практикам і науковцям масивний високоякісний набір даних для навчання та оцінки алгоритмів виявлення об’єктів.

Параметр	Значення
Загальна кількість зображень	160 000
Загальна кількість класів	80
Екземпляри об’єктів	Понад 1 млн.
Роздільна здатність	512×1512 пікселів
Типи класів	Тварини, транспортні засоби, предмети загального користування
Метрики оцінки	Середня середня точність (mAP), середня точність (AP) на різних порогових значеннях

Рисунок 2.8 – Основні характеристики набору даних GluonCV

8. Udacity Self Driving Car. Оригінальний набір даних Udacity Self Driving Car Dataset мав значну кількість відсутніх міток для пішоходів,

велосипедистів, автомобілів і світлофорів, що призводило до потенційних неточностей і низької продуктивності моделі. Це може призвести до небезпечних наслідків у застосуванні до технології безпілотних автомобілів. Однак набір даних доступний для завантаження у форматах VOC XML, COCO JSON, Tensorflow Object Detection TFRecords та інших форматах, що забезпечує більшу зручність і доступність.

Параметр	**Значення**
Загальна кількість зображень	15 000
Загальна кількість класів	11
Екземпляри об'єктів	97,942
Роздільна здатність	1920× 1200 пікселів
Типи класів	Машина, світлофор, вантажівка, водій, пішохід
Метрики оцінки	Точність, пригадування, оцінка F1

Рисунок 2.9 – Основні характеристики набору даних Udacity Self Driving Car Dataset

9. BDD100K (рисунок 2.10). BDD100K – це масивний набір відео та зображень, який використовується в розробці комп'ютерного зору та дослідженнях. Він був розроблений UC Berkeley і Baidu Research і оприлюднений у 2017 році.

Параметр	**Значення**
Загальна кількість зображень	100 000
Загальна кількість класів	10
Екземпляри об'єктів	1,8 млн.
Роздільна здатність	1280×720 пікселів
Типи класів	Автомобіль, вантажівка, автобус, мотоцикл, пішохід, велосипед, світлофор, дорожній знак, людина на велосипеді чи мотоциклі
Метрики оцінки	Середня точність

Рисунок 2.10 – Основні характеристики набору даних BDD100

Отже, набори даних для розпізнавання образів мають безліч застосувань у сферах штучного інтелекту та комп'ютерного бачення:

Розпізнавання об'єктів. Навчання моделей для виявлення та класифікації об'єктів на зображеннях.

Сегментація зображень. Визначення меж об'єктів на зображеннях та їх виділення.

Класифікація. Розділення зображень на різні категорії, такі як класифікація тварин, транспортних засобів тощо.

Візуальний пошук. Використання схожості зображень для пошуку подібних об'єктів чи сцен.

Автоматична обробка зображень. Застосування фільтрів, виправлення зображень або генерація нових зображень.

Набори даних для розпізнавання образів — це ключовий елемент у навчанні та покращенні моделей штучного інтелекту. Вони допомагають створювати моделі, що можуть розуміти та аналізувати візуальну інформацію навіть у складних умовах.

Завдяки підтримці передових досліджень і великих загальнодоступних наборів даних за останні роки область виявлення об’єктів значно розвинулася. Ці набори даних відіграють важливу роль в допомозі у навчанні, оцінці та вдосконаленні алгоритмів виявлення об’єктів.

2.3 Методи глибокого навчання в задачах управління роботизованними автомобілями

Автономні транспортні засоби виступають як трансформаційна технологія з широкими наслідками для майбутніх розумних міст, це революція для транспорту та оптимізація міської мобільності.

Виявлення об'єктів відіграє ключову роль у роботі автономних транспортних засобів, точно ідентифікуючи пішоходів, транспортні засоби та дорожні знаки для безпечного пересування. Підходи, засновані на глибокому навчанні, покращили підхід для виявлення об'єктів, використовуючи глибокі нейронні мережі для вилучення складних ознак з візуальних даних, що дозволяє досягти високої продуктивності в різних областях.

Дводвохетапні алгоритми, такі як R-FCN та Mask R-CNN, спрямовані на точну локалізацію об'єктів та сегментацію на рівні екземплярів, тоді як однодвохетапні алгоритми, такі як SSD, RetinaNet та YOLO, пропонують швидке розпізнавання через однопрохідну обробку. Для покращення результатів виявлення об'єктів для автономних транспортних засобів потрібні комплексні дослідження, особливо двох- та однодвохетапних алгоритмів.

Далі наводиться оцінка сильних та слабких сторін R-FCN, Mask R-CNN, SSD, RetinaNet та YOLO алгоритмів у контексті автономних транспортних засобів.

Надійне виявлення об'єктів є важливим для того, щоб автономні транспортні засоби могли сприймати своє оточення, приймати обґрунтовані рішення та безпечно рухатися в складних середовищах. Виявляючи та відстежуючи об'єкти у реальному часі, автономні транспортні засоби можуть передбачати потенційні небезпеки, реагувати на них відповідно та забезпечувати безпеку як пасажирів, так і пішоходів.

Підходи на основі глибокого навчання виступають як домінуюча парадигма в виявленні об'єктів [100, 101]. Використовуючи можливості глибоких нейронних мереж, ці підходи революціонізували галузь, автоматично вивчаючи та вилучаючи складні ознаки з візуальних даних [102, 103]. Вони продемонстрували високу продуктивність в різних галузях, включаючи автономні транспортні засоби [104], системи

спостереження та робототехніку [101]. Методи виявлення об'єктів на основі глибокого навчання відкрили шлях для значних досягнень у точності та обробці в реальному часі, дозволяючи створювати більш надійні та ефективні системи автономного водіння.

Дводвохетапні та однодвохетапні алгоритми виявлення об'єктів є двома популярними категоріями в галузі виявлення об'єктів на основі глибокого навчання [105]. У категорії дводвохетапних алгоритмів отримали популярність алгоритми, такі як R-FCN та Mask R-CNN [106]. R-FCN зосереджується на точній локалізації об'єктів за допомогою карт оцінок, чутливих до положення, тоді як Mask R-CNN вводить сегментацію на рівні екземплярів разом із виявленням об'єктів. У категорії однодвохетапних алгоритмів широко визнані алгоритми включають SSD, RetinaNet та YOLO. Ці моделі працюють з одним проходом над вхідними даними, пропонуючи реальний час роботи [107]. SSD використовує підхід з багатьма масштабами та типовими ящиками-якорями, RetinaNet вирішує дисбаланс класів за допомогою фокусової функції втрат, а YOLO досягає ефективного виявлення об'єктів, одночасно передбачаючи місцезнаходження об'єктів та ймовірності класів.

Дана частина розділу спрямована на розгляд алгоритмів у динамічних та складних міських умовах, враховуючи погодні умови, різноманітних учасників руху та складні сценарії дорожнього руху. І, як похідна від цього - створення комплексної системи метрик для оцінки, оскільки звичайні метрики можуть бути не змозі врахувати унікальні, поодинокі виклики, що виникають для автономних транспортних засобів. Також потрібні нові метрики - ті, які звертають увагу на безпеку та продуктивність у реальному часі, щоб надати точнішу оцінку ефективності алгоритмів у цьому вимірі.

Мета полягає в тому, щоб провести глибокий аналіз методів глибокого навчання для виявлення об'єктів, з фокусом на двох- та

однодвохетапних алгоритмах. Перш за все потрібно провести огляд попередніх досліджень, визначення їх внеску у науку та оцінку продуктивності, ефективності та застосування цих алгоритмів у контексті саме автономних транспортних засобів.

В дослідженні [108] проведено аналіз ефективності алгоритмів виявлення об'єктів для відеоспостереження за дорожнім рухом, з основним фокусом на використанні нейронних мереж. Вона оцінює ефективність та продуктивність різних алгоритмів на основі нейронних мереж у виявленні та відстеженні об'єктів у дорожніх сценах. Здійснюючи аналіз метрик продуктивності цих алгоритмів, стаття допомагає зрозуміти їх переваги, обмеження та застосовність у відеоспостереженні за дорожнім рухом.

У дослідженні [109] розглядалась реалізація системи реального часу для виявлення дорожніх знаків та об'єктів на дорозі за допомогою мобільних платформ, що базуються на GPU. Основна мета дослідження - розробка ефективного та надійного алгоритму, який може точно ідентифікувати та класифікувати дорожні знаки та інші об'єкти в реальному часі. Використовуючи мобільні платформи на базі GPU, система забезпечує високу продуктивність обробки та реактивність. Стаття докладно описує реалізацію, оцінку продуктивності та практичні наслідки запропонованого підходу.

Стаття, описана у [110], розкриває порівняльний аналіз алгоритмів на базі глибокого навчання для виявлення об'єктів на дорозі. Дослідження фокусується на методах двох- та однодвохетапного виявлення об'єктів, детально розглядаючи їх переваги, обмеження та продуктивність у різних ситуаціях застосування. Розглянуті алгоритми включають R-FCN, Mask R-CNN, SSD, RetinaNet та YOLO.

У статті, згаданій у [111], досліджується виявлення малих об'єктів у системах автономного водіння, використовуючи алгоритм YOLOv4. Основна мета дослідження - точне виявлення малих об'єктів, таких як

пішоходи або дорожні знаки, що є важливими для безпечного автономного водіння. Використовуючи YOLOv4, стаття пропонує підхід, що покращує продуктивність виявлення малих об'єктів у реальному часі. Дослідження оцінює ефективність алгоритму YOLOv4 та його застосовність у системах автономного водіння.

Класифіковані як двоетапні детектори об'єктів, архітектури глибокого навчання використовуються для виявлення об'єктів на зображеннях. Зазвичай вони мають два головні етапи: генерацію пропозицій регіонів та подальшу класифікацію або вдосконалення об'єктів. Ці детектори широко використовуються завдяки їхній здатності точно визначати розміщення та класифікувати об'єкти на зображеннях зі складними фонами. Згідно з [110], двома популярними двохетапними детекторами об'єктів є R-FCN та Mask R-CNN.

R-FCN – це модель виявлення об'єктів, яка діє в повністю згортковому режимі. Перший етап R-FCN включає у себе створення набору пропозицій регіонів за допомогою зовнішнього алгоритму, такого як Selective Search. Для зменшення пропозицій регіонів у R-CNN використовується жадібний алгоритм, який називається вибірковим пошуком. Вибірковий пошук - це жадібний алгоритм, який комбінує менші сегментовані регіони, щоб згенерувати пропозиції регіонів. Ці пропозиції регіонів визначають можливі місця знаходження об'єктів. На другому етапі R-FCN проводить класифікацію та покращення об'єктів, використовуючи інформацію про регіони. Замість використання повністю зв'язаних шарів, R-FCN використовує карти балів, чутливі до положення, які обчислюються згортками. Ці карти балів закодовують ймовірності класів у різних точках кожної пропозиції регіону. У завершенні застосовується операція пулінгу, чутлива до положення, для отримання фіксованого вектору ознак для кожного класу. R-FCN показує найкращі показники точності виявлення об'єктів, демонструючи більш

ефективне обчислення порівняно з іншими двохетапними детекторами. На рисунку 2.11 наведено архітектуру R-FCN.

Mask R-CNN є похідною від Faster R-CNN і також складається з двох ключових етапів. Перший етап включає у себе створення пропозицій регіонів за допомогою мережі RPN (Region Proposal Network), аналогічно до Faster R-CNN. Ці пропозиції піддаються подальшій обробці та класифікації на другому етапі, подібно до алгоритму Faster R-CNN. Але Mask R-CNN додає ще одну гілку, яка відповідає за передбачення індивідуальних масок для кожного обраного регіону інтересу (RoI). Ця гілка генерує бінарні маски, які точно відображають контур кожного об'єкта. Такий підхід дозволяє Mask R-CNN одночасно вирішувати завдання виявлення об'єктів і сегментації, надаючи потужні можливості для широкого спектру застосувань, включаючи сегментацію та відстеження об'єктів. На рисунку 2.12 показана архітектура Mask R-CNN.

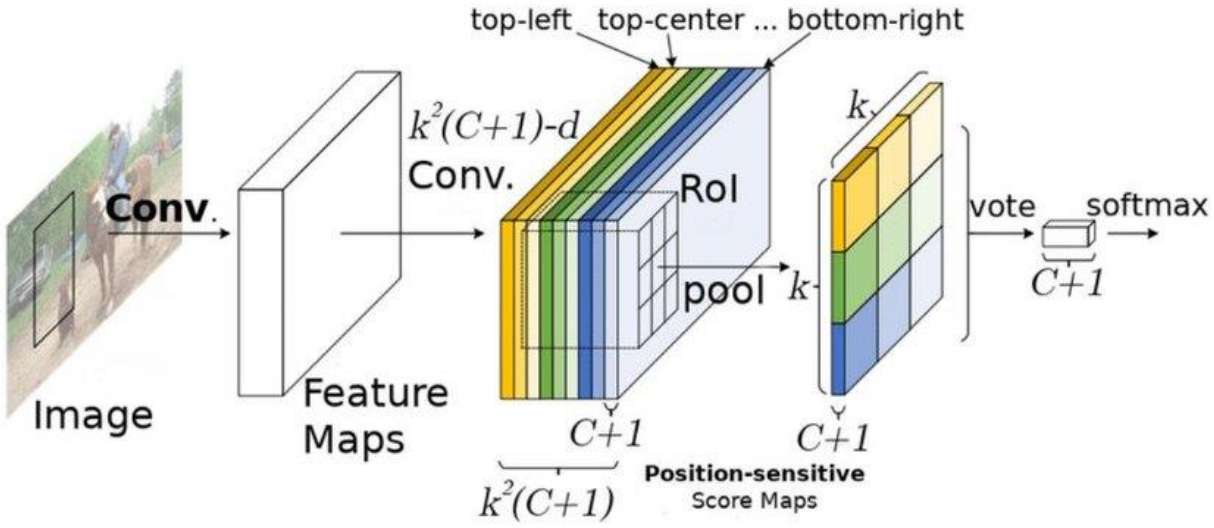

Рисунок 2.11 – Архітектура R-FCN

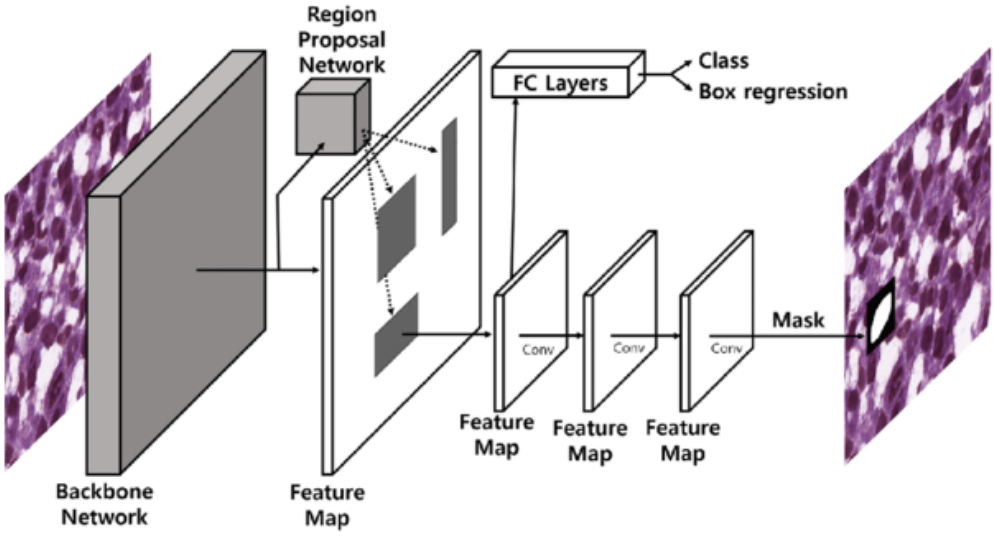

Рисунок 2.12 – Архітектура Mask R-CNN

Одно етапні детектори об'єктів – це підтип архітектури глибокого навчання, спрямований на завдання виявлення об'єктів. На відміну від двохетапних моделей, вони негайно прогнозують межі об'єктів та ймовірності класів за один прохід без необхідності явного створення пропозицій регіонів. Це робить одно етапні детектори швидшими та ефективнішими, що робить їх відмінними для застосувань у реальному часі. Дослідження [110] розглядає три популярних представників одно етапних детекторів об'єктів: Single Shot MultiBox Detector (SSD) [112], RetinaNet [113] та You Only Look Once (YOLO) [114].

SSD, або Single Shot MultiBox Detector, є ефективною моделлю для виявлення об'єктів у один етап, яка забезпечує збалансованість між високою точністю та швидкістю обробки у реальному часі. Підхід, застосований у SSD, полягає в розділенні вхідного зображення на сітку різних розмірів. Кожна комірка у цій сітці відповідає передбаченню меж об'єктів та ймовірностей класів для об'єктів у відповідній області. Цей підхід дозволяє SSD ефективно впізнавати об'єкти різних розмірів. Крім того, SSD використовує базові якорні коробки з різними

співвідношеннями сторін та масштабами, що поліпшує точність локалізації об'єктів. Завдяки послідовності згорткових шарів, які поступово зменшують просторові розміри, SSD ефективно передбачає межі об'єктів та ймовірності класів на різних масштабах за один прохід. Архітектуру SSD можна розглянути на рисунку 2.13.

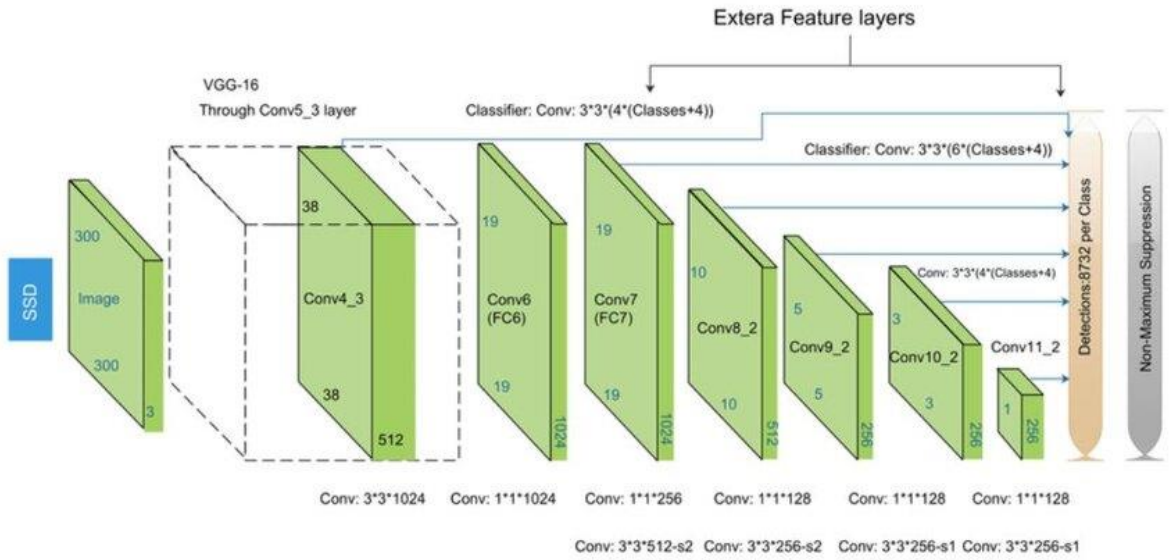

Рисунок 2.13 – Архітектура SSD

RetinaNet є потужною системою виявлення об'єктів у один етап, яка ефективно вирішує проблему незбалансованих класів під час тренування. Ця система використовує унікальну функцію втрат, відому як Focal Loss, яка фокусує увагу на складних прикладах, що неправильно класифікуються або важко класифікувати. Focal Loss надає менші ваги легким прикладам, які вже добре класифіковані, дозволяючи моделі більше уваги приділяти складним прикладам під час процесу навчання. Це дозволяє RetinaNet забезпечити кращий баланс між точністю та ефективністю. Як і SSD, RetinaNet використовує мережу функцій-пірамід (FPN), яка захоплює багаторозмірні ознаки для ефективного виявлення об'єктів. FPN інтегрує ознаки з різних рівнів

піраміди ознак для ефективної роботи з об'єктами різних розмірів. Ця інтеграція Focal Loss та використання мережі функцій-пірамід дозволяють RetinaNet досягати значного успіху в розв'язанні задач виявлення об'єктів, особливо в умовах, коли виникають проблеми з незбалансованими даними чи складними об'єктами. Архітектура RetinaNet представлена на рисунку 2.14.

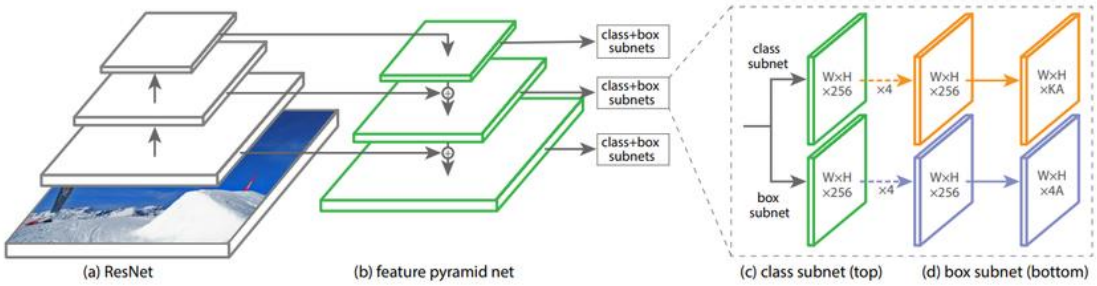

Рисунок 2.14 – Архітектура RetinaNet

YOLO є відмінним інструментом при виявлені об'єктів у реальному часі, використовує іншу концепцію порівняно з SSD та RetinaNet. Він опирається на одну нейромережу, що одночасно визначає місця розташування об'єктів та ймовірності класифікації, що прискорює процес обчислень. Ще одна особливість полягає в застосуванні різних типів для урахування різноманітних пропорцій та розмірів об'єктів. Ранні версії YOLO зазнавали труднощів із точним розпізнаванням невеликих об'єктів. Однак в YOLOv4 вирішили це, покращивши швидкість та продуктивність системи.

Аналіз продуктивності двоетапних та одноетапних детекторів об'єктів, а також аналіз продуктивності детекторів об'єктів на основі YOLO:

Крива PR (precision-recall) надає нам інформацію про продуктивність алгоритмів виявлення об'єктів [110]. Шляхом розгляду

цієї кривої можна оцінити баланс між точністю та повторюваністю й здійснити розумні порівняння між різними моделями.

На рисунку 2.15 відзначається, що YOLOv4 стабільно випереджає інші моделі за продуктивністю. Він досягає найвищих показників точності та повторюваності на всіх рівнях, демонструючи ефективність у точному виявленні об'єктів у різних умовах. Двоетапна модель виявлення Mask R-CNN відзначається помітною точністю та повторюваністю, перевищуючи RetinaNet, R-FCN та SSD, що свідчить про загальну вищу точність та ефективність виявлення.

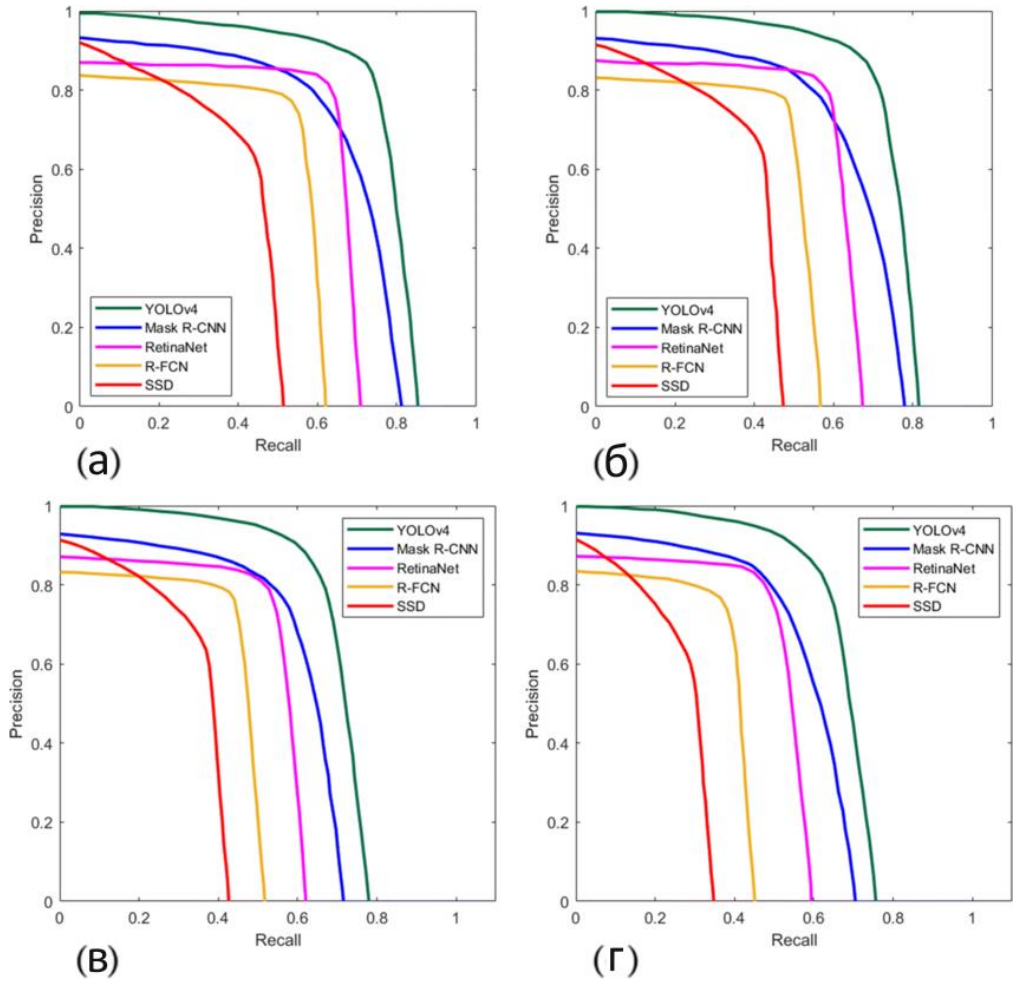

Рисунок 2.15 – Криві: точність-повторюваність (Precision-Recall, PR) для моделей виявлення об'єктів: а) транспортний засіб; б) пішохід в) дорожній знак г) світофор

Порівнюючи R-FCN з SSD, можна відзначити, що R-FCN перевершує SSD за показниками точності виявлення об'єктів на всіх рівнях. Це свідчить про те, що R-FCN надає більш точні результати виявлення, підвищуючи надійність цього процесу. Крім того, повторюваність Mask R-CNN практично збігається з YOLOv4, особливо у виявленні об'єктів з затемненням та обрізанням, що свідчить про можливість точного виявлення об'єктів Mask R-CNN у складних сценаріях з наявністю затемнення та обрізання.

Однак SSD демонструє найнижчу повторюваність серед усіх моделей, що свідчить про більший рівень пропущених виявлень об'єктів на різних рівнях. Це може свідчити про те, що SSD може мати проблеми з точністю виявлення об'єктів порівняно з іншими моделями.

Отже, згідно з аналізом кривої PR, YOLOv4 стає найбільш продуктивною моделлю загалом, за ним йде Mask R-CNN. R-FCN перевершує SSD за точністю, хоча SSD має найнижчу повторюваність. Ці висновки можуть стати корисними для вибору належного алгоритму виявлення об'єктів відповідно до конкретних вимог та пріоритетів.

Короткий опис різних характеристик моделей виявлення об'єктів наведений в табл. 2.1.

Таблиця 2.1 – Короткий опис характеристик моделей виявлення об'єктів

Модель	**Точність**	**Швидкість**	**Складність**	**Ефективність**
R-FCN	низька	повільна	середня	низька
Mask R- CNN	середня	повільна	висока	низька
SSD	низька	швидкий	низька	висока
RetinaNet	средня	помірна	висока	середня
YOLOv4	**висока**	**швидкий**	**середня**	**висока**

2.4 Аналіз результатів дослідження

В досліджені [115], розглядається ефективність моделей об'єктного виявлення YOLO на різних архітектурах CPU та GPU. З метою визначення найшвидших моделей YOLO для кожної GPU, враховуючи швидкість та продуктивність, базові моделі YOLO перевіряються на GPU від NVIDIA, таких як TESLA P100, TESLA V100, GTX 1080Ti та RTX 4090. Це дослідження надає корисні вказівки для вибору найбільш підходящої моделі YOLO з урахуванням конфігурацій апаратного забезпечення та вимог реальних застосувань.

Моделі YOLO призначені для точного виявлення об'єктів в реальному часі, використовуючи розбиття вхідного зображення на сітку для передбачення обмежувальних рамок – прямокутників та ймовірностей класів для кожної комірки сітки. Різні версії YOLO, такі як YOLOv4, YOLOv5, YOLOv6 та YOLOv7, пропонують компроміси між швидкістю та точністю. Наприклад, варіанти Nano та Tiny акцентують легкість та швидку продуктивність, тоді як наступні версії, наприклад YOLOv7, забезпечують вищу точність, виходячи за межі невеликого зниження швидкості. Цей аналіз продемонструє ефективність цих моделей на вказаних GPU від NVIDIA, щоб визначити найшвидшу модель для кожного пристрою GPU, допомагаючи вибрати оптимальну модель YOLO, яка збалансує швидкість, точність та конкретні вимоги апаратного забезпечення.

Як показано на рисунку 2.16, графік надає інформацію про продуктивність різних моделей YOLO на різних пристроях GPU. З цього графіка можна вивести обговорення найкращого методу з точки зору швидкості та пропускної здатності.

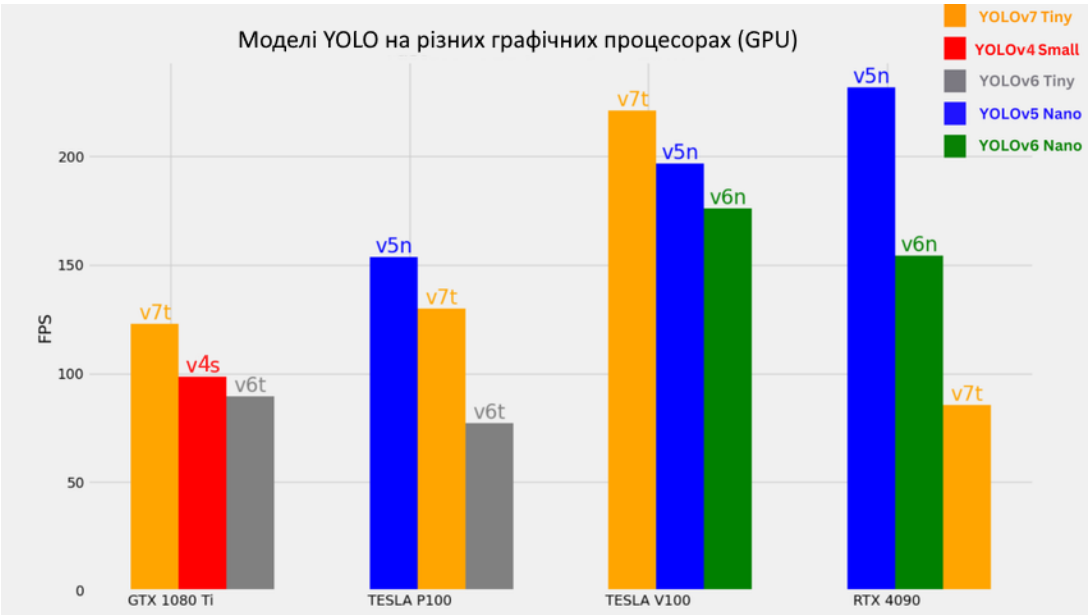

Рисунок 2.16 – Моделі YOLO на різних графічних платформах

По-перше, з графіка видно, що YOLOv5 видає найкращі результати за швидкістю на GPU RTX 4090 та TESLA P100. Це свідчить про те, що, якщо основний критерій - досягнення найвищої кількості кадрів на секунду (FPS) для точного виявлення в реальному часі, YOLOv5 Nano стане оптимальним вибором. По-друге, YOLOv7 Tiny виділяється як модель, яка надає найвищу пропускну здатність на GTX 1080 Ti та TESLA V100. Пропускна здатність визначається кількістю об'єктів, виявлених за одиницю часу, і YOLOv7 Tiny в цьому виявляється найкращим для цих конкретних пристроїв GPU. Це особливо корисно в сценаріях, коли точне виявлення більшої кількості об'єктів важливіше, ніж досягнення максимальної кількості кадрів на секунду.

На противагу, моделі YOLOv6 Nano та Tiny, хоч і не досягають такої ж швидкості, що YOLOv5 та YOLOv7, але все ж не вважаються дуже повільними. Хоча графік не містить точних даних про їхню продуктивність, він вказує на те, що ці моделі забезпечують баланс між швидкістю та точністю. Вони можуть бути чудовим вибором, коли

потрібна помірна швидкість, при цьому отримуючи задовільні результати об'єктивного виявлення.

Узагальнюючи, кращий вибір залежить від конкретних вимог завдання. YOLOv5 Nano ідеально підходить для реальних застосувань, де важлива максимальна кількість кадрів на секунду. YOLOv7 Tiny видається корисним у сценаріях, де важлива висока пропускна здатність більше, ніж продуктивність в реальному часі. Тим часом моделі YOLOv6 Nano і Tiny пропонують компроміс між швидкістю та точністю, що робить їх прийнятними у випадках, коли потрібна помірна швидкість без значних втрат у якості виявлення.

Як зображено на рисунку 2.17, на платформі процесорів, модель YOLOv5 отримує найвищий показник швидкості. Ця модель може працювати в реальному часі, перевищуючи 30 кадрів на секунду. Це означає, що вона може обробляти та аналізувати зображення або відеопотоки в реальному часі, що надає швидкі результати об'єктивного виявлення. Модель YOLOv5 добре оптимізована для ефективності та швидкості, що робить її добре пристосованими для процесорів споживчого класу, де реальний час має велике значення.

Згідно з наданими результатами виявлено, що менші моделі, як правило, проявляють вищу продуктивність. Це виявлено у тестах, де моделі YOLOv5 Nano та YOLOv6 Nano виявилися найшвидшими альтернативами. Важливо зазначити, що навіть на застарілій версії процесора i7 ці моделі здатні досягти вражаючої швидкості понад 30 кадрів на секунду (FPS). Це підтверджує ефективність та оптимізацію моделей YOLOv5, v6 Nano для обробки на процесорах масового споживання, роблячи їх чудовими виборами для об'єктивного виявлення в реальному часі на таких процесорах.

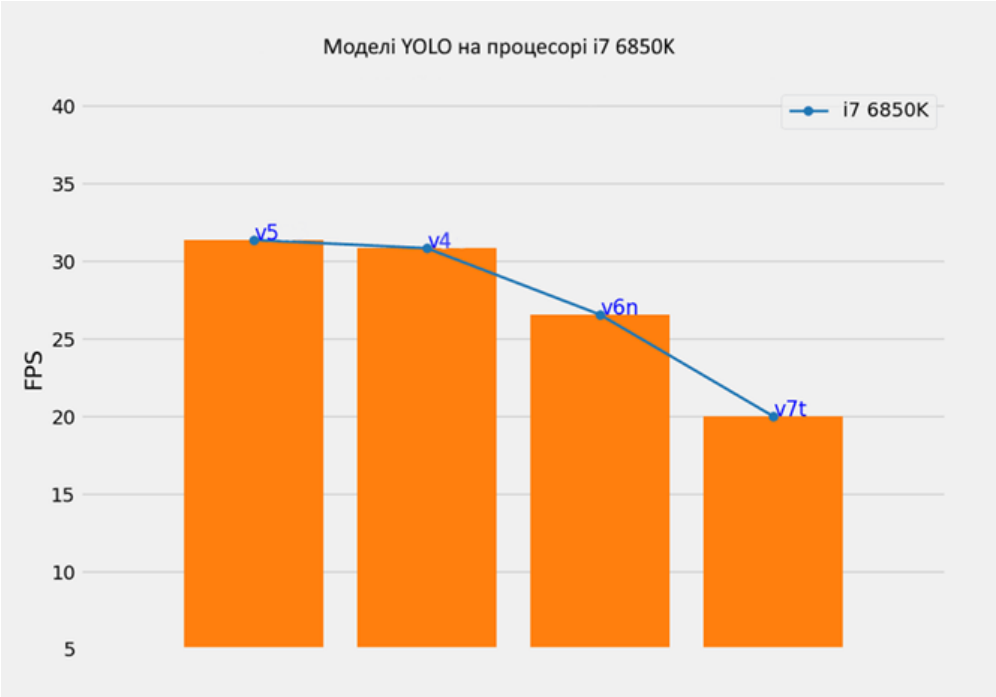

Рисунок 2.17 – Чотири найшвидші моделі виявлення об'єктів YOLO на процесорі i7 6850K

2.5 Висновки до розділу 2

Дане дослідження було направлене на ретельний та всебічний аналіз методів глибокого навчання для виявлення об'єктів, зосереджуючись зокрема на двох та одноетапних підходах. Основною метою цього дослідження було визначення найбільш перспективних алгоритмів у цій сфері та надання цінної інформації про їхні унікальні переваги, обмеження та загальну продуктивність. Особлива увага була приділена порівнянню та контрастуванню моделей виявлення об'єктів YOLO, зокрема YOLOv4, YOLOv5, YOLOv6 та YOLOv7, за показниками кадрів на секунду (FPS) та точності. Для забезпечення достовірності результатів, це дослідження виконувало експерименти на різних моделях GPU від NVIDIA, таких як GTX, RTX та TESLA. Ця оцінка дала

можливість покластися на нашу аналітику та здійснити значні порівняння між різними версіями YOLO. Знайдені у цьому дослідженні висновки сприяють кращому розумінню методів глибокого навчання виявлення об'єктів, дозволяючи дослідникам та фахівцям здійснювати обґрунтовані рішення при виборі найбільш підходящих алгоритмів для своїх конкретних завдань. Один із потенційних напрямків майбутніх досліджень може стати вивчення об'єднання алгоритмів виявлення об'єктів двох та одного етапу для використання їхніх відповідних переваг та покращення загальної продуктивності. Ще одним перспективним напрямком для майбутніх досліджень є адаптація алгоритмів виявлення об'єктів для використання на краї мережі з метою оптимізації моделей для ресурсозберігаючих пристроїв та забезпечення об'єктивного виявлення об'єктів у реальному часі.

Двоетапні детектори, як правило, відзначаються вищою точністю виявлення порівняно з одноетапними. Згідно з експериментальними результатами, одноетапний алгоритм виявлення YOLOv5 Nano демонструє видатну продуктивність в швидкості та точності, а також високу енергоефективність порівняно з іншими алгоритмами виявлення дорожніх об'єктів. Це досягається завдяки використанню CSPDarkNet-53, що підвищує точність класифікатора і детектора.

YOLOv5 збалансований алгоритм, надаючи високу точність при швидкому виявленні об'єктів на всіх рівнях категорій дорожніх об'єктів.

З урахуванням проведених досліджень, YOLOv5 виявляє найкращу швидкість роботи, що робить її найбільш оптимальним варіантом для сценаріїв, де вирішальне значення має максимальна кількість кадрів на секунду для точного виявлення в реальному часі.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ

3.1 Вимоги до системи розпізнавання об’єктів

3.1.1 Загальні положення

Комп'ютерне бачення, ключова область штучного інтелекту, охоплює вивчення комп'ютерів та програм, здатних розрізняти та розуміти зображення та ситуації. Воно включає кілька компонентів, таких як ідентифікація зображень, виявлення об'єктів, синтез зображень та інші.

Ідентифікація об'єктів є ключовим аспектом комп'ютерного зору через його широкий спектр практичних застосувань. Цей термін охоплює здатність програмних систем точно визначати просторові координати об'єктів на зображенні або в сцені та класифікувати та маркувати кожен ідентифікований об'єкт.

Система розпізнавання об’єктів в системі управління роботизованими автомобілями – це критично важлива складова, що дозволяє автономним транспортним засобам "бачити" та інтерпретувати їхнє навколишнє середовище з високою точністю та надійністю. Використовуючи комбінацію сенсорів, таких як камери, лідари, радари та ультразвукові датчики, система аналізує вхідні дані для ідентифікації та класифікації об'єктів, таких як інші автомобілі, пішоходи, велосипедисти, дорожні знаки та перешкоди.

В межах дослідження рекомендується використовувати модель YOLOV5 як потенційне рішення для системи онлайн ідентифікації об'єктів.

Системні вимоги:

– мова програмування Python версії 3.7;

– Anaconda версії 3.7;

– Jupiter notebook або Google Colab.

Вимоги до обладнання:

– операційна система: Windows, Linux;

– процесор: Intel i5;

– пам’ять: 8gb;

– жорсткий диск: 250gb.

3.1.2 Функціональні вимоги

Функціональні вимоги:

– Система має можливість виявляти об'єкти в режимі реального часу за допомогою моделі YOLOv5 з частотою кадрів не менше 30 кадрів в секунду (FPS), , при цьому затримка обробки одного кадру не повинна перевищувати 33 мілісекунди.

– Система повинна класифікувати об'єкти з точністю розпізнавання не меншою за 95% для таких категорій: автомобілі, пішоходи, велосипедисти, та статичні перешкоди.

– Система повинна визначати відстань до виявлених об'єктів з точністю до 10 см на дистанції до 100 метрів.

– Система повинна оцінювати швидкість рухомих об'єктів з точністю до ±5 км/год для швидкостей до 120 км/год.

– Система повинна мати вбудовані механізми відновлення після збоїв та аварійних ситуацій з часом відновлення функціональності не більше 1 години.

Крім того, система повинна забезпечити взаємодію з системами навігації, управління та аварійного реагування автомобіля. Обробка вхідних даних повинна відбуватися у режимі реального часу для забезпечення негайного реагування із забезпеченням високого рівня надійності, включно з резервними механізмами для запобігання відмов.

3.1.3 Нефункціональні вимоги

Нефункціональні вимоги:

- Система повинна демонструвати низьку затримку та здатність зберігати частоту кадрів не менше 30 кадрів в секунду для виявлення об'єктів у режимі реального часу.
- Точність системи при класифікації та виявленні об'єктів повинна бути не менше 90%.
- Система має забезпечувати обробку даних у режимі реального часу з максимальною затримкою не більше 1 сек.
- Система повинна бути доступна для використання 24/7 з допустимим часом простою не більше 1% на рік.
- Система повинна включати заходи безпеки для запобігання несанкціонованому доступу та маніпуляціям з даними.
- Мають бути реалізовані резервні механізми та стратегії відновлення у разі відмови або помилок.
- Система має підтримувати автоматичне масштабування ресурсів при збільшенні обсягу даних на 50% від початкового навантаження, зберігаючи при цьому існуючу швидкість обробки без зниження продуктивності.
- Інтерфейси повинні бути інтуїтивно зрозумілими та легкими в користуванні для операторів та технічного персоналу.

3.2 Системна архітектура

Архітектура системи для інтелектуального виявлення об’єктів у реальному часі на базі YOLOv5 включає кілька ключових компонентів, які взаємодіють між собою для оптимальної функціональності (рис. 3.1):

Джерела вхідних даних. Цей компонент включає в себе стрічки з камер в реальному часі та відеопотоки з попередньо записаних джерел. Він забезпечує надходження вхідних даних у систему.

Модуль виявлення об'єктів. Основний модуль, що використовує YOLO для точного виявлення та класифікації об’єктів у реальному часі.

Підтримка настоюваних об’єктів. Цей компонент надає можливість навчання та додавання користувацьких класів об’єктів для більш широкого розпізнавання.

Модуль попередньої обробки відповідає за покращення зображень для оптимізації процесу виявлення об’єктів шляхом налаштування розміру, зменшення шуму та підвищення контрастності.

Синхронізація кадрів забезпечує синхронізацію кадрів з різних камер для вирівнювання даних та забезпечення їхньої правильної обробки.

Модуль відстеження об'єктів відстежує та моніторить виявлені об’єкти за допомогою алгоритмів відстеження, наприклад, DeepSORT, що допомагає відслідковувати об’єкти в часі.

Інтерфейс користувача забезпечує зручний спосіб налаштування системи, моніторингу та перегляду результатів виявлення об’єктів через веб-панель.

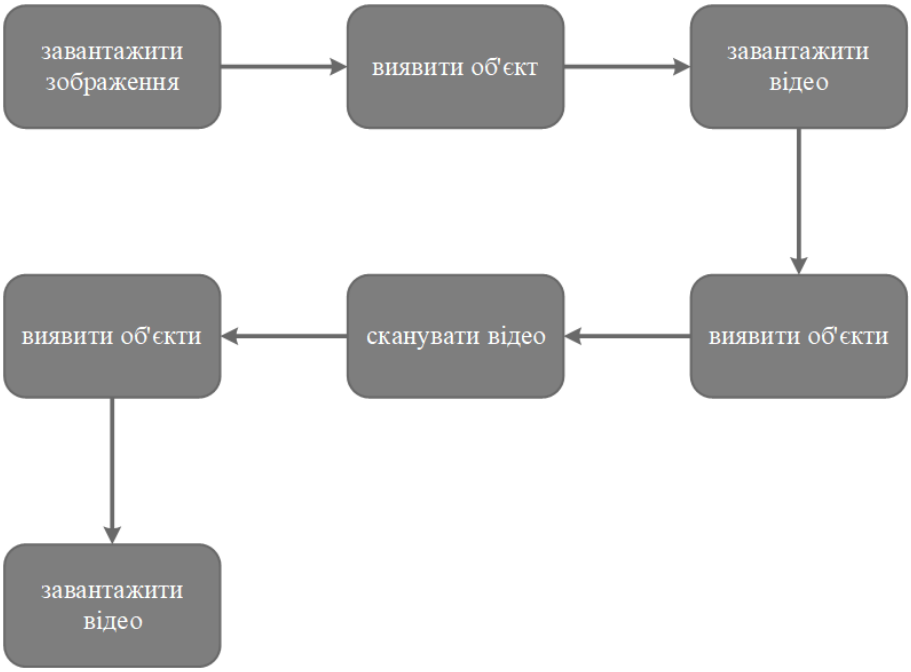

Рисунок 3.1 – Схема робочого процесу системи розпізнавання об'єктів

Ця архітектура надає основу для розробки системи інтелектуального виявлення об’єктів у реальному часі з використанням YOLOv5. Вона може бути налаштована та розширена відповідно до конкретних вимог та потреб для досягнення максимально ефективних результатів.

3.3 Діаграма потоків даних

Діаграма потоку даних (DFD) – це інструмент моделювання, який використовується для ілюстрації вхідних даних, операцій обробки та результатів, отриманих системою. Вона відображає різні елементи системи, такі як системні процеси, дані, з якими працює система,

зовнішні об'єкти та потоки інформації всередині системи. DFD дозволяє уявно переглядати та відображати потік інформації та перетворення даних всередині системи.

Цей інструмент використовує графічний підхід для відображення потоку даних та наступних перетворень, що відбуваються при переміщенні даних від вводу до виводу. DFD може бути використана для опису системи на різних рівнях абстракції, розглядаючи зростання рівнів потоку інформації та функціональної складності. Це може бути поділено на окремі рівні, які відповідають зростанню складності системи.

На рисунках 3.2 та 3.3 представлені DFD діаграми для системи розпізнавання об'єктів, які ілюструють потік інформації та функціональні елементи цієї системи на відповідних рівнях представлення.

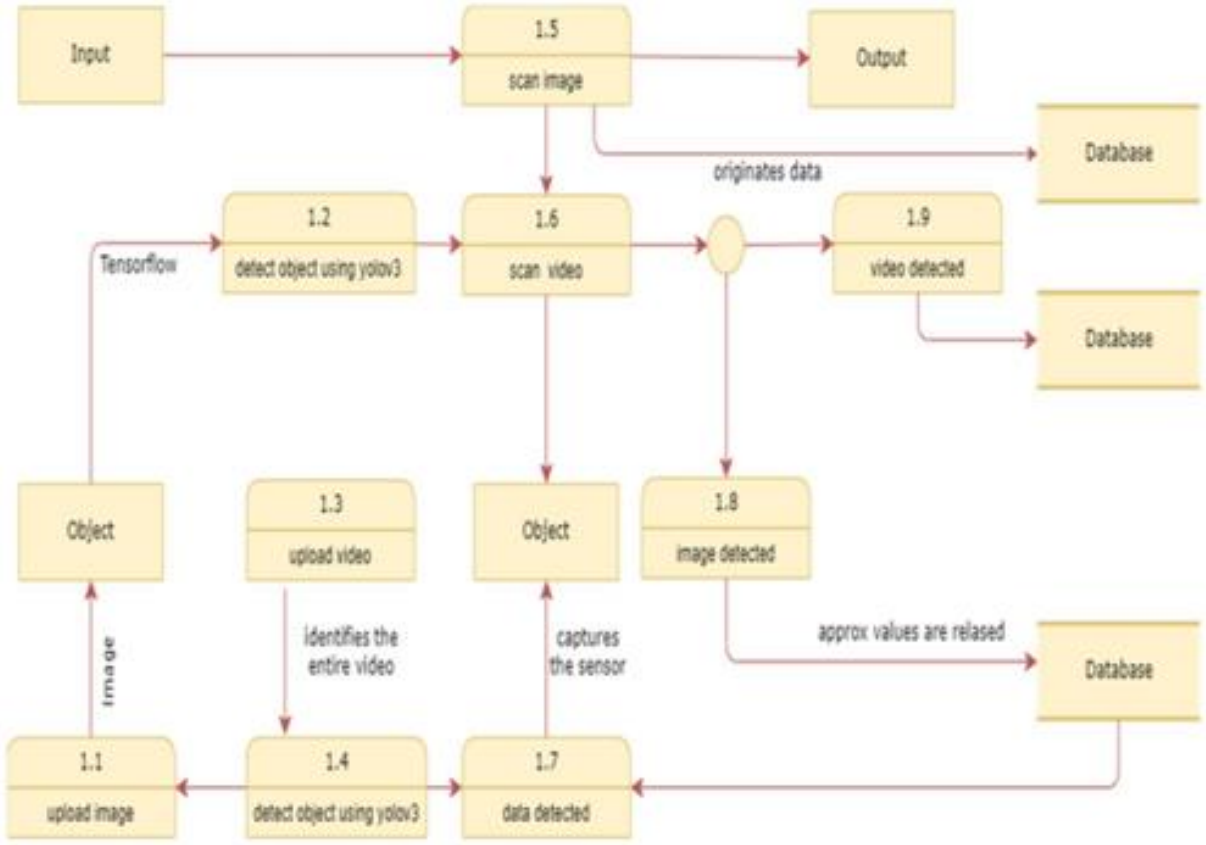

Рисунок 3.2 – DFD діаграма

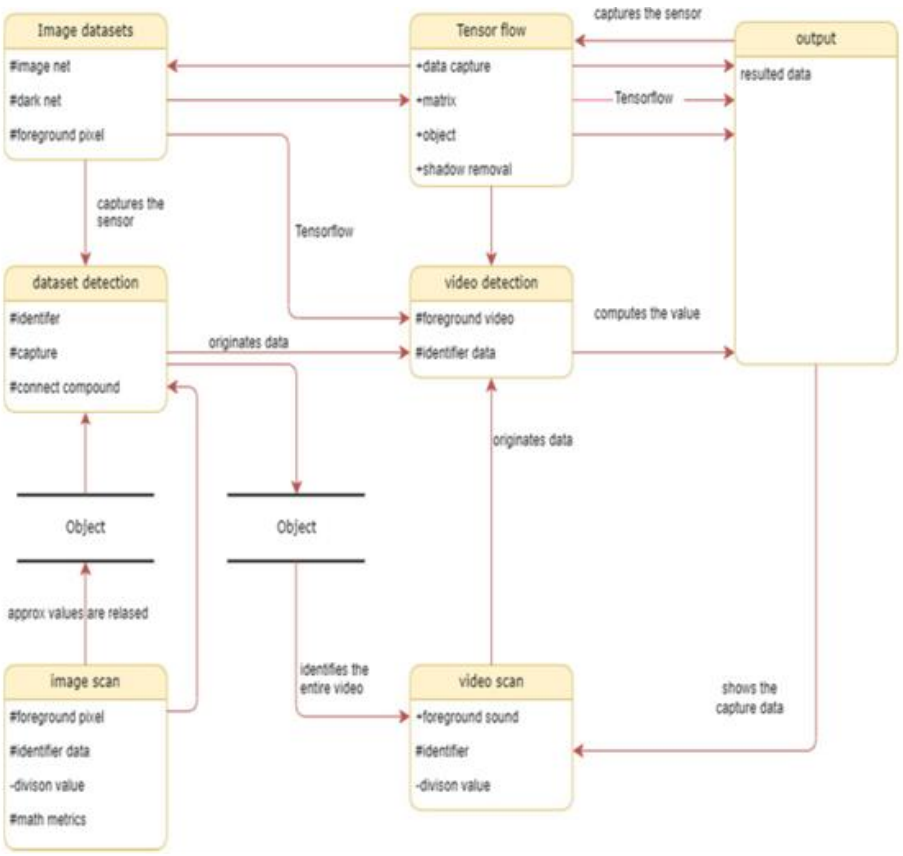

Рисунок 3.3 – Деталізація DFD діаграми

3.4 Проєктування програмних компонентів

3.4.1 Use-Case

Уніфікована мова моделювання (UML) є стандартизованою мовою для створення моделей програмного забезпечення. Ця мова використовується для визначення, візуалізації, побудови та документування артефактів програмних систем і бізнес-моделей.

Діаграма варіантів використання (use-case) – це тип поведінкової діаграми в рамках UML, що використовується для аналізу випадків використання системи. Її основна мета – надати графічне уявлення

функціональності системи з точки зору учасників, їх цілей та залежностей між варіантами використання.

На діаграмі варіантів використання представлені різні сценарії використання системи. Вона допомагає показати, які функції виконуються для кожного суб'єкта в системі. Це важливий інструмент для розуміння функціональних можливостей системи з точки зору користувачів чи зовнішніх агентів.

На рисунку 3.4 наведена діаграма варіантів використання для системи розпізнавання об'єктів, що ілюструє, як система взаємодіє з її користувачами або зовнішніми агентами через різні випадки використання. Це сприяє кращому розумінню того, як система використовується та які функції вона здійснює для своїх користувачів.

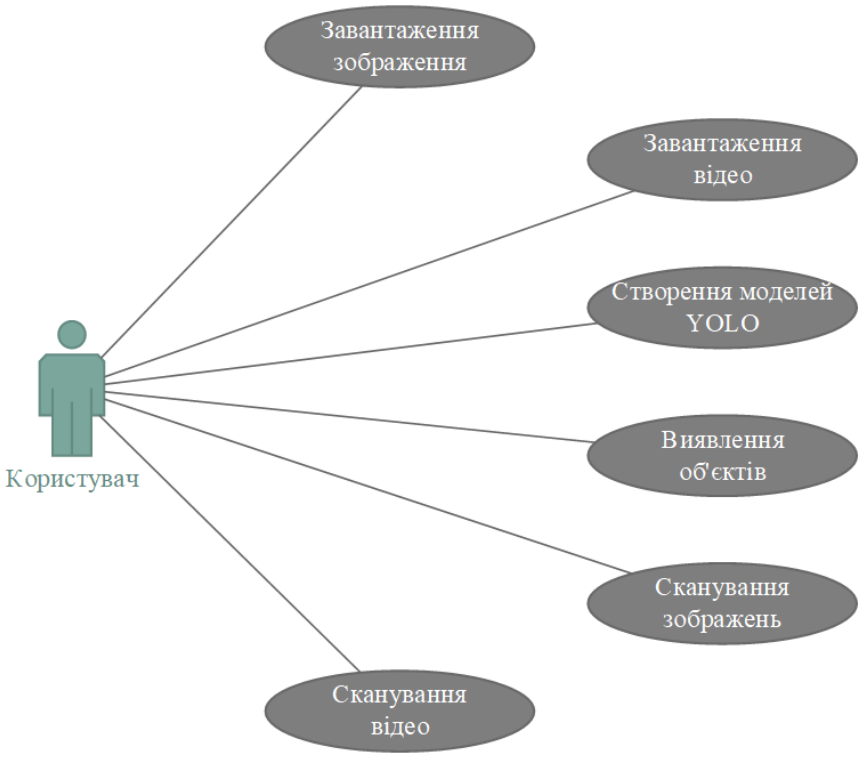

Рисунок 3.4 – Діаграма use-case

3.4.2 Діаграма класів

Діаграма класів в рамках UML використовується для уточнення структури системи, що виникає з випадків використання, та для детального опису класів, методів і атрибутів, які участь в цих сценаріях використання. Ця діаграма дозволяє класифікувати класи, що визначені на діаграмі випадків використання, і відобразити їх взаємозв'язки.

Кожен клас на діаграмі класів відображає окремий об'єкт або компонент системи. У цих класах визначаються методи (функції або операції), які виконуються цими класами, і атрибути (змінні), які вони мають. Взаємодія між цими класами і їхні зв'язки також можуть бути відображені на цій діаграмі.

На рисунку 3.5 надана діаграма класів для системи розпізнавання об'єктів, яка ілюструє класи системи, їхні методи та атрибути, а також взаємозв'язки між цими класами. Це допомагає виявити структуру системи та взаємодію між її складовими елементами для більш детального розуміння роботи системи. Діаграма відображає типи зв'язків між класами, такі як асоціації, залежності та спадковість, що дає змогу оцінити спосіб взаємодії компонентів системи та їх вплив на загальну поведінку програми. Такий підхід сприяє глибшому аналізу логіки роботи системи і може бути використаний для оптимізації дизайну та покращення масштабування. Чітке визначення структури та взаємозв'язків класів є ключовим аспектом при розробці модульних і легко адаптованих систем, що в свою чергу забезпечує легкість управління кодом та його подальшу підтримку. Діаграма класів слугує ключем до візуалізації ієрархії та взаємодій компонентів у системі. Такий підхід значно спрощує процес інтеграції нових модулів та тестування.

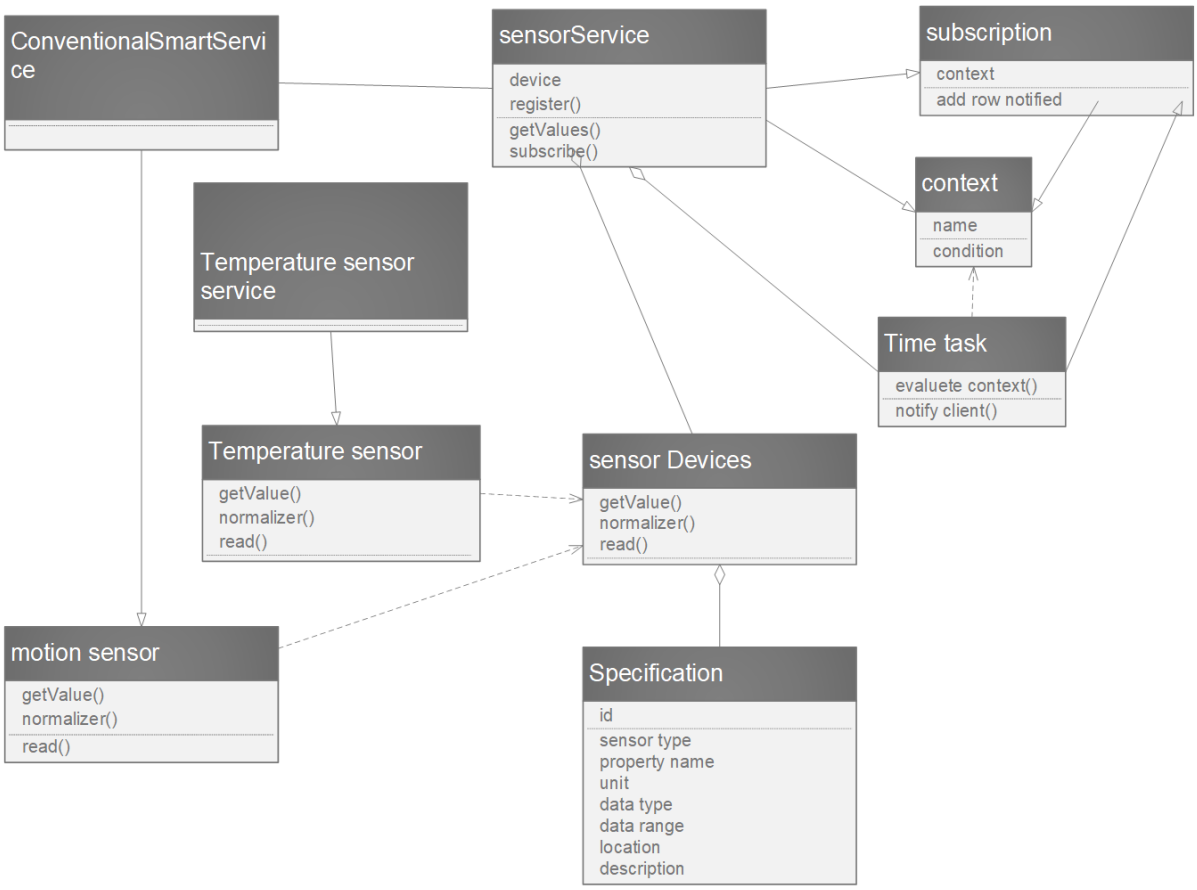

Рисунок 3.5 – Діаграма класів

3.4.3 Діаграма послідовності

Діаграма послідовності в уніфікованій мові моделювання (UML) використовується для візуалізації взаємодії між об'єктами системи відносно часу. Ця діаграма вказує послідовність повідомлень, що передаються між об'єктами під час виконання певної функції або сценарію.

На рисунку 3.6 представлені діаграми послідовності для системи розпізнавання об'єктів, які ілюструють взаємодію між різними компонентами системи. Лінії життя представляють об'єкти, які беруть участь у взаємодії, і показують їх активність відносно часу.

Повідомлення між об'єктами відображаються стрілками та показують порядок передачі повідомлень.

Діаграми послідовності допомагають уточнювати порядок виконання операцій та взаємодію об'єктів у конкретних сценаріях. Вони є корисним інструментом для визначення, як система реагує на певні події та взаємодіє з різними компонентами.

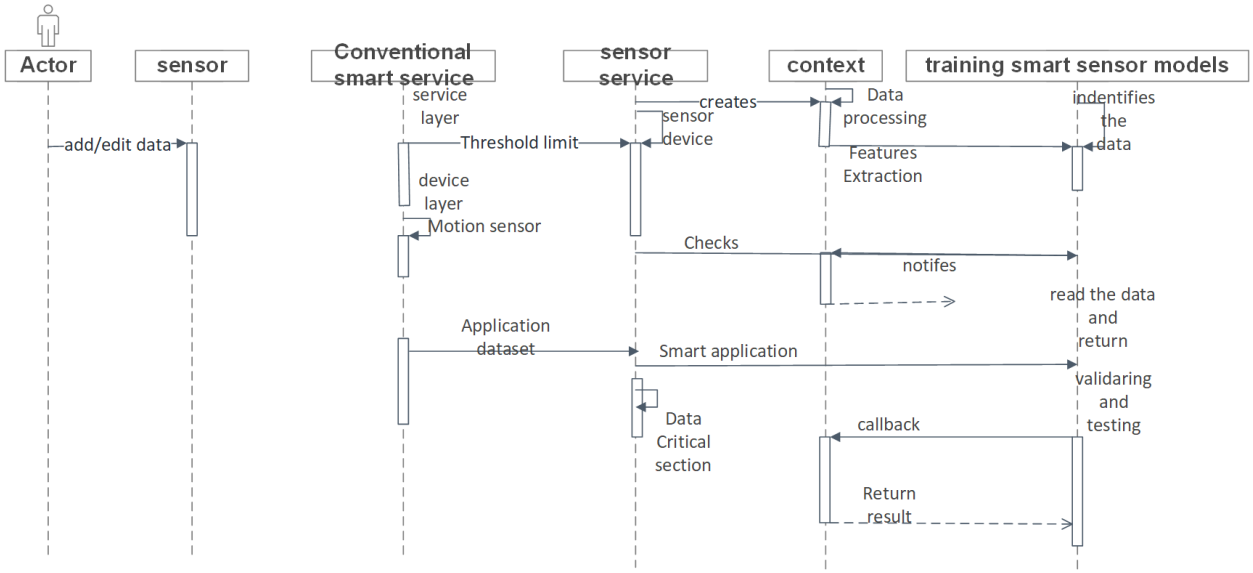

Рисунок 3.6 – Діаграма послідовності

3.4.4 Діаграма компонентів

Діаграма компонентів в UML використовується для візуалізації високорівневої архітектури системи та показує, які компоненти складають систему та як вони взаємодіють між собою.

Основні елементи діаграми компонентів включають компоненти (які представляють високорівневі частини системи), інтерфейси (що визначають, як компоненти взаємодіють), порти (точки взаємодії між

компонентами), асоціації (відносини між компонентами), артефакти (фізичні об'єкти, такі як файли чи бібліотеки), та залежності.

На діаграмі компонентів показано, як компоненти спілкуються один з одним через їхні інтерфейси та порти. Це дозволяє відобразити взаємодію між компонентами та їхню структуру на високому рівні.

Ця діаграма може служити основою для подальшого розроблення деталей архітектури системи та визначення, як компоненти реалізують функціональність системи на більш низькому рівні.

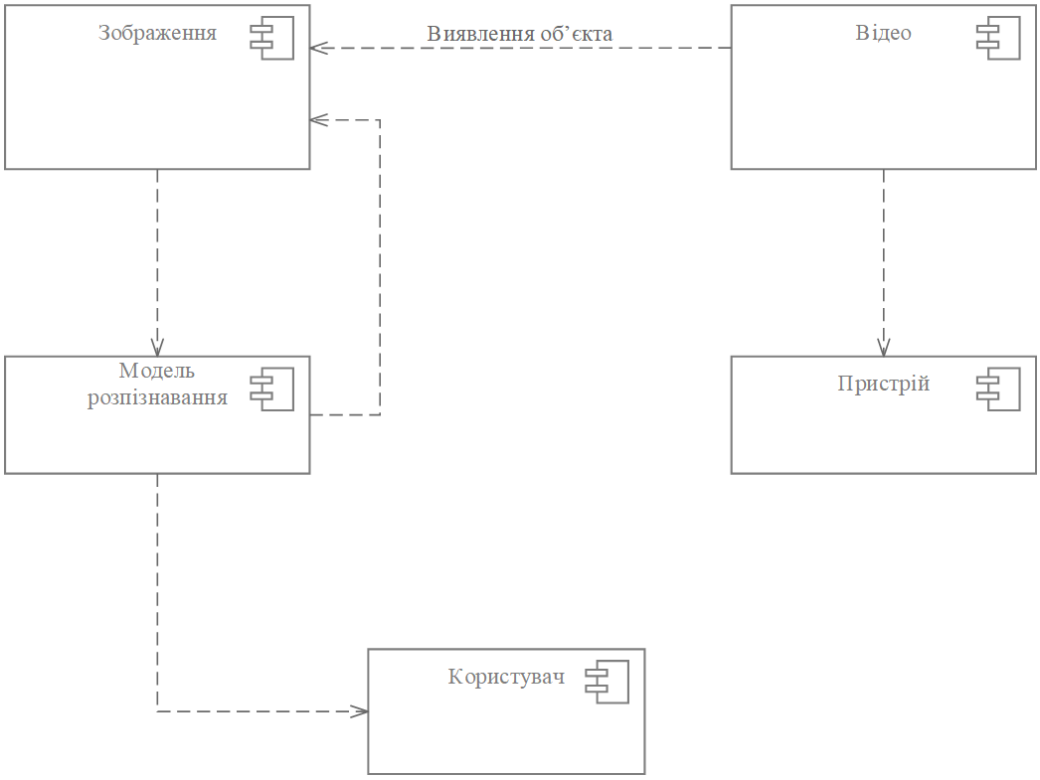

Рисунок 3.7 – Діаграма компонентів

На діаграмі компонентів для системи розпізнавання об’єктів видно, що дані спочатку передаються великому "прямокутнику", який може представляти собою або саму систему в цілому, або підсистеми та компоненти всієї системи. Це визначається контекстом і рівнем деталізації діаграми.

Далі дані направляються через різні з'єднання до внутрішніх компонентів, які відображені великим "прямокутником". Кожен з цих внутрішніх компонентів надає певні послуги та має наданий інтерфейс, який представляє доступні для використання служби.

Важливо відмітити, що інтерфейси зліва є наданими інтерфейсами, що означає, що вони представляють послуги, які надаються цими внутрішніми компонентами. Ці інтерфейси служать для взаємодії з іншими компонентами та системами.

3.4.5 Діаграма розгортання

Діаграма розгортання для системи розпізнавання об’єктів представлена на рисунку 3.8 представляє фізичні та програмні елементи системи. Ця діаграма дозволяє візуалізувати, як компоненти розміщені на різних вузлах або серверах, а також як вони взаємодіють між собою.

У діаграмі можна побачити різні вузли, які можуть включати апаратне забезпечення (наприклад, сервери) та програмне середовище виконання (наприклад, операційні системи чи середовища виконання програмного коду). Кожен вузол має свої власні компоненти, представлені прямокутниками з двома вкладками, що позначають програмні елементи.

Лінії, що з'єднують вузли та компоненти, показують залежності та зв'язки між ними. Залежності можуть вказувати, що один вузол або компонент використовує інший для своєї роботи, що підкреслює взаємозв'язок між різними частинами системи розпізнавання об’єктів.

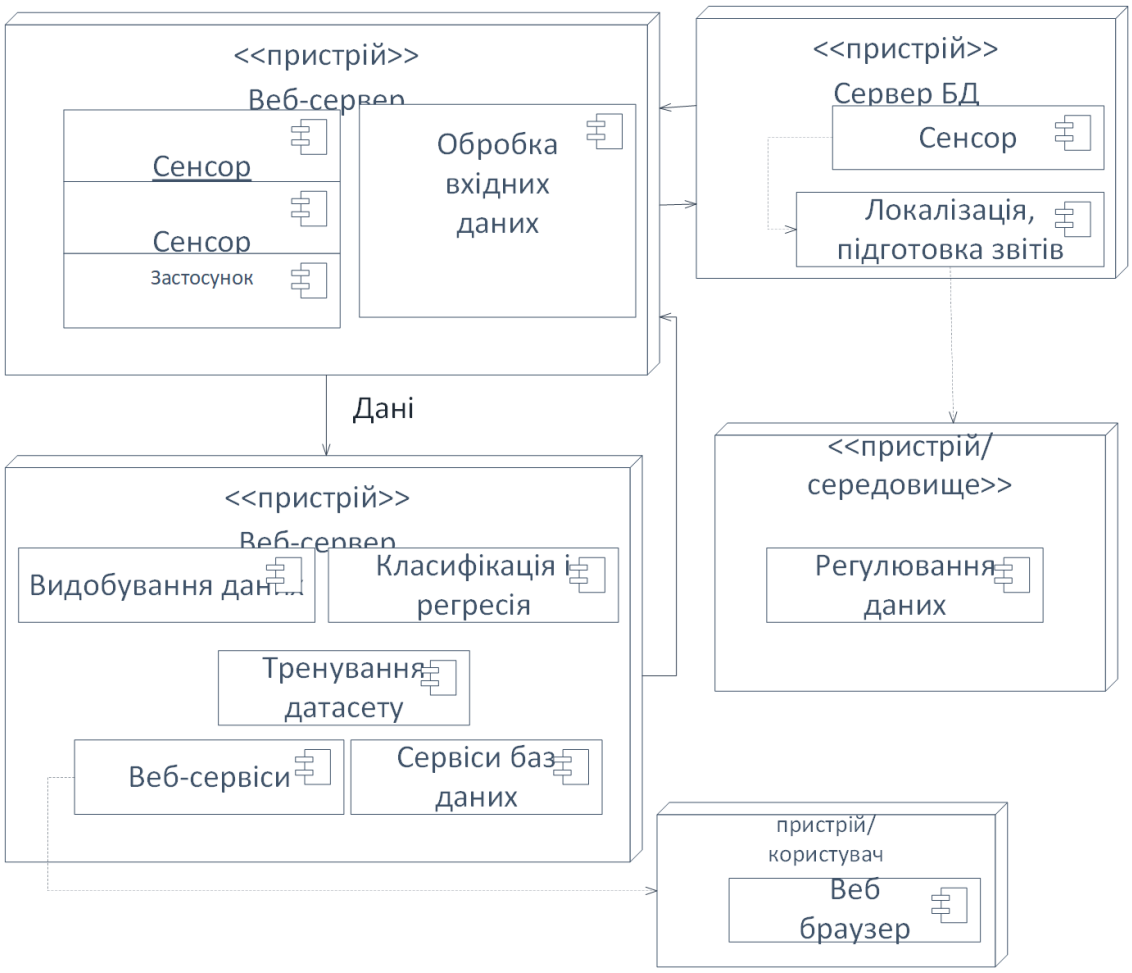

Рисунок 3.8 – Діаграма розгортання

3.5 Висновки до розділу 3

Ретельне проєктування та розробка програмних компонентів є критично важливими для створення ефективної системи розпізнавання об’єктів. Чітке визначення вимог до системи, розробка збалансованої системної архітектури, а також створення деталізованих діаграм потоків даних і програмних компонентів забезпечують надійність та продуктивність кінцевого продукту. Цей процес підкреслює необхідність глибокого аналізу та стратегічного планування на всіх етапах розробки.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ В СИСТЕМІ УПРАВЛІННЯ РОБОТИЗОВАННИМИ АВТОМОБІЛЯМИ

4.1 Розробка та аналіз тестових наборів для розпізнавання автомобілів за допомогою YOLO

YOLO, що розшифровується як "You Only Look Once" (Ти дивишся лише раз), стала революцією у галузі виявлення об'єктів у комп'ютерному зорі, пропонуючи швидкий та оптимізований підхід. Цей метод унікально спрощує процес виявлення, розглядаючи його як одноетапне завдання, яке поєднує локалізацію об'єкта та його класифікацію. Ця інноваційна методика дозволяє YOLO обробляти зображення та відео в режимі реального часу з високою точністю та здатне вирішувати виклики у виявленні об'єктів та сегментації зображень. Реальні часові можливості YOLO роблять її видатним вибором для застосувань, яким потрібна швидка та точна ідентифікація об'єктів.

Покроковий опис дій (представлення у вигляді коду Додаток Г):

Крок 1 Налаштування та ініціалізація

```
%%time

!git clone https://github.com/ultralytics/yolov5  # клонування репозиторію
!pip install -U pycocotools
!pip install -qr yolov5/requirements.txt  # встановлення залежностей
!cp yolov5/requirements.txt ./
```

Рис.4.1 – Клонування офіційного пакету YOLO v5

```
import warnings
warnings.filterwarnings('ignore')

import os
import shutil
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import yaml
from PIL import Image
from IPython.display import Video
```

Рис.4.2 – Встановлення бібліотек

Крок 2 Завантаження попередньо навченої моделі YOLO

Існує чітка тенденція: зі збільшенням розміру моделі спостерігається помітне покращення mAP, що вказує на підвищену точність. І навпаки, це збільшення відбувається за рахунок швидкості, оскільки більші моделі працюють повільніше. Усі моделі дотримуються стандартного вхідного розміру 640x640 пікселів, що оптимізує продуктивність у різноманітних програмах. Попередньо навчена модель навчена на наборі даних COCO, який включає класи «автомобіль» і «вантажівка» серед 80 різних категорій — саме те, що нам потрібно для нашого проекту. Перевірка моделі і результат на прикладі зображення, рис.4.3.

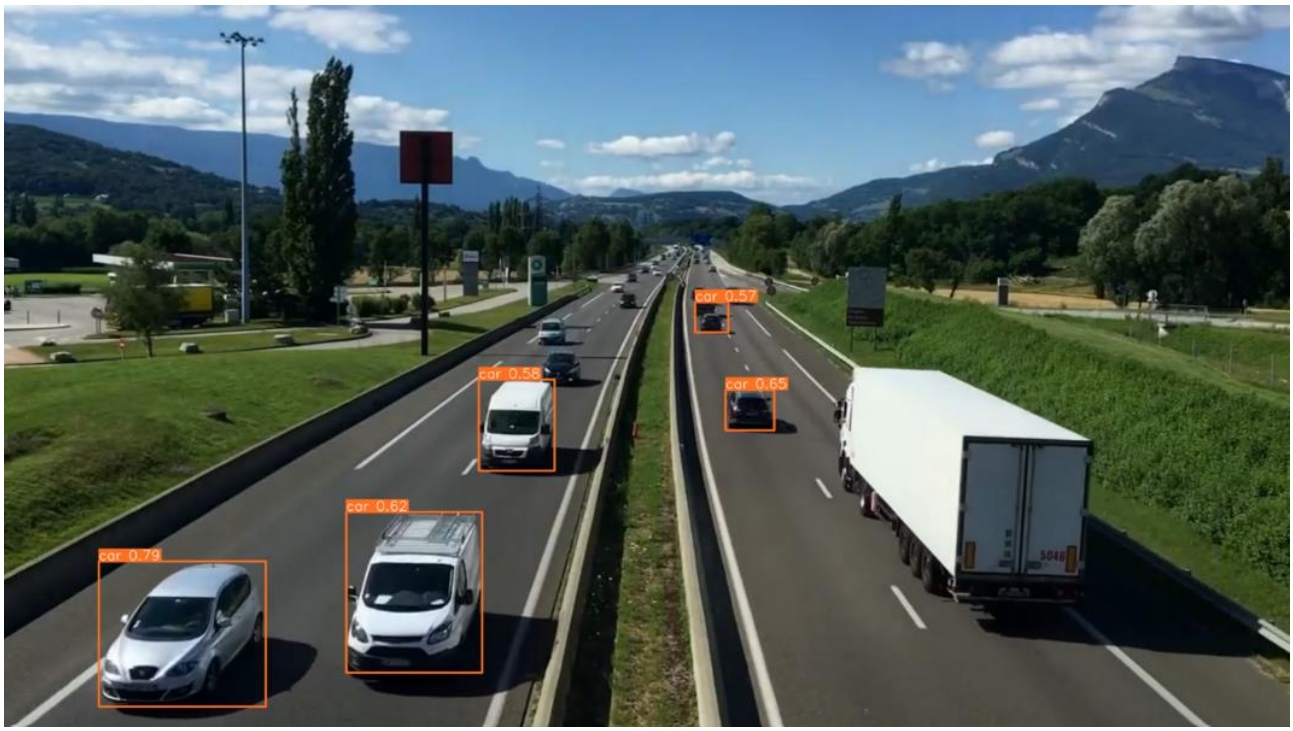

Рисунок 4.3 – Розпізнавання об’єктів на прикладі попередньо навченої моделі

Крок 3 Дослідження датасету

Щоб налаштувати попередньо підготовлену модель на спеціалізованому наборі даних, який зосереджується виключно на транспортних засобах, щоб вона могла навчитися точніше виявляти різні типи транспортних засобів, використовуємо набір даних. Зображення виявлення транспортних засобів у вигляді зверху. Набір даних зосереджується на класі «Транспортний засіб», охоплюючи широкий спектр транспортних засобів, таких як автомобілі, вантажівки та автобуси. Він складається з 626 зображень, отриманих з точки зору зверху, ретельно анотованих у форматі для ефективного виявлення транспортних засобів.

Набір даних проходить процес стандартизації, коли кожне зображення змінюється до однакової роздільної здатності 640x640 пікселів. Щоб підвищити здатність моделі до узагальнення, до

навчальних даних, які складаються з 536 зображень, було застосовано доповнення. Набір перевірки містить 90 зображень і залишається недоповненим, щоб зберегти цілісність оцінки ефективності.

```
Number of training images: 536
Number of validation images: 90
All training images have the same size: (640, 640)
All validation images have the same size: (640, 640)
```

Рисунок 4.4 – Вивід підрахованих зображень в наборах для навчання

Рисунок 4.5 – Приклад зображень з навчального набору даних

Крок 4 Оцінка продуктивності моделі

На цьому кроці відбувається процес тренування моделі глибокого навчання (рис. 4.6 та рис.4.7).

```
100 epochs completed in 0.245 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.3MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.235 🚀 Python-3.10.12 torch-2.0.0 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3005843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:02<00:
00,  1.40s/it]
                   all         90        937      0.925      0.933      0.976      0.747
Speed: 2.0ms preprocess, 2.3ms inference, 0.0ms loss, 0.9ms postprocess per image
Results saved to runs/detect/train
```

Рисунок 4.6 – Результат процесу тренування

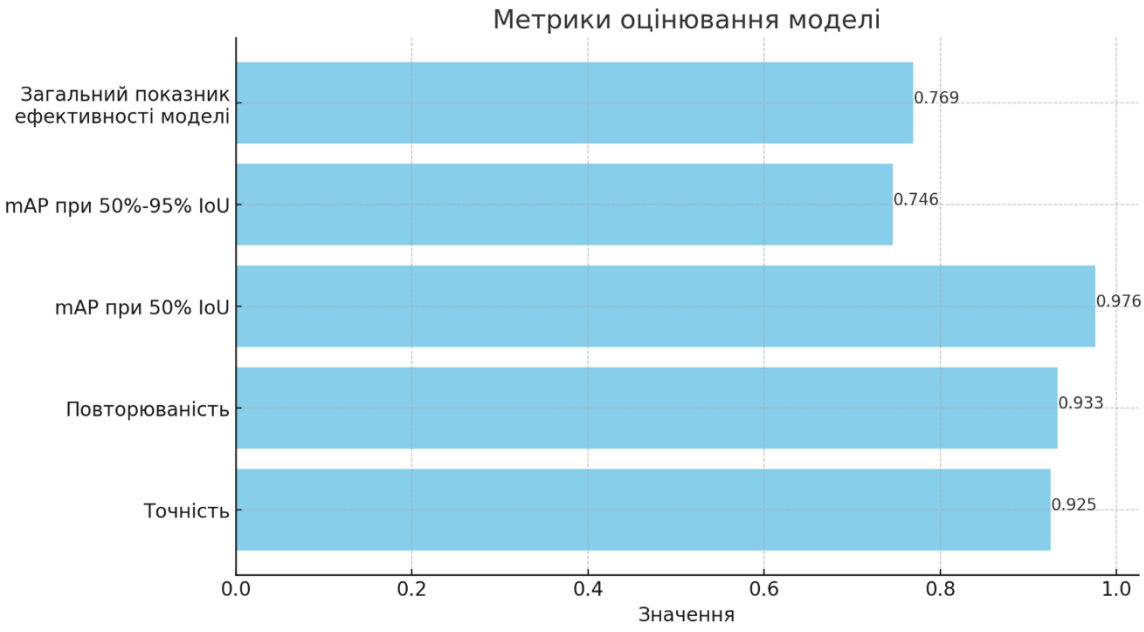

Рисунок 4.7 – Результати процесу тренування у вигляді графіку

Розуміння метрик:

1. Швидкість навчання для різних груп (lr/pg0, lr/pg1, lr/pg2): Ці значення представляють швидкість навчання для різних груп шарів нейронної мережі. Нижча швидкість навчання означає, що модель повільніше оновлює свої ваги під час тренування. Однакові швидкості навчання між групами вказують на однорідні коригування під час процесу навчання.

2. Середня точність у відсотках за 50% IoU (metrics/mAP50(B)): Ця метрика вимірює точність моделі у виявленні об'єктів з перекриттям

щонайменше 50% Intersection over Union (IoU) з істиною на землі. Рахунок 0.97 свідчить про високу точність моделі при цьому порозі IoU.

3. Середня точність у відсотках в діапазоні IoU від 50% до 95% (metrics/mAP50-95(B)): Це середнє значення mAP, розраховане за різними порогами IoU, від 50% до 95%. Рахунок 0.74 вказує на хорошу загальну точність на цих змінних порогах.

4. Точність (metrics/precision(B)): Точність вимірює співвідношення правильно передбачених позитивних спостережень до загальної кількості передбачених позитивів. Рахунок 0.92 означає, що модель високо точна у своїх передбаченнях.

5. Відгук (metrics/recall(B)): Відгук розраховує співвідношення правильно передбачених позитивних спостережень до всіх спостережень у фактичному класі. Відгук 0.93 показує, що модель дуже добре знаходить всі відповідні випадки у датасеті.

6. Обчислювальна складність моделі (model/GFLOPs): Вказує на обчислювальні вимоги моделі, де значення GFLOPs свідчить про помірну складність.

7. Параметри моделі: Загальна кількість навчальних параметрів у моделі. Майже 3 мільйони параметрів вказують на модель помірного розміру та складності.

8. Швидкість виведення (model/speed_PyTorch(ms)): Час, необхідний моделі для здійснення одного передбачення (виведення). 4 мс є досить швидким, що є хорошим для застосувань у реальному часі.

9. Втрати під час тренування (train/box_loss, train/cls_loss, train/dfl_loss): Це різні типи втрат під час тренування. 'box_loss' відноситься до помилки у передбаченнях обмежувальних рамок, 'cls_loss' - до помилки класифікації, а 'dfl_loss' - до втрати, зосередженої на розподілі. Нижчі значення вказують на кращу продуктивність.

10. Втрати під час валідації (val/box_loss, val/cls_loss, val/dfl_loss): Подібно до втрат під час тренування, це втрати, розраховані на

валідаційному наборі даних. Вони дають уявлення про те, як добре модель узагальнює нові, невидані дані. Майже однакові значення втрат для тренування та валідації вказують на те, що модель не перенавчена.

Крок 5 Виведення на зображеннях з валідаційного набору

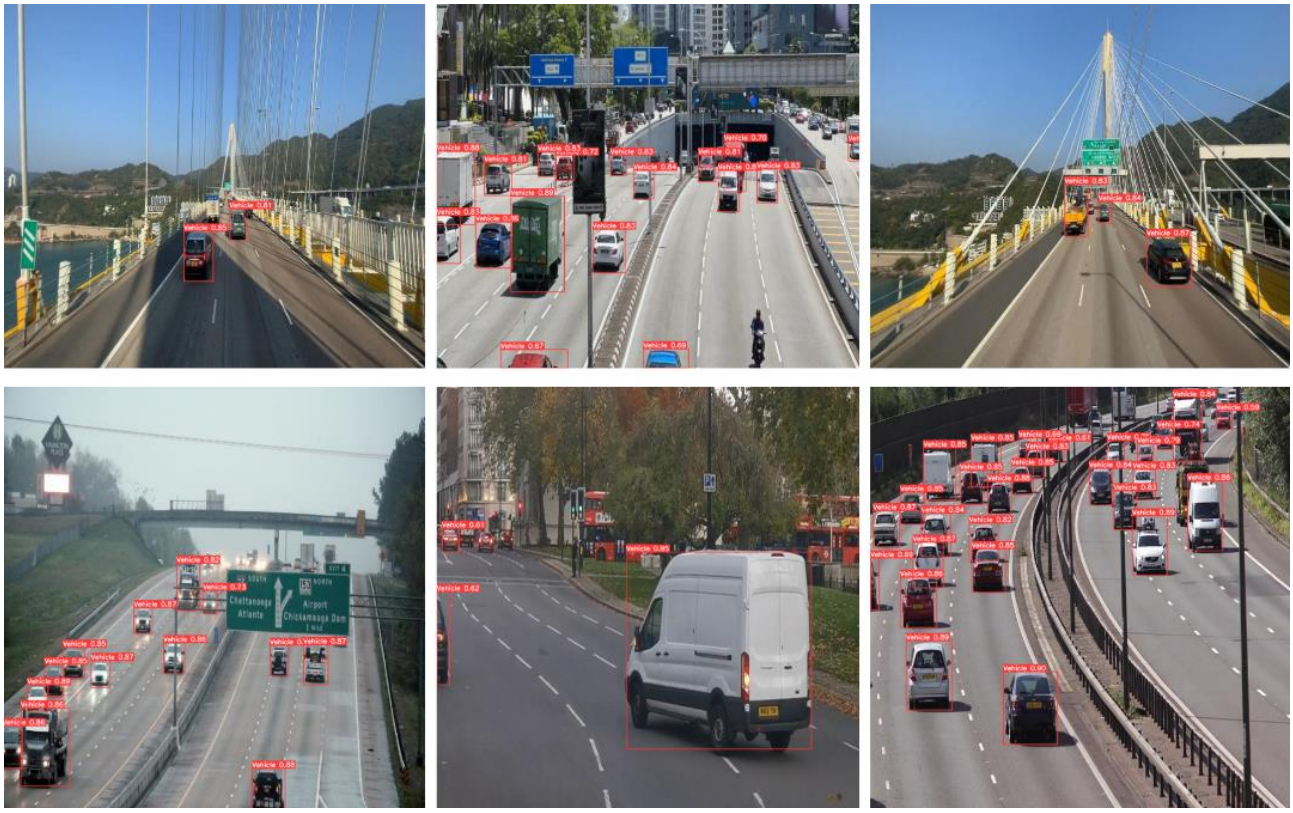

Рисунок 4.8 – Виведення зображень з валідаційного набору

Крок 6 Виведення на тестовому зображенні

Використовуємо найкращу версію нашої точно налаштованої моделі, щоб оцінити її можливості узагальнення. Перевірка здійснюється на тому самому зображенні, яке раніше було проаналізовано за допомогою попередньо підготовленої моделі на наборі даних COCO:

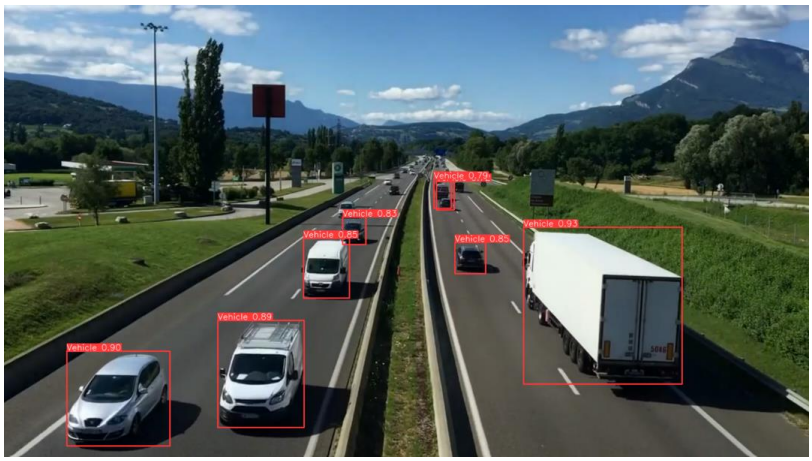

Рисунок 4.9 – Виявлення об’єктів на прикладі попередньо підготовленої моделі

Крок 7 Виведення на тестовому відео

Оцінка можливості узагальнення моделі на відео. Цей крок має вирішальне значення для демонстрації здатності моделі адаптуватися та точно працювати в реальних програмах, ще більше зміцнюючи її ефективність за межами середовища контрольованого набору даних.

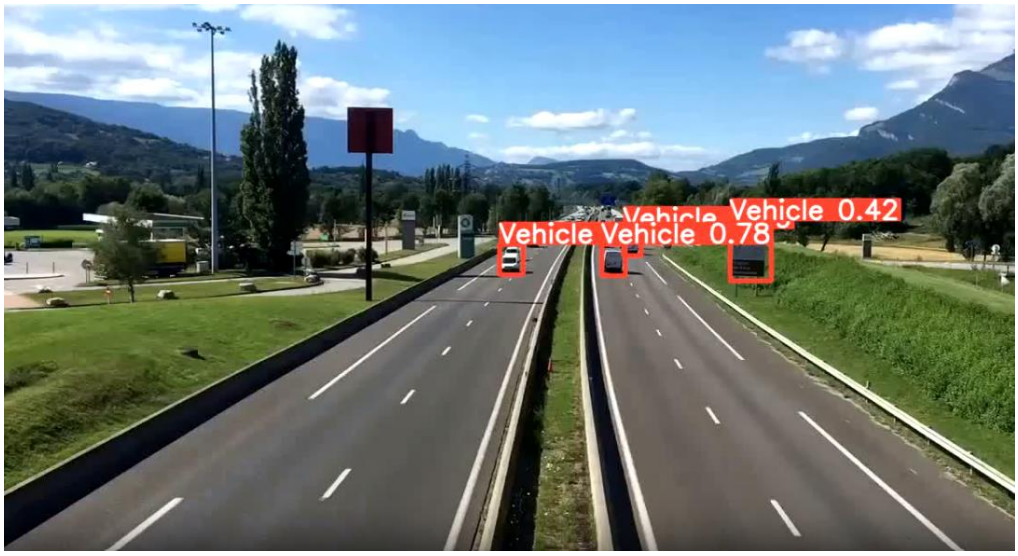

Рисунок 4.10 – Скріншот отриманого відео

Крок 8 Оцінка інтенсивності руху в реальному часі

Мета цього етапу полягає в кількісній оцінці дорожнього руху шляхом кадрового підрахунку транспортних засобів у межах визначених зон на смугах доріг. Аналіз не тільки виявить кількість транспортних засобів, але й оцінить інтенсивність руху, позначаючи його як «Завантажене» або «Помірне» на основі заздалегідь визначеного порогу. Статистика підрахунку та трафіку є ключовими для міського планування та управління трафіком.

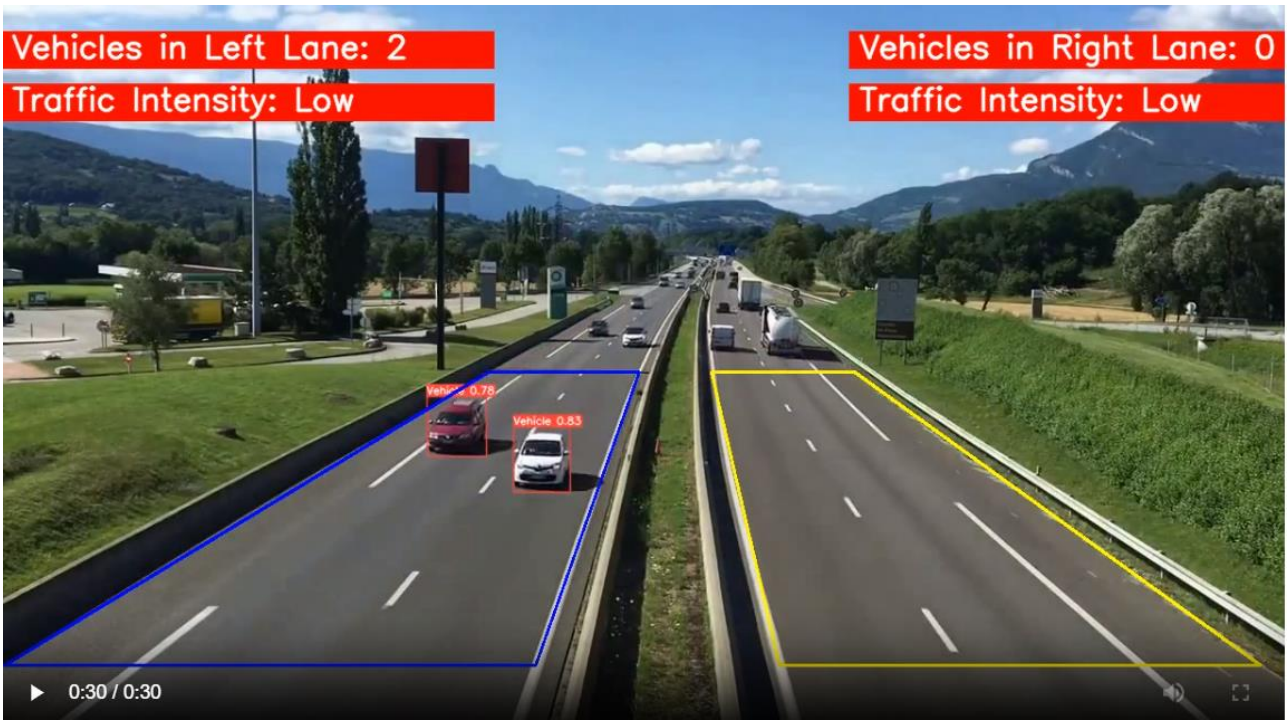

Рисунок 4.11 – Оцінка інтенсивності руху в реальному часі

4.2 Висновки до розділу 4

Розвиток автономних транспортних засобів вимагає високоточних систем розпізнавання об'єктів для адекватного сприйняття дорожнього

середовища. YOLO, як один з передових методів в області комп'ютерного зору, пропонує швидке та ефективне рішення для цієї проблеми. Його здатність швидко обробляти зображення в реальному часі робить його ідеальним для використання в автономних транспортних засобах.

Автономні транспортні засоби повинні адаптуватися до різноманітних дорожніх умов. YOLO може бути оптимізовано для точного розпізнавання об'єктів в різних погодних умовах і освітленні, що є критично важливим для безпеки.

Автономні транспортні засоби, інтегровані з цією технологією, значно краще "бачать" та "розуміють" оточення, знижуючи ризик аварій. Це особливо важливо для виявлення небезпечних ситуацій, таких як раптові перешкоди на дорозі або пішоходи, що несподівано з'являються на дорозі. Тестування та валідація цієї системи є ключовим аспектом її впровадження, щоб забезпечити надійність у різних дорожніх сценаріях.

Використання цього методу в системах автономного водіння відкриває нові можливості для збільшення безпеки на дорогах. Він не лише забезпечує точність в розпізнаванні об'єктів, але й необхідну швидкість обробки для адекватної реакції на динамічні зміни в дорожніх умовах. Майбутнє принесе подальший розвиток та вдосконалення цих технологій, роблячи автономне водіння ще безпечнішим та доступнішим.

Використання технології YOLO відкриває нові горизонти для забезпечення безпеки на дорогах. Це не лише забезпечує високу точність та швидкість обробки зображень, але й сприяє адекватній реакції на динамічні зміни в дорожніх умовах. Подальший розвиток та вдосконалення цих технологій зробить автономне водіння ще безпечнішим та доступнішим, відкриваючи нові можливості для транспортної індустрії майбутнього.

РОЗІЛ 5. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ

5.1 Обґрунтування доцільності розробки програмного забезпечення

У цьому розділі представлено обґрунтування проєкту, що зосереджується на розробці спеціалізованого програмного забезпечення (ПЗ) для розпізнавання об'єктів. Завдяки швидкому розвитку технологій у сфері комп'ютерного зору, сьогодні з'явилася унікальна можливість створення високоефективних систем розпізнавання. Це ПЗ використовує передові алгоритми для аналізу відеопотоків, що дозволяє точно ідентифікувати та відстежувати різні об'єкти.

Основна функція такого ПЗ полягає у генерації докладних характеристик об'єктів, заснованих на їх зображеннях, та зберіганні цих даних у базі. Також важливою є можливість ідентифікації об'єктів на основі зібраної інформації. Це дає змогу системі ефективно розпізнавати об'єкти в різних умовах, що є ключовим для широкого спектра застосувань, від безпеки до автоматизації процесів.

Важливою перевагою цього ПЗ є його доступність та висока точність. Завдяки оптимізації алгоритмів, програма забезпечує високу якість розпізнавання, не вимагаючи при цьому використання дорогого обладнання. Такий підхід робить ПЗ привабливим для підприємств групи Метінвест, дозволяючи йому конкурувати з дорогими аналогами, пропонуючи при цьому аналогічну або навіть вищу якість.

У підсумку, розробка такого ПЗ відіграє важливу роль у сфері комп'ютерного зору, надаючи можливість для створення більш

інтелектуальних та автономних систем, здатних точно розпізнавати та відстежувати об'єкти в динамічному середовищі.

5.2 Аналіз конкурентоспроможності розробленого ПЗ в порівнянні з існуючим аналогом

Для об'єктивної оцінки конкурентоспроможності розробленого програмного забезпечення (ПЗ) у порівнянні з аналогом, яким є програма Xiaomi Mi Home від syssoft, слід розглянути декілька ключових аспектів. Важливо зазначити, що вибір Xiaomi Mi Home як аналога зумовлений такими факторами, як відсутність в додаткових адаптаціях або модифікаціях для забезпечення сумісності, можливість масштабування ПЗ відповідно до зростаючих потреб бізнесу, а також наявність безкоштовної версії для дослідження та порівняння.

Оцінка конкурентоспроможності включає аналіз та порівняння обох програм за кількома критеріями:

1. Функціональне призначення. Оцінюється, наскільки добре обидва ПЗ виконують свої основні функції та задачі.

2. Технічні параметри. Тут аналізуються технічні характеристики ПЗ, такі як швидкість обробки даних, точність розпізнавання об'єктів, стабільність роботи тощо.

3. Експлуатаційні параметри. Вивчається простота встановлення, налаштування, інтеграція з іншими системами, а також зручність інтерфейсу.

4. Сфера використання. Розглядаються можливі сценарії використання ПЗ та їх відповідність потребам користувачів.

Експлуатаційно-технічний рівень ($J_{ЕТР}$) програмного забезпечення, що розроблено, визначається як узагальнена характеристика його

експлуатаційних властивостей і можливостей, ступеня новизни. Цей показник є основою для визначення якості ПЗ. В таблиці 5.1 представлені результати розрахунку $J_{ЕТР}$ за бально-індексним методом по п'ятибальній шкалі оцінювання.

Такий підхід дозволить всебічно оцінити розроблене ПЗ та визначити його переваги і недоліки у порівнянні з вибраним аналогом, що є критично важливим для розуміння його позиціонування на ринку.

Таблиця 5.1 – Розрахунок оцінки ефективності ПЗ

Показники якості програмного продукту	Коефіцієнт вагомості, B_j	ПЗ		Аналог	
		X_j	$X_j \cdot B_j$	X_j	$X_j \cdot B_j$
1. Зручність роботи	0,21	5	1,05	4	0,84
2.Новизна	0,11	3	0,33	3	0,33
3.Відповідність профілю діяльності замовника	0,21	5	1,05	2	0,41
4.Ресурсна ефективність	0,06	4	0,24	4	0,24
5. Надійність (захист даних)	0,1	2	0,2	3	0,3
6.Швидкість доступу до даних	0,14	3	0,42	2	0,28
7.Гнучкість налаштування	0,09	2	0,18	3	0,27
8.Здатність до навчання клієнтом	0,08	4	0,32	2	0,32
Комплексна оцінка ефективності ПЗ $J_{ЕТР}$			**4,74**		**2,99**

Для порівняння програмного забезпечення (ПЗ) та його аналога, було використано коєфіцієнт технічного рівня, формула 5.1, який розраховується як відношення узагальнених показників експлуатаційно-технічного рівня ($J_{ЕТР}$) обох програм.

$$A_{(k)} = J_{ЕТР2} / J_{ЕТР1} = 2.99 / 4.74 = 1.6 \quad (5.1)$$

Оскільки отриманий коефіцієнт 1,6 більший за 1, це свідчить про те, що розробка ПЗ, в порівнянні з аналогом, є технічно обґрунтованою.

Іншими словами, ПЗ має вищий технічний рівень, що вказує на його кращі експлуатаційні характеристики, новизну технологій та ефективність алгоритмів.

5.3 Планування етапів розробки та економічний аналіз проекту

Планування комплексу робіт з розробки програмного забезпечення та оцінка трудомісткості цих робіт є ключовими етапами управління проектами в галузі ІТ.

Для розробки програмного забезпечення залучаються дві основні ролі:

1. Керівник проекту - відповідає за формування завдань проекту, розробляє план виконання завдань та управління проектом та забезпечує консультації та координацію усіх аспектів проекту.

2. Виконавець (студент) - займається проектуванням архітектури ПЗ, реалізує алгоритми, інтерфейс користувача та інші завдання згідно з планом проекту.

Вибір комплексу робіт проекту відбувається відповідно до стандарту ISO/IEC 12207: 2008, що визначає процеси життєвого циклу систем і програмного забезпечення. Цей стандарт забезпечує уніфіковані підходи до розробки, тестування, підтримки та виведення ПЗ з експлуатації, гарантуючи високу якість та системність управління проектами в галузі програмного забезпечення.

При розробці програмного забезпечення, використання календарного графіку робіт є важливою частиною планування та управління проектом. Згідно зі стандартом, було обрано комплекс робіт і розроблений графік (таб.5.2), який відображає послідовність та взаємозв'язок між різними завданнями у проекті. Графік визначає

порядок виконання різних завдань, що допомагає впорядкувати процес розробки та забезпечує логічний перехід між етапами проекту. Показує залежності між різними завданнями та етапами, дозволяючи краще розуміти вплив затримок чи змін у одному завданні на загальний прогрес проекту. На основі таблиці було розроблено діаграму Ганта (рис.5.1) яка являє собою візуальний інструмент для представлення графіка робіт, включає часові інтервали для кожного завдання, демонструючи їх тривалість та перекриття.

Використання діаграми Ганта дозволяє ефективно управляти часом та ресурсами, передбачати ризики затримок і забезпечувати точне дотримання встановлених термінів. Цей інструмент є незамінним для керівників проектів, оскільки він забезпечує чітке уявлення про стан проекту та допомагає координувати роботу команди.

5.4 Розрахунок проєктних витрат на розробку ПЗ

Інвестиції у проекти, що охоплюють створення та імплементацію програмних продуктів, визначаються за формулою:

$$K = K_P + K_R \quad (5.2)$$

де - це кошти, вкладені у процес проектування (попередні витрати на виробництво), виражені у гривнях;

- це фінансування, спрямоване на реалізацію проекту, також у гривнях.

Ця формула демонструє, що загальні капіталовкладення в проект програмного забезпечення є сумою коштів, витрачених на етапи проектування та реалізації. Важливо усвідомлювати, що витрати на

проектування включають всі дії, пов'язані з підготовкою та плануванням проекту, в той час як витрати на реалізацію відображають практичне втілення задуманого, включаючи розробку програмного коду, тестування, інтеграцію та інші пов'язані дії.

Для визначення витрат на проект, що включає розробку програмного забезпечення, необхідно розрахувати основну заробітну плату виконавців (таб.5.3). Це робиться на основі кількості часу, яку кожен виконавець приділяє проекту, згідно з календарним графіком робіт. Розрахунок основної заробітної плати виконавців СН здійснюється, виходячи з припущення, що в місяці є 21 робочий день.

Формула розрахунку виглядає так: основна заробітна плата визначається як денна ставка, помножена на кількість робочих днів у місяці, а також на відсоток завантаження виконавця на проекті. Це дозволяє отримати точну оцінку витрат на трудові ресурси проекту, що є критично важливим для точного бюджетування і планування витрат проекту.

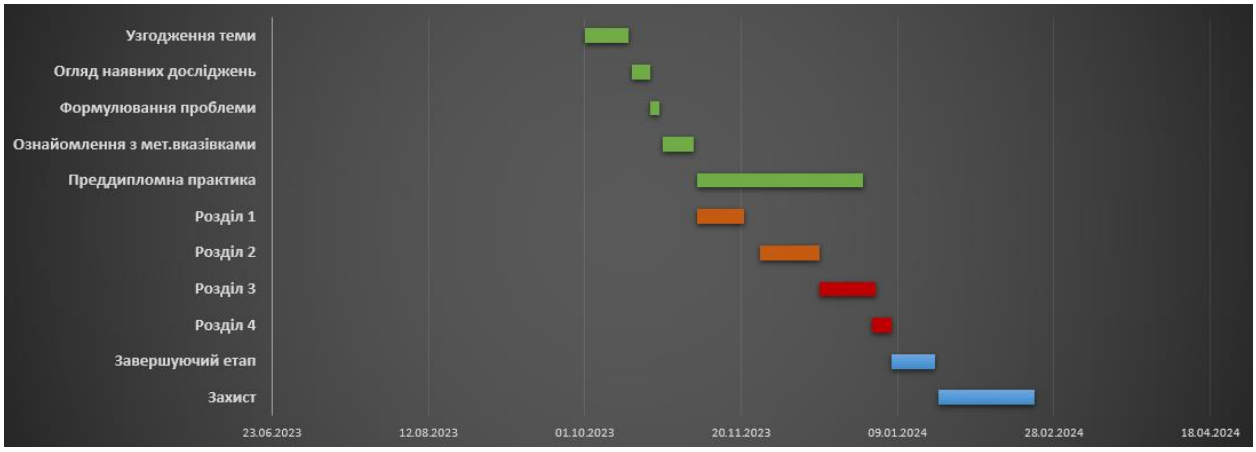

Рисунок 5.1 – Діаграма Ганта графіку часового планування

Таблиця 5.2 – Графік часового планування робочих процесів проекту

Task No.	Task	Task Owner	Start Date	Finish Date
1	**Підготовчий етап**			
1.1	Узгодження теми та обгрунтування актуальності	Керівник, студент	02.10.2023	02.10.2023
1.1.1	Огляд наявних досліджень	Керівник, студент	03.10.2023	03.10.2023
1.1.2	Формулювання проблеми	Керівник, студент	04.10.2023	04.10.2023
2	**Дослідницький етап**			
2,1	Преддипллomна практика	Керівник від підпр.	06.11.2023	06.11.2023
2.1.1	Огляд проблеми використання методів інтелектуального аналізу в задачах розпізнавання об'єктів	Студент	06.11.2023	06.11.2023
2.1.1	Вивчення та аналіз літератури для розділу 1 магістерської роботи	Студент	07.11.2023	07.11.2023
2.1.1	Огляд методів інтелектуального аналізу даних	Студент	07.11.2023	07.11.2023
2.1.2	Використання методів інтелектуального аналізу даних в задачах управління автомобілем	Студент	07.11.2023	07.11.2023
2.1.3	Огляд методів розпізнавання об'єктів	Студент	07.11.2023	07.11.2023
2.1.3	Постановка задачі дослідження	Студент	08.11.2023	08.11.2023
2.1.4	Огляд методів інтелектуального аналізу в задачах управління роботизованими автомобілями	Студент	11.11.2023	12.11.2023
2.2	Розвиток роботизованих автомобілів	Студент	14.11.2023	14.11.2023
2.2.1	Набори даних для побудови систем управління роботизованними автомобілями	Студент	15.11.2023	15.11.2023
2.2.2	Методи глибокого навчання в задачах управління роботизованними автомобілями	Студент	16.11.2023	17.11.2023
3	**Проєктування та розробка**			
3.1	Проектування та розробка	Керівник	22.11.2023	25.11.2023
3.1.1	Загальні відомості про систему	Студент	26.11.2023	28.11.2023
3.1.2	Програмна архітектура	Студент	28.11.2023	30.11.2023
3.1.3	Модель даних	Студент	05.12.2023	10.12.2023
4	**Етап контролю**			
4.1	Дослідження запропонованої системи розпізнавання об'єктів	Керівник, студент	25.12.2023	30.12.2023
4.2	Дослідження безпеки	Керівник, студент	07.01.2023	07.01.2023
5	**Завершальний етап**			
5.1	Проведення економічних розрахунків	Студент	11.01.2024	11.01.2024
5.2	Оформлення магістерської роботи відповідно до вимог	Студент	13.01.2024	13.01.2024
5.2.1	Перевірка на плагіат	Спеціаліст	15.01.2024	16.01.2024
5.2.2	Збір необхідних підписів	Студент	16.01.2024	16.01.2024
5.2.3	Підготовка презентації	Студент	16.01.2024	16.01.2024
5.2.4	Підготовка докладу	Студент	16.01.2024	17.01.2024
6	Захист магістерської роботи перед комісією	Студент	22.01.2024	22.01.2024

Таблиця 5.3 – Основна заробітна плата

Посада	Посадовий оклад, грн	Середня денна ставка C_{Ti}, грн.	Витрати часу на розробку t_i , людино-дні	Витрати на основну заробітну плату, C_{Hi}, грн.
Керівник	8424	8423/21=401,1	9	3609,9
Програміст	1230	1230/21= 58,6	115	6739
Усього, C_H				10348,9

Для розрахунку загальних витрат на проект, який включає розробку програмного забезпечення, важливо врахувати витрати на використання машинного часу та витрати на матеріали.

1. Витрати на Використання Машинного Часу ($М_Т$):

$$М_Т = 920 \cdot 5 \cdot 1 = 4600 \text{ грн.} \quad (5.3)$$

Це означає, що вартість використання обладнання або машинного часу для проекту становить 4600 гривень.

2. Витрати на Матеріали (СМ):

Для розрахунку цих витрат необхідно використовувати інформацію з таблиці 5.4, де вказані види матеріалів, які були придбані в процесі виконання проекту, та їх вартість.

Сума цих витрат дасть загальну вартість матеріалів, використаних у проекті.

Врахування цих двох компонентів дозволить отримати повну картину витрат, пов'язаних з розробкою програмного забезпечення. Це критично важливо для точного бюджетування проекту та ефективного управління фінансовими ресурсами.

Таблиця 5.4 – Витрати на матеріали

Матеріали	Од. вимірювання	Необхідна кількість	Ціна за од, грн.	Сума, грн.
Зошит загальний	шт.	1	30	30
Флешка	шт.	2	200	200
Тонер для принтера	шт.	1	120	120
Папір офісний	Пачка	1	100	100
Усього				450

Капітальні вкладення проєкту дорівнюють:

$$K_P = ((1+0,5)\times(1+0,22)+0,6)\ 10348,9 + 450 + 4600 = 15\ 398,9 \text{ грн.} \quad (5.4)$$

де:

(1+0.5) - коефіцієнт, який враховує додаткові витрати (наприклад, адміністративні витрати).

(1+0.22) - коефіцієнт, який враховує накладні витрати.

0.6 - коефіцієнт додаткових витрат, пов'язаних з проектом.

10348.9 - основна сума витрат на проектування.

450 - витрати на матеріали.

4600 - витрати на використання машинного часу.

Кошторис на розробку ПЗ наведено у табл. 5.5.

Таблиця 5.5 – Кошторис на розробку ПЗ

Статті витрат	Сума, грн.
Основна заробітна плата	10348,9
Додаткова зарплата	5121,4
Відрахування	3478.36
Витрати на матеріали	270
Витрати на машинний час	4600
Накладні витрати організації	5395.2
ВСЬОГО	28512.76

Загальна сума $К_Р$ дорівнює 15218.9 грн., що представляє собою повну вартість капітальних вкладень на проектування.

5.5 Розрахунок витрат на впровадження програмного забезпечення

Розрахунок витрат включає детальний аналіз капітальних вкладень, необхідних для впровадження програмного забезпечення в проекті:

$$K_1 = K_P + K_R = 15218.9 + 39250 = 44468,9 \text{ грн.} \quad (5.5)$$

де:

K_P – капітальні вкладення, пов'язані з проектуванням програмного забезпечення. Це може включати витрати на аналіз вимог, проектування архітектури, розробку прототипів, тестування тощо.

K_R – капітальні вкладення, пов'язані з реалізацією проекту. Тут включаються витрати на впровадження ПЗ, навчання персоналу, придбання обладнання, ліцензійні витрати та інші витрати, пов'язані з запуском та підтримкою ПЗ.

5.6 Калькуляція фінансових витрат на придбання та імплементацію альтернативного програмного рішення

Для розрахунку сумарних витрат, пов’язаних з впровадженням програми Xiaomi Mi Home як аналога до розробленого програмного забезпечення, ми враховуємо наступні складові:

Витрати на Придбання Програмного Продукту (K_{AE}) 36080 грн. за придбання програми Xiaomi Mi Home.

Витрати на Установку і Супровід Програмного Продукту: враховуючи, що компанія-розробник забезпечує безкоштовну установку та супровід програми Xiaomi Mi Home протягом 2 років, ці витрати не включаються в загальну суму.

Витрати на Основне та Допоміжне Обладнання (K_{PP}) 19000 грн. за ПК, який необхідний для впровадження програми Xiaomi Mi Home та 4000 грн. за додаткове програмне забезпечення.

Таким чином, сумарні витрати на впровадження програми Xiaomi Mi Home складаються з наступних пунктів:

Витрати на придбання програмного продукту: 36080 грн.

Витрати на обладнання: 19000 грн.

Витрати на додаткове програмне забезпечення: 4000 грн.

За формулою для розрахунку загальних капітальних вкладень на реалізацію проекту (K_R) ми маємо:

$$K_R = K_{AE} + K_{PP} = 36080 + 19000 + 4000 = 59080 \text{ грн.} \quad (5.6)$$

Таким чином, загальна сума капітальних вкладень на впровадження програми Xiaomi Mi Home складає 59080 грн.

Для розрахунку сумарних витрат на впровадження програми LRM, яка є аналогом розробленого вами програмного забезпечення, використовується формула:

$$K_2 = K_{AE} + K_R \quad (5.7)$$

де:

K_2 – загальна сума витрат на впровадження програми LRM.

K_{AE} – витрати на придбання програмного продукту, в цьому випадку 36080 грн.

K_R – витрати на реалізацію проекту, включаючи обладнання, установку та інші пов'язані витрати, у цьому випадку 59080 грн.

Застосувавши ці дані, отримуємо:

$K_2 = 36080 + 59080 = 95160$ грн.

Отже, сумарні витрати на впровадження програми LRM складають 95160 грн.

5.7 Розрахунок операційних витрат на експлуатацію програмного забезпечення та його аналога

Щоб провести розрахунки поточних експлуатаційних витрат програмного продукту (ПЗ та його аналога) на перший рік експлуатації, важливо врахувати витрати на заробітну плату розробників ПЗ. Витрати CS на заробітню плату розробників наведено в таблицях 5.6 та 5.7, розрахунок виглядає так:

Формула для розрахунку витрат на заробітну плату (CS1) для ПЗ:

$$C_{S1} = (1142.86 \cdot 12 + 500 \cdot 96)\,(1 + 0.4)\,(1 + 0.22) = 77104 \text{ грн за рік} \quad (5.8)$$

де:

- 1142.86 грн - місячна ставка заробітної плати одного розробника.

- 12 - кількість місяців у році.

- 500 грн - денна ставка заробітної плати.

- 96 - кількість робочих днів.

- 0.4 - додаткові витрати (наприклад, податки, соціальне страхування тощо).

- 0.22 - інші накладні витрати.

Таблиця 5.6 – Дані по заробітній платі користувачів ПЗ

Посада	Посадовий оклад, грн.	Середня денна ставка, грн./день	Витрати часу на роботу з ПЗ, людино-дні	Фонд з/п, грн.
Програміст	16000	761.9	1 день *12 міс = 12	9142,8
Співробітник відділу	9000	428.57	7 днів *12 міс =84	36000
Разом				61714,29

Програмний продукт, який є аналогом розробленого ПЗ, вимагає значно більше часу на технічну підтримку. Конкретно, він потребує у два рази більше часу порівняно з вашим розробленим ПЗ. Цей факт може мати важливі наслідки для оцінки експлуатаційних витрат, ефективності та загальної практичності використання обох програмних продуктів.

Таке співвідношення часу, витраченого на технічну підтримку, може бути використане для додаткового аналізу витрат, економічної ефективності та окупності обох програм. Наприклад, якщо ваше ПЗ вимагає менше часу на підтримку, це може призвести до зменшення витрат на обслуговування і, як наслідок, до зниження загальних експлуатаційних витрат. Це також може вплинути на задоволеність користувачів і загальну ефективність використання ПЗ у довгостроковій перспективі.

$$C_{S2} = (1666{,}67 \cdot 24 + 595{,}24 \cdot 96)(1 + 0{,}4)(1 + 0{,}22) = 123952$$

грн.(за рік).

Таблиця 5.7 – Розрахунок заробітної плати користувачів для аналога

Посада	Посадовий оклад, грн.	Середня денна ставка, грн./день	Витрати часу на роботу з програмою, людино-дні	Фонд з/п, грн.
системний адміністратор	25000	1190,47	24	28571,28
Співробітник відділу	11000	523,3	84	44000
Разом				72571,28

Сума амортизаційних відрахувань (C_D) для обладнання складе:

$$C_{D1}=(19000\cdot0,2\cdot1\cdot768)/2008=1453,38 \text{ грн.} \quad (5.9)$$

Сума амортизаційних відрахувань для аналога складе:

$$C_{D2} = \frac{19000\cdot0,2\cdot1\cdot848}{2008} = 1604,78 \text{ грн.} \quad (5.10)$$

Для розрахунку витрат на електроенергію, поточний ремонт обладнання та матеріали протягом року для розробленого ПЗ та його аналогу, можна використовувати наступні формули:

1. Витрати на Електроенергію СЕ:

- Для ПЗ: C_{E1} = 0.4 · 1 · 768 ·1.68 = 516.1 грн.
- Для аналога: C_{E2} = 0.4 · 1· 848 · 1.68 = 569.85 грн.

2. Витрати на поточний ремонт обладнання C_{RE}:

- Для ПЗ: C_{RE1} = 0.05 ·19000 · 768 / 2008 = 363.34 грн.
- Для аналогу: C_{RE2} = 0.05 ·19000 · 84 / 2008= 401.19 грн.

3. Витрати на Матеріали C_M:

- Оскільки використовується однакове обладнання (ПК однієї вартості), витрати на матеріали будуть однаковими для обох варіантів:

$C_M = C_{M2}$ = 19000 · 0.01 = 190грн.

4. Накладні Витрати:

- Вважається, що норматив накладних витрат становить 20% від прямих витрат, які включають перші п'ять статей витрат з табл. 5.8.

Ці розрахунки дозволяють оцінити повну вартість експлуатації розробленого ПЗ та його аналога, включаючи витрати на енергію, ремонт, матеріали та накладні витрати. Це дає змогу порівняти ці витрати між двома варіантами та визначити найбільш економічно ефективний варіант.

Таблиця 5.8 – Річні експлуатаційні витрати

Статі витрат	Витрати на ПЗ, грн.	Витрати на аналог, грн.
Основна і додаткова зарплата з відрахуванням, C_S	77104	123952
Амортизовані відрахування, C_D	1453,38	1604,78
Витрати на електроенергію, C_E	561,1	569,85
Витрати на поточний ремонт, C_{RE}	363,34	401,19
Витрати на матеріали C_M	190	190
Накладні витрати	15934,36	25343,56
Усього витрат, C_C	95606,18	152061,38

Накладні витрати для ПЗ складуть:

$$C_{OH1} = (105408 + 1290{,}84 + 580{,}61 + 322{,}71 + 150) \cdot 0{,}2 = 15934{,}36 \text{ грн.}$$

Накладні витрати для аналогу:

$$C_{OH2} = (165920 + 1434{,}26 + 645{,}12 + 358{,}57 + 150) \cdot 0{,}2 = 25343{,}56 \text{ грн.}$$

Отже, сумарні поточні експлуатаційні витрати для розробленого програмного забезпечення та його аналога становлять відповідно:

- Для розробленого ПЗ: C_C = 95606,18 грн.
- Для аналога ПЗ: C_C = 152061,38 грн.

Ці дані вказують на значну різницю у експлуатаційних витратах між розробленим ПЗ та його аналогом, демонструючи можливу перевагу у вартості утримання та експлуатації розробленого ПЗ порівняно з аналогом. Це важливий аспект при оцінці загальної економічної ефективності програмного продукту, особливо при довгостроковому використанні.

5.8 Розрахунок економічного ефекту від розробки програмного забезпечення

Оцінка економічної ефективності варіантів програмного забезпечення базується на розрахунку показників порівняльної економічної ефективності капітальних вкладень у їх розробку. Річний економічний ефект від використання розробленого ПЗ визначається на основі різниці між приведеними витратами на розробку та експлуатацію ПЗ за рік та витратами на закупку та експлуатацію його аналога (з урахуванням річного обсягу випуску продукції/послуг, робіт). Формула для цього розрахунку виглядає так:

$$E = (C_2 \cdot A_k - C_1) N_\Sigma \qquad (5.11)$$

де:

- E – річний економічний ефект.
- C_2 – витрати на закупку та експлуатацію аналога ПЗ.
- A_k – коефіцієнт порівняльної економічної ефективності.
- C_1 – приведені витрати на розробку та експлуатацію розробленого ПЗ.
- N_Σ – річний обсяг випуску продукції/послуг або обсяг робіт.

Цей розрахунок дозволяє оцінити економічну доцільність інвестицій у розробку ПЗ у порівнянні з варіантом закупки та використання існуючого аналогічного програмного продукту.

Для розрахунку економічного ефекту від використання розробленого програмного забезпечення порівняно з його аналогом, використовуються наступні формули:

1. Приведені витрати на виконання одиниці роботи (задачі) за допомогою ПЗ:

$C_1 = 95606,18 + 0,33 \cdot 48457,32 = 111597,09$ грн.

2. Приведені витрати на одиницю виконання роботи за аналогом:

$C_2 = 152061,38 + 0,33 \cdot 31499 = 162456,05$ грн.

Тоді, економічний ефект від використання розробленого ПЗ розраховується так:

$$E = (162456,05 \cdot 1,6 - 111597,09) = 120006,62 \text{ грн.}$$

Таким чином, економічний ефект від використання розробленого ПЗ складає 120006,62 грн.

Зведені дані за розрахунками економічного ефекту наведені у табл. 5.9.

Таблиця 5.9 – Економічний ефект розробленого ПЗ

Характеристика	Значення для	
	ПЗ	аналогу
Собівартість робіт (поточні експлуатаційні витрати програмного продукту), грн.	95606,18	152061,38
Сумарні витрати, які пов’язані з впровадженням ПЗ, грн.	48457,32	31499
Приведені витрати на одиницю робіт, грн.	111597,09	162456,05
Економічний ефект від використання програмного продукту, грн.	120006,62	

Для визначення терміну окупності програмного забезпечення використовується формула:

$$P_p = \text{Витрати на розробку} / \text{Річний економічний ефект} \quad (5.12)$$

де:

- P_p - термін окупності ПЗ.
- Витрати на розробку - 48457,32 грн.
- Річний економічний ефект - 120006,62 грн.

Термін окупності вираховується як:

$$P_p = 48457{,}32 / 120006{,}62 = 0{,}403 \text{ року} \quad (5.13)$$

Далі розраховуємо фактичний коефіцієнт економічної ефективності розробки (E_{fact}) і порівнюємо його з нормативним значенням коефіцієнта ефективності капітальних вкладень

$$E_n = 0{,}33 : E_{fact} = 1 / P_p = 1 / 0{,}403 = 2{,}48 \quad (5.14)$$

Фактичний коефіцієнт економічної ефективності розробки ПЗ становить 2,48, що більше нормативного коефіцієнта 0,33. Це означає, що розробка та впровадження ПЗ є економічно доцільною.

Отже, можна зробити висновок про технічну доцільність та економічну ефективність розробленого ПЗ.

5.9 Проектування бізнес-моделі пз за допомогою Lean Canvas

Lean Canvas (рисунок 5.2) – це адаптована версія бізнес-моделі Canvas, яку розробив Еш Мауря для використання в стартапах та підприємницьких проектах. Вона допомагає швидко та ефективно визначити ключові компоненти бізнесу. Ось її основні компоненти:

1. Проблема - опис проблем, які вирішує продукт або послуга.

2. Сегменти клієнтів - цільові групи або ринки, на які орієнтовано продукт.

3. Унікальна ціннісна пропозиція - чітке бачення того, чому продукт або послуга унікальні.

4. Рішення - конкретні функції або послуги, які вирішують зазначені проблеми.

5. Канали - шляхи, через які продукт або послуга досягають клієнта.

6. Доходи - механізми генерації доходів від продукту або послуги.

7. Структура витрат - основні витрати на створення та підтримку продукту або послуги.

8. Ключові метрики - показники, за якими вимірюється успіх бізнесу.

9. Конкурентна перевага - фактори, які ускладнюють копіювання або імітацію продукту або послуги конкурентами.

Lean Canvas використовується для швидкої візуалізації та тестування бізнес-ідей, що дозволяє підприємцям швидко адаптуватися і вносити зміни.

Рисунок 5.2 – Розгортання моделі Lean Canvas для розробки та впровадження програмного забезпечення

Розробка моделі Canvas дозволила чітко сформулювати та систематизувати бізнес-ідею, що полягає у створенні та впровадженні програмного забезпечення, спрямованого на зменшення часових витрат під час пошуку об’єктів та його ідентифікації в умовах недостатнього освітлення. Такий підхід має на меті значно покращити точність розпізнавання об’єктів, що є ключовим аспектом для багатьох сфер застосування, включаючи безпеку, контроль доступу та інші системи ідентифікації.

5.10 Висновки до розділу 5

Виконання економічної частини проекту з розробки програмного забезпечення охоплює ряд критично важливих аспектів, кожен з яких вносить свій внесок у загальний успіх та виправданість проекту. Починаючи з обґрунтування доцільності розробки ПЗ, важливо було проаналізувати ринкові потреби та визначити, як продукт може задовольнити ці потреби, пропонуючи унікальні рішення чи покращення у порівнянні з існуючими аналогами.

Аналіз конкурентоспроможності продукту виявив його сильні та слабкі сторони у порівнянні з існуючими рішеннями на ринку, допомагаючи визначити стратегії для підвищення його привабливості для потенційних користувачів.

Планування етапів розробки та виконання економічного аналізу проекту дозволило оцінити загальні витрати та ефективність розподілу ресурсів. Розрахунок проектних витрат на розробку ПЗ та витрат на його впровадження забезпечив чітке розуміння фінансової структури проекту.

Калькуляція витрат на альтернативні програмні рішення додала перспективи для оцінки цінності запропонованого ПЗ у порівнянні з іншими опціями на ринку. Розрахунок операційних витрат додатково підкреслив довгострокову вартість експлуатації та обслуговування ПЗ.

Нарешті, розрахунок економічного ефекту від розробки програмного забезпечення та проектування бізнес-моделі за допомогою Lean Canvas дав повне уявлення про потенційну вигоду та стратегічне позиціонування продукту на ринку.

ВИСНОВКИ

Дослідження інтелектуальної системи виявлення об'єктів у реальному часі на базі YOLOv5 є значним кроком вперед у сфері комп'ютерного зору та штучного інтелекту. Система, яка володіє передовими можливостями виявлення об'єктів, відкриває широкий спектр застосувань, включаючи безпеку, спостереження, автономні транспортні засоби, забезпечуючи високу точність, низьку затримку та масштабованість.

В роботі було ретельно вивчено необхідні функціональні та нефункціональні вимоги, а також системну архітектуру для ефективної роботи системи виявлення об'єктів. Наголошується на важливості забезпечення безпеки, зручності використання та конфіденційності даних.

Архітектура системи включає декілька ключових компонентів, таких як модулі введення, виявлення об'єктів, попередньої обробки, відстеження та генерації попереджень, а також забезпечує можливості інтеграції з існуючими робочими процесами.

З огляду на постійний розвиток технологій у галузі апаратного прискорення, алгоритмів глибокого навчання та комп'ютерного зору, очікується подальше покращення ефективності та точності систем виявлення об'єктів. Інтеграція нових технологій, таких як передові обчислення і мережі 5G, значно розширить можливості використання цих систем у різних сценаріях.

Ключовою метою дослідження було підвищення точності розпізнавання рухомих об'єктів роботизованим автомобілем для підвищення безпеки та ефективності руху на дорогах. Це передбачало виконання огляду існуючих методів і засобів розпізнавання, вивчення архітектур нейронних мереж, а також дослідження та вибір оптимальної моделі нейронної мережі для заданих цілей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dasiopoulou, Stamatia, et al. "Knowledge-assisted semantic video object detection." IEEE Transactions on Circuits and Systems for Video Technology 15.10 (2005): 121–124.

2. Tkachenko, O., Babichenko, D. (2022). Intellectualization of Moving Transport Objects Recognition. Digital Platform Information Technologies in Sociocultural Sphere, 5(2), 401-415. DOI: 10.31866/2617-796X.5.2.2022.270147. [CC BY 4.0].

3. «Штучна нейронна мережа» Українська Вікіпедія, Штучна нейронна мережа — Вікіпедія (wikipedia.org)

4. Борисов, Г.О., Гумен, Т.Ф. та Трапезон, К.О., 2020. Дослідження програмних особливостей об’єднання Android things на основі концепції Інтернету речей. Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, [e-journal] 31(70/1), с. 29-35. https://doi.org/10.32838/2663-5941/2020.1-1/0

5. Тимошин, Ю.А. та Орленко, С.П., 2018. Алгоритм розпізнавання обличчя людей на базі згорткової нейронної мережі. Адаптивні системи автоматичного управління, 1 (32), с.166-173.

6. Introduction to computer vision: what it is and how it works, 2018. DataRobot [online] 2 April. Available at: <https://www.datarobot.com/blog/introduction-to-computer-vision-what-it-is-and-how-it-works/> [Accessed 28 August 2022].

7. Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You Only Look Once: United, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), [e-journal] 27-30 June 2016. Institute of Electrical and Electronics Engineers. pp.779-788. https://doi.org/10.1109/CVPR.2016.91.

8. Ammar, A., Koubaa, A., Ahmed, M. and Saad, A., 2019. Aerial Images Processing for Car De-tection using Convolutional Neural Networks: Comparison between Faster R-CNN and YoloV3. Preprints, [e-journal] 17 October. DOI: 10.20944/preprints201910.0195.v1.

9. Svatiuk, D.R., Svatiuk, O.R. and Belei, O.I., 2020. Zastosuvannia zghortkovykh neironnykh merezh dlia bezpeky rozpiznavannia obiektiv u videopototsi [Application of convolutional neural networks for object recognition security in video stream]. Cybersecurity: Education, Science, Technique, [e-journal] 4(8), pp.97-112. https://doi.org/10.28925/2663-4023.2020.8.97112.

10. Nechiporenko, A.S., Gubarenko, E.V. and Gubarenko, M.S., 2019. Authentification of users of mobile devices by their motor reactions. Telecommunications and Radio Engineering, [e-jour-nal] 78(11), pp.987-1003. https://doi.org/10.1615/TelecomRadEng.v78.i11.60.

11. Markov, E., 2016. Fractal methods for extractingng artificial objects from the unmanned aerial vehicle images. Journal of Applied Remote Sensing, [e-journal] 10 (2), art. 25020. https://doi.org/10.1117/1.JRS.10.025020.

12. Bottou L., Bousquet O. The tradeoffs of large scale learning // Advances in neural information processing systems. – 2008. – С. 161–168.

13. Ross S., Gordon G., Bagnell D. A reduction of imitation learning and structured prediction to no-regret online learning // Proceedings of the fourteenth international conference on artificial intelligence and statistics. – 2011. – С. 627–635.

14. de Haan P., Jayaraman D., Levine S. Causal confusion in imitation learning // arXiv preprint arXiv:1905.11979. – 2019.

15. Torralba A., Efros A. A. et al. Unbiased look at dataset bias // CVPR, vol. 1, no. 2. – Citeseer, 2011. – С. 7.

16. Gupta A., Murali A., Gandhi D. P., Pinto L. Robot learning in homes: Improving generalization and reducing dataset bias // Advances in Neural Information Processing Systems. – 2018. – С. 90–91.

17. Wang Z., Bapst V., Heess N., Mnih V., Munos R., Kavukcuoglu K., de Freitas N. Sample efficient actor-critic with experience replay // arXiv preprint arXiv:1611.01224. – 2016.

18. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. – Cambridge, MA: MIT Press, 1998. – Т. 9.

19. Konda V. R., Tsitsiklis J. N. On actor-critic algorithms // SIAM Journal on Control and Optimization. – 2003. – Т. 42, № 4. – С. 43–66.

20. Watkins C. J., Dayan P. Q-learning // Machine Learning. – 1992. – Т. 8, № 3-4. – С. 279–292.

21. Gordon G. J. Stable function approximation in dynamic programming // Machine Learning Proceedings 1995. – Elsevier, 1995. – С. 261–268.

22. Tsitsiklis J. N., Van Roy B. Feature-based methods for large scale dynamic programming // Machine Learning. – 1996. – Т. 22, № 1-3. – С. 59–94.

23. Williams R. J. Reinforcement-learning connectionist systems. – College of Computer Science, Northeastern University, 1987.

24. Sutton R. S., McAllester D. A., Singh S. P., Mansour Y. Policy gradient methods for reinforcement learning with function approximation // Advances in Neural Information Processing Systems. – 2000. – С.57–63.

25. Riedmiller M., Peters J., Schaal S. Evaluation of policy gradient methods and variants on the cart-pole benchmark // Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on. – IEEE, 2007. – С. 254–261.

26. Silver D., Lever G., Heess N., Degris T., Wierstra D., Riedmiller M. Deterministic policy gradient algorithms. – 2014.

27. Mnih V., Badia A. P., Mirza M., Graves A., Lillicrap T., Harley T., Silver D., Kavukcuoglu K. Asynchronous methods for deep reinforcement learning // International Conference on Machine Learning. – 2016. – С. 1928–1937.

28. Arulkumaran K., Deisenroth M. P., Brundage M., Bharath A. A. Deep reinforcement learning: A brief survey // IEEE Signal Processing Magazine. – 2017. – Т. 34, № 6. – С. 26–38.

29. Grondman I., Busoniu L., Lopes G. A., Babuska R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). – 2012. – Т. 42, № 6. – С. 1291–1307.

30. Schulman J., Moritz P., Levine S., Jordan M., Abbeel P. High-dimensional continuous control using generalized advantage estimation // arXiv preprint arXiv:1506.02438. – 2015.

31. Yu G., Sethi I. K. Road-following with continuous learning // Intelligent Vehicles ’95 Symposium, Proceedings of the. – Detroit, MI, 1995.

32. Moriarty D. E., Handley S., Langley P. Learning distributed strategies for traffic control // Proceedings of the Fifth International Conference of the Society for Adaptive Behavior. – May 1998. – С. 437–446. – 1998.

33. Bojarski M., Del Testa D., Dworakowski D., Firner B., Flepp B., Goyal P., Jackel L. D., Monfort M., Muller U., Zhang J., Zhang X., Zhao J., Zieba K. End to End Learning for Self-Driving Cars. – May 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

34. Muller U., Ben J., Cosatto E., Flepp B., Cun Y. L. Off-road obstacle avoidance through end-to-end learning // Advances in Neural Information Processing Systems. – 2006. – С. 739–746.

35. Rausch V., Hansen A., Solowjow E., Liu C., Kreuzer E., Hedrick J. K. Learning a deep neural net policy for end-to-end control of autonomous

vehicles // 2017 American Control Conference (ACC). – IEEE, 2017. – C. 4914–4919.

36. Mechanical Simulation Corporation. CarSim. [Online]. Available: https://www.carsim.com

37. Eraqi H. M., Moustafa M. N., Honer J. End-to-end deep learning for steering autonomous vehicles considering temporal dependencies // arXiv preprint arXiv:1710.03804. – 2017.

38. Rothe R., Timofte R., Van Gool L. Dex: Deep expectation of apparent age from a single image // Proceedings of the IEEE International Conference on Computer Vision Workshops. – 2015. – C. 10–15.

39. Wang P., Chan C.-Y., de La Fortelle A. A reinforcement learning based approach for automated lane change maneuvers // 2018 IEEE Intelligent Vehicles Symposium (IV). – IEEE, 2018. – C. 1379–1384.

40. Pomerleau D. A. Alvinn: An autonomous land vehicle in a neural network // Advances in Neural Information Processing Systems 1. – 1989. – C. 305–313.

41. Pomerleau D. Neural network vision for robot driving // Intelligent Unmanned Ground Vehicles. – 1997. – C. 1–22.

42. Yu G., Sethi I. K. Road-following with continuous learning // Intelligent Vehicles ’95 Symposium, Proceedings of the. – Detroit, MI, 1995.

43. Lai T. L., Robbins H. Asymptotically efficient adaptive allocation rules // Advances in Applied Mathematics. – 1985. – Т. 6, № 1. – С. 4–22.

44. Bellemare M., Srinivasan S., Ostrovski G., Schaul T., Saxton D., Munos R. Unifying count-based exploration and intrinsic motivation // Advances in Neural Information Processing Systems. – 2016. – С. 1471–1479.

45. Schmidhuber J. A possibility for implementing curiosity and boredom in model-building neural controllers // Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats. – 1991. – С. 222–227.

46. Hecker S., Dai D., Van Gool L. End-to-end learning of driving models with surround-view cameras and route planners // Proceedings of the European Conference on Computer Vision (ECCV). – 2018. – С. 435–453.

47. Codevilla F., Muller M., López A., Koltun V., Dosovitskiy A. End-to-end driving via conditional imitation learning // 2018 IEEE International Conference on Robotics and Automation (ICRA). – IEEE, 2018. – С. 1–9.

48. Dosovitskiy A., Ros G., Codevilla F., Lopez A., Koltun V. CARLA: An open urban driving simulator // Proceedings of the 1st Annual Conference on Robot Learning. – 2017. – С. 1–16.

49. Codevilla F., Santana E., Lopez A. M., Gaidon A. Exploring the limitations of behavior cloning for autonomous driving // arXiv preprint arXiv:1904.08980. – 2019.

50. Paxton C., Raman V., Hager G. D., Kobilarov M. Combining neural networks and tree search for task and motion planning in challenging environments // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). – IEEE, 2017. – С. 6059–6066.

51. Bansal M., Krizhevsky A., Ogale A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst // arXiv preprint arXiv:1812.03079. – 2018.

52. Pan X., You Y., Wang Z., Lu C. Virtual to real reinforcement learning for autonomous driving // arXiv preprint arXiv:1704.03952. – 2017.

53. Muller M., Dosovitskiy A., Ghanem B., Koltun V. Driving policy transfer via modularity and abstraction // arXiv preprint arXiv:1804.09364. – 2018.

54. Dai X., Li C.-K., Rad A. B. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control // IEEE Transactions on Intelligent Transportation Systems. – 2005. – Т. 6, № 3. – С. 285–293.

55. Desjardins C., Chaib-draa B. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach // IEEE Transactions on Intelligent Transportation Systems. – 2011. – Т. 12, № 4. – С. 1248–1260.

56. Huang Z., Xu X., He H., Tan J., Sun Z. Parameterized Batch Reinforcement Learning for Longitudinal Control of Autonomous Land Vehicles // IEEE Transactions on Systems, Man, and Cybernetics: Systems. – 2017. – С. 1–12.

57. Chae H., Kang C. M., Kim B., Kim J., Chung C. C., Choi J. W. Autonomous braking system via deep reinforcement learning // 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). – IEEE, 2017. – С. 1–6.

58. Chen X., Zhai Y., Lu C., Gong J., Wang G. A Learning Model for Personalized Adaptive Cruise Control // Intelligent Vehicles Symposium (IV), 2017 IEEE. – 2017. – С. 379–384.

59. Zhao D., Xia Z., Zhang Q. Model-free optimal control based intelligent cruise control with hardware-in-the-loop demonstration [research frontier] // IEEE Computational Intelligence Magazine. – 2017. – Т. 12, № 2. – С. 56–69.

60. Zhao D., Wang B., Liu D. A supervised Actor-Critic approach for adaptive cruise control // Soft Computing. – 2013. – Т. 17, № 11. – С. 2089–2099.

61. Xia W., Li H., Li B. A control strategy of autonomous vehicles based on deep reinforcement learning // Computational Intelligence and Design (ISCID), 2016 9th International Symposium on, vol. 2. – IEEE, 2016. – С. 198–201.

62. Sallab A. E., Abdou M., Perot E., Yogamani S. End-to-end deep reinforcement learning for lane keeping assist // arXiv preprint arXiv:1612.04340. – 2016.

63. Zhang J., Cho K. Query-efficient imitation learning for end-to-end autonomous driving // arXiv preprint arXiv:1605.06450. – 2016.

64. Pan Y., Cheng C.-A., Saigol K., Lee K., Yan X., Theodorou E., Boots B. Agile autonomous driving using end-to-end deep imitation learning // Proceedings of Robotics: Science and Systems. – Pittsburgh, Pennsylvania, 2018.

65. Wang D., Devin C., Cai Q.-Z., Yu F., Darrell T. Deep object centric policies for autonomous driving // arXiv preprint arXiv:1811.05432. – 2018.

66. Porav H., Newman P. Imminent collision mitigation with reinforcement learning and vision // 2018 21st International Conference on Intelligent Transportation Systems (ITSC). – IEEE, 2018. – С. 958–964.

67. Wulfmeier M., Rao D., Wang D. Z., Ondruska P., Posner I. Large-scale cost function learning for path planning using deep inverse reinforcement learning // International Journal of Robotics Research. – 2017. – Т. 36, № 10. – С. 1073–1087.

68. Hecker S., Dai D., Van Gool L. End-to-end learning of driving models with surround-view cameras and route planners // Proceedings of the European Conference on Computer Vision (ECCV). – 2018. – С. 435–453.

69. Codevilla F., Muller M., López A., Koltun V., Dosovitskiy A. End-to-end driving via conditional imitation learning // 2018 IEEE International Conference on Robotics and Automation (ICRA). – IEEE, 2018. – С. 1–9.

70. Paxton C., Raman V., Hager G. D., Kobilarov M. Combining neural networks and tree search for task and motion planning in challenging environments // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). – IEEE, 2017. – С. 6059–6066.

71. Bansal M., Krizhevsky A., Ogale A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst // arXiv preprint arXiv:1812.03079. – 2018.

72. Geiger A., Lenz P., Stiller C., Urtasun R. Vision meets robotics: The KITTI dataset // International Journal of Robotics Research (IJRR). – 2013.

73. Geiger A., Lenz P., Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite // Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. – 2012.

74. “Waymo open dataset: An autonomous driving dataset,” 2019. [Online]. Available: https://www.waymo.com/open

75. Maddern W., Pascoe G., Linegar C., Newman P. 1 Year, 1000km: The Oxford RobotCar Dataset // The International Journal of Robotics Research (IJRR). – 2017. – Т. 36, № 1. – С. 3–15. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498

76. Huang X., Cheng X., Geng Q., Cao B., Zhou D., Wang P., Lin Y., Yang R. The apolloscape dataset for autonomous driving // arXiv preprint arXiv:1803.06184. – 2018.

77. Udacity Inc., “Udacity Self-driving Car Dataset,” 2018. [Online]. Available: https://github.com/udacity/self-driving-car

78. Ess A., Leibe B., Schindler K., van Gool L. A mobile vision system for robust multi-person tracking // Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on. – IEEE Press, June 2008.

79. Dollar P., Wojek C., Schiele B., Perona P. Pedestrian detection: A benchmark // Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on. – IEEE, 2009. – С. 304–311.

80. Yin H., Berger C. When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets // Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on. – IEEE, 2017. – С. 1–8.

81. NVIDIA Corporation, “Autonomous car development platform from NVIDIA DRIVE PX2,” 2018. [Online]. Available: https: //www.nvidia.com/en-us/self-driving-cars/drive-platform/

82. MobilEye, “The Evolution of EyeQ,” 2018. [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/

83. Intel Corporation. Cyclone V - Overview. – 2023. [Online]. Available: https://www.altera.com/products/fpga/cyclone-series/cyclone-v/overview.html

84. Liu S., Tang J., Zhang Z., Gaudiot J.-L. CAAD: Computer architecture for autonomous driving // arXiv preprint arXiv:1702.01894. – 2017.

85. Zabat, Stabile, Frascaroll, Browand. The Aerodynamic Performance of Platoons. ISSN 1055-1425. Архів оригіналу за 19 липня 2011.

86. Maurer M., Gerdes J. C., Lenz B., Winner H., Eds. Autonomous Driving: Technical, Legal and Social Aspects. – Heidelberg: Springer, 2016.

87. Bojarski M., Testa D. D., Dworakowski D., Firner B., Flepp B., Goyal P., Jackel L. D., Monfort M., Muller U., Zhang J., Zhang X., Zhao J., Zieba K. End to end learning for self-driving cars. – 2016.

88. Bojarski M., Yeres P., Choromanska A., Choromanski K., Firner B., Jackel L., Muller U. Explaining how a deep neural network trained with end-to-end learning steers a car. – 2017.

89. Ogata K. Modern Control Engineering. – 5th ed. – Pearson, 2010.

90. Falcone P., Borrelli F., Asgari J., Hrovat D. Low complexity MPC schemes for integrated vehicle dynamics control problems // International Symposium on Advanced Vehicle Control. – 2008.

91. Sommer L., Rick M., Folkers A., Buskens C. AO-Car: transfer of space technology to autonomous driving with the use of WORHP // Proceedings of the 7th International Conference on Astrodynamics Tools and Techniques. – 2018.

92. Knauer M., Buskens C. From WORHP to TransWORHP // Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques. – 2012.

93. Buskens C., Wassel D. The ESA NLP solver WORHP // Modeling and Optimization in Space Engineering. – 2013. – Т. 73. – С. 85–110.

94. Gu S., Holly E., Lillicrap T., Levine S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. – 2016.

95. Abbeel P., Coates A., Quigley M., Ng A. Y. An application of reinforcement learning to aerobatic helicopter flight // Advances in Neural Information Processing Systems 19, B. Schölkopf, J. C. Platt, T. Hoffman, Eds. – MIT Press. – 2007. – C. 1–8.

96. Isele D., Rahimi R., Cosgun A., Subramanian K., Fujimura K. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. – 2017.

97. Mirchevska B., Blum M., Louis L., Boedecker J., Werling M. Reinforcement learning for autonomous maneuvering in highway scenarios // Workshop Fahrerassistenzsysteme und automatisiertes Fahren. – 2017.

98. Sallab A. E., Abdou M., Perot E., Yogamani S. Deep reinforcement learning framework for autonomous driving // Electronic Imaging, Autonomous Vehicles and Machines. – 2017. – C. 70–76.

99. Lillicrap T. P., Hunt J. J., Pritzel A., Heess N., Erez T., Tassa Y., Silver D., Wierstra D. Continuous control with deep reinforcement learning. – 2015.

100. Mnih V., Badia A. P., Mirza M., Graves A., Lillicrap T., Harley T., Silver D., Kavukcuoglu K. Asynchronous methods for deep reinforcement learning // Proceedings of The 33rd International Conference on Machine Learning, M. F. Balcan and K. Q. Weinberger, Eds. – 2016. – T. 48. – C. 1928–1937.

101. Kosuru V. S. R., Venkitaraman A. K. Preventing the False Negatives of Vehicle Object Detection in Autonomous Driving Control Using Clear Object Filter Technique // 2022 Third International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE). – 2022. – IEEE. – C. 1–6.

102. Sharma T., Debaque B., Duclos N., Chehri A., Kinder B., Fortier P. Deep learning-based object detection and scene perception under bad weather conditions // Electronics. – 2022. – Т. 11, № 4. – С. 563.

103. Iftikhar S., Zhang Z., Asim M., Muthanna A., Koucheryavy A., Abd El-Latif A. A. Deep Learning-Based Pedestrian Detection in Autonomous Vehicles: Substantial Issues and Challenges // Electronics. – 2022. – Т. 11, № 21. – С. 35.

104. Alaba S., Gurbuz A., Ball J. A comprehensive survey of deep learning multisensor fusion-based 3d object detection for autonomous driving: Methods, challenges, open issues, and future directions. – 2022.

105. Ma X., Ouyang W., Simonelli A., Ricci E. 3D object detection from images for autonomous driving: a survey // arXiv preprint arXiv:2202.02980. – 2022.

106. Diwan T., Anirudh G., Tembhurne J. V. Object detection using YOLO: Challenges, architectural successors, datasets and applications // Multimedia Tools and Applications. – 2023. – Т. 82, № 6. – С. 92–97.

107. Du L., Zhang R., Wang X. Overview of two-stage object detection algorithms // Journal of Physics: Conference Series. – 2020. – Т. 1544, № 1. – С. 012033. IOP Publishing.

108. Carranza-García M., Torres-Mateo J., Lara-Benítez P., GarcíaGutiérrez J. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data // Remote Sensing. – 2020. – Т. 13, № 1. – С. 89.

109. N. Jain, S. Yerragolla, and T. Guha, "Performance Analysis of Object Detection and Tracking Algorithms for Traffic Surveillance Applications using Neural Networks," in 2019 Third International conference on ISMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2019: IEEE, pp. 690-696.

110. E. Güney, C. Bayilmiş, and B. Çakan, "An implementation of real-time traffic signs and road objects detection based on mobile GPU platforms," Ieee Access, vol. 10, pp. 86-93, 2022.

111. B. Mahaur, N. Singh, K. Mishra. Road object detection: a comparative study of deep learning-based algorithms // Multimedia Tools and Applications. – 2022. – Vol. 81, No. 10. – Pp. 142-148.

112. B. Mahaur, K. Mishra. Small-object detection based on YOLOv5 in autonomous driving systems // Pattern Recognition Letters. – 2023. – Vol. 168. – Pp. 115-122.

113. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I (pp. 21-37). Springer.

114. T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár. (2017). Focal loss for dense object detection. У Proceedings of the IEEE international conference on computer vision, с. 2980-2988.

115. Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

116. Rath, V. G. Sovit. Performance Comparison of YOLO Object Detection Models – An Intensive Study [Online] / V. G. Sovit Rath. – Available: https://learnopencv.com/performance-comparison-of-yolo-models. – Accessed: October 17, 2023.

ДОДАТОК А
ВІДОМІСТЬ РОБОТИ

Формат	№ п/п	Назва документу	Найменування об’єкта або виробу	Кількість сторінок
	1	Пояснювальна записка	КЦТПАР.122-22-1м.01.00.КР.ПЗ	125
Графічна частина				
А4	2	Об’єкт та предмет дослідження	КЦТПАР.122-22-1м.02.00.КР.ПЛ	1
А4	3	Мета та задачі дослідження	КЦТПАР.122-22-1м.03.00.КР.ПЛ	1
А4	4	Практична та теоретична цінність	КЦТПАР.122-22-1м.04.00.КР.ПЛ	1
А4	5	Сучасні архітектури для виявлення об'єктів	КЦТПАР.122-22-1м.05.00.КР.ПЛ	3
А4	6	Оцінка моделей виявлення об'єктів: Криві точність-повторюваність	КЦТПАР.122-22-1м.06.00.КР.ПЛ	1
А4	7	Порівняння характеристик алгоритмів виявлення об'єктів	КЦТПАР.122-22-1м.07.00.КР.ПЛ	
А4	8	Порівняння моделей YOLO на різноманітних графічних процесорах	КЦТПАР.122-22-1м.08.00.КР.ПЛ	1
А4	9	Порівняння продуктивності моделей YOLO	КЦТПАР.122-22-1м.09.00.КР.ПЛ	1
А4	10	Функціональні вимоги	КЦТПАР.122-22-1м.10.00.КР.ПЛ	1
А4	11	Нефункціональні вимоги	КЦТПАР.122-22-1м.11.00.КР.ПЛ	1
А4	12	Діаграма компонентів	КЦТПАР.122-22-1м.12.00.КР.ПЛ	1
А4	13	Експериментальні дослідження	КЦТПАР.122-22-1м.13.00.КР.ПЛ	1
А4	14	Ключові показники оцінювання моделі об'єктного виявлення	КЦТПАР.122-22-1м.14.00.КР.ПЛ	1
А4	15	Дослідження безпеки	КЦТПАР.122-22-1м.15.00.КР.ПЛ	1
А4	16	Висновки магістерської роботи	КЦТПАР.122-22-1м.16.00.КР.ПЛ	1

ДОДАТОК Б
ТЕХНІЧНЕ ЗАВДАННЯ

Б.1 ВСТУП

Розробка та аналіз тестових наборів даних для точного розпізнавання автомобілів за допомогою алгоритмів YOLO, що є частиною систем комп'ютерного зору для автономних транспортних засобів.

Б.2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Завдання на кваліфікаційну роботу магістра, затверджене факультетом «Цифрових технологій та автоматизації виробництва», кафедрою «Цифрових технологій та проєктно-аналітичних рішень» ТОВ «Технічний університет «МЕТІНВЕСТ ПОЛІТЕХНІКА».

Б.3 РОЗРОБКА ТА АНАЛІЗ ТЕСТОВИХ НАБОРІВ

- Аналіз ефективності розпізнавання за допомогою YOLO на різних наборах даних.
- Оцінка і впровадження методів збільшення розмаїтості даних для покращення точності моделі.

Б.4 ВИМОГИ ДО ПРОДУКТИВНОСТІ

– Система має можливість виявляти об'єкти в режимі реального часу за допомогою моделі YOLOv5 з частотою кадрів не менше 30 кадрів в секунду (FPS), , при цьому затримка обробки одного кадру не повинна перевищувати 33 мілісекунди.

– Система повинна класифікувати об'єкти з точністю розпізнавання не меншою за 95% для таких категорій: автомобілі, пішоходи, велосипедисти, та статичні перешкоди.

– Система повинна визначати відстань до виявлених об'єктів з точністю до 10 см на дистанції до 100 метрів.

– Система повинна оцінювати швидкість рухомих об'єктів з точністю до ±5 км/год для швидкостей до 120 км/год.

Б.5 ТЕСТУВАННЯ ТА ВАЛІДАЦІЯ

- Розробка комплексного плану тестування для оцінки точності розпізнавання на різних наборах даних.
- Виконання порівняльного аналізу різних конфігурацій YOLO для визначення оптимальних параметрів.

Б.6 ВИМОГИ ДО БЕЗПЕКИ ТА КОНФІДЕНЦІЙНОСТІ

Використання даних у відповідності до положень про конфіденційність та захист інформації.

Захист тестових наборів від несанкціонованого доступу.

Б.7 ВИМОГИ ДО ДОКУМЕНТАЦІЇ

Підготовка опису по розробці тестових наборів YOLO.

Б.8 ПЛАНУВАННЯ ТА ЕТАПИ РЕАЛІЗАЦІЇ

Визначення етапів розробки тестових наборів, включаючи збір даних, обробку, аналіз та оцінку.

Встановлення термінів для кожного етапу та моніторинг прогресу.

Б.8 БЮДЖЕТ ТА РЕСУРСИ

Визначення необхідних бюджетних та ресурсних витрат на реалізацію проекту.

Планування та розподіл ресурсів для забезпечення ефективності роботи над проектом.

ДОДАТОК В
ПЕРЕЛІК ПОЗНАК ТА СКОРОЧЕНЬ

Термін	Опис
Комп'ютерний зір (Computer Vision)	Галузь штучного інтелекту, що дозволяє комп'ютерам "бачити" та інтерпретувати зображення та відео.
Нейронна мережа (Neural Network)	Комп'ютерна модель для розпізнавання зразків та вивчення з даних, натхненна біологічними нейронними мережами.
Глибоке навчання (Deep Learning)	Підгалузь машинного навчання, що використовує множинні шари нейронних мереж для аналізу рівнів абстракції в даних.
YOLO (You Only Look Once)	Система реального часу для виявлення об'єктів, що ідентифікує об'єкти на зображеннях чи відео за один прохід.
Згорткова нейронна мережа (CNN)	Тип нейронної мережі, особливо ефективна для аналізу зображень.
Обробка зображень (Image Processing)	Обробка та аналіз цифрових зображень для поліпшення якості або екстракції інформації.
Детектування об'єктів (Object Detection)	Процес виявлення та ідентифікації об'єктів у цифрових зображеннях або відео.
Класифікація зображень (Image Classification)	Процес визначення категорії або класу зображення
Семантична сегментація (Semantic Segmentation)	Процес розподілу частин зображення на категорії або класи, базуючись на їх семантичному значенні.
Система реального часу (Real-time System)	Комп'ютерна система, здатна обробляти дані та відповідати на них у дуже короткі терміни.
Трекінг об'єктів (Object Tracking)	Процес відстеження руху об'єкта (або декількох об'єктів) через послідовність кадрів зображення або відео.
Комп'ютерне зір у робототехніці	Використання технік комп'ютерного зору в роботах для навігації, маніпуляції об'єктами, взаємодії з середовищем тощо.
Розпізнавання облич (Face Recognition)	Ідентифікація або перевірка особи людини на основі її обличчя.

ДОДАТОК Г
КОД НА МОВІ ПРОГРАМУВАННЯ PYTHON

```
heavy_traffic_threshold = 10
vertices1 = np.array([(465, 350), (609, 350), (510, 630), (2, 630)], dtype=np.int32)
vertices2 = np.array([(678, 350), (815, 350), (1203, 630), (743, 630)], dtype=np.int32)
x1, x2 = 325, 635
lane_threshold = 609
text_position_left_lane = (10, 50)
text_position_right_lane = (820, 50)
intensity_position_left_lane = (10, 100)
intensity_position_right_lane = (820, 100)
font = cv2.FONT_HERSHEY_SIMPLEX
font_scale = 1
font_color = (255, 255, 255)
background_color = (0, 0, 255)
cap = cv2.VideoCapture('sample_video.mp4')
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('traffic_density_analysis.avi', fourcc, 20.0, (int(cap.get(3)), int(cap.get(4))))
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        detection_frame = frame.copy()
        detection_frame[:x1, :] = 0
        detection_frame[x2:, :] = 0
        results = best_model.predict(detection_frame, imgsz=640, conf=0.4)
        processed_frame = results[0].plot(line_width=1)
        processed_frame[:x1, :] = frame[:x1, :].copy()
        processed_frame[x2:, :] = frame[x2:, :].copy()
        cv2.polylines(processed_frame, [vertices1], isClosed=True, color=(, 255, 0), thickness=2)
        cv2.polylines(processed_frame, [vertices2], isClosed=True, color=(255, 0, 0), thickness=2)
        bounding_boxes = results[0].boxes
        vehicles_in_left_lane = 0
        vehicles_in_right_lane = 0
        for box in bounding_boxes.xyxy:
            if box[0] < lane_threshold:
                vehicles_in_left_lane += 1
            else:
                vehicles_in_right_lane += 1
```

```
        traffic_intensity_left = "Hard" if vehicles_in_left_lane > heavy_traffic_threshold else "Low"
        traffic_intensity_right = "Hard" if vehicles_in_right_lane > heavy_traffic_threshold else "Low"
        cv2.rectangle(processed_frame, (text_position_left_lane[0]-10, text_position_left_lane[1] - 25),
                      (text_position_left_lane[0] + 460, text_position_left_lane[1] + 10), background_color, -1)
        cv2.putText(processed_frame, f'Vehicles in Left Lane: {vehicles_in_left_lane}',
text_position_left_lane,
                    font, font_scale, font_color, 2, cv2.LINE_AA)
        cv2.rectangle(processed_frame, (intensity_position_left_lane[0]-10, intensity_position_left_lane[1] -
25),
                      (intensity_position_left_lane[0] + 460, intensity_position_left_lane[1] + 10),
background_color, -1)
        cv2.putText(processed_frame, f'Traffic Intensity: {traffic_intensity_left}', intensity_position_left_lane,
                    font, font_scale, font_color, 2, cv2.LINE_AA)
        cv2.rectangle(processed_frame, (text_position_right_lane[0]-10, text_position_right_lane[1] - 25),
                      (text_position_right_lane[0] + 460, text_position_right_lane[1] + 10), background_color, -1)
        cv2.putText(processed_frame, f'Vehicles in Right Lane: {vehicles_in_right_lane}',
text_position_right_lane,
                    font, font_scale, font_color, 2, cv2.LINE_AA)
        cv2.rectangle(processed_frame, (intensity_position_right_lane[0]-10, intensity_position_right_lane[1]
- 25),
                      (intensity_position_right_lane[0] + 460, intensity_position_right_lane[1] + 10),
background_color, -1)
        cv2.putText(processed_frame, f'Traffic Intensity: {traffic_intensity_right}',
intensity_position_right_lane,
                    font, font_scale, font_color, 2, cv2.LINE_AA)
        out.write(processed_frame)
    else:
        break
cap.release()
out.release()
```

ДОДАТОК Д

РЕЗУЛЬТАТИ АПРОБАЦІЇ ДОСЛІДЖЕНЬ


СЕРТИФІКАТ
УЧАСНИКА

Якимов Юрій Миколайович

взяв(-ла) участь у VI Міжнародній науковій конференції

ТЕХНОЛОГІЇ, ІНСТРУМЕНТИ ТА СТРАТЕГІЇ РЕАЛІЗАЦІЇ НАУКОВИХ ДОСЛІДЖЕНЬ

4 СЕРПНЯ 2023 РОКУ ♦ ДНІПРО, УКРАЇНА

ВІЦЕ-ПРЕЗИДЕНТ МЦНД
ГОЛОВА ОРГКОМІТЕТУ
РАБЕЙ НАСТАСІЯ

МЦНД
МІЖНАРОДНИЙ ЦЕНТР НАУКОВИХ ДОСЛІДЖЕНЬ

Матеріали учасника конференції опубліковані та знаходяться у відкритому доступі за посиланням:
https://doi.org/10.36074/mcnd-04.08.2023

Організаційний комітет конференції рекомендує на підставі цього сертифікату зарахувати не менше 0,1 кредиту ЄКТС за результатами самоосвіти, як форми професійного навчання, науково-педагогічним та педагогічним працівникам, державним службовцям та іншим фахівцям, що проходять стажування.

Посвідчення УкрНТЕІ
№ 88 від 07.01.2023

МІЖНАРОДНИЙ ЦЕНТР НАУКОВИХ ДОСЛІДЖЕНЬ

Рисунок Д.1 – Технології, інструменти та стратегії реалізації наукових досліджень

Рисунок Д.2 – Проблеми інформатики та моделювання ПІМ – 2023

Рисунок Д.3 – MININGMETALTECH 2023