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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

«На правах рукопису»
УДК 004.4

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«___» _____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»
зі спеціальності 121 Інженерія програмного забезпечення
на тему «Інструментальні засоби аналізу якості дорожнього покриття»

Виконав: студент 2 курсу, групи ТВ-22мп
Чурчин Денис Андрійович _____
(прізвище, ім'я, по батькові) (підпис)
Науковий керівник доцент, к.е.н. Гусєва І.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н. Степанець О.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
(підпис)
«___» _____ 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Чурчину Денису Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Інструментальні засоби аналізу якості дорожнього покриття
Науковий керівник Гусєва Ірина Ігорівна, к.е.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «06» листопада 2023 року № 5152-с
2. Строк подання студентом дисертації 10 січня 2024 року
3. Об'єкт дослідження аналіз якості дорожнього покриття
4. Предмет дослідження програмне забезпечення для аналізу якості дорожнього покриття на основі обробки медіа даних
5. Перелік питань, які потрібно розробити
- розробити та навчити нейромережеву модель для виявлення дефектів дорожнього покриття;
- створити вебзастосунок для завантаження та аналізу медіа файлів;
- розробити алгоритм для виконання асоціації пошкодження дорожнього покриття з їх точною геолокацією;
- розробити програмний інтерфейс, що надає доступ до всіх функціональних можливостей вебзастосунку;
- виконати тестування системи з використанням реальних фото та відео.

6. Орієнтований перелік ілюстративного матеріалу архітектура модулів системи та внутрішньої взаємодії сервісів, алгоритм відстеження об’єктів, розпізнаних моделлю на основі комп’ютерного зору, алгоритм асоціації пошкоджень з географічним положенням на основі файлу з дискретними даними, інтерфейс програмного забезпечення.
7. Дата видачі завдання «01» листопада 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.11.2022	виконано
2	Дослідження предметної області	01.11.2022 – 01.12.2022	виконано
3	Постановка вимог до проектування системи	01.12.2022 – 15.12.2022	виконано
4	Дослідження існуючих рішень	15.12.2022 – 03.01.2023	виконано
5	Розробка програмного продукту	03.03.2023 – 20.09.2023	виконано
6	Тестування	20.09.2023 – 15.10.2023	виконано
7	Захист програмного продукту	16.10-2023 – 20.10.2023	виконано
8	Підготовка магістерської дисертації	21.10.2023 – 17.11.2023	виконано
9	Передзахист	18.12.2023 – 22.12.2023	виконано
10	Захист	15.01.2024 – 19.01.2024	виконано

Студент _____ Чурчин Д.А.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Гусєва І.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Структура й обсяг дипломної роботи. Магістерська дисертація складається зі вступу, п’яти розділів, висновків та додатків. Робота містить посилання на 26 джерел, 11 таблиць та 20 ілюстрацій. Основна частина викладена на 84 сторінках.

Актуальність. Від якості доріг залежить швидкість та безпека руху, оскільки розбиті дороги сповільнюють рух та пошкоджують автомобілі. Через це важливо швидко виявляти пошкодження та виконувати ремонт. Наразі ремонтом займаються Державне агентство автомобільних доріг України (Укравтодор) або органи місцевого самоврядування, а перевірка стану виконується візуально працівниками та через систему скарг від громадян. Ці методи є недостатньо ефективними, тож можуть бути покращені за допомогою автоматизації.

Метою роботи є створення інструментальних засобів на основі машинного навчання для ефективного вирішення проблеми комплексного аналізу якості дорожнього покриття.

Для досягнення поставленої мети необхідно виконати наступні **завдання:**

- розробити та навчити нейромережеву модель, спеціально оптимізовану для точного виявлення дефектів дорожнього покриття;
- створити вебзастосунок, який слугує інтерактивною платформою для завантаження та аналізу медіа файлів, забезпечуючи інтеграцію розробленої моделі в комплексну систему;
- розробити алгоритм для виконання асоціації пошкодження дорожнього покриття з їх точною геолокацією для нанесення на карту, використовуючи інформацію з медіа файлів або на основі даних з відповідних GPX-файлів;
- розробити програмний інтерфейс для доступу до всіх функціональних можливостей вебзастосунку, дозволяючи ширше застосовувати та впроваджувати систему, забезпечуючи її масштабованість та доступність;

- виконати тестування системи з використанням реальних відео та фото.

Об’єктом дослідження є аналіз якості дорожнього покриття.

Предметом дослідження є програмне забезпечення для аналізу якості дорожнього покриття на основі обробки медіа даних.

Методи дослідження: теоретичні, як аналіз та узагальнення, і емпіричні, як експериментування та вимірювання.

Практичне значення одержаних результатів полягає в розроблених алгоритмах автоматизованого відстеження руху об’єктів, розпізнаних моделлю на основі ком’ютерного зору, та алгоритмі поєднання об’єктів знайдених на відео, з географічним положенням, знайденим у файлі з дискретними точками.

Ключові слова: інструментальні засоби, дорожнє покриття, ями, комп’ютерний зір, штучний інтелект, аналіз якості, цифрова обробка зображення, цифрова обробка відео.

ABSTRACT

Structure and scope of the thesis. The master's thesis consists of an introduction, five chapters, conclusions, and appendices. The work contains references to 26 sources, 11 tables and 20 illustrations. The main part is presented on 84 pages.

Relevance of topic. The quality of roads affects the speed and safety of traffic, as broken roads slow down traffic and damage cars. It is therefore important to quickly identify damage and carry out repairs. Currently, repairs are carried out by the State Road Agency of Ukraine or local governments, and the condition is checked visually by employees and through a system of complaints from citizens. These methods are not efficient enough and can be improved by automation.

The goal of the research is to create machine learning-based tools to effectively solve the problem of comprehensive analysis of road surface quality.

To achieve this goal, the following **objectives** need to be achieved:

- to develop and train a neural network model specifically optimised for the accurate detection of road surface defects;
- to create a web application that serves as an interactive platform for uploading and analysing media files, ensuring the integration of the developed model into a comprehensive system;
- associate pavement damages with their exact geolocation for mapping, using information from media files or based on data from the corresponding GPX files;
- to develop an API that provides a software interface to all the functionality of the web application, allowing for wider use and implementation of the system, ensuring its scalability and accessibility;
- to test the system using real video and image data.

The object of the study is the analysis of road surface quality.

The subject of the study is software for analysing road surface quality based on media data processing.

Research methods: theoretical, such as analysis and generalisation, and empirical, such as experimentation and measurement.

The practical value of the results lies in the developed algorithms for automated tracking of the movement of objects recognized by a model based on computer vision, and an algorithm for combining objects found on video with the geographical position found in a file with discrete points.

Keywords: tools, road surface, potholes, computer vision, artificial intelligence, quality analysis, digital image processing, digital video processing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface, набір правил та протоколів, що дозволяє різним програмам взаємодіяти між собою;

ШІ – Штучний Інтелект (Artificial Intelligence), галузь комп'ютерних наук, яка зосереджена на створенні систем, здатних виконувати завдання, які зазвичай вимагають людського інтелекту.

ВСТУП

Важливість своєчасного ремонту дорожнього покриття не може бути переоцінена, особливо з огляду на статистику, яка підкреслює вплив якості доріг на безпеку дорожнього руху та економіку. Ефективний та своєчасний ремонт доріг не лише знижує ризики для водіїв та пішоходів, але й сприяє поліпшенню економічних показників, зменшуючи збитки від аварій та покращуючи транспортну доступність.

Очевидна необхідність у якісному аналізі дорожнього покриття задля його своєчасного ремонту ставить під сумнів ефективність традиційних методів перевірки, які переважно виконуються вручну Укравтодором або органами місцевого самоврядування. Ці методи, хоча й надійні, часто є трудомістким та не завжди точними.

Актуальність розробки нових програмних засобів для автоматизованого аналізу якості дорожнього покриття виходить з потреби модернізації існуючих процесів. Використання автоматизованих систем дозволить збільшити точність виявлення дефектів, оптимізувати процеси планування ремонтних робіт, знизити витрати та підвищити загальну безпеку дорожнього руху.

Впровадження таких технологій призведе до значних позитивних змін. Дорожні служби зможуть швидко реагувати на проблемні ділянки, що зменшить частоту дорожньо-транспортних пригод, покращить загальний стан інфраструктури та підвищить задоволеність громадян.

Запропонована система базується на застосуванні штучного інтелекту та алгоритмів комп'ютерного зору для аналізу фото та відео дорожнього покриття. Вона включає автоматизоване виявлення, класифікацію та геолокацію дефектів, інтегровану обробку даних, та надає інтерфейс для взаємодії з користувачами. Це дозволяє значно поліпшити процеси ідентифікації та ремонту дорожнього покриття.

У першому розділі описана постановка завдання аналізу якості дорожнього покриття, вимоги до системи, її призначення, задачі, що вона вирішує та необхідні алгоритми.

У другому розділі проведено аналіз предметної області, описано підходи до аналізу дорожнього покриття та проведено аналіз існуючих аналогів, визначено параметри для оцінювання дорожнього покриття.

У третьому розділі описуються функціональні вимоги до системи, а також розглядаються обрані засоби розробки та обґрунтовується їх застосування під час розробки програмного продукту.

У четвертому розділі описується програмна реалізація, структура системи, компоненти та способи зберігання даних, інтерфейс користувача. Також описуються реалізовані алгоритми забезпечення унікальності виявлених ям та географічного позиціювання.

У п’ятому розділі описано системні вимоги для запуску продукту та продемонстровано роботу користувача з системою.

У шостому розділі наведено опис стартап-проєкту, враховуючи плани розвитку та економічні показники.

У висновках описані результати з розробки програмного забезпечення, досягнені цілі, та можливості подальшого розвитку.

1 ПОСТАНОВКА ЗАВДАННЯ АНАЛІЗУ ЯКОСТІ ДОРОЖНЬОГО ПОКРИТТЯ

1.1 Мета роботи та постановка задач

Метою роботи є створення інструментальних засобів на основі машинного навчання для ефективного вирішення проблеми комплексного аналізу якості дорожнього покриття.

Основні завдання, які необхідно вирішити наведено далі.

1. Розробити та навчити нейромережеву модель на основі YOLOv8.
2. Розробити вебзастосунок для завантаження та аналізу медіа файлів, забезпечуючи інтеграцію розробленої моделі в комплексну систему.
3. Розробити алгоритм для асоціювання пошкодження дорожнього покриття з їх точною геолокацією для нанесення на карту.
4. Розробити програмний інтерфейс для доступу до всіх функціональних можливостей вебзастосунку.
5. Виконати тестування системи на реальних відео та зображеннях.

Таким чином, цілі дослідження спрямовані на подолання обмежень існуючих методів аналізу дорожнього покриття шляхом застосування технологій машинного навчання та методів програмної інженерії.

Це дослідження має на меті зробити внесок у цю важливу предметну область, пропонуючи комплексне та інтегроване рішення для оцінювання якості дорожнього покриття, зокрема, зосереджуючись на проблемах масштабованості, точності та ідентифікації дефектів у реальному часі.

З огляду на вражаючу статистику смертельних випадків та економічних збитків, спричинених незадовільним станом доріг, своєчасність та актуальність таких досліджень важко переоцінити.

Вхідні дані складаються із зображень дорожнього покриття або відео. В свою чергу вихідні дані включають оброблені зображення, де виділено вибоїни,

інформацію про геолокацію цих вибоїн, а також інтерактивну карту, що відображає цю інформацію.

1.2 Призначення системи

Розроблене програмне забезпечення, яке має на меті ефективний аналіз стану дорожнього покриття на великих площах, є важливим інструментом для різноманітних секторів і груп користувачів:

- органи державної влади: використання цієї технології може значно підвищити ефективність процесів технічного обслуговування доріг. Це не лише сприяє зниженню витрат, а й дозволяє органам влади більш стратегічно підходити до планування ремонтних робіт, забезпечуючи вищу безпеку та комфорт дорожнього руху;
- інженери-будівельники та містобудівники: професіонали в цій сфері можуть використовувати аналітичні дані для оптимізації планування міських просторів і дорожньої інфраструктури. Ця інформація може слугувати основою для розробки більш ефективних стратегій ремонту та обслуговування доріг;
- науково-дослідницька спільнота: ця технологія може бути використана як база для досліджень у сфері дорожньої інфраструктури. Вона надає можливість для розробки нових моделей і підходів в аналізі дорожнього покриття, сприяючи науковому прогресу в цій галузі;
- широка громадськість: учасники дорожнього руху отримують значну користь від підвищення безпеки і якості доріг. Зменшення кількості аварійних ситуацій і покращення загального стану дорожнього покриття сприятимуть безпеці та комфорту всіх учасників дорожнього руху.

З точки зору застосування, розроблена система має потенційну корисність у різних секторах, зокрема:

- обслуговування інфраструктури задля регулярного моніторингу та оперативного усунення дефектів доріг;
- міське планування задля допомоги у розподілі ресурсів на розвиток та утримання доріг.

Таким чином, розроблене програмне забезпечення виступає як важливий інструмент для поліпшення управління дорожньою інфраструктурою, сприяючи безпеці, ефективності та сталому розвитку в різних сферах господарської діяльності.

1.3 Реалізація алгоритму відстеження пошкоджень

Розробка моделі штучного інтелекту для виявлення та відстеження пошкоджень на дорогах вимагає виконання кількох ключових завдань. Перш за все, необхідно зібрати достатній набір даних, який включатиме зображення дорожнього покриття з різними видами пошкоджень. Ці дані мають бути різноманітними, щоб відображати широкий спектр умов, таких як різні освітлення, кути зйомки та типи дорожнього покриття.

Наступним кроком є тренування моделі штучного інтелекту, яка зможе ефективно виявляти ями на зображеннях. Така модель повинна бути достатньо точною, щоб розпізнавати ями різних розмірів та форм у різних умовах. Однак, оскільки модель буде обробляти окремі кадри з відео, виникає додаткова проблема: вона не зможе розпізнати, чи яма вже була виявлена на попередніх кадрах.

Для вирішення цієї проблеми потрібно розробити алгоритм, який здатен відстежувати переміщення вже виявлених ям між послідовними кадрами відео. Це вимагатиме застосування методів обробки зображень і аналізу відео, щоб визначити, чи яма, що з’являється в поточному кадрі, є новою, чи це та сама яма, що вже була виявлена раніше. Такий підхід дозволить уникнути дублювання виявлених ям і забезпечить точне відстеження унікальних дефектів дорожнього покриття протягом всього відео.

Завершальним етапом є інтеграція цього алгоритму відстеження з основною моделлю розпізнавання ям.

1.4 Реалізація алгоритму визначення географічного положення пошкоджень

Іншою задачею при створенні системи є точне визначення географічного положення виявлених ям. Для цього необхідно розробити алгоритм, який зможе інтегрувати дані, отримані з відеозапису, з географічним трекінгом. Цей процес починається з додавання до відео файлу трекінгу, який містить інформацію про точки, через які проходив оператор під час зйомки. Ці точки фіксуються з певною частотою і являють собою дискретні дані про місцеположення оператора у певний момент часу.

Основна складність полягає в знаходженні проміжних точок. Коли відома точка, що співпадає з часом виявлення ями, її можна безпосередньо використовувати як географічне положення ями. Проте, у випадках, коли час виявлення ями не збігається з часом фіксації точки у файлі трекінгу, потрібно визначити проміжну точку. Для цього алгоритм аналізує параметри руху, такі як середня швидкість між відомими точками, та враховує можливу похибку, щоб точно розрахувати географічне положення ями.

Цей алгоритм дозволяє з точністю визначити географічне положення кожної виявленої ями, інтегруючи дані про час та місцеположення з відеозапису. Такий підхід забезпечує високу точність відображення даних на географічній карті та є важливим для планування ремонтних робіт та аналізу стану дорожнього покриття.

1.5 Взаємодія користувача та інтеграція системи

Розроблена система аналізу якості дорожнього покриття представляє собою комплексне рішення, яке включає декілька ключових компонентів: модель штучного інтелекту, сховище даних, модуль для аналізу зображень та модуль для керування даними про ями. Модель ШІ призначена для виявлення та класифікації ям на зображеннях доріг. Сховище даних забезпечує зберігання інформації про виявлені ями, включаючи їхні характеристики та географічне положення. Модуль аналізу зображень відповідає за обробку вхідних зображень або відео, а модуль керування даними дозволяє користувачам управляти зібраною інформацією, включаючи перегляд, редагування та аналіз даних про ями.

Що стосується взаємодії з користувачами, система пропонує два основних підходи: через вебінтерфейс та програмний інтерфейс. Вебінтерфейс дозволяє користувачам взаємодіяти з системою через зручний та інтуїтивно зрозумілий графічний інтерфейс, який може бути використаний для завантаження зображень або відео, перегляду результатів аналізу та керування даними. Програмний інтерфейс надає більше можливостей для автоматизації та інтеграції системи з іншими програмними рішеннями, дозволяючи користувачам використовувати функціонал системи у своїх застосунках та сервісах.

Щодо вимог до функціональності програми, вони включають високу точність та надійність моделі ШІ у виявленні та класифікації ям, забезпечення безпечного та ефективного зберігання даних, зручний та доступний інтерфейс для користувачів, а також гнучкість та широкий функціонал програмного інтерфейсу для інтеграції з різними програмними середовищами. Важливою є також підтримка прозорості та інтерактивності у процесі відображення результатів, що дозволяє користувачам легко інтерпретувати та використовувати отримані дані.

Висновки до розділу 1

Отже, в цьому розділі було досліджено задачу створення інструментальних засобів для аналізу якості дорожнього покриття. Було визначено задачі для вирішення та поставлені наступні завдання:

- розробити та навчити нейромережеву модель на основі YOLOv8 для виявлення дефектів дорожнього покриття;
- створити вебзастосунок для завантаження та аналізу медіа файлів;
- асоціювати пошкодження дорожнього покриття з їх точною геолокацією;
- розробити програмний інтерфейс з доступом до всіх функціональних можливостей системи;
- виконати тестування системи з використанням реальних фото та відео.

Метою роботи визначено створення інструментальних засобів на основі машинного навчання для ефективного вирішення проблеми комплексного аналізу якості дорожнього покриття.

2 АНАЛІЗ ЛІТЕРАТУРИ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Підходи до аналізу якості дорожнього покриття

Аналіз якості дорожнього покриття є багатогранним процесом, що включає в себе різні методи, параметри та інструменти. Зазвичай метою є виявлення дефектів та зношеності поверхні. Підходи можна умовно поділити на ручні методи обстеження та автоматизовані методи, кожен з яких має свій власний набір інструментів та параметрів [1].

Традиційні підходи здебільшого ґрунтуються на ручному обстеженні, коли інженери на місцях оцінюють стан дорожнього покриття та класифікують його за ступенем тяжкості та типом дефектів. Раніше ці методи оцінки були схильними до помилок, їм бракувало стандартизації вимірювань і протоколів. Зі зростанням складності дорожніх матеріалів і технологій, виникли стандартизовані підходи, що дозволяють об’єктивно виміряти стан дорожнього покриття.

Індекс стану дорожнього покриття (PCI) розроблений Американським товариством з випробувань і матеріалів (ASTM), є стандартним методом вимірювання стану дорожнього покриття. Він передбачає детальний візуальний огляд з метою виявлення певних типів пошкоджень покриття. Пошкодження оцінюються кількісно, і виставляється оцінка PCI від 0 (незадовільно) до 100 (відмінно). Параметри для оцінювання можуть відрізнятися та залежать від типів дорожніх покриттів.

Для асфальтових доріг PCI бере до уваги такі типи пошкоджень, як тріщини (поздовжні, поперечні, алігаторні), колії, вибоїни, відшарування верхнього шару, зміни текстури поверхні, і ознаки старіння матеріалу. Оцінка також враховує ступінь водовідведення та загальний візуальний стан покриття [2].

Для бетонних доріг PCI аналізує тріщини (поздовжні, поперечні), вибоїни, руйнування швів, відшарування поверхні, а також деформації, такі як просадки та

нерівності на стиках плит. Важливість надається також цілісності швів та стикувальних зон [2].

Для гравійних доріг PCI фокусується на оцінці таких аспектів, як ступінь ущільнення, наявність вибоїн та борозен, розподіл і якість гравійного покриття. Важливим фактором є збереження рівномірності та щільності гравію на всій поверхні [3].

Для ґрунтових доріг PCI включає оцінку наявності колій, вибоїн, рівня ерозії, утримання води та загального ступеня ущільнення ґрунту. Також оцінюється здатність дороги витримувати погодні впливи та забезпечувати безпечний проїзд в різних погодних умовах [4].

З подальшим розвитком технологій підхід до аналізу дорожнього покриття значно змінився. Традиційні методи були доповнені, а іноді й повністю замінені сенсорними системами та обчислювальними моделями, пропонуючи більш науковий спосіб оцінки. Найсучасніші рішення використовують алгоритми штучного інтелекту та машинного навчання, які здатні в режимі реального часу з високою точністю виявляти дефекти дорожнього покриття, такі як вибоїни та тріщини. Однак інтеграція та масштабованість цих рішень все ще залишаються предметом постійних досліджень і розробок.

Лазерне та інфрачервоне сканування – цей підхід використовує лазерні сканери та інфрачервоні технології для виявлення нерівностей дорожнього покриття. Лазерні сканери надають 3D-зображення дороги з високою роздільною здатністю, виявляючи навіть незначні дефекти. Інфрачервона технологія дозволяє виявити температурні коливання, які свідчать про різні дорожні умови.

Георадар використовується для оцінки стану підземних шарів дороги. Він посилає електромагнітні хвилі в дорожнє покриття і вимірює відбиті сигнали. Цей метод є цінним для виявлення прихованих проблем, таких як утримання вологи та підповерхневі порожнечі, які у поєднанні з природним руйнуванням є основними факторами, що впливають на знос доріг. Порожнечі, якщо не піддаються ремонту

одразу, збільшуються за обсягом та, з рештою можуть спричинити руйнування основи, що значним чином підвищує складність та вартість ремонту [5].

Високошвидкісні транспортні засоби для збору даних: оснащені безліччю сенсорів, камер та обладнанням для лазерного сканування, ці транспортні засоби можуть швидко і точно збирати дані про стан дорожнього покриття під час руху зі звичайною швидкістю руху.

Розглянемо параметри, що вимірюються при такому способі оцінки якості дорожнього покриття. Першим є шорсткість поверхні – вимірюється в Міжнародному індексі шорсткості (IRI), цей параметр відображає гладкість дороги. Високі значення IRI вказують на нерівну поверхню, що може вплинути на якість їзди та знос транспортного засобу [6].

Тріщини – аналізується довжина, тип та глибина тріщин, що є важливими індикаторами зношування покриття.

Колійність та деформації – визначаються за глибиною утворених колій. Підвищена колійність може призвести до застою води на дорозі та зниження стійкості транспортних засобів.

Опір ковзанню – важливий для безпеки руху, цей параметр вимірюється за допомогою спеціального обладнання і показує, наскільки добре дорожнє покриття забезпечує зчеплення шин з дорогою.

Глибина текстури – вимірюється для оцінки якості текстури поверхні, що впливає на відведення води з поверхні дороги та опір ковзанню.

Отже, аналіз якості дорожнього покриття – це складний процес, який включає поєднання традиційних методів і новітніх технологій. Вибір методу та інструменту залежить від різних факторів, включаючи необхідний рівень деталізації, наявні ресурси та конкретні дорожні умови.

2.2 Опис аналогічних рішень

Візуальне обстеження з застосуванням заздалегідь визначених критеріїв оцінки залишається найбільш простим, але найменш ефективним рішенням проблеми аналізу якості доріг.

Це рішення найширше використовується в Україні організацією «Державне агентство автомобільних доріг України», що відповідає за управління та обслуговування автомобільних доріг державного значення в Україні. Організація має гілки в різних регіонах країни, що дозволяє забезпечувати оперативне управління та контроль за станом доріг на місцевому рівні.

«Укравтодор» виконує регулярні перевірки: дорожні служби проводять обходи та інспекції доріг, виявляючи та фіксуючи будь-які пошкодження. А також отримує звернення від громадян, які можуть повідомляти про проблеми на дорогах через спеціальні сервіси, такі як гарячі лінії або вебплатформи. «Укравтодор» також співпрацює з місцевими адміністраціями та комунальними службами для виявлення та усунення проблем на дорогах.

Основний недолік рішення даної організації – складність швидкого отримання інформації та передчасного виявлення пошкоджень, адже використання ручного методу є трудомістким та робить масштабування дорогим.

Використовуючи схожий підхід, частину проблем вирішує підсистема для попередження інших водіїв застосунку Waze. Waze є популярним застосунком для навігації, що використовує дані від користувачів для створення детальної карти дорожніх умов.

Цей застосунок дозволяє користувачам активно повідомляти про різні проблеми на дорозі, включаючи ями, вибоїни та інші дефекти покриття, які вони могли помітити [7]. Перевірка достовірності інформації виконується іншими користувачами системи, які можуть так само відмітити пошкодження та підтвердити або спростувати його існування як показано на рисунку 2.1.

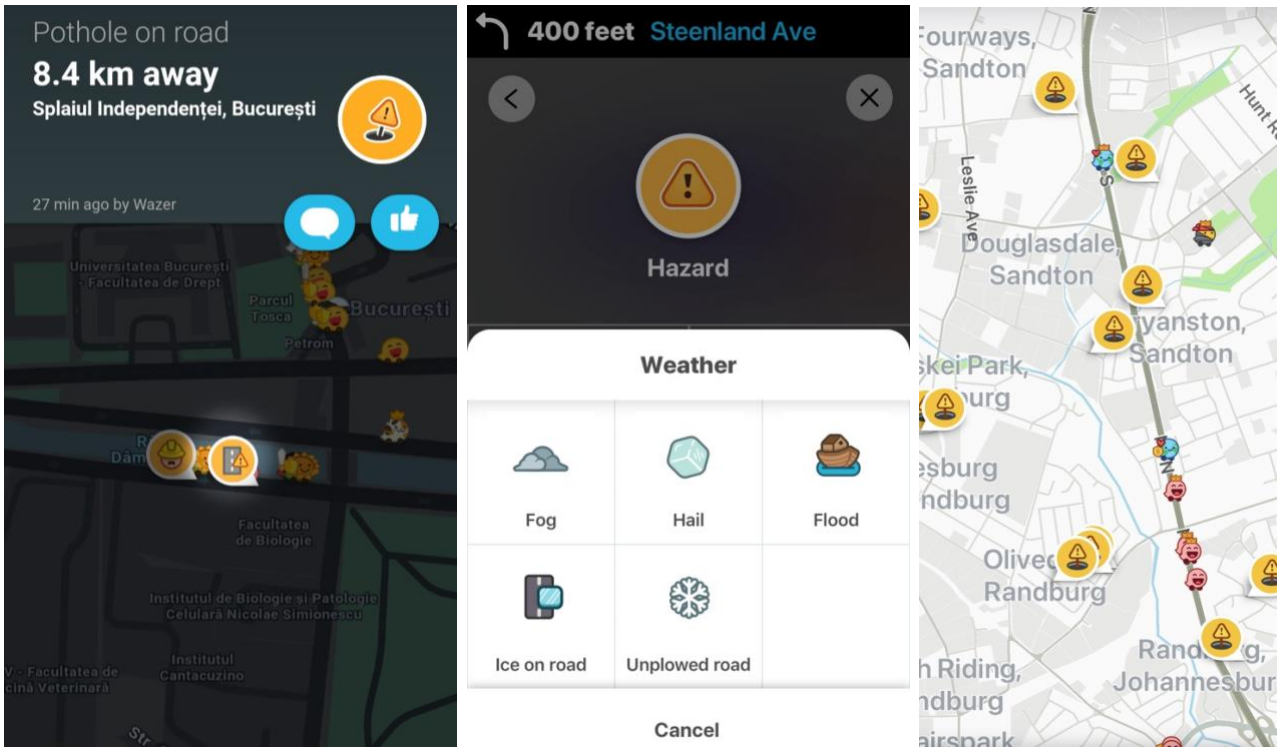


Рисунок 2.1 – Екрани мобільного застосунку Waze для створення та перегляду попереджень від користувачів системи про яму на дорозі

Переваги використання Waze для перевірки якості дорожнього покриття включають швидкість отримання інформації та широку поширеність застосунку. Завдяки краудсорсингу, Waze здатний оперативно збирати дані про стан доріг від великої кількості користувачів, що дозволяє швидко реагувати на появу нових дефектів. Поширеність застосунку означає, що велика кількість водіїв постійно доповнює базу даних, забезпечуючи актуальну інформацію.

Однак, Waze має і недоліки. Інформація, надана користувачами, не надає деталей про форму та критичність пошкодження, що створює виклики для дорожніх служб у плануванні ремонтів. Дані з Waze часто потребують додаткової перевірки та не мають стандартизованого програмного інтерфейсу для інтеграції в офіційні системи, що використовують організації, що займаються ремонтом доріг.

Технологічний прогрес призвів до впровадження сенсорних систем для моніторингу стану доріг. Акселерометри, гіроскопи і навіть мікрофони використовуються для виявлення вібрацій або звуків, які відповідають дефектам

доріг. Хоча ці методи забезпечують вищу ефективність порівняно з ручним оглядом, вони вимагають спеціалізованого обладнання і не завжди можуть бути практичними для широкомасштабного використання.

Однак декілька досліджень вказують на обмеження таких систем. Зокрема, в дослідженні, проведеному Fox та ін. [8], було вказано, що для збору даних і досягнення точності в 90% потрібно принаймні 10 транспортних засобів, що рухаються зі швидкістю 50 км/год. Що стосується методу простого голосування, то, як показано в дослідженні Zhang та ін. [9], цей метод ігнорує те, що джерела мають різні ступені надійності. Крім того, цей алгоритм не враховує тимчасову і ймовірнісну природу аномалій, не маючи інформації про особливості параметрів дорожнього покриття в цілому, або особливостей сенсорів конкретного пристрою. Параметри, на які спирається такий метод описані на рисунку 2.2 [10]. Можна побачити три види параметрів, положення сенсору відносно поверхні землі, корпусу авто та пристрою, де він знаходиться. Кожне з середовищ надає власні похибки, що сумарно погіршує точність визначення якості покриття.

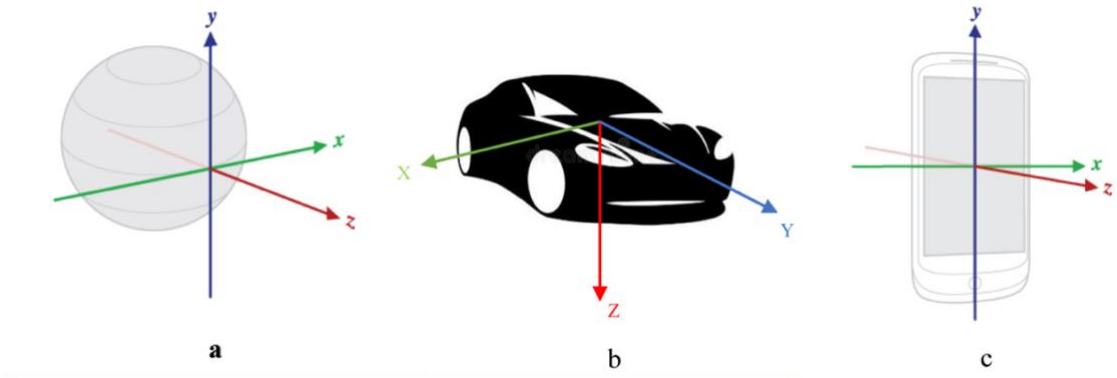


Рисунок 2.2 – Опис параметрів руху, що враховуються системами розпізнавання ям на основі сенсорів. a) – локальна система координат що відповідає за географічне розташування сенсора. b) – система координат положення сенсора відносно корпусу авто, де він розміщений. c) – особиста система координат сенсора

Сучасні технології штучного інтелекту відкривають шлях до автоматизованої обробки візуальних даних з фото та відеозаписів, адаптуючись до різноманітних

умов, таких як освітлення та кут зйомки. Одним із прикладів такої системи є CityROVER, яка використовує передові методи обробки зображень для виявлення дефектів дорожнього покриття [11]. Ця система включає вебзастосунок та спеціалізоване обладнання, яке можна встановити на автомобіль для збору відеоданих про стан доріг та їх подальшого аналізу на наявність пошкоджень, як показано на рисунку 2.3.

Рисунок 2.3 – Спеціалізоване обладнання для записування дороги та екран вебзастосунку CityROVER

Основною перевагою CityROVER є її швидкість та ефективність у виявленні дефектів дорожнього покриття, завдяки автоматизованому обладнанню, призначеному для роботи на транспортних засобах, що рухаються з великою швидкістю. Однак, використання такого рішення передбачає необхідність у застосуванні високотехнологічних та вартісних сенсорів.

Таким чином, існуючі автоматизовані засоби на основі транспорту надзвичайно ефективні, але занадто дорогі через спеціалізоване обладнання. Спеціалізоване обладнання може бути замінене звичайною камерою або використанням існуючих відео записів за умови використання натренованої моделі штучного інтелекту.

Найпоширенішим типом доріг є асфальтові, відповідно найкраще зосередити модель на розпізнаванні дефектів для цього покриття. Найпоширенішим прикладом пошкодження є яма, відповідно її можна визначити як ту, визначення якої є найефективнішим.

2.3 Алгоритми машинного навчання для аналізу якості дорожнього покриття

Розглянемо види алгоритмів машинного навчання, що можуть використовуватися для оцінювання загаданих параметрів поверхні, а також проведемо порівняння фундаментальних підходів у побудові моделей.

Останнім часом алгоритми машинного навчання та нейромережеві моделі показали значні перспективи в автоматизації завдання розпізнавання об’єктів на фото та відео. Для цього були представлені різні версії моделей згорткових нейронних мереж (CNN) та You Only Look Once (YOLO).

You Only Look Once (YOLO) – це найсучасніша модель для задач виявлення об’єктів, що може включати ідентифікацію дефектів на дорогах. Вона пропонує такі переваги, як обробка в реальному часі та висока точність. Хоча попередні версії, мали обмеження в обробці об’єктів малих розмірів, що часто зустрічається на зображеннях дорожніх дефектів. Останні її реалізації, усувають деякі з цих обмежень, використовуючи більш досконалу архітектуру та методи оптимізації. А також надає вищу швидкість роботи та потребує менших обчислювальних ресурсів.

Існують і інші моделі виявлення об'єктів, зокрема R-CNN, Fast R-CNN, Faster R-CNN і SSD (Single Shot Multibox Detector). Хоча ці моделі компетентні в конкретних випадках використання, вони часто не справляються з виявленням об'єктів у реальному часі або вимагають значних обчислювальних ресурсів.

1. R-CNN та її різновиди: незважаючи на високу точність, R-CNN та її ітерації (Fast R-CNN і Faster R-CNN) часто вимагають багаторазового

проходження через дані і представляють складний конвеєр, який включає в себе виявлення регіонів, вилучення ознак і класифікацію. Ці кроки ускладнюють аналіз у реальному часі.

2. SSD: хоча SSD розроблені для роботи в реальному часі, вони іноді йдуть на компроміс з точністю заради швидкості. Для застосунків, що вимагають високої точності, такий компроміс може бути критичним.

YOLO перевершує ці моделі в різних аспектах, що робить її більш придатною для нашого застосування:

- продуктивність в реальному часі: YOLO працює в режимі реального часу, що має вирішальне значення для негайного виявлення вибоїн;
- висока точність: YOLO забезпечує збалансований профіль швидкості та точності, пропонуючи високі значення mAP навіть під час роботи в режимі реального часу;
- масштабованість: розроблена для різних апаратних конфігурацій, YOLO може бути ефективно розгорнута на будь якій платформі від хмарних серверів до локальних пристроїв, забезпечуючи більш широке застосування;
- уніфікована архітектура: на відміну від R-CNN і її варіантів, YOLO пропонує спрощену архітектуру, яка виконує виявлення об'єктів за один прохід, зменшуючи складність і обчислювальні витрати.

На відміну від інших, модель YOLO базується на одній згортковій нейронній мережі (CNN), яка сканує все зображення за один прохід в одну сторону. Така архітектура значно скорочує час обчислень, дозволяючи виявляти об'єкти в реальному часі.

Відеокадри спочатку змінюються за розміром і нормалізуються, щоб відповідати вхідним розмірам, необхідним для YOLO. Цей процес нормалізації гарантує, що значення пікселів знаходяться в межах певної шкали, що дозволяє CNN більш ефективно вивчати об'єкти.

Серія згорткових шарів, оснащених різними фільтрами, витягує ознаки з вхідних кадрів. Ці шари доповнюються функціями активації, такими як ReLU, і методами нормалізації, такими як пакетна нормалізація, щоб стабілізувати і прискорити процес навчання.

Після вилучення ознак архітектура зосереджується на локалізації об'єктів у кадрі. YOLO використовує методи регресії обмежувальних рамок для розміщення координат обмежувальних рамок навколо виявлених об'єктів. Кожна обмежувальна рамка визначається координатами її центру, висотою та шириною.

Разом з координатами обмежувальних рамок мережа також прогнозує типи класів, визначаючи природу об'єкта (рис. 2.4)[12].

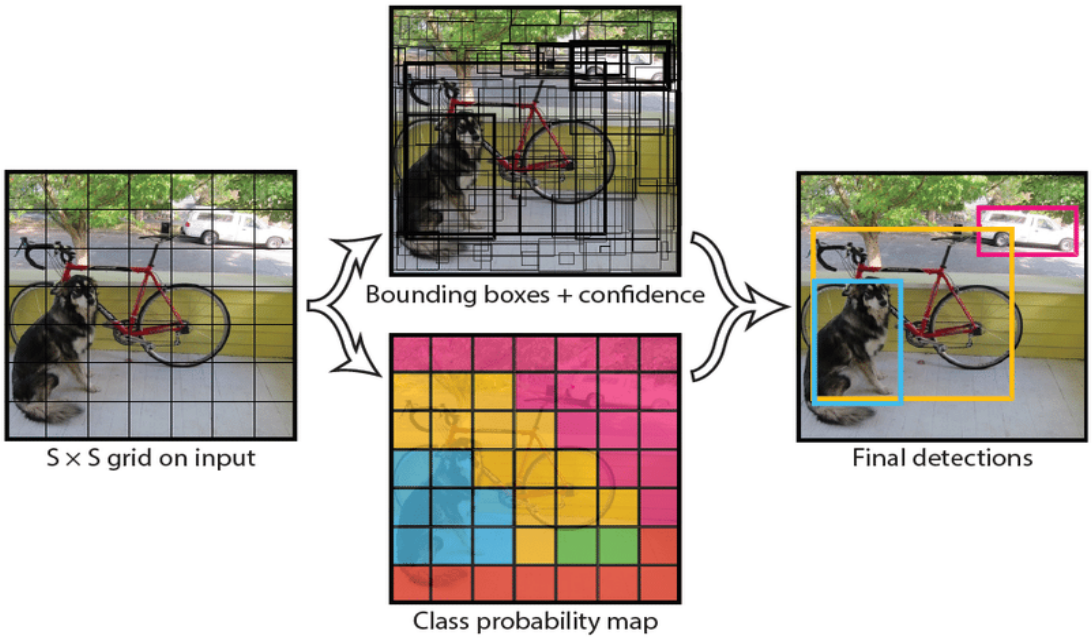


Рисунок 2.4 – Схема розпізнавання об’єктів за допомогою сітки та визначення меж

Для уточнення списку виявлених об'єктів YOLO застосовує алгоритм не максимального утиснення. Ця техніка видаляє надлишкові обмежувальні рамки, залишаючи лише ті, що мають найвищу вірогідність свого класу. Розглядаючи складні методи та алгоритми YOLO, ці аргументи підкреслюють її придатність та чудову продуктивність в контексті аналізу якості дорожнього покриття.

Навчання моделі YOLO включає процес, в якому модель тренують точно ідентифікувати та класифікувати об'єкти на зображеннях, в даному випадку аномалії дорожнього покриття, такі як вибоїни. На етапі навчання моделі подається великий набір даних з анотованих зображень. Ці зображення позначені інформацією про місцезнаходження та характеристики вибоїн або іншими важливими характеристиками. Модель вчиться розпізнавати шаблони та особливості, які відрізняють вибоїни від інших елементів дорожнього покриття, а потім отримує здатність повертати об’єкти з правильними ознаками у відповідь на нові вхідні зображення.

Ключова проблема, з якою стикаються системи аналізу відео, що працюють на основі штучного інтелекту, полягає у покадровому аналізі відео, де кожен окремий кадр подається на вхід моделі ШІ для розпізнавання об'єктів. На виході модель повертає визначені об'єкти, але вона не має можливості самостійно встановити зв'язок між об'єктами, знайденими на попередньому кадрі, та тими, що з'являються на поточному.

Ця проблема є особливо актуальною у сценаріях, де потрібно відстежувати та аналізувати об'єкти, які постійно змінюють своє положення чи орієнтацію. Вирішення цього завдання вимагає використання спеціалізованих алгоритмів, які можуть відстежувати об'єкти від одного кадру до іншого. Одним з таких алгоритмів є SORT (Simple Online and Realtime Tracking), який ефективно дозволяє вирішити проблему відстеження об'єктів у відео потоці, забезпечуючи зв'язок між послідовними виявленнями об'єктів.

SORT застосовує комбінацію методів для прогнозування положення об'єктів у нових кадрах, що дозволяє зберігати послідовність відстеження, навіть коли об'єкти зникають із поля зору або їх розпізнавання ускладнене іншими факторами.

Алгоритм використовує фільтр Калмана для прогнозування місця розташування вибоїни в наступному кадрі. Для підтвердження наявності та розташування вибоїни в новому кадрі алгоритм використовує метрику IOU. Використовуючи як прогнози фільтра Калмана, так і оцінки IOU, алгоритм може

встановити найбільш ймовірні збіги між новими виявленнями та вже відстеженими вибоїнами. Після порівняння і зіставлення нових виявлених об'єктів з уже відстеженими, алгоритм оновлює інформацію про вибоїни. Це оновлення включає в себе уточнення положення, розміру та інших відповідних атрибутів кожної вибоїни на основі останніх даних.

Фільтр Калмана – це потужний алгоритм, який використовується в різних галузях для оцінки стану динамічної системи в часі. Він особливо добре підходить для ситуацій, коли система піддається випадковим збуренням або шуму. Фільтр Калмана працює, роблячи прогнози про майбутній стан системи, оновлюючи ці прогнози на основі нових вимірювань і уточнюючи свої оцінки з плином часу [12].

Фільтр Калмана починається з фази прогнозування, де він використовує математичну модель для передбачення майбутнього стану системи. Ця модель зазвичай враховує поточний стан системи та відомі входи або сигнали керування. Прогнозування також включає оцінку невизначеності, яка представлена у вигляді коваріаційної матриці. Ця матриця дає кількісну оцінку очікуваної мінливості або помилки в прогнозі.

Коли з'являються нові вимірювання або спостереження, фільтр Калмана переходить у фазу оновлення. На цій фазі фільтр враховує нову інформацію, щоб виправити свій прогноз. Це робиться шляхом порівняння прогнозованого стану з фактичним вимірюванням, беручи до уваги невизначеність або шум, пов'язані як з прогнозом, так і з вимірюванням.

Фільтр Калмана поєднує прогнози та вимірювання за допомогою набору рівнянь, які зважують надійність кожного з них. Якщо прогноз вважається більш надійним, йому надається більша вага. І навпаки, якщо новий вимір вважається більш точним, йому надається більша вага. Результатом є оновлена оцінка стану системи, яка в ідеалі є найкращою як з точки зору прогнозу, так і з точки зору вимірювання.

Однією з ключових особливостей фільтра Калмана є його рекурсивна природа. Кожного разу, коли надходять нові дані, фільтр оновлює свої оцінки. Вихід

однієї ітерації фільтра стає входом для наступної ітерації. Таке рекурсивне оновлення дозволяє фільтру Калмана постійно вдосконалювати свої оцінки стану системи з плином часу.

У процесі відстеження об'єктів, таких як вибоїни на дорогах з відео, фільтр Калмана використовується для прогнозування місцеположення цих вибоїн у кожному новому кадрі. Цей фільтр працює таким чином: спочатку він робить припущення про те, де може знаходитися вибоїна в наступному кадрі. Потім, коли вибоїна реально з'являється у кадрі, фільтр Калмана використовує цю інформацію для коригування свого первісного прогнозу.

Головна перевага фільтра Калмана полягає в його здатності ефективно працювати з невизначеністю даних, отриманих від моделі. Він може адаптуватися до змін у русі вибоїни, а також до можливих неточностей при її виявленні на кожному кадрі.

Метрика «Перетин над об'єднанням» (Intersection Over Union, IOU) – це спосіб оцінювання точності в алгоритмах, які виявляють та відстежують об'єкти. Ця метрика порівнює дві області: одна, яку модель прогнозує як межу об'єкта, та інша – реальна межа об'єкта, відома як «фундаментальна істина» (ground truth). IOU вимірює, наскільки ці дві області перекриваються, допомагаючи зрозуміти, наскільки точно модель визначила положення та форму об'єкта [13]. Щоб краще зрозуміти метрику IOU, розглянемо наступні кроки що зображені на рисунку 2.5.

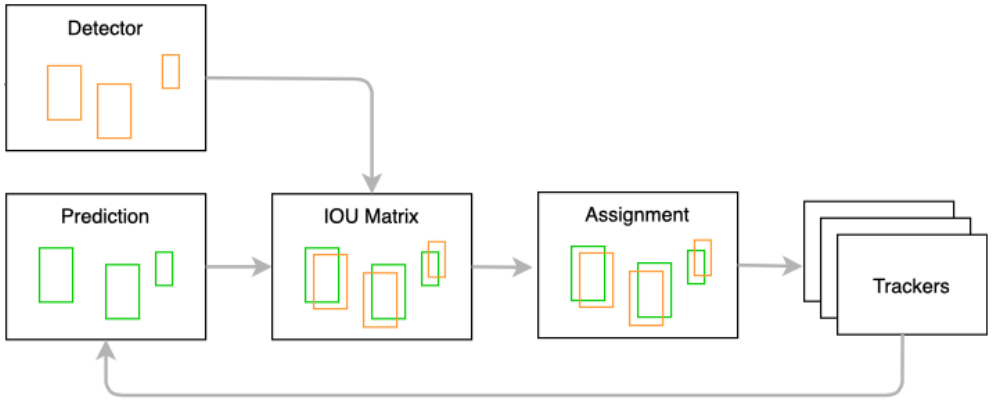


Рисунок 2.5 – Схема поєднаної роботи фільтру Калмана та метрики IOU

Спочатку визначаються межі об'єктів. У задачах виявлення об'єктів ці межі зазвичай окреслюються за допомогою обмежувальних рамок. Одна обмежувальна рамка представляє прогнозовану площу об'єкта (визначену ШІ-моделлю), а інша – фактичну площу (істину).

Перетин двох обмежувальних рамок – це площа, яку вони обидві покривають. Це перекриття між прогнозованою та фактичною областями. Об'єднанням двох граничних областей є загальна площа, що покривається обома областями. Вона включає всю площу як прогнозованої, так і фактичної рамки, за вирахуванням площі перетину (оскільки вона рахується двічі).

IOU обчислюється шляхом ділення площі перетину на площу об'єднання. Отримане значення знаходиться в діапазоні від 0 до 1, де 0 означає відсутність перетину (повне неспівпадіння), а 1 вказує на ідеальне вирівнювання (прогнозована гранична область точно збігається з істинною), як зазначено у формулі 2.1.

Математично це виглядає так:

$$IoU = \frac{S_i}{S_u} \tag{2.1}$$

де S_i – площа перетину, S_u – площа об’єднання.

На практиці метрика IOU використовується для оцінки точності моделей виявлення об'єктів. Вищий показник IOU вказує на те, що прогнози моделі є більш точними і тісно пов'язаними з реальними об'єктами. У контексті виявлення вибоїн у застосунку для аналізу доріг, IOU допомагає визначити, наскільки добре модель визначила місцезнаходження та розмір вибоїн у кожному кадрі відео, що є важливим для точного відстеження та дедуплікації виявлених об'єктів.

Висновки до розділу 2

У другому розділі було проведено аналіз літератури та огляд існуючих методів, що стосуються оцінювання якості дорожнього покриття.

Зроблено огляд існуючих рішень, включаючи системи на базі сенсорів, візуальний огляд та інші методи. Порівняно переваги та недоліки кожного з підходів, визначено, що візуальний огляд, хоч і ефективний, має певні обмеження, особливо у плані об’єктивності та швидкості обробки даних.

З огляду на розвиток технологій, було вирішено зосередитися на навчанні моделі на розпізнавання ям для асфальтових доріг, оскільки це видається найбільш перспективним напрямком для розвитку системи оцінювання якості дорожнього покриття.

Також розглянуто алгоритми машинного навчання для аналізу якості дорожнього покриття на відео. Визначено оптимальний алгоритм та розглянуто варіанти вирішення проблеми відстеження переміщення об’єктів, яка виникає при використанні моделей ШІ для аналізу відео через відсутність зв’язку між розпізнаваннями на послідовних кадрах.

3 ПРОГРАМНІ ІНСТРУМЕНТИ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОГО ЗАВДАННЯ

3.1 Технології контейнеризації та збереження даних

В основі архітектури розробленої системи лежать рішення для швидкого створення, налаштування та модифікації інфраструктури, що забезпечує автоматизоване розгортання та масштабованість системи. Оскільки система займається високонавантаженими комп'ютерними обчисленнями при аналізі відео, важливо мати можливість її запуску на будь-якому сервері з потужним обладнанням.

Контейнеризація є ключовою технологією в сучасній розробці програмного забезпечення, в контексті розробки та розгортання застосунків у ефективний спосіб. Контейнеризація передбачає інкапсуляцію програми та її залежностей у контейнер, який може бути виконаний у будь-якому сумісному обчислювальному середовищі. Такий підхід гарантує, що застосунок працює однаково і передбачувано, незалежно від того, де він розгорнутий, ізолюючи його від базової інфраструктури та інших застосунків [14].

У контексті системи, що розробляється, контейнеризація відіграє вирішальну роль. Застосунок було контейнеризовано за допомогою Docker, який є одним з найпопулярніших і широко використовуваних інструментів контейнеризації. Docker упаковує застосунки в контейнери, надаючи легкий, автономний, виконуваний пакет, який включає в себе все необхідне для запуску програми: код, середовище виконання, системні інструменти, системні бібліотеки та налаштування.

Основною перевагою є те, що після контейнеризації програми за допомогою Docker вона може працювати на будь-якій системі, що підтримує Docker, будь то локальна машина розробника або справжній сервер клієнта, без необхідності пере налаштувань.

Docker гарантує, що застосунок працює в ізольованому середовищі, відомому як контейнер. Ця ізоляція запобігає конфліктам між компонентами, а також між модулями та їх оточенням, забезпечуючи узгодженість між різними середовищами розробки, тестування та виробництва.

Контейнери використовують ядро хост-системи, але можуть бути обмежені у використанні певної кількості ресурсів, таких як процесор і пам'ять. Це робить їх більш легкими та ефективними порівняно з використанням окремих віртуальних машин для кожної програми.

Docker спрощує процес розгортання, надаючи простий спосіб пакування програми з усіма її залежностями. Це прискорює процеси розробки та розгортання, оскільки розробникам не потрібно турбуватися про специфічні для середовища конфігурації.

Хоча Docker є популярним вибором для контейнеризації, існують альтернативи, такі як Podman та Kubernetes. Кожен з цих інструментів має свої сильні сторони і може мати перевагу в певних сценаріях:

- Podman часто згадують за його бездемонічну архітектуру, що може бути перевагою з точки зору безпеки та простоти;
- Kubernetes не є прямою альтернативою Docker, але часто використовується для оркестрування контейнерів. Він відмінно справляється з управлінням великомасштабними розгортаннями контейнерів.

Docker був обраний для цього застосунку через його широке розповсюдження, простоту використання, багатий набір функцій та велику спільноту, яка його підтримує. Наявність Docker Hub, репозиторію для зображень Docker, також є значною перевагою, оскільки він надає доступ до безлічі готових зображень, що може значно скоротити час розробки.

В описаному застосунку Docker використовується для розміщення двох основних модулів – мікросервісу для управління та мікросервісу для аналізу зображень – кожен у своєму окремому контейнері. Це налаштування ілюструє

типовий випадок використання Docker, використовуючи його здатність ізолювати та керувати ресурсами для різних компонентів програми.

Веб-мікросервіс, що відповідає за завдання керування, зазвичай вимагає менше ресурсів порівняно з мікросервісом, що аналізує зображення. Останній виконує більш ресурсомісткі операції під час обробки та аналізу зображень, що може вимагати значних обчислювальних затрат. Контейнеризуючи ці модулі окремо, Docker дозволяє точно розподіляти і обмежувати ресурси процесора і пам'яті для кожного сервісу. Це гарантує, що мікросервіс аналізу зображень має достатньо ресурсів для ефективної роботи, в той час як мікросервіс веб-керування працює з меншим розподілом ресурсів, що відображає його менші вимоги.

Крім того, база даних MySQL, яка є невід'ємною частиною управління та зберігання даних програми, також розміщується в окремому контейнері Docker. Такий підхід має кілька переваг. По-перше, він підтримує базу даних в ізольованому середовищі, що підвищує безпеку і знижує ймовірність конфліктів з іншими сервісами. По-друге, використання Docker-контейнерів для бази даних спрощує процес управління базою даних, включаючи резервне копіювання та відновлення. Застосунок може швидко відновити базу даних зі знімка, якщо це необхідно, забезпечуючи мінімальний час простою та ефективне відновлення в разі втрати або пошкодження даних.

MySQL є оптимальним вибором для сценаріїв з високою інтенсивністю запису в першу чергу завдяки своїй відмінній продуктивності та надійності в управлінні паралельними операціями запису. Механізм зберігання даних InnoDB в MySQL спеціально оптимізований для обробки великих обсягів операцій запису, використовуючи блокування на рівні рядків для мінімізації конфліктів і максимізації пропускної здатності. Це робить його дуже придатним для систем, які генерують і обробляють великі обсяги даних у режимі реального часу [15].

Під час проектування було вирішено не зберігати зображення та відеодані безпосередньо в базі даних MySQL. Зберігання великих бінарних файлів, таких як зображення та відео, в базі даних може призвести до значної неефективності. Це

може не тільки збільшити розмір бази даних, але й погіршити продуктивність через збільшення навантаження під час операцій пошуку та зберігання даних. Робота з великими обсягами таких даних також може ускладнити процеси резервного копіювання та відновлення, що робить систему більш громіздкою в обслуговуванні.

Натомість програма розроблена для підтримки найпоширеніших форматів даних для зберігання зображень і відео – JPEG для зображень і MP4 для відео. Ці формати широко використовуються і визнані завдяки балансу якості та стиснення, що робить їх практичними для використання в різних застосунках, в тому числі тих, які включають значні завдання з обробки та аналізу даних.

Docker відіграє важливу роль в управлінні цими файлами. Однією з ключових переваг використання Docker є його здатність використовувати спільне сховище файлів для контейнерів. Це означає, що хоча різні контейнери програми (наприклад, контейнери для веб-сервера, сервіса обробки зображень і бази даних) працюють незалежно, вони мають доступ до спільної системи зберігання файлів і можуть спільно використовувати її. Це спрощує архітектуру, уникаючи дублювання даних у різних контейнерах, і гарантує, що всі частини програми мають послідовний і актуальний доступ до необхідних файлів.

3.2 Технології комунікації між компонентами

При розробці застосунку було обрано HTTP з RESTful архітектурою як основний засіб зв'язку між мікросервісами. Це рішення ґрунтується на універсальності, простоті та широкому розповсюдженні RESTful сервісів над HTTP, що робить його кращим вибором для мікросервісних архітектур. RESTful API надають протокол зв'язку без стану, що робить їх дуже масштабованими та адаптованими до різних типів даних та операцій. Вони дозволяють різним сервісам у застосунку безперешкодно взаємодіяти через Інтернет, надсилаючи запити та отримуючи відповіді у зручному для обробки форматі, наприклад, JSON або XML.

Хоча було обрано HTTP з REST, існують й інші популярні альтернативи для взаємодії мікросервісів. Наприклад, gRPC, високопродуктивний фреймворк RPC, розроблений Google, можна було б використовувати, особливо для застосунків, що вимагають низької затримки і високої пропускної здатності. Іншою альтернативою є GraphQL, яка набирає популярності завдяки тому, що дозволяє клієнтам запитувати саме ті дані, які їм потрібні. Однак ці альтернативи не були обрані через додаткову складність і специфічні вимоги, які вони задовольняють, що не були важливими для цієї системи.

У середовищі Docker зв'язок між контейнерами, що виконують різні мікросервіси, досягається за допомогою мережевих можливостей Docker. Docker надає віртуальну мережу, до якої можуть приєднуватися контейнери, дозволяючи їм спілкуватися один з одним так, ніби вони знаходяться в одній фізичній мережі.

Зважаючи на те, що мікросервіс, відповідальний за обробку відео, інтенсивно використовує процесор і графічний процесор, важливо ефективно керувати робочим навантаженням, щоб забезпечити стабільність і продуктивність системи. Для вирішення цієї проблеми ідеальним рішенням є механізм комунікації в черзі. Цей механізм дозволяє системі керувати чергою вхідних запитів на обробку відео, обробляючи їх по одному або невеликими пакетами, відповідно до наявних можливостей обробки.

Простим, але ефективним прикладом такого механізму може бути базова черга за принципом "першим прийшов - першим пішов" (FIFO). У цьому випадку, коли надходить новий запит на обробку відео, він додається в кінець черги. Потім мікросервіс обробки відео обробляє запити з початку черги по одному, переходячи до наступного запиту лише після завершення поточного. Такий підхід гарантує, що всі запити обробляються в порядку їх надходження, забезпечуючи справедливу і впорядковану систему обробки [16].

Для реалізації цієї черги FIFO в контексті програми можна використовувати легку систему обміну повідомленнями або просту таблицю бази даних. Наприклад, для зберігання черги можна використовувати реляційну базу даних, таку як MySQL,

де кожен рядок представлятиме запит на обробку відео. Таблиця міститиме поля для зберігання необхідної інформації про кожен запит, такої як унікальний ідентифікатор, деталі запиту і прапорець статусу, що вказує на те, чи запит очікує на обробку, чи перебуває в процесі обробки, чи вже виконаний, як показано на рисунку 3.1.

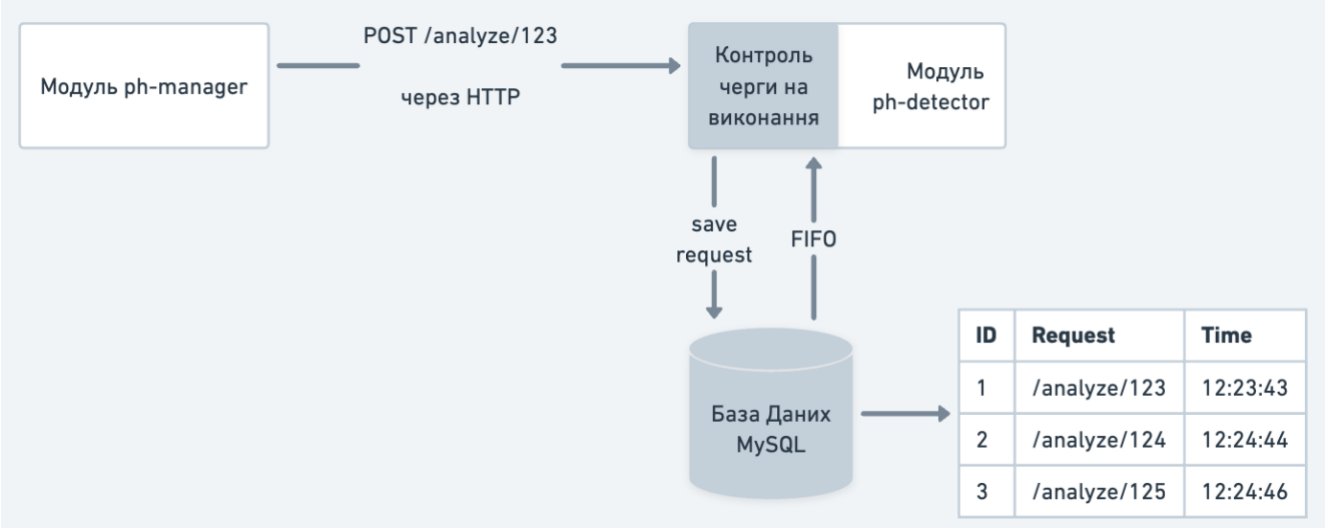


Рисунок 3.1 – Взаємодія мікросервісів через систему черги для високонавантажених запитів

Мікросервіс обробки відео регулярно перевіряє таблицю черги на наявність наступного запиту, обробляє його, а потім оновлює його статус у таблиці. Цей метод забезпечує простий і надійний спосіб управління чергою з мінімальними накладними витратами, використовуючи знайомі і широко підтримувані технології баз даних.

3.3 Технології розробки вебзастосунку

Мікросервіс, що відповідає за управління даними в застосунку, відіграє вирішальну роль у забезпеченні доступу до всіх функцій системи. Він робить це не лише через API, але й через вебінтерфейс. Цей вебінтерфейс призначений для повернення HTML-сторінок, що дозволяє користувачам взаємодіяти з системою

через браузер. Вибір Golang (Go) як мови програмування для розробки цього мікросервісу має кілька переваг.

По-перше, Golang відома своєю простотою та ефективністю. Вона має чистий і лаконічний синтаксис, що робить код легким для читання і підтримки. Ця простота особливо корисна для створення масштабованих і підтримуваних вебзастосунків, оскільки вона зменшує складність і ймовірність помилок [17].

По-друге, Go має високу продуктивність. Вона компілюється в машинний код, а це означає, що вона виконується дуже швидко і ефективно. Це дуже важливо для веб-сервісу, який обробляє багато запитів і повинен надавати швидкі відповіді. Ефективне виконання Go гарантує, що вебінтерфейс буде миттєво відповідати і зможе впоратися зі значним навантаженням, що дуже важливо для сервісу, який керує даними і надає доступ до різних функцій системи.

Ще однією значною перевагою Golang є її надійна стандартна бібліотека, особливо для веб-розробки. Стандартна бібліотека включає потужні інструменти для реалізації HTTP-серверів та клієнтів, HTML-шаблонів тощо. Це означає, що більша частина функціональності, необхідної для веб-сервісу, може бути реалізована за допомогою стандартних, добре протестованих та оптимізованих пакетів, що надаються Go, зменшуючи потребу в зовнішніх залежностях.

Крім того, Golang пропонує чудову підтримку паралельності. Її підпрограми та канали спрощують створення застосунків, які можуть ефективно обробляти декілька завдань паралельно. Це корисно для веб-сервісів, які повинні обробляти кілька запитів користувачів одночасно без шкоди для продуктивності.

Окрім використання шаблонів Go для рендерингу HTML-сторінок, вебзастосунок включає HTMX – розширення HTML на основі атрибутів, яке дозволяє динамічно оновлювати вміст. Інтеграція HTMX надає вебзастосунку кілька переваг, підвищуючи його інтерактивність та зручність для користувача [18].

HTMX дозволяє виконувати оновлення вебсторінки в стилі AJAX (асинхронний JavaScript і XML). Це означає, що певні частини вебсторінки можуть оновлюватися у відповідь на дії користувача без необхідності перезавантаження

всієї сторінки. Наприклад, коли користувач надсилає форму, оновлюється лише відповідний розділ сторінки, а не вся сторінка. Це призводить до більш плавного і чуйного користувацького досвіду.

На відміну від більш важких фреймворків JavaScript, HTMX зводить до мінімуму логіку на стороні клієнта. Це зменшує складність програми, що призводить до меншої кількості помилок і полегшує обслуговування.

Оновлюючи лише частини вебсторінки, HTMX мінімізує обсяг даних, що передаються між клієнтом і сервером. Це може значно підвищити продуктивність програми, особливо на повільних мережевих з'єднаннях.

На додаток до HTMX, концепція гіпермедійної системи відіграє вирішальну роль у застосунку. Гіпермедійна система в контексті вебзастосунків – це система, в якій інтерфейс, представлений користувачеві, динамічно генерується на основі поточного стану застосунка і взаємодії з ним користувача. Це основний принцип архітектурного стилю REST.

Переваги гіпермедійної системи включають динамічну природу гіпермедіа, яка дозволяє застосунку бути більш адаптивним і масштабованим, оскільки зміни у функціональності програми можуть бути відображені в інтерфейсі користувача без необхідності повного редизайну [19].

Використання гіпермедійного підходу призводить до роз'єднання між клієнтом і сервером, оскільки клієнту не потрібно мати попередніх знань про те, як формувати запити на виконання певних дій. Натомість сервер надає цю інформацію динамічно, що забезпечує більшу гнучкість у тому, як API розвивається з часом.

Golang також має сильну та активну спільноту, що означає наявність великої кількості ресурсів, бібліотек та інструментів. Підтримка спільноти може бути неоціненною при розробці складних вебзастосунків, оскільки вона надає доступ до широкого спектру сторонніх бібліотек та фреймворків, які можуть пришвидшити розробку та запропонувати рішення спільних проблем.

Однією з таких бібліотек є Gin – високопродуктивний веб-фреймворк для мови програмування Go. Він відомий своєю ефективністю і широко

використовується для створення вебзастосунків і мікросервісів. Однією з особливостей Gin є його потужний маршрутизатор, який відіграє важливу роль в обробці HTTP-запитів.

Gin розроблений для високої продуктивності. Його маршрутизатор оптимізований для швидкості і може обробляти запити швидше, ніж багато інших веб-фреймворків Go. Це робить його відмінним вибором для застосунків, де швидкість і ефективність мають вирішальне значення.

Незважаючи на свою продуктивність, Gin залишається зручним у використанні. Він пропонує простий та інтуїтивно зрозумілий API для визначення маршрутів та обробки HTTP-запитів. Розробники можуть швидко налаштувати правила маршрутизації та визначити, як різні шляхи будуть оброблятися системою.

Маршрутизатор Gin дозволяє використовувати проміжне програмне забезпечення, тобто функції, які можуть бути виконані до або після основних обробників запитів. Це корисно для таких завдань, як ведення журналів, аутентифікація користувачів та обробка даних, які потрібно виконувати для декількох маршрутів.

Gin дозволяє групувати маршрути, що особливо корисно для великих програм. Маршрути можуть бути згруповані на основі їх спільних характеристик, таких як спільні префікси шляху або проміжне програмне забезпечення, що робить кодову базу більш організованою та керованою.

Для обробки різних типів HTTP-запитів маршрутизатор Go може бути оснащений різними обробниками. Для GET-запитів, які в основному використовуються для завантаження HTML-сторінок, маршрутизатор використовує шаблони Go і HTMX.

Крім обробки GET-запитів, маршрутизатор також налаштований на управління такими діями, як завантаження або видалення, які виконуються за допомогою методів POST і DELETE відповідно. Ці дії змінюють стан програми – наприклад, додають новий відео або графічний файл або видаляють існуючий. Після завершення цих дій маршрутизатор повертає порції HTML, які динамічно

вставляються у вебсторінку. Цей процес включає в себе додавання нового контенту або заміну існуючих блоків на вебсторінці, що дозволяє оновлювати користувальницький інтерфейс в режимі реального часу на основі дій користувача.

Маршрутизатор забезпечує надійне управління помилками. Він дозволяє розробникам визначати власну логіку обробки помилок, гарантуючи, що застосунок може витончено обробляти несподівані ситуації та надавати змістовні відповіді клієнту.

Таким чином, використання Golang для розробки вебінтерфейсу мікросервісу управління даними значно підвищує продуктивність, масштабованість та ремонтопридатність застосунку. Ефективність, простота використання та потужна підтримка паралелізму роблять її ідеальним вибором для сучасної розробки вебзастосунків.

3.4 Технології розробки сервісу аналізу покриття

Мікросервіс аналізу стану доріг, що є центральним для оцінки дорожніх умов, потребує вбудованої підтримки моделей штучного інтелекту. У цьому контексті Python є найкращою мовою програмування з кількох вагомих причин.

Сумісність і простота інтеграції Python з моделями штучного інтелекту та машинного навчання (ML) не мають собі рівних [20]. Мова широко відома своєю широкою підтримкою у спільнотах ШІ та ML, що призвело до розробки численних фреймворків та бібліотек, спеціально розроблених для цих галузей. Такі бібліотеки, як TensorFlow, PyTorch та Scikit-learn - це лише кілька прикладів інструментів, які полегшують розробку та впровадження складних моделей ШІ на Python. Ці бібліотеки пропонують готові функції та модулі, які значно скорочують час і зусилля, необхідні для розробки складних алгоритмів штучного інтелекту, що робить Python ідеальним вибором для мікросервісу з аналізу доріг.

Крім того, сильні сторони Python в обробці зображень є ще однією причиною, чому він виділяється для цього конкретного сервісу. Такі бібліотеки, як OpenCV та

PIL (Python Imaging Library) надають надійні інструменти для маніпулювання та обробки зображень. Ці бібліотеки мають вирішальне значення для аналізу та обробки зображень доріг з метою виявлення дефектів, таких як вибоїни, тріщини та інші аномалії. Вони дозволяють програмі ефективно і точно інтерпретувати візуальні дані, що має фундаментальне значення для оцінки стану доріг.

Python також чудово справляється з паралельною обробкою та роботою з масивами даних, що дуже важливо в контексті обробки координат виявлених вибоїн та інших дорожніх особливостей. Такі бібліотеки, як NumPy та Pandas, забезпечують ефективну обробку великих наборів даних та масивів, а бібліотеки багатопроцесорної та багатопотокової обробки в Python дозволяють проводити паралельну обробку даних. Ця можливість особливо важлива в контексті аналізу доріг, де швидка та ефективна обробка великих обсягів даних зображень і пов'язаних з ними координат має вирішальне значення.

У системі Python запускає модель для аналізу зображень, завантажуючи її зі спеціального файлу. Завантаження моделі зазвичай виконується за допомогою функції, наданої PyTorch, ця функція десеріалізує збережений файл моделі та реконструює модель у середовищі Python, готову до виведення. Після завантаження модель можна використовувати для аналізу та обробки зображень, ідентифікації та класифікації об'єктів на них, наприклад, для виявлення вибоїн на зображеннях дорожнього покриття.

Обробка відео для виявлення вибоїн у застосунку включає ще один важливий аспект – визначення геолокації цих вибоїн. Для досягнення точного GPS-відстеження відео можна використовувати кілька методів, але найбільш ефективним рішенням є використання GPX-файлів.

GPX, що розшифровується як GPS Exchange Format, є широко використовуваною XML-схемою, призначеною для зберігання географічних даних, таких як точки, маршрути і треки. Цей формат особливо підходить для програми, оскільки його можна генерувати за допомогою різноманітного безкоштовного

програмного забезпечення, доступного для смартфонів, а також багатьох сучасних відео реєстраторів.

Коли користувач записує відео дороги, а його GPS-пристрій (наприклад, смартфон або відео реєстратор) також реєструє його місцезнаходження, отриманий GPX-файл містить серію координат з відповідними часовими мітками. Ці дані можна синхронізувати з відео, щоб точно визначити місцезнаходження кожної виявленої вибоїни.

Python, з його потужним набором бібліотек для аналізу та обробки даних, особливо добре підходить для роботи з GPX-файлами. Він пропонує ефективні та прості методи для аналізу цих XML-файлів, вилучення необхідних географічних даних та співставлення їх з результатами відео аналізу. Такі бібліотеки, як gpxpy для аналізу GPX-файлів і pandas для маніпулювання даними, полегшують обробку цих географічних даних, узгодження їх з виявленими вибоїнами на відео і, зрештою, надання точної геолокаційної інформації для кожної вибоїни.

Поєднання вбудованої підтримки Python для моделей штучного інтелекту, широких бібліотек для обробки зображень та здатності ефективно виконувати паралельну обробку і великі масиви даних робить її оптимальним вибором для розробки мікросервісу для аналізу доріг [21].

3.5 Технології розробки штучного інтелекту

Корисність та ефективність будь-якої програми на основі машинного навчання значною мірою залежить від вибору алгоритмів та моделей, що використовуються. Для завдання аналізу якості дорожнього покриття за допомогою відеозаписів в якості моделі типу YOLO ми обрали реалізацію YOLOv8 від компанії Ultralytics. Цей вибір зумовлений багатьма факторами, такими як продуктивність у реальному часі, висока середня точність (mAP), масштабованість та легкість інтеграції у більші системи [22].

Особливість даної реалізації полягає в тому, що в результаті навчання вона дозволяє генерувати декілька файлів з вагою різного розміру. Така гнучкість є особливо корисною, оскільки вона враховує різні можливості продуктивності комп'ютерів. Наприклад, компактніший файл з вагами можна використовувати на системі з обмеженою обчислювальною потужністю, гарантуючи, що модель все ще ефективно функціонує, не перевантажуючи ресурси системи. І навпаки, більш повний файл вагових коефіцієнтів, який може охоплювати більше нюансів даних, може бути використаний на високопродуктивній машині для досягнення більшої точності.

Після завершення навчання вивчені ваги моделі зберігаються у форматі «.pt». Цей формат ідеально підходить для застосунків на основі Python, оскільки він легко завантажується за допомогою бібліотек PyTorch – популярного вибору для систем глибокого навчання.

Вибір формату «.pt» як результуючого формату для навченої моделі добре узгоджується із загальною технологічною структурою програми. Він забезпечує безперешкодну інтеграцію моделі ШІ з компонентами системи, що базуються на Python. Крім того, наявність різних розмірів вагових файлів забезпечує гнучкість розгортання моделі в різних середовищах, задовольняючи широкий спектр вимог до продуктивності, тим самим підвищуючи універсальність і адаптивність програми.

Висновки до розділу 3

У третьому розділі роботи було обрано оптимальну архітектуру та технології для її реалізації. Було прийнято рішення про розробку системи на основі мікросервісної архітектури, що дозволяє забезпечити високий рівень гнучкості, масштабованості та надійності.

Основними компонентами системи є мікросервіси для обробки зображень та управління даними, що виконують важливі функції в рамках загальної архітектури.

Застосування технологій контейнеризації, зокрема Docker, сприяє ефективному розподілу ресурсів та управлінню середовищем виконання. Вибір мов програмування Python та Go відповідає сучасним стандартам розробки вебзастосунків та сервісів аналізу даних, забезпечуючи високу продуктивність та простоту утримання системи.

Особлива увага приділяється розробці штучного інтелекту на основі моделі YOLOv8, що є ключем до ефективного виявлення дефектів дорожнього покриття на зображеннях та відео. Технології, використані для розробки цієї складової системи, дозволяють точно та швидко обробляти великі обсяги візуальних даних, а також ефективно визначати геолокацію дефектів.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Архітектура системи

Основною метою архітектури системи є створення гнучкої, ефективної та масштабованої платформи для аналізу якості дорожнього покриття. З цією метою система розроблена як набір модульних компонентів, що дозволяє легко адаптуватися до різних вимог і розширюватися з часом.

Архітектура системи заснована на принципах мікросервісної архітектури, яка включає два ключових мікросервіси, що зображені на рисунку 4.1.

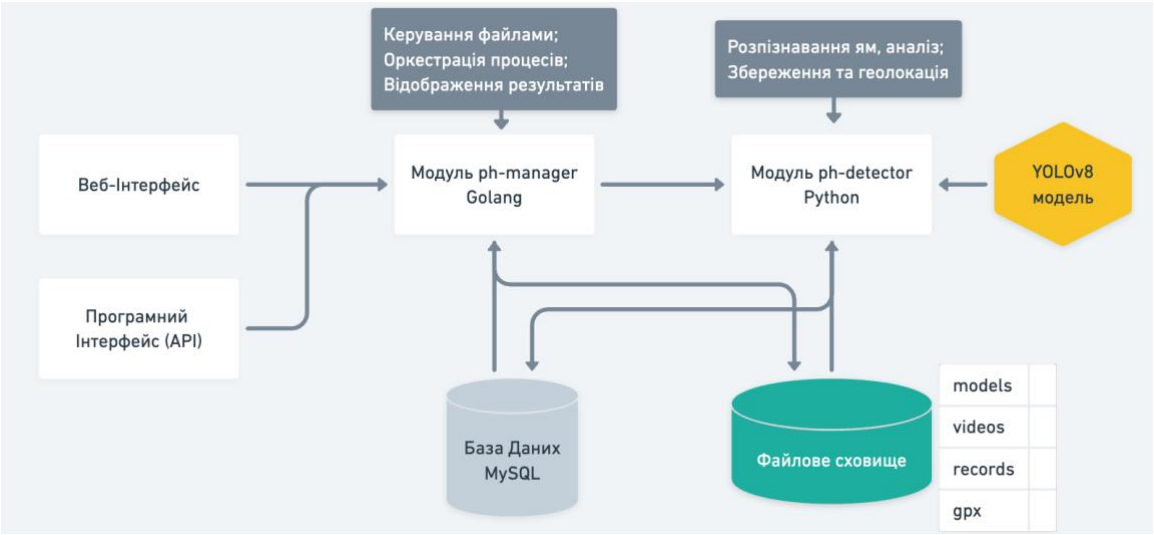


Рисунок 4.1 – Загальна архітектура модулів системи

Мікросервіс для аналізу зображень, відео та геоданих займається обробкою та аналізом вхідних медіа-файлів і геопросторових даних. Він використовує алгоритми машинного навчання та комп'ютерного зору для виявлення дефектів на дорожньому покритті. Він використовує зовнішню моделі ШІ для аналізу зображень. Модель ШІ навчена розпізнавати та класифікувати різні ями. В свою чергу мікросервіс має відправляти зображення до моделі ШІ, щоб отримувати результати аналізу, які потім використовуються для подальшої обробки та збереження. Окрім аналізу візуальних даних, мікросервіс також підтримує роботу з географічними даними. Це включає можливість обробки форматів, таких як GPX,

які використовуються для визначення географічного положення дефектів, виявлених на зображеннях.

Мікросервіс для управління даними та зберігання відповідає за зберігання та управління даними, які генеруються користувачами системи та процесом аналізу. Він включає використання бази даних для зберігання інформації про стан доріг, виявлені дефекти, а також зберігання відео та фото.

Обидва мікросервіси мають визначені інтерфейси (API) для взаємодії між собою та з зовнішніми системами, що дозволяє легко інтегрувати нові компоненти та функціональності у майбутньому.

Оскільки в системі основні функції розподілені між мікросервісами, що знаходяться в контейнерах, API виступає як міст, що з'єднує ці сервіси. Це дозволяє мікросервісу для аналізу зображень, відео та геоданих обмінюватися інформацією з мікросервісом для управління даними та зберігання. На рисунку 4.2 можна побачити схему реалізованої контейнеризації системи.

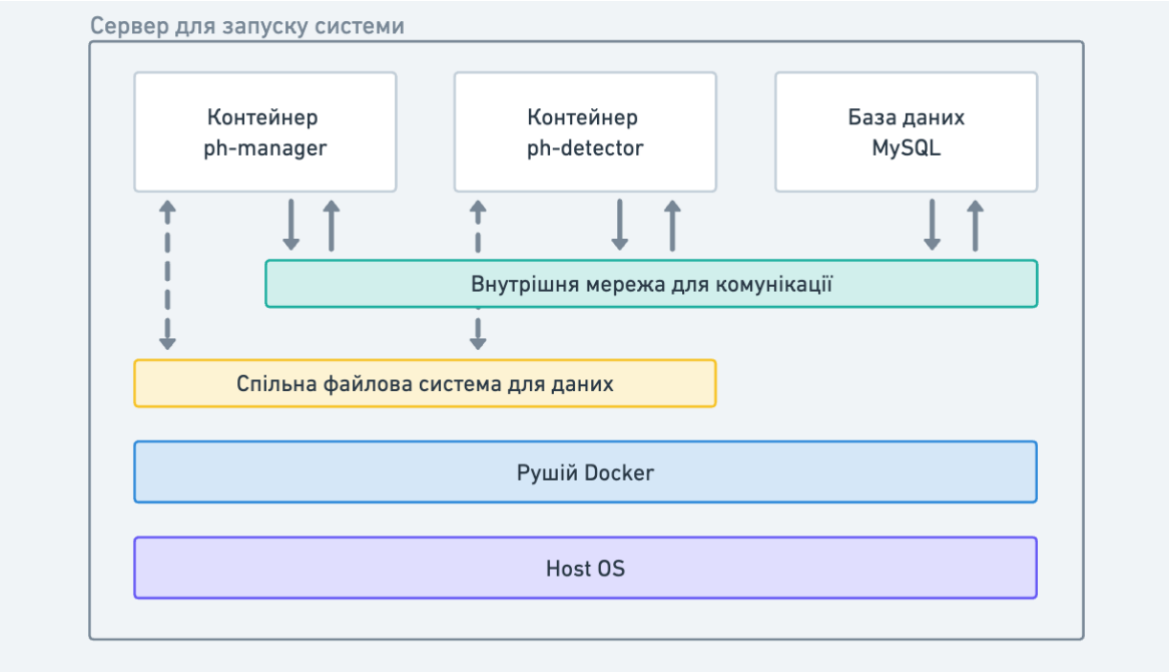


Рисунок 4.2 – Схема контейнерної архітектури системи

Відкритий API дозволяє іншим системам інтегруватися з розробленою системою, що розширює можливості її використання. Мікросервіс для управління

даними та зберігання надає API, що дає доступ до всіх основних функцій системи, таких як отримання даних про дорожнє покриття, управління відео та зображеннями.

Окрім API, система також надає вебінтерфейс, який дозволяє користувачам безпосередньо взаємодіяти з системою через веб-браузер. Це робить систему доступнішою для широкого кола користувачів, надаючи зручний інтерфейс для візуалізації даних та їх управління.

4.2 Структура сховища даних

В основі структури бази даних лежить таблиця «recordings». Кожен рядок у цій таблиці представляє запис, це можуть бути відео чи зображення, які користувачі завантажують для виявлення ям на дорогах. Ключові стовпці включають «id» як унікальний ідентифікатор, «file_name», який є унікальним для кожного запису, і «type», який вказує, чи є запис зображенням або відео. Пов'язаний з ним «status» вказує на стадію обробки кожного запису, починаючи від його створення до обробки, і, нарешті, на те, чи він завершений, чи стався якийсь збій. Для наочності, кожен запис також має позначку часу в колонці «created_at», яка вказує, коли запис було завантажено.

Для кожного виявленого дефекту дороги система заповнює таблицю «detections». Ця таблиця пов'язана з таблицею «recordings» через зовнішній ключ «recording_id». Тут зберігається така важлива інформація, як ім'я файлу виявлення, номер кадру у випадку відеозаписів та оцінка достовірності. Кожне виявлення має позначку часу у стовпчику «created_at», що забезпечує хронологічне відстеження.

Для включення географічної інформації вирішальну роль відіграють дві таблиці: «gpx» і «detection_location». Таблиця «gpx» зберігає файли геолокаційних даних, пов'язаних із записами, тоді як таблиця «detection_location» зберігає конкретну широту і довготу кожної виявленої вибоїни. Вона пов'язана як з

таблицею «detections» через «detection_id», так і з таблицею «gpx» через «gpx_id». Загальна схема приведена на рисунку 4.3.

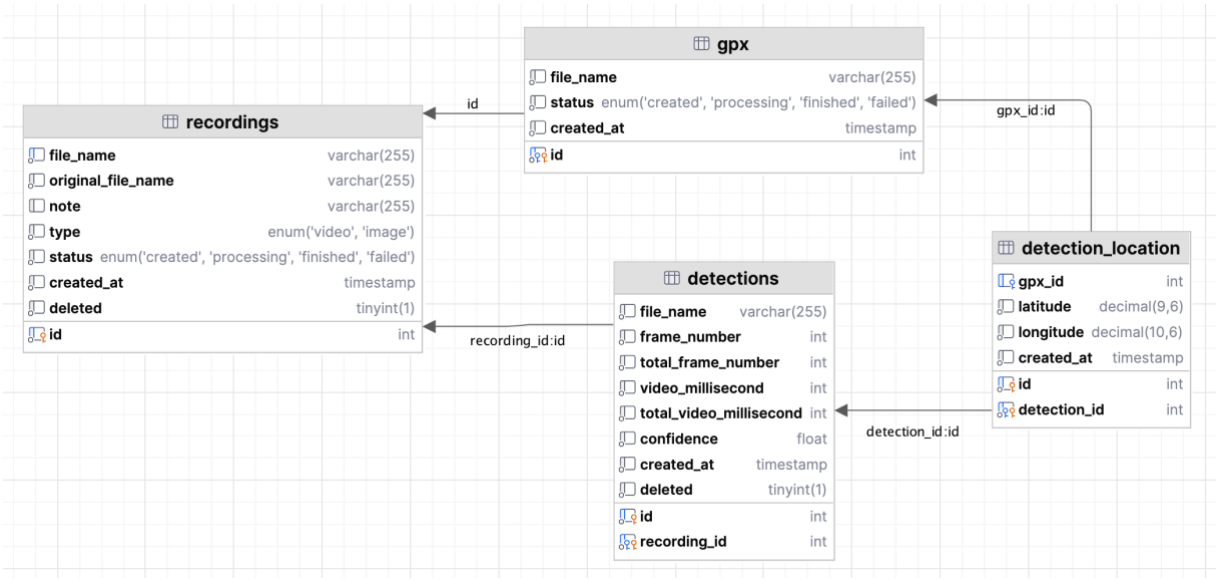


Рисунок 4.3 – Структура бази даних для збереження інформації про вибоїни

Файлове сховище системи структуроване для забезпечення ефективного управління та пошуку різних типів файлів. Основний каталог сховища, «/ph-storage/», є кореневим. Всередині він складається з чотирьох під-каталогів, кожен з яких призначений для певного типу файлів і має назву, що відображає їхній вміст.

«/gpx» – цей каталог призначений для зберігання файлів GPX, завантажених користувачами. Ці файли пов'язані з конкретними записами і мають вирішальне значення для надання географічних даних, пов'язаних з дорожніми умовами, що аналізуються. Кожному файлу в цьому каталозі для унікальності присвоюється ім'я з використанням випадкового цілого числа у форматі «g_<випадкове_число>.gpx», наприклад, «g_3812848044.gpx». Такий спосіб іменування забезпечує легку ідентифікацію та пошук GPX-файлів.

«/models» – каталог models містить файли у форматі «.pt», які є навченими моделями, що використовуються мікросервісом детектора ям на основі Python. Ці файли мають різний розмір, щоб відповідати різним обчислювальним потужностям хост-машин. Наявність декількох файлів моделей дозволяє системі обирати

відповідну модель, виходячи з наявних ресурсів, забезпечуючи оптимальну продуктивність.

«/videos» – цей каталог має подвійне призначення. Тут зберігаються завантажені користувачами відеофайли у форматі «.mp4», названі у форматі «v_<випадкове_число>.mp4», наприклад, «v_350948142.mp4». Крім того, тут зберігаються зображення, завантажені для аналізу, які називаються так само, але з префіксом «i», наприклад, «i_3483289438.jpg». Така організована система іменування полегшує управління великим обсягом завантаженого користувачем контенту.

«/records» – у цьому каталозі створюються підкаталоги для кожного запису (відео або зображення), що підлягає аналізу. Назви підкаталогів включають ім'я файлу запису та мітку часу в мілісекундах, коли розпочався аналіз, наприклад, «/records/v_350948142_1723823871» або «/records/i_3483289438_1723864871». У кожній підтеці зберігаються зображення всіх виявлених вибоїн у цьому записі. Ці зображення мають назву з префіксом «rec_» та ідентифікатором вибоїни в базі даних, наприклад, «rec_23.jpg». Така структура забезпечує чіткий і організований спосіб зберігання результатів кожного сеансу аналізу.

У базі даних зберігаються записи для всіх папок та імен файлів, пов'язуючи їх з відповідним файлом у сховищі. Сюди входять їхні оригінальні назви та часові позначки, коли вони були завантажені. Така прив'язка до бази даних гарантує, що існує чіткий і доступний шлях від даних, що зберігаються в системі, до їх фізичного представлення у файловому сховищі. Такий підхід не лише оптимізує процес зберігання та пошуку, але й підтримує високий рівень організації та зрозумілості в системі.

4.3 Програмний інтерфейс

Розроблений застосунок використовує надійну архітектурну основу для аналізу та пошуку вибоїн на відео та зображеннях. В основі системи лежать дві

основні функції: аналіз записів і подальша геолокація, діаграма послідовності для яких зображена на рисунку 4.4.

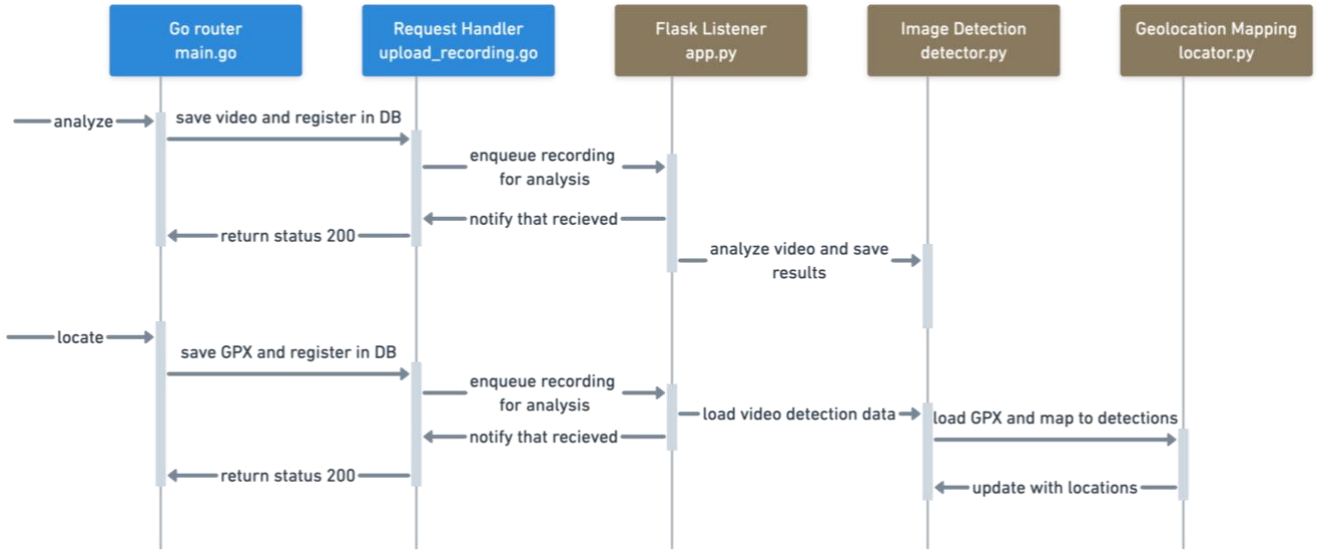


Рисунок 4.4 – Діаграма послідовності для основних функцій системи

За запитом «analyze» система ініціює фоновий процес аналізу відповідного запису (відео або зображення). Аналогічно, за запитом «locate» у фоновому режимі запускається процес картографування геолокації. Така асинхронна обробка гарантує, що сервер швидко відповідає користувачеві, в той час як інтенсивні операції виконуються паралельно.

Для аналізу відео програма зчитує відеофайл і отримує важливі метадані, такі як загальна кількість кадрів і загальна тривалість відео в мілісекундах. З кожною ітерацією над кадрами система зчитує кадр зображення, встановлює лінію виявлення (яка є областю, що представляє інтерес для виявлення вибоїн) і згодом ідентифікує будь-які вибоїни. Вибір лінії виявлення має вирішальне значення і може бути вдосконалений за допомогою передових методів, таких як виявлення горизонту на зображенні. Потім виявлені об'єкти відстежуються кадр за кадром, щоб врахувати можливість появи однієї і тієї ж вибоїни на кількох послідовних кадрах. Цей крок зменшує дублювання і гарантує, що кожна виявлена вибоїна є

унікальною. Для відео, прогрес повідомляється в реальному часі за допомогою індикатора виконання, що забезпечує інтуїтивну візуалізацію процесу аналізу.

Для зображень процес аналізу є відносно простим. Після зчитування зображення система негайно виявляє вибоїни і відстежує їх. Крім того, вона збирає геолокаційну інформацію, вбудовану в зображення, якщо вона доступна, для подальшого процесу геолокації.

Функція «локалізації» починається з перевірки файлу GPX, пов'язаного із записом. Цей файл зберігає географічні дані, що відповідають кожному кадру відео. Після перевірки вона зчитує і обробляє файл, щоб витягти географічні точки з часовими мітками. Для кожної виявленої вибоїни програма обчислює час, що минув, виходячи з положення кадру на відео, а потім інтерполює місцезнаходження, використовуючи найближчі доступні точки даних з файлу GPX. Місцезнаходження кожної виявленої вибоїни зберігається в базі даних як об'єкт «DetectionLocation».

Протягом усього процесу ретельна обробка помилок гарантує, що будь-який несподіваний сценарій або збій буде зареєстровано, а відповідний статус GPX буде оновлено. Такий рівень детального логування допомагає в усуненні несправностей і вдосконаленні системи.

Структура шляхів API наведена далі.

1) створити запис:
    - кінцева точка: «/api/v1/records»;
    - метод: POST;
    - функціональність: полегшує завантаження файлу, який може бути як відео, так і зображенням. Після успішного завантаження вона повертає унікальний ідентифікатор запису для подальших посилань;
2) аналізувати запис:
    - кінцева точка: «/api/v1/recordings/<recording_id>/analyze»;
    - метод: POST;

- функціональність: запускає процес аналізу для конкретного запису, ідентифікованого за унікальним ідентифікатором запису. Надсилає запит на сервер Python Flask для початку процесу виявлення вибоїн;

3) отримати дані про виявлення:
   - кінцева точка: «/api/v1/recordings/<recording_id>/detections»;
   - метод: GET;
   - функціональність: отримати всі виявлені вибоїни разом з їх геолокацією для певного запису;
4) видалення хибних виявлень:
   - кінцева точка: «/api/v1/recordings/<record_id>/detections/<detection_id>»;
   - метод: DELETE;
   - функціональність: дозволяє користувачам видаляти неточно ідентифіковані вибоїни, гарантуючи, що дані про виявлення залишаються точними;
5) завантажити GPX-файл:
   - кінцева точка: «/api/v1/recordings/<recording_id>/gpx»;
   - метод: POST;
   - функціонал: завантажує GPX-файл, який містить дані геолокації, для відеозапису. Це дуже важливо для нанесення на карту виявлених вибоїн з точним географічним розташуванням;
6) отримати мета-інформацію:
   - кінцева точка: «/api/v1/meta/storage»;
   - метод: GET;
   - функціональність: отримання метаданих, таких як місця зберігання завантажених і оброблених файлів. Це гарантує, що користувачі та розробники мають чітке уявлення про фізичне зберігання даних.

Загальні можливості користувача програмного інтерфейсу наведено на діаграмі прецедентів (рис. 4.5). Описані дії відповідають назвам кінцевих точок.

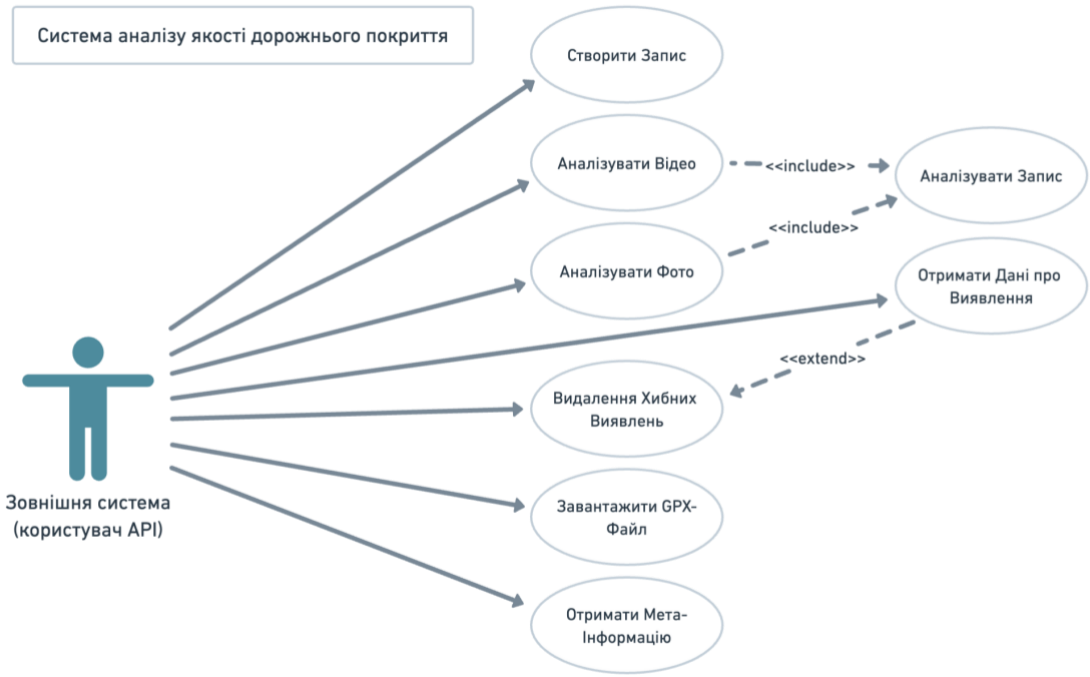


Рисунок 4.5 – Діаграма прецедентів для користувача програмного API

API застосунку на Go пропонує структуровані та ефективні засоби взаємодії з системою, а також забезпечує чіткий інтерфейс інтеграції в інші застосунки. Визначений дизайн гарантує програмний доступ до всіх функціональних можливостей, від завантаження записів до пошуку виявлених порушень і пов'язаних з ними геолокаційних даних.

4.4 Програмне забезпечення вебзастосунку

Вебзастосунок системи забезпечує взаємодію з базою даних та мікросервісом виявлення зображень. Ця частина в основному побудована з використанням Go, який відповідає за внутрішні функції, включаючи з’єднання з базою даних і зв'язок з мікросервісом розпізнавання.

Одним з ключових компонентів вебзастосунку є маршрутизатор Gin, високопродуктивний веб-фреймворк HTTP, що використовується в Go. Маршрутизатор Gin налаштований для управління статичними шляхами, які мають вирішальне значення для відображення зображень і відео на веб-сайті. По суті, він

діє як проксі-сервер до потрібних папок у файловому сховищі, зокрема, до каталогів відео та записів. Таке налаштування гарантує, що коли користувач отримує доступ до певного зображення або відео на сайті, маршрутизатор Gin спрямовує запит до відповідного файлу в системі зберігання, забезпечуючи ефективний пошук і відображення цих медіа файлів (рис. 4.6).

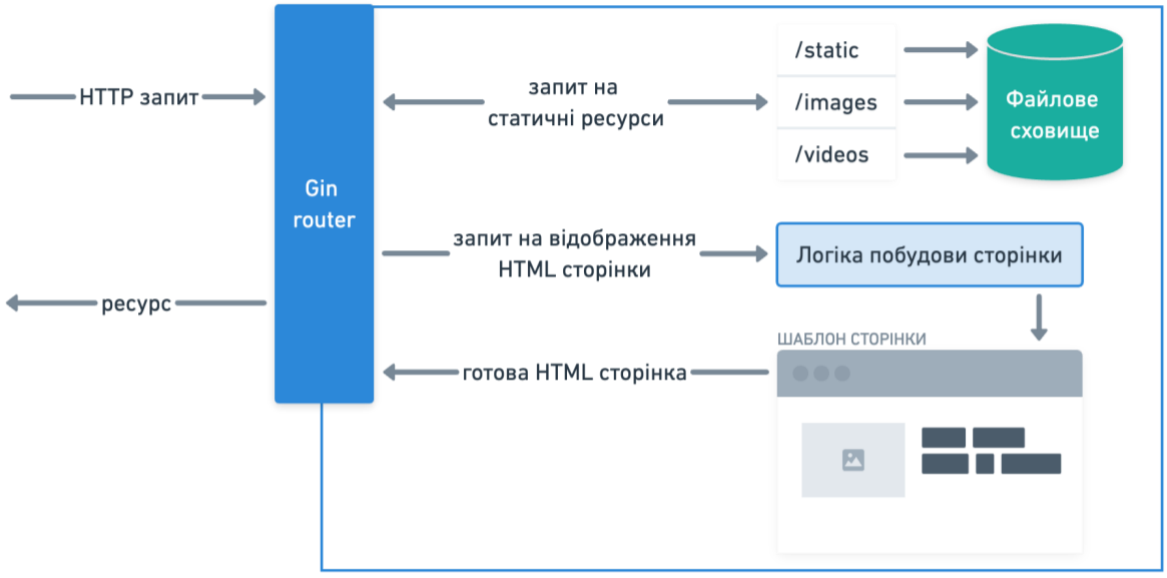


Рисунок 4.6 – Схема роботи маршрутизатора Gin для доступу до ресурсів

На додаток до стандартних функцій, що надаються вебзастосунком, існує спеціалізована кінцева точка, призначена для покращення користувацького досвіду шляхом інтеграції інтерактивних функцій карти. Ця кінцева точка відіграє ключову роль у поверненні геолокаційних даних про вибоїни у форматі JSON, які потім використовуються на HTML-сторінці.

Реалізація цієї функції є спрощеною, але не менш ефективною. Вебсторінка використовує Alpine.js для взаємодії з цією кінцевою точкою. Alpine.js відповідає за запити до кінцевої точки, коли це необхідно, отримуючи геолокаційні дані про вибоїни. Отримавши ці дані, Alpine.js передає їх до Google Maps API, який потім використовує цю інформацію для нанесення вибоїн на карту. Інтеграція з картами – це єдина частина вебзастосунку, яка використовує JavaScript, спеціально для того, щоб скористатися можливостями, які надає Google Maps API.

Таким чином, вебзастосунок системи є складним поєднанням внутрішніх можливостей Go, ефективної обробки статичних шляхів і HTTP-запитів маршрутизатором Gin, а також динамічної інтерактивності, що забезпечується HTMX. Результатом такого поєднання є надійний і зручний вебінтерфейс, який ефективно полегшує взаємодію користувача з основними функціональними можливостями програми.

4.5 Алгоритми аналізу графічних даних

Розробка алгоритму аналізу зображень є важливою складовою мікросервісу Python, який називається «ph-detector». Цей мікросервіс відповідає за складне завдання аналізу записів, які можуть бути як відео, так і зображеннями, для виявлення аномалій дорожнього покриття, таких як вибоїни.

Коли мікросервіс «ph-detector» отримує команду проаналізувати запис, його першим кроком є визначення типу отриманого запису. Ця ідентифікація дуже важлива, оскільки підхід до аналізу відео відрізняється від підходу до аналізу зображень. Відео вимагає по кадрового аналізу, тоді як зображення обробляються як єдине ціле.

Після визначення типу запису мікросервіс переходить до завантаження моделі ШІ в пам'ять. Цей крок є дуже важливим, оскільки він готує систему до аналізу запису за допомогою складних можливостей розпізнавання образів, закладених у моделі ШІ. Завантаження моделі в пам'ять є важливим процесом, який гарантує, що система готова і оснащена для ефективного проведення аналізу.

Після завантаження моделі ШІ «ph-detector» починає обробку запису. У випадку відео мікросервіс аналізує його по кадрам, тоді як для зображення він обробляє всю картинку за один раз. Ця обробка передбачає застосування моделі штучного інтелекту для виявлення вибоїн або інших важливих особливостей на записі.

Після завершення обробки мікросервіс виконує кілька важливих завдань. По-перше, він зберігає виявлені результати. Це може включати запис місця розташування виявлених вибоїн, їхніх розмірів та інших важливих даних, які можуть знадобитися для подальших дій або аналізу. По-друге, мікросервіс генерує журнали стану. Ці журнали містять цінну інформацію про процес обробки, таку як кількість виявлених вибоїн, час обробки, а також будь-які помилки або проблеми, що виникли.

Нарешті, після завершення аналізу запису «ph-detector» перевіряє, чи є наступний елемент у черзі на аналіз. Якщо є ще один запис, який очікує на аналіз, мікросервіс переходить до його обробки, виконуючи ті самі кроки визначення типу запису, завантаження моделі ШІ, обробки запису, збереження результатів і реєстрації статусу.

Механізм аналізу відео в застосунку призначений для ефективного вирішення проблеми дедуплікації – поширеної проблеми, коли один і той самий об'єкт виявляється кілька разів на різних кадрах відео. Для вирішення цієї проблеми було розроблено унікальний алгоритм відстеження вибоїн, що поєднує в собі можливості розпізнавання та відстеження об'єктів.

Суть алгоритму полягає в обробці окремих кадрів відео за допомогою моделі штучного інтелекту, яка ідентифікує вибоїни в кожному кадрі. Однак, оскільки модель ШІ обробляє кожен кадр незалежно, вона не розпізнає, чи була вибоїна виявлена на попередніх кадрах. Саме тут виникає потреба у складному механізмі відстеження, як показано на рисунку 4.7.

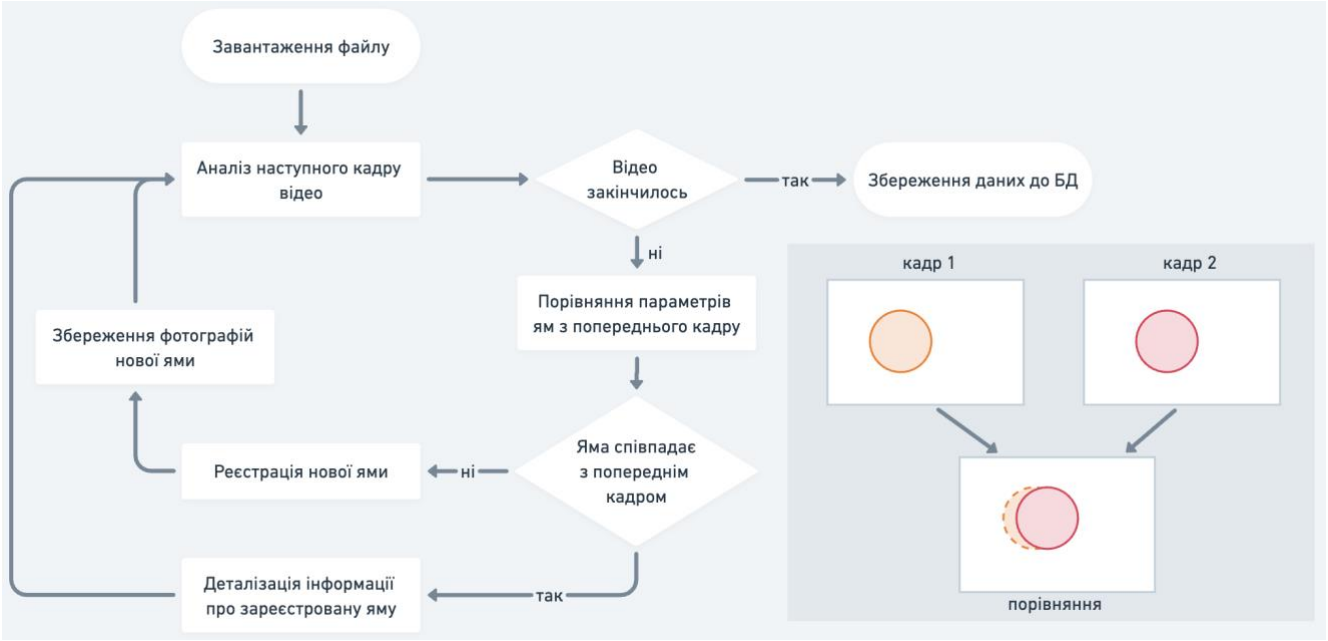


Рисунок 4.7 – Схема алгоритму відстеження унікальних ям

Для відстеження руху вже виявлених на відео вибоїн був реалізований алгоритм SORT (Simple Online and Realtime Tracking – просте відстеження в режимі реального часу) [13].

Алгоримт покадрово аналізує відео на наявність ям, та зберігає їх параметри. Далі виконується передбачення за допомогою фільтру Калмана, та проводиться аналіз наступного кадру. Якщо нові виявлення співпадають з передбаченням, то відбувається оновлення стану. Ями, що не співпали, реєструються як нові.

Таким чином, з використанням цих алгоритмів, поєднанням їх результатів для кожної окремої визначеної ями, можна забезпечувати унікальність виявлень та підвищувати точність. Коли виявлення визначено як зафіксоване (коли воно досягає певної області на відео), воно буде збережено до бази даних, з усіма своїми мета параметрами та зображенням, де воно обведено своєю рамкою для наочності.

Інший алгоритм вирішує проблему визначення географічного положення ями. Це складний процес, який передбачає співвіднесення виявлених вибоїн з географічними даними. Це особливо складно, коли вибоїни виявлені на відеозаписі, оскільки географічне розташування повинно бути точно співвіднесене з кожним кадром відео. Принцип його роботи описано на рисунку 4.8.

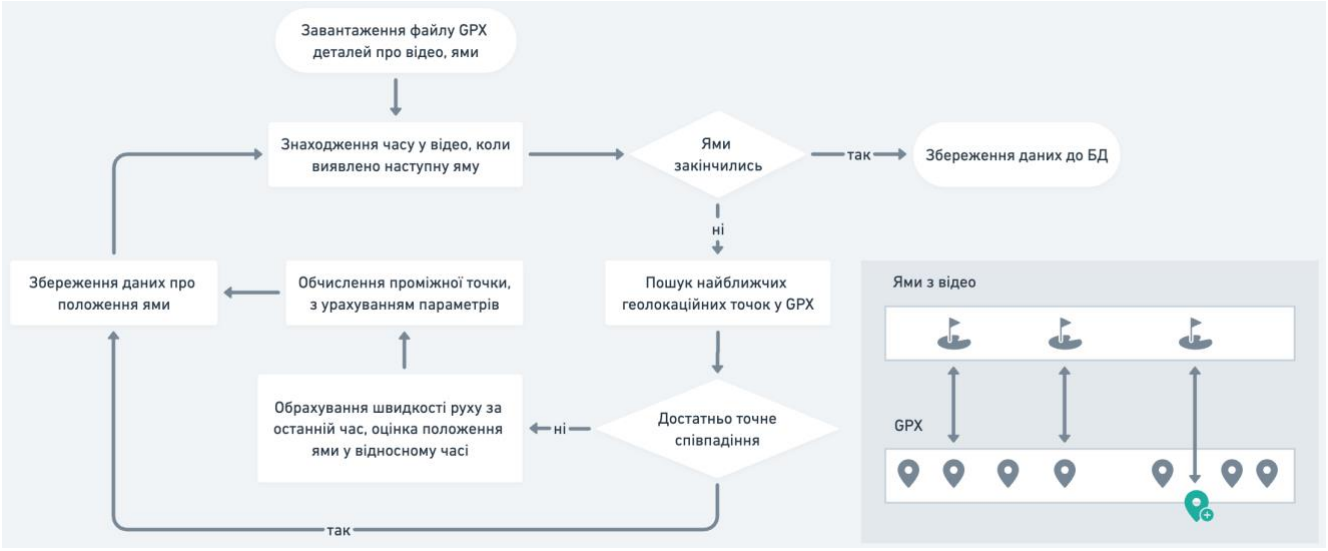


Рисунок 4.8 – Схема алгоритму асоціації географічної інформації з ямами

Алгоритм отримує два основні набори вхідних даних. Перший – це файл, що містить точки GPS-даних, які відображають шлях, пройдений відео оператором під час запису. Ці точки записуються через певні проміжки часу (частота), фіксуючи географічне розташування оператора в часі. Другий вхідний файл – це дані про виявлені вибоїни, включаючи точну позначку часу, коли кожна вибоїна була ідентифікована на відео.

Основним завданням алгоритму є синхронізація часових міток виявлених вибоїн з точками GPS-даних. Він порівнює час виявлення кожної вибоїни з часовими мітками точок GPS.

Якщо мітка часу виявленої вибоїни збігається з міткою часу точки GPS, алгоритм присвоює вибоїні географічні координати цієї точки GPS, припускаючи, що вибоїна знаходиться там, де в цей час перебував відео оператор.

У багатьох випадках може не бути точного збігу між часовими мітками вибоїн і GPS-точок. У таких випадках алгоритм обчислює проміжну точку, щоб оцінити місцезнаходження вибоїни. Цей розрахунок враховує параметри руху оператора, такі як середня швидкість між зафіксованими точками GPS і потенційну похибку в оцінці. Інтерполюючи ці параметри, алгоритм може оцінити найбільш ймовірне географічне розташування вибоїни на момент її фіксації.

Результатом цього процесу є географічна точка, яка відповідає точному розташуванню вибоїни. Ця точка потім використовується для нанесення вибоїни на географічну карту, надаючи цінні дані для технічного обслуговування та ремонту доріг.

Цей алгоритм ефективно долає розрив між відео аналітикою (виявленням ям) та географічним картографуванням, дозволяючи проводити комплексний аналіз стану доріг. Він гарантує, що кожна виявлена вибоїна не тільки ідентифікується на відео, але й точно визначається в реальному світі, що має вирішальне значення для практичних застосувань, таких як планування технічного обслуговування доріг та відстеження якості дорожнього покриття в часі.

Визначити місцезнаходження за фотографією значно простіше, ніж за відео, оскільки фотографія – це статичне зображення без руху. Більшість сучасних цифрових зображень, особливо у форматі JPEG, часто містять метадані, відомі як EXIF. Ці метадані можуть містити різноманітну інформацію про фотографію, включаючи географічні координати – широту і довготу місця, де було зроблено фотографію. Ці GPS-дані вбудовуються у фотографію під час зйомки, як правило, пристроєм, яким вона була зроблена, наприклад, смартфоном або цифровою камерою з функцією GPS. Таким чином, вилучення GPS-координат з фотографії передбачає лише доступ до метаданих EXIF, що робить процес визначення точного географічного місця, де було зроблено фотографію, простим і ефективним.

4.6 Модель штучного інтелекту

Модель YOLOv8 було навчено виявляти вибоїни за допомогою комплексного набору даних, що складається з понад 7000 зображень, отриманих з різних каналів. Набір даних включав зображення з набору даних про вибоїни Roboflow, анотовані вручну зображення з відео з YouTube, а також зображення з набору даних RDD2022. Після кількох раундів корекції анотацій фінальний набір даних складався з 6963 навчальних зображень і 272 валідаційних зображеннь.

Навчання моделі YOLOv8 на цьому користувацькому наборі даних вибоїн викликало значні труднощі через різні розміри вибоїн, від малих до великих. Для процесу навчання було ретельно підібрано декілька гіперпараметрів:

- кожна модель навчалася на 50 епохах. Враховуючи великий розмір набору даних, очікувалося, що така тривалість дасть пристойні результати при обмеженому навчанні;
- щоб забезпечити узгодженість між експериментами, розмір партії був встановлений на рівні 8 для всіх моделей;
- враховуючи малий розмір деяких вибоїн на зображеннях, під час навчання роздільну здатність зображення було встановлено на 1280, що перевищує стандартну роздільну здатність 640. Це налаштування мало на меті покращити результати виявлення, хоча і ціною збільшення часу навчання.

Першою пройшла навчання модель YOLO8 Nano, як найкомпактніший представник сімейства YOLOv8 з 3.2 мільйонами параметрів. Незважаючи на свій відносно невеликий розмір, вона продемонструвала здатність працювати в режимі реального часу навіть на центральному процесорі. Навчання, яке охоплювало 50 епох з роздільною здатністю зображення 1280 і розміром партії 8, дозволило досягти значних показників точності. Модель досягла максимальної середньої точності 40.2 mAP при 0.50 IoU і 33.5 mAP при 0.50-0.95 IoU, продемонструвавши свою ефективність у виявленні вибоїн, незважаючи на свою компактну архітектуру.

Слідом за моделлю Nano, модель YOLOv8 Small пройшла навчання, досягнувши вражаючих результатів точності. Модель майже досягла значення mAP 46.2 при 0.50 IoU. Фактично зафіксований показник mAP склав 47, що є значним досягненням для моделі такого масштабу, значною мірою завдяки навчанню з високою роздільною здатністю.

Нарешті, модель YOLOv8 Medium була навчена, досягнувши порівнянних, якщо не вищих, результатів точності. Модель Medium також досягла значення mAP майже 45.3, а фактичне значення mAP становило 46 за 50 епох. Це було лише трохи нижче, ніж у малої моделі, але все одно свідчить про високу ефективність процесу

навчання і параметрів. Порівняння точності моделей від кількості ітерацій можна побачити на рисунку 4.9.

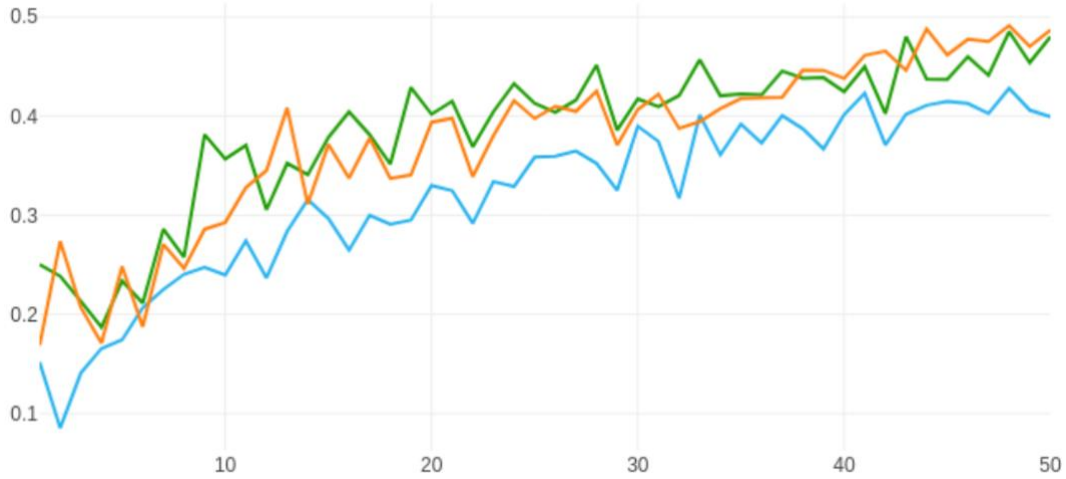


Рисунок 4.9 – Графік залежності точності моделей від кількості ітерацій. Умовні позначення: зелена лінія – модель Medium; синя лінія – модель Nano; помаранчева лінія – модель Small

Навчання високоточних моделей для виявлення вибоїн у застосунку стикається з помітною проблемою через обмежену доступність навчальних даних. З обмеженим набором даних навчання моделей для досягнення надзвичайно високого рівня точності може бути складним. Однак у програмі реалізовано функції, які пом'якшують це обмеження і сприяють постійному вдосконаленню моделей.

Однією з ключових функцій є можливість для адміністраторів вручну переглядати і видаляти помилкові виявлення. Ця можливість має вирішальне значення, оскільки вона дозволяє вдосконалити результати, отримані за допомогою моделей ШІ.

Крім того, програма призначена для зберігання зображень виявлених вибоїн. Це сховище зображень слугує подвійній меті. По-перше, він надає адміністраторам і користувачам точку відліку для перевірки і розуміння виявлених ШІ-моделей. По-друге, що більш важливо, ці зображення можна використовувати як новий набір даних для подальшого навчання і вдосконалення існуючих моделей.

Накопичення додаткових даних зображень з часом, особливо тих випадків, коли результати роботи моделі були скориговані адміністраторами, надає цінну можливість для перенавчання та вдосконалення моделі. Такий підхід відповідає принципам машинного навчання, де моделі ітеративно вдосконалюються шляхом навчання на нових і різноманітних наборах даних.

Висновки до розділу 4

У четвертому розділі, наведено детальний огляд програмного забезпечення для системи аналізу якості дорожнього покриття, описано компоненти та методології, що були використані при його створенні. Такі ключові області, як розробка сховища даних, де детально розглядається структура бази даних та управління файлами, а також розробка програмного забезпечення API.

Описано розробку програмного забезпечення для вебзастосунку, включаючи опис доступних функцій. Значну увагу було приділено розробці алгоритмів аналізу медіа даних для аналізу відео. Це включало складні методи відстеження ям під час відтворення відео та методи визначення геолокації ям як на відео, так і на фотографіях.

Крім того, описано розробку моделі штучного інтелекту, висвітлено процес збору даних, навчання та отриману в результаті точність моделі у визначенні ям. Всі ці елементи разом сприяють надійності та ефективності системи у вирішенні проблем виявлення та аналізу ям.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Системні вимоги

Опишемо ключові технічні параметри, необхідні для ефективної роботи системи аналізу якості дорожнього покриття.

Для забезпечення швидкого доступу до відеоданих і їхньої обробки, система повинна бути оснащена твердотільним диском (SSD). SSD забезпечує значно вищу швидкість читання та запису даних порівняно з традиційними жорсткими дисками, що критично важливо для обробки великих об'ємів відеозаписів.

Для підключення до мережі інтернет необхідна швидкість з'єднання не менше 1 гігабіта, щоб забезпечити швидкий обмін даними з сервером та іншими компонентами системи.

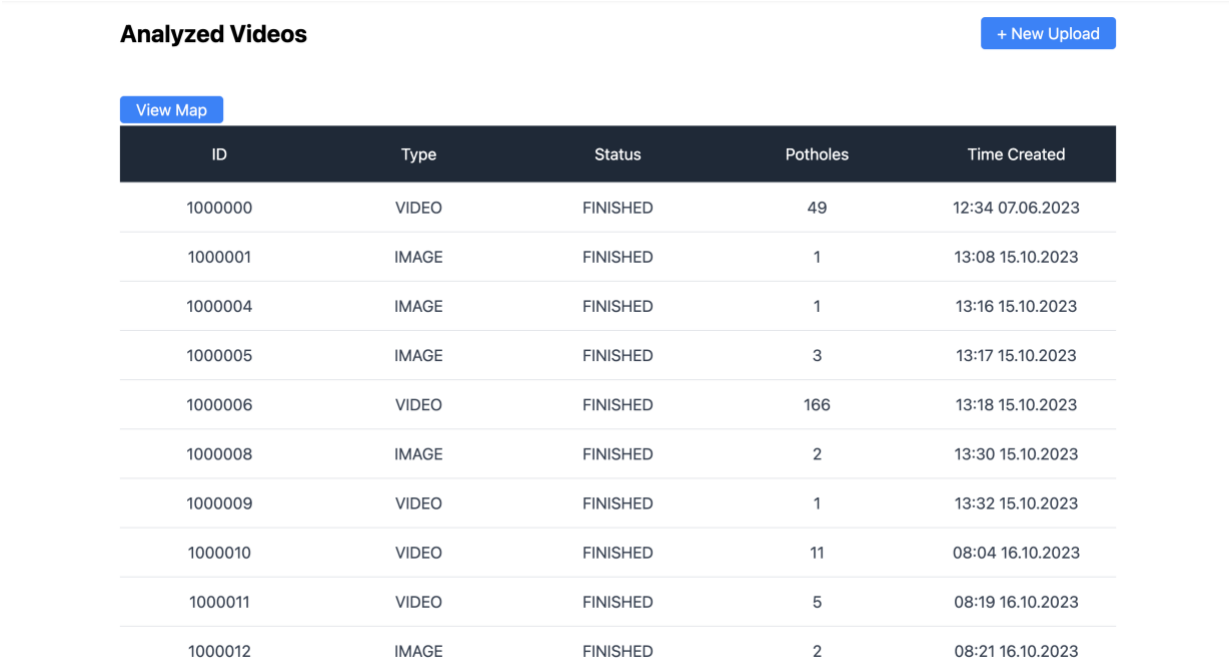
Важливим компонентом системи є відеокарта з підтримкою технологій CUDA або MPS, оскільки це забезпечує оптимальну продуктивність при роботі з моделями штучного інтелекту, які використовуються для розпізнавання дефектів дорожнього покриття. Мінімальний обсяг відео пам’яті складає 2 ГБ (гігабайти) для забезпечення достатньої обчислювальної потужності при роботі з великими зображеннями та відео.

Завдяки використанню контейнеризації з Docker, вибір операційної системи не є критичним, оскільки Docker дозволяє запускати програмне забезпечення в ізольованому середовищі, незалежно від ОС сервера.

Оптимальний об’єм оперативної пам'яті складає 8 ГБ. Це обумовлено вимогами до обробки відео, яке вимагає значного обсягу пам'яті. В свою чергу мікросервіс на мові програмування Go, максимально оптимізований та вимагає не більше ніж 512 МБ (мегабайт) оперативної пам'яті для швидкої роботи.

5.2 Вебзастосунок для аналізу якості дорожнього покриття

При вході в систему користувач спочатку бачить інформаційну панель з переліком усіх завантажених записів. Макет, організований у вигляді таблиці, надає цілісне уявлення про унікальний ідентифікатор кожного запису, формат (відео або зображення), статус аналізу, кількість виявлених вибоїн та час створення запису (рис. 5.1). Ця сторінка слугує централізованим центром для користувачів, де вони можуть швидко переглянути хід і результати завантаження.



ID	Type	Status	Potholes	Time Created
1000000	VIDEO	FINISHED	49	12:34 07.06.2023
1000001	IMAGE	FINISHED	1	13:08 15.10.2023
1000004	IMAGE	FINISHED	1	13:16 15.10.2023
1000005	IMAGE	FINISHED	3	13:17 15.10.2023
1000006	VIDEO	FINISHED	166	13:18 15.10.2023
1000008	IMAGE	FINISHED	2	13:30 15.10.2023
1000009	VIDEO	FINISHED	1	13:32 15.10.2023
1000010	VIDEO	FINISHED	11	08:04 16.10.2023
1000011	VIDEO	FINISHED	5	08:19 16.10.2023
1000012	IMAGE	FINISHED	2	08:21 16.10.2023

Рисунок 5.1 – Сторінка всіх завантажених записів

При натисканні на кнопку «+ New Upload», користувач знаходить спеціальне місце для створення нових завантажень. Розроблена для простоти та ефективності, ця сторінка дозволяє завантажувати як зображення, так і відеофайли. Додаткове текстове поле дозволяє користувачам коментувати або надавати примітки щодо файлу, який вони завантажують. Кнопка «Create recording», розміщена під формою, запускає аналіз. Але перед тим, як дійти до цього кроку, надійний механізм перевірки системи перевіряє наявність будь-яких розбіжностей або помилок, гарантуючи, що користувач завжди має чітке уявлення про стан завантаження, що

можна побачити на рисунку 5.2. Якщо виникають якісь помилки, система запам'ятовує попередні дії користувача, та описує етап, на якому виникли складності у вигляді жовтого інформаційного вікна, що робить виправлення помилок безпроблемним. Можливі помилки, що контролюються системою налічують неправильний формат медіа файлів, відсутність файлів, відсутність зв’язку з базою даних або файловим сховищем.

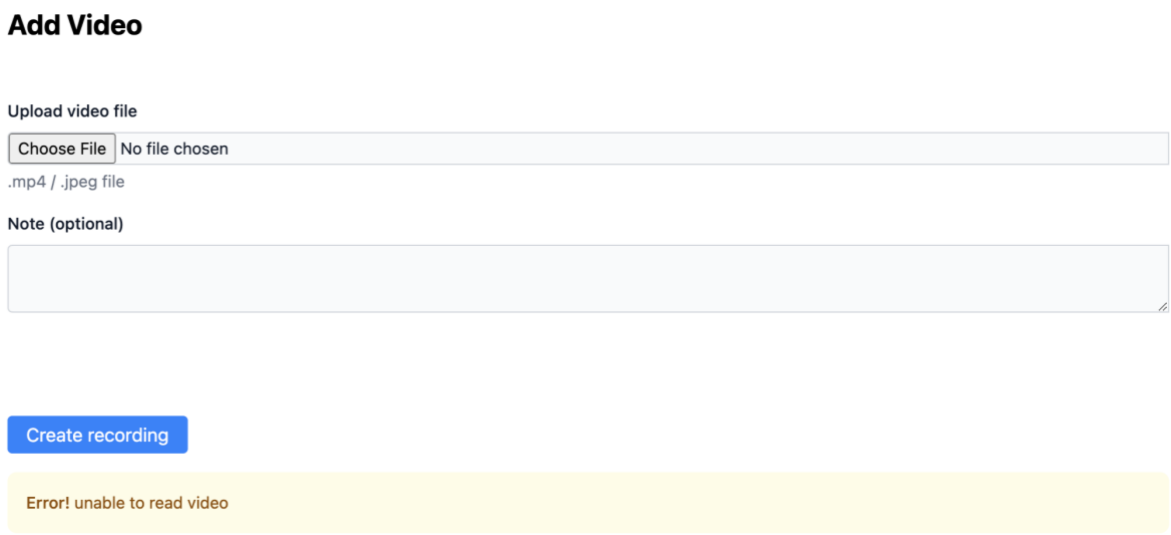


Рисунок 5.2 – Сторінка для створення нового завантаження

Кожен відеозапис, після завантаження, має свою окрему сторінку. Тут користувачі можуть відстежувати статус аналізу в реальному часі і стежити за ходом виконання GPX-файлу. Якщо система виявляє відсутність GPX-файлу, вона пропонує користувачам легкодоступну можливість завантажити його. Окрім функціональних елементів, сторінка також демонструє оригінальний попередній перегляд відео, що підвищує залученість користувачів (рис. 5.3). Унікальною особливістю є діаграма статистики, яка візуально відображає розподіл вибоїн на часовій шкалі відео. Вертикальна вісь відображає кількість ям у конкретний момент часу, а горизонтальна вісь відповідає загальному часу відео.

Для більш детального аналізу кожна виявлена вибоїна відображається у вигляді списку із зображенням дефекту зліва, та детальною інформацією справа, що складається з ідентифікатора виявлення, геолокації та оцінки достовірності. Якщо

якесь виявлення здається неправильним, користувачі можуть видалити його, натиснувши на червону кнопку.

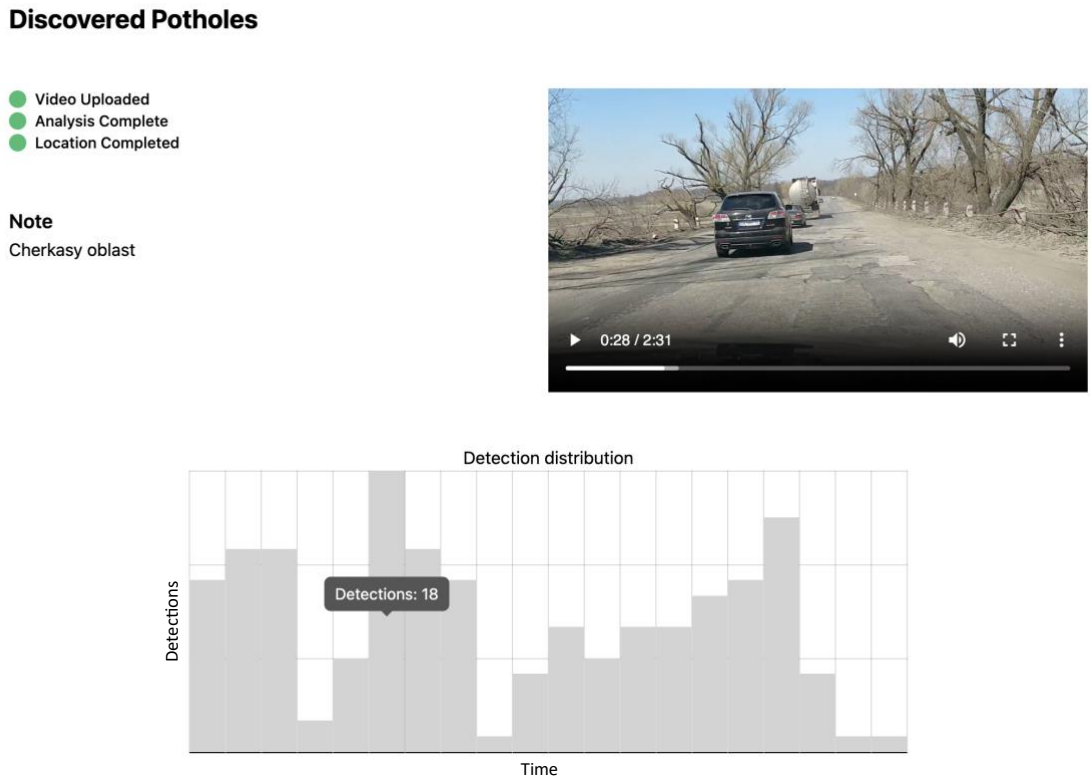


Рисунок 5.3 – Сторінка для перегляду відеозапису

Процес перегляду зображень подібний до процесу перегляду відео, але враховує нюанси нерухомих зображень. Коли користувачі отримують доступ до цієї сторінки, вони отримують високоякісний попередній перегляд оригінального зображення.

Одночасно з цим, статус оновлення інформує про етап обробки. Кожна виявлена вибоїна ретельно окреслена, з акцентом на її точному розташуванні на зображенні. Як і у випадку з відео, кожне виявлення супроводжується такими деталями, як ідентифікатор, геолокація та оцінка достовірності. І, звичайно, будь-які невідповідності можуть бути виправлені користувачем за допомогою опції видалення, червона кнопка на рисунку 5.4.

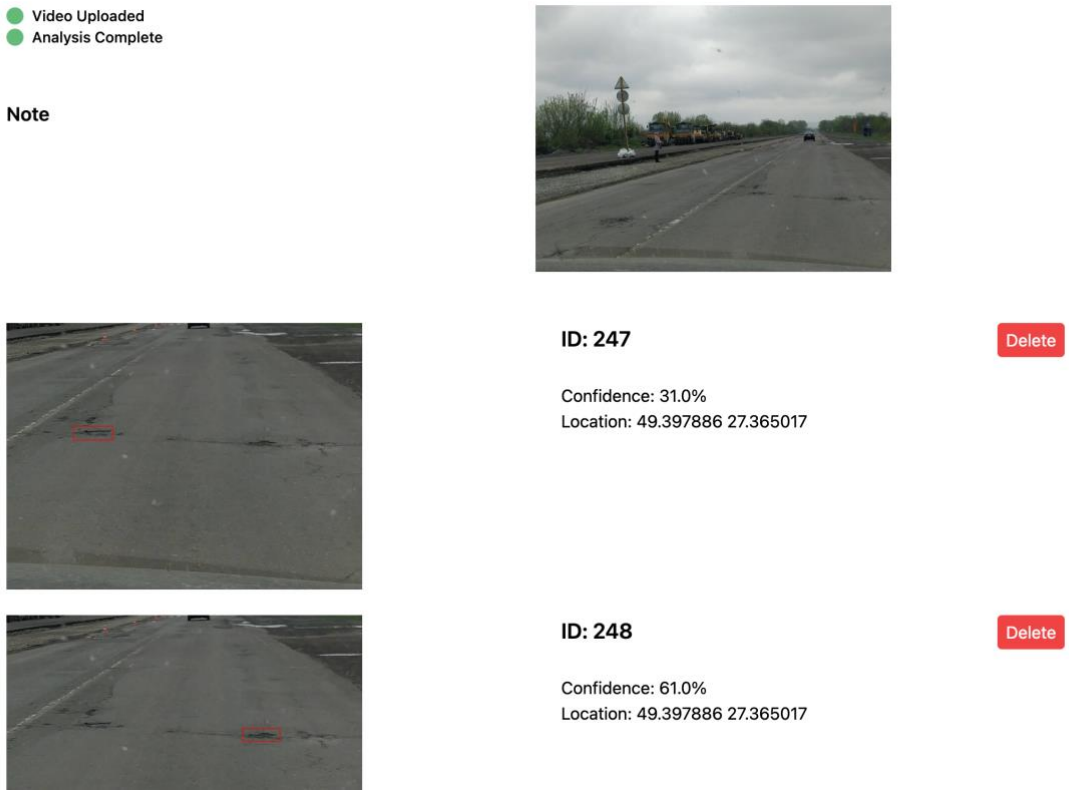


Рисунок 5.4 – Сторінка для перегляду зображення

Переходячи від статичних списків та окремих записів, програма вводить інтерактивний елемент – сторінку карти. Використовуючи можливості Google Maps, ця функція відображає всі виявлені вибоїни в різних географічних регіонах. Користувачі можуть інтуїтивно взаємодіяти з картою, а після вибору будь-якої точки отримують детальну інформацію про вибоїни в цьому конкретному місці. Сюди входять зображення, ідентифікатори записів, ідентифікатори виявлення, місцезнаходження та оцінка достовірності (рис. 5.5). Ця сторінка пропонує не лише просторову інформацію, але й цікавий спосіб для користувачів візуалізувати розподіл та щільність дефектів на дорогах.

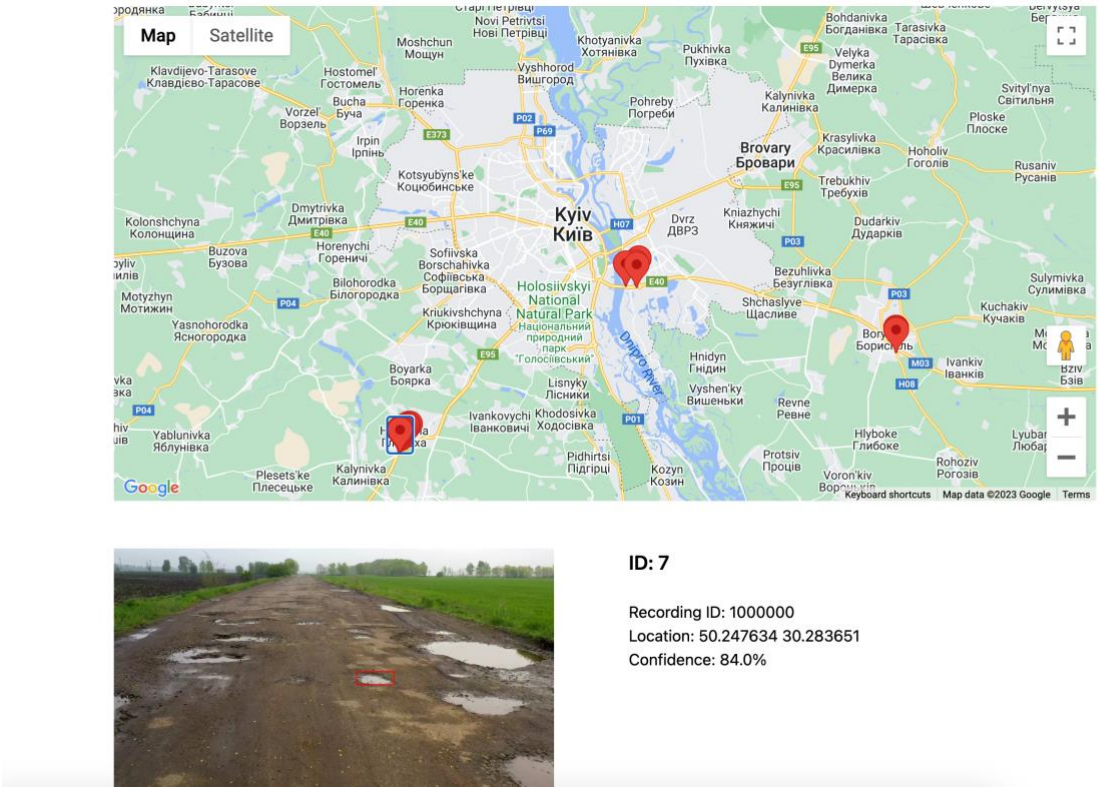


Рисунок 5.5 – Сторінка мапи

Фактично, програмне забезпечення пропонує всебічний, інтуїтивно зрозумілий та інтерактивний досвід. Кожна сторінка продумана з урахуванням зручності та ефективності роботи користувача, що робить аналіз дефектів доріг не складним завданням, а швидкою подорожжю.

Висновки до розділу 5

У п’ятому розділі наведено детальний опис вимог до системи, що запускає розроблений вебзастосунок та проведено огляд роботи користувача з системою для аналізу якості дорожнього покриття.

Описано процес взаємодії користувача з вебзастосунком для аналізу фото та відео файлів на наявність дефектів дорожнього покриття. Перелічені усі доступні сторінки системи, їх призначення, доступні для використання функціональності та представлена на цих сторінках інформація.

6 РОЗРОБКА СТАРТАП-ПРОЄКТУ

6.1 Опис ідеї проєкту

Основною ідеєю стартап-проєкту є створення автоматизованої системи для аналізу якості дорожнього покриття. Цей проєкт має за мету розробку програмного забезпечення, яке використовує передові технології штучного інтелекту та комп'ютерного зору для ідентифікації та аналізу дефектів доріг, таких як ями, тріщини та інші недоліки. Можливі напрями застосування та вигоди для користувачів наведені у таблиці 6.1.

Таблиця 6.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувачів
Розробка і впровадження інструментальних засобів для автоматизованого аналізу якості дорожнього покриття	Автоматизація процесів оцінки та моніторингу стану дорожнього покриття	Підвищення точності та об'єктивності оцінки стану доріг, зниження витрат часу та ресурсів на проведення оцінок
	Оптимізація робочих процесів в організаціях, що займаються обслуговуванням доріг	Ефективне планування та розподіл ресурсів для ремонту та обслуговування дорожньої інфраструктури
	Використання органами держави для управління дорожньою інфраструктурою	Забезпечення прозорості та оперативності отримання інформації про стан доріг, поліпшення умов для прийняття управлінських рішень.

Основною відмінністю та перевагою цього проєкту від існуючих рішень на ринку, таких як візуальний огляд людьми або системи на основі сенсорів прискорення, є використання комплексного підходу до аналізу доріг. Наш проєкт надає більш точну та об'єктивну оцінку стану дорожнього покриття, використовуючи автоматизовані методи і ШІ, що забезпечує високу ефективність і оперативність обробки великих обсягів даних. Водночас, система має здатність адаптуватися до різних умов і типів дорожнього покриття, що робить її універсальним інструментом для широкого спектру застосувань.

Таблиця 6.2 містить опис переваг та недоліків кожної системи в порівнянні з іншими, надаючи обґрунтований вибір на користь запропонованої системи.

Таблиця 6.2 – Порівняння можливостей систем серед основних параметрів

№ п/п	Критерій оцінки	Система на основі візуального огляду людьми	Система на основі датчиків прискорення	Моя система
1.	Точність розпізнавання	висока	низька	висока
2.	Швидкість обробки	низька	висока	висока
3.	Універсальність	висока	низька	висока
4.	Вартість впровадження	висока	низька	середня
5.	Можливість масштабуватися	низька	середня	висока

6.2 Технологічний аудит проєкту

Для оцінки технологічної здійсненності проєкту з аналізу якості дорожнього покриття необхідно визначити наявність та доступність використовуваних технологій. Нижче наведена таблиця 6.3, що відображає цю оцінку.

Таблиця 6.3 – Технологічна здійсненність проєкту

№ п/п	Складова проєкту	Технології реалізації	Наявність технологій	Доступність технологій
1.	База даних для зберігання інформації про виявлені дефекти	MySQL	Наявна	Вільно доступна, відкрите програмне забезпечення
2.	Контейнеризація для ізоляції та управління середовищем виконання	Docker	Наявна	Вільно доступний, частково відкрите програмне забезпечення
3.	Розробка серверної частини системи	Go	Наявна	Вільно доступна, відкрите програмне забезпечення
4.	Розробка моделі штучного інтелекту для аналізу відео та зображень	YOLOv8	Наявна	Вільно доступна, відкрите програмне забезпечення
5.	Розробка сервісу аналізу покриття	Python	Наявна	Вільно доступна, відкрите програмне забезпечення

З цієї таблиці видно, що всі технології, необхідні для реалізації проєкту, наявні та вільно доступні. Більшість з них є відкритим програмним забезпеченням, що робить їх використання безкоштовним та гнучким. Це дозволяє зробити висновок про можливість технологічної реалізації проєкту з аналізу якості дорожнього покриття.

6.3 Аналіз ринкових можливостей стартап-проєкту

Ринкові тенденції показують зростаючу увагу до безпеки дорожнього руху та ефективності використання бюджетних коштів на ремонт доріг, що стимулює попит на інноваційні технології в цій сфері. Збільшення інвестицій у цифрову трансформацію муніципальних послуг також сприяє зацікавленості в таких проєктах.

Наслідки незадовільного стану доріг не обмежуються лише деградацією фізичної інфраструктури, але й мають серйозні соціальні та економічні наслідки. Згідно з останніми статистичними даними, кількість смертей, спричинених дорожньо-транспортними пригодами, зросла майже на 7% з січня по травень 2022 року порівняно з аналогічним періодом 2021 року [23]. Лише за перші п'ять місяців 2022 року в аваріях на дорогах загинуло понад 2000 людей. Аналіз конкретних випадків деяких країн показує, що понад 10% цих аварій були спричинені поганим станом доріг, зокрема вибоїнами та тріщинами.

Загальна статистика показує, що відсоток таких ДТП значний та якість доріг опосередковано впливає на дії водіїв, після яких відбувається дорожньо-транспортна пригода. Як приклад, детальне вивчення статистики ДТП з 2018 по 2020 рік підтверджує ці занепокоєння [24].

Ці дані підкреслюють причини занепокоєння суспільства, та підтверджують присутність місця на ринку для систем, що можуть допомогти покращити ситуацію.

Для аналізу ринкових можливостей та визначення потенційних загроз важливо розуміти ринкове середовище, потреби цільової аудиторії та конкурентне

положення стартапу. В таблиці 6.4 визначено потенційні групи клієнтів та сформовано вимоги до продукту.

Таблиця 6.4 — Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Вимоги споживачів до продукту
1.	Автоматизація аналізу дорожнього покриття	Державні та муніципальні служби	Висока точність аналізу, простота використання
2.	Ефективний моніторинг та планування ремонтних робіт	Інженерно-дорожні компанії	Швидкість обробки даних, можливість інтеграції з іншими системами

Аналіз ринку включає визначення факторів, що сприяють та перешкоджають ринковому впровадженню проєкту (табл. 6.5, 6.6).

Таблиця 6.5 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція	Поява конкурентів зі схожим функціоналом	Розвиток унікальних характеристик продукту
2.	Зміни у законодавстві	Введення нових правил або обмежень для технологій ШІ	Проактивне відстеження законодавчих змін та адаптація продукту

Таблиця 6.6 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Високий попит	Зростаюча потреба в автоматизації аналізу дорожнього покриття	Розширення ринку, просування продукту
2.	Технологічний прогрес	Розвиток технологій ШІ та обробки зображень	Впровадження інноваційних рішень та оновлення продукту

На основі цього аналізу можна сформувати стратегію розвитку продукту, враховуючи ринкові можливості та загрози, а також зосередитися на задоволенні потреб цільових груп клієнтів.

На основі аналізу конкурентного середовища на ринку інструментальних засобів для аналізу якості дорожнього покриття, складена таблиця 6.7, яка допомагає визначити різні аспекти конкуренції та можливі стратегічні реакції компанії для підтримки конкурентоспроможності.

Таблиця 6.7 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства
1	2	3
Тип конкуренції: чиста	На ринку присутні декілька компаній зі схожими продуктами	Підтримка високої якості продукту та постійне вдосконалення

Продовження таблиці 6.7

1	2	3
Галузева ознака: міжгалузева	Використання продукту в різних галузях	Розширення функціональності продукту для задоволення потреб різних галузей
Рівень конкурентної боротьби: глобальний	Конкуренція з міжнародними компаніями	Розвиток продукту та вихід на міжнародні ринки
Конкуренція за видами товарів: товарно-видова	Конкуренція між схожими продуктами на ринку	Нововведення та вдосконалення продукту для відмінності від конкурентів
Характер конкурентних переваг: нецінова	Фокус на унікальних функціональних можливостях	Розвиток інноваційних функціональних можливостей продукту
Інтенсивність: не марочна	Більша вага якості продукту, ніж бренду	Зосередження на високій якості та надійності програмного продукту

Ця таблиця демонструє, що для успішного конкурентного позиціонування на ринку інструментальних засобів аналізу дорожнього покриття необхідно фокусуватися на якості продукту, його унікальних функціональних можливостях, а також стратегічно планувати вихід на нові ринки і взаємодію з різними галузями.

Для аналізу ринкових можливостей інструментальних засобів для аналізу якості дорожнього покриття, складена SWOT-матриця (табл. 6.8). Такий SWOT-аналіз дозволяє всебічно оцінити перспективи проєкту, виявити потенційні виклики та скласти стратегію для успішної реалізації та розвитку проєкту [25].

Таблиця 6.8 – Матриця SWOT аналізу інструментальних засобів аналізу дорожнього покриття:

Сильні сторони (Strengths)	Слабкі сторони (Weaknesses)
- Використання ШІ для точного розпізнавання дефектів - Геолокація дефектів для точного відображення на карті - Автоматизація процесу аналізу, економія часу та ресурсів - Інтерактивний вебінтерфейс для зручного доступу користувачів	- Необхідність високоякісних вхідних даних для аналізу - Залежність від якості відео та фото матеріалів - Високі вимоги до обладнання для обробки даних - Потреба у постійному оновленні та підтримці програмного забезпечення
Можливості (Opportunities)	Загрози (Threats)
- Зростаюча потреба у автоматизації аналізу дорожнього покриття - Потенціал для розширення на нові ринки та різні типи доріг - Використання результатів аналізу для планування ремонтних робіт - Можливість інтеграції з існуючими системами управління дорожньою інфраструктурою	- Конкуренція з іншими розробниками аналогічних систем - Високі витрати на розвиток та підтримку системи - Технічні виклики у забезпеченні точності та надійності аналізу - Зміни в законодавстві та стандартах, що впливають на використання системи

Загалом, аналіз показав, що проєкт має високий потенціал для здобуття позицій на ринку, особливо завдяки його інноваційному характеру та відповідності актуальним ринковим потребам.

Були розглянуті різні альтернативи ринкового впровадження. Кожна з них оцінюється за ймовірністю отримання ресурсів та строками реалізації. Важливим аспектом є вибір оптимального підходу, який забезпечує максимальну ефективність

при впровадженні проєкту на ринок. Результати розгляду можна переглянути на таблиці 6.9.

Таблиця 6.9 — Альтернативи ринкового впровадження стартап-проєкту:

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Проведення демо-конференції для міжнародного ринку	70%	9-12 місяців
2.	Безкоштовне розповсюдження створеного продукту	45%	10 місяців
3.	Розповсюдження продукту за підпискою	80%	19 місяців

Виходячи з проведеного аналізу, рекомендується розпочати з розповсюдження продукту за підпискою, оскільки цей варіант пропонує найвищу ймовірність успіху та ефективності ресурсів з відносно швидким строком реалізації.

6.4 Розроблення ринкової стратегії продукту

Визначення потенційних груп клієнтів включало аналіз різних сегментів ринку, таких як міські адміністрації, служби дорожнього обслуговування та приватні компанії, що займаються ремонтом та утриманням доріг. Особлива увага була приділена адаптації продукту до специфічних потреб кожної групи клієнтів.

Стратегії залучення та утримання клієнтів включають активні маркетингові кампанії, налагодження відносин з ключовими клієнтами та надання якісного

сервісу після продажу. Для залучення нових клієнтів передбачено проведення демонстрацій ефективності системи, а також організацію семінарів та вебінарів.

У цілому, розроблена ринкова стратегія забезпечує збалансований підхід до виходу на ринок, підкреслюючи сильні сторони продукту, водночас враховуючи потенційні виклики та забезпечуючи ефективні шляхи залучення та утримання клієнтів.

В контексті розробки інструментальних засобів для аналізу якості дорожнього покриття, потенційні групи клієнтів можуть включати міські адміністрації, служби дорожнього обслуговування та приватні компанії. Для цільової аудиторії та їх характеристик можна скласти таблиці 6.10, 6.11.

Таблиця 6.10 – Характеристика потенційних клієнтів

№ п/п	Потенційні клієнти	Готовність до сприйняття продукту	Очікуваний попит	Інтенсивність конкуренції	Простота входу в сегмент
1.	Міські адміністрації	Висока	Високий	Середня	Середньо
2.	Служби дорожнього обслуговування	Середня	Високий	Висока	Важко
3.	Приватні компанії (ремонт та утримання доріг)	Низька	Середній	Висока	Важко

Таблиця 6.11 – Оцінка конкуренції та стратегії розвитку

№ п/п	Обрана стратегія розвитку	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
1.	Співпраця з міськими адміністраціями	Вибірковий розподіл	Новітні технології, точність даних, автоматизація	Диференціація
2.	Співпраця зі службами дорожнього обслуговування	Концентрований розподіл	Ефективність виявлення дефектів, оптимізація витрат	Фокусування
3.	Співпраця з приватними компаніями	Вибірковий розподіл	Гнучкість рішення, інтеграція з іншими системами	Диференціація

Ці таблиці надають уявлення про ринкову стратегію та потенційну базу клієнтів, а також їх готовність до впровадження такого роду технологій.

6.5 Розроблення маркетингової програми стартап проєкту

Маркетингова програма для стартап-проєкту з автоматизації аналізу якості дорожнього покриття була розроблена з метою створення сильного бренду та ефективного просування продукту на ринку. Основним фокусом маркетингової стратегії є підвищення обізнаності потенційних клієнтів про унікальні переваги та функціональні можливості продукту, а також демонстрація його ефективності та вартості для цільових аудиторій [26].

Програма охоплює ряд маркетингових ініціатив, спрямованих на залучення уваги та інтересу до продукту. Одним з аспектів є присутність у цифрових медіа,

включаючи розробку та підтримку вебсайту, на якому будуть представлені інформація про продукт, відгуки користувачів, новини та оновлення проєкту.

Крім того, важливим елементом програми є проведення рекламних кампаній у соціальних мережах та інших цифрових платформах, з метою залучення потенційних клієнтів і підвищення брендової свідомості. Зусилля будуть спрямовані на створення цікавого та привабливого контенту, який би висвітлював ключові переваги продукту та його вплив на покращення інфраструктури доріг.

Маркетингова програма в цілому націлена на створення стійкого попиту на продукт, зміцнення репутації бренду та забезпечення довгострокового зростання та успіху проєкту на ринку.

Висновки до розділу 6

У розділі, присвяченому розробці стартап-проєкту показано, що проєкт виявився обґрунтованим та перспективним. Детальний аналіз усіх аспектів розробки, від технологій та ринкових можливостей до стратегічного маркетингу, показав, що проєкт має суттєвий потенціал для успішного впровадження на ринок.

Проєкт використовує інноваційні підходи та технології, що робить його здатним задовольняти актуальні потреби в сфері дорожнього господарства. Технологічний аудит виявив сильні сторони розробки, включно з використанням ШІ та машинного навчання для ефективного аналізу дорожнього покриття, а також вказав на потенційні виклики, які потребують подальшого вирішення.

Ринковий аналіз підтвердив наявність значного попиту на такі технології, особливо з боку державних органів. Розроблена маркетингова стратегія та програма покликані ефективно підкріпити впровадження продукту на ринок, забезпечуючи його розпізнаваність та привабливість для потенційних клієнтів.

У цілому, стартап-проєкт демонструє, що система має всі необхідні складові для успіху: інноваційність, ринкову актуальність, добре продуману маркетингову стратегію та потенціал для реалізації на великому ринку.

ВИСНОВКИ

В ході роботи було створено інструментальні засоби для аналізу дорожнього покриття на основі машинного навчання.

Розроблено та навчено нейромережеву модель на основі YOLOv8, оптимізовану для точного виявлення дефектів дорожнього покриття. Ця модель демонструє високу точність, швидкість і ефективність у виявленні обраного основного типу дефектів, як ями на дорогах з різного покриття.

Розроблено вебзастосунок, який не лише забезпечує інтерактивну платформу для завантаження та аналізу медіа файлів, але й інтегрує розроблену модель у комплексну систему. Цей застосунок відкриває шляхи для легкого та ефективного використання системи кінцевими користувачами, забезпечуючи їм швидкий доступ до інструментів для визначення стану дорожнього покриття, можливість ручної модифікації даних створених системою. Крім цього, був розроблений власний алгоритм, який поєднує використання ШІ-моделі з підходами для відстеження, з метою забезпечення унікальності розпізнаних пошкоджень дорожнього покриття. Цей алгоритм ефективно ідентифікує та відстежує кожне пошкодження, підвищуючи точність і уникаючи дублювання даних при аналізі відео або зображень.

Розроблено алгоритми, що виконують асоціювання дефектів дорожнього покриття, знайдених у відео, з їх точною географічною локацією, що дозволяє не тільки ідентифікувати дефекти, але й точно відображати їх розташування на карті. Це забезпечує додаткову цінність для планування ремонтних робіт та стратегічного управління дорожньою інфраструктурою держави.

Розроблено програмний інтерфейс для інтеграції системи у різноманітні програмні середовища, що значно розширює її функціональність і застосування. Це не тільки підвищує масштабованість системи, але й робить її доступною для ширшого кола користувачів, а також дозволяє доповнити існуючі бази даних про пошкодження доріг.

Проєкт також включав тестування системи на реальних відео та зображеннях, що підтвердило її високу ефективність та надійність. Результати тестування виявилися позитивними, підкреслюючи здатність системи адекватно виявляти дефекти в різних умовах.

Напрями вдосконалення можуть включати розширення набору даних для покращення точності моделі, оптимізації алгоритмів та моделі для підвищення ефективності та зниження обчислювальних витрат, а також розробку додаткових функцій для вебзастосунку та оптимізацію інтеграційних можливостей програмного інтерфейсу.

Робота виконана в межах міжнародного проєкту «Визначення подібності зображень загального призначення для гетерогенного застосування» (спільно з Політехнічним інститутом м. Томар, Португалія).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. РВ. 2.3-218-02071168/02070915-819:2013. Рекомендації щодо застосування різних методів оцінки поздовжньої рівності за критерієм впливу нерівностей поверхні покриттів на споживачів дорожніх послуг. Київ: Державна служба автомобільних доріг України, 2012. — 30 с.
2. Delatte, N.J. Concrete pavement design, construction, and performance (2nd ed.) // Boca Raton. — 2014. — с. 125.
3. Giroud, J.P., and Han, J. "Mechanisms governing the performance of unpaved roads incorporating geosynthetics" // Geosynthetics. — 2016. — с. 22-36.
4. Giroud, J.P., and Han, J. "Design Method for Geogrid-Reinforced Unpaved Roads. I Development of Design Method" // Journal of Geotechnical and Geoenvironmental Engineering. — 2004. — с. 775-786.
5. The Deterioration of Asphalt Pavement and its Causes [Електронний ресурс]. — Режим доступу: https://www.pavemanpro.com/article/deterioration_asphalt_causes/.
6. СОУ 45.2-00018112-078:2012. Автомобільні дороги. Оцінка рівності дорожніх покриттів за Міжнародним Індексом Рівності (IRI). Київ: Державна служба автомобільних доріг України, 2012. — 32 с.
7. Waze Official website. [Електронний ресурс]. — Режим доступу: https://www.waze.com/.
8. Fox A., Kumar B.V., Chen J., Bai F. Crowdsourcing undersampled vehicular sensor data for pothole detection // Proceedings of the 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). — Seattle, WA, USA, 22–25 June 2015. — с. 515–523.
9. Zhang L., Qi G.J., Zhang D., Tang J. Latent Dirichlet Truth Discovery: Separating Trustworthy and Untrustworthy Components in Data Sources // IEEE Access. — 2018. — № 6. — с. 1741–1752.

10. Sattar S., Li S., Chapman M. Road Surface Monitoring Using Smartphone Sensors: A Review // Sensors. — 2018. — № 18(11). — P. 3845. — Режим доступу до журналу: https://doi.org/10.3390/s18113845
11. CityRover official website [Електронний ресурс]. — Режим доступу: https://www.cityrover.com/
12. Ultralytics YOLOv8 Docs [Електронний ресурс]. — Режим доступу: https://docs.ultralytics.com/.
13. Dadhich, A. Practical Computer Vision. — Packt Publishing, 2018.
14. Docker: Up & Running: Shipping Reliable Containers in Production. 1st Edition. — O'Reilly Media, 2015. — 227 с.
15. Efficient MySQL Performance: Best Practices and Techniques. 1st Edition. — O'Reilly Media, 2021. — 335 с.
16. Bellemare, A. Building Event-Driven Microservices: Leveraging Organizational Data at Scale // O'Reilly Media. — 1st ed. — August 11, 2020. — 321 с.
17. Donovan A.A., Kernighan B.W. The Go Programming Language. — Addison-Wesley Professional, October 2015. — 400 с.
18. HTMX [Електронний ресурс]. — Режим доступу: https://htmx.org/.
19. Extending HTML As Hypermedia [Електронний ресурс]. — Режим доступу: https://hypermedia.systems/extending-html-as-hypermedia/.
20. Beazley D., Jones B.K. Python Cookbook. — 3rd Edition. — O'Reilly Media, 2013. — 706 с.
21. Artasanchez, A., Joshi, P. Artificial Intelligence with Python: Your Complete Guide to Building Intelligent Apps Using Python 3.x and TensorFlow 2 (2nd ed.) // Packt Publishing. — 2020.
22. Evaluation of Machine Learning Algorithms for Object Detection in Technical Drawings like P&IDs and Circuit Diagrams [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/figure/YOLO-Detection-Schema-17_fig6_340307540

23. Road accident deaths up by 7% in first 5 months of 2022 [Електронний ресурс]. — Режим доступу: https://timesofindia.indiatimes.com/city/bhubaneswar/road-accident-deaths-up-by-7-in-first-5-months-of-2022/articleshow/93062935.cms
24. Accident data during the last 7 days [Електронний ресурс]. — Режим доступу: https://www.wbtrafficpolice.com/acccidenta-during-last-7-days.php
25. Розроблення стартап-проекту: Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.
26. Маркетинг стартап-проектів: навч. посіб. для усіх спеціальностей другого освітнього ступеню «магістр» / За заг. ред. С. О. Солнцева / С.О. Солнцев, О.В. Зозульов, Н. В. Юдіна, Т. О. Царьова, Н. В. Язвінська ; Київ : КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2019. – 218 с

ДОДАТКИ

ДОДАТОК А Лістинг розробленої програми

File detector.py

```
import logging
import math
import os
import time
from datetime import datetime
import cv2
from ultralytics import YOLO
from locator import get_geolocation
from persistence import *
from settings import *
from sort import *

# Constants
MODEL_PATH = os.path.join(STORAGE_PATH, MODEL_FOLDER,
MODEL_NAME)
CLASS_NAMES = ["pothole"]
VERTICAL_GAP = 60
model = YOLO(MODEL_PATH)


class AnalysisState:
    def __init__(self, recording_id, cap, records_folder, records_path, total_frames,
total_milliseconds):
        self.recording_id = recording_id
        self.cap = cap
        self.records_folder = records_folder
        self.records_path = records_path
        self.current_frame = 0
```

```
        self.total_frames = total_frames
        self.current_millisecond = 0
        self.total_milliseconds = total_milliseconds
        self.detection_line = []
        self.unique_results = []
        self.location = None


class Analyzer:
    def __init__(self):
        self.db = Database(HOST, USER, PASSWORD, DATABASE)
        self.tracker = Sort(max_age=20, min_hits=3, iou_threshold=0.3)
        self.analysis_strategy = {
            "VIDEO": self._analyze_video,
            "IMAGE": self._analyze_image
        }

    def analyze(self, recording_id):
        rec = self.db.get_recording_by_id(recording_id)
        if not rec:
            logging.warning(f"Recording with ID {recording_id} not found!")
            return
        if rec.status in ["FINISHED", "PROCESSING"]:
            logging.warning(f"Recording with ID {recording_id} already {rec.status}!")
            return
        self.db.update_recording_status(rec.id, "PROCESSING")

        try:
            # Prepare state constants
            file_path = os.path.join(STORAGE_PATH, VIDEO_FOLDER, rec.file_name)
            unique_records_folder = rec.file_name.split(".")[0] + "_" +
str(int(time.time_ns() / 1_000_000))
            records_path = os.path.join(STORAGE_PATH, RECORD_FOLDER,
unique_records_folder)

            unique_results = self.analysis_strategy[rec.type](rec, file_path,
unique_records_folder, records_path)
```

```
            self.db.update_recording_status(rec.id, "FINISHED")
            return unique_results
        except Exception as e:
            logging.error("Error while processing video: " + str(e))
            self.db.update_recording_status(rec.id, "FAILED")
            return 0
        finally:
            self.db.close()

    def _analyze_video(self, rec, file_path, unique_records_folder, records_path):
        cap = cv2.VideoCapture(file_path)
        video_frames_total = cap.get(cv2.CAP_PROP_FRAME_COUNT)
        video_milliseconds_total = int(cap.get(cv2.CAP_PROP_FRAME_COUNT) *
1_000 / cap.get(cv2.CAP_PROP_FPS))
        state = AnalysisState(rec.id, cap, unique_records_folder, records_path,
video_frames_total,
                              video_milliseconds_total)
        while cap.isOpened():
            success, original_image = cap.read()
            if not success:
                break
            state.detection_line = [0, original_image.shape[0] * 3 // 5,
original_image.shape[1],
                                    original_image.shape[0] * 3 // 5]
            state.current_frame = int(cap.get(cv2.CAP_PROP_POS_FRAMES))
            state.current_millisecond = int(cap.get(cv2.CAP_PROP_POS_MSEC))
            # Filling tracker with detected objects at this frame
            detections = collect_detections(original_image)
            tracker_results = self.tracker.update(detections)

            # Storing detected objects
            self._find_records_video(tracker_results, original_image, state)
            print_progress_bar(state.current_frame, state.total_frames, prefix=rec.file_name,
                               suffix="| Detected:" + str(len(state.unique_results)), length=50)
        return len(state.unique_results)

    def _analyze_image(self, rec, file_path, unique_records_folder, records_path):
```

```
        original_image = cv2.imread(file_path)
        state = AnalysisState(rec.id, original_image, unique_records_folder, records_path,
0, 0)
        state.location = get_geolocation(file_path)
        # Filling tracker with detected objects
        detections = collect_detections(original_image)
        tracker_results = self.tracker.update(detections)
        # Storing detected objects
        self._find_records_img(tracker_results, original_image, state)
        return len(state.unique_results)

    def _find_records_video(self, records, image, state: AnalysisState):
        limits = state.detection_line
        for result in records:
            x1, y1, x2, y2 = map(int, result[:4])
            confidence = result[4]
            record_id = int(result[5])
            w, h = x2 - x1, y2 - y1
            center_x, center_y = x1 + w // 2, y1 + h // 2
            if limits[0] < center_x < limits[2] and limits[1] - VERTICAL_GAP < center_y <
limits[1] + VERTICAL_GAP:
                if record_id not in state.unique_results:  # Object detected
                    state.unique_results.append(record_id)
                    image_copy = image.copy()
                    cv2.rectangle(image_copy, (x1, y1), (x2, y2), (0, 0, 255), 2)  # outline the
hole at the frame
                    self._store_record(record_id, image_copy, confidence, state)

    def _find_records_img(self, records, image, state: AnalysisState):
        for result in records:
            x1, y1, x2, y2 = map(int, result[:4])
            confidence = result[4]
            record_id = int(result[5])
            state.unique_results.append(record_id)
            image_copy = image.copy()
            cv2.rectangle(image_copy, (x1, y1), (x2, y2), (0, 0, 255), 4)  # outline the hole at
the frame
```

```
            cropped_image = crop_around_detection(image_copy, x1, y1, x2, y2)
            detection_id = self._store_record(record_id, cropped_image, confidence, state)
            if state.location is not None:
                detection_location = DetectionLocation(None, detection_id, None,
state.location[0], state.location[1],
                                                       datetime.utcnow())
                self.db.save_location(detection_location)

    def _store_record(self, record_id, image, confidence, state: AnalysisState):
        if not os.path.exists(state.records_path):
            os.makedirs(state.records_path)
        record_file_name = "rec_" + str(record_id) + ".jpg"
        record_file_path = os.path.join(state.records_path, record_file_name)
        cv2.imwrite(record_file_path, image)
        short_path = os.path.join(state.records_folder, record_file_name)
        detection = Detection(None, state.recording_id, short_path, state.current_frame,
state.total_frames,
                              state.current_millisecond, state.total_milliseconds, confidence,
datetime.utcnow())
        return self.db.save_detection(detection)


def collect_detections(detectable_image):
    results = model(detectable_image, device="mps", stream=True)
    detection_stack = np.empty((0, 5))

    for r in results:
        for box in r.boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            conf = math.ceil((box.conf[0] * 100)) / 100
            class_name = CLASS_NAMES[int(box.cls[0])]
            # Remembering detected object
            if class_name in ["pothole"] and conf > 0.30:
                detection = np.array([x1, y1, x2, y2, conf])
                detection_stack = np.vstack((detection_stack, detection))

    return detection_stack
```

```
def print_progress_bar(iteration, total, prefix='', suffix='', decimals=1, length=100,
fill='█'):
    percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
    filled_length = int(length * iteration // total)
    bar = fill * filled_length + '-' * (length - filled_length)

    logging.info(f'\r{prefix} |{bar}| {percent}% {suffix}')


def crop_around_detection(image, x1, y1, x2, y2):
    h, w, _ = image.shape
    crop_width = w // 2
    crop_height = h // 2
    # Default starting points for cropping
    start_x = 0
    start_y = 0

    # Adjust start_x if detection is on the right half
    if x2 > crop_width:
        start_x = min(w - crop_width, x1)

    # Adjust start_y if detection is on the bottom half
    if y2 > crop_height:
        start_y = min(h - crop_height, y1)

    end_x = start_x + crop_width
    end_y = start_y + crop_height
    cropped_img = image[start_y:end_y, start_x:end_x]
    return cropped_img
```

File locator.py

```
import logging
import os
import xml.etree.ElementTree as ET
from datetime import datetime, timedelta
from GPSPhoto import gpsphoto
from persistence import *
from settings import *


class AnalysisState:
    def __init__(self, recording_id, cap, records_folder, records_path, total_frames,
total_milliseconds):
        self.recording_id = recording_id
        self.cap = cap
        self.records_folder = records_folder
        self.records_path = records_path
        self.current_frame = 0
        self.total_frames = total_frames
        self.current_millisecond = 0
        self.total_milliseconds = total_milliseconds
        self.detection_line = []
        self.unique_results = []
        self.location = None


class Locator:
    def __init__(self):
        self.db = Database(HOST, USER, PASSWORD, DATABASE)

    def parse_gpx(self, file_path):
        """Parse GPX file and return list of [timestamp, lat, lon]."""
        tree = ET.parse(file_path)
        root = tree.getroot()

        data = []
```

```
        for wpt in root.findall('.//wpt'):
            timestamp = wpt.find('./time').text
            lat = wpt.get('lat')
            lon = wpt.get('lon')
            data.append([timestamp, lat, lon])
        return data

    def locate(self, recording_id):
        gpx = self.db.get_gpx_by_id(recording_id)
        if not gpx:
            logging.warning(f"GPX with ID {recording_id} not found!")
            return
        if gpx.status in ["FINISHED", "PROCESSING"]:
            logging.warning(f"GPX with ID {recording_id} already {gpx.status}!")
            return
        self.db.update_gpx_status(gpx.id, "PROCESSING")

        try:
            file_path = os.path.join(STORAGE_PATH, GPX_FOLDER, gpx.file_name)
            location_data = self.parse_gpx(file_path)
            # For demonstration: printing the parsed data
            for item in location_data:
                logging.info(f"Timestamp: {item[0]}, Latitude: {item[1]}, Longitude:
{item[2]}")
            detections = self.db.get_detections_by_recording_id(recording_id)
            total_frames = detections[0].total_frame_number  # e.g. 560
            start_time = datetime.fromisoformat(location_data[0][0])
            end_time = datetime.fromisoformat(location_data[-1][0])
            total_duration = (end_time - start_time).total_seconds()  # Total video duration
in seconds
            frame_duration = total_duration / total_frames  # Duration of each frame in
seconds
            for detection in detections:
                current_frame = detection.frame_number  # e.g. 10

                # Calculate elapsed time for the current frame
```

```
            elapsed_time = start_time + timedelta(seconds=frame_duration *
current_frame)

            # Find two closest location data points
            prev_point = location_data[0]
            for point in location_data[1:]:
                if datetime.fromisoformat(point[0]) > elapsed_time:
                    break
                prev_point = point

            # Interpolate location between prev_point and point based on time
            prev_time = datetime.fromisoformat(prev_point[0])
            next_time = datetime.fromisoformat(point[0])
            fraction = (elapsed_time - prev_time) / (next_time - prev_time)
            lat = float(prev_point[1]) + fraction * (float(point[1]) - float(prev_point[1]))
            lon = float(prev_point[2]) + fraction * (float(point[2]) - float(prev_point[2]))
            # Create and save the DetectionLocation object to the database
            detection_location = DetectionLocation(None, detection.id, gpx.id, lat, lon,
datetime.utcnow())
            self.db.save_location(detection_location)
        self.db.update_gpx_status(gpx.id, "FINISHED")
        return True
    except Exception as e:
        logging.error("Error while processing location: " + str(e))
        self.db.update_gpx_status(gpx.id, "FAILED")
        return False
    finally:
        self.db.close()


def get_geolocation(img_path):
    gps_data = gpsphoto.getGPSData(img_path)
    if gps_data is None:
        return None
    logging.info(f"GPS data: {gps_data}")
    return [gps_data['Latitude'], gps_data['Longitude']]
```

ДОДАТОК Б Презентація

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра Інженерії програмного забезпечення в енергетиці.

Інструментальні засоби аналізу якості дорожнього покриття

Виконав: студент магістерського рівня 2-го року навчання, групи ТВ-22мп
Чурчин Денис Андрійович
Керівник: к.е.н., доцент, Гусєва Ірина Ігорівна

АКТУАЛЬНІСТЬ ТЕМИ

- Проблема якості дорожнього покриття є важливою через її вплив на безпеку та комфорт дорожнього руху.
- Традиційні методи аналізу часто виявляються повільнішими та не завжди точними.
- Автоматизація надає можливість більш ефективного та об'єктивного оцінювання стану дорожнього покриття.

ЗАВДАННЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ЯКОСТІ ДОРОЖНЬОГО ПОКРИТТЯ

Мета: Розробка автоматизованої системи для аналізу якості дорожнього покриття

Об'єкт дослідження: Дефекти дорожнього покриття, такі як ями

Предмет дослідження: Програмне забезпечення для аналізу якості дорожнього покриття на основі обробки медіа даних із визначенням на них вибоїн

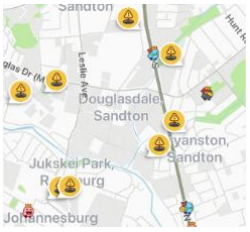
Завдання для вирішення:

1. Розробити та навчити модель ШІ для виявлення дефектів дорожнього покриття.
2. Асоціювати пошкодження з їх геолокацією для нанесення на мапу.
3. Створити веб-застосунок для завантаження та аналізу медіа файлів, перегляду результатів.
4. Розробити API, що надає програмний інтерфейс до всіх функціональних можливостей веб-застосунку.

МЕТОДИ АНАЛІЗУ ДОРОЖНЬОГО ПОКРИТТЯ

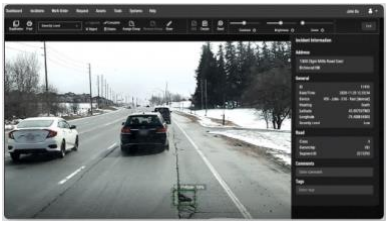
waze.com

- Візуальна людська оцінка, інформування на основі скарг
  - Переваги: широка впровадженість, більше видів пошкоджень
  - Недоліки: неефективність, суб'єктивність, непрозорість, складність організації доступу до інформації



cityrover.com

- Системи на основі комп'ютерного зору
  - Переваги: масштабованість, доступність даних, деталізація
  - Недоліки: висока ціна (можлива оптимізація)



РОЗРОБКА МОДЕЛІ ДЛЯ РОЗПІЗНАВАННЯ ТА ЗАБЕЗПЕЧЕННЯ УНІКАЛЬНОСТІ

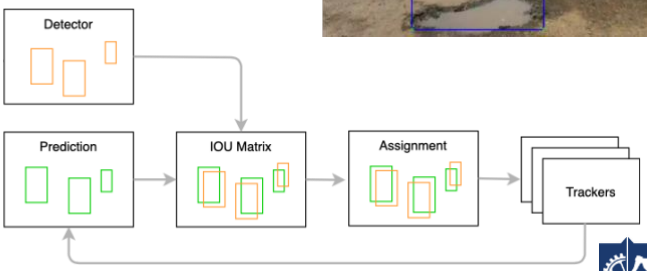
Навчено модель з сімейства YOLO (you only look once)

Навчальний набір: 7000+ зображень

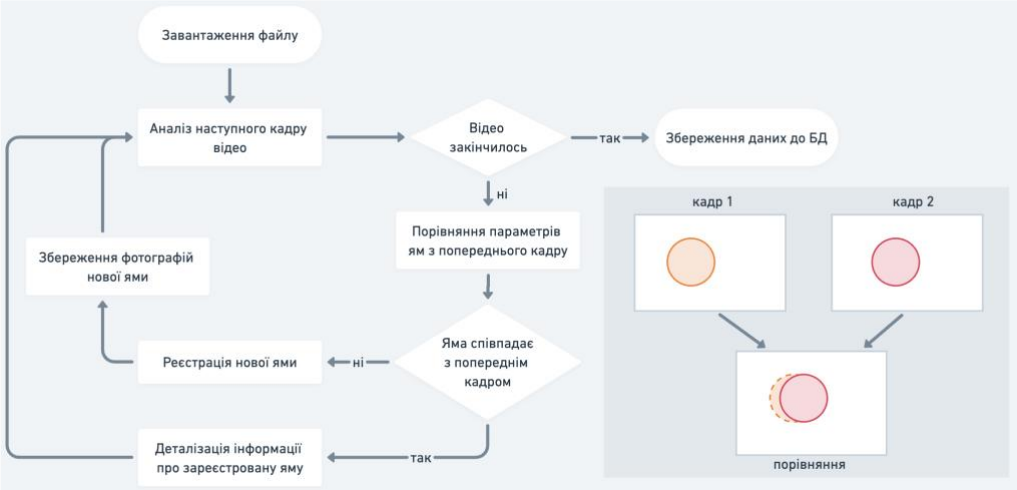
SORT (Simple Online and Realtime Tracking) алгоритм:

Фільтр Калмана – виконує прогнозування місцезнаходження об'єкта в наступному кадрі відео.

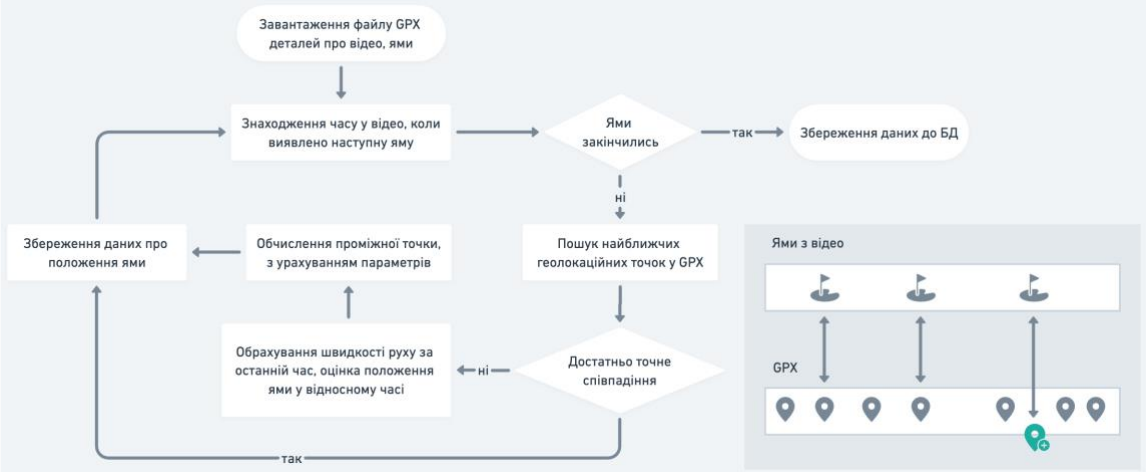
Метрика IOU - вимірює перетин між прогнозованими та фактичними областями об'єктів для точного відстеження.



АЛГОРИТМ ВІДСТЕЖУВАННЯ УНІКАЛЬНИХ ЯМ

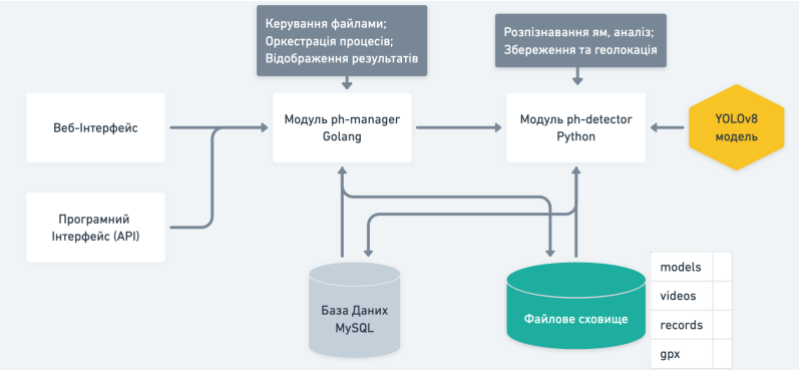


АЛГОРИТМ ПОЄДНАННЯ ТОЧОК GPX ФАЙЛУ ТА ВИЗНАЧЕНИХ ДЕФЕКТІВ



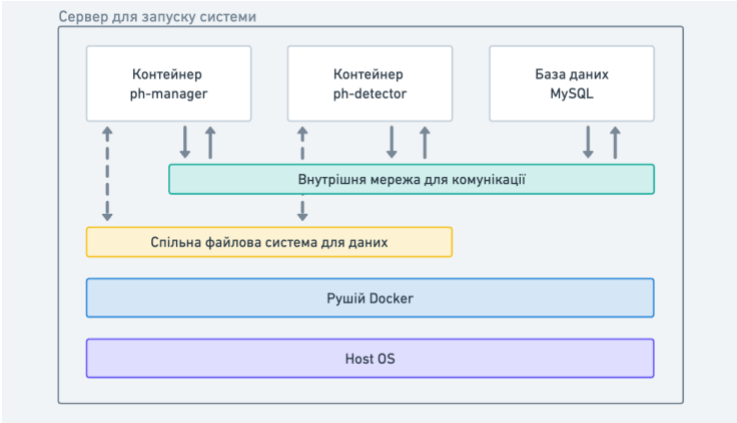
АРХІТЕКТУРА КОМПОНЕНТІВ

- Мікросервіс для аналізу зображень (ph-detector)
- Мікросервіс для оркестрації даних (ph-manager)
  - Веб та API інтерфейси
- Сховище даних

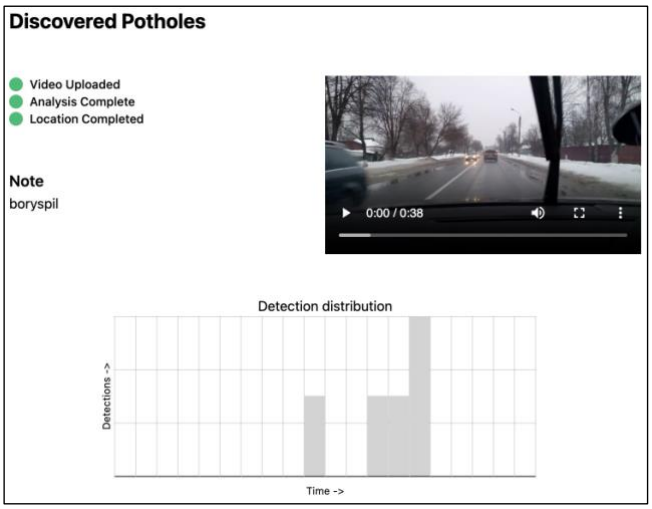


ЗАСОБИ РОЗРОБКИ

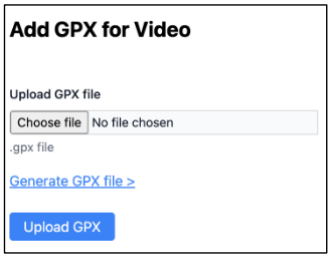
- Docker
- MySQL
- Golang
- Python + YOLOv8

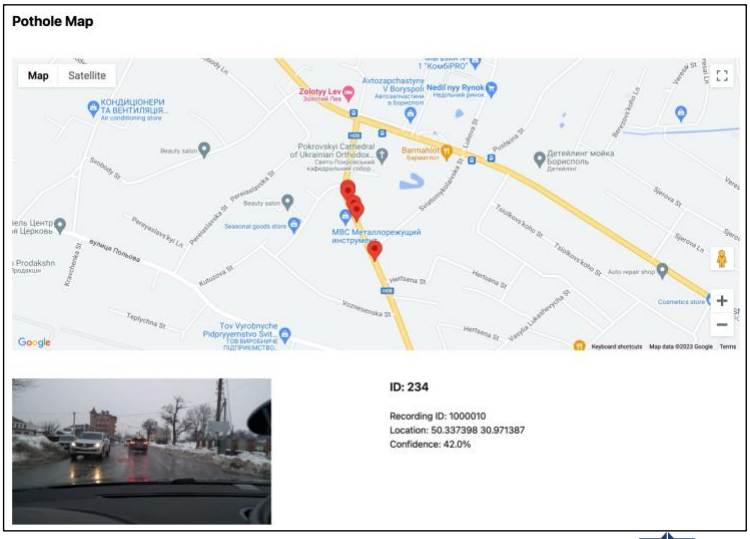


РЕЗУЛЬТАТИ РОЗРОБКИ



РЕЗУЛЬТАТИ РОЗРОБКИ





РЕЗУЛЬТАТИ РОЗРОБКИ

Analyzed Videos + New Upload

View Map

ID	Type	Status	Potholes	Time Created
1000000	VIDEO	FINISHED	49	12:34 07.06.2023
1000001	IMAGE	FINISHED	1	13:08 15.10.2023
1000004	IMAGE	FINISHED	1	13:16 15.10.2023
1000005	IMAGE	FINISHED	3	13:17 15.10.2023
1000006	VIDEO	FINISHED	166	13:18 15.10.2023
1000008	IMAGE	FINISHED	2	13:30 15.10.2023
1000009	VIDEO	FINISHED	1	13:32 15.10.2023
1000010	VIDEO	FINISHED	11	08:04 16.10.2023
1000011	VIDEO	FINISHED	5	08:19 16.10.2023
1000012	IMAGE	FINISHED	2	08:21 16.10.2023

ВИСНОВКИ

В ході роботи було створено інструментальні засоби для аналізу дорожнього покриття на основі машинного навчання.

1. Розроблено модель ШІ для виявлення дефектів дорожнього покриття.
2. Розроблено алгоритм для доповнення пошкоджень їх точною геолокацією.
3. Створено веб-застосунок для аналізу медіа файлів.
4. Розроблено API, що дозволяє ширше впроваджувати систему.

Розроблена система вирішує ряд ключових проблем у сфері обслуговування дорожньої інфраструктури, автоматизуючи процес, підвищуючи точність та прискорюючи геолокацію дефектів.

Потенційні шляхи для розвитку включають в себе регулярне перенавчання моделі ШІ з використанням свіжих даних, що забезпечить її актуальність і точність.