Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, факультету)

Кафедра Автоматизації та інтелектуальних інформаційних технологій
(повна назва кафедри)

Магістерська кваліфікаційна робота
на тему:

«Система розпізнавання об'єктів дорожнього руху»

Виконав: студент 2-го курсу,
групи 1АКІТР-23м
спеціальності 174 – Автоматизація та комп'ютерно-інтегровані технології та робототехніка
(шифр і назва спеціальності)

Ігор СЛОБОДЯН
(ім'я та прізвище)

Керівник: к.т.н., доцент. каф. АІІТ
Володимир КОЦЮБИНСЬКИЙ
(ім'я та прізвище)
«___» _____ 2024 р.

Опонент д.т.н. проф кафедри КН
Ярослав ІВАНЧУК
«16» грудня 2024 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф.
_____Олег БІСІКАЛО.
(ім'я та прізвище)
«16» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти другий (магістерський)
Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Зав.кафедри АІІТ
, д.т.н., проф. Олег БІСІКАЛО
« 19 » вересня 2024
року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
студенту Слободяну Ігорю Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Система розпізнавання об'єктів дорожнього руху
керівник роботи Коцюбинський Володимир Юрійович, доцент кафедри АІІТ
затверджені наказом ВНТУ від "17" вересня 2024 року №310
2. Строк подання студентом роботи 12 грудня 2024 року
3. Вихідні дані до роботи: Система розпізнавання об'єктів. Мінімальні системні вимоги Операційна система: Ubuntu. Мова програмування: Kotlin. Обсяг оперативної пам'яті: 8гб.
CPU intel core I 5 3.4GHz
4. Вступ. Аналіз предметної області та огляд існуючих аналогів та систем. Вибір та огляд технологій для розробки даної системи. Вибір архітектури системи. Розробка та тестування розробленої системи. Економічний розділ. Висновки, Список використаних джерел. Додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) UML діаграма прецедентів, UML діаграма розгортання, UML діаграма класів, UML діаграма функціонування системи, екранні форми системи, презентація

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Володимир КОЦЮБИНСЬКИЙ к.т.н., доцент. каф. АІІТ		
4	Володимир КОЗЛОВСЬКИЙ к.е.н , професор. каф. ЕПтаВМ		

7. Дата видачі завдання "18" _вересня_ 2024 року

Календарний план

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Дослідження об'єкта автоматизації та постановка задач дослідження	19.09.2024р.	
2	Дослідження основних аспектів системи розпізнавання	22.09.2024р.	
3	Аналіз і порівняння існуючих систем розпізнавання	01.10.2024р.	
4	Проектування системи розпізнавання	15.11.2024р.	
5	Апробація результатів дослідження	20.11.2024р.	
6	Публікації		
7	Оформлення пояснювальної записки, графічного матеріалу і презентації	29.11.2024р.	
8	Графічні матеріали: – UML-діаграми – вигляд екранів розробленого веб додатку	04.10.2024 р 18.10. 2024 р.	
9	Захист МКР	20.12. 2024 р.	

Студент (підпис) Ігор СЛОБОДЯН (ім'я і ПРІЗВИЩЕ)

Керівник роботи (підпис) Володимир КОЦЮБИНСЬКИЙ (ім'я і ПРІЗВИЩЕ)

АНОТАЦІЯ

УДК 004.75:371.3

Слободян І.О. Система розпізнавання об'єктів дорожнього руху. Магістерська кваліфікаційна робота зі спеціальності 174 - Автоматизація, комп’ютерно-інтегровані технології та робототехніка. Вінниця: ВНТУ, 2024,113 с.

На укр. мові. Бібліогр.: 26 назв; рис.: 15; табл. 7.

У теоретично-методичній частині роботи розглянуто та проаналізовано предметну область та об’єкт автоматизації - процес розпізнавання об’єктів та графічних зображень, основні принципи реалізації та елементи системи. Визначено основні проблеми, переваги, недоліки та перспективи розвитку системи в майбутньому.

У практичній частині було розроблено алгоритмічне та програмне забезпечення для реалізації основних функцій системи, визначені вимоги що до архітектури системи та проведено тестування програмного забезпечення.

Метою роботи є підвищення ефективності процесу розпізнавання об’єктів за рахунок впровадження технології машинного навчання

Ключові слова: система, розпізнавання об’єктів, оптимізація збору даних, машинне навчання

ABSTRACT

Slobodian I.O. Traffic objects recognition system. Master's qualification work in specialty 174 - Automation, computer-integrated technologies and robotics. Vinnytsia: VNTU, 2024, 113p.

In Ukrainian. Bibliography: 26 titles; Fig.: 15; Table 7.

In the theoretical and methodological part of the work, the subject area and the object of automation - the process of recognizing objects and graphic images, the main principles of implementation and elements of the system - are considered and analyzed. The main problems, advantages, disadvantages and prospects for the development of the system in the future are identified.

In the practical part, algorithmic and software were developed to implement the main functions of the system, requirements for the system architecture were determined, and software testing was carried out.

The aim of the work is to increase the efficiency of the object recognition process by implementing machine learning technology. Keywords: system, object recognition, data collection optimization, machine learning

ЗМІСТ

ВСТУП 3
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ДОРОЖНЬОГО РУХУ 6
1.1 Огляд процесу розпізнавання об'єктів та його потреб 6
1.2 Аналіз процесу автоматизації розпізнавання об'єктів 8
1.3 Проблеми при використанні моделей для розпізнавання 11
1.4 Порівняння ручного та автоматизованого розпізнавання об'єктів на фото 14
1.5 Можливості, способи та перспективи розвитку системи у майбутньому 16
1.6 Аналіз існуючих систем розпізнавання об'єктів 18
2 ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ 24
2.1 Постановка задачі та визначення основних функцій системи 24
2.2 Проектування архітектури системи 28
2.3 Аналіз та обґрунтування технологій розробки 30
2.4 Аналіз моделей розпізнавання об'єктів та їх навчання 38
3 РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ 44
3.1 Моделювання архітектури системи 44
3.2 Розробка послідовності функціонування системи розпізнавання об'єктів дорожнього руху 46
3.3 Моделювання класів системи розпізнавання об'єктів дорожнього руху 50
3.4 Розробка програмного забезпечення 56
3.5 Тестування програмного забезпечення 64
4 ЕКОНОМІЧНИЙ РОЗДІЛ 71

4.1 Технологічний аудит розробленої системи розпізнавання об’єктів дорожнього руху 71
4.2 Розрахунок витрат на розроблення системи розпізнавання об’єктів дорожнього руху 78
4.3 Розрахунок економічного ефекту від можливої комерціалізації нашої розробки 82
ВИСНОВКИ..... 91
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 93
ДОДАТКИ..... 97
Додаток А (Обов’язковий) Технічне завдання 98
Додаток Б (обов’язковий). Ілюстративна частина..... 100
Додаток В (обов’язковий) Лістинг програми..... 103
Додаток Г (обов’язковий) Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень..... 108

ВСТУП

У сучасному суспільстві інформації, де обсяги даних зростають кожен день, автоматизація процесів і розпізнавання предметів у веб-додатках стають все більше важливими. Це особливо вірно для таких сфер як безпека, онлайн-торгівля, системи навігації автомобілів, створення медичних інструментів та багато інших галузей. Поєднання новітніх технологій автоматичного аналізу зображень і відео значно підвищує ефективність роботи таких систем, зменшуючи навантаження на людський ресурс а також мінімізуючи ризик помилок через людський фактор. Ці новаторські підходи не лише зменшують потрібність ручного втручання а й відкривають нові можливості для створення рішень що раніше здавалися неможливими особливо у сферах, де швидкість і точність аналізу мають вирішальне значення. У сучасному суспільстві, яке характеризується безпрецедентним зростанням обсягів інформації, обробка даних стала критично важливою для багатьох галузей. Щодня створюються терабайти нових даних, які вимагають швидкої та точної обробки.

Ці технології вже впливають на численні сфери людської діяльності: від забезпечення безпеки до створення нових бізнес-моделей в онлайн-торгівлі, розробки інструментів для медичної діагностики та вдосконалення системи автономної навігації. Їх впровадження дозволяє значно підвищити ефективність роботи системи, зменшити залежність від людського ресурсу та мінімізувати ризик помилок, які можуть виникнути

Актуальність теми. обумовлена розвитком нових технологій, таких як машинне навчання, штучний інтелект та комп'ютерне бачення. Алгоритми глибокого навчання стали стандартом у задачах розпізнавання зображень, завдяки їхній здатності обробляти великі обсяги даних із високою точністю. [1]. Автоматизація завдань обробки даних та розпізнавання об'єктів сприяє значному підвищенню точності та зменшенню часу, витраченого на виконання рутинних завдань[2].

Це відкриває багато можливостей для впровадження подібних систем у веб-додатках, які стають важливим інструментом для різних сферах промисловості та послуг. У майбутньому розвиток цих технологій стане великим фактором для інноваційного прогресу, сприяючи створенню більш зручних, точних і ефективних систем у всіх аспектах життя.

Метою дослідження є підвищення ефективності процесу розпізнавання об’єктів за рахунок впровадження технології машинного навчання

Робота спрямована на вирішення важливих задач в сфері автоматизації та поліпшення процесів, що пов'язані з розпізнаванням об'єктів у різних варіантах використання. Веб-додаток, розроблений в цій роботі, має показувати здатність знаходити і класифікувати предмети у реальному часі що дає можливість інтегрувати його в різні області прикладних знань.

Досягнення цієї цілі потребує виконання низки завдань, серед яких: створення основи системи, підтримка її розширюваності та інтеграція з веб-технологіями. Також важливим моментом роботи є перевірка та покращення системи для досягнення високої точності розпізнавання при менших витратах обчислень. У рамках дослідження планується провести порівняння ефективності різних способів, таких як глибокі нейронні мережі (CNN), а також їх поєднання з старими способами комп'ютерного бачення. Таким чином, кінцевий продукт не лише доведе технічну доцільність, але й стане прикладом для застосування сучасних технологій у вирішенні реальних задач.

Об'єктом дослідження є процес розпізнавання об’єктів та графічних зображень з метою покращення збирання даних, та підвищення якості інформації, що отримується.

Предметом дослідження є методи отримання та аналізу інформації, щодо розпізнавання графічних зображень в реальному часі.

Науково-технічний результат полягає у удосконаленні процесу автоматизації обробки об'єктів дорожнього руху шляхом розробки клієнт-серверної системи з використанням технологій машинного навчання для вирішення задач пов'язаних із патернізацією великих наборів даних

Практична цінність полягає у розробці алгоритмічного та програмного забезпечення, яке робить процес розпізнавання та класифікації об’єктів на фото, більш ефективним.

Апробація. Представлені в роботі результати апробовані під час участі в конференції ВНТУ: «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)».

Публікації. Слободян І.О**.** Квєтний Р.Н. «Використання моделей штучного інтелекту у мовах програмування kotlin, java», «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», 2024. URL:https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20902/17351

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ ДЛЯ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ ДОРОЖНЬОГО РУХУ

1.1 Огляд процесу розпізнавання об’єктів та його потреб

Використання штучного інтелекту для вирішення простих, та у деяких випадках навіть складних задач стало невід’ємною частиною процесу розробки новітніх систем. Це дозволило компаніям прискорити та покращити вирішення задач, що пов’язані зі збором даних, обробкою інформації та створення складних алгоритмів. Глибокі нейронні мережі є одним із найпотужніших інструментів для розпізнавання об'єктів у сучасних автоматизованих системах [3]. Також штучний інтелект може застосовуватися у тих випадках, коли покращення системи не є критичним, а впровадження шляхом використання альтернативних методів їх реалізації було нерентабельним, що дозволяє проводити збір даних більш точно та покращити точність і якість складних алгоритмів. Створення складних систем вимагає великих наборів даних та виконання другорядних процесів для оптимізації їх виконання. Автоматизація процесів обслуговування алгоритмів та збору даних дозволяє компаніям створювати точніші, оптимізованіші та складніші системи, що в умовах широкої конкуренції та стрімкого розвитку, стає невід’ємною частиною процесу їх розробки. Оптимізація алгоритмів дозволяє зменшити витрати ресурсів і підвищити продуктивність [4].

Застосування штучного інтелекту має ряд переваг

- Автоматизація рутинних завдань;
- Швидкість і масштабованість — швидка обробка даних та можливість залучення миттєвих додаткових ресурсів залежно від обсягів задачі;
- Економія — ресурсів дозволяє економити ресурси розробників та оптимізувати навантаження;

- Оптимізація процесів — скорочення часу виконання операцій і мінімізація простоїв.

Впровадження штучного інтелекту дозволяє компаніям бути гнучкішими, ефективнішими та орієнтованими на майбутнє, що стає критично важливим у сучасному динамічному середовищі.

Але незважаючи на переваги впровадження ШІ має ряд недоліків:

- Високі початкові витрати — Потреба в спеціалізованому обладнанні та програмному забезпеченні;
- Необхідність у великій кількості даних — Для якісного навчання алгоритмів необхідно збирати та обробляти великі обсяги даних;
- Складність навчання — Навчання моделі є складним процесом із кількома етапами, які впливають на якість та точність моделі. Будь яка помилка може призвести до помилок у роботі моделі;
- Підвищення системних вимог — моделі штучного інтелекту вимагають додаткових ресурсів для роботи.

Компанії, що впроваджують ШІ повинні врахувати ці недоліки, оскільки це дозволяє заздалегідь оцінити ризики та уникнути можливих проблем.

Для ефективного провадження ШІ у систему необхідно врахувати наступні ключові фактори

- Чітке визначення мети та задач — формулювання конкретних цілей, які повинен вирішувати ШІ та Визначення очікуваних результатів і очікувану ефективність;
- Інфраструктура та обладнання — переведення задачі під виконання ШІ може призвести до потреб у зміні інфраструктури системи та збільшити навантаження на обладнання;
- Бюджетування та планування — необхідно провести оцінку вартості провадження та врахувати ризики для розуміння рентабельності процесу.

Використання штучного інтелекту відкриває можливості до спрощення у виконанні задач, та оптимізації ресурсів, дозволяє створити більш складні

алгоритми та системи, але провадження таких змін залежить від правильного підходу та планування.

1.2 Аналіз процесу автоматизації розпізнавання об’єктів

В контексті автоматизації процесу розпізнавання об’єктів, аналіз об'єкта автоматизації охоплює багато ключових аспектів.

Основна мета аналізу - визначення потреб та можливостей для впровадження рішень для подальшої автоматизації, а також ідентифікація переваг, ризиків та вимог, пов'язаних з автоматизацією.

Під час аналізу об'єкта можна використовувати різні підходи та методи, спрямовані на комплексне вивчення та оптимізацію цього процесу. Деякі з них включають:

- Системний аналіз:
  - Опис процесів і взаємодії. Розгляд ідентифікації ключових етапів і взаємодії між елементами системи;
  - Визначення вимог. Аналіз та визначення функціональних і нефункціональних вимог для досягнення поставлених цілей.
- Технічний аналіз:
  - Вибір технологій та платформ. Визначення оптимальних технічних рішень для інтеграції, враховуючи доступні технології та платформи.
- Економічний аналіз:
  - Оцінка вартості інтеграції. Розгляд витрат на впровадження і підтримку системи;
  - Визначення ефективності від інтеграції. Аналіз ефективності інтеграції системи розпізнавання з точки зору виконання задачі.
- Оцінка впливу на процес збору даних

- Збір та аналіз статистики. Використання даних та статистики для оцінки впливу системи розпізнавання на збір даних;
- Оцінка якості розпізнавання.

Автоматизація процесу розпізнавання об’єктів передбачає використання спеціалізованих алгоритмів і програмного забезпечення для ефективного та швидкого виконання задачі, з можливістю швидкого залучення додаткових ресурсів у залежності від обсягів та потреб. Масштабованість автоматизованих системи забезпечують швидкість та точність при розпізнанні об’єктів та дозволяють економити ресурси при низькому навантаженні, що особливо важливо для великих компаній, обсяги даних яких можуть варіюватися від надзвичайно великих, до зовсім малих.

Особливості об'єкта автоматизації. Процес розпізнавання об’єктів та графічних зображень є багатокомпонентним завданням, яке включає в себе аналіз різних факторів для виконання задачі.

Основні етапи процесу розпізнання об’єктів:

- Отримання набору даних;
- Визначення потреб задачі;
- Перегляд наборів даних.
- Формування, вихідних матеріалів відповідно до потреб задачі

Автоматизована система повинна не лише виконувати задачі із розпізнання, але й мати можливість масштабування для економії ресурсів під час відсутності задачі.

Складові об'єкта автоматизації. Процес розпізнавання об’єктів включає в себе такі ключові елементи, що потребують автоматизації:

- Розпізнання об’єктів. Цей етап включає безпосереднє визначення об’єкта на зображенні;
- Класифікація об’єктів. Це етап, який потрібен для визначення типу об’єктів шляхом аналізу його характеристик;

- Визначення його параметрів та координат. Етап що має на меті отримання точного розташування об’єкта в просторі;
- Визначення точності здогадки. Показує впевненість моделі що усі попередні етапи є вірними та відповідають її розумінню характеристик розпізнаного об’єкта;
- Отримання результатів. Безпосереднє отримання результатів розпізнання.

Критерії ефективності автоматизації. Для оцінки ефективності автоматизованої системи необхідно враховувати кілька ключових критеріїв:

- Час, який затрачений на розпізнання. Однією з основних переваг автоматизації є скорочення часу на виконання розпізнання. Система повинна забезпечувати швидке виконання цього завдання, навіть для складних фото із великою кількістю об’єктів;
- Точність розпізнання. Автоматизовані системи повинні забезпечувати високу точність розпізнання;
- Якість масштабування системи. Через непостійність таких задач та різницю у обсягах даних система повинна витрачати якнайменше ресурсів під час своєї роботи за відсутності задач, та швидко залучати додаткові ресурси для швидкого виконання завдання. Це забезпечить економію ресурсів та коштів, та забезпечить швидкість, яку не досягти іншими відомими способами.

Виклики при автоматизації розпізнання об’єктів на фото. Незважаючи на всі переваги автоматизації, є й певні виклики, з якими можна стикнутися в процесі впровадження:

- Складність збору даних. Навчання моделі потребує великих наборів даних, які повинні бути збалансовані та не приводити до її перенавчання;
- Великий обсяг моделі. Моделі високої точності зазвичай важать багато. Це може призвести до довгої ініціалізації системи, а також до

потреб конфігурації репозиторіїв із додатковими системами зберігання, як на приклад GitHub LFS(Large File Storage);

- Необхідність адаптації. Автоматизація процесу розпізнавання може привести до необхідності адаптації частини системи через розбіжності у змінних різних бібліотек;
- Безпека.Використання будь яких сторонніх бібліотек веде за собою можливі вразливості, які потрібно моніторити та вчасно виправляти, штучний інтелект не є вийнятком. Безпека моделей машинного навчання є важливим аспектом їх застосування в реальних умовах, оскільки вони вразливі до атак, що можуть викривити результати розпізнавання [5].

1.3 Проблеми при використанні моделей для розпізнавання

Найкращим способом для швидкої роботи системи розпізнавання, чи інших видів штучного інтелекту який працює із матрицями є використання графічного процесора (GPU) – окремого пристрою що був створений для відображення графіки шляхом паралельних обчислень та ідеально підходить для важких матричних перетворень. Для взаємодії процесора CPU та графічного процесора GPU використовується ядро Kernel, у яке записуються дані для паралельного обчислення. У наш час існує багато готових інструментів для більш складної алгоритміки роботи із Kernel

Основними інструментами роботи із GPU є:

- CUDA (Compute Unified Device Architecture):
  - Спеціалізований інструмент для роботи з NVIDIA GPU.
  - Забезпечує високу продуктивність завдяки глибокій інтеграції з апаратним забезпеченням;
  - Має багатий набір бібліотек, зокрема cuDNN для глибокого навчання, що робить його стандартом у сфері ШІ;

    - Обмеження: Працює лише з графічними процесорами NVIDIA.
- ROCm (Radeon Open Compute):
    - Відкрита платформа для AMD GPU;
    - Підтримує аналогічний до CUDA стиль програмування через HIP;
    - Розвивається швидкими темпами, але поки має менше підтримки в популярних фреймворках ШІ, таких як TensorFlow або PyTorch.
- Vulkan Compute:
    - Графічний API із підтримкою обчислювальних шейдерів;
    - Забезпечує низькорівневий доступ до апаратних ресурсів;
    - Ідеально підходить для інтеграції з графікою, але вимагає високої технічної підготовки.
- OpenCL (Open Computing Language):
    - Відкрита платформа, яка працює з GPU, CPU та іншими пристроями різних виробників;
    - Перевага: Універсальність;
    - Недолік: Менша продуктивність і обмежений функціонал у порівнянні з CUDA чи ROCm.
- SYCL (Compute Abstraction Layer):
    - Сучасна альтернатива OpenCL, що забезпечує підтримку різних типів пристроїв (CPU, GPU, FPGA);
    - Інтегрується з OneAPI для забезпечення універсальності.

Не зважаючи на велику кількість інструментів для використання паралельного обчислення моделі штучного інтелекту зазвичай мають підтримку лише одного, який вважається стандартом – CUDA. CUDA дозволяє значно підвищити продуктивність моделей машинного навчання завдяки

оптимізації обчислень на GPU[6]. Що обмежує вибір GPU як для навчання так і для подальшої роботи. Причиною цьому є відсутність конкурентноздатних інструментів на момент зародження алгоритмів штучного інтелекту, та формування постулатів.

Як наслідок це може призвести до непередбачуваних зростань у ціні серверів із GPU, як це було у період 2022-2023 років, що було пов’язано із бумом “майнингу криптовалют”, алгоритми яких також почали використовувати CUDA для обчислень.

Проте напрямок ШІ знаходиться на початкових етапах своєї історії та має перспективи розвитку та вирішення цієї проблеми

Основні перспективи подолання монополії CUDA:

- Розвиток ROCm. AMD активно працює над поліпшенням ROCm, що може зробити GPU цієї компанії більш конкурентоспроможними у сфері ШІ;
- Універсальні інструменти. Впровадження стандартів, таких як SYCL чи OneAPI, дозволить уникнути прив’язки до одного виробника й створити універсальні рішення для гетерогенних обчислень;
- TPU та альтернативи. Tensor Processing Units (TPU) від Google спеціалізуються на задачах глибокого навчання, забезпечуючи високу продуктивність та енергоефективність.

Зараз починається тенденція впровадження альтернативних інструментів для навчання та запуску різних рушіїв, що може з часом вирішити проблему монополізації напрямку інструментами компанії Nvidia

1.4 Порівняння ручного та автоматизованого розпізнавання об’єктів на фото

Процес розпізнавання об’єктів на фото є важливою складовою сучасних технологій, особливо в контексті автоматизації та аналізу даних. Однак ручний підхід до цього процесу часто супроводжується низкою обмежень, таких як суб’єктивність, низька швидкість і трудомісткість. У таких умовах автоматизація стає необхідною для забезпечення ефективності та точності в розпізнаванні об’єктів.

Проблеми ручного розпізнавання об’єктів

Ручне розпізнавання об’єктів на фото дозволяє операторам чи фахівцям враховувати специфічні деталі зображень, однак воно має суттєві недоліки:

- Суб'єктивність і людський фактор. Результати розпізнавання можуть залежати від досвіду, уваги та професійних якостей оператора. Ручний підхід може призводити до помилок через втому чи брак часу.
- Низька швидкість. Процес потребує значного часу для аналізу кожного зображення, особливо якщо обсяг даних великий;
- Обмежена масштабованість. Ручний аналіз стає неможливим при роботі з великими наборами зображень, наприклад, для систем відеоспостереження чи аналізу супутникових даних;
- Відсутність стандартизації. Різні фахівці можуть використовувати різні підходи до аналізу зображень, що створює труднощі з досягненням послідовності результатів.

Переваги автоматизованого розпізнавання об’єктів

Автоматизація процесу розпізнавання об’єктів значно підвищує ефективність завдяки використанню алгоритмів комп’ютерного зору та штучного інтелекту (ШІ). Основні переваги автоматизованого підходу:

- Точність і об’єктивність:

  - Використання алгоритмів машинного навчання усуває суб’єктивність і вплив людського фактора;
  - Моделі ШІ здатні виявляти об’єкти з високою точністю навіть на складних або зашумлених зображеннях.
- Швидкість обробки:
  - Алгоритми можуть обробляти тисячі зображень за кілька хвилин, що значно перевищує можливості людини.
- Масштабованість:
  - Автоматизовані системи легко адаптуються до великих обсягів даних, забезпечуючи ефективний аналіз у реальному часі.
- Гнучкість:
  - Системи можуть бути налаштовані на розпізнавання об’єктів у різних умовах: з низьким освітленням, при перекриттях, або на складних фонах.

Автоматизовані системи забезпечують на 65% швидшу обробку даних, ніж ручні методи, і значно знижують ризик суб'єктивних помилок[7].

Більш детальне порівняння ручного та автоматизованого підходу зображено в таблиці 1.1:

Таблиця 1.1 Порівняння ручного та автоматизованого підходу

Критерій	Ручний підхід	Автоматизований підхід
Швидкість	Низька, потребує значного часу на аналіз кожного зображення	Висока, обробка тисячі зображень за хвилини, за необхідності
Об'єктивність	Суб’єктивний підхід через вплив людських вражень	Об’єктивний, ґрунтується на чітко визначених параметрах
Гнучкість	Низька, потребує значних зусиль для адаптації до нових завдань	Висока, можливість перенавчання моделей для різних сценаріїв

Точність	Залежить від оператора, можливі помилки через людський фактор	Висока, завдяки використанню алгоритмів ШІ
Витрати	Потребує залучення кваліфікованого персоналу, висока трудомісткість	Початкові витрати на розробку та налаштування, але низькі експлуатаційні
Масштабованість	Обмежена, неефективна для нестабільних обсягів даних	Легко масштабується для великих наборів даних

Автоматизація процесу розпізнавання об’єктів допомагає підвищити ефективність виконання задачі, роблячи її більш структурованою та оптимізованою. Завдяки об'єктивності алгоритмів, ми отримуємо однакову точність, також стоє зауважити що якість розпізнання залежить від степені виснаженості оператора, тоді як ШІ позбавлений цього фактору. Крім того, автоматизовані системи можуть легко інтегруватися з іншими системами.

1.5 Можливості, способи та перспективи розвитку системи у майбутньому

Стрімкий розвиток штучного інтелекту (ШІ) та популяризація генеративних моделей відкриває широкі можливості для вдосконалення систем розпізнавання об’єктів. Сучасні досягнення в оптимізації алгоритмів та апаратного забезпечення дозволяють створювати ефективніші та швидші моделі, які здатні працювати на різних платформах і в різних середовищах.

Можливості розвитку:

- Удосконалення конволюційних нейронних мереж (CNN) та трансформерів для роботи з візуальними даними:

- Використання генеративних моделей, таких як DALL-E чи Stable Diffusion, для синтезу тренувальних даних і розширення можливостей моделі[8];
- Впровадження алгоритмів квантового машинного навчання для підвищення швидкості обробки;
- Оптимізація алгоритмів розпізнавання.
- Покращення продуктивності:
- Оптимізація рушіїв (наприклад, TensorRT чи ONNX Runtime) для швидкого розпізнавання об’єктів у реальному часі. Розвиток конволюційних нейронних мереж (CNN) і трансформерів сприяє покращенню продуктивності систем розпізнавання[9];
- Використання прецизійних обчислень (FP16, INT8) для зменшення вимог до апаратного забезпечення без втрати точності.
- Розширення наборів даних:
- Створення більш комплексних та різноманітних наборів даних із включенням рідкісних або нестандартних об’єктів;
- Використання анотаційних інструментів з автоматичним маркуванням для прискорення розробки нових моделей.

Також є ряд покращень які можуть бути впроваджені завдяки інтеграції нових технологій:

- Розвиток апаратного забезпечення:
  - ARM-процесори: Розробка оптимізованих моделей для мобільних пристроїв, що працюють на енергоефективних ARM-чіпах. Це дозволить портативним пристроям виконувати завдання розпізнавання без втрати точності;
  - GPU і TPU: Використання графічних процесорів та тензорних обчислювачів для досягнення високої продуктивності при обробці великих наборів зображень.
- Використання мобільних платформ:

  - Портативність систем розпізнавання дозволить застосовувати їх у смартфонах, планшетах та інших мобільних пристроях, забезпечуючи доступність технології для більш широкої аудиторії;
  - Розробка офлайн-моделей, які можуть працювати без підключення до інтернету, що важливо для систем безпеки чи польових застосунків.
- Інтеграція з іншими технологіями:
  - Впровадження розпізнавання об’єктів у доповнену реальність (AR) для інтерактивного навчання чи професійних завдань;
  - Застосування технологій інтернету речей (IoT) для збирання й аналізу даних із розумних камер та сенсорів.

1.6 Аналіз існуючих систем розпізнавання об’єктів

У сучасному світі, де обробка та аналіз великих обсягів даних набувають вирішального значення, вирішення задач із патернізацією стає невід'ємною частиною розвитку передових алгоритмів і технологій. Здатність розпізнавати та ідентифікувати необхідні дані в складних структурах є основою для створення інтелектуальних систем, які можуть автоматизувати процеси, забезпечувати точність і швидкість обробки інформації. У цьому контексті система, що дозволяє ефективно виконувати розпізнавання об'єктів та аналіз даних, стає не просто важливим інструментом, а необхідністю.

Такі системи сприяють не лише оптимізації процесів, але й розширенню можливостей для досліджень і розробок у багатьох галузях. Наприклад, у машинному навчанні та штучному інтелекті вони дозволяють швидко та точно створювати навчальні набори даних, ідентифікуючи об'єкти, які є критично важливими для побудови моделей. Крім того, автоматизація розпізнавання

даних значно знижує витрати часу та людських ресурсів, одночасно підвищуючи якість та надійність отриманих результатів.

Система, що забезпечує такі можливості, стає ключовим елементом у багатьох галузях, включаючи медицину, транспорт, безпеку, маркетинг, електронну комерцію та багато інших. Її впровадження дозволяє створювати передові рішення, які сприяють подальшому розвитку технологій і підвищують конкурентоспроможність у динамічно змінюваному технологічному світі.

Мета цього аналізу – докладне вивчення функціональності, та ефективності двох вже відомих додатків: Amazon Rekognition, Microsoft Azure Computer Vision, та власного Detector. Кожен з цих інструментів володіє своїми особливостями та перевагами, спрямованими на задоволення різних потреб користувачів.

Далі розглядатимемо кожен з цих додатків для того, щоб з'ясувати, як вони відрізняються та які можливості вони надають для вирішення поставленої задачі.

Amazon Rekognition - це хмарний сервіс від Amazon Web Services (AWS), який надає можливості аналізу зображень та відео, включно з розпізнаванням об'єктів, облич, тексту, сцен, активностей та виявленням небажаного контенту. Сервіс також підтримує розпізнавання знаменитостей, аналіз емоцій та персональних захисних засобів.

Інтеграція Amazon Rekognition з різними мовами програмування через AWS Software Development Kits (SDKs), що спрощує розробку застосунків. AWS надає SDK для популярних мов, таких як Java, Python (Boto3), JavaScript, .NET, PHP та інші. Кожен SDK містить API, приклади коду та документацію

Налаштування AWS CLI та SDK. Для взаємодії з Amazon Rekognition через командний рядок або програмно необхідно налаштувати AWS Command Line Interface (CLI) та відповідні SDK. Це включає встановлення AWS CLI, створення облікових даних доступу та налаштування регіону AWS.

Azure Computer Vision - це хмарний сервіс, який надає API для аналізу зображень та відео, включно з розпізнаванням об'єктів, облич, тексту та сцен. Він дозволяє розробникам інтегрувати можливості комп'ютерного зору у свої застосунки без необхідності створення та навчання власних моделей машинного навчання.

Функціональність:

- Розпізнавання об'єктів: Виявлення та ідентифікація різних об'єктів на зображеннях.
- Аналіз облич: Визначення наявності облич, їхніх атрибутів (вік, стать, емоції) та ідентифікація осіб.
- Оптичне розпізнавання символів (OCR): Витяг тексту з зображень та документів.
- Опис зображень: Генерація текстових описів на основі вмісту зображення.

Інтеграція з Azure AI Vision здійснюється через офіційні SDK, доступні для різних мов програмування, таких як C#, Python, Java та JavaScript. Це спрощує розробку та впровадження можливостей комп'ютерного зору у застосунки.

Google Cloud Vision API - це сервіс, що надає розробникам можливість інтегрувати потужні моделі комп'ютерного зору у свої застосунки через REST та RPC API. Він дозволяє автоматизувати обробку зображень, отримувати цінну інформацію та покращувати бізнес-процеси.

Основні можливості Google Cloud Vision API:

- Розпізнавання об'єктів та облич: Виявлення та класифікація об'єктів, сцен та облич на зображеннях;
- Оптичне розпізнавання символів (OCR): Витяг тексту з зображень, включаючи друкований та рукописний текст, а також обробка багатосторінкових документів у форматах PDF та TIFF;

- Визначення властивостей зображення: Аналіз домінуючих кольорів, генерація підказок для кадрування та оцінка аспектного співвідношення;
- Модерація контенту: Виявлення потенційно небажаного або неприйнятного контенту, включаючи дорослий, насильницький чи медичний характер;
- Пошук подібних зображень: Використання можливостей Google Image Search для пошуку схожих зображень в інтернеті та визначення тематичних сутностей, таких як новини, події чи відомі особистості.

Clarifai - це провідна платформа штучного інтелекту, яка надає інструменти для створення, розгортання та управління моделями комп'ютерного зору та обробки природної мови. Заснована у 2013 році, компанія швидко стала лідером у сфері комп'ютерного зору, пропонуючи рішення для аналізу зображень, відео, тексту та аудіо.

Основні можливості Clarifai:

- Розпізнавання зображень та відео: Clarifai використовує глибоке навчання для ідентифікації об'єктів, людей, сцен та інших елементів на зображеннях та відео. Платформа пропонує попередньо навчені моделі, здатні розпізнавати тисячі концептів, а також можливість створення власних моделей для специфічних бізнес-завдань;
- Оптичне розпізнавання символів (OCR): Перетворення зображень з друкованим або рукописним текстом у машинозчитуваний формат, що дозволяє автоматизувати обробку документів та покращити пошук інформації;
- Модерація контенту: Виявлення небажаного або неприйнятного контенту, включаючи насильницькі, дорослі чи інші неприйнятні матеріали, для забезпечення безпеки та відповідності стандартам;

- Пошук подібних зображень: Використання векторних представлень для пошуку схожих зображень у великих наборах даних, що спрощує організацію та управління візуальним контентом;
- Генеративний ШІ: Платформа підтримує роботу з великими мовними моделями (LLM) та генеративним ШІ, що дозволяє створювати новий контент на основі існуючих даних та покращувати взаємодію з користувачами.

Clarifai пропонує повноцінну екосистему для розробки та впровадження рішень на основі штучного інтелекту, включаючи:

- AI Lake™: Централізоване сховище для організації, спільного використання та повторного застосування моделей ШІ, анотацій, наборів даних, робочих процесів та інтерфейсів користувача, що сприяє кращій співпраці команд;
- Mesh Workflows: Інструмент для створення складних робочих процесів, що об'єднують кілька моделей та логічних операторів, дозволяючи автоматизувати складні завдання та інтегрувати ШІ в бізнес-процеси;
- Scribe Label: Інструмент для автоматизованого маркування даних, що прискорює процес створення навчальних наборів даних та підвищує точність моделей.

Розгортання та інтеграція. Clarifai підтримує гнучкі варіанти розгортання, включаючи хмарні сервіси, локальні сервери та периферійні пристрої, що дозволяє інтегрувати рішення в існуючу IT-інфраструктуру підприємства.

Клієнти та визнання. Clarifai обслуговує клієнтів у комерційному та державному секторах, допомагаючи їм впроваджувати рішення на основі штучного інтелекту для покращення бізнес-процесів та підвищення ефективності. Компанія отримала визнання від таких організацій, як Forrester та

IDC MarketScape, за свої досягнення у сфері комп'ютерного зору та платформ штучного інтелекту.

Висновки

На основі нашого аналізу було з’ясовано, що можливість інтеграції з іншими системами, швидкість виконання задачі та масштабованість, яка може бути налаштована, є важливими складовими успішної реалізації системи розпізнавання об’єктів дорожнього руху. На основі проведених досліджень, до системи, що розробляється були сформовані такі вимоги:

- Проаналізувати сучасні системи розпізнавання об’єктів дорожнього руху та визначити існуючі реалізації подібних системи;
- Сформувати вимоги та реалізувати архітектуру системи розпізнавання об’єктів дорожнього руху, що розробляється;
- Сформулювати принципи функціонування системи та реалізувати діаграми функціонування системи та діаграми прецедентів;
- Визначити технології для розробки програмного забезпечення системи та обгрунтувати їх доцільність
- Провести всебічне тестування розробленого програмного забезпечення та довести відповідність системи вихідним вимогам.

2 ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Постановка задачі та визначення основних функцій системи

Визначення завдань є критичним етапом у розробці будь-якого проекту. Цей процес встановлює ключові цілі, вимоги та обмеження проекту, створюючи основу для подальших кроків. Правильна постановка завдань сприяє уточненню вимог та очікувань, а також визначає основні критерії успіху для майбутньої розробки та оцінки її результатів. Цей процес дозволяє команді проекту зосередитися на важливих аспектах, забезпечуючи узгодженість між всіма учасниками та ефективне використання ресурсів для досягнення поставлених цілей[10].

В сучасних умовах ринку, коли компанії конкурують між собою, автоматизація примітивних процесів є важливою складовою життя компанії. Такі впровадження можуть звільнити ресурси для розробки чи покращення більш складних процесів, та збільшення швидкості та точності залежних від автоматизованих процесів механізмів. Відсутність автоматизованих інструментів ускладнює роботу команд розробки, робить управління ресурсами неефективним та слабким.

Однак проблема не обмежується лише необхідністю розпізнавання об’єктів дорожнього руху. Система розпізнавання, що спроектована тільки для виконання одного типу задач, не є життєздатною в довгостроковій перспективі. Вона потребує можливості розширення функціоналу, щоб при виникненні нових типів задач мати можливість швидкого та безпроблемного додавання функціоналу.

Метою даної роботи є розробка системи розпізнавання об'єктів дорожнього руху, яка забезпечує автоматизацію процесу розпізнавання даних дорожнього руху на графічних зображеннях. Проте для забезпечення стабільної та повноцінної роботи цієї системи, необхідно впровадити більш

загальну інфраструктуру, яка підтримуватиме розширення функціоналу на усіх етапах своєї роботи. Таким чином, система за потреби може стати універсальною платформою для розпізнання з можливістю подальшої обробки розпізнаних даних.

Щоб забезпечити повноцінну роботу системи для розпізнання об’єктів, серверна частина має виконувати низку важливих функцій:

- Автоматизація розпізнавання:
  - Розпізнавання проводиться автоматично, розпізнані дані повинні повертатися залежно від конфігурації запуску процесу;
  - Збалансованість навантаження. Система не повинна залучати додаткові ресурси при малій степені навантаження.
- Управління навантаженням:
  - Система повинна мати метрики навантаження для відслідковування та оптимізації масштабування;
  - Метрики допоможуть у виявленні погано оптимізованих моментів роботи системи для кращої економії ресурсів у майбутньому, та відслідковуванню змін продуктивності після нововведень.
- Обмін інформацією:
  - Система повинна мати функціонал швидкого впровадження нових типів обробки та отримання даних.

Хоча основним завданням є автоматизація процесу розпізнавання об’єктів, система не може існувати лише для одного проєкту. Щоб система була життєздатною, вона повинна мати легку імплементацію для впровадження у інші проєкти із схожим типом задач[11].

Це означає, що вона має бути спроектована з урахуванням наступних розширень, таких як:

- Легка конфігурація:
  - Вибір та налаштування моделі повинні задаватися простими параметрами для легкого використання;

  - Вибір типу вихідних даних має бути гнучким та відповідати потребам користувача.
- Управління моделями ШІ:
  - Кожна модель повинна мати легку ініціалізацію для швидкого впровадження у систему.
- Обмін інформацією:
  - Система повинна мати функціонал швидкого впровадження нових типів обробки та отримання даних.

Таким чином, кінцевий продукт не лише полегшить процес розпізнавання об’єктів, але й стане гнучкою системою для автоматизації вирішення задач із розпізнання. Це дозволить автоматизувати задачі схожого типу у майбутньому.

Визначимо основні функції, які буде виконувати розроблена система:

- Завантаження набору графічних даних для розпізнавання;
- Вибір моделі та її конфігурації;
- Вибір типу отримання даних;
- Можливість проведення порівняння із оригіналом.

В процесі виконання вказаних завдань маємо намір розробити інноваційний проект, який допоможе автоматизувати розпізнання об’єктів та оптимізувати збір даних. Він сприятиме зручності та задоволенню користувачів, дозволяючи їм вирішувати задачі даного типу з високою точністю та швидкістю та підлаштовувати систему під свої потреби.

Визначившись із постановкою задачі, складемо UML-діаграму варіантів використання (UML Use Case діаграма).

На рис. 2.1, зображена UML Use Case діаграма взаємодії користувача з системою розпізнавання онлайн:

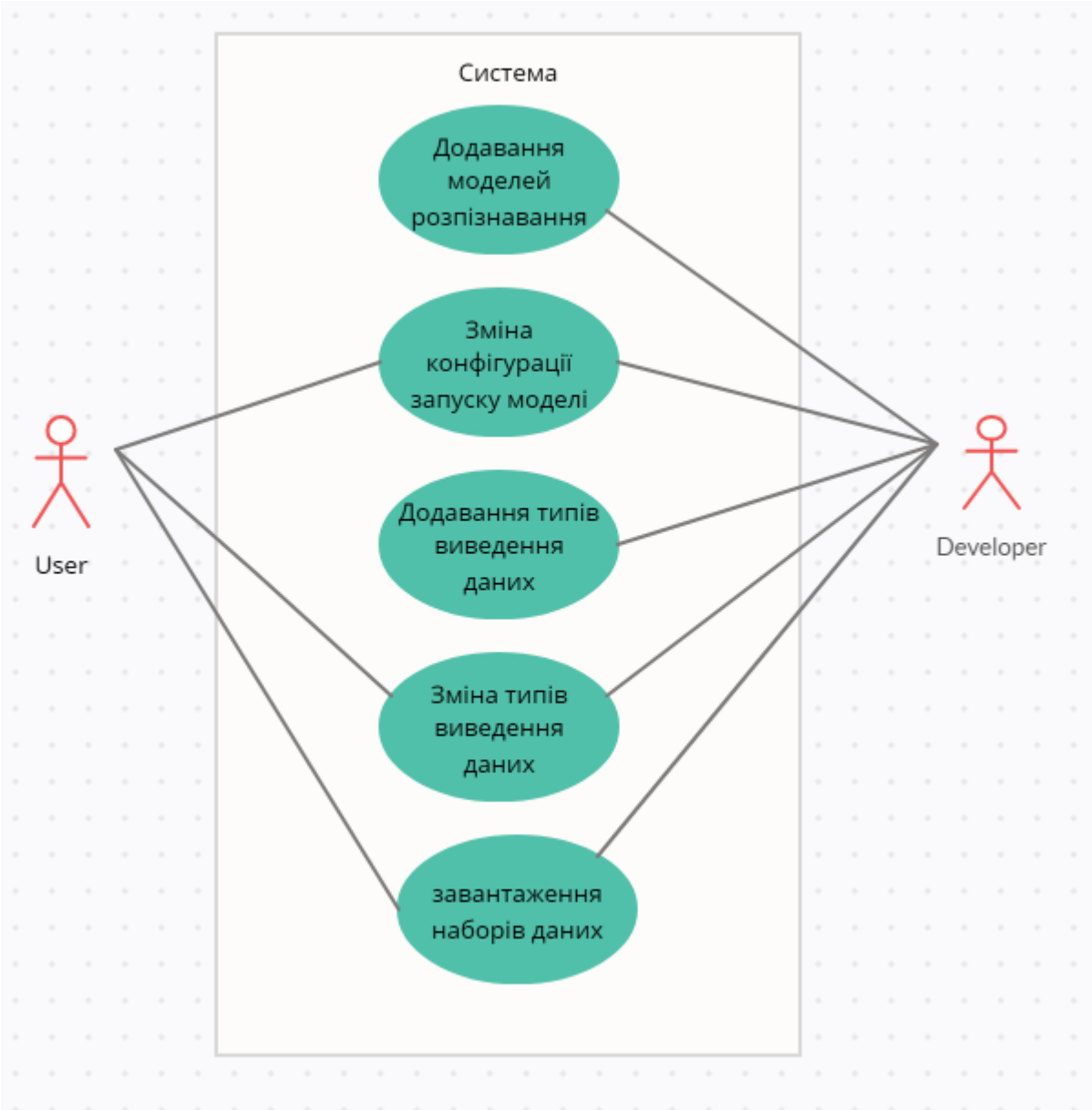


Рисунок 2.1 – Use Case діаграма взаємодії користувача з системою

Дана діаграма показує функціональні можливості автоматизованої системи для розпізнавання об’єктів дорожнього руху, в якій беруть участь три основних актора: вчитель, учень та адміністратор. Кожен актор має доступ до своїх прецедентів і як пов’язані між собою функції через include і extend, забезпечуючи послідовність та логіку в системі.

2.2 Проектування архітектури системи

Система[12] для автоматизації процесу розпізнавання об’єктів дорожнього руху повинна мати гнучку та масштабовану архітектуру, що забезпечить стабільну роботу та можливість розширення функціоналу. Система повинна обробляти великий обсяг даних та забезпечувати широкий спектр масштабування.

Для реалізації функціоналу системи обрана трирівнева архітектура, яка розділяє систему на окремі компоненти, кожен з яких виконує свою функцію.

Основними компонентами є:

- Клієнтський шар (Frontend Layer);
- Шар бізнес-логіки (Backend Layer);
- Шар запуску моделей ШІ (Detector Layer).

архітектурну схема системи можна знайти у Додатку В2

Клієнтський шар.

Цей шар взаємодіє безпосередньо з користувачами системи через графічний інтерфейс, надаючи їм можливість виконувати певні дії. Клієнт надсилає запити до сервера через інтерфейс API і отримує відповідні дані. Клієнтська частина може бути реалізована у вигляді веб додатку.

Основні функції:

- Відображення даних користувачеві;
- Надсилання запитів до серверу;
- Обробка отриманих відповідей та оновлення інтерфейсу відповідно до результатів дій.

Шар бізнес-логіки.

Цей шар відповідає за ініціалізацію процесу розпізнавання об’єктів. Він обробляє запити, отримані від клієнтського шару, виконує необхідну логіку та взаємодіє з шаром розпізнавання даних, передаючи йому інформацію про потрібні налаштування моделі, набір даних та тип даних що мають бути повернені користувачеві.

Основні функції:

- Формування конфігурації запуску розпізнавання;
- Вичитка та підготовка графічних зображень для розпізнавання;
- Обробка вихідних даних перед їх передачею користувачу.

Шар запуску моделі ШІ.

Цей шар забезпечує ініціалізацію моделі, та запуск процесу розпізнавання керуючись конфігурацією, отриманою від шару бізнес-логіки

Основні функції:

- Ініціалізація моделей;
- Проведення запуску процесу розпізнавання;
- Структуризація даних, що були отримані з процесу;

Взаємодія між компонентами.

Взаємодія між компонентами архітектури відбувається за допомогою чітко визначених інтерфейсів і протоколів. Основний потік взаємодії включає наступні етапи:

- Запит від клієнта. Користувач через інтерфейс (веб-додаток) надсилає запит на проведення розпізнавання до сервера. Запит містить назву та конфігурацію моделі
- Обробка запиту на рівні бізнес-логіки. Сервер обробляє отриманий запит, створює конфігурацію запуску та передає її на сервіс ініціалізації розпізнавання
- Отримання даних Шар запуску моделі ШІ. Виконується з допомогою сервісу, який починає етап розпізнавання та подальшої обробки даних залежно від отриманої з шару бізнес-логіки інформації
- Формування відповіді. Після виконання всіх дій та отримання структурованих даних, сервер формує відповідь і відправляє її клієнту;
- Оновлення інтерфейсу. Клієнт отримує відповідь від сервера і оновлює користувацький інтерфейс на основі результатів.

2.3 Аналіз та обґрунтування технологій розробки

Розробка клієнт-серверної системи для автоматизації процесу розпізнавання об’єктів дорожнього руху вимагає використання сучасних технологій, які забезпечують надійність, масштабованість та ефективність функціонування системи. Для кожного компонента архітектури необхідно вибрати відповідні технологічні засоби, що відповідають вимогам проекту[13]. Нижче наводиться аналіз та обґрунтування вибору ключових технологій для розробки серверної частини системи.

Мова програмування[14] є визначальним фактором у створенні надійної серверної архітектури веб-додатків. Правильний вибір мови дозволяє підвищити продуктивність, забезпечити зручність підтримки та оптимізувати розвиток проекту в довгостроковій перспективі. Для цього проекту, враховуючи актуальні тенденції та вимоги до сучасних веб-рішень, найбільш підходящими є JavaScript/Reactjs (через платформу Node.js) і Kotlin. Кожна з цих мов має потужні інструменти, великий набір бібліотек і унікальні переваги, які підвищують ефективність розробки та підтримки коду.

Node.js є однією з найбільш затребуваних платформ для розробки серверної частини веб-додатків і мікросервісів. [15] Платформа базується на рушієві V8 від Google, що забезпечує високу швидкодію JavaScript-коду. Це дозволяє працювати з асинхронними запитами, які обробляються в однопоточному режимі з використанням подій, що значно підвищує продуктивність і масштабованість систем. Основні переваги Node.js включають:

- Швидкодія. Архітектура Node.js дозволяє обробляти одночасно десятки тисяч запитів без значних затримок. Асинхронна модель обробки запитів зменшує навантаження на сервер, оптимізує роботу з мережевими ресурсами та забезпечує швидкий час відгуку;

- Універсальність. JavaScript підтримується як на клієнтській, так і на серверній стороні. Це забезпечує повну інтеграцію логіки додатка, скорочуючи розрив між фронтендом і бекендом. Така єдність коду дозволяє зменшити кількість контекстних перемикань між мовами та забезпечує більш послідовну підтримку проєкту;
- Модульність та розширюваність. Одним із сильних аспектів Node.js є екосистема npm, що містить мільйони готових до використання модулів та бібліотек. Це значно скорочує час на розробку, оскільки більшість необхідної функціональності вже реалізована. Модульний підхід дозволяє легко підключати і видаляти різні компоненти в залежності від потреб проекту.

ReactJS так переробити все це бібліотека JavaScript, яка забезпечує ефективну розробку інтерактивних та компонентно-орієнтованих інтерфейсів. Використання ReactJS робить код більш структурованим, що є надзвичайно важливим для великих і складних проєктів із численними компонентами та залежностями. ReactJS сприяє кращій читабельності коду, забезпечує повторне використання компонентів та зменшує кількість помилок завдяки передбачуваності роботи його архітектури. React забезпечує компонентний підхід до розробки веб-додатків, що робить його ідеальним вибором для масштабованих проектів[16].

Kotlin, відомий своїм лаконічним і безпечним синтаксисом, є потужним інструментом для розробки серверної логіки, особливо у випадках, коли проєкт вимагає високої продуктивності, масштабованості та зручності роботи з Java-екосистемою. Ця мова активно використовується для створення бекенд-сервісів, мобільних додатків (Android) та мультиплатформних проєктів.

Основні переваги Kotlin:

- Лаконічність і читабельність:
  - Kotlin дозволяє писати чистий і зрозумілий код завдяки скороченню шаблонних конструкцій, що прискорює розробку та полегшує підтримку.

  - Зменшення кількості коду знижує ймовірність помилок, робить код зрозумілим для рев'ю і спрощує процес навчання нових членів команди.
- Безпечність:
  - Kotlin має вбудовану систему для запобігання помилкам, пов’язаним із null-значеннями (null safety). Це знижує ризик виникнення винятків типу `NullPointerException`, що забезпечує стабільність серверної логіки.
- Широка екосистема:
  - Kotlin інтегрується з Java, дозволяючи використовувати багатий набір бібліотек та фреймворків Java-екосистеми, таких як Spring Boot для серверної розробки або Hibernate для роботи з базами даних.
  - Існують також нативні бібліотеки, такі як Ktor, які спрощують створення REST API і обробку HTTP-запитів.
- Масштабованість та продуктивність:
  - Kotlin відмінно підходить для багатопоточних додатків завдяки підтримці корутин (coroutines), що забезпечує ефективну асинхронну роботу та масштабованість серверних додатків.
- Мультиплатформність:
  - Kotlin дозволяє створювати код, який працює на різних платформах (Android, iOS, веб, сервери) без необхідності дублювання. Це спрощує розробку уніфікованих додатків.
- Активна підтримка та розвиток:
  - Мова має підтримку від JetBrains, компанії, відомої своїми інструментами для розробників (IntelliJ IDEA). Активна спільнота розробників сприяє швидкому вдосконаленню мови, надає доступ до великої кількості документації та ресурсів.

Вибір фреймворку для серверної частини веб-додатка є ключовим етапом, оскільки від нього залежать масштабованість, продуктивність і зручність інтеграції з API. При проєктуванні ефективної бекенд-архітектури важливо враховувати специфічні потреби проєкту, такі як швидка обробка запитів, забезпечення безпеки, підтримка складних баз даних і можливість легкого впровадження нових функцій. З огляду на ці вимоги, оптимальним вибором для даного проєкту стане фреймворк Spring.

Spring — це потужний фреймворк для Java, який забезпечує розробку складних, масштабованих і безпечних веб-додатків. Завдяки модульній архітектурі Spring дозволяє легко налаштовувати додатки відповідно до потреб проєкту, а також інтегрувати сторонні сервіси та бібліотеки. Основна перевага Spring полягає в його гнучкості та підтримці сучасних стандартів розробки.

Основні переваги Spring:

- Швидка розробка:
  - Spring надає багатий набір інструментів для швидкого створення веб-додатків, таких як Spring Boot, який спрощує конфігурацію і запуск проєкту;
  - Підтримка компонентів для автентифікації, управління сесіями, інтеграції з базами даних через Spring Data JPA, а також побудови REST API через Spring Web;
  - Автоматизація багатьох аспектів розробки дозволяє зосередитися на бізнес-логіці, а не на технічних деталях.
- Висока продуктивність:
  - Spring оптимізований для роботи в середовищах з високим навантаженням, забезпечуючи швидкий відгук завдяки вбудованим механізмам кешування та ефективній роботі з потоками;
  - Spring Reactive дозволяє будувати реактивні додатки, які ефективно обробляють великі обсяги даних у реальному часі;
  - Підтримка горизонтального масштабування дозволяє адаптувати додаток до зростаючої кількості користувачів.

- Модульність та інтеграція:
  - Фреймворк має модульну структуру, що дозволяє використовувати лише необхідні компоненти, такі як Spring MVC, Spring Data, або Spring Cloud;
  - Легка інтеграція з популярними базами даних (PostgreSQL, MySQL) та зовнішніми сервісами (Kafka, RabbitMQ, Elasticsearch);
  - Завдяки підтримці мікросервісної архітектури Spring є ідеальним вибором для побудови складних розподілених систем.
- Активна спільнота та документація:
  - Spring підтримується однією з найбільших спільнот розробників Java, що забезпечує доступ до безлічі ресурсів, прикладів та розв’язків для типових задач;
  - Велика кількість детальної документації, офіційних посібників та навчальних матеріалів спрощує вивчення фреймворка і його функціоналу;
  - Регулярні оновлення гарантують актуальність і підтримку сучасних стандартів веб-розробки.

Spring підходить для проєктів, які потребують:

- Масштабованості: Легка інтеграція мікросервісів дозволяє адаптувати систему до високих навантажень;
- Безпеки: Вбудовані інструменти забезпечують захист даних і користувачів;
- Гнучкості: Модульна структура дозволяє використовувати лише необхідні компоненти;
- Сучасних стандартів: Підтримка реактивного програмування та хмарних технологій через Spring Cloud.

Spring — це ідеальний вибір для створення складних, багатофункціональних додатків, які потребують високої продуктивності, надійності та інтеграції з сучасними технологіями.

Вибір інструмента для запуску моделей штучного інтелекту є важливим етапом, оскільки від цього залежить продуктивність, гнучкість інтеграції з іншими системами та можливість масштабування. При розробці інфраструктури для роботи з моделями ШІ необхідно враховувати специфічні потреби проєкту, такі як підтримка різних форматів моделей, оптимізація обчислень, інтеграція з сучасними апаратними засобами (GPU/TPU) і можливість легкого розгортання нових алгоритмів. Виходячи з цих вимог, оптимальним вибором для даного проєкту є фреймворк DJL (Deep Java Library).

Deep Java Library це сучасна бібліотека для роботи з моделями машинного навчання на мові програмування Java. DJL забезпечує простий і гнучкий інтерфейс для інтеграції моделей ШІ у Java-додатки, дозволяючи розробникам легко виконувати завдання, пов’язані з навчанням та інференсом моделей. Завдяки підтримці різних рушіїв глибокого навчання (MXNet, PyTorch, TensorFlow тощо), DJL підходить для багатьох сценаріїв, включаючи розпізнавання зображень, обробку текстів і прогнозування.

Основні переваги DJL:

- Підтримка мультифреймворків:
  - DJL інтегрується з кількома популярними фреймворками машинного навчання, такими як PyTorch, TensorFlow, MXNet, ONNX Runtime та інші. Це забезпечує гнучкість і можливість вибору найкращого інструменту для ваших потреб.
- Простота використання:
  - Завдяки інтуїтивному API, DJL дозволяє розробникам швидко почати роботу без необхідності вивчати деталі внутрішньої реалізації моделей.
- Мультиплатформеність:
  - Залежно від рушія DJL підтримує різні апаратні платформи, включаючи CPU, GPU та TPU, дозволяючи масштабувати додаток залежно від апаратних можливостей.

- Автоматичне завантаження моделей:
  - Бібліотека має вбудований менеджер для автоматичного завантаження попередньо навчених моделей із відкритих репозиторіїв.
- Масштабованість:
  - DJL дозволяє працювати як з невеликими локальними проєктами, так і з масштабованими серверними додатками завдяки простій інтеграції з хмарними сервісами та контейнерами.
- Активна підтримка:
  - Велика спільнота, регулярні оновлення та добре структурована документація роблять DJL зручним інструментом для сучасної розробки.

DJL підходить для:

- Розгортання моделей машинного навчання у Java-додатках;
- Реалізації інтерфейсу у реальному часі, наприклад, для розпізнавання зображень чи текстів;
- Інтеграції ШІ у хмарні рішення, такі як AWS або Google Cloud;
- Виконання кросплатформених обчислень на CPU та GPU.

Deep Java Library — це потужний інструмент, що поєднує простоту використання, підтримку сучасних стандартів і продуктивність, роблячи його ідеальним вибором для впровадження штучного інтелекту в Java-додатки.

Для забезпечення ефективної та стабільної взаємодії між клієнтською і серверною частинами веб-додатка було обрано протокол HTTP(S). Цей стандартний протокол веб-розробки забезпечує безпечну передачу даних через мережу і є оптимальним вибором для сучасних додатків. Використання HTTP(S) дозволяє обмінюватися даними у текстовому форматі, що сумісно з більшістю клієнтських платформ. Це сприяє легкості інтеграції з різними пристроями та додатками, забезпечуючи їх стабільну роботу.

REST API (Representational State Transfer API)

Для побудови взаємодії між клієнтом і сервером застосовується архітектурний стиль REST API, який є одним із найпоширеніших підходів для створення веб-сервісів. REST API базується на стандартизованих HTTP-методах, таких як GET, POST, PUT і DELETE, що дозволяє легко управляти ресурсами на сервері. Цей підхід забезпечує простоту розробки, високу продуктивність та зручність інтеграції.

Переваги REST API:

- Простота використання та налаштування:
  - REST API надає уніфіковані URI-адреси для доступу до ресурсів, що спрощує процес інтеграції з клієнтськими додатками, такими як веб або мобільні платформи;
  - Його налаштування є швидким і зрозумілим, дозволяючи розробникам створювати структури запитів і відповідей, які легко інтегруються у клієнтський код.
- Масштабованість:
  - Завдяки стандартизованим методам REST API здатен обробляти велику кількість одночасних запитів, що робить його оптимальним для веб-додатків із великим навантаженням;
  - REST API підтримує додавання нових ресурсів і функцій без значного втручання в існуючу архітектуру, що дозволяє адаптувати систему до зростання вимог.
- Розділення клієнта та сервера:
  - REST API забезпечує незалежність серверної і клієнтської частин, дозволяючи обробляти запити сервером незалежно від типу клієнтського додатка (веб, мобільний чи інший сервер);
  - Такий підхід спрощує оновлення та підтримку системи, адже сервер і клієнт можна модернізувати окремо.
- Оптимізація трафіку:

  - REST API підтримує кешування, що зменшує навантаження на сервер і скорочує кількість запитів. Це особливо важливо для високонавантажених систем, адже кешування покращує швидкість доступу до інформації.
- Сумісність із сучасними технологіями:
  - REST API є універсальним рішенням, яке підтримується популярними фреймворками та платформами, такими як React, Angular, iOS і Android. Це гарантує легку інтеграцію з сучасними клієнтськими технологіями, мінімізуючи ризики несумісності.

2.4 Аналіз моделей розпізнавання об’єктів та їх навчання

Моделі розпізнавання об’єктів займають ключову роль[17] у задачах комп’ютерного зору. Вони здатні виявляти, класифікувати та локалізувати об’єкти на зображеннях або у відео, використовуючи передові алгоритми глибокого навчання. Їх ефективність визначається низкою факторів, таких як архітектура моделі, особливості навчання, використання даних та вимоги до продуктивності. Моделі розпізнавання об’єктів мають ряд основних функцій:

- Виявлення об’єктів (Object Detection) – Моделі здатні ідентифікувати об’єкти та визначати їхнє місцезнаходження у вигляді обмежувальних рамок (bounding boxes);
- Класифікація об’єктів (Object Classification) – Визначення типу об’єкта або його належності до певного класу (наприклад, "автомобіль", "людина", "тварина");
- Сегментація об’єктів (Object Segmentation) – Покращення Object Detection. Складніший та вимогливіший за попередника, має точну ідентифікацію пікселів для детальнішого окресленняоб’єктів;

- Аналіз сцен – Виявлення відносного розташування та взаємодії між об’єктами, що особливо важливо для складних завдань, таких як автономне водіння.

Архітектура моделі визначає ключові особливості роботи із нею, а розуміння особливостей архітектури моделей дає можливість вибрати найкращу для поставленої задачі. Сучасні моделі розпізнавання мають велику кількість архітектур, кожна з яких має свої особливості. Ключовими у цій сфері є такі архітектури[18]:

- YOLO (You Only Look Once):
  - Виконує розпізнавання об’єктів у реальному часі, використовуючи єдину нейронну мережу;
  - Підходить для застосувань, де потрібна висока швидкість, наприклад, відеоспостереження чи моніторинг.
- Faster R-CNN:
  - Зосереджена на точності, забезпечуючи високу якість розпізнавання, жертвуючи швидкодією;
  - Використовується у задачах, де пріоритетом є точність, наприклад, у медицині.
- SSD (Single Shot MultiBox Detector):
  - Поєднує швидкість і точність, підходить для застосування на мобільних пристроях;
  - Використовує багаторівневу архітектуру для аналізу об’єктів різних масштабів. Потребує великих наборів даних, високий ризик перенавчання
- Mask R-CNN:
  - Розширює Faster R-CNN, додаючи можливість піксельно-точного виділення об’єктів через сегментацію;
  - Застосовується для задач, де важливо не лише визначити об’єкт, але й отримати його контур.
- EfficientDet:

- Орієнтована на оптимізацію обчислювальних ресурсів;
- Підходить для застосувань, де потрібна висока продуктивність при обмежених ресурсах, коли точність не є ключовим фактором;
- Потребує великих наборів даних для навчання. Має низьку точність.

Беручи до уваги специфіку поставленої задачі, та обмеження, створимо таблицю порівняння, що базуватиметься на критеріях відповідності до наших запитів. Порівняння архітектур моделей розпізнавання об’єктів наведено на таблиці 2.1:

Таблиця 2.1 Порівняння архітектур моделей розпізнавання об’єктів

Критерій	Швидкість роботи	Навантаження	Точність	Кількість даних для навчання	Складність реалізації
YOLO	Висока	Середня	Висока	Середня	Середня
Faster R-CNN	Низька	Висока	Висока	Висока	Висока
SSD	Висока	Середнє	Висока	Висока	Висока
Mask R-CNN	Низька	Висока	Висока	Висока	Висока
EfficientDet	Висока	Низька	Низька	Висока	Середня

Беручи до уваги постановку нашої задачі та економічні показники найкращим варіантом для створення системи розпізнавання об'єктів дорожнього руху є використання моделі Object Detection на архітектурі від YOLO (You Only Look Once). Дана модель швидко виконувати задачі з ідентифікації на машинах із середнім типом заліза та матиме високу точність навіть із порівняно малими наборами даних для навчання.

Для виконання задачі з виявлення об’єктів важливим етапом є навчання моделі. Модель навчається на зображеннях, які попередньо пройшли процедуру labeling.

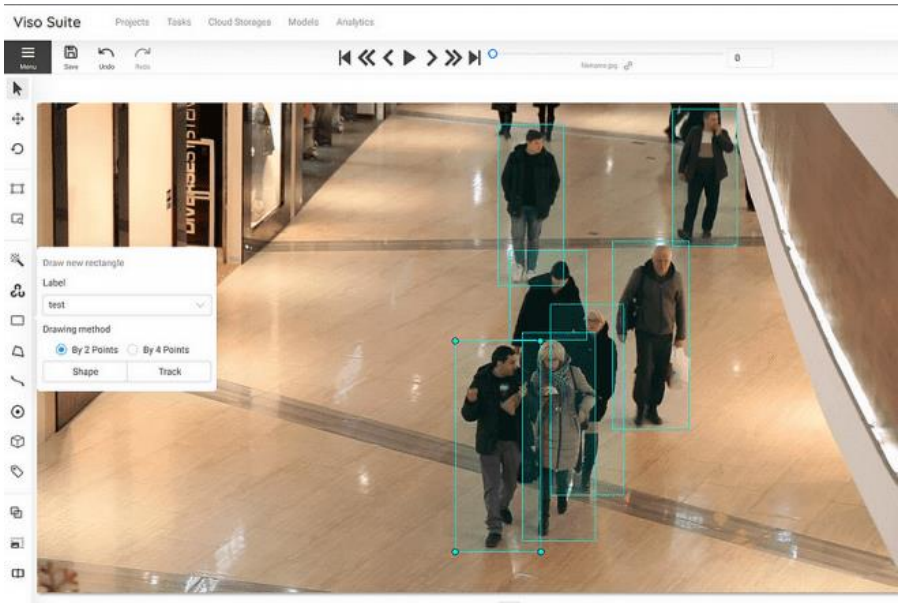


Рисунок 2.2 інтерфейс cvat

Labeling – ручне визначення об’єктів на зображеннях та їх запис у спеціальні конфігураційні файли, що різні для кожної архітектури. Дана процедура проводиться за допомогою спеціалізованих програм, що надають інструменти для підкреслення об’єктів та створення конфігураційних файлів та мають підтримку великої кількості архітектур.

Найпопулярнішими серед них є cvat – web додаток та labelImg – десктопна програма, що протребує Python, інтерфейси яких зображені на рисунках. 2.2 та 2.3 відповідно. Ці програми мають однаковий функціонал та допомагають у створенні навчального набору даних.

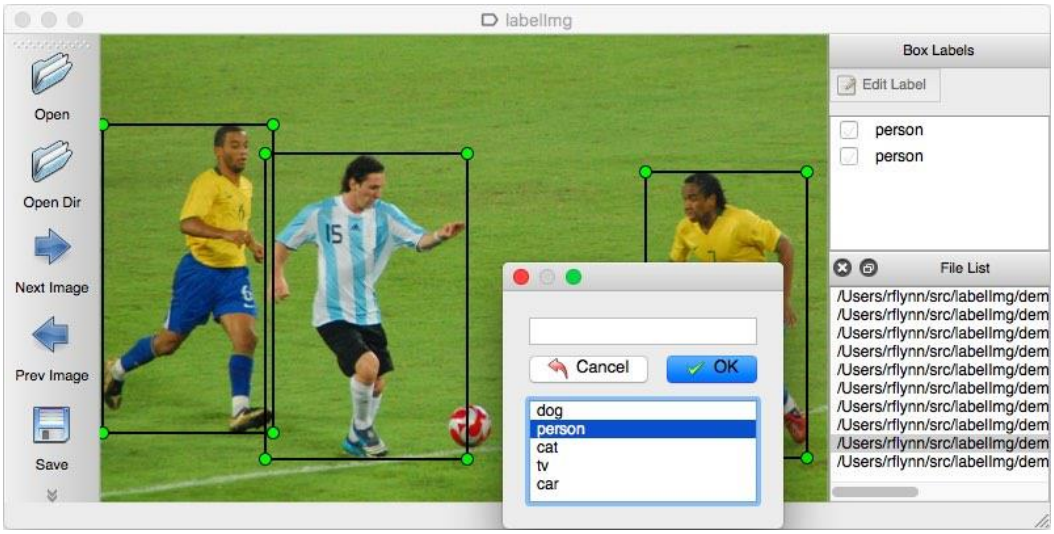


Рисунок 2.3 інтерфейс labelImg

Для покращення наборів даних та додаткового наповнення вдаються до хитрощів таких як[19]:

- Обернення даних на фото;
- Накладання додаткових об’єктів на вже використані фото;
- Додавання шуму;
- Підміна або стирання об’єктів, якщо вони звичай знаходяться у відношенні до інших об’єктів, на приклад машинні знаки;

Ці хитрощі допомагають створювати набори даних більшої кількості при обмежених ресурсах, а також допоможуть зменшити шанс перенавчання – моменту коли модель стає гірше розпізнавати об’єкти на інших фото. Створений набір даних розділяється на три частини у відповідності до етапів навчання моделі:

- Тренування – найважливіший етап під час якого відбувається навчання моделі розумінню закономірностей позначених класів;
- Валідація - етап використовується для оцінки продуктивності моделі під час її навчання. Часто при погано зробленому наборі на цьому етапі проявляється перенавчання;

– Тестування - використовуються для остаточної перевірки якості моделі. Дані на цьому етапі ніяк не впливають на подальше навчання чи налаштування моделі.

Зазвичай набір даних розподіляється у відношенні 70% для тренувальних, 15% для валідації та 15% для тестування.

Дані для етапу тренування повинні підбиратися особливо ретельно, різниця у кількості класів на зображеннях не повинна бути високою, також важливо додати трохи фото де відсутні деякі, а також всі класи.

Правильно створений набір даних є ключовим фактором правильної роботи системи з розпізнавання об’єктів дорожнього руху, тому важливо відповідально підійти до кожного із етапів.

3 РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

3.1 Моделювання архітектури системи

Моделювання архітектури — це процес розробки абстрактного уявлення програмної системи, її складових частин та взаємодії між ними. Воно дозволяє глибше зрозуміти, як система буде реалізована, визначити ключові компоненти, зв'язки між ними та переконатися, що архітектура відповідає вимогам користувачів і бізнесу.

Основна мета моделювання архітектури — створення чітке уявлення про всі аспекти системи перед процесом її реалізації, що сприяє спрощенню обговорення, аналізу та прийняття рішень. Таке моделювання також дозволяє виявити потенційні ризики, оптимізувати використання ресурсів і гарантувати масштабованість, продуктивність та надійність майбутньої системи.

Діаграма розгортання (UML Deployment diagram) — це тип діаграми, яка використовується для моделювання фізичного розгортання компонентів програмної системи на апаратному забезпеченні. Вона показує, як програмні компоненти взаємодіють з апаратними вузлами (сервери, клієнтські машини, бази даних тощо) у робочому середовищі.

Основні елементи діаграми розгортання:

- Вузли (Nodes):
  - Це апаратні чи програмні ресурси, які виконують певні завдання;
  - В UML вузол позначається у вигляді тривимірного куба;
  - Наприклад: сервери, пристрої користувачів, мережеве обладнання.
- Артефакти (Artifacts):
  - Це програмні компоненти, які розгортаються на вузлах, наприклад, виконувані файли, скрипти, бази даних;
  - В UML вони позначаються прямокутниками з написом типу артефакта.

- З'єднання (Communication Paths):
  - Лінії, які показують взаємодію між вузлами;
  - Вказують на те, як дані передаються між вузлами (наприклад, через мережу).
- Стереотипи:
  - Використовуються для уточнення ролей вузлів чи артефактів (наприклад, «сервер», «клієнт», «база даних»).

Діаграма розгортання зображена на рис. 3.1:

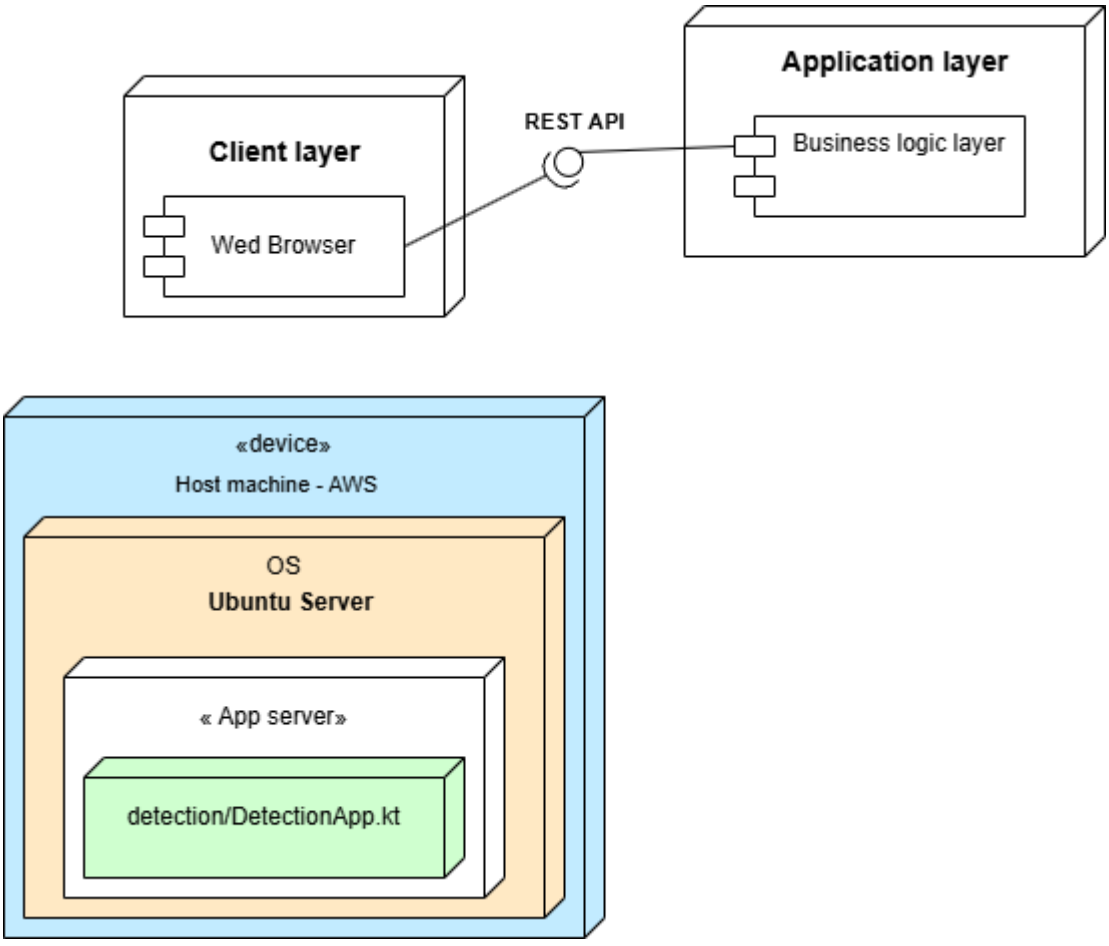


Рисунок 3.1 – UML Діаграма розгортання

На цій діаграмі зображена система, яка складається з кількох частин, що працюють разом:

- Клієнтський шар (Client Layer) - це комп'ютер, де користувач використовує веб-браузер для доступу до програми;
- Прикладний шар (Business Logic and Application Layer) - виконує бізнес-логіку (основні обчислення та правила програми). Зв'язується з шаром запуску моделі через сервіс ініціалізації
- Хост-машина в AWS - уся система розгорнута на сервері в хмарі (AWS - Amazon Web Services). Встановлена операційна система Ubuntu;
- Складові хост-машини – це App Server (сервер додатків): Використовує WSGI для запуску Kotlin-додатку

3.2 Розробка послідовності функціонування системи розпізнавання об’єктів дорожнього руху

Створення послідовності функціонування системи є важливим етапом розробки системи розпізнавання об’єктів дорожнього руху, який дозволяє:

- Краще зрозуміти функціонал та можливості системи;
- Знайти подальші можливості для вдосконалення та розширення;
- Зрозуміти недоліки системи.

Цей крок дозволяє нам отримати чітку картину системи та покращити ведення документації, що може бути корисним при розширенні команди розробки для подальшої її підтримки, що є критично важливим в умовах стрімкого розвитку систем штучного інтелекту та машинного навчання

Одним із методів графічного подання функціонування системи є Aplication flow діаграма.

Aplication flow діаграма - це графічне представлення логіки функціонування системи. Вона показує послідовність етапів системи і зв’язки між ними, які виконуються під час роботи програми. Така діаграма

використовуються для візуалізації процесів і спрощення розуміння функціонування системи.

Діаграма “Aplication flow” для системи розпізнавання об’єктів дорожнього руху зображена на Рисунку 3.2

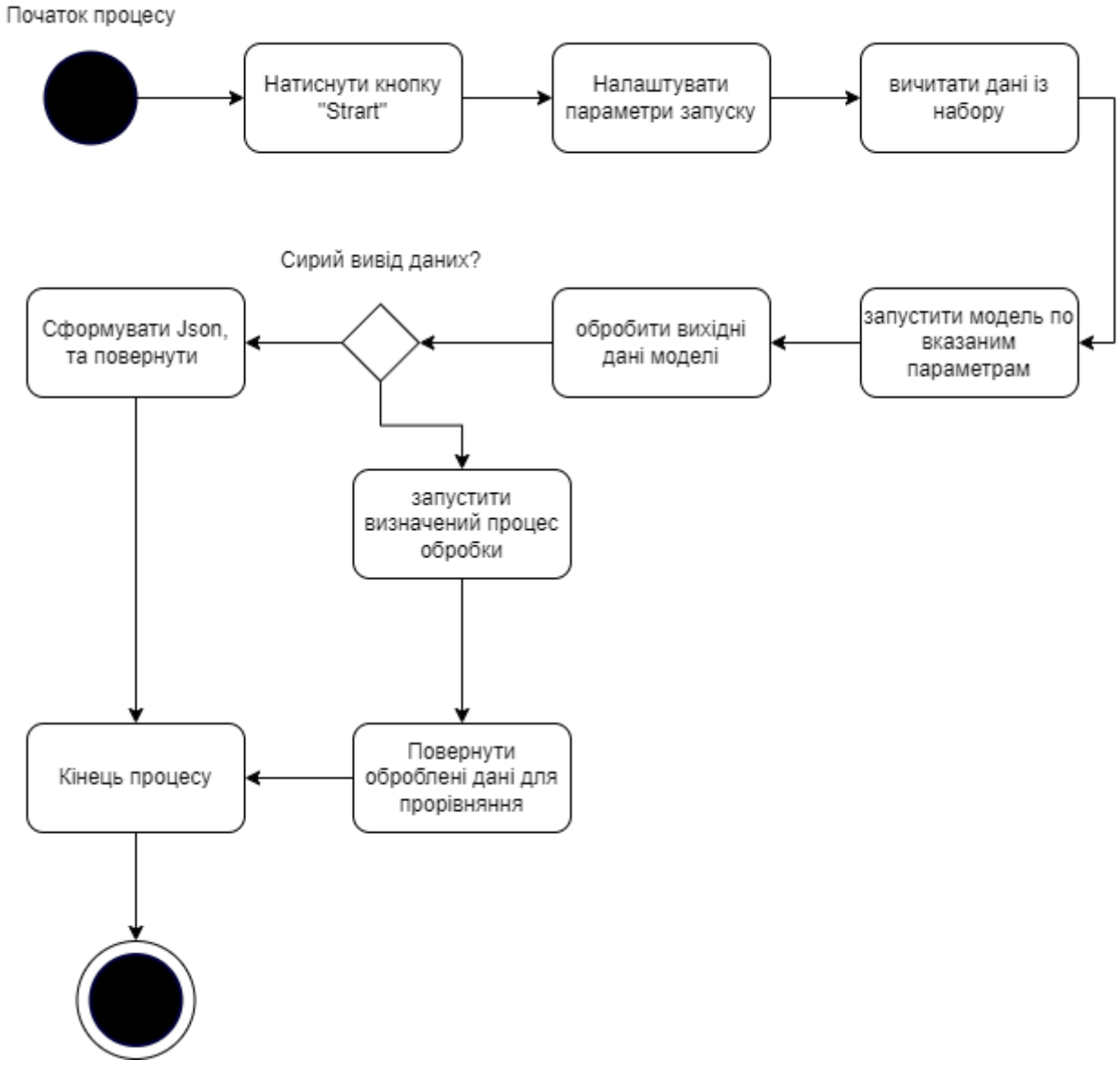


Рисунок 3.2 – Application flow діаграма

Aplication flow діаграма надає структуровану інформацію про функціонування системи розпізнавання об’єктів дорожнього руху та дозволяє отримати краще розуміння роботи користувача з системою

Після запуску системи користувач бачить вікно налаштування запуску процесу розпізнавання яке дає змогу вибрати параметри запуску:

- Вибір доступної моделі для розпізнавання;
- Вибір об’єктів, які модель може розпізнати;
- Вибір типів виведення обробленої інформації;
- Імпортування набору даних які повинні бути оброблені.

Вибір об’єктів для розпізнання залежить від вибраної моделі. Як набір даних система розпізнавання об’єктів дорожнього руху може прийняти або зображення у відповідному форматі, на приклад png, jpg або ZIP архів який містить у собі велику кількість цих зображень.

Після запуску системи розпізнавання об’єктів дорожнього руху проводиться вичитка даних із набору який був наданий користувачем та знаходиться на сервері, проводиться фільтрація даних по їх відповідності до типів розширень з якими працює система Вичитування даних із архіву проводиться паралельно на всіх доступних системі потоках, що прискорює процес, даючи змогу працювати із великими наборами даних. Вичитані дані зберігаються у тимчасовій директорії, шлях до якої система буде використовувати на подальших етапах функціонування

Після завершення етапу вичитки система починає запуск моделі по вказаним користувачем налаштуванням та починає підготовку зображень. Усі моделі зберігаються на сервері, та окрім наданих користувачем конфігураційних даних також мають додаткові, такі як запуск на найбільш продуктивному типові прискорювачів у послідовності

- GPU – найбільш продуктивний;
- TPU – Середньої продуктивності;
- CPU – найменш продуктивний.

Система проводить пошук та вибирає найкращий прискорювач який доступний їй на даний момент. Також одним із параметрів є показник фільтрації об’єктів залежно від впевненості моделі у достовірності визначеного об’єкта. Це числовий показник який показує мінімальну впевненість системи у знайденому об’єкті який вона може показувати користувачеві

Так як усі моделі object detection працюють із матричними показниками зображень, знаходячи закономірності та проводячи патернізацію, а обчислення, що проводяться із матрицями є дуже складними та використовують багато часу, більшість моделей будують свій підхід саме на використанні квадратних матриць, що дозволяє знизити кількість необхідних обчислень не погіршуючи, при цьому точність та якість роботи. Тому для виконання процесу розпізнавання усі фото підлягають редагуванню та зміні їх роздільної здатності відповідно до вимог моделі, зазвичай це 640x640, або 480x480.

По закінченню конфігурації модель завантажується та починає процес розпізнавання набору даних. Після успішного виконання процесу розпізнавання модель надає список класів які починають проходити процес первинної обробки на етапі обробки вихідних даних. Цей етап включає:

- Зворотнє масштабування координат об’єктів для відповідності до оригінальних зображень;
- Формування списку Entity об’єктів, що містять інформацію потрібну для кінцевої обробки:
  - Назва оригінального фото;
  - Список типів об’єктів;
  - Список координат у порядку зі списком об’єктів.

Зворотнє масштабування потрібне для правильного відображення BoundingBox-ів об’єктів на оригінальному зображенні. Альтернативним підходом є впровадження змін напряму у змінене фото, але такий підхід має великий мінус у вигляді втрати деталей, при чому чим більша роздільна здатність оригінального фото у порівнянні із обробленим, тим більше деталей

ризикують бути втрачені. Координати представляють собою масив із двох елементів які є верхньою лівою межею та нижньою правою межею BoundingBox-а розпізнаного об’єкта.

Наступний етап роботи програми залежить від вибраного користувачем типу output під час етапу вибору конфігурації та може загально називатися кінцевою обробкою даних.

Якщо користувач вибрав подання необроблених даних – система починає створення об’єктів що формуються із тих, що були створені на етапі первинної обробки та записує їх у json файл із назвою оригінального фото, куди записується інформація про координати об’єкта що був виявлених та відповідний йому тип. Json елементи поставляються у тимчасову теку із оригінальними фото та повертаються користувачеві архівом.

При виборі обробки даних система запускає відповідний обробник та починає процедуру редагування фото. З оброблених фото формується набір даних, що записується у новостворену тимчасову директорію та повертається користувачеві у клієнт для порівняльної характеристики та можливого подальшого вивантаження.

Після закінчення роботи роботи усі тимчасові директорії видаляються

3.3 Моделювання класів системи розпізнавання об’єктів дорожнього руху

Моделювання класів і об'єктів є важливим етапом у розробці серверної частини клієнт-серверної системи. На цьому етапі розробляються UML (діаграми класів та об'єктів, які дають змогу візуалізувати структуру системи, взаємозв'язки між її елементами та визначити основні компоненти, що будуть реалізовані у вигляді класів у програмному коді.

UML (Unified Modeling Language) — це уніфікована мова моделювання, яка дозволяє візуально представляти структуру та поведінку програмних

систем. Одним з основних інструментів UML є діаграми класів і об'єктів, які допомагають розробникам проєктувати системи, моделювати зв'язки між компонентами, планувати інтеграцію та забезпечувати кращу підтримку коду.

Моделювання за допомогою UML-діаграм класів і об'єктів надає детальне бачення архітектури програмного забезпечення. Це забезпечує можливість:

- визначити необхідні класи та їхні властивості;
- зрозуміти зв’язки та взаємодії між компонентами;
- уникнути надмірності та ускладнень у структурі;
- забезпечити узгодженість системи на всіх рівнях розробки та впровадження.

Таким чином, UML-діаграми класів і об'єктів дозволяють не лише оптимально спланувати структуру серверної частини, але й спрощують наступні етапи тестування та інтеграції.

Діаграма класів (UML Class Diagram) - є статичною моделлю, яка фокусується на структурі системи. Вона показує класи, їх атрибути, методи та зв'язки між ними, забезпечуючи загальне уявлення про те, з яких компонентів складається система.

Основні елементи діаграми класів:

- Клас — основний елемент, який відображається у вигляді прямокутника з трьома секціями;
- Ім'я класу — розташовується у верхній частині;
- Атрибути — розміщуються в середній частині та представляють властивості або змінні класу;
- Методи — у нижній частині, представляючи функціонал або поведінку класу;
- Взаємозв’язки між класами — ключовий аспект діаграми, який відображає типи зв’язків між класами, як-от асоціація, агрегація, композиція та наслідування.

Типи зв'язків між класами:

- Асоціація — відображає зв’язок між двома класами, де один клас може використовувати функціонал іншого. Зазвичай представлений суцільною лінією;
- Агрегація — показує, що один клас складається з кількох інших, які можуть існувати окремо від нього. На діаграмі відображається порожньою ромбоподібною позначкою на боці класу, що містить інші;
- Композиція — вказує на те, що один клас є складовою частиною іншого і не може існувати окремо. Позначається суцільним ромбом на боці класу-власника;
- Успадкування (або генералізація) — показує ієрархічний зв’язок між класами, де один клас успадковує властивості іншого. Відображається порожньою стрілкою, спрямованою до батьківського класу.

Створимо UML діаграму класів для нашої системи, яка складається з наступних атрибутів, методів та зв’язків між класами.

UML діаграми класів зображені на рис. 3.3

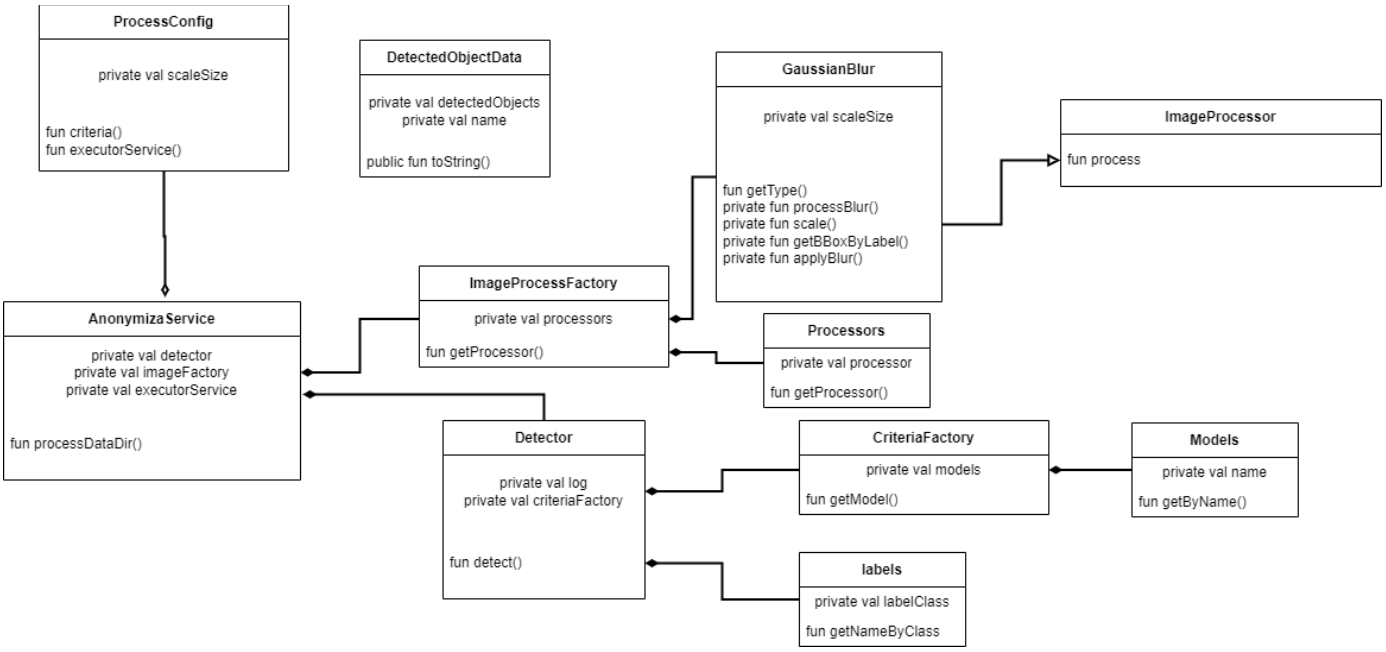


Рисунок 3.3 – UML діаграма основних класів системи

Структура основних атрибутів та методів основних класів наведена нижче:

- Клас «AnonymizeService»:
  - Атрибути: Немає.
  - Методи:
    - processData(): ініціює запуск моделі ШІ та обробляє вихідні дані
  - Зв'язок: Немає
- Клас «Detector»
  - Атрибути:
    - criteria: вибрана модель ШІ
    - log: для логування процесу
  - Методи
    - detect(): запускає процес розпізнавання
  - Зв'язки:
    - Агрегація з ModelZoo і Predictor:
- Клас «ProcessConfig»:
  - Атрибути
    - scaleSize: Вказує на степінь зжаття
  - Методи
    - criteria(): Повертає зконфігуровану модель ШІ
  - Зв’язок
- Клас «Controller»:
  - Атрибути
    - anonymizeService
  - Методи
    - startProcess(): Надсилає набір даних та дані конфігурації для прочатку процесу
  - Зв’язок: Немає

Інтерфейс «ImageProcessor»:

- Методи:
  - process()
  - getType()
- Зв’язок: є інтерфейсом для JsonOutputProcessor

- Клас «GaussianBlur»:
  - Атрибути
    - scaleSize: показує роздільну здатність з якою працювала модель
    - getType: повертає тип процесора
  - Методи
    - Process(): Обробляє дані та переводить інформацію про фото до масштабів оригінальної роздільної здатності
  - Зв’язок
    - Імплементує інтерфейс ImageProcessor
- Клас «CriteriaFactory»:
  - Атрибути
    - models: мапа, яка асоціює ім'я моделі (modelName) з відповідним об'єктом
    - Criteria<Any, Any>. Ця мапа формується на основі списку критеріїв, переданих у конструктор.Методи
  - Методи
    - getModel(): повертає модель по її назві
  - Зв’язок: Немає
- Клас «ImageProcessorFactory»:
  - Атрибути
    - processors: мапа, яка асоціює ім'я обробника (imageProcessor) з відповідним об'єктом
  - Методи

        - getProcessor(): повертає відповідну імплементацію процесора по його типу
    - Зв’язок: Немає
- Клас «ImageProcessorFactory»:
    - Атрибути
        - processors: мапа, яка асоціює ім'я обробника (imageProcessor) з відповідним об'єктом
    - Методи
        - getProcessor(): повертає відповідну імплементацію процесора по його типу
    - Зв’язок: Немає
- Клас «detectedObjectData»:
    - Атрибути
        - detectedObjects: список знайдених об’єктів
        - name: Ім’я оригіналу фото
    - Методи: немає
    - Зв’язок: Немає
- Enum«Models»: Перечислення моделей
- Enum«ProcessorTypes»: Перечислення типів процесорів
- Enum«Labels»: Перечислення об’єктів що може розпізнати модель
    - Атрибути
        - Models: список моделей що мають можливість розпізнавати цей об’єкт
    - Методи
        - getByModel(): повертає можливі об’єкти які може розпізнати модель

3.4 Розробка програмного забезпечення

Розробка програмного забезпечення є ключовим етапом створення клієнт-серверної системи для автоматизації процесу розпізнання об’єктів дорожнього руху. Цей процес передбачає не лише технічну реалізацію функціоналу серверної частини, але й забезпечення її інтеграції з клієнтськими додатками, відповідність вимогам продуктивності, надійності та масштабованості.

Серверна частина системи виконує центральну роль, забезпечуючи обробку даних, що є критично важливим для реалізації цілей автоматизації.

Одним із принципів проєктування програмного забезпечення є розподілення класів по чітко визначеним завданням кожного шару. Це розділення дозволяє створювати масштабовану, зрозумілу та легку в обслуговуванні систему.

В даному ПЗ буде використано три частини: Routes/Services/Modules.

Routes (Маршрути) - відповідають за обробку HTTP-запитів і передачу даних між клієнтом (користувачем API) та сервером. Це точка входу для клієнтських запитів.

Services (Сервіси) - містять бізнес-логіку додатка, наприклад, як саме обробляються запити, взаємодія з базою даних, обчислення або складні операції. Тут реалізується основний "мозок" системи.

Modules (Модулі) - відповідають за інфраструктуру програми

Частина 1. Запуск системи розпізнавання

Інтерфейс користувача:

Меню:

```
Modal.setAppElement("#root");

const Menu = () => {
    const style = {
```

```
    height: "70px",
    width: "240px"
  }
  const [modalIsOpen, setModalIsOpen] = useState(false);

  const openModal = () => setModalIsOpen(true);
  const closeModal = () => setModalIsOpen(false);

return (
   <div className='menu'>
     <div id='div1'>
       <Modal
         isOpen={modalIsOpen}
         onRequestClose={closeModal}
         contentLabel="Example Modal"
         style={{
           overlay: {
             backgroundColor: "rgba(0, 0, 0, 0.9)",
           },
           content: {
             top: "50%",
             left: "50%",
             right: "auto",
             bottom: "auto",
             marginRight: "-50%",
             transform: "translate(-50%, -50%)",
             padding: "20px",
             borderRadius: "8px",
             color: "black",
             boxShadow: "0 4px 6px rgba(0, 0, 0, 0.1)",
```

```
                    },
                }}
            >
                <h2 className='t' >Choose parameters</h2>
                <p  className='t'  value={outputTypes} onChange={setOutput}>output Type<select className='t'>
                </select></p>
                <p  className='t'  value={modelTypes} onChange={setModel}>Model Type<select className='t'>
                </select></p>
                <p  className='t'  value={modelType} onChange={setType}>Label Types<select className='t'>
                </select></p>
                <p> Confidence <input value={confienceValue}/> </p>
                <p><button onClick={importData}>import data</button></p>
                <p></p>
                <div>
                  <button  className='button' onClick={closeModal}>Close</button>
                  <button  className='button'  onClick={sendRunInfo(dataset, outputType, modelType, confienceValue, )}>StartRun</button>
                </div>
              </Modal>
            </div>
            <div  className="menu"><button  className='button'  role="button" onClick={openModal} style={style}>Start</button></div>
          </div>
        )
      }
```

```
export default Menu
```

Наведений код представляє собою меню, що займає лише 20% від екрану користувача, та відповідає за запуск процесу розпізнавання об’єктів, на початку присутня кнопка старт, яка викликає модаль із набором налаштувань запуску та можливістю імпорту набору зображень, якщо вибрати тип output – json, то програма видасть архів із набором об’єктів класу detectedObjectData, якщо вибрати інші варіанти, то усі види подальших маніпуляцій будуть здійснені на головному інтерфейсі

Головний інтерфейс

```
const style = {
  height: "480px"
}
const Workspace = () => {
  function left() {
    // ControllClient.get()
  }
  function right() {
    ControllClient.get()
  }
  const [imageD, setImage] = useState('')
  var origImage = require(data[index])
  var blured =require(data[index])
  return (
    <div className='workspace'>
      <div className='imgSpace'>
        <div id='blocks'><img style={style} src={origImage} alt="My logo" /></div>
        <div id='blocks'><img style={style} src={blured} alt="My logo" /></div>
      </div>
```

```
            <div className='buttonsspace'>
              <RxArrowLeft id='workspace-buttons' onClick={index -
1}></RxArrowLeft>
              <RxArrowRight                                id='workspace-buttons'
onClick={index+1}></RxArrowRight>
            </div>
          </div>
        )
      }
      export default Workspace
```

Наведений код представляє собою головний інтерфейс, що прихований спочатку, та показується лише за наявності даних для порівняння(якщо при запуску вибрати відповідні output).

Він має кнопки для проходження по списках фото, що отримані із контролера, для проведення порівняння роботи моделі та її здатності виявляти вибрані об’єкти.

Частина 2. Логіка запуску та обробки зображень

ProcessConfig

```
    @Bean
      fun criteria(): Criteria<Image, DetectedObjects> {
        val pipeline = Pipeline()
        pipeline.add(Resize(scaleSize))
        pipeline.add(ToTensor())
        val translator: ObjectDetectionTranslator = YoloV5Translator
          .builder()
          .setPipeline(pipeline)
          .optThreshold(0.8f)
          .optSynsetArtifactName("anonymize_model/synset.txt")
          .build()
        return Criteria.builder()
```

```
            .setTypes(Image::class.java, DetectedObjects::class.java)
            .optProgress(ProgressBar())
            .optModelUrls("/home/ihor/IdeaProjects/demo/src/main/resources/")
            .optModelName("anonymize_model/anonymize.torchscript")
            .optEngine("PyTorch")
            .optTranslator(translator)
            .build()
    }
```

Наведений код описує клас, що відповідає за ініціалізацію моделей ШІ при запуску системи. Завдяки Spring boot та анотації @Bean ми маємо можливість викликати об’єкт ініалізованої моделі будь де використовуючи CriteriaFactory

CriteriaFactory

```
package org.example
import ai.djl.repository.zoo.Criteria
class CriteriaFactory(vararg criterias: Criteria<Any, Any>) {
   private val models: Map<String,Criteria<Any,Any>> = criterias.associateBy { criteria -> criteria.modelName }
   fun getModel(name:String): Criteria<Any, Any>? {
      return models[name]
   }
}
```

Наведений код представляє собою клас, створений із використанням паттерну розробки Factory[3], який буде зберігати посилання усі екземпляри Criteria, та отримувати доступ відповідного екземпляра по імені моделі. Обрана Criteria далі запускається під час виконання методу класу Detector

Detector

```
@Component
class Detector(){
```

```
    private val log = LoggerFactory.getLogger(javaClass)
    fun detect(images: Set<Path>, criteria: Criteria<Any, Any>): Map<Path,
DetectedObjects> {
        val imageFactory = ImageFactory.getInstance()
        val model = ModelZoo.loadModel(criteria)
        val predictor = model.newPredictor()
        val map: MutableMap<Path, DetectedObjects> = images.stream()
            .collect(Collectors.toMap({ it }, {
                predictor.predict(imageFactory.fromFile(it))
            }))
        model.close()
        log.info("Model successfully loaded, started prediction!")
        return map
    }
```

Код наведений вище представляє собою клас Detector запускається виконанням методу processData класу AnonymizeService. Відповідає за запуск сконфігурованої моделі, збір даних по її запуску, та передачі до AnonymizeService для остаточного

AnonymizeService

```
@Service
class AnonymizeService(
    private val detector: Detector,
    private val gaussianBlur: GaussianBlur,
    private val executorService: ExecutorService,
    private val criteriaFactory: CriteriaFactory,
    private val processFactory: ProcessFactory
) {
    private val log = LoggerFactory.getLogger(javaClass)
```

```
    fun processData(inputDir: File, outputDir: File, type: ProvessType, modelName: String, vararg labels: Labels): Int {
        val model = criteriaFactory.getModel(modelName)
        val process = processFactory.getProcess(type)
        var processedImages = 0
        val images: Set<Path> = inputDir.listFiles()!!.asList().stream()
            .map { it.toPath() }.collect(Collectors.toSet())
        val map: Map<Path, DetectedObjects> = detector.detect(images, model)
        log.info("Detection completed. Started image processing")
        ListUtils.partition(map.keys.stream().toList(), Runtime.getRuntime().availableProcessors())
            .forEach {
                CompletableFuture.supplyAsync({
                    process.process(map, it, outputDir, *labels)
                }, executorService)
                    .thenAccept { processedImages += it }
            }
        return processedImages
    }
}
```

Код наведений вище представляє собою клас AnonymizeService викликається контролером та є класом основної бізнес логіки, його задача це конфігурація та запуск моделі через клас Detector, обробка даних по отриманій із запиту конфігурації з допомогою ProcessFactory та остаточна передача даних клієнту

ProcessorFactory

```
class ProcessorFactory(vararg processes: ImageProcessor) {
    private val processes:Map<ProcessType,ImageProcessor> = processes.associateBy { it.getType() }
}
```

Код наведений вище представляє собою клас який, створений з допомогою патерну Factory, який має посилання на усі екземпляри класів обробки даних та дає можливість отримати потрібний по його типу

У підсумку можна сказати, що код розподілений за функціоналом, тому демонструє добре структуровану систему, яка легко підтримується. Використання абстракцій та паттернів дають можливість швидкого та легкого розширення функціоналу. Коментарі пояснюють кожен крок, роблячи код зрозумілим навіть для стороннього розробника.

3.5 Тестування програмного забезпечення

Тестування програмного забезпечення є невід'ємною частиною життєвого циклу розробки будь-якої програмної системи, особливо у випадках створення клієнт-серверних систем, які мають забезпечувати стабільну та безпечну роботу в умовах великого навантаження.[20] Метою тестування є виявлення помилок у функціональності, продуктивності, безпеці та сумісності, що дозволяє підвищити якість програмного продукту та мінімізувати ризики, пов’язані з його використанням.

У процесі тестування серверної частини клієнт-серверної системи для автоматизації процесу розпізнавання особливу увагу приділяють перевірці правильності обробки запитів, відповідності API задекларованим специфікаціям, а також стійкості системи до некоректних даних чи збоїв. Даний етап дозволяє гарантувати, що створене програмне забезпечення відповідає вимогам користувачів, забезпечує належний рівень продуктивності та здатне функціонувати стабільно у реальному середовищі.[21]

У процесі розробки та тестування серверної частини клієнт-серверної системи для автоматизації процесу формування команд учнів ми використовуватимемо Swagger як інструмент для документування та тестування API[22].

Swagger — це набір відкритих інструментів для створення, опису, документування та тестування REST API. Основна мета Swagger — зробити API зрозумілим як для розробників, так і для кінцевих користувачів, надаючи детальну специфікацію його функціоналу у зрозумілому форматі. Swagger підтримує стандарт OpenAPI, що є широко розповсюдженим і прийнятим у сучасній індустрії розробки програмного забезпечення.

Отже, використання Swagger забезпечує ефективну взаємодію між розробниками та тестувальниками, дозволяючи перевіряти коректність роботи API у зручному середовищі.

Тому в межах цього пункту ми проведемо тестування основних функцій з метою знаходження та усунення багів.

Першим етапом у системі розпізнавання об’єктів дорожнього руху є отримання даних для запуску процесу, цей етап включає в себе виклик трьох запитів для на отримання, а саме:

- Типи output;
- Моделі ШІ;
- Об’єкти з якими працюють моделі ШІ;

Виклик об’єктів не може бути зроблений без попереднього виклику та отримання моделей ШІ, тому що залежить від його response

Тест 1 Отримання доступних видів output

Під час початку роботи користувач отримує всі види output, які відображатимуться у списку під час конфігурації та дадуть можливість вибору потрібного типу. Отримання проводиться за допомогою запиту HTTP типу GET на сервер із URI: /types.

Тестування отримання типів наведено на рис. 3.4

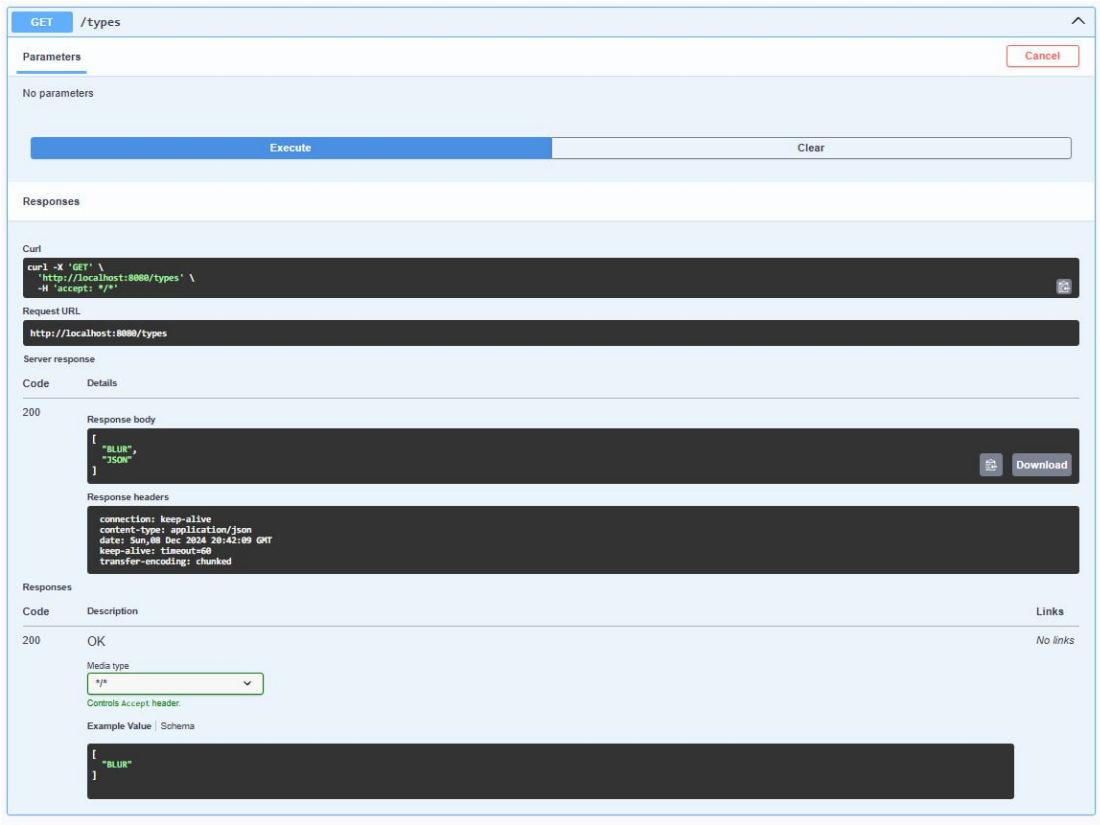


Рисунок 3.4 – Отримання доступних видів output

Результатом проведення тестування запуску отримання типів обробки, які доступні користувачеві є список доступних видів output із кодом HTTP 200, що є правильною поведінкою для цього запиту. Отримані дані у форматі Json передаються клієнту та використовуються у вікні вибору конфігурації, яке зображене на рисунку Б.2

Наступним кроком є отримання моделей ШІ для розпізнавання, він може бути запущений незалежно від виклику проведеному у попередньому кроці

Тест 2 Отримання доступних моделей

Користувач отримує доступні йому види моделей які відображатимуться у списку під час конфігурації та дадуть можливість вибору потрібної моделі. Отримання проводиться за допомогою запиту HTTP типу GET на сервер із URI: /models

Тестування отримання моделей наведено на рис. 3.5

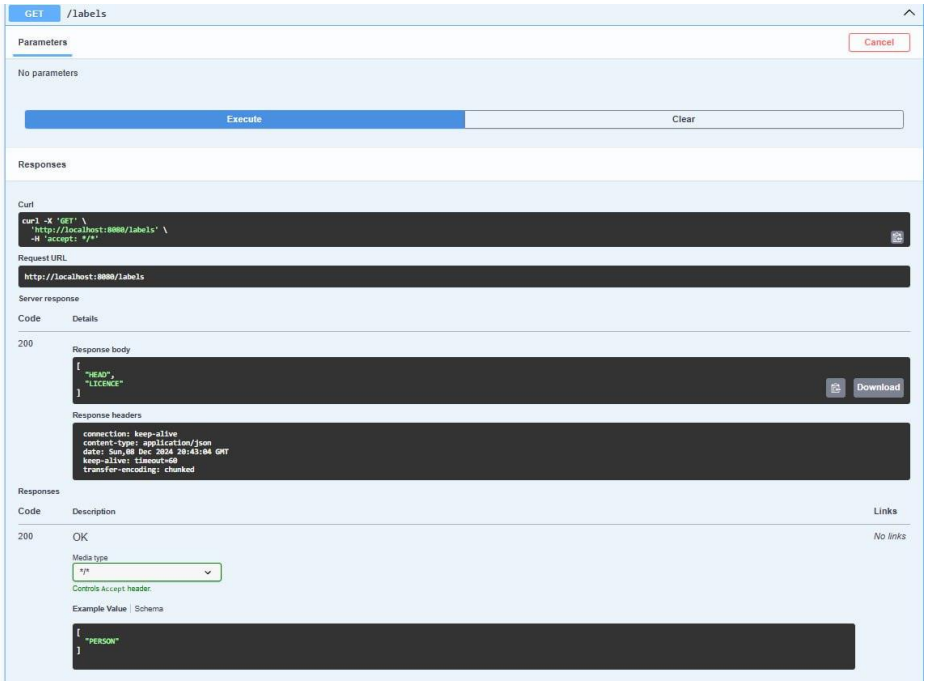


Рисунок 3.5 – Отримання доступних моделей

Результатом проведення тестування запуску отримання доступних моделей є response із кодом 200, що сигналізує про успішне отримання доступних моделей ШІ. Отримані дані у форматі Json передаються клієнту та використовуються у вікні вибору конфігурації.

Після вибору моделі у відповідному вікні користувачеві стає доступний вибір типів об’єктів, які модель поставляє, який раніше був закритий через нестачу інформації для проведення відповідного запиту на сервер

Наступним кроком є отримання моделей об’єктів , які може розпізнати модель. Цей крок залежить від попереднього тому не може бути виконаний незалежно і тому відповідна колонка у вікні конфігурації є неактивною до моменту вибору моделі

Тест 3 Отримання доступних видів label

Користувач отримує доступні йому види об’єктів, які може розпізнати вибрана ним модель та які відображатимуться у списку під час конфігурації та після вибору моделі, вони дадуть можливість вибору потрібних об’єктів які будуть розпізнані. Отримання label проводиться за допомогою запиту HTTP типу GET на сервер із URI: /labels та має наступні параметри:

– modelName – Тип: String. Назва моделі для якої повинні бути повернені типи розпізнаваних об’єктів.

Тестування отримання видів output наведено на рис. 3.6

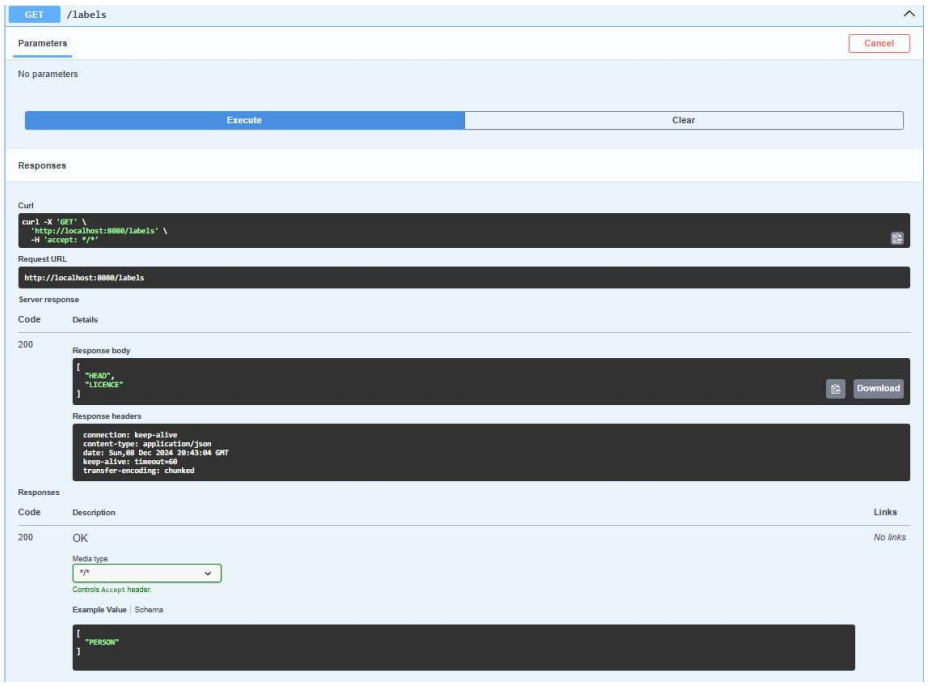


Рисунок 3.6 – Отримання доступних видів output

Результатом проведення тестування запуску отримання доступних видів об’єктів, що розпізнає модель є response із кодом 200, що сигналізує про успішне отримання доступних видів об’єктів. Отримані дані у форматі Json передаються клієнту та використовуються у вікні вибору конфігурації.

Кінцевим етапом є запуск процесу розпізнавання, який може бути запущений лише після усіх попередніх етапів, тому що використовує інформацію, отриману у них

Тест 4 Запуск розпізнавання

Процес розпізнавання проводиться за допомогою запиту HTTP типу POST на сервер із URI /detect та має параметри:

– label – Тип: Spring[]. Масив типів об’єктів, що потрібно розпізнати

- outputType – Тип: String. Тип output, який система розпізнавання об'єктів дорожнього руху повинна представити користувачеві
- confidenceValue – Тип Int. Числовий показник, який вказує відсотковий поріг впевненості моделі у виявленому об’єкті який є прийнятний для користувача. Мінімальним значенням є значення задане у моделі а максимальним є значення 100. При виході за межі значень не враховується
- model – Тип: String. Назва моделі, яка буде використана під час виконання процесу розпізнавання
- file – Тип: MultipartFile. Набір даних який завантажує користувач

Тестування запуску процесу розпізнавання наведено на рис. 3.7

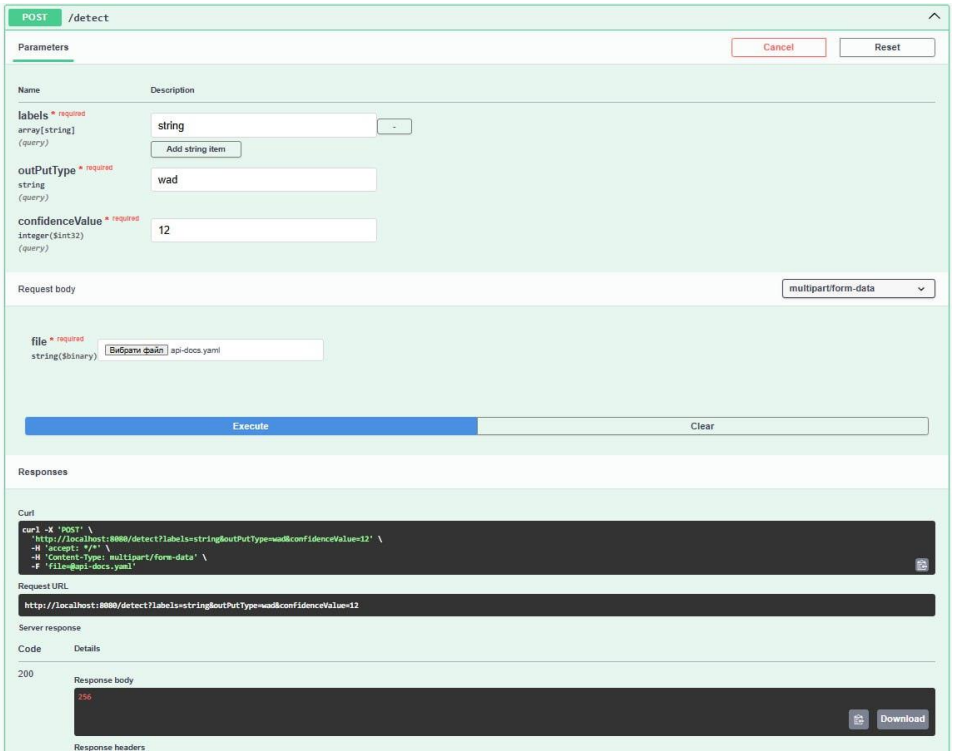


Рисунок 3.7 – Тестування процесу запуску розпізнавання

Результатом проведення тестування запуску розпізнання є response із кодом 200, що сигналізує про успішне виконання процесу розпізнавання та обробки. Користувач отримав кількість оброблених фото та архів із необробленими

даними. Інформація з сервера під час виконання процесу розпізнавання зображена на рисунку Б.3

В Результаті виконання даного підпункту було протестовано функціональність виконання запитів на API за допомогою інструменту Swagger, що дозволило побачити коректну взаємодію сервера з клієнтською частиною та підтвердити справність роботи системи розпізнавання об'єктів дорожнього руху.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит розробленої системи розпізнавання об’єктів дорожнього руху

Як було зазначено раніше, у сучасному інформаційному суспільстві автоматизація та розпізнавання об'єктів у веб-додатках стають ключовими аспектами розвитку технологій, особливо в таких сферах, як безпека, комерція, автомобільна індустрія, медицина та багато інших. Системи, які здатні автоматично аналізувати зображення та відео, дозволяють значно поліпшити ефективність роботи та зменшити необхідність ручного втручання.

Більш того, з розвитком технологій машинного навчання, штучного інтелекту та комп'ютерного зору потреба в автоматизованих системах розпізнавання об'єктів значно зросла, що викликає необхідність автоматизації цього процесу та підвищення точності і швидкості розпізнавання різних об’єктів, зокрема – обличь людей.

Тому метою виконаної нами магістерської кваліфікаційної роботи була система розпізнавання об'єктів дорожнього руху з використанням сучасних алгоритмів машинного навчання та бібліотек комп'ютерного зору, таких як OpenCV, TensorFlow або YOLO.

В результаті виконаної роботи нами було розроблено систему алгоритмів та інструментів для автоматизованого розпізнавання об’єктів дорожнього руху, яка базується на використанні передових технологій машинного навчання, пов’язаних із патернізацією великого набору даних.

Для встановлення комерційного потенціалу розробленої нами системи розпізнавання об’єктів дорожнього руху (в подальшому – автоматизованої системи) було запрошено 3-х експертів-практиків – фахівців у цій галузі знань, які безпосередньо працюють над роз’язанням цієї проблеми: Антонюка Олега (Software engineer), Копицю Вадима (Software engineer), та

Довгого Олексія (Software engineer), які здійснили експертне оцінювання комерційного потенціалу нашої розробки за критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно	Ціна продукту дещо	Ціна продукту приблизно дорівнює	Ціна продукту дещо нижче за	Ціна продукту значно нижче за ціни аналогів

	вища за ціни аналогів	вища за ціни аналогів	цінам аналогів	ціни аналогів	
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела фінансува	Потрібні значні фінансові ресурси.	Потрібні незначні фінансові ресурси.	Не потребує додаткового фінансування

	Джерела фінансування ідеї відсутні	ння відсутні	Джерела фінансування є	Джерела фінансуванн я є	
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка	Необхідно	Процедура	Необхідно тільки	Відсутні будь-які

	регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	повідомлення відповідним органам про виробництво та реалізацію продукту	регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти-практики оцінили комерційний потенціал розробленої нами автоматизованої системи таким чином (див. табл. 4.2):

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробленої автоматизованої системи (за шкалою оцінювання 0-1-2-3-4)

Критерії	Ініціали, прізвище експертів		
	О. Антонюк	В. Копиця	О. Довгий
	Бали, що їх виставили експерти:		
1	4	4	3
2	4	3	4

3	4	3	4
4	4	4	4
5	4	4	4
6	4	4	4
7	3	4	4
8	4	4	3
9	4	4	3
10	4	4	4
11	4	4	4
12	3	3	4
Сума балів	$СБ_1 = 46$	$СБ_2 = 45$	$СБ_3 = 45$
Середньоариф метична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{46+45+45}{3} = \frac{136}{3} = 45,33$		

Для встановлення комерційного потенціалу розробленої нами автоматизованої системи скористаємося рекомендаціями, наведеними в таблиці 4.3

Таблиця 4.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький

11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 45,33 балів (із максимально можливих 48-ми балів), то це свідчить, що розроблена нами система розпізнавання об’єктів дорожнього руху має рівень комерційного потенціалу, який вважається «високим».

Це пояснюється тим, що розроблена нами автоматизована система розпізнавання об’єктів дорожнього руху здійснює автоматизацію задач, які розв’язує людина, дозволяє пришвидшити збір даних, покращити алгоритми та створювати нові рішення, що раніше вимагало значно більших людських ресурсів, оскільки ці рішення були ризикові та часто не релевантні.

4.2 Розрахунок витрат на розроблення системи розпізнавання об’єктів дорожнього руху

При розробленні системи розпізнавання об ‘єктів дорожнього руху були зроблені такі витрати:

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн},$$

(4.1)

де М – місячний посадовий оклад розробника, грн; приймемо, що:

М = (8000…25000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 24 дні;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 4.4:

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів (годин) роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	25000	1041,67	20 годин	(1041,67 / 6) × 20 = ≈ 3472 (при 6-годинному робочому дні)
2. Здобувач-магістрант (виконавець)	2000 грн (беремо 8000 грн)	333,33	88 днів	≈ 29333
3. Консультант з економічної частини	18000	750,00	1,5 години	≈ 186 (при 6-годинному робочому дні)
Загалом				$З_о$ = 32991 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \qquad (4.2)$$

Приймемо, що α = 0,11. Тоді для нашого випадку отримаємо:

$З_д$ = 0,11 × 32991 = 3629,01 ≈ 3629 грн.

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \qquad (4.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. В 2024 році β = 22%. Тоді:

$НЗН_{зп} = (32991 + 3629) \times 0,22 = 8056,40 \approx 8057$ грн.

Г). Амортизація основних засобів А, які використовувались під час виконання магістерської кваліфікаційної роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн,} \qquad (4.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Спрощено можна прийняти, що $Н_а = (2,5...25)\%$;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 4.5.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	80000	25	3,2 (при 80% використанні	4266,66 ≈ 4267
2. Приміщення університету, факультету, кафедри	25000	4	3,2 при 25% використанні	66,66 ≈ 67
Всього				**А = 4334 грн**

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad грн, \qquad (4.5)$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го най-менування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \quad грн, \qquad (4.6)$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання магістерської кваліфікаційної роботи загальні витрати на матеріали та комплектуючі склали укрупнено приблизно 1800 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \qquad (4.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2024 р. В ≈ 4,8 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,3 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 270 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,81.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,68.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,8 \cdot 1,3 \cdot 270 \cdot 0,81}{0,68} = 2006,89 \approx 2007 \quad грн.$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (4.8)$$

Для нашого випадку отримаємо:

$В_{інш} = 1,1 \times 32991 = 36290,1 \approx 36290$ грн.

К). Сума всіх попередніх статей витрат складає витрати на виконання нашої магістерської роботи безпосередньо розробником-магістрантом – В.

В = 32991+ 3629 + 8057+ 4334 + 1800 + 2007 + 36290 = 89108 грн.

Л). Загальні витрати на розробку та можливу комерціалізацію розробленої нами автоматизованої системи розпізнавання об‘єктів дорожнього руху розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи.

Оскільки наша розробка на цей момент часу практично готова до впровадження, то можна прийняти, що, $\beta \approx 0,9$

Тоді: $В_{заг} = \frac{89108}{0,9} = 99008,88$ грн або приблизно 99 тисяч грн.

Тобто прогнозовані загальні витрати на розробку та можливу комерціалізацію розробленої нами автоматизованої системи розпізнавання об‘єктів дорожнього руху становлять приблизно 99 тисяч грн.

4.3 Розрахунок економічного ефекту від можливої комерціалізації нашої розробки

Проведене нами дослідження ринку показало, що розроблена нами автоматизована система розпізнавання об‘єктів дорожнього руху, яка

орієнтована на осіб і організацій, які займаються збиранням аналогічних даних, знайде широкий попит на ринку і відкриває широкий спектр можливостей для застосування нашої розробки при розв’язанні низки інших подібних задач.

Аналіз місткості ринку цього продукту також показує, що на сьогодні в Україні кількість реальних користувачів подібних автоматизованих систем може становити приблизно 200 фізичних та юридичних осіб щороку. Можна також очікувати зростання попиту на нашу розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо наша розробка буде впроваджена з 1 січня 2025 року (оскільки вона практично готова до впровадження), то її результати будуть виявлятися протягом 2025-го, 2026-го та 2027-го років.

Прогноз зростання попиту на нашу розробку може складати по роках:

а) 2025 р. – приблизно +20 шт. (відносно базового року);

б) 2026 р. – +30 шт. (відносно базового року);

в) 2027 р. – +15 шт. (відносно базового року), що пояснюється високою ймовірністю появи нових, більш кращих розробок.

Економічний ефект від можливої комерціалізації розробленої нами автоматизованої системи розпізнавання об‘єктів дорожнього руху пояснюється її значно кращими функціональними можливостями. Тому нашу розробку можна буде реалізовувати на ринку дещо дорожче, ніж аналогічні за функціями розробки. Так, якщо подібні за функціями розробки у 2024 році коштували на ринку приблизно 100 тисяч грн, то нашу розробку можна буде реалізовувати на ринку приблизно за 125 тисяч грн або на 25 тисяч грн дорожче.

Тоді можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації, тобто виведення нашої розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}),$$

(4.10)

де $\Delta Ц_о$ – покращення основного якісного показника від впровадження результатів розробки. Для нашого випадку це є збільшення ціни реалізації нашої розробки $\Delta Ц_о$ = 125 - 100 = + 25 тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 200 шт.;

Δ N – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення становитиме по роках, відповідно: у 2025 році – + 20 шт., у 2026 році +30 шт., та у 2027 році +15 шт. (до базового 2024 року);

$Ц_о$ – основний якісний показник (тобто ціна), який являє собою ціну реалізації нашої розробки після її виведення на ринок, грн; $Ц_о$= 125 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,4;

υ – ставка податку на прибуток. У 2024 році υ = 18%.

У 2025-2027 роках будемо очікувати, що υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження нашої розробки (2025 р.) становитиме:

$$\Delta П_1 = [25 \cdot 200 + 125 \cdot 20] \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - \frac{18}{100}) = 2049{,}92 \approx 2050 \text{ тисяч}$$

грн.

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження нашої розробки протягом другого (2026 р.) року становитиме:

$$\Delta П_2 = [25 \cdot 200 + 125 \cdot 30] \cdot 0,8333 \cdot 0,4 \cdot (1 - \frac{18}{100}) = 2391,57 \approx 2392$$ тисяч грн.

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження нашої розробки протягом третього (2027 р.) року становитиме:

$$\Delta П_3 = [25 \cdot 200 + 125 \cdot 15] \cdot 0,8333 \cdot 0,4 \cdot (1 - \frac{18}{100}) = 1879,09 \approx 1880$$ тис. грн.

Приведена вартість зростання для потенційного інвестора всіх чистих прибутків від можливого впровадження і комерціалізації нашої розробки становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої нашої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для нашого випадку т = 3 роки;

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення автоматизованої системи розпізнавання об‘єктів дорожнього руху до моменту отримання можливих чистих прибутків від її впровадження і виведення на ринок.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, складе:

$$ПП = \frac{2050}{(1+0,1)^2} + \frac{2392}{(1+0,1)^3} + \frac{1880}{(1+0,1)^4} \approx 1694 + 1797 + 1284 = 4775$$ тисяч грн.

Теперішня вартість інвестицій PV, що повинні бути вкладені в реалізацію нашої розробки: $PV = (1,0…5,0) \times B_{заг}$.

Для нашого випадку $PV = (1,0…5,0) \times 99 = 5,0 \times 99 = 495$ тисяч грн.

Розраховуємо абсолютний ефект від можливих вкладених інвестицій $E_{абс}$.

$$E_{абс} = ПП - PV, \quad (4.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливого впровадження нашої розробки, грн;

Абсолютний ефект від можливого впровадження нашої розробки (при прогнозованому ринку збуту) за три роки складе:

$E_{абс} = 4775 – 495 = 4280$ тисяч грн.

Оскільки $E_{абс} > 0$, то комерціалізація нашої розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $E_в$ вкладених інвестицій:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.13)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс} = 4280$ тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 495 тисяч грн;

$T_ж$ – життєвий цикл розробки, роки.

$T_ж = 4$ роки (2024-й, 2025-й, 2026-й, 2027-й роки)

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{4280}{495}} - 1 = \sqrt[4]{1 + 8,6465} - 1 = \sqrt[4]{9,6465} - 1 = 1,762 - 1 = 0,762 \approx 76,2\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,10...0,12), а окремі банки встановлюють ставку за депозитними операціями на рівні до 18%. Приймемо, що τ = 18%.

f – показник, що характеризує ризикованість вкладень; f = (0,05...0,30).

В умовах війни приймемо, що f = 50%, тобто f = 0,5.

Тоді для нашого випадку отримаємо:

$\tau_{мін} = 0,18 + 0,50 = 0,68$ або $\tau_{мін} = 68\%$.

Оскільки величина $E_в = 76,2\% > \tau_{мін} = 68\%$, то потенційний інвестор у принципі може бути зацікавлений у комерціалізації розробленої нами автоматизованої системи розпізнавання об‘єктів дорожнього руху.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленої нами автоматизованої системи розпізнавання об‘єктів дорожнього руху.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.15)$$

Для нашого випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0,762} = 1,31 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленої автоматизованої системи розпізнавання об‘єктів дорожнього руху.

Далі проведемо моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні. Як відомо, в наступні роки через військову агресію росії рівень інфляції в Україні, на жаль, може суттєво зрости. Прийнявши рівень інфляції у 30%, отримаємо:

$$ПП = \frac{2050}{(1+0,3)^2} + \frac{2392}{(1+0,3)^3} + \frac{1880}{(1+0,3)^4} \approx 1213 + 1089 + 658 = 2960 \text{ тисяч грн.}$$

Тоді абсолютний економічний ефект від можливого впровадження нашої розробки за три роки складе:

$E_{абс}$ = 2960 – 495 = 2465 тисяч грн.

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 2465 тисяч грн;

PV – теперішня вартість початкових інвестицій PV = 495 тисяч грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{2465}{495}} - 1 = \sqrt[4]{1 + 4,9798} - 1 = \sqrt[4]{5,9798} - 1 = 1,564 - 1 = 0,564 \approx 56,4\%.$$

Зроблені розрахунки у вигляді графіків наведено на рис. 4.1.

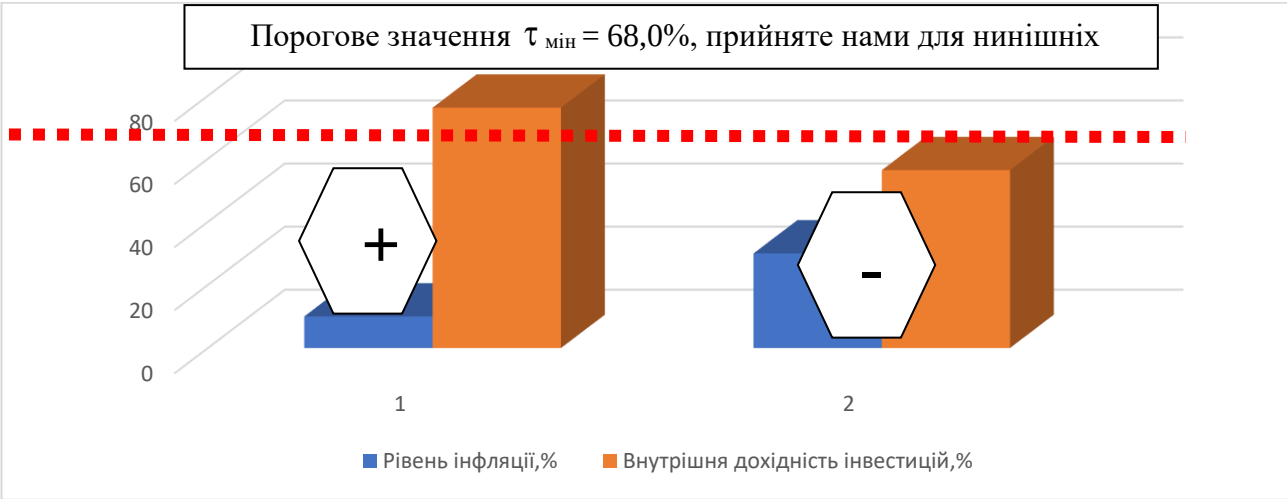


Рисунок 4.1 – Моделювання залежності величини внутрішньої дохідності

потенційних інвестицій від рівня інфляції в країні

Аналіз графіка на рис 4.1 показує, що при рівні інфляції в 10% величина внутрішньої дохідності вкладених інвестицій становить $E_в$ = 76,2%, що більше порогового значення, яке було встановлене нами величиною $\tau_{мін}$ = 68,0%, і тому комерціалізація нашої розробки для потенційного інвестора може бути доцільною.

При рівні інфляції в 30% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію нашої розробки, становить $E_в$ = 56,4%,

що менше порогового значення, яке було визначено нами, $\tau_{мін} = 68\%$, і тому для прийняття остаточного рішення потрібно провести додаткові обгрунтування і розрахунки (наприклад, знизити рівень прийнятого ризику, підняти ціну реалізації нашої розробки, збільшити попит на нашу розробку тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Не більше 100 тисяч грн	99 тисяч грн.	Досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	Не менше 4000 тисяч грн (за три роки)	4280 тисяч грн (при 10% інфляції)	Виконано
3. Внутрішня дохідність інвестицій, %	не менше 68%	76,2%	Досягнуто
4. Термін окупності інвестицій, роки	до 3-ти років	1,31 років	Виконано

Таким чином, основні техніко-економічні показники розробленої нами автоматизованої системи розпізнавання об‘єктів дорожнього руху, визначені у технічному завданні, виконані.

ВИСНОВКИ

У результаті виконання даної магістерської кваліфікаційної роботи було досягнуто основну мету — підвищення ефективності процесу розпізнавання об’єктів за рахунок впровадження технології машинного навчання та її практичну реалізацію шляхом створення системи автоматизованого розпізнавання об'єктів у веб-додатку.

На основі проведеного аналізу предметної області визначено основні проблеми, пов’язані із застосуванням технологій розпізнавання об'єктів у веб-середовищі, а також переваги автоматизованого підходу до аналізу даних в реальному часі.

Під час виконання роботи було здійснено огляд існуючих методів і систем розпізнавання об'єктів, що дозволило чітко сформулювати вимоги до розробки. Обґрунтовано вибір архітектури клієнт-серверного додатку та технологій, які забезпечують високу продуктивність, масштабованість і зручність використання.

Також були розглянуті архітектурні підходи та патерни, що застосовуються для побудови клієнт-серверних систем. Було проведено аналіз класифікації патернів та їхнього призначення. Для структурування серверної частини обрано архітектуру на основі REST API та шаблон MVC, який забезпечує розподіл відповідальності та управління між окремими компонентами системи. Досліджено ключові переваги такого підходу. У процесі створення системи були реалізовані основні API та проведено їх тестування для перевірки коректного функціонування.

Розробка програмного забезпечення включала створення функціоналу для автоматичного аналізу зображень, розпізнавання об'єктів за допомогою сучасних алгоритмів машинного навчання. Тестування системи підтвердило її коректність, надійність та відповідність встановленим вимогам.

В результаті розрахунку економічного ефекту від можливої комерціалізації даної розробки було обраховано наступні економічні

показники: витрати на розроблення системи розпізнавання - 4 тисяч грн; Абсолютний економічний ефект від впровадження розробки - 1,29 млн грн; Внутрішню дохідність потенційних інвестицій – 76,2 %; Термін окупності потенційних інвестицій – 1,31 років.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LeCun, Y., Bengio, Y., & Hinton, G. Deep learning // Nature. – 2015. – Т. 521, № 7553. – С. 436–444.

2. Smith, J., & Brown, K. Artificial Intelligence in Web Applications // IEEE Transactions on Web Engineering. – 2022. – Т. 34, № 3. – С. 123–130.

3. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. – MIT Press, 2016. – 775 p.

4. Han, S., et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1510.00149.

5. Papernot, N., et al. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1511.04508.

6. NVIDIA. CUDA Toolkit Documentation. – NVIDIA Developer, 2023.

7. Zheng, L., et al. Comparative Analysis of Manual and Automated Image Recognition Systems // Journal of Computer Vision. – 2021. – Т. 18, № 2. – С. 87–102.

8. Ramesh, A., et al. Zero-Shot Text-to-Image Generation [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2102.12092.

9. Vaswani, A., et al. Attention is All You Need // Advances in Neural Information Processing Systems. – 2017. – С. 5998–6008.

10. Глобальні інформаційні системи та технології: моделі ефективного аналізу, опрацювання та захисту даних: монографія / В.В.Пасічник, П. І. Жежнич, Р. Б. Кравець, А. М. Пелещишин, Д. О. Тарасов. – Львів: Видавництво Львівської політехніки, 2006. – 348 с. – ISBN 966-553-578-1.

11. Бевз О. М., Папінов В. М., Скидан Ю. А. Проектування програмних засобів систем управління: навчальний посібник. – Вінниця: ВНТУ, 2010. – 125 с.

12. Семков Д. Розуміння Клієнт-Серверної Архітектури на прикладах [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/44636/.

13. Бунке О. С. Серверні Web-Технології: навчальний посібник. – Київ: КПІ ім. Ігоря Сікорського, 2023. – 109 с.

14. Мови програмування: список сучасних мов, з якої мови почати вивчення програмування [Електронний ресурс]. – Ukrainian Digital Community. – Режим доступу: ukrainiandigital.com/strong-movy-prohramuvannia-dlia-pochatkivtsiv-ta-dosvidchenykh-ohliad-populiarnykh.

15. Колмогоров Н. Все, що вам потрібно знати про Node.js [Електронний ресурс]. – Режим доступу: https://medium.com/webbdev/js-db3d35ffed7e.

16. Fowler, M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002. – 557 p.

17. Гудзь С. О., Андрєєв О. М., Мірошниченко А. С. Системи комп'ютерного зору: методи, моделі, алгоритми. Київ: Наука, 2019. 256 с.

18. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // Computer Vision and Pattern Recognition (CVPR). 2015. URL: https://arxiv.org/abs/1409.1556 (дата звернення: 14.11.2024).

19. Russakovsky O., Deng J., Su H., et al. ImageNet Large Scale Visual Recognition Challenge // International Journal of Computer Vision. 2015. URL: https://arxiv.org/abs/1409.0575 (дата звернення: 21.11.2024).

20. Вільямс С. Тестування програмного забезпечення: практика та теорія. Москва : Вільямс, 2018. 456 с.

21. Філдінг Р. Архітектурні стилі та проектування програмних архітектур на основі мереж [Електронний ресурс] : дисертація. Каліфорнійський університет, Ірвайн, 2000. Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

22. OpenAPI Specification [Електронний ресурс]. OpenAPI Initiative. Режим доступу: https://github.com/OAI/OpenAPI-Specification.

23. Wightman, R., Touvron, H., Vedaldi, A. EfficientNetV2: Smaller Models and Faster Training. International Conference on Learning Representations (ICLR), 2021. URL: https://arxiv.org/abs/2104.00298

24. Amazon Web Services. Amazon Rekognition Developer Guide. – AWS Documentation, 2023.

25. Цапар В. С., Жученко О. А. Сучасні програмні засоби автоматизації технологічних процесів / НТУУ «КПІ». – Київ: НТУУ «КПІ», 2012.

26. Слободян І.О., Квєтний Р.Н. «Використання моделей штучного інтелекту у мовах програмування kotlin, java», «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», 2024 [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20902/17351.

27. Microsoft Azure. Azure Computer Vision Documentation. – Microsoft, 2023.

28. Russakovsky, O., et al. ImageNet Large Scale Visual Recognition Challenge // International Journal of Computer Vision. – 2015. – Т. 115, № 3.

29. Волошин О. Deep dive into using patterns with Selenium: огляд 8 шаблонів, що стануть у пригоді [Електронний ресурс]. – Режим доступу: https://www.globallogic.com/ua/insights/blogs/dive-into-the-selenium-patterns/.

30. He, K., et al. Deep Residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016.

31. Biamonte, J., et al. Quantum Machine Learning // Nature. – 2017. – Т. 549, № 7671.

32. Гамма Е., Хелм Р., Джонсон Р., Влісідес Дж. Патерни проєктування. Перевірені рішення повторюваних задач об’єктно-орієнтованого програмування. Київ: Вид. дім "СофтПрес", 2009. 395 с.

33. Abramov Д., Кларк А. Реактивне програмування у React: використання хуків та контекстів. Львів: Видавництво Івана Франка, 2021. 280 с.

34. Масленников С. Розробка REST API для сучасних додатків. Київ: Видавництво Наукова думка, 2020. 315 с.

35. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, 2000. URL:

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата звернення: 11 жовтня 2024 р.).

36. Liu, Y., et al. Deep Learning for Autonomous Driving: A Survey // IEEE Transactions on Neural Networks and Learning Systems. – 2020. – Т. 31, № 7.

37. Facebook Developers. React Documentation. – Meta Platforms, Inc., 2022.

38. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement // Computer Vision and Pattern Recognition (CVPR). 2018. URL: https://arxiv.org/abs/1804.02767

39. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // Advances in Neural Information Processing Systems. 2015. URL: https://arxiv.org/abs/1506.01497

40. Wightman R., Touvron H., Vedaldi A. EfficientNetV2: Smaller Models and Faster Training // International Conference on Learning Representations (ICLR). 2021. URL: https://arxiv.org/abs/2104.00298

41. Майерс Г. Мистецтво тестування програмного забезпечення / пер. з англ. Київ : Видавництво \u201cНаукова думка\u201d, 2011. 300 с.

42. Вільямс С. Тестування програмного забезпечення: практика та теорія. Москва : Вільямс, 2018. 456 с.

ДОДАТКИ

Додаток А
(Обов’язковий)
Технічне завдання

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри АІІТ ВНТУ,
д.т.н., професор
Олег БІСІКАЛО

“ ___ ” _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Система розпізнавання об'єктів дорожнього руху
08-31.МКР.007.02.000 ТЗ

Студент групи 1АКІТР-24м

_____ Ігор СЛОБОДЯН
Підпис *Ім’я ПРІЗВИЩЕ*

Керівник д.т.н., доцент. каф. АІІТ

_____

_____ Володимир КОЦЮБИНСЬКИЙ
Підпис *Ім’я ПРІЗВИЩЕ*

Вінниця 2024

1. Назва та галузь застосування

1.1. Назва – Система розпізнавання об'єктів дорожнього руху

1.2. Галузь застосування – Інформаційні технології у сфері збору даних.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ № 247 від 18-09-2024 р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є розробка та впровадження системи розпізнавання об’єктів, спрямованої на підвищення ефективності збору даних, зниження витрат та покращення якості ,що дозволить компаніям оптимізувати збір даних для розробки алгоритмів.

4. Джерела розробки.

4.1. Перелік головних етапів виконання розробки:

- визначити аспекти функціонування системи
- дослідити функціональні можливості існуючих аналогічних
- скласти вимоги до системи, що розробляється;
- спроектувати та розробити систему у відповідності до поставлених вимог

5. Показники призначення

5.1 Мінімальні вимоги до техніки:

- ОС: Ubuntu 20.04;
- Процесор: Intel Core i5 3.4 GHz або вище;
- Об’єм оперативної пам’яті: 8 Gb або більше.

5.2 Середовище розробки та запуску IntellijIdea;

6. Економічні показники

До економічних показників входять:

- витрати на розробку – не менше 4 тис. грн;
- абсолютний економічний ефект від впровадження розробки – не менше 1 млн грн;
- внутрішня дохідність потенційних інвестицій – не менше 68%;
- термін окупності потенційних інвестицій – до 3-ох років.

7. Стадії та етапи розробки.

7.1 Розділ 1 «АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ ДЛЯ РОЗПІЗНАВАННЯ» має бути виконаний до 12.10.2024.

7.2 Розділ 2 «ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ» має бути виконаний до 30.10.2024.

7.3 Розділ 3 «РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ» має бути виконаний до 12.11.2024.

7.4 Економічний розділ має бути виконаний до 16.11.2024.

8. Порядок контролю і приймання.

8.1 Рубіжний контроль провести до 15.11.2024.

8.2 Попередній захист магістерської кваліфікаційної роботи провести до 5.12.2024.

8.3 Захист магістерської кваліфікаційної роботи провести до 20.12.2024.

Розробив студент групи 1АКІТР-23м _____ Ігор СЛОБОДЯН

Додаток Б

(обов’язковий).

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

Система розпізнавання об'єктів дорожнього руху

Студент групи 1АКІТР-23м

_____Ігор СЛОБОДЯН

Керівник к.т.н., доцент каф. АІІТ

_____ Володимир КОЦЮБИНСЬКИЙ

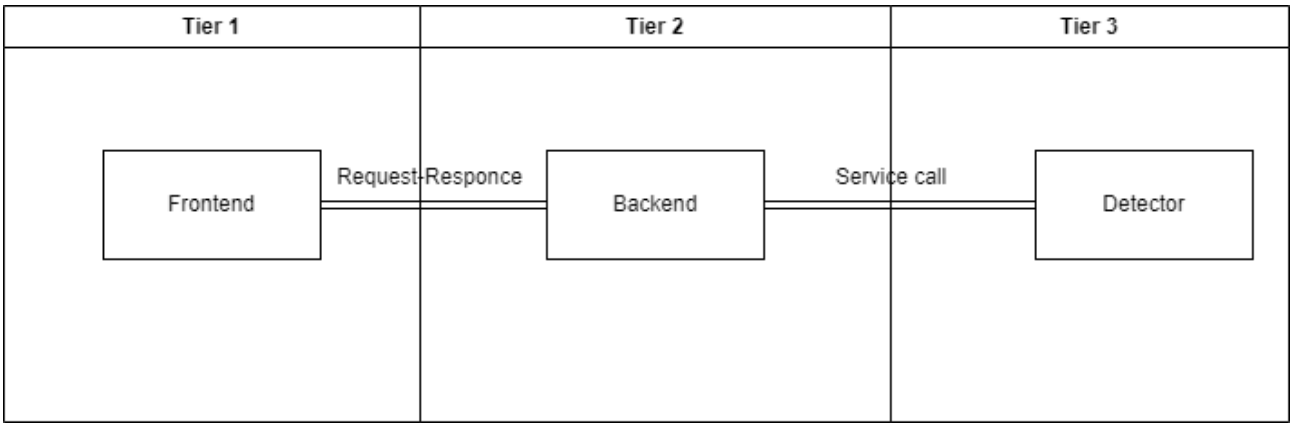


Рисунок Б.1 – Архітектура системи

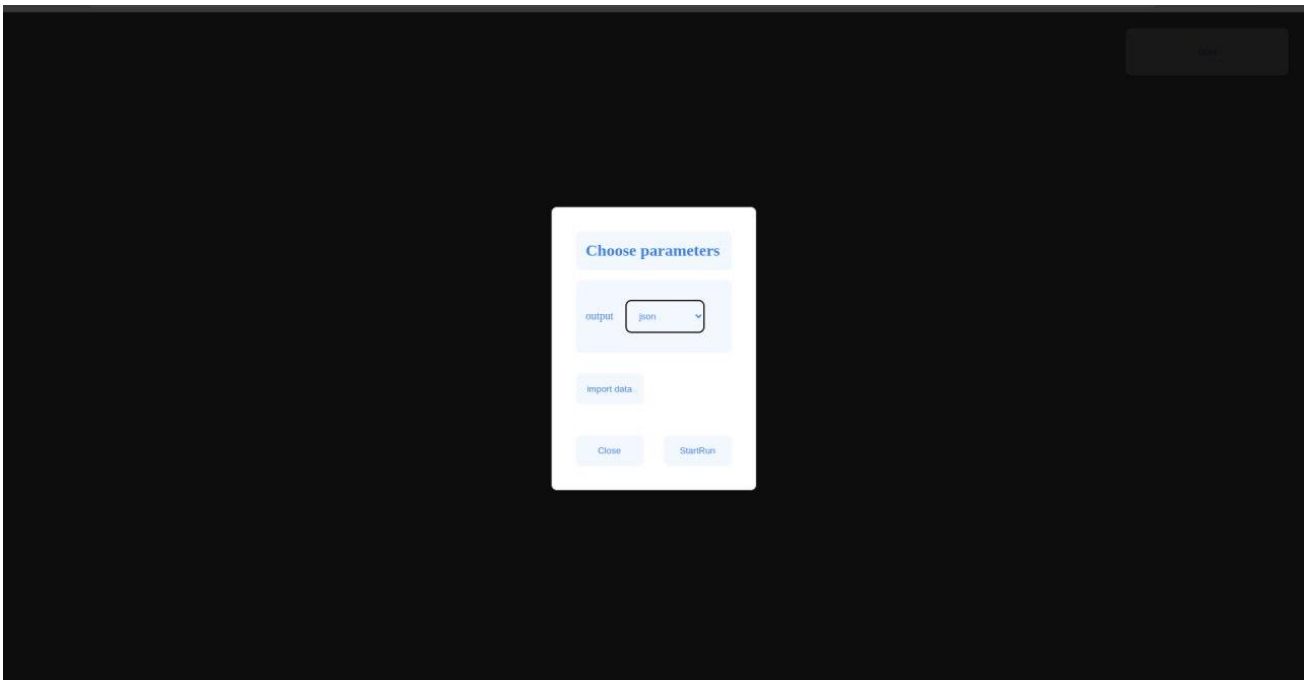


Рисунок Б.2– Меню запуску розпізнавання набору даних

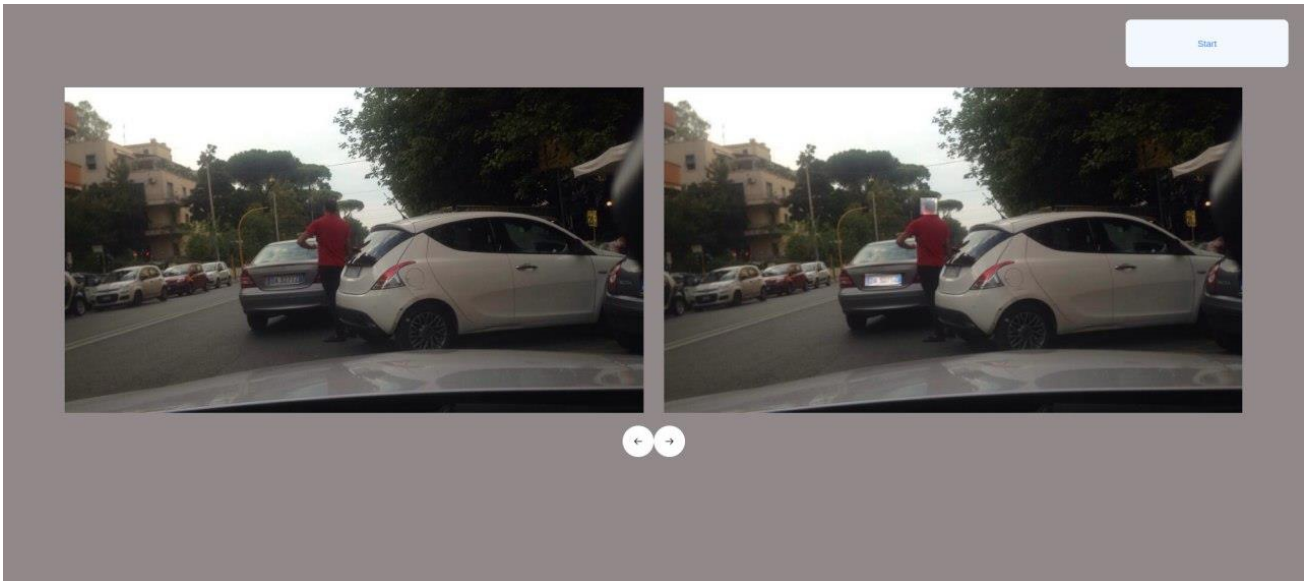


Рисунок Б.3– Меню Порівняльної характеристики після запуску програми

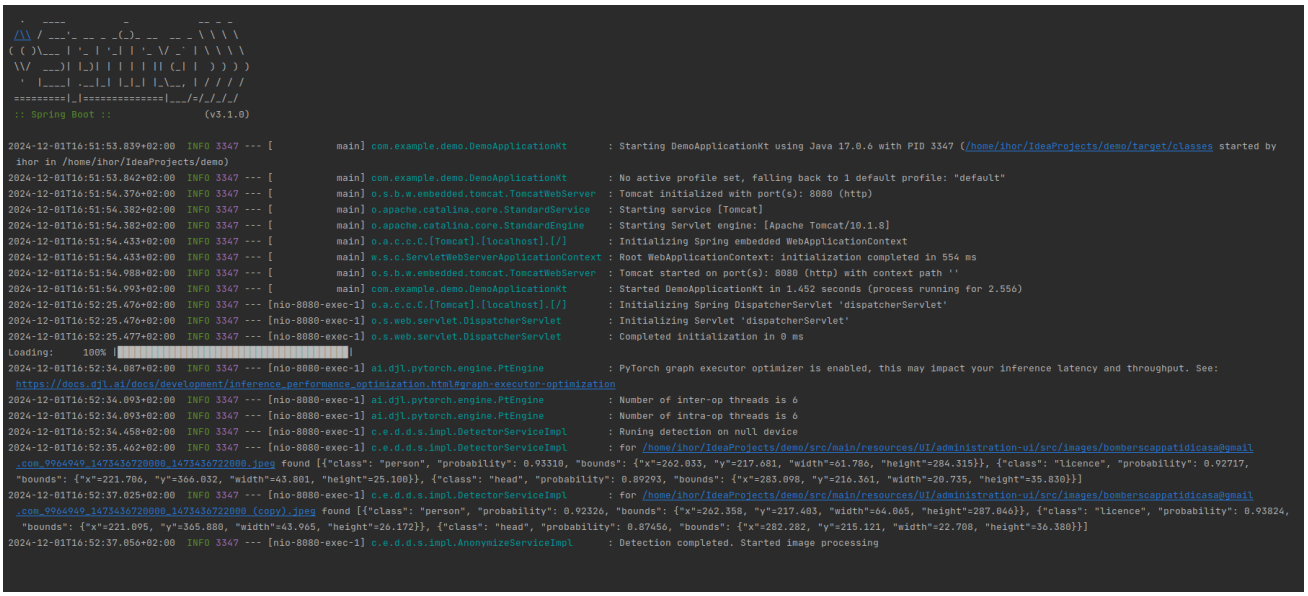


Рисунок Б.4– інформаційне повідомлення про успішне завантаження моделі та виявлення об’єкту.

Додаток В (обов’язковий)

Лістинг програми

```
const style = {
  height: "480px"
}
const Workspace = () => {

  function left() {
    // ControllClient.get()
  }

  function right() {
    ControllClient.get()
  }

  const [imageD, setImage] = useState('')

  var origImage = require(data[index])
  var blured =require(data[index])
  return (
    <div className='workspace'>
      <div className='imgSpace'>
        <div id='blocks'><img style={style} src={origImage} alt="My logo" /></div>
        <div id='blocks'><img style={style} src={blured} alt="My logo" /></div>
      </div>
      <div className='buttonsspace'>
```

```
            <RxArrowLeft id='workspace-buttons' onClick={index -
1}></RxArrowLeft>
            <RxArrowRight                                    id='workspace-buttons'
onClick={index+1}></RxArrowRight>
          </div>
        </div>
      )
    }

    export default Workspace

    ProcessConfig
  @Bean
    fun criteria(): Criteria<Image, DetectedObjects> {
      val pipeline = Pipeline()
      pipeline.add(Resize(scaleSize))
      pipeline.add(ToTensor())
      val translator: ObjectDetectionTranslator = YoloV5Translator
        .builder()
        .setPipeline(pipeline)
        .optThreshold(0.8f)
        .optSynsetArtifactName("anonymize_model/synset.txt")
        .build()
      return Criteria.builder()
        .setTypes(Image::class.java, DetectedObjects::class.java)
        .optProgress(ProgressBar())
        .optModelUrls("/home/ihor/IdeaProjects/demo/src/main/resources/")
        .optModelName("anonymize_model/anonymize.torchscript")
        .optEngine("PyTorch")
        .optTranslator(translator)
```

```
            .build()
        }
```

CriteriaFactory

```
package org.example

import ai.djl.repository.zoo.Criteria

class CriteriaFactory(vararg criterias: Criteria<Any, Any>) {

    private val models: Map<String,Criteria<Any,Any>> = criterias.associateBy
{ criteria -> criteria.modelName }

    fun getModel(name:String): Criteria<Any, Any>? {
        return models[name]
    }
}
```

Detector

```
@Component
class Detector(){

    private val log = LoggerFactory.getLogger(javaClass)

    fun detect(images: Set<Path>, criteria: Criteria<Any, Any>): Map<Path,
DetectedObjects> {
        val imageFactory = ImageFactory.getInstance()
        val model = ModelZoo.loadModel(criteria)
        val predictor = model.newPredictor()
        val map: MutableMap<Path, DetectedObjects> = images.stream()
```

```
            .collect(Collectors.toMap({ it }, {
                predictor.predict(imageFactory.fromFile(it))
            }))
        model.close()
        log.info("Model successfully loaded, started prediction!")
        return map
    }
```

AnonymizeService

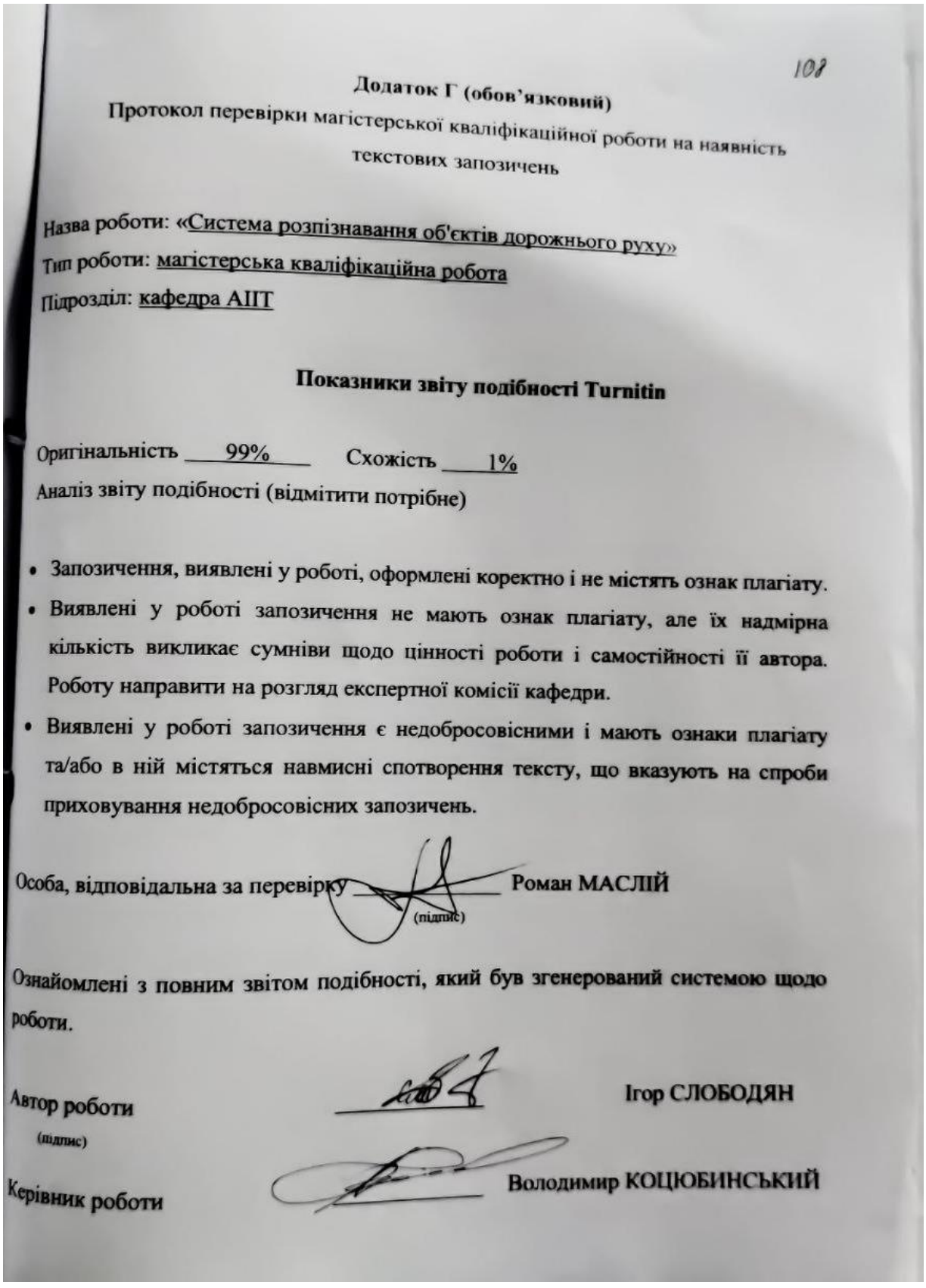
```
@Service
class AnonymizeService(
    private val detector: Detector,
    private val gaussianBlur: GaussianBlur,
    private val executorService: ExecutorService,
    private val criteriaFactory: CriteriaFactory,
    private val processFactory: ProcessFactory
) {
    private val log = LoggerFactory.getLogger(javaClass)

    fun processData(inputDir: File, outputDir: File, type: ProvessType,
modelName: String, vararg labels: Labels): Int {
        val model = criteriaFactory.getModel(modelName)
        val process = processFactory.getProcess(type)
        var processedImages = 0
        val images: Set<Path> = inputDir.listFiles()!!.asList().stream()
            .map { it.toPath() }.collect(Collectors.toSet())
        val map: Map<Path, DetectedObjects> = detector.detect(images, model)
        log.info("Detection completed. Started image processing")
        ListUtils.partition(map.keys.stream().toList(),
Runtime.getRuntime().availableProcessors())
```

```
            .forEach {
                CompletableFuture.supplyAsync({
                    process.process(map, it, outputDir, *labels)
                }, executorService)
                    .thenAccept { processedImages += it }
            }
        return processedImages
    }
```

Додаток Г (обов'язковий)

Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень



108

Додаток Г (обов'язковий)

Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Система розпізнавання об'єктів дорожнього руху»
Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра АІІТ

Показники звіту подібності Turnitin

Оригінальність 99% Схожість 1%
Аналіз звіту подібності (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою щодо роботи.

Автор роботи _____ Ігор СЛОБОДЯН
(підпис)

Керівник роботи _____ Володимир КОЦЮБИНСЬКИЙ

Додаток Д

Акт впровадження результатів магістерської роботи

Науково-виробниче підприємство
“СПІЛЬНА СПРАВА” Товариство з обмеженою відповідальністю
Україна, 21000, м. Вінниця, вул. Магістратська, 36/3, тел. +38(096)-558-24-64
код ЄДРПОУ 31041670, код платників податків 310416702282, свідоцтво № 0183329
IBAN : UA 41 322313 0000026004000003226 в філії АТ «Укрексімбанк» в м. Вінниці

Затверджую

Директор

НВП «СПІЛЬНА СПРАВА», ТОВ

Малачковська Роксолана Ігорівна

«01» грудня 2021 р.

АКТ

впровадження результатів магістерської роботи

Слободяна Ігоря Олександровича «Система розпізнавання об’єктів дорожнього руху»

Комісія у складі головного спеціаліста, керівника відділу обробки даних С. Житанського та керівника відділу розробки інформаційних систем, О. Кириленка склали цей акт про те, що у Науково-виробничому підприємстві “Спільна Справа”, ТОВ впроваджуються результати, які отримані магістром Слободяном І. О. при виконанні магістерської кваліфікаційної роботи мають практичну цінність. Технології машинного навчання сприяють підвищенню ефективності процесу розпізнавання об’єктів, що дозволяє економити ресурси, прискорити процес та дає ширшу можливість для його масштабування.

Таким чином, актуальність роботи І. О. Слободяна не викликає сумнівів, а низка методик та підходів та результати їх застосування можуть бути застосованими у практичній діяльності.

Науково-виробничому підприємству “Спільна Справа”, ТОВ магістром Слободяном І. О. передано програмне забезпечення, яке дозволяє підвищити ефективність процесу розпізнавання об’єктів дорожнього руху.

Члени комісії:

Головний спеціаліст. Сергій ЖИТАНСЬКИЙ

Керівник відділу Олександр КИРИЛЕНКО