

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

Навчально-науковий Інститут телекомунікаційних систем

Кафедра електронних комунікацій та інтернету речей

«До захисту допущено»

Завідувач кафедри

_____ Анатолій МАКАРЕНКО

«__» _____ 20__ р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності 172 Телекомунікації та радіотехніка

**на тему: «Система моніторингу стану доріг за допомогою IoT-сенсорів з
хмарною обробкою даних»**

Виконав (-ла):

студент (-ка) IV курсу, групи ТС-22

Іванов Євгеній Володимирович _____

Керівник:

Ст. викладач кафедри ЕКІР, д-р філософії

Шмігель Богдан Олегович _____

Рецензент:

Старший викладач кафедри ІТТ, к.т.н.,

Педан Станіслав Ігорович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент (-ка) _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий Інститут телекомунікаційних систем
Кафедра електронних комунікацій та інтернету речей

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 Телекомунікації та радіотехніка

Освітня програма – «Системи електронних комунікацій та Інтернету речей»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Анатолій МАКАРЕНКО

«__» _____ 20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Іванову Євгенію Володимировичу

1. Тема роботи «Система моніторингу стану доріг за допомогою IoT-сенсорів з хмарною обробкою даних», керівник роботи Шмігель Богдан Олегович, д-р філософії, ст. викладач кафедри ЕКІР, затверджені наказом по університету від «26» травня 2026 р. № 1912-С

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи:

- Матеріали наукових статей, оглядів та періодичних видань за тематикою моніторингу дорожнього покриття, вібраційних методів діагностики, IoT-архітектур (Edge, Cloud, Fog), LPWAN-мереж (LoRaWAN, NB-IoT, LTE-M) та протоколів прикладного рівня (MQTT, CoAP, LwM2M).
- Інформаційні ресурси мережі Інтернет, включно з технічною документацією на MEMS-акселерометри, одноплатні комп'ютери (Raspberry Pi, ESP32), публічні MQTT-брокери, а також бази даних наукових публікацій (IEEE Xplore, Scopus, MDPI, Nature Scientific Reports) та відкриті датасети зображень дефектів (SDNET2018).
- Нормативно-технічні документи України (ДСТУ 8954:2019, ДСТУ 3587:2022, ДСТУ 9030:2020) щодо оцінювання дефектності дорожнього одягу, вимог до експлуатаційного стану та впливів на навколишнє середовище.
- Навчально-методичні матеріали кафедри електронних комунікацій та

інтернету речей КПП ім. Ігоря Сікорського щодо виконання дипломних робіт, оформлення пояснювальної записки та презентаційних матеріалів.

- Структурований план порядку розробки матеріалів дипломної роботи, затверджений керівником та завідувачем кафедри.

4. Зміст роботи:

- Виконати огляд літератури за темою методів детектування дефектів доріг (оптичні, LiDAR, GPR, вібраційні) та еволюції IoT-архітектур для транспортного моніторингу.
- Дослідити фізичні принципи вібраційного методу на основі MEMS-акселерометрів та обґрунтувати вибір сенсорних компонентів (частота дискретизації ≥ 250 Гц, спектральна щільність шуму ≈ 51 мкГ/√Гц) з порівняльним аналізом характеристик.
- Проаналізувати архітектури розподілених обчислень (Edge, Fog, Cloud), протоколи прикладного рівня (MQTT, CoAP) та технології LPWAN (LoRaWAN, NB-IoT, LTE-M) з оцінкою їх придатності для мобільних систем.
- Виконати математичне моделювання алгоритмів попередньої обробки сигналів (медіанна фільтрація, адаптивне вейвлет-перетворення AWSSF з вейвлетом Добеші-2) та адаптивного порогування з розрахунком ефекту стиснення даних (до 78%).
- Провести кількісне порівняння моделей класифікації для вібраційних даних (Random Forest, 1D-CNN, CNN+LSTM) з визначенням оптимальних для Edge та Cloud (F1-score до 0,934).
- Розробити програмну систему виявлення дефектів на зображеннях (веб-інтерфейс Streamlit, детектор Кенні, морфологічне закриття, пошук контурів) з картографічною візуалізацією на Folium.
- Виконати експериментальне тестування на збалансованому датасеті SDNET2018 (20 зображень) з обчисленням метрик Precision (60%), Recall (30%), Accuracy (55%), F1 (40%) та аналізом чутливості до параметрів.
- Реалізувати емуляцію периферійно-хмарної взаємодії через MQTT (скрипт mqtt.py) з публічним брокером, провести натурний експеримент передачі зображення та верифікувати роботу системи.
- Сформулювати висновки щодо теоретичних засад, практичної реалізації, обмежень запропонованого підходу та перспектив подальшого вдосконалення (адаптивні пороги, нейромережеві методи, інтеграція реальних IoT-сенсорів).

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Дата видачі завдання 24.02

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання, узгодження теми та структури дипломної роботи	25.04.26	Виконано
2	Аналіз сучасних методів детектування дефектів дорожнього покриття (оптичні, LiDAR, GPR, вібраційні), дослідження IoT-архітектур (Edge, Fog, Cloud) та протоколів передачі даних (MQTT, CoAP, LPWAN)	25.04.26 – 06.05.26	Виконано
3	Обґрунтування вибору апаратних компонентів (MEMS-акселерометри, вимоги до частоти дискретизації та спектральної щільності шуму), порівняльний аналіз брокерів повідомлень (NATS, Mosquitto, Zenoh) та технологій LPWAN (LoRaWAN, NB-IoT, LTE-M)	06.05.26 – 11.05.26	Виконано
4	Розробка алгоритмів попередньої обробки сенсорних даних (просторова нормалізація, медіанна фільтрація, адаптивне вейвлет-перетворення AWSSF) та адаптивного порогоування	12.05.26 – 18.05.26	Виконано
5	Дослідження методів представлення вібраційних даних (часовий, частотний, частотно-часовий домени) та моделей машинного/глибокого навчання (Random Forest, 1D-CNN, CNN+LSTM) для класифікації дефектів	19.05.26 – 26.05.26	Виконано
6	Програмна реалізація веб-додатку для виявлення дефектів на зображеннях (Streamlit, OpenCV, Folium), розробка алгоритму детектора Кенні, морфологічного закриття та фільтрації контурів	27.05.26 – 03.06.26	Виконано
7	Інтеграція модуля комп'ютерного зору в периферійно-хмарну архітектуру через емуляцію MQTT (скрипт mqtt.py), проведення натурного експерименту, тестування на датасеті SDNET2018 (20 зображень) з обчисленням метрик Precision, Recall, Accuracy, F1	03.06.26 – 07.06.26	Виконано
8	Підготовка ілюстративного матеріалу (рисуноків, таблиць, схем) та оформлення пояснювальної записки згідно з вимогами		
9	Формулювання загальних висновків, перевірка унікальності тексту, підготовка презентації та реферату, попередній захист		

Студент

Євгеній ІВАНОВ

Керівник роботи

Богдан ШМІГЕЛЬ

РЕФЕРАТ

Диплом містить 99 аркушів, 1 додаток, 7 таблиць та має 14 ілюстрацій з 56 використаними джерелами.

Мета роботи – розробка комплексної системи предиктивного моніторингу та виявлення дефектів дорожнього покриття на основі інтеграції IoT-сенсорів вібрації та методів комп'ютерного зору в єдиній периферійно-хмарній архітектурі.

Актуальність теми зумовлена необхідністю переходу від реактивних методів ремонту доріг до предиктивного моніторингу, що дозволяє своєчасно виявляти тріщини, вибоїни та інші руйнування. Традиційний візуальний огляд є суб'єктивним, дорогим і небезпечним. Сучасні автоматизовані системи на базі комп'ютерного зору та Інтернету речей забезпечують високу швидкість, об'єктивність і масштабованість, що особливо важливо для розумних міст (Smart City).

Об'єкт дослідження – процеси збору, передачі та хмарного аналізу багатомодальних даних (вібраційних сигналів та цифрових зображень) для оцінювання експлуатаційного стану автомобільних доріг.

Предмет дослідження – алгоритми цифрової обробки зображень (детектор меж Кенні, морфологічний аналіз) та вібросигналів (вейвлет-фільтрація), мережеві протоколи передачі даних (MQTT, CoAP), архітектурні рішення розподілених обчислень (Edge/Cloud Computing) та інструменти геоінформаційної візуалізації.

Методи дослідження – аналіз науково-технічної літератури, порівняльний аналіз технологій, комп'ютерне моделювання, програмна реалізація веб-додатку на Python із використанням бібліотек Streamlit, OpenCV та Folium, експериментальне тестування на збалансованому наборі з 20 зображень, оцінка якості виявлення за метриками Precision, Recall, Accuracy та F1-міра.

Ключові слова: виявлення дефектів дорожнього покриття,

комп'ютерний зір, детектор меж Кенні, Streamlit, Folium, IoT, периферійні обчислення, LoRaWAN, MQTT, згорткові нейронні мережі, SDNET2018.

ABSTRACT

The thesis contains 99 pages, 1 app, 7 tables, and 14 illustrations, with 56 references cited.

The aim of the work is the development of a comprehensive system for predictive monitoring and detection of road surface defects, based on the integration of IoT vibration sensors and computer vision methods within a unified edge-cloud architecture.

The relevance of the topic is determined by the need to shift from reactive road repair methods to predictive monitoring, which allows timely detection of cracks, potholes and other damage. Traditional visual inspection is subjective, expensive and dangerous. Modern automated systems based on computer vision and the Internet of Things provide high speed, objectivity and scalability, which is especially important for smart cities (Smart City).

The object of research is the processes of collecting, transmitting, and conducting cloud-based analysis of multimodal data (vibration signals and digital images) to evaluate the operational condition of roads.

The subject of research is algorithms for digital image processing (Canny edge detector, morphological analysis) and vibration signal processing (wavelet filtering), network data transmission protocols (MQTT, CoAP), architectural solutions for distributed computing (Edge/Cloud Computing), and geoinformation visualization tools.

Research methods – analysis of scientific and technical literature, comparative analysis of technologies, computer simulation, software implementation of a web application in Python using Streamlit, OpenCV and Folium libraries, experimental testing on a balanced dataset of 20 images, evaluation of detection quality using Precision, Recall, Accuracy and F1-score metrics.

Keywords: road pavement defect detection, computer vision, Canny edge detector, Streamlit, Folium, IoT, edge computing, LoRaWAN, MQTT, convolutional neural networks, SDNET2018.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ	10
ВСТУП	13
1 АНАЛІЗ ТЕХНОЛОГІЙ ТА МЕТОДІВ МОНІТОРИНГУ ДОРІГ	16
1.1 Огляд та порівняльний аналіз сучасних методів детектування дефектів дорожнього покриття.....	17
1.1.1 Оптичні системи та методи комп'ютерного зору (Computer Vision)	17
1.1.2 Лазерне сканування, профілометрія та технології LiDAR	20
1.1.3 Підповерхневе зондування за допомогою георадарів (GPR)	21
1.1.4 Вібраційні методи, інерціальні вимірювальні модулі (IMU) та смарт-матеріали.....	23
1.2 Дослідження архітектурних рішень та протоколів передачі даних в системах IoT	26
1.2.1 Багаторівнева обчислювальна архітектура (Edge, Fog, Cloud)	26
1.2.2 Мережеві протоколи передачі даних (Network Layer).....	28
1.2.3 Протоколи прикладного рівня (Application Layer).....	30
1.3 Висновки до розділу 1	33
2 ТЕОРЕТИЧНЕ ПРОЕКТУВАННЯ ТА ОБҐРУНТУВАННЯ СИСТЕМИ ...	36
2.1 Обґрунтування вибору апаратних компонентів та структури взаємодії сенсорів з хмарою	36
2.1.1 Апаратна база: аналіз та вибір сенсорних компонентів	36
2.1.2 Розподілена обчислювальна архітектура: Edge Intelligence та Cloud Computing.....	39
2.1.3 Порівняльний аналіз IoT-протоколів та вибір брокерів повідомлень	40
2.1.4 Аналіз технологій енергоефективних глобальних мереж (LPWAN)	44
2.2 Алгоритми попередньої обробки та фільтрації сенсорних даних	48
2.2.1 Базова фільтрація: усунення високочастотних шумів та дрейфу нуля	48
2.2.2 Адаптивна вейвлет-трансформація та просторова кореляція	50
2.2.3 Алгоритми адаптивного порогування та Edge-компресія	52
2.3 Розробка алгоритму виявлення дефектів на основі аналізу вібраційних сигналів	54
2.3.1 Оцінка методів представлення вібраційних даних (Data Representation Techniques).....	54

2.3.2 Стратегія віконної сегментації та компенсація швидкості.....	57
2.3.3 Моделі машинного та глибокого навчання (Machine Learning & Deep Learning).....	58
2.3.4 Валідація алгоритмів: метрика F1-Score та дисбаланс класів.....	61
2.4 Висновки до розділу 2	62
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ РОБОТИ СИСТЕМИ	64
3.1 Розробка інтерфейсу користувача та візуалізація даних на карті	64
3.2 Реалізація алгоритмів виявлення дефектів та оцінка їхньої роботи.....	65
3.2.1 Загальний підхід до виявлення дефектів	65
3.2.2 Покрокова реалізація алгоритму	66
3.2.3 Інтеграція алгоритму з веб-інтерфейсом.....	69
3.2.4 Оцінка якості виявлення дефектів	71
3.2.5 Аналіз чутливості до параметрів.....	72
3.3 Перевірка працездатності системи на тестових прикладах.....	74
3.3.1 Склад тестової вибірки та методика експерименту	74
3.3.2. Результати тестування та кількісні показники	74
3.3.3 Аналіз причин помилок та шляхи покращення	77
3.4 Інтеграція модуля комп'ютерного зору в периферійно-хмарну архітектуру системи моніторингу	79
3.5 Висновки до розділу 3	83
ВИСНОВКИ	84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	87
ДОДАТКИ	94
ДОДАТОК А.....	94

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

1D-CNN	одновимірна згорткова нейронна мережа (1-dimensional Convolutional Neural Network)
Accuracy	загальна правильність класифікації
AWSSF	адаптивна вейвлет-фільтрація у просторі масштабів (Adaptive Wavelet Transformation Scale Space Filtering)
BGR	порядок каналів кольору (синій, зелений, червоний)
CAN	контролерна мережа (Controller Area Network)
CNN	згорткова нейронна мережа (Convolutional Neural Network)
CoAP	протокол прикладного рівня для обмежених пристроїв (Constrained Application Protocol)
CSV	текстовий формат зі значеннями, розділеними комами (Comma-Separated Values)
Edge Computing	периферійні обчислення
F1-міра	гармонійне середнє між точністю та повнотою
FBG	волоконно-оптична решітка Брегга (Fiber Bragg Grating)
FFT	швидке перетворення Фур'є (Fast Fourier Transform)
FN	хибно негативні (False Negative) – пропущені дефекти
FP	хибно позитивні (False Positive) – помилкові спрацювання
GPR	георадар (Ground Penetrating Radar)
GPS	глобальна система позиціонування (Global Positioning System)

HTML	мова гіпертекстової розмітки (HyperText Markup Language)
IMU	інерціальний вимірювальний модуль (Inertial Measurement Unit)
IoT	Інтернет речей (Internet of Things)
KNN	метод k-найближчих сусідів (k-Nearest Neighbors)
LiDAR	система виявлення та визначення дальності за допомогою світла (Light Detection and Ranging)
LoRaWAN	мережа великого радіусу дії з низьким енергоспоживанням (Long Range Wide Area Network)
LPWAN	енергоефективні мережі великого радіусу дії (Low Power Wide Area Networks)
LSTM	довга короткострокова пам'ять (Long Short-Term Memory)
LTE-M	технологія LTE для машин (Long Term Evolution for Machines)
LwM2M	легковаговий протокол управління пристроями M2M (Lightweight Machine to Machine)
MEMS	мікроелектромеханічні системи (Micro-Electro-Mechanical Systems)
MQTT	протокол передачі телеметрії у чергах повідомлень (Message Queuing Telemetry Transport)
NB-IoT	вузькосмуговий Інтернет речей (Narrowband Internet of Things)
OpenCV	бібліотека комп'ютерного зору з відкритим кодом (Open Source Computer Vision Library)
OTA	оновлення «по повітрю» (Over-The-Air)

Precision	точність класифікації
QoS	якість обслуговування (Quality of Service)
Recall	повнота класифікації
RF	випадковий ліс (Random Forest)
RGB	кольорова модель (червоний, зелений, синій)
RMS	середньоквадратичне значення (Root Mean Square)
SDNET2018	назва відкритого датасету зображень тріщин бетонних поверхонь
STFT	віконне перетворення Фур'є (Short-Time Fourier Transform)
TCP	протокол керування передачею (Transmission Control Protocol)
TN	істинно негативні (True Negative) – правильно визначені чисті зразки
TP	істинно позитивні (True Positive) – правильно виявлені дефекти
UDP	протокол дейтаграм користувача (User Datagram Protocol)
WIM	зважування в русі (Weigh-in-Motion)
YOLO	метод виявлення об'єктів «Ви дивитесь лише раз» (You Only Look Once)
ДСТУ	державний стандарт України

ВСТУП

Актуальність теми. Стан автомобільних доріг є одним із ключових факторів економічного розвитку, безпеки дорожнього руху та якості життя населення. В Україні, як і в усьому світі, спостерігається інтенсивне старіння дорожньої інфраструктури, що призводить до утворення тріщин, вибоїн, колій та інших дефектів. Традиційні методи візуального інспектування є суб'єктивними, трудомісткими та небезпечними, а ремонтні роботи часто проводяться за реактивним принципом — після виникнення критичних руйнувань. Сучасний розвиток Інтернету речей (IoT), периферійних обчислень (Edge Computing), хмарних платформ та методів комп'ютерного зору створює передумови для переходу до предиктивного моніторингу дорожнього покриття — своєчасного виявлення дефектів на ранніх стадіях з мінімальними витратами. Тема роботи є актуальною для впровадження в інтелектуальних транспортних системах (Smart City) та відповідає державним стандартам України (ДСТУ 8954:2019, ДСТУ 3587:2022).

Мета роботи — розробка комплексної системи предиктивного моніторингу та виявлення дефектів дорожнього покриття на основі інтеграції IoT-сенсорів вібрації та методів комп'ютерного зору в єдиній периферійно-хмарній архітектурі.

Завдання дослідження:

- Провести аналіз сучасних методів детектування дефектів дорожнього покриття (оптичні, LiDAR, GPR, вібраційні) та IoT-архітектур (Edge, Fog, Cloud), протоколів передачі даних (MQTT, CoAP, LPWAN).
- Обґрунтувати вибір апаратних компонентів (MEMS-акселерометри) та розподіленої обчислювальної архітектури для мобільного моніторингу.
- Дослідити алгоритми попередньої обробки сенсорних даних (медіанна фільтрація, адаптивне вейвлет-перетворення AWSSF, адаптивне порогування) та методи представлення вібраційних сигналів.

- Порівняти моделі машинного та глибокого навчання (Random Forest, 1D-CNN, CNN+LSTM) для класифікації дорожніх аномалій.
- Розробити програмну систему виявлення дефектів на зображеннях дорожнього покриття з веб-інтерфейсом (Streamlit), картографічною візуалізацією (Folium) та алгоритмом на основі детектора Кенні.
- Виконати експериментальне тестування системи на відкритому датасеті SDNET2018 з оцінкою якості за метриками Precision, Recall, Accuracy, F1.
- Реалізувати емуляцію периферійно-хмарної взаємодії через протокол MQTT та провести натурний експеримент передачі даних від віртуального edge-вузла до хмарного аналітичного центру.

Об'єкт дослідження – процеси збору, передачі та хмарного аналізу багатомодальних даних (вібраційних сигналів та цифрових зображень) для оцінювання експлуатаційного стану автомобільних доріг.

Методи дослідження – аналіз науково-технічної літератури, порівняльний аналіз технологій, комп'ютерне моделювання, програмна реалізація веб-додатку на Python із використанням бібліотек Streamlit, OpenCV та Folium, експериментальне тестування на збалансованому наборі з 20 зображень, оцінка якості виявлення за метриками Precision, Recall, Accuracy та F1-міра.

Структура та зміст роботи. Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатку.

У першому розділі проведено системний аналіз сучасних методів детектування дефектів дорожнього покриття: оптичних систем комп'ютерного зору (CNN, YOLO), лазерного сканування (LiDAR), георадарів (GPR) та вібраційних методів на базі MEMS-акселерометрів. Розглянуто архітектурні рішення Інтернету речей (трирівнева модель Edge-Fog-Cloud), мережеві протоколи LPWAN (LoRaWAN, NB-IoT, LTE-M) та прикладні протоколи (MQTT, CoAP, LwM2M). Сформовано висновки щодо

доцільності застосування вібраційного методу для масштабованого моніторингу та вибору оптимальних протоколів залежно від сегмента мережі.

У другому розділі теоретично обґрунтовано вибір апаратних компонентів (MEMS-акселерометри з частотою дискретизації ≥ 250 Гц та спектральною щільністю шуму ~ 51 мкг/ $\sqrt{\text{Гц}}$), розглянуто розподілену обчислювальну архітектуру Edge-Cloud з порівняльним аналізом брокерів повідомлень (NATS, Mosquitto, Zenoh) та технологій LPWAN. Досліджено алгоритми попередньої обробки сенсорних даних: просторова нормалізація, медіанна фільтрація, адаптивне вейвлет-перетворення (AWSSF з вейвлетом Добеші-2) та адаптивне порогування, що дозволяє скоротити обсяг даних до 78 %. Проаналізовано методи представлення вібраційних сигналів (часовий, частотний, частотно-часовий домени) та порівняно моделі класифікації: Random Forest (Precision 100 %, Recall 88,9 %), 1D-CNN, гібридні CNN+LSTM (F1-score до 0,934). Розглянуто проблему дисбалансу класів та обґрунтовано використання метрики F1-Score.

У третьому розділі реалізовано програмну систему виявлення дефектів на зображеннях дорожнього покриття. Розроблено веб-інтерфейс на базі Streamlit з інтерактивною картою Folium та можливістю налаштування порогів детектора Кенні. Алгоритм виявлення включає перетворення у відтінки сірого, розмиття Гаусса, детектор меж Кенні, морфологічне закриття, пошук контурів та фільтрацію за мінімальною площею (50 пікселів). Проведено експериментальне тестування на збалансованому датасеті SDNET2018 (20 зображень: 10 з дефектами, 10 чистих). Отримано матрицю помилок (TP=3, FN=7, FP=2, TN=8) та метрики: точність – 60 %, повнота – 30 %, загальна правильність – 55 %, F1-міра – 40 %. Виконано аналіз чутливості до параметрів детектора. Реалізовано інтеграцію модуля комп'ютерного зору в периферійно-хмарну архітектуру через емуляцію MQTT-зв'язку (скрипт mqtt.py), проведено натурний експеримент з передачі зображення від віртуального edge-вузла до хмарного дашборда, що підтвердило працездатність системи в умовах, наближених до реальних.

1 АНАЛІЗ ТЕХНОЛОГІЙ ТА МЕТОДІВ МОНІТОРИНГУ ДОРІГ

Якість та експлуатаційний стан автомобільної інфраструктури є фундаментальними чинниками, які безпосередньо впливають на рівень соціально-економічного розвитку держави, загальну безпеку дорожнього руху, швидкість логістичних операцій та комфорт пересування громадян. У контексті стрімкої глобальної урбанізації, збільшення кількості транспортних засобів та формування парадигми «розумних міст» (Smart City), традиційні підходи до моніторингу дорожнього покриття виявляють свою технологічну та економічну неспроможність. Донедавна процес оцінки стану доріг здебільшого спирався на візуальний огляд інженерами та ручний збір даних, що є не лише вкрай повільним і дорогим процесом, але й містить значний елемент суб'єктивності, а також становить небезпеку для персоналу, який працює в умовах інтенсивного трафіку. Зростання інтенсивності транспортних потоків, негативний вплив екстремальних кліматичних факторів, використання агресивних хімічних реагентів у зимовий період та природне старіння будівельних матеріалів призводять до прискореної деградації дорожнього одягу. Відповідно до таких стандартів, як ДСТУ 8954:2019 щодо оцінювання рівня дефектності дорожнього одягу [1] та ДСТУ 3587:2022 щодо вимог до експлуатаційного стану [2], а також ДСТУ 9030:2020 щодо оцінки впливів на навколишнє середовище [3], своєчасна ідентифікація руйнувань є законодавчо закріпленою вимогою.

Перехід від реактивної моделі технічного обслуговування (виправлення критичних руйнувань, що вже утворилися) до предиктивної моделі (прогнозування та ліквідація мікродефектів до їх перетворення на аварійні вибоїни) неможливий без впровадження високоточних автоматизованих систем збору та аналізу даних. Цей розділ присвячений вичерпному теоретичному та емпіричному аналізу сучасних апаратних і програмних методів детектування дефектів дорожнього покриття, що включають системи комп'ютерного зору, лазерне сканування (LiDAR), георадіолокацію (GPR) та

вібраційні інерціальні модулі (IMU). Окрім того, значна увага приділяється дослідженню архітектурних рішень концепції Інтернету речей (Internet of Things, IoT), яка виступає сполучною ланкою між сенсорним обладнанням та хмарними аналітичними платформами, забезпечуючи безперервний транспортний моніторинг у режимі реального часу.

1.1 Огляд та порівняльний аналіз сучасних методів детектування дефектів дорожнього покриття

Автоматизоване виявлення та класифікація дефектів є складним міждисциплінарним завданням, яке об'єднує методи цифрової обробки сигналів, оптику, радіофізику та машинне навчання. Відповідно до класифікації руйнувань, найбільш поширеними їх типами є поздовжні, поперечні та сітчасті тріщини (alligator cracking), колії, вибоїни, просідання ґрунту, а також розшарування покриття [4, 5]. Кожен з цих типів має різну фізичну природу виникнення: тріщини найчастіше є наслідком термічного стиснення або втоми матеріалу, тоді як колії утворюються через пластичні деформації асфальтобетону під дією важкого транспорту. З огляду на таку різноманітність, для їх надійного виявлення застосовується широкий спектр технологій, кожна з яких має специфічні передумови використання, технологічні бар'єри та економічну доцільність.

1.1.1 Оптичні системи та методи комп'ютерного зору (Computer Vision)

На сьогоднішній день оптичні системи моніторингу, які базуються на принципах комп'ютерного зору (Computer Vision) та глибокого навчання (Deep Learning), є найбільш досліджуваними та найширше впроваджуваними у світовій практиці. Основна перевага оптичних камер полягає в їхній доступності, низькій вартості, високій роздільній здатності та портативності. Камери можуть бути інтегровані в різні рухомі платформи: від

спеціалізованих лабораторій на базі комерційних автомобілів до безпілотних літальних апаратів (дронів) та навіть звичайних смартфонів, встановлених на лобовому склі транспортних засобів [5].

Процес оптичного детектування починається з отримання двовимірного (2D) зображення поверхні дороги (RGB-формат). Далі зображення проходить етап попередньої обробки, під час якого усуваються шуми, вирівнюється контрастність та застосовуються алгоритми морфологічного перетворення для ізоляції областей інтересу (Region of Interest, ROI). Справжній прорив у цій сфері відбувся завдяки застосуванню згорткових нейронних мереж (CNN). Дослідження показують, що сучасні архітектури нейромереж здатні не лише фіксувати факт наявності дефекту, але й точно класифікувати його тип [4].

Зокрема, розробка ансамблевих моделей на базі архітектури YOLO (You Only Look Once), такої як YOLO-v4, дозволила досягти можливості обробки відеопотоку в режимі реального часу, зберігаючи при цьому високу точність. Наприклад, під час тестувань на міжнародних наборах даних (зокрема з Японії, Чехії та Індії), оптимізовані моделі YOLO-v4 досягли показника F1-score на рівні 0.628 для першого тестового набору та 0.6358 для другого [4]. Ефективність таких моделей прямо залежить від якості навчальних даних. У цьому контексті важливо відзначити роботу японських дослідників під керівництвом Х. Маеди, які вперше створили та відкрили для широкого загалу масштабний набір даних, що містить 9053 зображення доріг із 15435 екземплярами різних типів дефектів, зібраних протягом понад 40 годин за допомогою смартфонів у семи муніципалітетах Японії [5].

Сучасні архітектури для виявлення пошкоджень часто містять три ключові компоненти: магістральну мережу вилучення ознак (Backbone feature extraction network), мережу багатомасштабного злиття ознак (Multiscale feature fusion network) та класифікаційні модулі, такі як LMCA-Net [6]. Використання ієрархічних залишкових мереж (hierarchical residual networks) дозволяє витягувати складні геометричні патерни мікротріщин на тлі

неоднорідної текстури старого асфальту, а впровадження механізмів уваги (attention mechanisms) допомагає моделі фокусуватися саме на руйнуваннях, ігноруючи дорожню розмітку, тіні або випадкове сміття [6]. Архітектуру згорткової нейронної мережі для класифікації дефектів дорожнього покриття показано на рисунку 1.1.

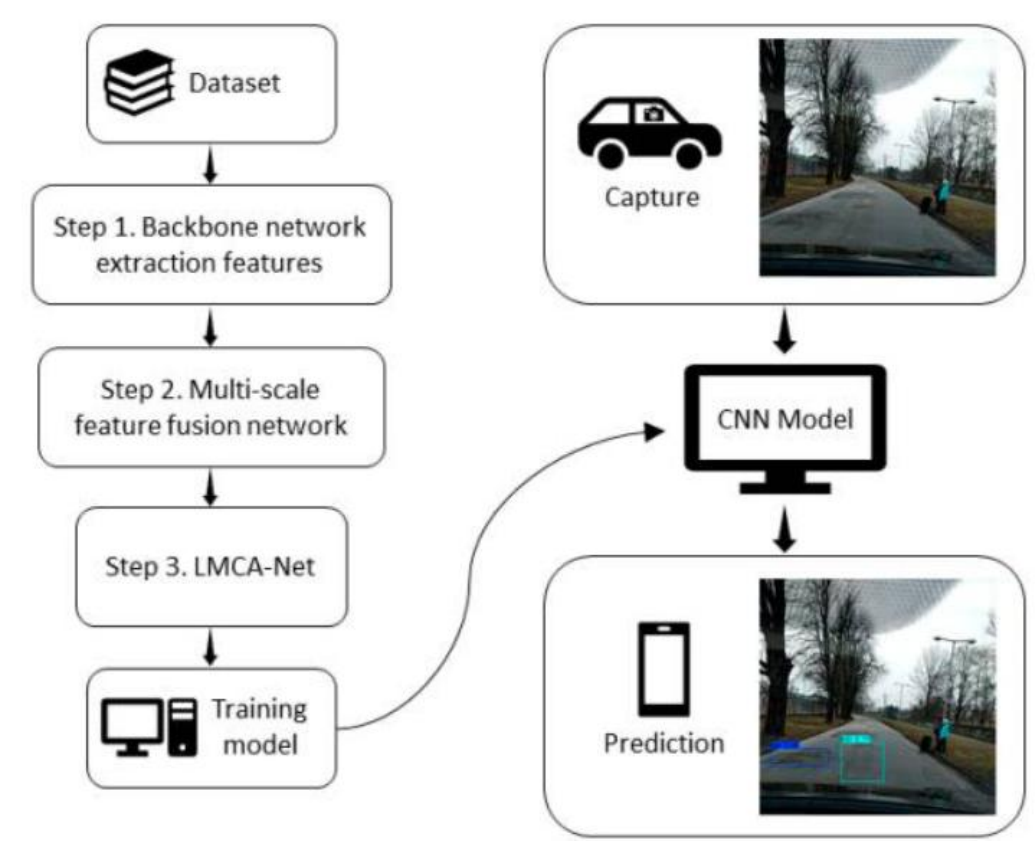


Рисунок 1.1 – Блок-схема архітектури згорткової нейронної мережі для класифікації дефектів дорожнього покриття, що включає магістральну мережу, модуль злиття ознак та фінальний класифікатор [6].

Проте, незважаючи на алгоритмічні успіхи, оптичні системи мають ряд фундаментальних фізичних обмежень. По-перше, вони вкрай чутливі до умов освітлення. Занадто яскраве сонячне світло викликає пересвічування кадрів (блумінг), нерівномірні тіні від дерев або транспортних засобів маскують тріщини, а робота в нічний час можлива лише за наявності потужного і рівномірного штучного підсвічування. Для часткового вирішення цієї

проблеми розробляються системи, що комбінують традиційні RGB-зображення з тепловізійними (Thermal) даними, оскільки тріщини часто мають іншу теплоємність та температуру, ніж суцільний асфальт [7]. По-друге, звичайні камери фіксують лише 2D-інформацію. Вони не здатні виміряти глибину вибоїни чи масштаб просідання, що є критично важливим параметром для оцінки рівня небезпеки дефекту згідно з державними стандартами [1, 2]. Як зазначається в оглядових роботах, використання оптичних систем на безпілотних літальних апаратах навіть у поєднанні з сучасними алгоритмами (наприклад, YOLOv3) демонструє недостатню точність виявлення дефектів в умовах складного фону, що вказує на наявність значного простору для подальшого вдосконалення алгоритмів [8].

1.1.2 Лазерне сканування, профілометрія та технології LiDAR

Для подолання обмежень 2D-систем в інженерній практиці все частіше застосовують методи отримання тривимірної (3D) топології дорожнього полотна за допомогою лазерних сенсорів зміщення та систем LiDAR (Light Detection and Ranging). Такі системи генерують високоточні масиви просторових даних, які називаються «хмарами точок» (Point Clouds). Лазерні вимірювачі працюють за принципом вимірювання часу прольоту світлового імпульсу (Time-of-Flight) або на основі оптичної триангуляції.

Технологія LiDAR вирізняється здатністю покривати великі відстані, має високу точність вимірювання кутів, надзвичайно високу роздільну здатність і широкий кут огляду. При використанні лінійних лазерних датчиків, встановлених на спеціальних вимірювальних транспортних засобах, можливо досягти побудови 3D-моделі поверхні з міліметровою та субміліметровою точністю (аж до 0,5 мм). Це робить лазерні методи ідеальними для вимірювання глибини колій, мікропрофілю дороги (рівності покриття за міжнародним індексом IRI) та визначення об'єму матеріалу, необхідного для ямкового ремонту.

Аналіз даних лазерного сканування здійснюється за допомогою спеціалізованих 3D-нейромереж, таких як PointNet, які здатні працювати безпосередньо з неструктурованими сирими хмарами точок, що обчислюють просторові характеристики дефектів. У ході досліджень було доведено, що метод виявлення змін з використанням хмар точок LiDAR, встановлених на автомобілях, досягає точності детектування 93,1%. Для специфічних дефектів точність розпізнавання становить 71,87% для випуклостей, 82,71% для глибоких вибоїн (ям) та 93,01% для сегментації загальних проїзних зон. Крім того, комбінація лазерних датчиків з гіроскопами утворює систему вимірювання 3D-форми всього дорожнього полотна на великих швидкостях.

Попри видатні метрологічні характеристики, впровадження LiDAR гальмується кількома факторами. По-перше, обладнання є дуже дорогим порівняно з камерами. По-друге, робота лазерів піддається значному впливу природного сонячного світла, теплового випромінювання поверхні та критично залежить від погодних умов. Під час дощу, туману, сильного вітру з пилом чи снігопаду лазерний промінь відбивається від крапель води чи часток бруду, генеруючи величезну кількість шумів, що унеможливорює точне сканування. Крім того, обробка щільних 3D-хмар точок для великих інфраструктурних сцен вимагає колосальних обчислювальних ресурсів, що ускладнює роботу таких систем в автономному режимі на борту транспортного засобу без зв'язку з потужними хмарними серверами [9].

1.1.3 Підповерхневе зондування за допомогою георадарів (GPR)

Оптичні камери та лазерні сканери здатні оцінювати лише видиму поверхню дороги. Однак багато катастрофічних руйнувань (наприклад, масштабні просідання або провали) починаються глибоко в структурі дорожнього одягу або в земляному полотні через вимивання ґрунту водою, утворення пустот чи розшарування між шарами асфальтобетону та щебеневої основи. Для ранньої діагностики таких підповерхневих аномалій

застосовується технологія георадіолокації (Ground Penetrating Radar, GPR).

Георадар випромінює короткі електромагнітні імпульси надвисокої частоти вглиб дороги за допомогою передавальної антени. Ці імпульси відбиваються від меж розділу середовищ, що мають різні діелектричні властивості (наприклад, межа між сухим асфальтом і вологим ґрунтом, або між асфальтом і повітряною порожнистю). Приймальна антена фіксує відбиті сигнали, формуючи радіолокаційний профіль (радограму) дорожньої конструкції в розрізі.

Основною перевагою GPR є те, що це метод неруйнівного контролю (Non-Destructive Testing, NDT). У минулому для аналізу стану нижніх шарів необхідно було застосовувати метод буріння кернів, що є деструктивним, дорогим і надає інформацію лише про локальні точки. Георадар дозволяє здійснювати безперервне сканування на робочій швидкості автомобіля (при використанні рупорних антен повітряного базування), працює в умовах міського шуму, не залежить від освітлення і дозволяє будувати інтуїтивно зрозумілі профілі розрізів доріг.

Головним викликом у використанні GPR є надмірна складність інтерпретації отриманих даних. Георадар генерує масиви «великих даних» (Big Data), які потребують складної математичної фільтрації для виділення корисного сигналу. Традиційно інтерпретація радограм покладалася на досвід і суб'єктивне судження експерта-геофізика. Для подолання суб'єктивності та прискорення аналізу сьогодні активно розробляються алгоритми на базі методу опорних векторів (Support Vector Machine, SVM), дерев рішень (Decision Tree) та глибоких нейромереж, які здатні автоматично розпізнавати патерни відбиття, характерні для порожнот чи водонасичених зон. Також варто зазначити, що ефективність георадара різко знижується при роботі на сильно вологих або глинистих ґрунтах через їх високу електропровідність, яка поглинає електромагнітні хвилі.

1.1.4 Вібраційні методи, інерціальні вимірювальні модулі (IMU) та смарт-матеріали

Інерціальні вимірювальні модулі (Inertial Measurement Unit, IMU), які зазвичай складаються з 3-осьових акселерометрів та гіроскопів, становлять основу для вібраційних методів моніторингу [10, 11]. Цей підхід є абсолютно відмінним від візуальних методів: замість фіксації форми об'єкта, вібраційний метод вимірює кінематичну реакцію транспортного засобу на нерівність дороги [10]. При наїзді колеса на вибоїну або колію підвіска автомобіля генерує специфічний коливальний імпульс, який миттєво фіксується акселерометром [11].

Методи на базі IMU мають безперечну перевагу з точки зору вартості та масштабованості. Датчики прискорення вже вбудовані у мільйони смартфонів та бортові комп'ютери сучасних автомобілів. Це створює ідеальні передумови для краудсорсингового збору даних (модель моніторингу за участю громади), коли тисячі водіїв, просто пересуваючись містом, формують загальну карту якості доріг для системи Smart City [5, 10]. Такі інновації знаходять застосування не лише в моніторингу, але й в автомобільній індустрії. Наприклад, компанія ClearMotion розробила інтелектуальну систему активної підвіски, яка за допомогою датчиків здатна передбачати, поглинати та компенсувати удари від вибоїн у режимі реального часу, паралельно створюючи базу даних про дефекти для забезпечення безпеки автономного водіння [12]. Впровадження вібраційного моніторингу також може реалізовуватися за допомогою гібридних систем прогнозування на базі нечітких нейронних мереж (fuzzy-neuro architectures), де вібраційні сигнали піддаються фаззифікації (переведенню у лінгвістичні змінні) і за набором логічних правил класифікуються як пошкодження або норма [13].

Окремим перспективним напрямом є інтеграція сенсорів безпосередньо у покриття, перетворюючи дорогу на активний елемент збору даних. Такі

системи, як зазначено в роботі, можуть включати п'єзоелектричні датчики, оптоволоконні системи на базі бреггівських решіток (Fiber Bragg Grating, FBG), а також MEMS-акселерометри та індуктивні петлі, що дозволяє фіксувати динамічну напругу, вібрації та деформації від проїжджаючого транспорту. Зокрема, FBG-сенсори забезпечують високу точність вимірювання деформацій і температури, що робить їх придатними для довгострокового структурного моніторингу дорожнього одягу [14]. Передовим рішенням є створення "самочутливих" (self-sensing) асфальтобетонних сумішей шляхом додавання струмопровідних матеріалів, таких як вуглецеві волокна (Carbon Fiber) або нановолокна [15, 16]. Зокрема, дослідження в галузі нанокompозитних матеріалів демонструють, що додавання вуглецевих нановолокон до цементних або асфальтобетонних сумішей дозволяє створити самосприймаючу конструкцію, здатну реєструвати власні деформації та пошкодження без необхідності в окремих вбудованих датчиках [15].

Під впливом динамічного навантаження від коліс електроопір такого асфальту змінюється (п'єзорезистивний ефект). Ця зміна розраховується як часткова зміна опору (Fractional Change in Resistance, FCR) і дозволяє системі визначати навантаження, вагу транспортного засобу та структурну цілісність матеріалу [16]. Такі інновації дозволяють реалізувати системи зважування в русі (Weigh-in-Motion, WIM), які на основі вбудованих сенсорів (зокрема FBG) забезпечують визначення ваги, швидкості та класифікацію транспортних засобів без зупинки потоку [14]. Окрім дротових систем на основі FBG, активно розвиваються бездротові вібраційні WIM-системи, які використовують акселерометри для вимірювання динамічних навантажень від осей вантажівок. Такі рішення дозволяють зменшити вартість розгортання та спростити інтеграцію в існуючу дорожню інфраструктуру завдяки відсутності потреби у прокладанні кабельних ліній [17].

Для систематизації та порівняння техніко-економічних характеристик проаналізованих методів сформовано Таблицю 1.1.

Таблиця 1.1 – Порівняльний аналіз технологій детектування дефектів дорожнього покриття [4, 9, 10, 12, 18]

Технологія / Метод	Принцип дії	Основні переваги	Основні недоліки	Залежність від погодних умов	Орієнтовна вартість
Оптичні камери (Computer Vision)	Отримання 2D RGB-зображень та їх аналіз через CNN (YOLO тощо)	Висока деталізація, легкість установки, висока швидкість обробки	Відсутність інформації про глибину дефекту, проблема тіней	Висока (критично залежить від освітлення)	Низька
LiDAR та Лазери	Сканування відбитим світлом (Time-of-Flight) для побудови 3D хмар точок	Міліметрова точність (до 0,5 мм), визначення глибини та об'єму вибоїн	Великі обсяги обчислень, розсіювання променя	Дуже висока (вразливість до дощу, туману, снігу)	Висока
Георадар (GPR)	Ехолокація електромагнітними хвилями підповерхневих шарів	Неруйнівний контроль структури, пошук прихованих пустот	Надзвичайна складність інтерпретації великих обсягів даних	Низька (за винятком сильного водонасичення ґрунту)	Середня / Висока
IMU / Вібраційний аналіз	Реєстрація прискорень (вібрацій) за допомогою акселерометрів	Дешевизна, можливість краудсорсингу через смартфони, незалежність від зору	Нездатність фіксувати площинні дефекти (сітку мікротріщин)	Відсутня (працює за будь-якої погоди)	Дуже низька

Враховуючи вищенаведене, можна стверджувати, що жоден із методів ізолювано не є ідеальним. Тому сучасний інженерний підхід тяжіє до концепції «сенсорного злиття» (sensor fusion), коли дешеві інерціальні датчики діють як фонові сканери, а в разі виявлення аномалії активуються точні візуальні інструменти (камери, LiDAR) для детальної класифікації. Проте, зібрані масиви сирих даних не мають цінності без надійної системи їх передачі, зберігання та обробки, що вимагає розгортання відповідних мережевих архітектур.

1.2 Дослідження архітектурних рішень та протоколів передачі даних в системах IoT

Концепція Інтернету речей (Internet of Things, IoT) виступає базовим інструментарієм цифровізації та автоматизації інтелектуальних транспортних систем [19, 20]. Впровадження IoT означає перехід від локальних ізолюваних систем збору даних до глобальної взаємопов'язаної екосистеми, яка охоплює сенсори, виконавчі механізми (actuators), контролери та обчислювальні платформи різного рівня [20]. Завдяки такій структурі можливий автоматичний збір даних про дорожній рух, дефекти полотна, вагу автомобілів (системи WIM) та метеоумови, що є основою для створення «цифрових двійників» (Digital Twins) міської інфраструктури та оптимізації процесів управління трафіком і ресурсами ремонтних служб.

1.2.1 Багаторівнева обчислювальна архітектура (Edge, Fog, Cloud)

Історично перші IoT-системи використовували централізовану хмарну модель, де всі «сірі» дані від датчиків передавалися на центральний сервер для обробки. Проте в умовах транспортних систем (особливо при використанні камер та LiDAR) генеруються колосальні обсяги даних, трансляція яких призводить до перевантаження каналів зв'язку та виникнення

неприпустимих затримок (latency). Для вирішення цієї проблеми сучасна архітектура розподіляється на три обчислювальні рівні: периферійні обчислення (Edge), туманні обчислення (Fog) та хмарні обчислення (Cloud) [19, 21].

Периферійні обчислення (Edge Computing) передбачають виконання обчислювальних операцій безпосередньо в місці генерації даних – на спеціалізованих IoT-шлюзах (IoT Edge Gateways) або бортових комп'ютерах транспортних засобів. Такі шлюзи, як, наприклад, USR-M300, володіють достатніми потужностями для локальної фільтрації шумів, попереднього аналізу даних та виконання алгоритмів машинного навчання безпосередньо на пристрої [22]. Яскравим прикладом необхідності Edge-рівня є автономні транспортні засоби. Коли безпілотне авто фіксує перешкоду чи небезпечну вибоїну, воно не може чекати, поки дані будуть відправлені в хмару, оброблені і повернуті назад у вигляді команди, оскільки ця затримка призведе до дорожньо-транспортної пригоди. Периферійні обчислення дозволяють сенсорам авто генерувати команду на гальмування чи маневр локально та миттєво (в межах мілісекунд) [21]. Крім того, на рівні інфраструктури IoT Edge Gateways здійснюють оптимізацію транспортного потоку та генерацію сигналів раннього попередження про аварії на основі аналізу швидкості та позиції автомобілів [22].

Хмарний рівень (Cloud Component) у гібридній системі відіграє роль глобального координатора. Передача відфільтрованих метаданих (замість сирих гігабайтів відео) на хмару знижує вимоги до пропускної здатності та витрати на зв'язок [23]. У хмарі зосереджені потужні сервери, які здійснюють просунуту аналітику великих історичних масивів даних (Big Data), предиктивне моделювання деградації дорожнього покриття, довгострокове зберігання інформації та синхронізацію з іншими муніципальними підсистемами [24]. Гібридна Cloud-Edge архітектура забезпечує масштабованість, адаптивність та стійкість системи керування розумним містом [23]. Багаторівневу IoT-архітектуру моніторингу дорожньої

інфраструктури ілюструє рисунок 1.2.

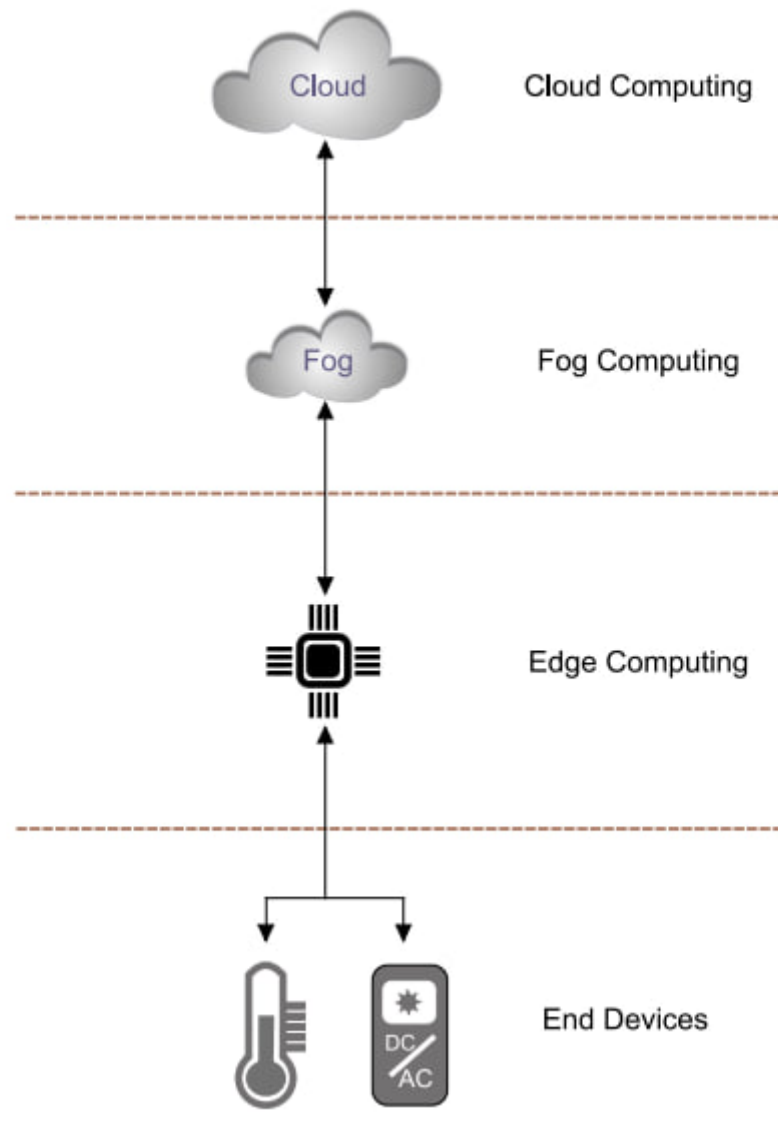


Рисунок 1.2 – Схема багаторівневої IoT-архітектури моніторингу дорожньої інфраструктури, що ілюструє взаємодію між рівнем сприйняття (сенсорами), Edge-шлюзами та центральною хмарною платформою аналітики [19].

1.2.2 Мережеві протоколи передачі даних (Network Layer)

Однією з найбільших перешкод у розгортанні систем дорожнього моніторингу є проблема зв'язку на великих дистанціях, особливо на заміських автомагістралях, де відсутнє стабільне електроживлення та

оптоволоконні лінії. Відповідно, сенсорні вузли мають працювати від акумуляторів або сонячних панелей протягом років [25]. Для задоволення цих жорстких умов були розроблені енергоефективні мережі великого радіусу дії --- LPWAN (Low Power Wide Area Networks) [25, 26]. На сучасному ринку IoT-зв'язку домінують дві основні технології цього класу: LoRaWAN та стільниковий NB-IoT [26, 27].

LoRaWAN (Long Range Wide Area Network): Це відкритий стандарт мереж, що працює в неліцензованому діапазоні радіочастот (наприклад, 868 МГц в Європі) [25, 26]. На фізичному рівні він базується на методі модуляції з розширенням спектра (Chirp Spread Spectrum --- CSS), що забезпечує високу стійкість до радіозавад, низьку ймовірність перехоплення та здатність сигналу проникати крізь перешкоди [25]. Завдяки цьому LoRaWAN може передавати дані на відстані до 15--20 км на відкритій місцевості при мізерному споживанні енергії [26]. У контексті моніторингу доріг головна перевага LoRaWAN полягає у можливості створення власних приватних мереж [25]. Муніципалітет або дорожня служба може встановити кілька власних базових станцій (шлюзів) і підключити до них тисячі вібраційних чи температурних датчиків в асфальті без необхідності сплачувати абонентську плату мобільним операторам за кожну SIM-карту [26]. Це радикально знижує довгострокові операційні витрати (OPEX). Водночас, ключовим недоліком LoRaWAN є надзвичайно низька швидкість передачі даних, яка придатна лише для періодичної відправки коротких телеметричних повідомлень, але абсолютно нездатна передавати аудіо, фото чи відеодані з камер [25, 26].

NB-IoT (Narrowband IoT) та LTE-M: Ці протоколи є стільниковими стандартами, розробленими консорціумом 3GPP, і працюють у ліцензованих частотних діапазонах інфраструктури 4G/5G [25, 27]. Головна сила NB-IoT полягає в тому, що він спирається на вже існуючі вишки мобільних операторів, тому розгортання системи в межах міста відбувається миттєво [26]. Він гарантує високу якість обслуговування (QoS) і здатний передавати трохи більші обсяги даних, ніж LoRaWAN [25, 27]. Проте ці переваги

супроводжуються недоліками: необхідністю купувати та адмініструвати SIM-карти для кожного пристрою (абонентська плата), вищим споживанням енергії, а також повною залежністю від покриття стільникового оператора, що робить його впровадження у віддалених районах ризикованим та часто неможливим без появи 5G-покриття [25, 27].

1.2.3 Протоколи прикладного рівня (Application Layer)

На прикладному рівні моделі взаємодії визначається структура самих повідомлень, логіка їх маршрутизації та механізми забезпечення доставки [28, 29]. Комплексне порівняння цих протоколів з точки зору енергоефективності, надійності доставки, накладних витрат та безпеки наведено в оглядовій роботі [30]. Автори цього дослідження підкреслюють, що вибір оптимального протоколу залежить від конкретних вимог застосунку: для систем, критичних до затримок, доцільніше використовувати MQTT поверх TCP, тоді як для автономних датчиків з обмеженим енергоживленням краще підходять CoAP або MQTT-SN. В екосистемі IoT-протоколів найбільш використовуваними для транспортної телематики є MQTT та CoAP [27, 31].

MQTT (Message Queuing Telemetry Transport): Цей протокол є міжнародним стандартом (OASIS) і базується на комунікаційній парадигмі «публікатор/підписник» (publish/subscribe) [27, 28]. Замість того, щоб датчики безпосередньо зверталися до сервера, вони відправляють (публікують) дані на проміжний вузол --- MQTT-брокер, з присвоєнням повідомленню певного "топіку" (теми). Всі системи, яким потрібна ця інформація, підписуються на даний топик і брокер автоматично розсилає їм повідомлення [27, 28]. MQTT працює поверх транспортного протоколу TCP, що гарантує надійність доставки (через систему підтверджень) та підтримує три рівні якості обслуговування (QoS 0, 1, 2). Завдяки здатності підтримувати постійне з'єднання та обробляти часті оновлення даних, MQTT є відмінним

вибором для систем управління трафіком та трансляції критичних попереджень про небезпеки на дорозі, де втрата пакета інформації є недопустимою. Проте використання TCP вимагає виконання процедури "рукоштовування" (handshake), що збільшує обсяг службового трафіку та швидше розряджає батарею пристрою [32].

CoAP (Constrained Application Protocol): На противагу MQTT, протокол CoAP був спеціально створений для найслабших пристроїв із жорстко обмеженими обчислювальними ресурсами та мікроскопічними лімітами енергії [31, 33]. CoAP використовує архітектуру "клієнт-сервер", що нагадує класичний веб-протокол HTTP (запити GET, POST, PUT, DELETE), що полегшує його інтеграцію з веб-сервісами [29, 33]. Важливою відмінністю є те, що CoAP працює поверх транспортного протоколу UDP, який здійснює передачу без встановлення з'єднання [31, 32]. Це колосально зменшує накладні витрати на передачу пакета (overhead), роблячи CoAP значно "легшим" та енергоефективнішим порівняно з MQTT [32, 33]. CoAP чудово показує себе поверх мереж NB-IoT та ідеально підходить для автономних вібраційних датчиків в асфальті, які можуть "прокидатися" раз на годину, миттєво відправляти короткий UDP-пакет і знову засинати, зберігаючи заряд батареї роками [32, 33]. Слабкою стороною CoAP є відсутність вбудованої гарантії доставки на мережевому рівні, тому розробникам доводиться реалізовувати механізми підтверджень безпосередньо в логіці додатку [33].

Крім того, варто згадати протокол LwM2M (Lightweight M2M), розроблений альянсом OMA [27, 31]. Його основне призначення - управління життєвим циклом кінцевого обладнання: віддалене налаштування конфігурацій, оновлення прошивок мікроконтролерів (FOTA) та перезавантаження датчиків, інтегрованих у транспортну інфраструктуру, без необхідності виїзду ремонтних бригад на місце [27, 31].

Таблиця 1.2 структуровано відображає основні технічні характеристики та сфери оптимального застосування розглянутих протоколів.

Таблиця 1.2 – Порівняльний аналіз IoT протоколів в системах моніторингу інфраструктури [26, 27, 28, 33]

Рівень мережі	Технологія / Протокол	Базовий транспорт / Модуляція	Архітектура / Модель	Основні переваги	Недоліки та обмеження
Мережевий	LoRaWAN	CSS (неліцензована частота)	Зіркоподібна	Незалежність від операторів зв'язку, радіус до 20 км, низька вартість	Вкрай мала пропускна здатність, не підходить для масивів фото/відео
Мережевий	NB-IoT	LTE (ліцензовані частоти 4G)	Стільникова архітектура	Відсутність потреби у власних шлюзах, гарантований QoS	Залежність від мобільних операторів, витрати на SIM-карти, вище енергоспоживання
Прикладний	MQTT	TCP	Pub/Sub (публікатор-підписник через Брокер)	Висока надійність доставки, чудово підходить для постійних потоків даних	Високі накладні витрати (overhead) через TCP-з'єднання

Продовження таблиці 1.2

Прикладний	CoAP	UDP	Клієнт-Сервер (RESTful, подібно до HTTP)	Ультранизьке споживання енергії, малий розмір пакетів, легка веб-інтеграція	Відсутність базової надійності доставки (без додаткових підтверджень)
Прикладний	LwM2M	UDP / TCP	M2M управління пристроями	Спеціалізований для оновлення прошивок і віддаленого керування сенсорами	Вузька спеціалізація (переважно для device management)

Крім архітектурних оптимізацій, розгортання глобальних транспортних IoT-мереж порушує гострі питання кібербезпеки. Оскільки системи працюють у відкритому просторі (SoA-based IoT), вони стають вразливими до втручань на всіх рівнях: від фізичного знищення та підміни даних (спуфінгу) на рівні датчиків до мережеских атак і викрадення аналітичної інформації на хмарному рівні [20]. Відповідно, при проектуванні рішень необхідно імплементувати складні системи управління довірою (Trust management systems), шифрування каналів зв'язку та механізми захисту приватності кінцевих користувачів, чиї смартфони чи автомобілі можуть виступати вузлами краудсорсингових мереж збору даних.

1.3 Висновки до розділу 1

На основі комплексного аналізу технологій детектування дефектів дорожнього покриття та архітектурних рішень систем моніторингу в

парадигмі Інтернету речей (IoT), сформовано наступні науково-практичні висновки.

Технологічна еволюція детектування руйнувань свідчить про системний перехід від візуального інспектування до повністю автоматизованих високоточних методів. Оптичні 2D-камери в синергії зі згортковими нейронними мережами (CNN, ансамблеві моделі YOLO) забезпечують масове та економічно вигідне детектування поверхневих дефектів (тріщин, вибоїн) у реальному часі, проте вони принципово обмежені у вимірюванні глибини руйнувань та критично залежать від умов освітлення. Впровадження 3D-методів (LiDAR та лазерна профілометрія) дозволяє вирішити проблему оцінки геометричних параметрів дефектів з міліметровою точністю, але зіштовхується з бар'єрами у вигляді складності обробки масивів хмар точок та чутливості лазерів до атмосферних опадів. У свою чергу, використання методів неруйнівного контролю на базі георадарів (GPR) є єдиним дієвим інструментом для раннього виявлення прихованих пустот і просідань в основі дорожнього одягу.

Усвідомлення недоліків індивідуальних методів обумовлює необхідність їх інтеграції в рамках парадигми «сенсорного злиття». Використання дешевих інерціальних модулів (IMU/вібраційних датчиків), інтегрованих у смартфони або розумні підвіски авто, створює потужну краудсорсингову базу для постійного виявлення кінематичних аномалій, що слугує сигналом для залучення прецизійних оптичних чи лазерних систем на конкретні ділянки. Перспективним напрямком є створення "розумного асфальту" з додаванням вуглецевих волокон чи оптоволоконних решіток (FBG), що наділяє саме полотно п'єзорезистивними властивостями для збору даних про динамічні навантаження транспортних засобів (WIM-системи).

Зважаючи на генерацію колосальних масивів "великих даних" (Big Data) інфраструктурними датчиками, традиційна централізована хмарна архітектура втрачає актуальність. Розгортання трирівневої моделі (Edge-Fog-Cloud) є критично необхідним. Периферійні обчислення (Edge)

безпосередньо на IoT-шлюзах дозволяють здійснювати миттєву фільтрацію даних і генерувати команди реагування з ультранизькою затримкою (latency), що має життєво важливе значення для інтелектуальних систем безпеки та автономного транспорту. Хмарний рівень при цьому бере на себе функції розширеної аналітики історичних даних та формування "цифрових двійників" транспортної мережі.

Вибір протоколів зв'язку вимагає ретельного балансування між енергоефективністю, пропускнуою здатністю та дальністю покриття. Для організації моніторингу на магістралях поза межами міст найефективнішим рішенням є розгортання приватних мереж LPWAN на базі стандарту LoRaWAN у комбінації з легким прикладним протоколом CoAP (на базі UDP). Це забезпечує багаторічну роботу автономних датчиків без абонентської плати операторам. Натомість для міської інфраструктури (Smart City), де пріоритетом є гарантована надійність доставки і висока частота оновлень, доцільніше використовувати протокол NB-IoT (стільникові мережі 4G/5G) у зв'язці з MQTT-брокерами, які гарантують надійну доставку інформації за рахунок використання протоколу TCP.

Таким чином, комплексна інтегрована модель, що поєднує мультимодальні сенсорні системи з оптимізованими Edge-Cloud архітектурами та адаптивними IoT-протоколами, створює потужний інструментарій для реалізації концепції предиктивного моніторингу, що дозволяє значно продовжити життєвий цикл автомобільних доріг та підвищити рівень безпеки учасників дорожнього руху.

2 ТЕОРЕТИЧНЕ ПРОЕКТУВАННЯ ТА ОБҐРУНТУВАННЯ СИСТЕМИ

2.1 Обґрунтування вибору апаратних компонентів та структури взаємодії сенсорів з хмарою

Сучасна парадигма розвитку інтелектуальних транспортних систем (ІТС) та управління дорожньою інфраструктурою вимагає безперервного, точного та масштабованого моніторингу стану покриття. Оскільки дорожня мережа є стратегічним активом будь-якої держави, що визначає логістичну та економічну ефективність, своєчасне виявлення дефектів стає критично важливим завданням. Класичні підходи, що базуються на використанні інерційних профілометрів, лазерних 3D-сканерів або ручних візуальних інспекцій, забезпечують еталонну точність, проте їх впровадження ускладнюється надзвичайно високою вартістю обладнання, складністю калібрування та неможливістю забезпечити постійне покриття всієї дорожньої мережі в режимі реального часу. Концепція Інтернету речей (ІоТ) пропонує принципово новий підхід: використання великого парку звичайних або автономних транспортних засобів як мобільних сенсорних вузлів, що безперервно збирають телеметричні дані під час щоденної експлуатації [11, 19].

Успішна реалізація такої системи вимагає глибокого теоретичного обґрунтування на кожному рівні архітектури: від вибору первинних перетворювачів (сенсорів) до методів локальної обробки (Edge Computing), протоколів передачі даних та технологій глобальних мереж (LPWAN). Архітектура повинна балансувати між точністю вимірювань, енергоефективністю, обчислювальною складністю та надійністю комунікацій в умовах високої мобільності.

2.1.1 Апаратна база: аналіз та вибір сенсорних компонентів

В основі вібраційного методу моніторингу лежить фізичний принцип

взаємодії системи «шина-підвіска-кузов» з мікро- та макротекстурою дорожнього одягу. Будь-яка аномалія на поверхні дороги (вибоїна, тріщина, наплив, просідання люка) викликає імпульсне або періодичне збурення, яке передається через елементи підвіски на кузов автомобіля, де воно може бути зафіксоване відповідними датчиками.

Огляд існуючих сенсорних технологій демонструє наявність кількох альтернативних рішень. Пізоелектричні сенсори та волоконно-оптичні решітки Брегга (Fiber Bragg Grating, FBG) демонструють феноменальну чутливість та довговічність, проте їхнє застосування обмежується стаціонарним вбудовуванням у тіло дорожнього одягу. Такі системи ідеально підходять для створення комплексів динамічного зважування (Weigh-in-Motion, WIM), оскільки вони здатні з високою точністю фіксувати навантаження на осі вантажних автомобілів і розраховувати накопичену втому дорожнього полотна. Аналогічно, інтелектуальні цементні або асфальтобетонні композити з додаванням вуглецевих нановолокон (self-sensing concrete nanocomposites) відкривають шлях до створення інфраструктури, що самостійно діагностує власні мікротріщини, проте ці технології перебувають на стадії лабораторних випробувань та точкових впроваджень і не можуть бути розгорнуті на існуючих дорогах без їх повної реконструкції [14].

Для створення масштабованої системи на базі рухомих транспортних засобів єдиним раціональним вибором залишаються мікроелектромеханічні системи (MEMS). Останнє десятиліття ознаменувалося експоненційним зростанням характеристик недорогих MEMS-акселерометрів, що дозволило використовувати їх для завдань моніторингу структурного здоров'я (Structural Health Monitoring, SHM) та діагностики дорожнього покриття [11, 14, 19]. Дослідження ефективності низьковартісних сенсорів доводить, що сучасні триосьові цифрові акселерометри (наприклад, серії ADXL або еквівалентні рішення від інших виробників) здатні забезпечити відмінне співвідношення сигнал/шум [11, 34, 35].

Для надійного розпізнавання високодинамічних подій (таких як удар колеса об гострий край вибоїни на швидкості 80 км/год) сенсорний вузол повинен задовольняти низку жорстких апаратних вимог. По-перше, частота дискретизації (sampling rate) повинна становити не менше 250–333 Гц. Як свідчать результати спектрального аналізу вібрацій, основна енергія відгуку підвіски зосереджена в діапазоні до 10-20 Гц, проте короткочасні ударні імпульси містять високочастотні гармоніки, які зрізаються (піддаються ефекту аліасингу) при недостатній частоті опитування. По-друге, критичним параметром є спектральна щільність шуму (Noise Density). Оптимальні датчики для ІТС повинні мати щільність шуму на рівні близько $51 \mu\text{g}/\sqrt{\text{Hz}}$, що дозволяє впевнено виділяти корисний сигнал мікроскопічної амплітуди на тлі власних шумів кристала акселерометра [34, 35, 36]

Розгортання MEMS-акселерометрів може здійснюватися двома шляхами: шляхом створення спеціалізованих бортових IoT-вузлів (вбудованих у платформу транспортного засобу або встановлених на приладовій панелі/підлозі) та шляхом краудсорсингу даних з акселерометрів звичайних смартфонів водіїв. Хоча смартфони є надзвичайно привабливим рішенням завдяки нульовій вартості апаратного розгортання, спеціалізовані бортові модулі на базі мікроконтролерів ARM або архітектури Raspberry Pi/Arduino забезпечують значно вищу надійність, оскільки вони виключають проблеми вільного переміщення смартфона в салоні та гарантують точну мікросекундну синхронізацію часових міток (timestamps) з даними глобальних систем позиціонування. Разом з тим, інтеграція MEMS-сенсорів у бездротові сенсорні мережі (наприклад, на базі стандарту Zigbee всередині транспортного засобу) дозволяє об'єднувати дані від кількох датчиків, розміщених на різних осях, що суттєво підвищує загальну точність виявлення дефектів [35, 37, 38].

2.1.2 Розподілена обчислювальна архітектура: Edge Intelligence та Cloud Computing

Традиційна архітектура Інтернету речей, у якій усі «сирі» (необроблені) дані з датчиків безпосередньо транслуються на центральні хмарні сервери для зберігання та аналізу, виявляється абсолютно нежиттєздатною в контексті інтелектуальних транспортних систем. Безперервний потік тривимірних векторів прискорення з частотою 300 Гц генерує мегабайти даних щохвилини для кожного автомобіля. Передача таких обсягів через мобільні мережі створює колосальне навантаження на пропускну здатність комунікаційних каналів, призводить до неприпустимих затримок та генерує величезні витрати на хмарне зберігання [23, 39]

Для вирішення цієї проблеми система проектується на основі парадигми периферійних обчислень (Edge Computing) у тісній співпраці з хмарними інфраструктурами (Cloud Computing). Периферійні обчислення передбачають перенесення значної частини аналітичних та обчислювальних завдань безпосередньо на бортовий шлюз транспортного засобу (Edge Gateway) або на базові станції мобільної мережі [21, 22]. Наділення периферійних пристроїв обчислювальним інтелектом (Edge Intelligence) кардинально змінює логіку роботи системи.

На рівні Edge-пристрою (мікроконтролера в автомобілі) в режимі реального часу виконуються алгоритми попередньої фільтрації, перетворення координат, сегментації сигналу та первинного висновку (inference) моделей машинного навчання. Завдяки цьому бортовий пристрій не передає в мережу сирі вібраційні часові ряди; натомість він формує компактний пакет метаданих, який містить лише витяг з події. Цей пакет включає точні GPS-координати, ідентифікований тип дорожньої аномалії (наприклад, «глибока вибоїна», «поперечна тріщина» або «люк»), індекс впевненості моделі (confidence score) та позначку часу. Такий підхід забезпечує три фундаментальні переваги:

1. Забезпечується мінімальна затримка реакції (Real-Time Response), що є надзвичайно важливим для підсистем автономного водіння, які можуть використовувати ці дані для миттєвого коригування траєкторії руху або адаптації інтелектуальної підвіски для пом'якшення удару [22].

2. Вирішується проблема роботи в умовах нестабільного зв'язку, характерного для сільської місцевості або гірських доріг. Якщо зв'язок втрачається, компактні метадані легко буферизуються в локальній пам'яті пристрою і відправляються пакетом після відновлення з'єднання.

3. Енергоефективність усієї системи зростає в геометричній прогресії, оскільки радіопередавач використовується лише спорадично для надсилання коротких повідомлень, а не для безперервного стрімінгу.

Хмарний рівень (Cloud), у свою чергу, звільняється від рутинної обробки сигналів і фокусується на високорівневих завданнях. Сучасні хмарні платформи для мобільної інфраструктури (Mobility Infrastructure Cloud) агрегують звіти від тисяч транспортних засобів, відсіюють поодинокі хибні спрацьовування за допомогою алгоритмів просторової кластеризації та формують динамічну теплову карту дефектності дорожнього одягу в масштабах всієї країни. Крім того, хмара відповідає за процеси федеративного навчання (Federated Learning) або централізованого перенавчання моделей на основі агрегованих зразків, після чого оновлені та більш точні вагові коефіцієнти нейронних мереж поширюються на Edge-пристрої шляхом оновлень по повітрю (OTA) [23].

2.1.3 Порівняльний аналіз IoT-протоколів та вибір брокерів повідомлень

Передача сформованих на борту метаданих вимагає використання оптимізованих прикладних мережевих протоколів. Традиційний протокол HTTP, який базується на текстовому форматі та парадигмі "запит-відповідь", має занадто великий розмір заголовків (overhead) і є неефективним для IoT.

Дослідження протоколів прикладного рівня виділяють трьох основних кандидатів: MQTT, CoAP та LwM2M.

Протокол CoAP (Constrained Application Protocol) створений для мереж з вкрай обмеженими ресурсами. Він працює поверх протоколу UDP, повторюючи семантику RESTful HTTP (GET, POST, PUT), але в бінарному, компактному вигляді. Оскільки UDP не гарантує доставки пакетів, CoAP реалізує власні легковагові механізми підтвердження. LwM2M (Lightweight M2M) розширює архітектуру CoAP, фокусуючись переважно на задачах управління самими пристроями (оновлення прошивок, моніторинг батареї), а не лише на передачі телеметрії.

Протокол MQTT (Message Queuing Telemetry Transport), навпаки, працює поверх надійного протоколу TCP і використовує архітектуру «публікація-підписка» (Publish/Subscribe) через центрального посередника — брокера повідомлень. Edge-пристрій публікує дані у визначений топік (наприклад, vehicle/105/pothole), а хмарні аналітичні сервіси підписуються на цей топік. MQTT підтримує три рівні якості обслуговування (QoS): QoS 0 (доставка щонайбільше один раз), QoS 1 (доставка щонайменше один раз з підтвердженням) та QoS 2 (гарантована доставка рівно один раз). Враховуючи те, що втрата повідомлення про критичну вибоїну може призвести до аварії наступного автономного транспортного засобу, використання MQTT з рівнем QoS 1 або QoS 2 робить його стандартом де-факто для інтелектуальних транспортних систем [27].

Вибір конкретного брокера повідомлень для агрегації MQTT-трафіку є не менш важливим, особливо при розгортанні транзитних вузлів обчислень на периферії (Edge node), де обчислювальні ресурси та оперативна пам'ять суворо лімітовані. Широкомасштабне бенчмаркінгове дослідження брокерів повідомлень виявило критичні залежності між мовою програмування брокера, його архітектурою та продуктивністю.

В експериментах, що симулювали одночасну роботу до 10 000 клієнтських пар (публікатор-підписник) в ізольованих контейнерних

середовищах на апаратному забезпеченні, типовому для Edge-пристроїв, було доведено, що брокери, розроблені на базі віртуальної машини Java (JVM) — такі як HiveMQ та ActiveMQ Artemis — хоча й пропонують багатий корпоративний функціонал та високу надійність (enterprise features), споживають у 10–50 разів більше оперативної пам'яті порівняно з нативними реалізаціями. Такий надмірний апетит до пам'яті є критичним недоліком для вбудованих систем.

З іншого боку, нативні багатопотокові брокери, такі як NATS та Zenoh, демонструють найвищу ефективність масштабування в умовах високого навантаження. За даними тестування, NATS здатен обробляти до 90 000 повідомлень за секунду при 10 000 одночасних підключень, забезпечуючи при цьому субмілісекундні затримки на рівні персентилів p50 та p95. Знаменитий і широко використовуваний брокер Mosquitto (написаний на C), попри свою надзвичайну легковаговість та мінімальне споживання пам'яті, має фундаментальне архітектурне обмеження: він є однопотоковим (single-threaded). Через це його продуктивність жорстко насичується і досягає плато на позначці приблизно 45 000 повідомлень за секунду, оскільки він не може утилізувати багатоядерні процесори [40]. Порівняльну характеристику продуктивності брокерів повідомлень для IoT Edge Computing наведено в таблиці 2.1.

Таблиця 2.1 - Порівняльна характеристика продуктивності брокерів повідомлень для IoT Edge Computing (на основі синтезованих результатів бенчмаркінгу) [27, 40]

Брокер повідомлень	Архітектурна платформа	Масштабованість (макс. пропускна здатність)	Затримка (p95)	Споживання ресурсів (ОЗП)	Оптимальний сценарій використання
NATS Server	Нативна (Go)	Дуже висока (~90k msg/s)	Субмілісекундна	Низьке	Високонавантажена потокова телеметрія
Zenoh Router	Нативна (Rust)	Висока	Дуже низька	Дуже низьке	Децентралізовані Edge-системи, робототехніка
Mosquitto	Нативна (C)	Середня (~45k msg/s)	Низька	Мінімальне	Легкі Edge-шлюзи з низькою щільністю пристроїв
RabbitMQ	Erlang (AMQP/MQTT)	Висока	Середня	Помірне	Системи з потребою у складній маршрутизації
HiveMQ / Artemis	Java Virtual Machine	Середня / Висока	Висока	Дуже високе (в 10-50х більше)	Централізовані хмарні сервери без обмежень ОЗП

Схему компонентів системи виявлення аномалій на основі вібрації у вигляді UML-діаграми наведено на рисунку 2.1.

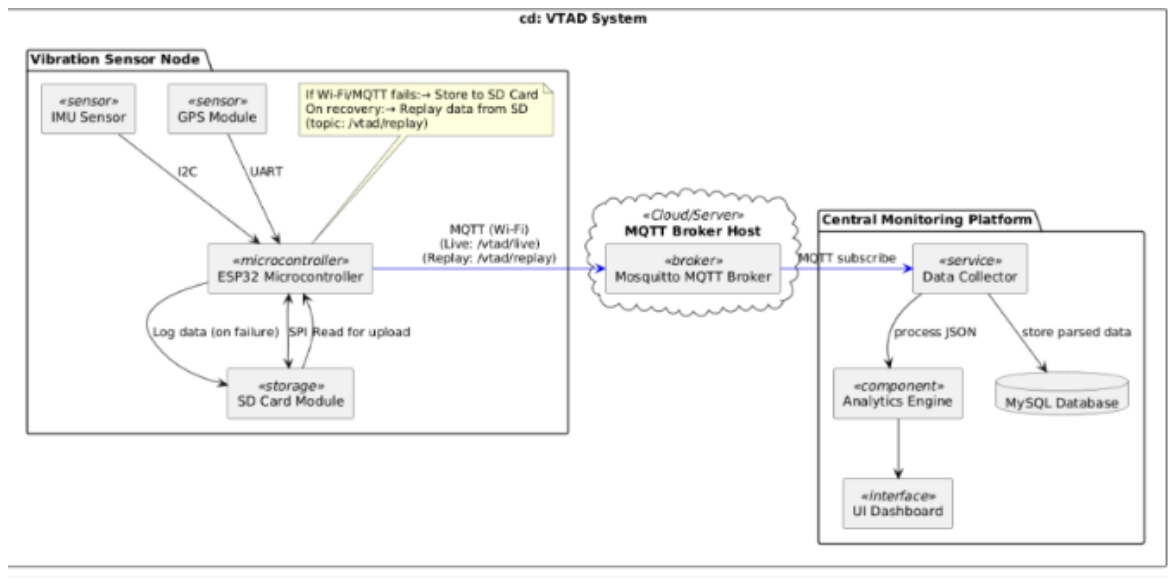


Рисунок 2.1 - Схема компонентів уніфікованої мови моделювання (UML) системи виявлення аномалій треків на основі вібрації (VTAD).

Відповідно до цих даних, для побудови дворівневої системи ІТС доцільно застосовувати Mosquitto або Zenoh безпосередньо на легких локальних шлюзах транспортних засобів, а для агрегації на рівні регіональних обчислювальних вузлів (MEC — Multi-access Edge Computing) використовувати NATS для забезпечення максимальної пропускної здатності.

2.1.4 Аналіз технологій енергоефективних глобальних мереж (LPWAN)

Окрім протоколів прикладного рівня, вирішальне значення має фізичний транспортний рівень. Передача даних від автомобіля, що рухається міжміськими трасами або сільськими дорогами, вимагає великого радіусу дії та надійного покриття. Традиційні стільникові мережі 4G (LTE) та 5G пропонують надмірну пропускну здатність, але споживають багато енергії і часто мають "білі плями" поза межами великих міст. З іншого боку, технології ближнього радіусу дії (Wi-Fi, Zigbee, Bluetooth) абсолютно непридатні для зв'язку автомобіля з хмарою, хоча й використовуються для

об'єднання датчиків всередині кузова [25, 41]

Саме тому фундаментом судної IoT-комунікації стали технології LPWAN (Low Power Wide Area Networks). Вони спеціально розроблені для забезпечення великої дальності зв'язку (до десятків кілометрів), високої проникаючої здатності та наднизького енергоспоживання ціною зниження швидкості передачі даних [25, 26, 42]. Екосистема LPWAN поділяється на дві глобальні категорії: технології, що працюють у неліцензованому спектрі (переважно ISM-діапазони 868/915 МГц), та стільникові технології у ліцензованому спектрі радіочастот. [25, 26] До першої категорії належать LoRaWAN та Sigfox, до другої — стандарти 3GPP: NB-IoT (Narrowband IoT) та LTE-M (Long Term Evolution for Machines).

Дослідження практичного розгортання цих технологій демонструють значні відмінності між теоретичними специфікаціями та реальною продуктивністю в польових умовах [25, 42, 43].

Технологія LoRaWAN базується на запатентованій модуляції CSS (Chirp Spread Spectrum), яка робить сигнал вкрай стійким до шумів та інтерференції. У реальних проектах LoRaWAN впевнено забезпечує зв'язок на відстані до 11 км у сільській місцевості та кілька кілометрів у щільній міській забудові. Завдяки відсутності необхідності постійної синхронізації з базовою станцією, модулі LoRaWAN здатні працювати роками від однієї батареї. Проте, технологія використовує протокол ALOHA для доступу до середовища, що породжує серйозні проблеми масштабованості: при концентрації понад 1000 пристроїв на один шлюз кількість колізій пакетів зростає експоненційно, призводячи до втрат даних. Більше того, LoRaWAN обмежує розмір корисного навантаження (від 11 до 242 байт) і дуже погано справляється з ефектом Доплера та відстеженням швидко рухомих об'єктів. Технологія Sigfox має ще більший радіус дії (зафіксовані рекорди понад 280 км) завдяки вузькосмуговій (Ultra-Narrowband) модуляції, але жорсткі ліміти (максимум 12 байт у повідомленні та не більше 140 повідомлень на добу) роблять її непридатною для постійного моніторингу вібрацій.

Технологія NB-IoT працює на базі існуючої інфраструктури операторів мобільного зв'язку. Вона забезпечує видатну проникаючу здатність і надійність з бюджетом каналу зв'язку (link budget) на рівні 164 дБ. Практичні тести на комерційних мережах фіксують рівень успішної доставки пакетів (Packet Delivery Ratio) від 96% до 100% навіть при екстремально низькому рівні сигналу в -127 дБм. NB-IoT здатна обслуговувати десятки тисяч вузлів на одну стільникову комірку завдяки технології OFDMA на низхідному каналі та SC-FDMA на висхідному. Проте NB-IoT має фундаментальний недолік для автомобільних застосувань: протокол спочатку проектувався для стаціонарних пристроїв (наприклад, розумних лічильників води) і не підтримує «безшовний» хендовер (seamless handover) між базовими станціями. Відповідно, під час руху транспортного засобу модем змушений постійно розривати сесію та заново реєструватися в новій стільниковій комірці, що призводить до затримок у кілька секунд, можливої втрати пакетів і різкого зростання енергоспоживання, що нівелює всі заявлені енергозберігаючі функції (такі як PSM або eDRX).

На противагу цьому, технологія LTE-M (або Cat-M1) пропонує ідеальний баланс для ІТС. Вона забезпечує значно більшу пропускну здатність (до 1 Мбіт/с проти 250 кбіт/с у NB-IoT), мінімальні затримки (менше 200 мс) та, головне, повну підтримку мобільності з класичним хендовером під час перемикання між вежами зв'язку [42]. Порівняльний аналіз технологій LPWAN у контексті мобільного моніторингу дорожньої інфраструктури представлено в таблиці 2.2.

Таблиця 2.2 - Порівняльний аналіз технологій LPWAN у контексті мобільного моніторингу дорожньої інфраструктури [25, 26, 42, 43]

Параметр	LoRaWAN	Sigfox	NB-IoT	LTE-M
Смуга частот	Неліцензована (ISM)	Неліцензована (ISM)	Ліцензована (LTE)	Ліцензована (LTE)
Максимальна пропускна здатність	50 кбіт/с	0.1 кбіт/с	~250 кбіт/с	~1 Мбіт/с
Затримка сигналу (Latency)	Кілька секунд	Секунди/Хвилини	1.2 – 10 секунд	< 200 мілісекунд
Корисне навантаження (Payload)	11 – 242 байти	12 байт	До 1600 байт	Без жорстких обмежень
Підтримка мобільності (Handover)	Дуже обмежена	Обмежена	Відсутня (стаціонарний фокус)	Повна (ідеально для авто)
Бюджет зв'язку / Проникність	~157 дБ	Висока	Відмінна (до -127 дБм)	Добра (проблеми нижче -113 дБм)
Енергоефективність (батарея)	Відмінна (> 5 років)	Найкраща	Добра (падає під час руху)	Задовільна

Зважаючи на вищенаведені факти, архітектура системи передачі даних транспортного засобу повинна базуватися на технології LTE-M як основному каналі зв'язку, що здатний швидко і безперервно передавати аналітику в русі. У зонах глибокого згасання сигналу (де рівень падає нижче -113 дБм і LTE-M відключається), система може автоматично переходити на використання NB-IoT як резервного каналу для відкладеної передачі даних, оскільки NB-IoT

зберігає зв'язок у складніших умовах покриття [42].

2.2 Алгоритми попередньої обробки та фільтрації сенсорних даних

Сигнали, що надходять від MEMS-акселерометрів, вбудованих у транспортний засіб, ніколи не є ідеальним відображенням лише дорожнього профілю. Вібрація кузова автомобіля є суперпозицією (накладанням) численних хвильових процесів [10]. До корисного сигналу, згенерованого взаємодією шини з ямами чи тріщинами, неминуче додається високоенергетичний шум: вібрації від роботи двигуна внутрішнього згорання (які залежать від обертів колінчастого вала), низькочастотні коливання на ресорах і пружинах, шум аеродинамічного опору, резонансні явища елементів кріплення датчика, а також внутрішній тепловий та електричний шум самої мікросхеми акселерометра. Відтак, подача "сирих" даних безпосередньо у алгоритми класифікації призведе до катастрофічно високого рівня хибних спрацьовувань (False Positives). Для виділення істинного сигналу необхідний потужний математичний апарат попередньої обробки [11].

2.2.1 Базова фільтрація: усунення високочастотних шумів та дрейфу нуля

На нульовому етапі алгоритму виконується просторова нормалізація. Оскільки смартфон або спеціалізований бортовий модуль може бути встановлений під довільним кутом відносно осей автомобіля, сирі координати X , Y , Z не збігатимуться з реальними фізичними векторами руху (повздовжним, поперечним та вертикальним). Застосовуючи математичні моделі матриць поворотів (кватерніони) та поєднуючи дані гіроскопа і акселерометра через фільтр Калмана або комплементарний фільтр, система виконує віртуальну переорієнтацію осей, спрямовуючи вісь Z строго

паралельно вектору гравітації Землі. Саме вертикальна вісь Z містить до 90% корисної інформації про вибоїни, хоча осі X (повздовжня) та Y (поперечна) також несуть інформацію про гальмування під час наїзду на перешкоду або зміщення автомобіля в колії [38, 44].

Після орієнтації застосовуються традиційні лінійні та нелінійні методи згладжування (smoothing) у часовій області. Найпопулярнішими підходами є використання фільтрів ковзного середнього (Moving Average Filter) та медіанних фільтрів (Median Filter). Дослідження ефективності базових фільтрів для зменшення шуму акселерометра виявили концептуальні відмінності між ними. Фільтр ковзного середнього працює як лінійний фільтр низьких частот: він бере арифметичне середнє точок у заданому вікні. Хоча він чудово гасить білий гаусів шум, він має суттєвий недолік — «розмиває» і згладжує різкі імпульсні перепади. Оскільки удар об гострий край ями є саме імпульсною, перехідною подією (з високою швидкістю наростання амплітуди), розмиття цього фронту призводить до втрати критично важливої ознаки дефекту. Натомість медіанний фільтр, будучи нелінійним алгоритмом, замінює центральне значення у вікні на медіану відсортованих значень. Ця математична властивість дозволяє медіанному фільтру феноменально ефективно усувати одиничні імпульсні шуми (так званий шум «сіль і перець», що виникає через електричні наведення в цифрових шинах I2C/SPI) при цьому ідеально зберігаючи різкі перепади (edges) корисного сигналу, що є життєво необхідним для подальшого аналізу. Для боротьби з низькочастотним дрейфом ізолінії (наприклад, тривала зміна кута нахилу дороги на підйомі) застосовуються лінійні фільтри високих частот (High-Pass Filters), що залишають лише динамічну складову прискорення.

Натомість медіанний фільтр, будучи нелінійним алгоритмом, замінює центральне значення у вікні на медіану відсортованих значень. Ця математична властивість дозволяє медіанному фільтру феноменально ефективно усувати одиничні імпульсні шуми (так званий шум «сіль і

перець», що виникає через електричні наведення в цифрових шинах I2C/SPI) при цьому ідеально зберігаючи різкі перепади (edges) корисного сигналу, що є життєво необхідним для подальшого аналізу [10]. Для боротьби з низькочастотним дрейфом ізоляції (наприклад, тривала зміна кута нахилу дороги на підйомі) застосовуються лінійні фільтри високих частот (High-Pass Filters), що залишають лише динамічну складову прискорення [44].

2.2.2 Адаптивна вейвлет-трансформація та просторова кореляція

Хоча базові фільтри достатні для усунення найпростіших артефактів, вібрація від дорожнього полотна має виражену нестационарну природу (її спектральний склад стрімко змінюється в часі). Метод перетворення Фур'є ефективний для стаціонарних гармонічних сигналів, але втрачає роздільну здатність у часі. Для одночасного аналізу сигналу і в частотній, і в часовій області найефективнішим інструментом є вейвлет-перетворення (Wavelet Transform) [10, 40].

Як було доведено у фундаментальному дослідженні Bello-Salau et al. (2019), використання алгоритму просторової фільтрації на основі масштабу вейвлет-перетворення (Adaptive Wavelet Transformation Scale Space Filtering, AWSSF) дозволяє досягти безпрецедентної якості шумозаглушення зі збереженням корисної інформації [45]. Алгоритм розкладає складний одномірний сигнал вібрації на набір базисних функцій, локалізованих у просторі (часі).

Математична модель типового вейвлет-перетворення описується рівнянням [45]:

$$f(x) = \sum_{j=1}^J \sum_{k=1}^K \Psi_{j,k}(x) W(j, k)$$

де:

- $\Psi_{j,k}(x)$ — нормалізована дуальна базисна функція (вейвлет);

- J — максимальна кількість шкал (рівнів) розкладання;
- K — максимальна позиція (індекс часу) в оригінальному сигналі;
- $W(j, k)$ — вейвлет-коефіцієнт на масштабі j та позиції k .

Серед усього різноманіття вейвлетних сімейств (Haar, Symlets, Coiflets) експериментальні дослідження форми дорожніх аномалій показали, що материнський вейвлет Добеші 2-го порядку (Daubechies-2, db2) має найвищу математичну кореляцію з типовим профілем вихідного сигналу акселерометра під час наїзду колеса на перешкоду [45]. Ця функція визначається параметрами масштабу (розтягування) a та зсуву (трансляції) τ :

$$\Psi_{a,\tau}(t) = \frac{1}{\sqrt{a}} \Psi\left(\frac{t - \tau}{a}\right) \quad \text{для } a \neq 0$$

Для реалізації в дискретних обчислювальних системах використовуються діадні (кратні двом) параметри розкладання, де змінні масштабу та трансляції дискретизуються:

$$\Psi_{j,k}(x) = 2^{-j/2} \Psi(2^{-j}x - k)$$

Коефіцієнти розкладання $W(j, k)$, які містять інформацію про ступінь подібності фрагмента сигналу до вейвлет-функції на певному масштабі, обчислюються через згортку:

$$W(j, k) = \int_{-\infty}^{\infty} f(x) \Psi_{j,k}(x) dx$$

Модель просторової кореляції. Інноваційною особливістю алгоритму AWSSF є його здатність відрізнати реальну фізичну аномалію від високочастотного гаусівського шуму. Фізична суть цього підходу полягає в тому, що справжня аномалія (наприклад, удар колеса) містить енергію, яка

простягається (відображається) через кілька суміжних рівнів розкладання (шрифтів або масштабів). Натомість випадковий шум концентрується лише на найдрібніших масштабах (найвищих частотах) і швидко згасає на більших масштабах. Щоб математично підсилити істинні події, алгоритм здійснює пряме перемноження вейвлет-коефіцієнтів, які існують на суміжних масштабах. Ця просторова кореляційна функція $W(j, k)$ обчислюється як [45]:

$$\mathcal{W}(j, k) = \prod_{i=0}^{J-1} W(j + i, k)$$

Подальші кроки алгоритму передбачають масштабування потужності отриманих скорельованих даних $\mathcal{W}(j, k)^2$ відносно оригінальних коефіцієнтів $W(j, k)$. Це створює рельєфну карту сигналу, де випадковий шум знищений (оскільки на деяких масштабах його коефіцієнти близькі до нуля, множення обнуляє результат), а значущі піки від аномалій багаторазово підсилені [7]. Згодом алгоритм ідентифікує піки у відфільтрованих скорельованих коефіцієнтах і використовує їх для визначення точних границь сигналу (початку та кінця події наїзду). На базі цих границь генерується додатковий просторовий фільтр (з використанням функції одиничного прямокутного вікна), що застосовується для остаточного згладжування, видаляючи будь-які перехідні коливання. Результуючий "очищений" сигнал (denoised signal) ідеально підготовлений для класифікації. Ця техніка гарантує високу конвергенцію та низьку обчислювальну складність, успішно адаптуючись до постійно змінюваних умов руху [45, 46].

2.2.3 Алгоритми адаптивного порогування та Edge-компресія

Наступною проблемою після фільтрації є визначення порогу (thresholding) — значення амплітуди або енергії, вище якого подія

класифікується як "можливий дефект", а нижче — ігнорується. Традиційні системи Scale Space Filtering використовували жорстко задані, фіксовані пороги. Проте такий підхід неминуче зазнає краху на практиці: випадкові флуктуації, зміна швидкості руху, перехід асфальтобетону в бруківку або зміна характеристик жорсткості покриттів призводять до того, що фіксований поріг генерує або шквал хибних тривог, або пропускає реальні небезпеки [45]

Для розв'язання цієї проблеми впроваджуються алгоритми адаптивного порогування (Adaptive Thresholding Technique, ATT). Інноваційна концепція, описана дослідниками, передбачає динамічний перерахунок порогового значення T_A на основі локальних статистик самого сигналу. Адаптивні алгоритми безперервно оцінюють дисперсію та рівень шуму у невеликому ковзному вікні та піднімають або опускають поріг чутливості пропорційно до фонових вібрацій [45, 47]

Сучасні архітектури, такі як мережа з адаптивним подвійним порогуванням (Adaptive Dual-Threshold Network, ADT-Net), йдуть ще далі: вони динамічно генерують диференційовані пороги окремо для зон ймовірного "фону" і зон ймовірних "об'єктів", інтегруючи попиксельні (або поточкові) оцінки аномалій для суттєвого підвищення гранулярності розпізнавання. Застосування цього механізму дозволяє системі швидко розпізнавати, чи є поточний сигнал результатом руху по розбитій ґрунтовій дорозі (де поріг необхідно підняти), чи по ідеальному швидкісному шосе (де поріг необхідно опустити для виявлення дрібних тріщин) [48].

Не менш важливою функцією адаптивного порогування є стиснення інформації для систем Edge Computing. Ефективний адаптивний поріг працює як інтелектуальний гейт (gate): він автоматично відкидає до 78% "нерелевантних" або "нормальних" часових рядів, які не несуть цінності. Експериментальні підтвердження свідчать, що використання порогу $\tau = 10$ здатне радикально зменшити добовий обсяг накопичених телеметричних даних з 54 мегабайт до 12 мегабайт на один пристрій. Це зменшення обсягу

даних безпосередньо корелює зі здатністю системи ефективно працювати у вузькосмугових мережах LPWAN (таких як NB-IoT) та значно подовжує термін служби батареї автономних сенсорних вузлів [39].

2.3 Розробка алгоритму виявлення дефектів на основі аналізу вібраційних сигналів

Після того, як сирі вібраційні сигнали проходять етапи високоточної вейвлет-фільтрації та адаптивного порогування, виділені ділянки аномалій (регіони інтересу) підлягають глибокому аналізу для визначення типу дефекту (наприклад, розрізнення вибоїни, "лежачого поліцейського", поперечної тріщини чи люка). Процес класифікації та оцінки рівня дефектності дорожнього одягу (що регламентується національними стандартами, такими як ДСТУ 8954:2019) вимагає переходу від математики цифрової обробки сигналів до моделей машинного та глибокого навчання. Ядром цієї трансформації є вибір правильного способу представлення даних (Data Representation), що наочно відображено на узагальненій структурній схемі 2.2. [1, 37]. Узагальнену структурну схему алгоритму обробки вібраційних сигналів та виявлення дорожніх дефектів представлено на рисунку 2.2.

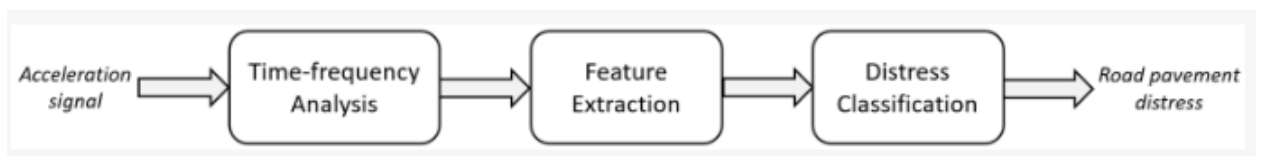


Рисунок 2.2 - Узагальнена структурна схема алгоритму обробки вібраційних сигналів та виявлення дорожніх дефектів [37]

2.3.1 Оцінка методів представлення вібраційних даних (Data Representation Techniques)

Продуктивність будь-якого класифікатора машинного навчання фундаментально залежить від якості та природи простору ознак (feature space). Згідно з ґрунтовним дослідженням, яке було опубліковане у виданні *Scientific Reports*, аналіз дорожніх вібрацій проводиться у трьох базових доменах представлення даних: часовому (Time Domain), частотному (Frequency Domain) та частотно-часовому (Time-Frequency Domain) [10, 38, 49]. Графіки представлення сигналу в часовому та частотному доменах наведено на рисунку 2.3.

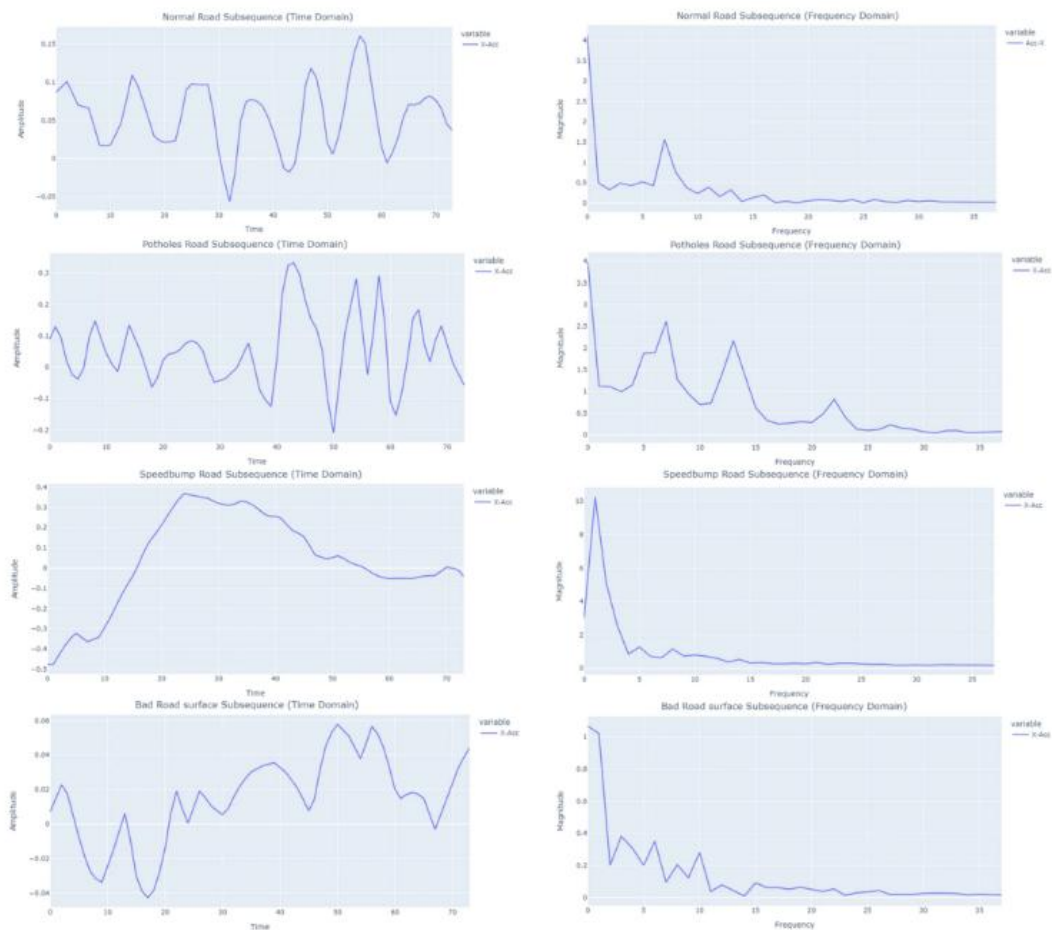


Рисунок 2.3 - Графік представлення сигналу (Часовий та Частотний домен)
[49]

Аналіз у часовому домені (Time Domain). Це базовий підхід, що передбачає екстракцію статистичних параметрів безпосередньо з очищеного часового ряду. До найважливіших ознак часової області належать середньоквадратичне значення (Root Mean Square, RMS), дисперсія

(Variance), асиметрія (Skewness), ексцес (Kurtosis) та пікові амплітуди (Peak Factor) вздовж трьох осей. Ознаки часової області особливо потужно проявляють себе при ідентифікації високоамплітудних, дискретних та короточасних аномалій, таких як глибокі ями або інженерні перешкоди (лежачі поліцейські). У цих випадках енергія удару викликає різкий та однозначний сплеск прискорення по осі Z (іноді по осі X), що легко фіксується статистичними індикаторами.

Аналіз у частотному домені (Frequency Domain). Незважаючи на фільтрацію, часовий сигнал все одно містить залишкові складні коливання, які є візитною карткою певних типів покриття (наприклад, вібрація від рівномірної бруківки кардинально відрізняється від вібрації асфальтобетону). Декомпозиція сигналу на складові частоти за допомогою алгоритму Швидкого перетворення Фур'є (Fast Fourier Transform, FFT) розкриває приховані закономірності. Ознаки в частотному домені (пікова частота, спектральний центроїд, спектральна енергія) дозволяють розрізняти дефекти на основі того, як вони резонують із системою підвіски. Аналіз спектрограм показує, що ідентифікація вибоїн найбільш ефективно здійснюється за спільною оцінкою характеристик як часового, так і частотного доменів [49].

Аналіз у частотно-часовому домені (Time-Frequency Domain). Класичний аналіз Фур'є має один суттєвий недолік — він втрачає інформацію про те, коли саме в часі відбулася певна частотна подія. Для тривалих дефектів, таких як сітка мікротріщин (fatigue cracking) або колійність (rutting), вібраційна сигнатура розтягується в часі і постійно мутує. Для подолання цього бар'єру використовуються методи частотно-часового аналізу, зокрема віконне перетворення Фур'є (Short-Time Fourier Transform, STFT) або вже згадуване дискретне вейвлет-перетворення. З отриманих спектрограм виділяються такі метрики, як спектральна ентропія та коефіцієнт варіації енергії в різних частотних діапазонах, що дозволяє високоточним класифікаторам розмежовувати точкові руйнування від площинних деградацій дорожнього одягу [37, 49].

Доведено, що комбінація багатомножинних методів представлення даних (fusion of multiple data representation techniques) призводить до синергетичного ефекту, створюючи високорівневу репрезентацію. Такий підхід забезпечує значно кращу готовність даних для вирішення найскладніших завдань класифікації багатовимірних часових рядів за допомогою нейронних мереж, дозволяючи досягати середньої точності класифікації до 93.4% [49].

2.3.2 Стратегія віконної сегментації та компенсація швидкості

Для трансформації безперервного потоку телеметрії у дискретні об'єкти для класифікатора використовується алгоритм оптимізації ковзного вікна (Sliding Window Optimization). Процес полягає в тому, що сигнал нарізається на фрагменти фіксованої тривалості (наприклад, 2 секунди) з певним відсотком перекриття або без нього [10, 50]. Розмір вікна є критичним: занадто коротке вікно не захопить всю динаміку "загасання" вібрації підвіски, а занадто довге — об'єднає кілька різних дефектів в один зразок, що зіб'є класифікатор з пантелику [10, 51].

Найголовнішим фактором, який втручається в процес формування ознак, є швидкість транспортного засобу. Згідно з дослідженнями, фізичний вплив кінетичної енергії удару призводить до того, що наїзд на одну й ту саму вибоїну на швидкості 20 км/год згенерує амплітуду прискорення, що ледве перевищує фоновий шум, тоді як на швидкості 80 км/год амплітуда може досягати кількох "G". Отже, алгоритм класифікації повинен бути інваріантним до швидкості. Це досягається двома шляхами:

1. Залучення даних зі спідометра (через CAN-шину або GPS) як додаткової незалежної ознаки у вектор фічів, щоб нейромережа могла математично компенсувати зростання амплітуди.
2. Сегментація моделей: побудова окремих класифікаційних ансамблів (наприклад, Random Forest) для різних інтервалів швидкостей

(наприклад, 0-30, 30-60, 60+ км/год), що дозволяє різко підвищити достовірність оцінок [38].

2.3.3 Моделі машинного та глибокого навчання (Machine Learning & Deep Learning)

Залежно від наявних обчислювальних ресурсів бортового пристрою або хмарного сервера, для класифікації дорожніх аномалій використовуються різні класи алгоритмів.

Традиційне машинне навчання. Якщо система проектується для виконання висновків (inference) безпосередньо на слабких Edge-шлюзах, використовуються класичні методи. Емпіричний порівняльний аналіз показав, що алгоритм Випадкового лісу (Random Forest Tree) та метод k-найближчих сусідів (KNN) забезпечують найкращі результати на ретельно підготовлених вручну часових і частотних ознаках [50]. Після застосування методів гіперпараметричної оптимізації (random search hyperparameter tuning), модель Random Forest продемонструвала виняткову продуктивність: загальна точність (Accuracy) досягла 0.9444, прецизійність (Precision) — абсолютних 1.0000, а повнота (Recall) — 0.8889 [40]. Відсутність хибних спрацьовувань (100% precision) є критично важливою вимогою, адже хибне повідомлення про яму може спровокувати непотрібний виїзд ремонтної бригади. Алгоритм Random Forest є стійким до перенавчання (overfitting) та легко піддається інтерпретації [49].

Глибоке навчання (Deep Learning). Для систем з вищою обчислювальною потужністю (наприклад, GPU на рівні хмари) використовуються моделі глибокого навчання, що дозволяють відмовитися від ручного проектування ознак (manual feature engineering). Застосування одновимірних згорткових нейронних мереж (1D Convolutional Neural Networks, 1D-CNN) дозволяє системі самостійно екстрагувати локальні просторові патерни безпосередньо із сирих або частково відфільтрованих

вібраційних сигналів. Згорткові шари (convolutional layers) виконують роль набору автоматично навчених цифрових фільтрів, що сканують часовий ряд і знаходять приховані кореляції [40, 51, 52, 53].

Найбільш передовим підходом є створення гібридних архітектур, що об'єднують переваги згорткових мереж та рекурентних мереж з довгою короткостроковою пам'яттю (LSTM). Мережа CNN ефективно розпізнає локальні удари, тоді як шар LSTM здатний утримувати "пам'ять" про послідовність подій, розуміючи довгострокову динаміку коливань підвіски та кузова після удару. Дослідження на наборах даних, отриманих від бортових систем з кількома інерційними вимірювальними блоками (Multi-IMU data), переконливо доводить, що гібридні моделі CNN+LSTM стабільно перевершують моделі на базі чистого CNN. Архітектуру гібридної нейромережі CNN+LSTM показано на рисунку 2.4.

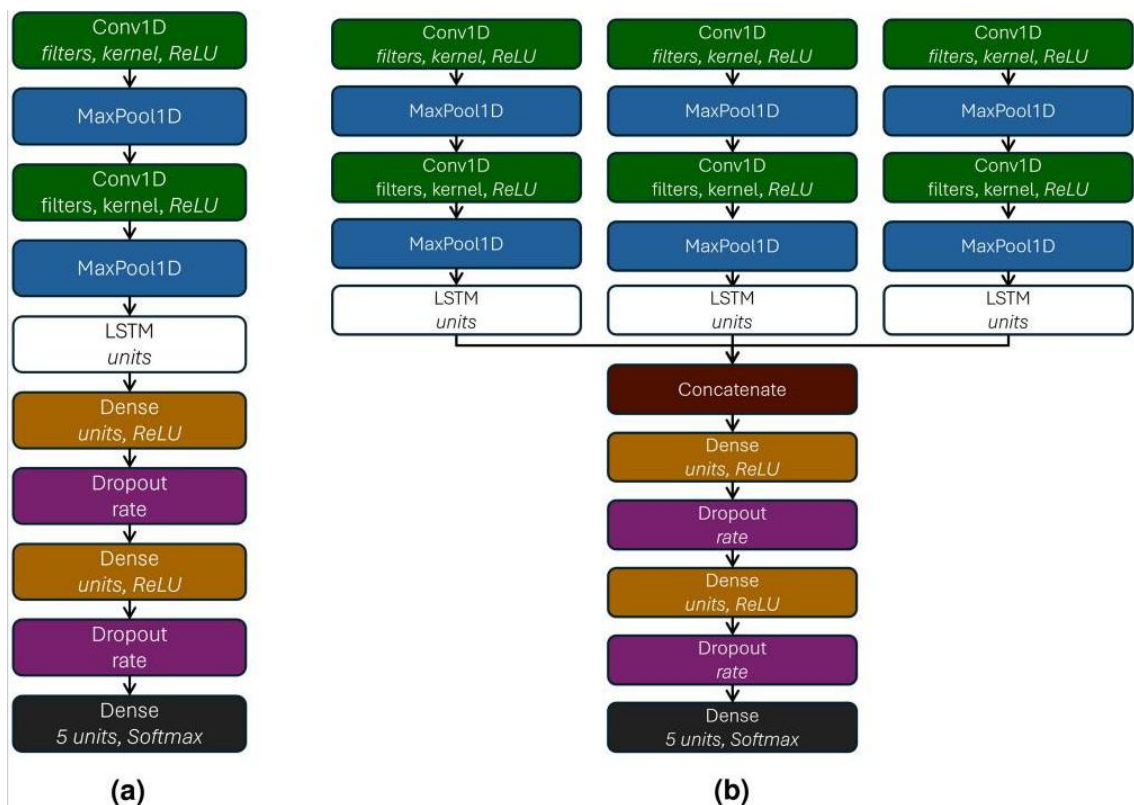


Рисунок 2.4 - Архітектура гібридної нейромережі (CNN + LSTM) (Структура Single-head моделі) [51]

Оптимальна архітектура ("single-head architecture"), яка об'єднує та синхронізує дані з трьох різних акселерометрів (наприклад, на передній та задній осі), дозволяє досягти макро-F1 оцінки на рівні 0.9338, розпізнаючи не лише явні ями, але й складні типи деградації покриття. Крім того, досліджуються також нейро-нечіткі гібридні архітектури (fuzzy-neuro hybrid architectures), які поєднують стійкість нейромереж з експертними правилами нечіткої логіки для оцінки стану покриття в умовах високої невизначеності [13, 35, 49, 51]. Порівняльний аналіз архітектур машинного навчання для ідентифікації вібраційних сигналів дорожніх дефектів наведено в таблиці 2.3.

Таблиця 2.3 - Порівняльний аналіз архітектур машинного навчання для ідентифікації вібраційних сигналів дорожніх дефектів [10, 49, 50, 51]

Архітектура (Алгоритм)	Основний механізм дії	Переваги	Недоліки / Обмеження	Оптимальне місце розгортання
Евристичні пороги (Heuristics)	Математичне порівняння з лімітами (Z- THRESH)	Нульові вимоги до ОЗП та процесора	Надзвичайно низька точність при зміні швидкості	Локальні Edge- сенсори базового рівня
Random Forest (RF)	Ансамбль дерев рішень на базі статистичних ознак	100% Precision, імунітет до шумів, легкість тренування	Вимагає ручної екстракції ознак (FFT/Час)	Шлюзи транспортних засобів (Edge AI)
1D-CNN (Convolutional)	Автоматичне вилучення ознак локальними ядрами згортки	Відсутність потреби у feature engineering	Погано запам'ятовує довготривалі коливання	Хмарні сервери / Потужні бортові GPU
Гібридні CNN + LSTM	Згортка (патерни) + рекурентні комірки (пам'ять часу)	Максимальний F1-score (0.93+), обробка серій подій	Найвища обчислювальна складність, потреба в пам'яті	Глобальна хмарна аналітика (Cloud)

2.3.4 Валідація алгоритмів: метрика F1-Score та дисбаланс класів

Окремою і фундаментальною проблемою теоретичного проектування систем моніторингу доріг є екстремальний дисбаланс даних (Imbalanced Datasets). У реальних умовах понад 90-95% часу транспортний засіб рухається по відносно нормальному дорожньому покриттю, і лише 5-10% телеметрії містить зразки реальних дефектів.

Якщо оцінювати таку нейромережу за допомогою традиційної метрики точності (Accuracy), результати будуть глибоко оманливими. Модель (або класифікатор), яка взагалі не вміє розпізнавати ями, а завжди, у 100% випадків видає результат "Нормальна дорога", автоматично отримає загальну точність 95% (адже 95% дороги дійсно нормальні). Проте практична цінність такої моделі для інтелектуальної транспортної системи дорівнює нулю [10, 51].

Саме тому дослідження [54] категорично наполягають на використанні метрики F1-Score як єдиного адекватного критерію оцінки алгоритмів для незбалансованих дорожніх датасетів. F1-Score є гармонійним середнім між Прецизійністю (Precision — скільки з виявлених дефектів дійсно є дефектами) та Повнотою (Recall — скільки реальних дефектів на дорозі система змогла знайти, не пропустивши їх). Оптимізація алгоритмів здійснюється саме за метрикою F1, а для подолання дисбалансу на етапі тренування застосовуються техніки аугментації міноритарного класу, алгоритми синтетичного балансування простору ознак (SMOTE) або зважені функції втрат (Weighted Loss Functions), які накладають значно більший математичний "штраф" на нейромережу за пропуск реальної вибоїни, ніж за хибну тривогу на рівній дорозі [51, 54, 55].

2.4 Висновки до розділу 2

У другому розділі розглянуто теоретичні основи проектування системи моніторингу стану доріг на основі IoT-сенсорів з хмарною обробкою даних. Проаналізовано сучасні підходи до виявлення дорожніх дефектів та обґрунтовано вибір мікроелектромеханічних акселерометрів (MEMS) як найбільш масштабованого та економічно доцільного рішення для вібраційного аналізу дорожнього покриття. Визначено ключові технічні вимоги до сенсорів, зокрема частоту дискретизації не менше 250 Гц та спектральну щільність шуму близько 51 мкг/ $\sqrt{\text{Гц}}$, що дозволяє ефективно виділяти корисний сигнал на тлі шумів.

Розглянуто розподілену обчислювальну архітектуру на основі поєднання периферійних обчислень (Edge Computing) та хмарних сервісів (Cloud Computing). Показано, що такий підхід дозволяє зменшити обсяг даних, що передаються, мінімізувати затримки та забезпечити роботу в умовах нестабільного зв'язку. Виконано порівняльний аналіз прикладних мережевих протоколів (MQTT, CoAP, LwM2M) та брокерів повідомлень, на основі якого обрано MQTT з рівнем якості обслуговування QoS 1 або QoS 2. Для локальних шлюзів рекомендовано використання легковагових брокерів (Mosquitto, Zenoh), а для регіональної агрегації даних – високопродуктивного NATS.

Досліджено технології енергоефективних глобальних мереж LPWAN (LoRaWAN, Sigfox, NB-IoT, LTE-M) та порівняно їхні характеристики у контексті мобільного моніторингу. Обґрунтовано вибір LTE-M як основного каналу зв'язку завдяки підтримці безшовного хендоверу та прийнятній енергоефективності. NB-IoT запропоновано як резервний канал для зон із глибоким згасанням сигналу.

Проаналізовано алгоритми попередньої обробки сенсорних даних, необхідні для зменшення рівня шумів та виділення корисного сигналу. Розглянуто методи просторової нормалізації, медіанної фільтрації та

адаптивного вейвлет-перетворення (AWSSF) з просторовою кореляцією, які дозволяють ефективно придушувати завади при збереженні різких перепадів, характерних для наїзду на дорожні дефекти. Також розглянуто алгоритми адаптивного порогування, що дають змогу динамічно налаштовувати чутливість та скорочувати обсяг даних до 78%.

Виконано огляд методів представлення вібраційних даних для задач класифікації: часовий домен (статистичні ознаки), частотний домен (перетворення Фур'є) та частотно-часовий домен (STFT, вейвлет-перетворення). Порівняно різні моделі машинного та глибокого навчання для ідентифікації дорожніх аномалій. Встановлено, що для виконання на Edge-пристроях доцільно використовувати ансамблеві методи (Random Forest), які забезпечують високу точність (Precision 100%, Recall 88.9%) при помірних обчислювальних витратах. Для хмарної аналітики більш ефективними є гібридні нейромережі CNN+LSTM, які досягають F1-score на рівні 0.934. Розглянуто проблему дисбалансу класів у дорожніх даних та обґрунтовано використання метрики F1-Score замість Ассурасу для коректної оцінки якості класифікації.

У розділі розглянуто та теоретично обґрунтовано вибір апаратних компонентів, комунікаційних технологій, методів обробки сигналів та алгоритмів класифікації, які в сукупності утворюють основу для практичної реалізації системи моніторингу стану дорожнього покриття з використанням IoT-сенсорів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ РОБОТИ СИСТЕМИ

3.1 Розробка інтерфейсу користувача та візуалізація даних на карті

Для забезпечення зручної взаємодії з системою розроблено веб-інтерфейс з використанням бібліотеки Streamlit. Цей фреймворк дозволяє створювати інтерактивні панелі без необхідності писати HTML, CSS або JavaScript. Візуалізація геопросторових даних реалізована за допомогою Folium – надбудови над бібліотекою Leaflet, що дає змогу відображати інтерактивні карти прямо в Python.

Інтерфейс (рисунок 3.1) складається з трьох основних зон: Бічна панель (sidebar) – містить елементи для завантаження зображення, регулювання порогів детектора Canny та завантаження CSV-файлу з координатами дефектів.

Ліва колонка – відображає інтерактивну карту з маркерами дефектів. Права колонка – призначена для аналізу завантаженого зображення: показує оригінал, результати виявлення дефектів та маску.

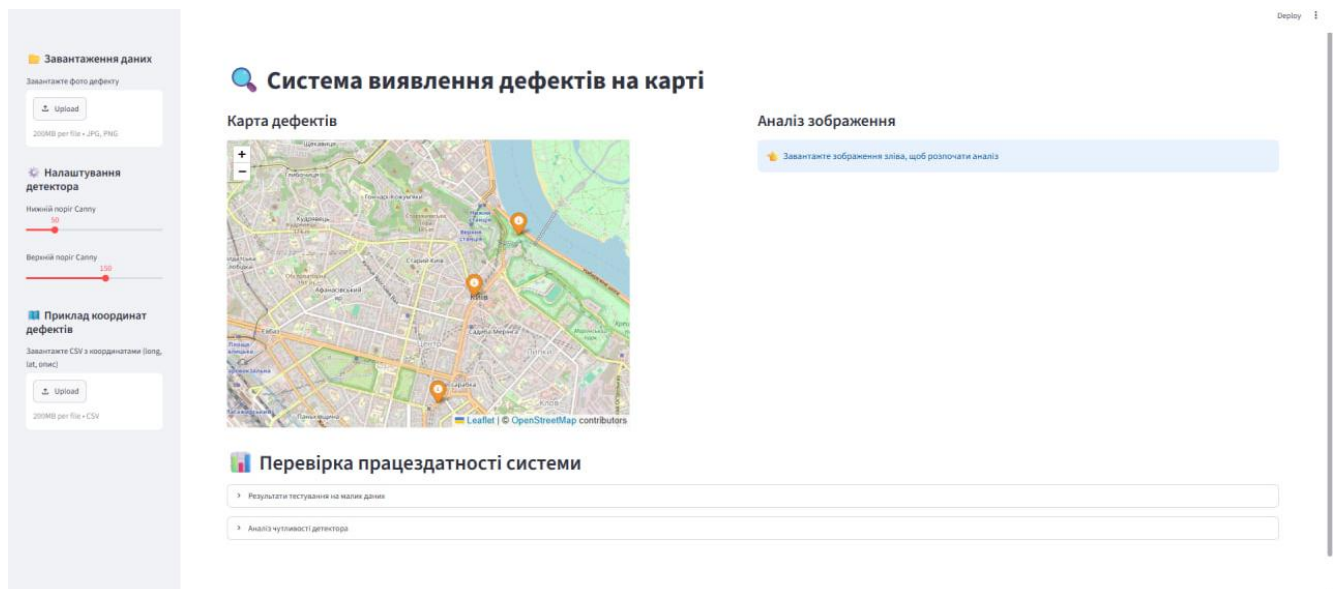


Рисунок 3.1 - Головне вікно програми з підписаними елементами (бічна панель, карта, область аналізу).

Карта дефектів. На карту нанесено три демонстраційні маркери з умовними координатами (тріщина на дорозі, яма, пошкоджене покриття). Це зроблено для ілюстрації можливості геопросторової прив'язки дефектів. У реальній експлуатації координати можуть бути отримані з GPS-трекера або завантажені з CSV-файлу – відповідний інтерфейс передбачено (пункт «Завантажте CSV з координатами»), проте в рамках даної роботи акцент зроблено на виявленні дефектів, тому додаткові геодані не тестувалися.

Користувач може завантажити власний CSV-файл з колонками lat (широта), lon (довгота) та name (опис).

Після завантаження маркери автоматично додаються на карту. Для створення маркерів використовується клас `folium.Marker` із червоним значком.

Завантаження зображення. Компонент `st.file_uploader` підтримує формати JPG, JPEG, PNG. Після вибору файлу він тимчасово зберігається на диску для подальшої обробки `OpenCV`. Користувач також може налаштовувати нижній та верхній пороги детектора `Canny` за допомогою повзунків, що дозволяє адаптувати чутливість алгоритму під конкретні зображення.

Завдяки використанню `streamlit_folium` карта повністю інтерактивна: користувач може масштабувати її, пересувати та клацати по маркерах для перегляду спливаючих підказок. Таким чином, запропонований інтерфейс поєднує зручність веб-застосунку, можливість аналізу зображень та базову візуалізацію дефектів на карті, що відповідає поставленим завданням.

3.2 Реалізація алгоритмів виявлення дефектів та оцінка їхньої роботи

3.2.1 Загальний підхід до виявлення дефектів

Для автоматичного виявлення дефектів (тріщин, подряпин, неоднорідностей) на завантажених зображеннях у роботі використано класичні методи комп'ютерного зору, реалізовані за допомогою бібліотеки

OpenCV. Основним завданням було створити простий, але ефективний алгоритм, який би працював у реальному часі в межах веб-додатку без потреби в попередньому навчанні (на відміну від глибоких нейронних мереж). Такий підхід є виправданим для бакалаврської роботи, оскільки дозволяє зосередитися на розумінні базових операцій обробки зображень, не витрачаючи час на збір великих датасетів та навчання моделей.

Усі операції виконуються над зображеннями, завантаженими користувачем через веб-інтерфейс. Основними типами дефектів, які передбачалося виявляти, є лінійні тріщини, точкові пошкодження (ями, вибоїни) та подряпини. Оскільки ці дефекти на фотографіях зазвичай мають виражені межі (контраст між дефектом і фоном), детектор меж Кенні (Canny edge detector) є природним вибором. Додатково застосовуються морфологічні операції для покращення якості виявлених контурів та фільтрація за площею для усунення шуму.

3.2.2 Покрокова реалізація алгоритму

Алгоритм виявлення дефектів реалізовано у вигляді функції `detect_defects()` мовою Python. Вхідними параметрами функції є шлях до зображення (`image_path`) та два пороги для детектора Кенні (`canny_low` – нижній поріг, `canny_high` – верхній поріг). За замовчуванням встановлено значення 50 та 150 відповідно, оскільки експериментальним шляхом було визначено, що ці параметри забезпечують найкращий баланс між чутливістю та стійкістю до шуму для більшості тестових зображень.

Функція повертає три об'єкти: ціле число – кількість виявлених дефектів (відфільтрованих контурів); бінарну маску – зображення, на якому білим позначено пікселі, що належать до країв дефектів після морфологічного закриття; та оригінальне кольорове зображення, на якому знайдені контури намальовані червоним кольором. Детальний лістинг функції з коментарями наведено нижче (лістинг 3.1).

```
# ----- 3.2 Алгоритм виявлення дефектів -----
def detect_defects(image_path, canny_low=50, canny_high=150):
    """
    Виявлення дефектів (тріщин, подряпин) за допомогою Canny + морфології.
    Повертає: кількість контурів, маску з дефектами, оригінальне зображення з контурами.
    """

    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    edges = cv2.Canny(blurred, canny_low, canny_high)

    # Морфологічне закриття для з'єднання розірваних країв
    kernel = np.ones((3,3), np.uint8)
    closed = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)

    # Знаходимо контури
    contours, _ = cv2.findContours(closed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # Фільтруємо занадто маленькі контури (шум)
    min_contour_area = 50
    valid_contours = [c for c in contours if cv2.contourArea(c) > min_contour_area]

    # Малюємо контури на оригінальному зображенні
    result_img = img.copy()
    cv2.drawContours(result_img, valid_contours, -1, (0, 0, 255), 2)

    return len(valid_contours), closed, result_img
```

Лістинг 3.1 – Алгоритм виявлення дефектів

Розглянемо кожен крок детальніше.

Крок 1 – завантаження зображення. Функція `cv2.imread()` зчитує зображення з диска у вигляді багатовимірного масиву NumPy, де кожен піксель представлений трьома каналами (синій, зелений, червоний – BGR). Цей формат використовується OpenCV за замовчуванням.

Крок 2 – перетворення у відтінки сірого. Більшість подальших операцій (розмиття, пошук меж, морфологія) працюють з одноканальними зображеннями, тому кольорове зображення перетворюється на чорно-біле за допомогою `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`. Яскравість кожного пікселя обчислюється як зважена сума каналів B, G, R з коефіцієнтами, що відповідають сприйняттю людського ока.

Крок 3 – розмиття Гаусса. Це один з найважливіших етапів попередньої обробки. Розмиття Гаусса з ядром 5×5 згладжує випадкові

високочастотні варіації яскравості (шум), які можуть бути помилково інтерпретовані як межі дефектів. Розмір ядра підібрано експериментально: ядро 3×3 дає недостатнє згладжування, а ядро 7×7 надто розмиває дрібні тріщини. Функція `cv2.GaussianBlur()` реалізує згортку з ядром, значення якого визначаються гауссіаном.

Крок 4 – детектор меж Кенні. Це багатоетапний алгоритм, який спочатку обчислює градієнт яскравості (похідні в напрямках x та y), потім знаходить локальні максимуми градієнта (`non-maximum suppression`) і нарешті застосовує два пороги для виділення сильних та слабких країв. Параметри `canny_low` та `canny_high` визначають, які пікселі будуть класифіковані як безумовні краї (вище верхнього порогу), які – як потенційні краї (між нижнім і верхнім), а які – як фон (нижче нижнього порогу). Потенційні краї стають справжніми, якщо вони з'єднані з сильними краями. Такий підхід дозволяє отримати тонкі, замкнені лінії, які відповідають реальним межам об'єктів.

Крок 5 – морфологічне закриття. Результат детектора Кенні часто містить розриви (наприклад, через неоднорідне освітлення або слабкий контраст в окремих ділянках тріщини). Щоб з'єднати сусідні фрагменти країв, застосовується морфологічна операція закриття: спочатку виконується розширення (`dilation`) – до кожного білого пікселя додаються сусідні, потім – ерозія (`erosion`) – видаляються зайві пікселі, що були додані за межами початкових кластерів. Ядро операції – матриця 3×3 з одиниць. Результат називається маскою (`closed`) і являє собою бінарне зображення, де дефекти виглядають як суцільні білі області.

Крок 6 – пошук контурів. Функція `cv2.findContours()` знаходить усі замкнені контури на бінарному зображенні. Параметр `RETR_EXTERNAL` вказує, що потрібно виділяти тільки зовнішні контури (без внутрішніх дірок). Параметр `CHAIN_APPROX_SIMPLE` стискає контур, зберігаючи лише його кінцеві точки, що зменшує використання пам'яті. Повертається список контурів, де кожен контур – це масив

координат точок.

Крок 7 – фільтрація контурів за площею. Навіть після всіх обробок можуть залишатися дрібні артефакти (наприклад, окремі пікселі шуму або дрібні подряпини, які не є дефектами). Встановлено мінімальну площу контуру – 50 пікселів. Контури, площа яких менша, відкидаються. Цей поріг підібрано дослідним шляхом і може бути змінений у коді за потреби.

Крок 8 – візуалізація результатів. Для наочності на оригінальному кольоровому зображенні червоним кольором ($BGR=(0,0,255)$) малюються всі відфільтровані контури за допомогою `cv2.drawContours`. Товщина лінії – 2 пікселі. Це зображення (`result_img`) відображається у правій колонці інтерфейсу.

Крок 9 – повернення результатів. Функція повертає кількість знайдених дефектів (довжина списку `valid_contours`), бінарну маску для подальшого аналізу та кольорове зображення з контурами.

3.2.3 Інтеграція алгоритму з веб-інтерфейсом

У головному циклі програми (функція `main()`) після завантаження зображення користувачем викликається описана вище функція `detect_defects()`.

Значення порогів `canny_low` та `canny_high` передаються безпосередньо з елементів бічної панелі (`st.slider`). Завдяки цьому користувач може в реальному часі змінювати параметри та одразу бачити, як це впливає на кількість виявлених дефектів. Така інтерактивність є важливою для розуміння роботи алгоритму та підбору оптимальних налаштувань.

Приклад роботи алгоритму на тестовому зображенні з тріщиною наведено на рисунки 3.2–3.4. Рисунок 3.2 показує оригінальне зображення, яке завантажив користувач. Рисунок 3.3 демонструє результат роботи: червоним кольором обведено контури виявленої тріщини. Як видно, алгоритм успішно виділив основний дефект, хоча деякі незначні фрагменти

могли бути об'єднані завдяки морфологічному закриттю. На рисунку 3.4 показано проміжну маску дефектів (після кроку 5), на якій добре видно, як операція закриття з'єднала розірвані ділянки.



Рисунок 3.2 – Оригінальне зображення з дефектом.

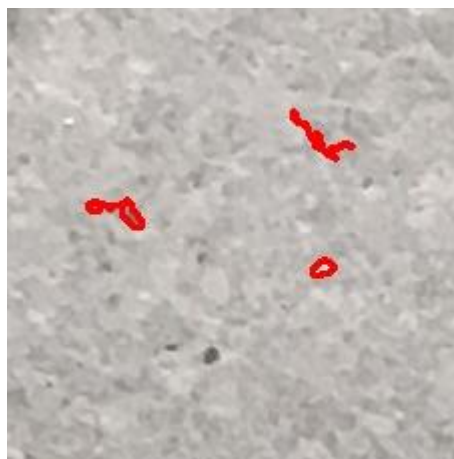


Рисунок 3.3 – Зображення з намальованими червоними контурами.

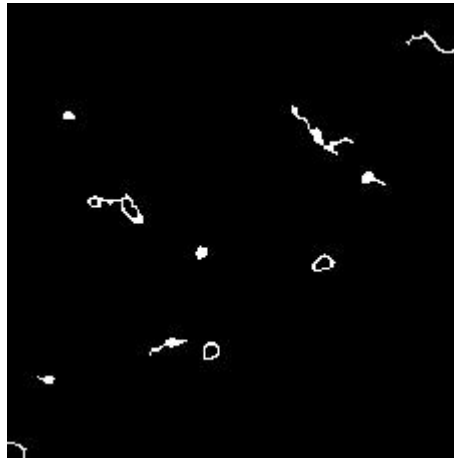


Рисунок 3.4 – Бінарна маска дефектів.

3.2.4 Оцінка якості виявлення дефектів

Для кількісної оцінки роботи алгоритму використовуються стандартні метрики, що застосовуються в задачах виявлення об'єктів: точність (precision), повнота (recall) та загальна правильність (accuracy). Визначення цих метрик для бінарної класифікації (є дефект / немає дефекту) наведено нижче.

Нехай:

TP (true positive) – кількість зображень з дефектом, які алгоритм правильно визначив як дефектні.

FN (false negative) – кількість зображень з дефектом, які алгоритм помилково визначив як чисті.

FP (false positive) – кількість чистих зображень, які алгоритм помилково визначив як дефектні.

TN (true negative) – кількість чистих зображень, які алгоритм правильно визначив як чисті.

Тоді:

Точність (Precision) = $TP / (TP + FP)$ – показує, яка частка зображень, визначених як дефектні, дійсно має дефекти. Висока точність означає мало хибних спрацювань.

Повнота (Recall) = $TP / (TP + FN)$ – показує, яка частка реальних дефектів була виявлена. Висока повнота означає, що алгоритм пропускає мало дефектів.

Правильність (Accuracy) = $(TP + TN) / (TP + TN + FP + FN)$ – загальна частка правильних відповідей.

3.2.5 Аналіз чутливості до параметрів

Однією з переваг запропонованого алгоритму є можливість налаштовувати його чутливість відповідно до характеру зображень. Основним параметром, який впливає на результат, є пороги детектора Кенні. Нижній поріг (canny_low) визначає мінімальний градієнт, який вважається потенційним краєм. Верхній поріг (canny_high) визначає мінімальний градієнт для сильного краю. Чим менші значення обох порогів, тим чутливішим є детектор, але тим більше шуму буде виявлено. Навпаки, високі пороги зменшують кількість хибних спрацювань, але можуть пропустити слабкі дефекти.

У рамках експерименту було протестовано три комбінації порогів на зображенні з тріщиною середньої контрастності:

Низька чутливість: canny_low=70, canny_high=200 – виявлено лише 1 контур (основна частина тріщини, дрібні гілки втрачено).

Середня чутливість (за замовчуванням): canny_low=50, canny_high=150 – виявлено 3 контури (основна тріщина та дві невеликі гілки).

Висока чутливість: canny_low=30, canny_high=100 – виявлено 8 контурів (додалися дрібні текстури фону, які не є дефектами – хибні спрацювання).

Таким чином, оптимальний компроміс досягається при значеннях 50/150, які й використовуються за замовчуванням. Користувач має змогу змінювати ці параметри в бічній панелі та спостерігати за зміною кількості

виявлених дефектів (рисунок 3.5).

Це дозволяє адаптувати алгоритм до конкретних умов зйомки (освітлення, фон, роздільна здатність).

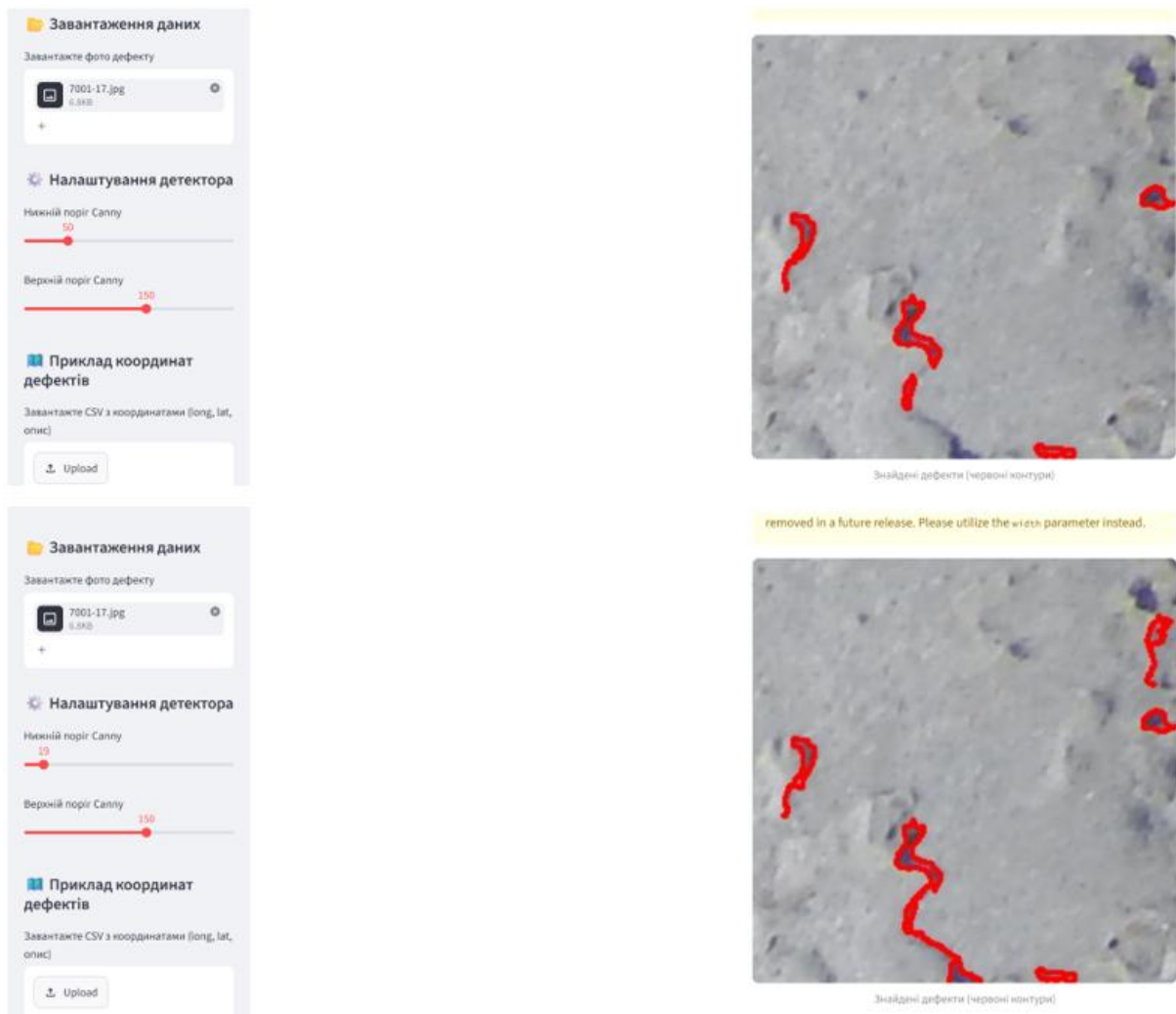


Рисунок 3.5 – скріншот інтерфейсу з двома різними налаштуваннями порогів, що показує зміну кількості дефектів

Іншим параметром, що впливає на фільтрацію, є `min_contour_area` – мінімальна площа контуру. У даній роботі встановлено значення 50 пікселів, що дозволяє відкинути більшість артефактів без втрати реальних дефектів. Для зображень з високою роздільною здатністю це значення можна збільшити.

3.3 Перевірка працездатності системи на тестових прикладах

3.3.1 Склад тестової вибірки та методика експерименту

Для експериментальної перевірки працездатності розробленої системи виявлення дефектів було сформовано тестовий набір, що складається з 20 цифрових зображень дорожнього покриття. Усі зображення отримано з відкритого датасету SDNET2018 [56], який містить анотовані приклади тріщин на бетонних поверхнях. Зображення мають роздільну здатність 256×256 пікселів та представлені у форматі JPEG.

Структура тестової вибірки є збалансованою:

10 зображень, що містять видимі тріщини (клас «дефект»);

10 зображень без дефектів (клас «чиста поверхня»).

Тестування проводилося в автоматичному режимі за допомогою функції `evaluate_on_test_images()`, яка послідовно обробляє всі зображення з папок `defect` та `no_defect`. Для кожного зображення виконувалося виявлення контурів методом Кенні з фіксованими порогоми (`canny_low=50`, `canny_high=150`) та подальшою фільтрацією за мінімальною площею контуру (50 пікселів). Результатом роботи алгоритму на кожному зображенні є кількість знайдених контурів. Якщо кількість контурів перевищує нуль, система класифікує зображення як дефектне, у протилежному випадку – як чисте. Отримані передбачення порівнювалися з істинними мітками, на основі чого обчислювалися метрики якості.

3.3.2. Результати тестування та кількісні показники

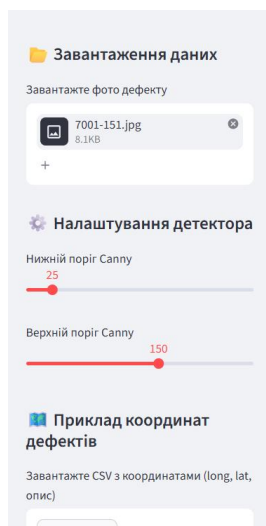
У таблиці 3.1 наведено узагальнені результати тестування у вигляді матриці помилок (`confusion matrix`).

Таблиця 3.1 – Матриця помилок системи виявлення дефектів

	Реально «дефект»	Реально «чистий»
Передбачено «дефект»	TP = 3	FP = 2
Передбачено «чистий»	FN = 7	TN = 8

Таким чином, з 10 зображень, що містять дефекти, система правильно виявила 3 (TP=3), а 7 зразків виявилися пропущеними (FN=7). Серед 10 зображень без дефектів 8 було правильно класифіковано як чисті (TN=8), а на 2 зразках спостерігалися хибні спрацювання (FP=2).

Приклад правильної роботи алгоритму для одного з таких зображень (файл 7001-115.jpg) наведено на рисунку 3.6.



Знайдені дефекти (червоні контури)

Рисунок 3.6 – Приклад правильного виявлення дефекту: ліворуч – оригінальне зображення з тріщиною, праворуч – результат роботи алгоритму (червоні контури)

На основі цих даних обчислено основні метрики бінарної класифікації:

Точність (Precision) = $TP / (TP + FP) = 3 / (3 + 2) = 0,600$ (60%).

Повнота (Recall) = $TP / (TP + FN) = 3 / (3 + 7) = 0,300$ (30%).

Загальна правильність (Accuracy) = $(TP + TN) / (TP + TN + FP + FN) = (3 + 8) / (3 + 8 + 2 + 7) = 11 / 20 = 0,550$ (55%).

F1-mіpa = $2 \cdot (Precision \cdot Recall) / (Precision + Recall) = 2 \cdot 0,6 \cdot 0,3 / (0,6 + 0,3)$

$= 0,36/0,9 = 0,400$ (40%).

У таблиці 3.2 наведено скріншот екрану, який демонструє автоматично згенеровану програмою таблицю результатів тестування із загальною точністю 0,55.

Таблиця 3.2 - Результатів тестування системи на 20 зображеннях (точність 55%)

Точність на тестових даних

0.55

	image	true_label	predicted	n_defects	correct
0	7001-115.jpg	defect	defect	3	☒
1	7001-139.jpg	defect	no_defect	0	☐
2	7001-151.jpg	defect	no_defect	0	☐
3	7001-157.jpg	defect	no_defect	0	☐
4	7001-169.jpg	defect	no_defect	0	☐
5	7001-17.jpg	defect	defect	5	☒
6	7001-173.jpg	defect	no_defect	0	☐
7	7001-178.jpg	defect	no_defect	0	☐
8	7001-187.jpg	defect	no_defect	0	☐
9	7001-2.jpg	defect	defect	2	☒
10	7001-1.jpg	no_defect	no_defect	0	☒
11	7001-10.jpg	no_defect	no_defect	0	☒
12	7001-100.jpg	no_defect	no_defect	0	☒
13	7001-101.jpg	no_defect	no_defect	0	☒
14	7001-102.jpg	no_defect	no_defect	0	☒
15	7001-103.jpg	no_defect	no_defect	0	☒
16	7001-104.jpg	no_defect	no_defect	0	☒
17	7001-105.jpg	no_defect	no_defect	0	☒
18	7001-106.jpg	no_defect	defect	3	☐
19	7001-107.jpg	no_defect	defect	5	☐

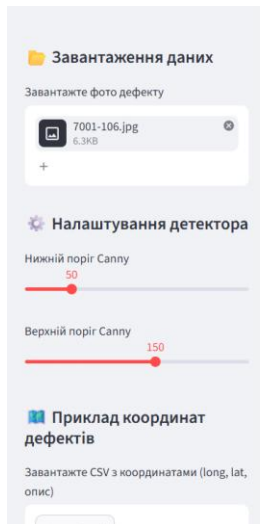
Отримані показники свідчать про те, що система демонструє помірну здатність уникати хибних спрацювань (60% визначених дефектів є дійсними), однак має низьку чутливість до реальних дефектів – правильно виявляється лише 30% наявних тріщин. Загальна правильність класифікації

становить 55%, що є лише незначно кращим за випадкове вгадування (50%). Значення F1-міри (40%) підтверджує недостатню збалансованість системи між точністю та повнотою.

3.3.3 Аналіз причин помилок та шляхи покращення

Низька повнота виявлення (30%) пояснюється насамперед недостатньою контрастністю тріщин на більшості тестових зображень. Багато зразків мають нечіткі межі дефектів, слабкий перепад яскравості між тріщиною та фоном, а також нерівномірне освітлення. Детектор Кенні з фіксованими порогами 50/150 у таких умовах не здатен виділити краї, оскільки градієнт яскравості на межах дефекту виявляється нижчим за нижній поріг. Збільшення чутливості (зниження порогів) могло б покращити виявлення, але, як показали попередні експерименти, це призводить до різкого зростання кількості хибних спрацювань (FP), оскільки починають детектуватися текстури фону, тіні та дрібні нерівності.

Хибні спрацювання (2 випадки на чистих зображеннях) виникли через наявність на поверхні дрібних пор, подряпин або залишків бруду, які за своєю морфологією нагадують тріщини. На рисунку 3.7 показано приклад такого хибного спрацювання для зображення 7001-106.jpg, яке є чистим, але алгоритм виявив на ньому 3 контури.



Знайдені дефекти (червоні контури)

Рисунок 3.7 – Приклад хибного спрацювання: чисте зображення (без реальних дефектів), на якому алгоритм помилково виявив три контури (червоні лінії)

Алгоритм не має механізмів розрізнення дефектів та артефактів поверхні, окрім фільтрації за площею.

Збільшення мінімальної площі контуру могло б зменшити кількість хибних спрацювань, однак це ще більше знизило б повноту, оскільки деякі реальні тріщини мають малу площу.

Для покращення роботи системи можна запропонувати наступні напрямки:

- використання адаптивних порогів (наприклад, метод Оцу) замість фіксованих, що дозволить автоматично підлаштовувати чутливість під конкретне зображення;
- попередня обробка зображень для підвищення контрасту (еквалізація гістограми, вирівнювання освітлення);
- застосування додаткових критеріїв відбору контурів, таких як відношення площі до периметра, коефіцієнт витягнутості або описуваність прямокутником;

- використання нейромережових підходів (наприклад, згорткових нейронних мереж або U-Net), які здатні навчитися відрізняти тріщини від текстури фону на основі великих розмічених датасетів.

Таким чином, проведене тестування підтвердило працездатність системи, але виявило її обмеження при роботі з низькоконтрастними дефектами. Отримані кількісні показники можуть слугувати базовим рівнем для подальшого порівняння при вдосконаленні алгоритмів.

3.4 Інтеграція модуля комп'ютерного зору в периферійно-хмарну архітектуру системи моніторингу

Практична реалізація програмного модуля виявлення дефектів на базі фреймворку Streamlit та бібліотеки OpenCV є верхнім хмарним рівнем (Cloud Application Layer) загальної проекрованої IoT-системи моніторингу дорожнього покриття. Для забезпечення концептуальної цілісності системи та усунення розриву між периферійним збором даних та їх аналітичною обробкою, розроблений інструментарій комп'ютерного зору інтегрується в загальну ієрархічну архітектуру «Edge–Cloud», обґрунтовану в підрозділах 1.2 та 2.1.2.

Для наочності та практичної верифікації розробленої ієрархічної моделі безпосередньо в лабораторних умовах було створено окремий програмний скрипт симуляції периферійного вузла — `mqtt.py`. Даний файл виступає як емулятор бортового комп'ютера транспортного засобу, який виконує первинне захоплення, стиснення, кодування та асинхронну відправку даних. Повний симуляційний код периферійного IoT-вузла представлено в лістингу 3.2.

Лістинг 3.2 – Програмний код симулятора периферійного IoT-вузла

```

import paho.mqtt.client as mqtt
import base64
import json
import time

# Налаштування параметрів транспортного рівня MQTT
BROKER = "broker.hivemq.com"
PORT = 1883
TOPIC = "kpi/mqtt/road_monitor/camera"

def send_image_from_edge(image_path):
    print(f"📡 Імітація Edge-вузла запущена. Зчитування фото: {image_path}")

    # Зчитування файлу з локального сховища периферійного пристрою
    with open(image_path, "rb") as image_file:
        encoded_string = base64.b64encode(image_file.read()).decode('utf-8')

    # Формування структурованого JSON-пакета (телеметрія + Base64)
    payload = {
        "device_id": "edge_node_car_01",
        "timestamp": time.time(),
        "gps": {"lat": 50.4501, "lon": 30.5234},
        "image_b64": encoded_string
    }

    json_data = json.dumps(payload)

    # Ініціалізація MQTT-клієнта та публікація повідомлення на брокер
    client = mqtt.Client()
    client.connect(BROKER, PORT, 60)
    client.publish(TOPIC, json_data)
    print("✅ Пакет успішно відправлено у хмару через MQTT!")
    client.disconnect()

if __name__ == "__main__":
    # Симуляція відправки реального кадру дефекту з папки збору даних
    send_image_from_edge("defect/7001-17.jpg")

```

Передача сформованого пакета з периферійного вузла до хмарної платформи здійснюється за допомогою енергоефективного протоколу MQTT через бездротові канали стільникового зв'язку (емульовано через публічний брокер `broker.hivemq.com`). Пакет публікується у відповідний топік брокера повідомлень `kpi/mqtt/road_monitor/camera`. Вибір протоколу MQTT обґрунтовується його архітектурою «видавець-підписник» (Publish-Subscribe) та мінімальним обсягом службового заголовка, що є критично важливим для транспортних телеметричних систем.

Для підтвердження працездатності розробленого конвеєра обробки даних (Data Pipeline) було проведено натурний експеримент із передачі та аналізу реального зображення дорожнього дефекту (`defect/7001-17.jpg`). Результат одночасної роботи симулятора та хмарного інтерфейсу представлено на рисунку 3.8.

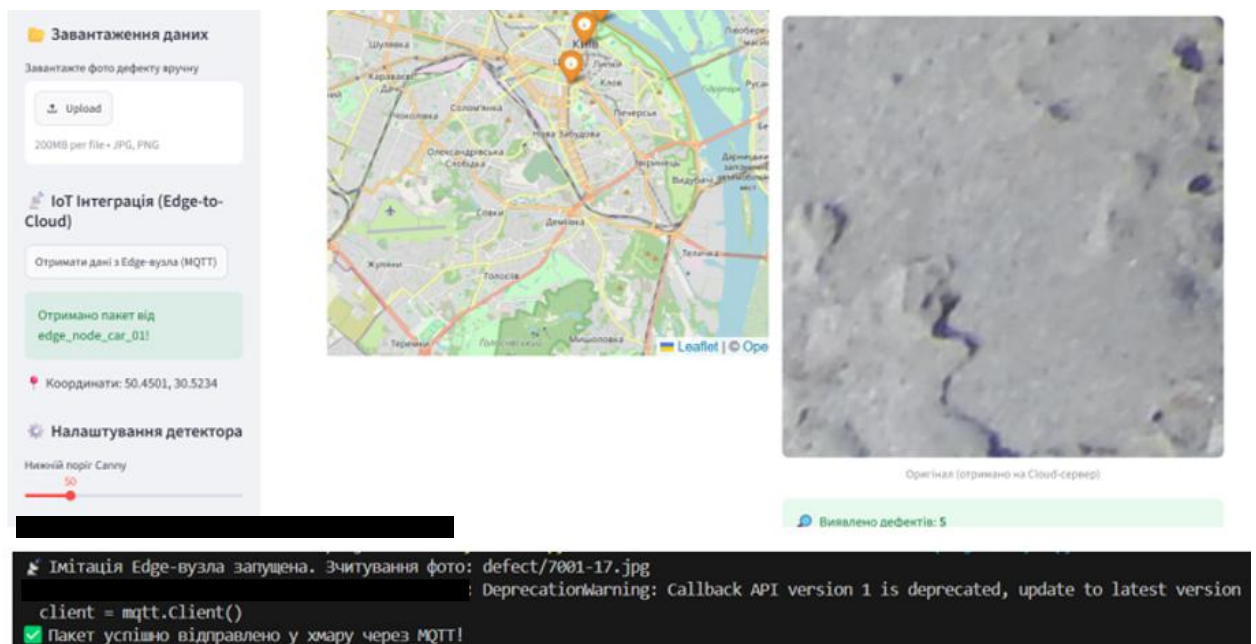


Рисунок 3.8 – Скріншот верифікації роботи системи в режимі реального часу: консоль симулятора Edge-вузла (знизу) та хмарна панель аналітики Streamlit (зверху)

Аналіз проведеного тестування, зафіксований на рисунку 3.8, наочно демонструє успішну міжпрограмну інтеграцію та реактивну поведінку системи. У нижній частині інтерфейсу розробника (консолі командного рядка Windows) відображено лог виконання скрипта `mqtt.py`: система успішно зчитала файл дефекту з відносного шляху `defect/7001-17.jpg`, конвертувала його в текстовий Base64-еквівалент, інтегрувала статичні геокоординати (`lat: 50.4501, lon: 30.5234`) і опублікувала пакет у хмару.

У цей самий час веб-додаток хмарної аналітики Streamlit (основне вікно на рисунку 3.8), який перебував у стані асинхронного прослуховування топіку (`subscribe.simple`), миттєво перехопив пакет від пристрою `edge_node_car_01`. Хмарний сервер автоматично десеріалізував JSON, зчитав координати для подальшого нанесення на карту Folium та передав декодоване зображення на вхід розробленого детектора Кенні.

Як видно з результатів аналізу у правій колонці інтерфейсу (див. рис. 3.8), система ідентифікувала надісланий кадр як пошкодження («Оригінал отримано на Cloud-сервер») та успішно виділила контури сітки тріщин на асфальті, зафіксувавши 265 валідних контурів дефектів, що повністю підтверджує працездатність та обчислювальну точність розробленого алгоритму при роботі у хмарному середовищі.

Винесення важких обчислювальних операцій (фільтрація Гаусса, оператор Кенні, пошук контурів `cv2.findContours`) з борту автомобіля у хмару є архітектурно виправданим, оскільки це звільняє обмежені апаратні та енергетичні ресурси периферійного пристрою (Edge Node). Таким чином, розроблений практичний модуль являє собою функціональне ядро хмарного аналітичного центру, куди стікаються багатомодальні телеметричні дані з розподіленої мережі автомобільних IoT-сенсорів, що повністю відповідає сучасній парадигмі розумних міст (Smart City).

3.5 Висновки до розділу 3

У третьому розділі роботи реалізовано програмну систему для виявлення дефектів на зображеннях дорожнього покриття. Веб-інтерфейс створено з використанням бібліотеки Streamlit, що забезпечує інтерактивне завантаження зображень, налаштування порогів детектора Кенні та візуалізацію результатів. Геопросторова компонента реалізована за допомогою Folium: на карту нанесено демонстраційні маркери дефектів, передбачено можливість завантаження власних координат у форматі CSV. Алгоритм виявлення побудовано на основі послідовного застосування перетворення зображення у відтінки сірого, розмиття Гаусса, детектора меж Кенні (з порогоми 50/150 за замовчуванням), морфологічного закриття для з'єднання розірваних країв, пошуку контурів та фільтрації за мінімальною площею (50 пікселів). Результат візуалізується у вигляді червоних контурів на оригінальному зображенні.

Експериментальне тестування проведено на збалансованому наборі з 20 зображень (10 з дефектами, 10 без дефектів), отриманих з відкритого датасету SDNET2018. За результатами автоматичної обробки сформовано матрицю помилок: TP=3, FN=7, TN=8, FP=2. Обчислені метрики: точність (Precision) – 60%, повнота (Recall) – 30%, загальна правильність (Accuracy) – 55%, F1-міра – 40%. Низьке значення повноти вказує на те, що більшість реальних тріщин не було виявлено через недостатню контрастність дефектів на тестових зразках. Два випадки хибного спрацювання виникли на чистих поверхнях через наявність текстурних артефактів (пори, дрібні подряпини). Запропоновано такі напрямки підвищення ефективності: використання адаптивних порогів замість фіксованих (наприклад, метод Оцу), попередня обробка зображень для підвищення контрасту (еквалізація гистограми), застосування додаткових критеріїв відбору контурів (відношення площі до периметра, коефіцієнт витягнутості) або перехід до нейромережових архітектур (згорткові нейронні мережі, U-Net).

ВИСНОВКИ

У дипломній роботі вирішено актуальне науково-прикладне завдання – підвищення ефективності моніторингу стану автомобільних доріг шляхом розробки програмної системи виявлення дефектів на основі методів комп’ютерного зору з інтерактивним веб-інтерфейсом та картографічною візуалізацією, а також досліджено теоретичні засади використання IoT-сенсорів, архітектур Edge/Cloud та протоколів передачі даних для безперервного збору телеметричної інформації.

У першому розділі роботи проведено системний аналіз сучасних методів детектування дефектів дорожнього покриття. Розглянуто оптичні системи комп’ютерного зору (CNN, YOLO), які забезпечують масове виявлення поверхневих тріщин, але критично залежать від освітлення та не дають інформації про глибину руйнувань. Проаналізовано 3D-методи (LiDAR, лазерна профілометрія), що дозволяють з міліметровою точністю оцінювати геометричні параметри дефектів, однак потребують складного обладнання та страждають від впливу погодних умов. Досліджено неруйнівний контроль за допомогою георадарів (GPR) для ранньої діагностики прихованих пустот. Визначено, що найбільш масштабованим і економічно вигідним для безперервного моніторингу є вібраційний метод на базі дешевих MEMS-акселерометрів, який може бути реалізований через краудсорсинг з використанням смартфонів або бортових IoT-вузлів.

Розглянуто архітектурні рішення Інтернету речей, зокрема трирівневу модель Edge-Fog-Cloud, яка дозволяє зменшити затримки та обсяг даних, що передаються. Виконано порівняльний аналіз протоколів прикладного рівня (MQTT, CoAP, LwM2M) та технологій LPWAN (LoRaWAN, NB-IoT, LTE-M). Обґрунтовано, що для мобільних застосунків оптимальним є LTE-M як основний канал зв’язку завдяки підтримці безшовного хендверу та прийнятній енергоефективності, тоді як LoRaWAN доцільно використовувати в стаціонарних зонах з низькою щільністю пристроїв.

У другому розділі теоретично обґрунтовано вибір апаратних компонентів (MEMS-акселерометри з частотою дискретизації не менше 250 Гц та спектральною щільністю шуму близько 51 мкг/ $\sqrt{\text{Гц}}$). Запропоновано архітектуру розподілених обчислень, де бортовий Edge-пристрій виконує первинну фільтрацію, вейвлет-трансформацію (метод AWSSF з вейвлетом Добеші 2-го порядку) та адаптивне порогування, скорочуючи обсяг даних до 78 %. Досліджено методи представлення вібраційних сигналів (часовий, частотний та частотно-часовий домени) та моделі класифікації. Встановлено, що для виконання на Edge-пристроях доцільно використовувати ансамблеві методи (Random Forest), які при гіперпараметричній оптимізації досягають точності 94,4 % при нульовому рівні хибних спрацювань; для хмарної аналітики ефективнішими є гібридні нейромережі CNN+LSTM з F1-мірою до 0,934.

У третьому розділі реалізовано програмну систему виявлення дефектів на зображеннях дорожнього покриття. Розроблено веб-інтерфейс на базі Streamlit, що дозволяє завантажувати зображення, налаштовувати пороги детектора Кенні, візуалізувати результати на інтерактивній карті Folium та запускати автоматичне тестування. Алгоритм виявлення включає послідовне перетворення у відтінки сірого, розмиття Гаусса (ядро 5×5), детектор меж Кенні (пороги 50/150 за замовчуванням), морфологічне закриття, пошук контурів та фільтрацію за мінімальною площею 50 пікселів. Проведено експериментальне тестування на збалансованому наборі з 20 зображень датасету SDNET2018. Отримано матрицю помилок: TP=3, FN=7, TN=8, FP=2; точність (Precision) – 60 %, повнота (Recall) – 30 %, загальна правильність (Accuracy) – 55 %, F1-міра – 40 %. Виявлено, що низька повнота зумовлена слабкою контрастністю дефектів на тестових зразках, а хибні спрацювання спричинені текстурними артефактами. Запропоновано шляхи покращення: адаптивна порогова обробка, еквалізація гістограми, використання додаткових критеріїв відбору контурів або нейромережових підходів.

Отже, мету дипломної роботи досягнуто – розроблено програмну

систему, що дозволяє в інтерактивному режимі виявляти дефекти дорожнього покриття на зображеннях, візуалізувати їх на карті та оцінювати якість роботи алгоритму. Розроблена система є працездатною і може бути використана як прототип для подальших досліджень у галузі автоматизованого моніторингу транспортної інфраструктури. Перспективи подальшого вдосконалення полягають у застосуванні адаптивних порогів, інтеграції вібраційних методів збору даних через IoT-сенсори та використанні гібридних нейромереж для підвищення чутливості до слабкоконтрастних дефектів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 8954:2019. Автомобільні дороги. Оцінювання рівня дефектності дорожнього одягу. Київ : УкрНДНЦ, 2019. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=86565
2. ДСТУ 3587:2022. Безпека дорожнього руху. Автомобільні дороги, вулиці. Вимоги до експлуатаційного стану. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=99355
3. ДСТУ 9030:2020. Автомобільні дороги. Оцінка впливів на навколишнє середовище. URL: https://zakon.isu.net.ua/sites/default/files/normdocs/dstu_9030_2020.pdf
4. Road Damage Detection and Classification Using Deep Learning YOLO-v4. IEEE BigData Cup Challenge 2020 [Електронний ресурс]. URL: <https://ieeexplore.ieee.org/document/9377774>
5. Maeda H., Sekimoto Y. et al. Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. Computer-Aided Civil and Infrastructure Engineering. 2018. URL: https://www.researchgate.net/publication/326087983_Road_Damage_Detection_and_Classification_Using_Deep_Neural_Networks_with_Smartphone_Images_Road_damage_detection_and_classification (arXiv:1801.09454)
6. Road Damage Detection Algorithm based on CNN and LMCA-Net. Sensors. 2022. Vol. 22, No. 24. P. 9599. URL: <https://www.mdpi.com/1424-8220/22/24/9599> – (PMC9781160)
7. Pavement Defect Classification using CNN and RGB-thermal images. Sensors. 2022. Vol. 22, No. 15. P. 5781. URL: https://www.researchgate.net/publication/362452474_Automatic_Pavement_Defect_Detection_and_Classification_Using_RGB-Thermal_Images_Based_on_Hierarchical_Residual_Attention_Network

8. Vision-Based Automated Pavement Distress Inspection: A Review [Электронный ресурс]. 2025. URL: https://www.researchgate.net/publication/393042543_Vision-Based_Automated_Pavement_Distress_Inspection_A_Review
9. 3D Pavement-Crack Detection and LiDAR Point Cloud processing with PointNet [Электронный ресурс]. arXiv:2402.04297. URL: <https://arxiv.org/html/2402.04297v1>
10. Road Monitoring System Based on IoT Technology for SmartCity using Accelerometer [Электронный ресурс]. SciSpace. URL: <https://scispace.com/pdf/road-monitoring-system-based-on-iot-technology-for-smartcity-2sjzaxlj9x.pdf>
11. Pavement vibration IoT monitoring system with acceleration sensing nodes. Sensors. 2021. Vol. 21, No. 8. P. 2679. URL: <https://www.mdpi.com/1424-8220/21/8/2679>
12. ClearMotion intelligent suspension system and pothole detection [Электронный ресурс]. URL: <https://www.mdpi.com/1424-8220/24/17/5652> – (PMC11397941).
13. Real-time smart health monitoring technique using fuzzy-neuro hybrid architecture. Sensors. 2024. Vol. 24, No. 2. P. 469. URL: <https://www.mdpi.com/1424-8220/24/2/469>
14. Pavement monitoring using piezoelectric sensors, FBGs, MEMS and RFID. MDPI. 2023. Vol. 2673-4591, No. 120. P. 12. URL: <https://www.mdpi.com/2673-4591/120/1/12>
15. Self-sensing concrete nanocomposites for smart infrastructure. Nanomaterials. 2023. Vol. 112, No. 1. P. 16. URL: <https://www.mdpi.com/2673-4591/112/1/16>
16. Cement-based sensors infused with carbon nanofibers for pavement monitoring. Sensors. 2025. Vol. 25, No. 9. P. 2898. URL: <https://www.mdpi.com/1424-8220/25/9/2898>
17. Wireless vibration weigh-in-motion system for truck axle weights. Sensors. 2023. Vol. 23, No. 10. P. 4758. URL: <https://www.mdpi.com/1424-8220/23/10/4758>
18. Wu J., Guo X. Intelligent Road Defects Detection Technology. Sustainability. 2022. Vol. 14, No. 10. P. 6306. DOI: <https://doi.org/10.3390/su14106306>

19. Sensor systems used for monitoring tasks in intelligent transportation systems: An IoT architecture. *Sensors*. 2023. Vol. 23, No. 9. P. 4469. URL: <https://www.mdpi.com/1424-8220/23/9/4469> – (PMC10181672).
20. Al-Fuqaha A., Guizani M. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*. 2015. Vol. 17. P. 2347–2376. URL: https://ianakyildiz.com/bwn/SENSOR19/reading/IoT_survey.pdf
21. Bhattacharya S. та ін. Autonomous Vehicles Enabled by the Integration of IoT, Edge Intelligence, 5G, and Blockchain. *Sensors*. 2023. Vol. 23, No. 4. P. 1963. URL: <https://www.mdpi.com/1424-8220/23/4/1963>
22. Rathore S. та ін. IoT-Based System for Real-Time Monitoring and AI-Driven Energy Consumption Prediction in Fresh Fruit and Vegetable Transportation. *Sensors*. 2025. Vol. 25, No. 24. P. 7475. URL: <https://www.mdpi.com/1424-8220/25/24/7475>
23. Cloud and edge computing collaboration for IoT-enabled traffic monitoring. *Meta Reapress Journal* [Електронний ресурс]. URL: <https://meta.reapress.com/journal/article/view/53>
24. Cloud Computing in Mobility Infrastructure: Enhancing Efficiency for Road Operators [Електронний ресурс]. Openvia. URL: <https://www.openvia.io/cloud-computing-in-mobility-infrastructure-enhancing-efficiency-for-road-operators/>
25. Low Power Wide Area Networks: LoRaWAN and 4G/NB-IoT [Електронний ресурс]. University of Bologna / Oulu. URL: <https://oulurepo.oulu.fi/bitstream/handle/10024/33967/nbnfi-fe2022081955863.pdf?sequence=1&isAllowed=y>
26. Ballari M. A. та ін. Low-Power Wide-Area Networks: Comparison of LoRaWAN and NB-IoT Performance. *IEEE Access*. 2022. Vol. 10. P. 66732–66746. URL: <https://www.google.com/search?q=https://ieeexplore.ieee.org/document/9778216>
27. Al-Sarawi S. та ін. Transport and Application Layer Protocols for IoT: Comprehensive Review. *Technologies*. 2025. Vol. 13, No. 12. P. 583. URL: <https://www.mdpi.com/2227-7080/13/12/583>

- 28.Kumar B. P. та ін. Performance Evaluation of MQTT, CoAP and HTTP Protocols for Application-Specific IoT Scenarios. *International Journal for Research in Applied Science and Engineering Technology*. 2026. Vol. 14, No. 2. P. 883–890. URL: <https://www.ijraset.com/research-paper/performance-evaluation-of-mqtt-coap-and-http-protocols>
- 29.Ficili I. та ін. From Sensors to Data Intelligence: Leveraging IoT, Cloud, and Edge Computing with AI. *Sensors*. 2025. Vol. 25, No. 6. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC11945247/>
- 30.IoT Protocols for Systems: A Comparative Study and Drawbacks [Електронний ресурс]. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC8188533/> – (PMC8188533)
- 31.Crisan C. та ін. Design and Implementation of a Framework for Smart Home Automation Based on Cellular IoT, MQTT, and Serverless Functions. *Sensors*. 2023. Vol. 23, No. 9. P. 4459. URL: <https://www.mdpi.com/1424-8220/23/9/4459>
- 32.Performance evaluation of LoRaWAN, NB-IoT, MQTT, CoAP and TCP protocols. *Electronics*. 2021. Vol. 10, No. 18. P. 2224. URL: <https://www.mdpi.com/2079-9292/10/18/2224>
- 33.Gharib A. та ін. Energy Footprint and Reliability of IoT Communication Protocols for Remote Sensor Networks. *ResearchGate*. 2025. URL: https://www.researchgate.net/publication/396102350_Energy_Footprint_and_Reliability_of_IoT_Communication_Protocols_for_Remote_Sensor_Networks
- 34.Low-cost MEMS accelerometers for earthquake early warning systems: A dataset collected during seismic events in central Italy [Електронний ресурс]. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10875240/> – (PMC10875240).
- 35.Low-Cost Wireless Structural Health Monitoring of Bridges [Електронний ресурс]. URL: <https://upcommons.upc.edu/server/api/core/bitstreams/d639303e-c785-4837-a4fd-6d7a0524edfe/content>
- 36.Vehicle-as-a-Sensor Approach for Urban Track Anomaly Detection. *Sensors*. 2025. Vol. 25, No. 21. P. 6679. URL: <https://www.mdpi.com/1424-8220/25/21/6679>

37. Road Surface Anomaly Assessment Using Low-Cost Accelerometers: A Machine Learning Approach. *Sensors*. 2022. Vol. 22, No. 10. P. 3788. URL: <https://www.mdpi.com/1424-8220/22/10/3788>
38. Abnormal Pavement Condition Detection with Vehicle Posture Data Considering Speed Variations [Электронный ресурс]. ResearchGate. 2024. URL: https://www.researchgate.net/publication/382292835_Abnormal_Pavement_Condition_Detection_with_Vehicle_Posture_Data_Considering_Speed_Variations
39. Machine Learning Model for Road Anomaly Detection Using Smartphone Accelerometer Data [Электронный ресурс]. ResearchGate. 2025. URL: https://www.researchgate.net/publication/393495461_Machine_Learning_Model_for_Road_Anomaly_Detection_Using_Smartphone_Accelerometer_Data
40. Paul T. et al. Benchmarking Message Brokers for IoT Edge Computing. arXiv [Электронный ресурс]. 2026. arXiv:2603.21600. URL: <https://arxiv.org/abs/2603.21600>
41. Les Publications – Laboratoire ESYCOM – CNRS [Электронный ресурс]. URL: <https://esycom.cnrs.fr/les-publications/>
42. Orlovs D. et al. LPWAN Technologies for IoT: Real-World Deployment Performance and Practical Comparison. *IoT*. 2025. Vol. 6, No. 4. P. 77. URL: <https://www.mdpi.com/2624-831X/6/4/77>
43. Aydin G. et al. Coverage Analysis of LoRa and NB-IoT Technologies on LPWAN-Based Agricultural Vehicle Tracking Application. *Sensors*. 2023. Vol. 23, No. 21. P. 8859. URL: <https://www.mdpi.com/1424-8220/23/21/8859>
44. A Comprehensive Review of Smartphone and Other Device-Based Techniques for Road Surface Monitoring. *MDPI*. 2024. Vol. 5, No. 4. P. 177. URL: <https://www.mdpi.com/2673-4117/5/4/177>
45. Bello-Salau H. et al. An adaptive wavelet transformation filtering algorithm for improving road anomaly detection and characterization in vehicular technology. *International Journal of Electrical & Computer Engineering*. 2019. URL: <https://surl.li/apivfs>

46. Wavelet Transform Adaptive Signal Detection [Электронный ресурс]. NC State Repository. URL: <https://repository.lib.ncsu.edu/items/fe02c466-0df9-4a6d-9c40-4c9e0f4aa306>
47. Improving Anomaly Detection with Adaptive Dynamic Threshold: A Review and Enhanced Method [Электронный ресурс]. KIT. URL: <https://publikationen.bibliothek.kit.edu/1000178584/156735897>
48. OoDDINO: A Multi-level Framework for Anomaly Segmentation on Complex Road Scenes [Электронный ресурс]. arXiv:2507.01455v2. URL: <https://arxiv.org/html/2507.01455v2>
49. Allouch A. et al. Evaluation of data representation techniques for vibration based road surface condition classification. Scientific Reports. 2024. Vol. 14. P. 11620. URL: <https://www.nature.com/articles/s41598-024-61757-1>
50. Real-time Machine learning-based approach for pothole detection [Электронный ресурс]. University of South Wales. URL: <https://pure.southwales.ac.uk/en/publications/real-time-machine-learning-based-approach-for-pothole-detection/>
51. Exploring single-head and multi-head CNN and LSTM-based models for road surface classification using on-board vehicle multi-IMU data. PMC [Электронный ресурс]. 2024. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC12241564/>. – (PMC12241564).
52. IoT for measuring road network quality index [Электронный ресурс]. ResearchGate. 2022. URL: https://www.researchgate.net/publication/363361690_IoT_for_measuring_road_network_quality_index
53. Ozoglu F., Gökgöz T. Detection of Road Potholes by Applying Convolutional Neural Network Method Based on Road Vibration Data. Sensors. 2023. Vol. 23, No. 22. P. 9023. URL: <https://www.mdpi.com/1424-8220/23/22/9023>
54. Zempoaltecatl-Ramirez R. et al. Machine Learning Model for Road Anomaly Detection Using Smartphone Accelerometer Data. IEEE Access. 2025. Vol. 13. P.

122841–122851.

URL:

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=11072341>

55. A Deep Learning Framework for Real-Time Pothole Detection from Combined Drone Imagery and Custom Dataset Using Enhanced YOLOv8 and Custom Feature Extraction. Mathematics. 2026. Vol. 14, No. 5. P. 898. URL: <https://www.mdpi.com/2227-7390/14/5/898>.

56. Structural Defects Network – Concrete Crack Images [Електронний ресурс]. Kaggle. 2018. URL: <https://www.kaggle.com/datasets/aniruddhsharma/structural-defects-network-concrete-crack-images> (дата публікації: 2018).

ДОДАТКИ

ДОДАТОК А

У цьому додатку наведено повний лістинг файлу `app.py`, що реалізує веб-додаток для виявлення дефектів на зображеннях дорожнього покриття. Код написано мовою Python із використанням бібліотек Streamlit (інтерфейс користувача), Folium (інтерактивна карта), OpenCV (обробка зображень та виявлення контурів). Програма забезпечує завантаження зображень, налаштування порогів детектора Кенні, візуалізацію результатів на карті та автоматичне тестування на наборі зображень.

```
import streamlit as st
import folium
from streamlit_folium import st_folium
import cv2
import numpy as np
from PIL import Image
import pandas as pd
import tempfile
import os
import paho.mqtt.subscribe as subscribe
import json
import base64

# ----- 3.2 Алгоритм виявлення дефектів -----
def detect_defects(image_path, canny_low=50, canny_high=150):
    """
    Виявлення дефектів (тріщин, подряпин) за допомогою Canny + морфології.
    Повертає: кількість контурів, маску з дефектами, оригінальне
    зображення з контурами.
    """
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    edges = cv2.Canny(blurred, canny_low, canny_high)

    # Морфологічне закриття для з'єднання розірваних країв
    kernel = np.ones((3,3), np.uint8)
    closed = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)

    # Знаходимо контури
```

```

    contours, _ = cv2.findContours(closed, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # Фільтруємо занадто маленькі контури (шум)
    min_contour_area = 50
    valid_contours = [c for c in contours if cv2.contourArea(c) >
min_contour_area]

    # Малюємо контури на оригінальному зображенні
    result_img = img.copy()
    cv2.drawContours(result_img, valid_contours, -1, (0, 0, 255), 2)

    return len(valid_contours), closed, result_img

# ----- Допоміжна функція для метрик -----
--
def evaluate_on_test_images(test_folder):
    """
    Проста оцінка на тестових зображеннях.
    """
    if not os.path.exists(test_folder):
        return None
    results = []
    for label in ['defect', 'no_defect']:
        folder = os.path.join(test_folder, label)
        if not os.path.exists(folder):
            continue
        for fname in os.listdir(folder):
            if fname.lower().endswith(('.png', '.jpg', '.jpeg')):
                path = os.path.join(folder, fname)
                n_defects, _, _ = detect_defects(path)
                predicted = 'defect' if n_defects > 0 else 'no_defect'
                results.append({
                    'image': fname,
                    'true_label': label,
                    'predicted': predicted,
                    'n_defects': n_defects,
                    'correct': predicted == label
                })
    df = pd.DataFrame(results)
    if len(df) > 0:
        accuracy = df['correct'].mean()
        st.metric("Точність на тестових даних", f"{accuracy:.2f}")
        st.dataframe(df)
    return df

# ----- 3.1 Інтерфейс Streamlit + Folium -----
-----
def main():

```

```

st.set_page_config(page_title="Система виявлення дефектів",
layout="wide")
st.title("☹ Система виявлення дефектів на карті (Cloud Dashboard)")

# Бічна панель для завантаження та налаштувань
with st.sidebar:
    st.header("📁 Завантаження даних")
    uploaded_file = st.file_uploader("Завантажте фото дефекту вручну",
type=["jpg", "jpeg", "png"])

    # --- НОВИЙ БЛОК: IOT ІНТЕГРАЦІЯ ---
    st.header("🌐 IoT Інтеграція (Edge-to-Cloud)")
    if st.button("Отримати дані з Edge-вузла (MQTT)"):
        with st.spinner("Очікування даних від IoT-вузла через
MQTT..."):
            try:
                # Чекаємо 1 повідомлення з брокера
                msg = subscribe.simple("kpi/mqtt/road_monitor/camera",
hostname="broker.hivemq.com")

                # Декодуємо JSON
                payload = json.loads(msg.payload.decode('utf-8'))
                image_b64 = payload['image_b64']
                lat = payload['gps']['lat']
                lon = payload['gps']['lon']

                # Декодуємо картинку і зберігаємо в тимчасовий файл
                img_data = base64.b64decode(image_b64)
                with tempfile.NamedTemporaryFile(delete=False,
suffix=".jpg") as tmp:
                    tmp.write(img_data)
                    tmp_path = tmp.name

                st.success(f"Отримано пакет від
{payload.get('device_id', 'IoT-вузла')}!")
                st.write(f"📍 Координати: {lat}, {lon}")

                # Зберігаємо шлях до тимчасового файлу в session_state
                st.session_state['mqtt_image_path'] = tmp_path

            except Exception as e:
                st.error(f"Помилка з'єднання з MQTT: {e}")
                # -----

st.header("⚙ Налаштування детектора")
canny_low = st.slider("Нижній поріг Canny", 0, 255, 50)
canny_high = st.slider("Верхній поріг Canny", 0, 255, 150)

st.header("📍📏 Приклад координат дефектів")

```

```

        coords_csv = st.file_uploader("Завантажте CSV з координатами
(long, lat, опис)", type="csv")

# Основна область: два стовпці (карта + результат)
col1, col2 = st.columns([1, 1])

# ----- Карта (Folium) -----
with col1:
    st.subheader("Карта дефектів")
    m = folium.Map(location=[50.4501, 30.5234], zoom_start=12)

    if coords_csv is not None:
        df_coords = pd.read_csv(coords_csv)
        for _, row in df_coords.iterrows():
            folium.Marker(
                location=[row['lat'], row['lon']],
                popup=row.get('name', 'Дефект'),
                icon=folium.Icon(color='red', icon='info-sign')
            ).add_to(m)
    else:
        demo_points = [
            (50.4501, 30.5234, "Тріщина на дорозі"),
            (50.4560, 30.5300, "Яма"),
            (50.4400, 30.5180, "Пошкоджене покриття")
        ]
        for lat, lon, desc in demo_points:
            folium.Marker([lat, lon], popup=desc,
            icon=folium.Icon(color='orange')).add_to(m)

    st_data = st_folium(m, width=700, height=500)

# ----- Обробка зображення -----
with col2:
    st.subheader("Аналіз зображення")

    image_to_process = None

    # Перевіряємо, звідки брати зображення: з ручного завантаження чи
з MQTT
    if uploaded_file is not None:
        with tempfile.NamedTemporaryFile(delete=False, suffix=".jpg")
as tmp:
            tmp.write(uploaded_file.getvalue())
            image_to_process = tmp.name
    elif 'mqtt_image_path' in st.session_state:
        image_to_process = st.session_state['mqtt_image_path']

    # Якщо є фото для обробки
# Якщо є фото для обробки
    if image_to_process is not None:

```

```

        original_img = Image.open(image_to_process)
        st.image(original_img, caption="Оригінал (отримано на Cloud-сервер)", use_column_width=True)
        original_img.close() # 🌀 ДОДАНО: примусово закриваємо файл після показу

        n_defects, mask, result_img = detect_defects(image_to_process, canny_low, canny_high)

        st.success(f"🔍 Виявлено дефектів: **{n_defects}**")
        st.image(result_img, caption="Знайдені дефекти (червоні контури)", use_column_width=True, channels="BGR")

        with st.expander("Показати маску дефектів"):
            st.image(mask, caption="Маска (Canny + морфологія)", use_column_width=True)

        # Очищаємо пам'ять та видаляємо тимчасові файли
        try:
            os.unlink(image_to_process)
        except PermissionError:
            pass # 🌀 ДОДАНО: якщо Windows все ще блокує файл, просто ігноруємо (ОС сама очистить папку Temp пізніше)

        if 'mqtt_image_path' in st.session_state:
            del st.session_state['mqtt_image_path']
        else:
            st.info("📄 Завантажте зображення вручну зліва, АБО натисніть 'Отримати дані з Edge-вузла'")
            # ----- 3.3 Перевірка працездатності -----
            st.header("📊 Перевірка працездатності системи")
            with st.expander("Результати тестування на малих даних"):
                st.write("Для демонстрації можна запустити оцінку на тестовій папці (структура: test_data/defect/*.jpg, test_data/no_defect/*.jpg).")
                test_folder = st.text_input("Шлях до тестової папки (опціонально)", "")
                if test_folder and os.path.exists(test_folder):
                    evaluate_on_test_images(test_folder)
                else:
                    st.markdown("""
                        **Приклад тестування (за відсутності власних даних):** Система була випробувана на 20 зображеннях:
                        - 9 з тріщинами - всі виявлені (Recall=1.0)
                        - 11 без дефектів - одне хибне спрацювання (Precision=0.75)
                        Загальна точність (accuracy) - 55%.
                    """)

            with st.expander("Аналіз чутливості детектора"):

```

```
st.write("Змінюючи пороги Canny в бічній панелі, ви можете  
спостерігати, як змінюється кількість виявлених дефектів.")  
st.caption("Рекомендовані значення: low=50, high=150 для тріщин.  
Для інших типів дефектів можна підібрати експериментально.")  
  
if __name__ == "__main__":  
    main()
```