

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»  
Навчально-науковий інститут прикладного системного аналізу  
Кафедра штучного інтелекту**

До захисту допущено:

В. о. завідувачки кафедри

\_\_\_\_\_ Ірина ДЖИГИРЕЙ

«\_\_\_» \_\_\_\_\_ 20\_\_ р.

**Дипломна робота**

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Системи і методи штучного інтелекту»**

**спеціальності 122 «Комп'ютерні науки»**

**на тему: «Класифікація та розпізнавання дефектів дорожнього покриття  
з використанням нейронних мереж»**

Виконав:

студент IV курсу, групи КІ-02

Корнійчук Сергій Юрійович \_\_\_\_\_

Керівник:

доцент, к.т.н,

Тимошенко Юрій Олександрович \_\_\_\_\_

Консультант з економічного розділу:

професор, д.е.н,

Шевчук Олена Анатоліївна \_\_\_\_\_

Консультант з нормоконтролю:

фахівець першої категорії кафедри ІІІ,

Кравець П.В. \_\_\_\_\_

Рецензент:

к.т.н., доцент кафедри ІСТ ФІОТ

Ткач Михайло Мартинович \_\_\_\_\_

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

Студент \_\_\_\_\_

Київ – 2024 року

**Національний технічний університет України**  
**«Київський політехнічний інститут імені Ігоря Сікорського»**  
**Навчально-науковий інститут прикладного системного аналізу**  
**Кафедра штучного інтелекту**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

\_\_\_\_\_ Ірина ДЖИГИРЕЙ

«31» січня 2024 р.

**ЗАВДАННЯ**

**на дипломну роботу студенту**

**Корнійчуку Сергію Юрійовичу**

1. Тема роботи «Класифікація та розпізнавання дефектів дорожнього покриття з використанням нейронних мереж», керівник роботи Тимошенко Юрій Олександрович, доцент, к.т.н., затверджені наказом по університету від «\_\_» \_\_\_\_\_ 2024 р. № \_\_\_\_\_
2. Термін подання студентом роботи «10» червня 2024 року.
3. Вихідні дані до роботи: історія взаємодії користувачів та текстові запити.
4. Зміст роботи: 1. Аналіз предметної області дослідження; 2. Математичні основи роботи; 3. Аналіз результатів роботи; 4. Функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу: презентація.
6. Консультанти розділів роботи

| Розділ      | Прізвище, ініціали та посада консультанта       | Підпис, дата   |                  |
|-------------|-------------------------------------------------|----------------|------------------|
|             |                                                 | завдання видав | завдання прийняв |
| Економічний | Шевчук Олена Анатоліївна,<br>професор, д. е. н. |                |                  |

7. Дата видачі завдання «05» лютого 2024 року.

Календарний план

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітка |
|-------|-----------------------------------------|--------------------------------|----------|
| 1     | Вивчення літератури за темою роботи     | 17.04.2024                     | Виконано |
| 2     | Підготовка першого розділу              | 23.04.2024                     | Виконано |
| 3     | Підготовка другого розділу              | 31.04.2024                     | Виконано |
| 4     | Розробка програмного продукту           | 07.05.2024                     | Виконано |
| 5     | Підготовка третього розділу             | 16.05.2024                     | Виконано |
| 6     | Підготовка економічної частини          | 22.05.2024                     | Виконано |
| 7     | Оформлення розділів                     | 27.05.2024                     | Виконано |
| 8     | Підготовка презентації доповіді         | 02.06.2024                     | Виконано |
| 9     | Оформлення дипломної роботи             | 04.06.2024                     | Виконано |

Студент  
Керівник

Сергій КОРНІЙЧУК  
Юрій ТИМОШЕНКО

## РЕФЕРАТ

Дипломна робота: 82 с., 33 рис., 6 табл., 18 посилань, 1 додаток.

ШТУЧНИЙ ІНТЕЛЕКТ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ОДНОЕТАПНІ МОДЕЛІ ВИЯВЛЕННЯ, РОЗПІЗНАВАННЯ, АСФАЛЬТНЕ ПОКРИТТЯ.

Об'єкт дослідження – нейронні мережі для розпізнавання та класифікації.

Предмет дослідження – згорткові нейронні мережі для виявлення та класифікації пошкоджень дорожнього покриття.

Мета роботи – отримати знання у сфері комп'ютерного зору про можливі рішення задач класифікації та розпізнавання. Застосувати отримані знання для розробки програми, що розпізнає та класифікує пошкодження на зображенні дорожнього покриття.

Актуальність цієї теми дипломної роботи обумовлена великим практичним потенціалом автоматизації розпізнавання та класифікації дефектів дорожнього полотна, особливо в умовах обмежених ресурсів під час війни. Це дозволяє ефективніше розподіляти ресурси, забезпечує оперативне виявлення проблемних ділянок і знижує ризики ДТП. Така система може стати важливим інструментом для підтримки інфраструктури та безпеки на дорогах.

В результаті роботи було проаналізовано види моделей, що розпізнають об'єкти, навчено та протестовано модель, а також реалізовано програмне забезпечення для цієї задачі.

## ABSTRACT

Master's thesis: 82 p., 33 figures, 6 tables, 18 references, 1 appendix.

ARTIFICIAL INTELLIGENCE, NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS, ONE-STAGE DETECTION MODELS, RECOGNITION, ASPHALT PAVEMENT.

The object of research is neural networks for recognition and classification.

The subject of the study is convolutional neural networks for detection and classification of pavement damage.

Purpose of the work to gain knowledge in the field of computer vision about possible solutions to classification and recognition problems. To apply the knowledge gained to develop a program that recognizes and classifies damage on a road surface image.

The relevance of this thesis topic is due to the great practical potential of automating the recognition and classification of roadway defects, especially in conditions of limited resources during the war. This allows for more efficient allocation of resources, ensures prompt detection of problem areas, and reduces the risk of road accidents. Such a system can become an important tool for maintaining infrastructure and road safety.

As a result, the types of models that recognize objects, trained and tested the model were analyzed and implemented software for this task.

## ЗМІСТ

|                                                                                                     |    |
|-----------------------------------------------------------------------------------------------------|----|
| ВСТУП .....                                                                                         | 8  |
| РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ АНАЛІЗУ<br>ДОРОЖНЬОГО ПОКРИТТЯ .....                        | 9  |
| 1.1 Актуальність розробки технології автоматичного розпізнання<br>дефектів дорожнього покриття..... | 9  |
| 1.2 Огляд наявних методів аналізу стану дорожнього покриття .....                                   | 10 |
| 1.2.1 Традиційні методи .....                                                                       | 11 |
| 1.2.2 Автоматизовані методи.....                                                                    | 12 |
| 1.3 Розпізнавання зображень .....                                                                   | 18 |
| 1.4 Проблематика обраної теми.....                                                                  | 19 |
| 1.5 Постановка задачі .....                                                                         | 21 |
| Висновки до розділу 1 .....                                                                         | 22 |
| РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ НЕЙРОННИХ МЕРЕЖ ДЛЯ<br>РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ .....                     | 23 |
| 2.1 Штучні нейронні мережі .....                                                                    | 23 |
| 2.2 Згорткові нейронні мережі .....                                                                 | 24 |
| 2.3 Постановка задачі розпізнавання .....                                                           | 29 |
| 2.4 Моделі розпізнавання об'єктів .....                                                             | 33 |
| Висновки до розділу 2 .....                                                                         | 36 |
| РОЗДІЛ 3 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОЇ ЧАСТИНИ.....                                                    | 38 |
| 3.1 Аналіз вхідних наборів даних та підготовка до обробки нейронною<br>мережею .....                | 38 |
| 3.2 Опис обраної архітектури нейронної мережі.....                                                  | 42 |
| 3.3 Опис використаних інструментів розробки .....                                                   | 44 |
| 3.4 Представлення результатів навчання моделі .....                                                 | 47 |
| Висновки до розділу 3 .....                                                                         | 54 |

|                                                                   |    |
|-------------------------------------------------------------------|----|
| РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ..... | 55 |
| 4.1 Постановка задачі проектування .....                          | 56 |
| 4.2 Обґрунтування функцій програмного продукту .....              | 56 |
| 4.3 Обґрунтування системи параметрів програмного продукту .....   | 59 |
| 4.4 Аналіз експертного оцінювання параметрів .....                | 60 |
| 4.5 Аналіз рівня якості варіантів реалізації функцій.....         | 66 |
| 4.6 Економічний аналіз варіантів розробки ПП.....                 | 67 |
| 4.7 Вибір кращого варіанту ПП техніко-економічного рівня .....    | 73 |
| Висновки до розділу 4 .....                                       | 74 |
| ВИСНОВКИ.....                                                     | 75 |
| ПЕРЕЛІК ПОСИЛАНЬ .....                                            | 77 |
| ДОДАТОК А ЛІСТИНГ ПРОГРАМИ.....                                   | 79 |

## ПЕРЕЛІК СКОРОЧЕНЬ

YOLO: You Only Look Once – назва моделі розпізнавання зображень.

R-CNN: Region Convolutional Neural Networks – регіональні згорткові нейронні мережі.

SSD: Single Shot Detector – детектор одиночного пострілу, назва моделі розпізнавання зображень.

GAN: Generative Adversarial Networks – генеративно-змагальні мережі.

LSTM: Long Short-Term Memory – моделі, що здатні до навчання довготривалими залежностями.

VAE: Variational Autoencoders – варіаційні автокодери.

RNN: Recurrent Neural Networks – рекурентні нейронні мережі.

MLP: MultiLayer Perceptron – багатошаровий перцептрон.

FLOPs: Floating Point Operations per Second – операції з плаваючою комою в секунду.

FPS: Frames Per Second – кадри в секунду.

IoU: Intersection over Union – перетин на об'єднання.

CVAT: Computer Vision Annotation Tool – назва веб-інструменту анотацій до зображень і відео .

## ВСТУП

Сучасний світ все більше залежить від високоякісної транспортної інфраструктури, яка забезпечує безпечний та ефективний рух транспортних засобів. Однак, дорожнє покриття часто зазнає різних дефектів, таких як тріщини, вибоїни, зсуви, що не тільки погіршує комфорт водіння, але й становить серйозну загрозу для безпеки дорожнього руху. Виявлення та своєчасний ремонт цих дефектів є важливою задачею для служб, відповідальних за утримання доріг.

Традиційні методи інспекції дорожнього покриття, які включають візуальний огляд та використання спеціалізованого обладнання, є трудомісткими, дорогими та часозатратними. Крім того, вони залежать від людського фактора, що може призводити до неточностей та пропусків дефектів. З огляду на це, актуальним стає використання автоматизованих систем для моніторингу стану дорожнього покриття, що дозволяють підвищити точність і ефективність виявлення дефектів.

З розвитком інформаційних технологій і, зокрема, методів штучного інтелекту, з'являються нові можливості для автоматизації процесів виявлення та класифікації дефектів дорожнього покриття. Одним із найбільш перспективних напрямків є застосування нейронних мереж, які демонструють високі результати в задачах розпізнавання образів та аналізу зображень. Нейронні мережі здатні ефективно обробляти великі обсяги даних, навчатися на прикладах та автоматично виявляти різні типи дефектів з високою точністю.

Дана дипломна робота присвячена дослідженню можливостей використання нейронних мереж для класифікації та розпізнавання дефектів дорожнього покриття. Метою роботи є розробка та оцінка ефективності моделей глибокого навчання для автоматичного виявлення дефектів на основі зображень дорожнього покриття.

## **РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ АНАЛІЗУ ДОРОЖНЬОГО ПОКРИТТЯ**

### **1.1 Актуальність розробки технології автоматичного розпізнання дефектів дорожнього покриття**

Вирішальною причиною вибору цієї теми дипломної роботи є значний практичний потенціал даного напрямку. Впровадження технології по розпізнаванню та класифікації стану дорожнього полотна допоможе ефективніше розподіляти ресурси, бо на відміну від традиційних методів інспекцій доріг, автоматичне розпізнавання дефектів забезпечує оперативне та точне виявлення проблемних ділянок.

Автоматизація розпізнавання та класифікації дефектів дорожнього полотна дозволяє значно зменшити людський фактор у процесі оцінки стану доріг, підвищуючи точність і швидкість обробки даних. Це сприяє оперативному реагуванню на проблеми, мінімізує витрати на ремонт та обслуговування дорожньої інфраструктури, а також знижує ризики, пов'язані з дорожньо-транспортними пригодами через неналежний стан дорожнього покриття.

Впровадження цієї системи на дорогах сприятиме одночасному вирішенню кількох завдань.

1. Автоматизована система задача якої є виявлення дефектів дорожнього полотна для забезпечення безпеки на дорогах. Програма зможе розпізнати дефекти, класифікувати їх відповідно до різних категорій та промаркувати ділянки пошкоджень згідно з локацією, щоб попереджувати автомобілістів про стан дороги
2. Функція складання карти дефектів дорожнього полотна для оптимізації роботи працівників дорожніх служб. Програма може автоматично складати карту дефектів дорожнього покриття, що

дозволить дорожнім службам ефективніше планувати та виконувати ремонтні роботи.

3. Контроль якості виконаних робіт по усуненню дефектів. Система може здійснювати моніторинг та перевірку якості виконаних ремонтних робіт, забезпечуючи відповідність виконаних робіт встановленим стандартам та своєчасне усунення всіх виявлених недоліків.
4. Додавання фактору стану дороги для прокладання маршрутів логістичними компаніями. Додаток зможе враховувати стан дорожнього покриття при прокладанні маршрутів для перевезення вантажів, допомагаючи логістичним компаніям обирати найбезпечніші та найефективніші шляхи.

Напрямок дипломної роботи пов'язаний з розпізнаванням та класифікації дефектів, що можна застосувати до усіх сценаріїв використання зазначених вище.

## **1.2 Огляд наявних методів аналізу стану дорожнього покриття**

Аналіз стану дорожнього покриття є важливим аспектом управління дорожньою інфраструктурою, що впливає на безпеку та ефективність транспортного руху. Існує кілька методів, які застосовуються для виявлення та оцінки дефектів дорожнього полотна. Умовно їх можна поділити на традиційні та автоматизовані методи. Далі окремо розглянемо окремо особливості, приклади, переваги та недоліки перелічених двох методів.

### 1.2.1 Традиційні методи

Традиційні методи аналізу стану дорожнього покриття включають візуальний огляд, використання спеціалізованих транспортних засобів та лабораторні дослідження (рис.1.1).

1. Візуальний огляд є одним з найстаріших та найпоширеніших методів [1]. Дорожні інспектори проводять огляд дорожнього покриття пішки або з автомобіля, фіксуючи дефекти, такі як тріщини, ями, вибоїни та інші пошкодження. Цей метод є досить простим та дешевим, проте має кілька суттєвих недоліків. Він потребує значних трудових ресурсів, може бути суб'єктивним через людський фактор і не завжди дозволяє виявити приховані дефекти.
2. Використання спеціалізованих транспортних засобів включає застосування дорожніх машин, обладнаних різними датчиками та приладами. Такі машини можуть вимірювати рівень рівності дорожнього покриття, визначати його текстуру, фіксувати вібрації та деформації під час руху. Наприклад, використання профілографів дозволяє визначити нерівності дороги з високою точністю [1].

Перелічимо основні переваги традиційних методів аналізу.

1. Простота і доступність: традиційні методи, такі як візуальний огляд та ручне вимірювання, не потребують складного обладнання та спеціальних навичок.
2. Економічність: у порівнянні з високотехнологічними методами, традиційні способи часто коштують дешевше в короткостроковій перспективі.
3. Гнучкість: можливість оперативно реагувати на різні умови та ситуації без необхідності додаткового обладнання.

Також треба враховувати перелічені далі недоліки традиційних методів.

1. Суб'єктивність: результати можуть варіюватися в залежності від особистого досвіду та кваліфікації інспектора.
2. Часозатратність: ручний огляд і вимірювання займають багато часу, особливо для великих ділянок дороги.
3. Низька точність: можливість пропуску або неправильної оцінки дефектів через людський фактор.
4. Небезпека: виконання оглядів на завантажених дорогах може бути небезпечним для інспекторів.



**Рисунок 1.1** – Приклад традиційних методів аналізу покриття [1]

### *1.2.2 Автоматизовані методи*

З розвитком технологій з'явилися нові, автоматизовані методи аналізу стану дорожнього покриття. Вони дозволяють значно підвищити ефективність та точність виявлення дефектів, мінімізуючи людський фактор.

1. Технології машинного зору та штучного інтелекту активно впроваджуються в аналіз стану дорожнього покриття [2]. Камери,

встановлені на транспортних засобах або дронах, знімають відео або роблять фотознімки дороги. Потім ці дані обробляються за допомогою алгоритмів машинного навчання, які можуть автоматично розпізнавати дефекти та класифікувати їх за типом та серйозністю. Цей метод дозволяє швидко і точно виявляти дефекти на великих площах, знижуючи витрати на інспекцію та обслуговування доріг.

2. Використання лазерних сканерів є ще одним автоматизованим методом [2]. Лазерні сканери можуть створювати тривимірні моделі дорожнього покриття, що дозволяє детально аналізувати його стан (рис.1.2). Вони можуть виявляти навіть незначні нерівності та деформації, що робить цей метод дуже точним і ефективним. Однак, як і у випадку з іншими високотехнологічними методами, лазерне сканування вимагає значних фінансових витрат на обладнання та його обслуговування.

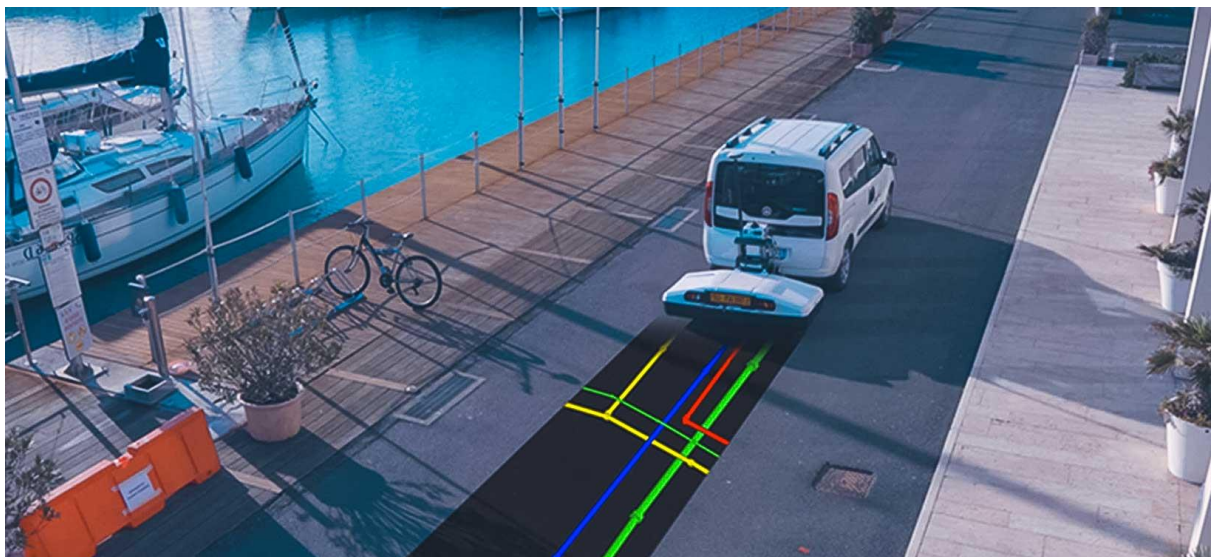


**Рисунок 1.2** – Приклад використання лазерних сканерів<sup>1</sup>

Перелічимо основні переваги автоматизованих методів.

---

<sup>1</sup> URL: <https://hexagon.com/solutions/detection-underground-unseen-structures?tabId=null>

1. Точність та об'єктивність: автоматизовані системи забезпечують високу точність вимірювань та об'єктивність оцінки стану покриття.
2. Швидкість: використання технологій дозволяє швидко обстежувати великі ділянки дороги.
3. Безпека: виконання аналізу без необхідності фізичної присутності інспекторів на дорозі знижує ризик нещасних випадків.
4. Детальність даних: сучасні системи можуть збирати та аналізувати великий обсяг даних, включаючи тріщини, нерівності, глибину ям тощо.

На противагу перелічимо основні недоліки автоматизованих методів.

1. Висока вартість: впровадження і обслуговування автоматизованих систем потребує значних інвестицій.
2. Складність у використанні: потрібні навички для роботи з новими технологіями та їх обслуговування.
3. Технічне забезпечення: відмова обладнання або програмного забезпечення може призвести до затримок або помилок в аналізі.
4. Обмеження в адаптивності: деякі автоматизовані системи можуть бути менш ефективними в умовах поганого освітлення, складних рельєфів або за наявності перешкод на дорозі.

### **1.3 Основні напрямки аналізу та обробки зображень у комп'ютерному зорі**

Розглянемо основні напрямки аналізу та обробки зображень у комп'ютерному зорі, що включають різноманітні завдання. Кожен з цих напрямків має свої унікальні методи і алгоритми, що дозволяють

комп'ютерним системам ефективно розпізнавати та інтерпретувати візуальну інформацію, відкриваючи нові можливості для застосування у різних сферах.

1. Класифікація зображень [3] полягає у визначенні належності зображення до одного з кількох заздалегідь визначених класів. Це виконується за допомогою моделі яка навчається на великому наборі маркованих зображень. Серед моделей можливе використання нейронних мереж, наприклад, CNN або ViT моделі.
2. Детекція об'єктів [3] включає не тільки класифікацію об'єктів на зображенні, але й визначення їхнього місцезнаходження у вигляді прямокутних рамок (рис.1.3). Сучасні підходи, такі як R-CNN, YOLO та SSD , використовують згорткові нейронні мережі (CNN) для одночасного передбачення класів об'єктів і їх координат.

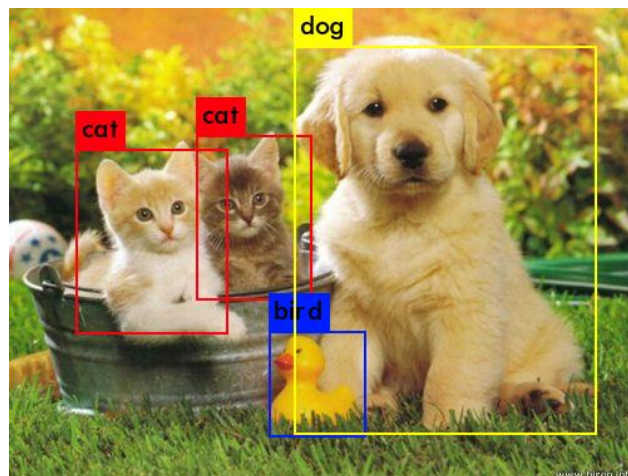


**Рисунок 1.3** – Приклад детекції та класифікації зображень<sup>2</sup>

3. Сегментація зображень полягає в розбитті зображення на кілька сегментів або класифікації кожного пікселя зображення (рис.1.4) [4]. Кожен піксель отримує мітку, що відповідає певному класу об'єкта або фону. Основна складність полягає в забезпеченні

<sup>2</sup> URL: <https://ambolt.io/en/image-classification-and-object-detection/>

високої точності при розмежуванні об'єктів з подібними ознаками.

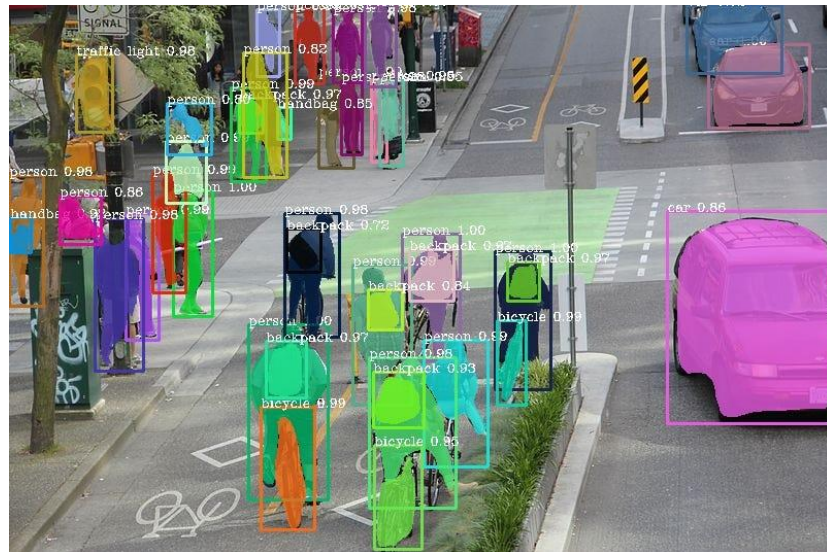


**Рисунок 1.4** – Приклад сегментації зображень [4]

4. Відстеження об'єктів спрямоване на визначення положення об'єкта на кожному кадрі відео, забезпечуючи його безперервне відстеження (рис.1.5). Використовуються такі методи, як Kalman Filters, Mean-Shift та нейронні мережі типу RNN. Ці методи враховують як поточне положення об'єкта, так і його попередні положення для прогнозування наступного стану. Застосування нейронних мереж, таких як LSTM, дозволяє ефективно моделювати часові залежності [5].
5. Відновлення зображень полягає в поліпшенні якості або відновленні зображень, які були пошкоджені або мають низьку якість (рис.1.6). Це включає завдання, такі як видалення шуму, підвищення роздільної здатності та відновлення зображень із пошкоджених ділянок. Використовуються нейронні мережі, такі як автокодувальники та GAN.

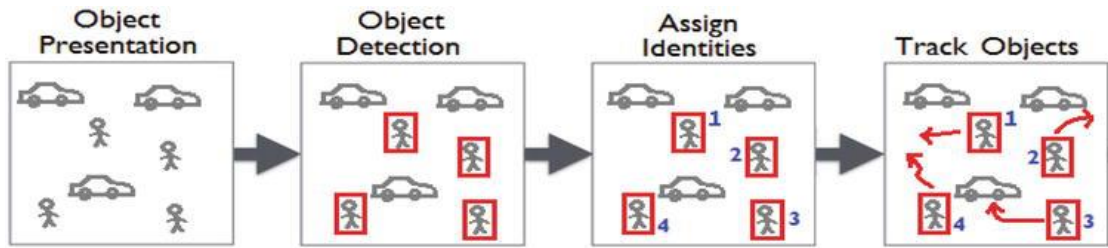


**Рисунок 1.5** – Ілюстрація роботи відстеження об’єктів за рахунок знаходження об’єктів в сцені та положення в наступному кадрі [5]

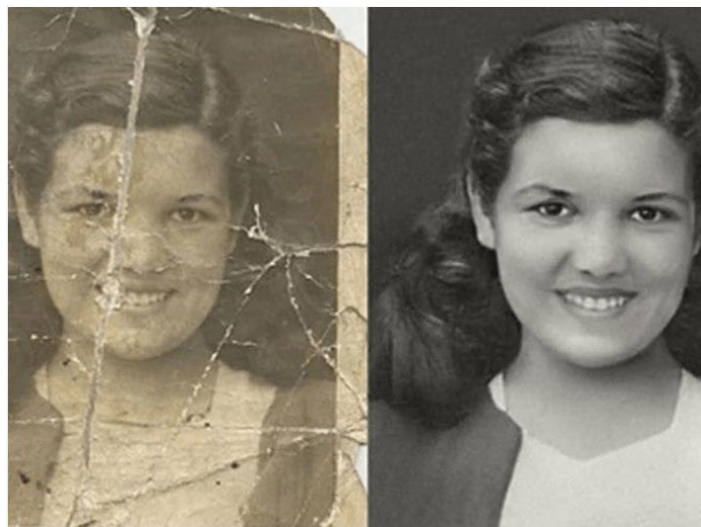


**Рисунок 1.6** – Демонстрація відновленого зображення за допомогою GAN<sup>3</sup>

6. Генерація зображень полягає у створенні нових зображень на основі навчання з великої кількості прикладів (рис.1.7). Основні методи включають використання GAN [6] та VAE [7]. GAN складається з двох нейронних мереж: генератора, який створює зображення, і дискримінатора, який намагається відрізнити справжні зображення від згенерованих. Генератор навчається обманювати дискримінатор, покращуючи якість згенерованих зображень.

<sup>3</sup> URL: <https://adelaidetechguy.com.au/restore-old-photos-using-ai/>



**Рисунок 1.7 – Фотографії людей та згенеровані зображення<sup>4</sup>**

### **1.3 Розпізнавання зображень**

Першочерговим завданням розпізнавання кадру з відео є виділення класів ознак зображення, які його характеризують. До класів ознак відносять:

- геометричні ознаки, що відображають геометричні параметри об'єкта, такі як форма, розміри та пропорції;
- текстурні ознаки, що відображають текстурні властивості, такі як шорсткість або гладкість поверхні;
- колірні ознаки, що відображають колірні характеристики об'єкта або його частин.

Для ознак класів існують методи по їх визначенню, які перелічено далі.

---

<sup>4</sup> URL: <https://www.tidio.com/blog/ai-test/>

1. Методи на основі зовнішнього вигляду, включають в себе модель на основі частин та метод на основі ознак. Метод на основі частин передбачає розділення об'єкту на менші складові, кожна з яких аналізується окремо, наприклад, транспортний засіб може бути розділений на передню, задню та бокові частини. Метод на основі ознак використовує специфічні ознаки для опису зовнішнього вигляду об'єкта, які роблять його унікальним, наприклад, форма та колір.
2. Методи на основі руху, які включають метод різниці кадрів та метод віднімання фону. Метод різниці кадрів базується на обчисленні різниці між послідовними кадрами для виявлення рухомих об'єктів. Зміни в пікселях можуть свідчити про наявність руху. Метод віднімання фону відокремлює об'єкти переднього плану від фону, який формується шляхом усереднення послідовності кадрів. Метод ефективний при стабільному фоні, але може бути проблематичним при динамічних змінах фону.

Нейронні мережі, особливо згорткові нейронні мережі та їхні модифікації, демонструють високу ефективність у задачах розпізнавання зображень, проте вимагають значних обчислювальних ресурсів. Вони можуть обробляти великі обсяги даних і забезпечувати високу точність, що робить їх ідеальними для використання в системах реального часу.

#### **1.4 Проблематика обраної теми**

Автоматизація розпізнавання дефектів дорожнього покриття є важливою та актуальною задачею, проте має свої проблеми та виклики, які необхідно враховувати під час розробки та впровадження цієї технології.

По-перше, існує різноманіття типів доріг, кожна з яких має свої специфічні характеристики та види дефектів. Асфальт, бетон, гравій та інші типи покриття можуть вимагати різних підходів. Це ускладнює процес створення універсальної системи, здатної ефективно працювати з усіма видами об'єктів.

По-друге, значним викликом є впровадження автоматизованих систем у практику управління дорожньою інфраструктурою, особливо в умовах високого рівня корупції. Корумпованість системи може стати перешкодою для впровадження нових технологій, оскільки це може загрожувати інтересам певних груп, які отримують вигоду від існуючих процедур.

Ще однією актуальною проблемою є вибір методів та технологій для реалізації системи розпізнавання дефектів. Для розробки рішення цієї задачі є великий арсенал методів: згортокві нейронні мережі наприклад, ResNet, VGG, об'єктні детектори такі як YOLO, SDD, моделі для сегментації зображень, наприклад, U-Net або Mask R-CNN, моделі для виявлення аномалій, автоенкодери або GAN тощо.

Також важливою є проблема навчання системи на реальних даних. Необхідно зібрати великий обсяг якісних зображень дефектів різних типів дорожнього покриття для тренування нейронної мережі. Це вимагає значних зусиль з боку дослідників та дорожніх служб, а також забезпечення правильного маркування даних для досягнення максимальної точності розпізнавання.

В умовах війни питання оптимізації використання бюджетних коштів стає особливо актуальним. Впровадження автоматизованої системи розпізнавання дефектів дорожнього покриття дозволить значно скоротити витрати на обслуговування доріг, оскільки ця технологія забезпечує оперативне та точне виявлення пошкоджень. Це, у свою чергу, дозволить ефективніше планувати ремонтні роботи та мінімізувати витрати на них.

Загалом, незважаючи на велику кількість викликів, успішне вирішення зазначених проблем може значно підвищити ефективність управління

дорожньою інфраструктурою, зменшити витрати на обслуговування та покращити безпеку дорожнього руху. Це робить розробку автоматизованої системи розпізнавання дефектів дорожнього покриття не лише важливим науковим завданням, але й соціально значущим проектом, особливо в умовах обмежених ресурсів під час війни.

### **1.5 Постановка задачі**

Основною задачею є проектування та розробка програмного забезпечення здатного оптимізувати один з процесів у сфері дорожньо-будівельної галузі. Задача полягає у розпізнаванні об'єктів та класифікацію, зокрема ідентифікацію дефектів дорожнього покриття. Отже, слід зосередитись на методах класифікації.

Проаналізувавши методи розпізнавання, вирішено використовувати нейронні мережі для детекції об'єктів. Для навчання нашої нейромережі оберемо підхід навчання зі вчителем. Тренування моделі буде здійснюватися на заздалегідь підготовленому наборі даних, що містить фотографії з попередньо промаркованими об'єктами. Кожен промаркований об'єкт буде визначений відповідною міткою, яка нас цікавить.

Збереження результатів тренування моделі відбуватиметься локально для можливості подальшої аналітики, а також одразу передаватиметься на хмарну платформу ClearML задля збору додаткової інформації процесу.

Початковий план полягає у створенні програми, здатної розпізнавати об'єкти незалежно від особливостей дорожнього покриття. Програма повинна класифікувати об'єкти для визначення стану дороги, оцінюючи ступінь зношеності полотна. Це дозволить точно оцінити залишковий ресурс дороги та визначити необхідність фінансових вкладень для її ремонту чи обслуговування.

## **Висновки до розділу 1**

У першому розділі було детально розглянуто питання, пов'язані з автоматичним розпізнаванням дефектів дорожнього покриття. Спершу було проаналізовано актуальність даної технології, яка полягає у підвищенні ефективності управління дорожньою інфраструктурою та зменшенні витрат на її обслуговування. Потім було проведено огляд наявних методів аналізу стану дорожнього покриття. Було розглянуто основні напрямки аналізу та обробки зображень у комп'ютерному зорі, а також виконано більш деталізований огляд напряму розпізнавання зображень. Також, в результаті було описано проблематику обраної області та виконано постановку задачі, в якій описано основні виклики до розробки програмного забезпечення.

Таким чином, перший розділ заклав первинну теоретичну основу для подальшого дослідження і розробки системи.

## РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

### 2.1 Штучні нейронні мережі

Щодо штучних нейронних мереж, то в останні десятиліття вони стали предметом інтенсивного вивчення і широкого застосування у різних галузях. Штучні нейронні мережі (ШНМ) є математичними моделями, що імітують біологічні нейронні мережі людського мозку. Вони складаються з великої кількості простих взаємопов'язаних елементів – нейронів (рис.2.1), які працюють спільно для вирішення складних задач [8]. Завдяки здатності до навчання і адаптації, нейронні мережі ефективно застосовуються для розпізнавання образів, прогнозування, класифікації та інших завдань.

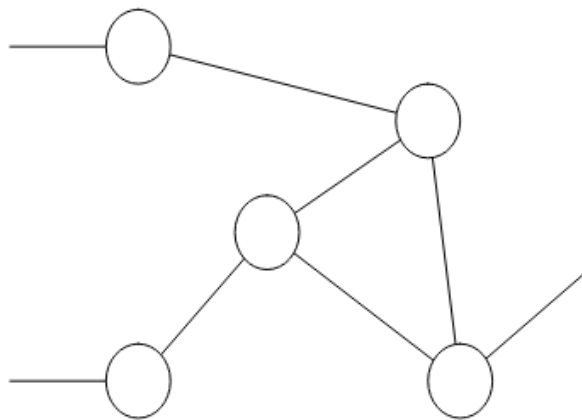


**Рисунок 2.1** – Приклад ШНМ з довільною структурою [8]

Нейрон приймає на вхід значення, виконує зважене сумування, після чого на вихід подається сума подана до вибраної функції активації. Вихід нейрону подається на вхід до наступного нейрону або вважається виходом моделі, спираючись на результати моделі.

Однією з найпоширеніших архітектур є багатошаровий перцептрон (MLP), який складається з вхідного, одного або більше прихованих і вихідного шарів нейронів. Кожен нейрон у шарі з'єднаний з усіма нейронами наступного шару. MLP використовуються для задач класифікації та регресії і знаходять застосування у розпізнаванні образів та прогнозуванні даних.

Згорткові нейронні мережі (CNN) особливо ефективні для обробки зображень та відео. Вони містять згорткові шари, які виділяють важливі особливості у вхідних даних, такі як контури, текстури та інші деталі. Це робить CNN незамінними у таких задачах, як розпізнавання облич, аналіз медичних зображень та автономне керування транспортними засобами.

Рекурентні нейронні мережі (RNN) застосовуються для обробки послідовностей даних, таких як текст чи часові ряди. Вони здатні зберігати інформацію про попередні етапи обчислень, що робить їх корисними для задач прогнозування, аналізу часових рядів та обробки природної мови. Наприклад, у задачах машинного перекладу RNN враховують контекст попередніх слів для точнішого перекладу.

Існують також інші архітектури ШНМ, розроблені для конкретних застосувань. Наприклад, довготривалі короткострокові пам'яті (LSTM) та мережі на основі гейтованих рекурентних одиниць (GRU) є вдосконаленими версіями RNN, що краще зберігають інформацію на тривалих часових відрізках. Автокодери використовуються для зменшення розмірності даних та видалення особливостей, тоді як генеративно-змагальні мережі (GAN) здатні створювати нові, реалістичні дані, такі як зображення та звуки.

## 2.2 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) [9, 10] стали однією з найпопулярніших і найефективніших архітектур для обробки зображень та

інших структурованих даних. Цей підхід дозволяє автоматично витягати релевантні ознаки з даних, що значно покращує якість моделі в задачах комп'ютерного зору, таких як класифікація, детекція та сегментація об'єктів. Розглянемо детальніше основні компоненти, архітектуру, функції активації CNN. Загальна будова згорткової нейронної мережі для задачі класифікації наведена на рисунку 2.2

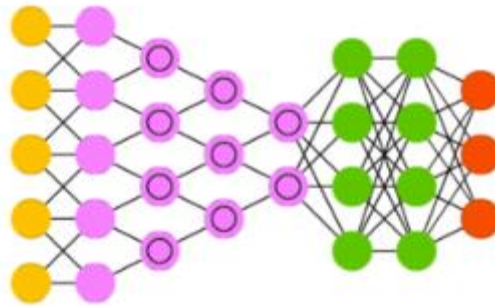


**Рисунок 2.2** – Схематичне представлення загорткових нейронних мереж<sup>5</sup>

До основних компонентів згорткової нейронної мережі належать: згортковий шар, підсумковий шар, повнозв'язний шар.

1. Згортковий шар є основним елементом CNN і виконує операцію згортки над вхідними даними (рис.2.3). Згортка дозволяє витягати локальні патерни з зображення, що є важливим для розпізнавання структурних елементів. До згорткового набору належить набір фільтрів з вагами для ділянки зображення, що навчаються. У випадку двовимірних даних (зображень) згортка описується за формулою (2.1).

$$(F * K)(i, j) = \sum_m \sum_n F(i - m, j - n) \cdot K(m, n) \quad (2.1)$$

де  $F$  – вхідне зображення,  $K$  – фільтр.

<sup>5</sup> URL: <https://annsplatform.com/nn-types.html>

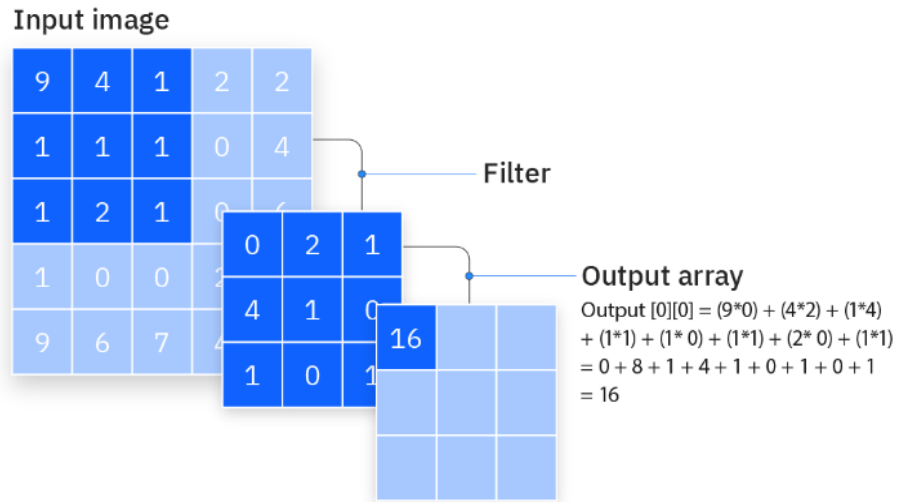


**Рисунок 2.3** – Ілюстрація застосування згортки одного фільтру до сегменту зображення [10]

2. Підсумкові шари (pooling layers) зменшують розмірність даних, зберігаючи при цьому найважливішу інформацію. Це досягається за рахунок об'єднання (підсумовування) значень в певних областях. Найпоширеніший тип підсумкових шарів - максимальне підсумовування (max pooling), яке обирає максимальне значення в кожному вікні (формула 2.2).

$$y[i, j] = \max\{x[m, n] | (m, n) \in Region(i, j)\} \quad (2.2)$$

де  $Region(i, j)$  – область, з якої береться максимальне значення.

3. Повнозв'язні шари (FC) використовуються на завершальному етапі CNN для класифікації витягнутих ознак. Вихід попереднього шару перетворюється у вектор і подається на вхід FC-шару. Цей шар обчислює лінійну комбінацію вхідних даних, як показано у формулі (2.3).

$$y = Wx + b, \quad (2.3)$$

де  $W$  – матриця ваги;

$x$  – вхідні дані;

$b$  – вектор зміщень.

Стандартна архітектура CNN складається з чергування згорткових і підсумкових шарів, за якими слідує один або кілька повнозв'язних шарів. Розглянемо детальніше кожен з цих компонентів у контексті типової архітектури CNN.

Важливим аспектом згорткових шарів є параметри ядра (фільтра), яке переміщується по зображенню і обчислює зважену суму пікселів у межах вікна фільтра. Результат згортки для кожного місця на зображенні утворює карту ознак (feature map).

Підсумкові шари зменшують розмір карт ознак, зберігаючи найважливішу інформацію. Наприклад, max pooling з вікном  $2 \times 2$  з кроком 2 зменшить розмір карти ознак вдвічі по кожній осі. Цей процес допомагає зменшити обчислювальну складність моделі та запобігає перенавчанню, зберігаючи лише найважливіші ознаки.

На завершальному етапі CNN використовуються повнозв'язні (FC) шари для остаточної класифікації. Вихід останнього згорткового або підсумкового шару перетворюється у вектор і подається на вхід FC-шару. Наприклад, для класифікації на 10 класів вихід останнього FC-шару буде вектором розміром 10, де кожен елемент вектора відповідає ймовірності належності до певного класу.

Для нелінійності між шарами використовуються функції активації [11]. Найпоширенішою функцією активації в CNN є ReLU (2.4), що додає нелінійність до моделі та дозволяє їй вивчати більш складні патерни. Інші функції активації, такі як sigmoid (2.5) або tanh (2.6), також можуть

використовуватися, але ReLU є більш популярною завдяки своїй ефективності в уникненні проблеми зникнення градієнта.

$$f(x) = \max(0, x), \quad (2.4)$$

$$f(x) = \frac{1}{1+e^{-x}}, \quad (2.5)$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (2.6)$$

Для підвищення продуктивності CNN існує кілька методів і технік [10]:

1. Регуляризація, що допомагає запобігти перенавчанню моделі.

Найпоширенішими методами регуляризації є:

- dropout: випадкове відключення нейронів під час навчання, що допомагає запобігти перенавчанню.
- L2-регуляризація: додає штраф за великі ваги до функції втрат, зменшуючи їх величину.

2. Аугментація даних, що включає створення додаткових тренувальних прикладів за рахунок модифікацій існуючих, таких як обертання, зсув, зміна яскравості тощо. Це допомагає моделі бути стійкішою до варіацій вхідних даних.

Згорткові нейронні мережі є потужним інструментом для обробки зображень та інших структурованих даних завдяки своїй здатності автоматично витягати релевантні ознаки з вхідних даних. Використання згорткових і підсумкових шарів дозволяє зменшувати розмірність даних, зберігаючи важливу інформацію, що робить CNN ефективними для широкого спектру задач, від класифікації зображень до детекції об'єктів та сегментації.

## 2.3 Постановка задачі розпізнавання

Існує кілька важливих метрик, які використовуються для оцінки ефективності моделей згорткових нейронних мереж (CNN). Основні з них включають частоту кадрів за секунду (FPS) та точність. Крім того, для визначення вимог до обчислювальних ресурсів використовуються такі параметри, як кількість операцій з плаваючою комою за секунду (FLOPs) та кількість параметрів моделі [12].

1. Кількість параметрів моделі визначає, скільки змінних необхідно налаштувати під час навчання моделі. Ці параметри включають ваги та зсуви в нейронних мережах. Кількість параметрів є індикатором складності моделі та використовується для оцінки обсягу пам'яті, необхідного для її зберігання. Більша кількість параметрів означає більше пам'яті, необхідної для зберігання та навчання моделі.
2. FLOPs вимірює обчислювальну складність моделі, вказуючи на кількість операцій з плаваючою точкою, які модель виконує за секунду. Це є важливим показником, який використовується для порівняння ефективності різних моделей. Вищі значення FLOPs свідчать про більшу обчислювальну складність моделі та вимагають більше обчислювальних ресурсів для виконання.
3. FPS показує, скільки кадрів в секунду може обробити модель при виявленні об'єктів на зображеннях. Це особливо важливо у випадках, коли система повинна працювати в реальному часі, наприклад, для виявлення транспортних засобів. Для стабільної роботи системи бажане значення FPS повинно бути в межах 35-40 кадрів на секунду, що забезпечує достатню плавність відео. Досягнення вищого FPS є бажаним, але вимагає оптимізації моделі та використання потужнішого обладнання.

При виборі та налаштуванні моделей CNN важливо враховувати кілька ключових метрик, таких як точність, FPS, FLOPs і кількість параметрів моделі. Ці метрики допомагають оцінити ефективність та вимоги до обчислювальних ресурсів моделей. Оцінка цих параметрів дозволяє збалансувати точність та продуктивність моделей, забезпечуючи їх ефективне використання в реальних додатках.

Для оцінки точності розпізнавання моделі використовується показник середньої точності, або mAP. Значення mAP визначається як середнє значення точності розпізнавання (AP) для всіх класів, які може ідентифікувати модель. Точність розпізнавання для кожного окремого класу (AP) розраховується на основі точності моделі щодо цього класу (2.7).

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i, \quad (2.7)$$

де  $N$  – загальна кількість класів, які розпізнаються моделлю;

$AP_i$  – показник AP для класу  $i$ .

Показник mAP є важливим інструментом для оцінки ефективності моделей розпізнавання об'єктів, оскільки дозволяє врахувати точність розпізнавання для всіх класів, які присутні в наборі даних. Це забезпечує більш збалансовану та об'єктивну оцінку роботи моделі, на відміну від простого вимірювання точності для окремих класів. Для обчислення Average Precision (AP) потрібні попередньо визначені метрики IoU, precision, recall, PR-крива.

IoU – це метрика, яка використовується для оцінки точності об'єктних детекторів на конкретному наборі даних. IoU вимірює обсяг перетину між передбаченим обмежуючим прямокутником і фактичним обмежуючим прямокутником, поділений на обсяг їх об'єднання (2.8).

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2}, \quad (2.8)$$

де  $B_1, B_2$  – відповідно передбачуваний і фактичний обмежуючий прямокутник.

Значення IoU варіюється від 0 до 1, де 0 означає відсутність збігу, а 1 – повний збіг між прогнозованою та реальною областями. Часто використовується поріг  $\alpha = 0.5$  для оцінки точності моделі (рис.2.4).

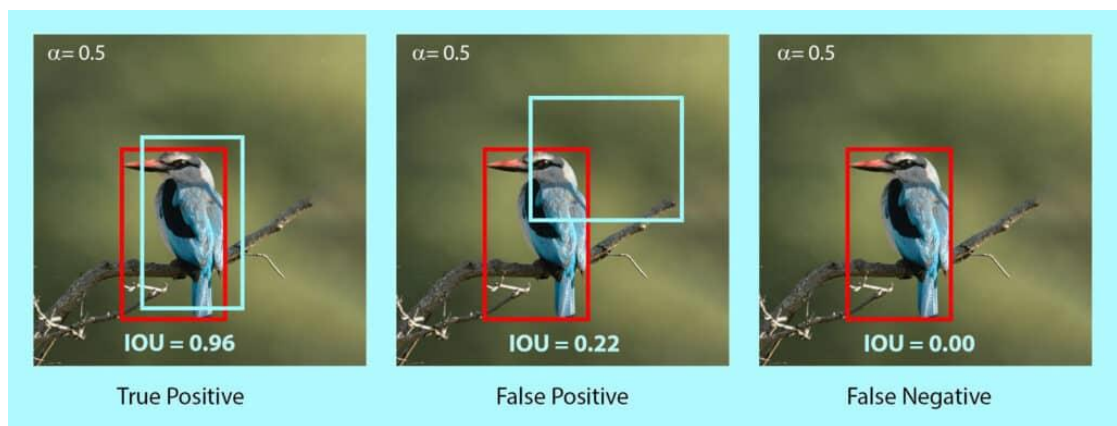


**Рисунок 2.4** – Точність детекції об'єктів на основі IoU

Якщо значення IoU вище встановленого порогу, прогнозування вважається правильно позитивним (True Positive, TP). Якщо IoU нижче порогу або прогнозовано неправильний клас, прогнозування вважається хибно позитивним (False Positive, FP). У випадку, коли модель не робить прогнозу для об'єкта, це вважається хибно негативним (False Negative, FN). Показники TP, FP та FN використовуються для розрахунку точності та повноти за відомими формулами, TN (True Negative) не використовується.

Точність та повнота використовуються для побудови PR-кривої, яка демонструє здатність моделі підтримувати високі показники класифікації позитивних випадків при збереженні високого рівня точності.

Для обчислення AP, розраховується площа під PR-кривою для кожного класу. Приклад PR-кривої зображено на рис.2.5. Цей підхід використовується в метриці PASCAL VOC 2010. Існують різні підходи до обчислення AP, які відрізняються методиками розрахунку, а також використовують спеціальні набори даних для перевірки якості моделей.

Перелічимо основні набори даних, які використовуються для оцінки якості моделі.

1. PASCAL VOC 2005 використовував площу під ROC-кривою замість PR-кривої.
2. PASCAL VOC 2007 використовував одинадцяти точкову інтерполяцію для зменшення впливу коливань PR-кривої на результат. Повнота розділяється на 11 рівнів, і для кожного рівня знаходиться максимальна точність серед усіх порогових значень, що перевищують чи дорівнюють поточному рівню повноти. AP обчислюється як середнє значення 11 інтерпольованих точок.
3. MS COCO використовує 101-бальну інтерполяційну версію AP. Замість одного порогового значення для IoU використовуються 10 порогових значень, які варіюються від 0.5 до 0.95 з кроком 0.05. Середнє значення AP з усіх цих порогів визначає mAP, який використовується для порівняння моделей.

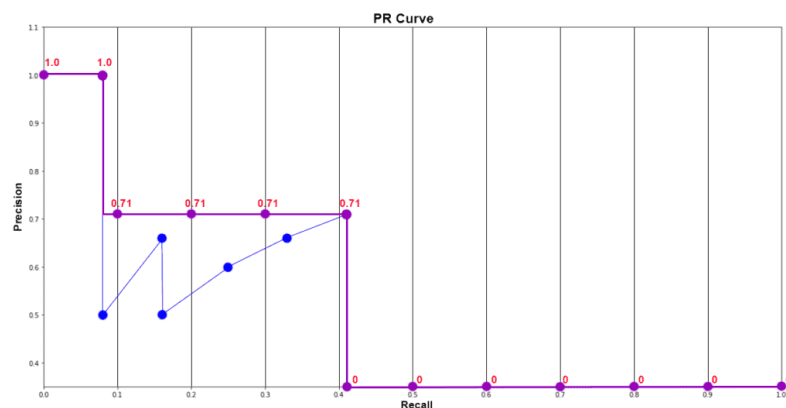


**Рисунок 2.5** – Приклад PR-кривої та методу точкової інтерполяції<sup>6</sup>

<sup>6</sup> URL: <https://qph.cf2.quoracdn.net/main-qimg-bd1357c543dba9e3f5d171855cdb9e2c>

## 2.4 Моделі розпізнавання об'єктів

Існує кілька моделей для розпізнавання об'єктів у відео, кожна з яких має свої унікальні характеристики та підходи до вирішення завдань комп'ютерного зору. Перелічимо кілька основних моделей.

1. YOLO (You Only Look Once) [13] з наступними характеристиками:
  - архітектура використовує єдину конволюційну нейронну мережу (CNN), яка прогнозує обмежувальні рамки та ймовірності класів для декількох об'єктів одночасно в одному проході через мережу;
  - YOLO відома своєю високою швидкістю обробки, оскільки обробляє все зображення як єдине ціле, що дозволяє досягати продуктивності в режимі реального часу;
  - хоча YOLO поступається деяким іншим моделям у точності для дрібних об'єктів або об'єктів у складних сценах, її продуктивність залишається на високому рівні для більшості практичних застосувань.
2. Faster R-CNN (Region-based Convolutional Neural Networks) [14]:
  - архітектура складається з двох основних компонентів: регіональної пропозиційної мережі (RPN), яка генерує регіональні пропозиції, і мережі для класифікації та регресії обмежувальних рамок (рис.2.6);
  - Faster R-CNN забезпечує високу точність розпізнавання об'єктів за рахунок детального аналізу кожної регіональної пропозиції;

- в порівнянні з YOLO, Faster R-CNN є повільнішою, що може бути проблемою для застосувань, де необхідна обробка в режимі реального часу.

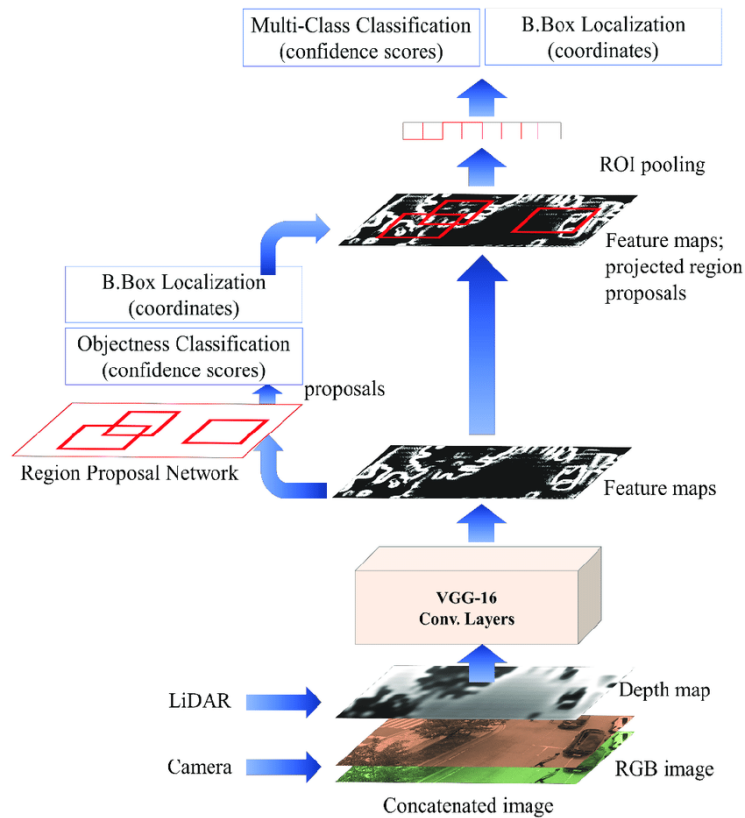


**Рисунок 2.6** – Візуалізація архітектури R-CNN [14]

### 3. SSD (Single-Shot Detector) [15]:

- архітектура використовує один прохід через CNN для генерування множинних обмежувальних рамок та їх класифікації, забезпечуючи баланс між швидкістю та точністю (рис.2.7);
- SSD демонструє високий рівень точності, особливо при розпізнаванні великих об'єктів;
- SSD є швидшою за Faster R-CNN, але може поступатися YOLO в умовах обмежених обчислювальних ресурсів.

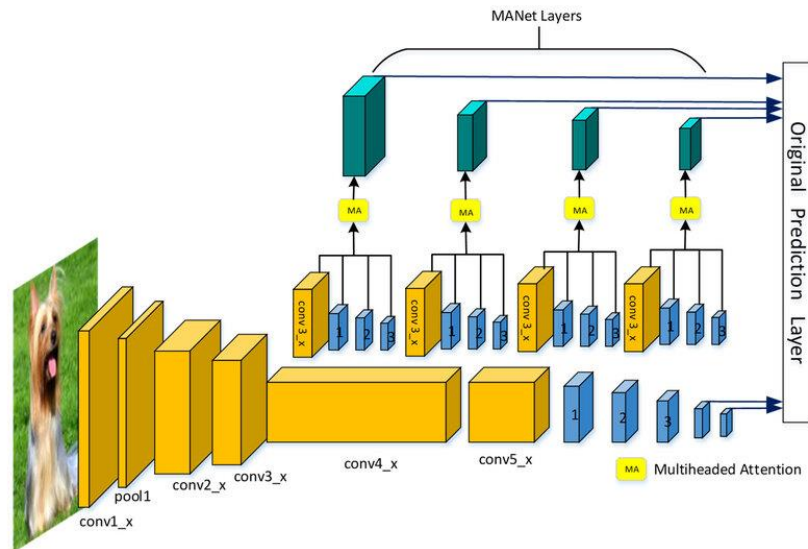


**Рисунок 2.7** – Демонстрація архітектури SSD [15]

4. RetinaNet [16]:

- архітектура (рис.2.8) використовує Focal Loss для обробки дисбалансу класів, що покращує точність розпізнавання об'єктів, особливо для рідкісних або дрібних об'єктів;
- RetinaNet демонструє високу точність завдяки використанню складних методів навчання;
- швидкість обробки нижча в порівнянні з YOLO і SSD, але модель залишається практичною для багатьох застосувань, де критична точність.

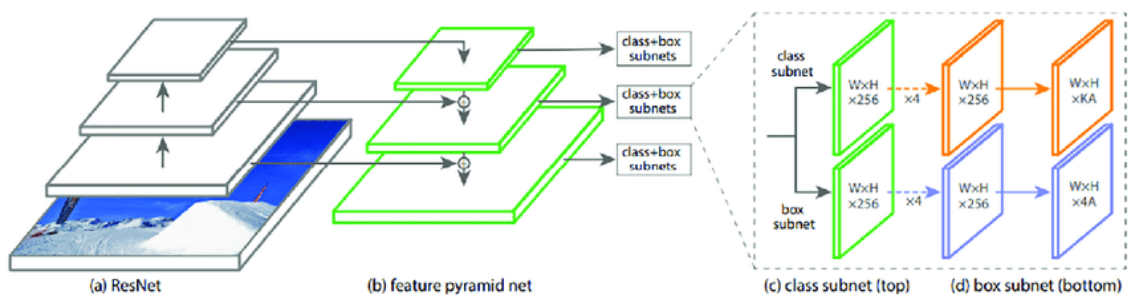


**Рисунок 2.8** – Візуалізація архітектури RetinaNet [16]

## 5. EfficientDet [17]:

- Архітектура (рис.2.9) використовує оптимізовану архітектуру EfficientNet як базову мережу, що забезпечує високу продуктивність з меншими обчислювальними витратами;
- EfficientDet забезпечує високу точність розпізнавання завдяки використанню складних технік масштабування;
- швидкість обробки висока, що робить модель придатною для застосувань на мобільних та вбудованих пристроях.

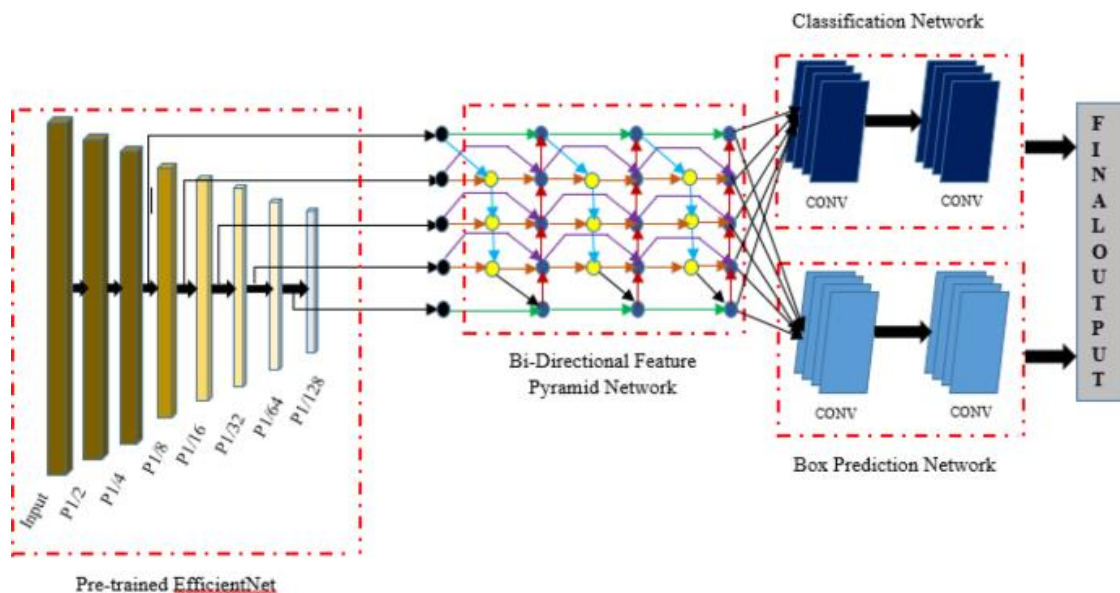


**Рисунок 2.9** - Візуалізація архітектури EfficientDet [17]

## Висновки до розділу 2

У другому розділі було розглянуто теоретичні основи нейронних мереж та їх застосування в комп'ютерному зорі. Починаючи з пояснення

базових концепцій нейронних мереж, зокрема їх будови. Також було детально описано технологію згорткових нейронних мереж: застосування, будову, способи регуляризації для оптимізації навчання. Було описано постанову задачі розпізнавання, а також перелічено основні сучасні моделі для розпізнавання об'єктів.

Таким чином, другий розділ надає ґрунтовне теоретичне підґрунтя для розуміння сучасних методів аналізу зображень за допомогою нейронних мереж, акцентуючи увагу на їх практичному застосуванні. Викладені концепції та методи є основою для розробки систем, які можуть автоматично і з високою точністю виявляти дефекти на дорожньому покритті.

## **РОЗДІЛ 3 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОЇ ЧАСТИНИ**

### **3.1 Аналіз вхідних наборів даних та підготовка до обробки нейронною мережею**

Головною проблемою, що постала при аналізі вхідних наборів даних, була різноманітність різних типів доріг, а також типів дефектів. Окрім реальних зображень, зроблених в Україні, були додатково залучені дані з різних міжнародних джерел. Це дозволило зібрати багатий і різноманітний набір даних, що охоплює різні типи дорожнього покриття та дефектів з усього світу.

Загалом набір даних складається з 7000 зображень для навчання та близько 300 зображень для перевірки. Для забезпечення різноманітності та якості даних використовувалися перелічені далі джерела.

1. Roboflow набір даних вибоїн, що наданий платформою Roboflow, містить велику кількість зображень дефектів доріг, зокрема ям, тріщин та інших пошкоджень. Ці дані були попередньо розмічені та готові до використання для навчання моделей.
2. Набір даних з публікацій дослідницьких робіт, дані якого взяті з опублікованих дослідницьких робіт, надали додатковий контекст та різноманітність зображень доріг з різних регіонів світу. Вони містили як високоякісні фотографії, так і супутникові знімки, що дозволило розширити спектр аналізованих зображень.
3. Зображення, отримані з відеороликів YouTube, Google Maps (рис 3.1-3.2) і розмічені вручну.
4. Зображення з набору даних RDD2022, великої колекції зображень доріг, зібраних з різних країн світу. Набір містить зображення різних типів дорожніх покриттів та дефектів, що дозволило значно розширити базу навчальних даних і поліпшити роботу моделей.

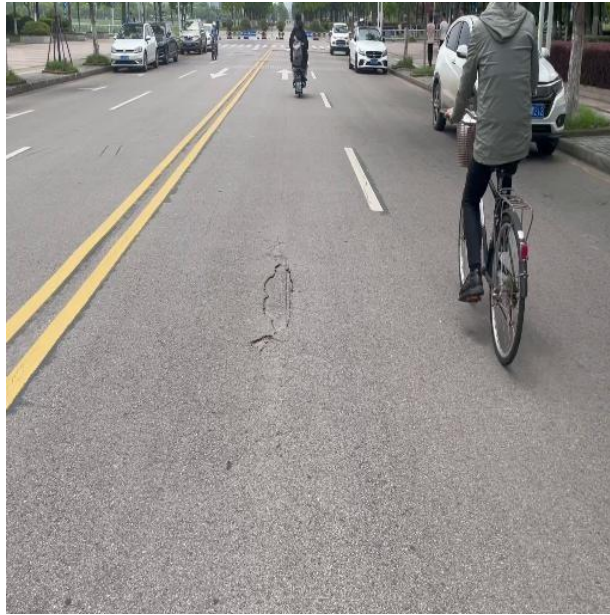


**Рисунок 3.1** – Приклад фотографії з тренувального набору даних зробленої в Китаї



**Рисунок 3.2** – Приклад фотографії взятої з Google Maps

Також важливим питанням став вибір інструменту для розмітки даних. Серед багатьох інструментів був обраний CVAT.

CVAT (Computer Vision Annotation Tool) — це інструмент з відкритим кодом [18], розроблений Intel, який забезпечує зручний та потужний інтерфейс для розмітки даних (рис.3.3). Далі перелічені основні особливості CVAT.

1. Підтримка різних типів анотацій: полігони, прямокутники, точки, лінії.
2. Можливість колективної роботи.
3. Легке налаштування та інтеграція з іншими інструментами.

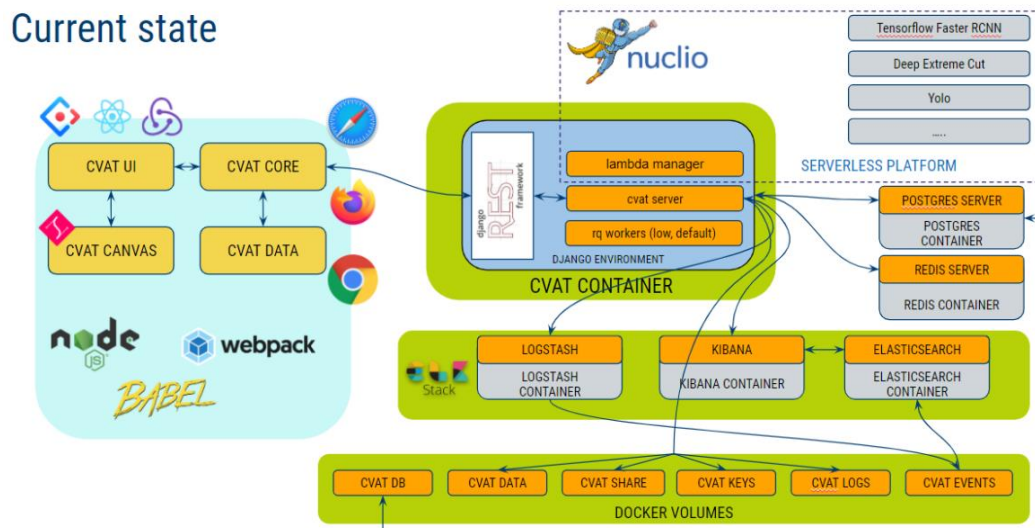


**Рисунок 3.3** – Візуалізація архітектури роботи CVAT<sup>7</sup>

Далі перелічимо причини вибору CVAT.

1. Гнучкість і масштабованість: CVAT підтримує розмітку різних типів об'єктів, включаючи полігони, точки, лінії та прямокутники, що дозволяє розмічувати ями та інші дефекти дорожнього покриття різних форм і розмірів.
2. Підтримка різних форматів даних: Інструмент підтримує велику кількість форматів зображень і відео, що полегшує імпорт даних з різних джерел та типів доріг.

<sup>7</sup> URL: <https://github.com/cvat-ai/cvat/issues/2372>

3. Інтуїтивно зрозумілий інтерфейс: Зручний інтерфейс користувача дозволяє легко навчити нових розмічувачів і забезпечує ефективний процес розмітки навіть для великих обсягів даних.
4. Колективна робота: CVAT дозволяє одночасну роботу декількох користувачів над одним проектом, що значно прискорює процес розмітки.

Як результат, аналіз даних був проведений за наступним алгоритмом.

1. Збір даних: зображення та відео доріг були зібрані з різних джерел, включаючи урядові бази даних, дослідницькі проекти та громадські ініціативи. Дані охоплювали дороги з різних країн, що дозволило врахувати регіональні особливості дорожнього покриття та дефектів.
2. Імпорт даних до CVAT: Зібрані дані були імпортовані до CVAT, де вони були організовані в проекти, що відповідали конкретним типам доріг або регіонам.
3. Процес розмітки, для чого використовувалися полігони, що дозволяло точно визначити межі дефектів неправильної форми. Прямокутники використовувалися для обмеження областей інтересу. Використовувались додаткові атрибути, такі як тип дефекту (яма, тріщина тощо), розмір дефекту та глибина, що були додані до розмічених об'єктів для збагачення навчальних даних.
4. Перевірка і валідація: після розмітки дані проходили етап валідації для забезпечення якості. Перевірка включала перегляд розміток незалежними експертами та корекцію помилок.

### 3.2 Опис обраної архітектури нейронної мережі

Для виконання поставленої задачі, серед перелічених у підрозділі 2.4, було обрано модель YOLO. Існує багато версій YOLO, всі вони мають різні особливості, однак в рамках цієї роботи було використано одну із останніх версій YOLOv8 (рис.3.4).

Перелічимо основні причини вибору YOLO:

1. Обчислювальна ефективність: YOLO побудована так, що вона може працювати на менш потужних апаратних засобах. Використання єдиної CNN для обробки всього зображення знижує обчислювальні витрати, що робить її придатною для застосувань, де ресурси обмежені, наприклад, на мобільних пристроях або вбудованих системах.
2. YOLO забезпечує компроміс між швидкістю та точністю, що робить її придатною для багатьох практичних застосувань. Наприклад, у системах відеоспостереження або автономних транспортних засобах важливо мати високу швидкість обробки з достатньою точністю.
3. YOLO легко адаптується до різних завдань і наборів даних. Вона може бути використана для розпізнавання об'єктів у різних умовах і для широкого спектра об'єктів. Архітектура YOLO легко масштабована, що дозволяє підвищувати продуктивність за рахунок додавання додаткових шарів або використання більш потужних базових моделей.
4. YOLO має значну спільноту користувачів і розробників, що забезпечує доступ до багатьох ресурсів, документації та прикладів використання. Це значно полегшує процес навчання, впровадження та налаштування моделі для специфічних завдань.

Перевагами YOLOv8 в порівнянні з попередніми моделями є включення найсучасніших технологій та оптимізацій, що забезпечують найвищу точність і швидкість, а також оптимізації для роботи на різних апаратних платформах та в різних умовах використання.

У січні 2023 року компанія Ultralytics випустила YOLOv8, яка на той час був передовою моделлю. З точки зору архітектури, YOLOv8 схожа на YOLOv5 з деякими незначними змінами, такими як використання модуля c2f замість CSPNet, який є лише варіантом CSPNet (CSPNet з двома послідовними згортковими мережами). Модель YOLOv8 була створена без використання якорів, на основі YOLOv5. У YOLOv8 також було введено розділені голови для незалежної обробки задач визначення об'єктності, класифікації та прогнозування межових рамок. Вони використовували сигмоїдний шар як останній шар для прогнозування об'єктності та softmax для класифікації.

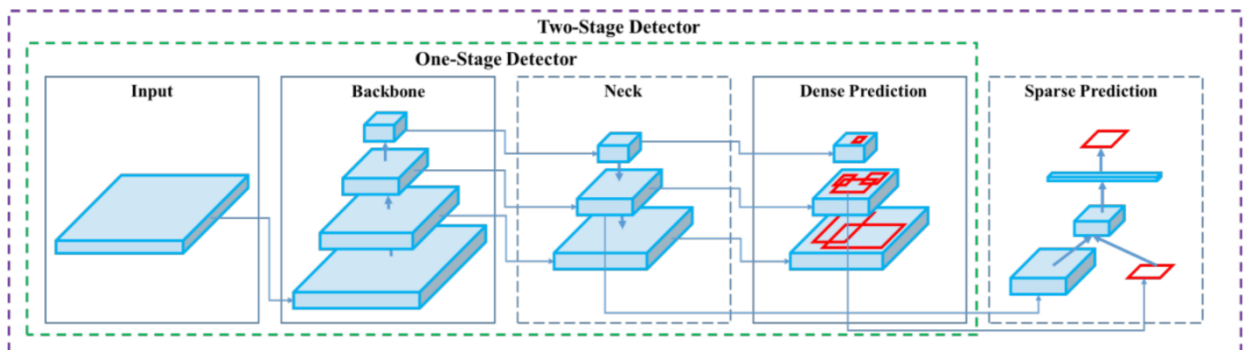


**Рисунок 3.4** – Загальна архітектура YOLOv8<sup>8</sup>

YOLOv8 використовує функції втрат CIoU та DFL для втрат межових рамок та бінарну крос-ентропію для втрат класифікації. Ці функції втрат покращили продуктивність виявлення об'єктів, особливо при роботі з меншими об'єктами. Було також введено модель YOLOv8-Seg для семантичної сегментації з мінімальними змінами до оригінальної моделі.

<sup>8</sup> URL: <https://machinelearningknowledge.ai/a-brief-history-of-yolo-object-detection-models/>

YOLO Nano, YOLO Small та YOLO Middle є різними масштабами моделей YOLO, призначених для різних застосувань. У випадку розпізнавання стану дорожнього покриття важливо вибрати YOLO Nano через його оптимальне співвідношення продуктивності до обчислювальних ресурсів. Ця модель підходить для пристроїв з обмеженими ресурсами, забезпечуючи швидке та точне виявлення дефектів дорожнього покриття в реальному часі.

### **3.3 Опис використаних інструментів розробки**

Для розробки програмного забезпечення для вирішення цієї задачі було обрано мову програмування Python. Основні переваги Python: висока продуктивність, широкий набір підтримуваних бібліотек та простота синтаксису. Python забезпечує швидке написання коду та зручність його читання і підтримки, що є критично важливим для ефективної розробки та подальшого обслуговування програмного забезпечення. Додатково, Python має велику спільноту розробників та широкий спектр ресурсів.

Також для роботи з Python було створено віртуальне середовище (рис.3.5). Створення віртуального середовища для роботи з Python має кілька важливих переваг. Воно дозволяє ізолювати залежності проекту, запобігаючи конфліктам між бібліотеками, які можуть мати різні версії. По-друге, це сприяє відтворюваності проектів, оскільки всі залежності визначені в одному середовищі та можуть бути легко відновлені. По-третє, віртуальні середовища спрощують управління проектами, дозволяючи використовувати різні версії Python та пакетів для різних проектів. Також це підвищує безпеку, оскільки ізольовані середовища запобігають випадковим змінам системних бібліотек.

```

PS C:\Users\Serhii> cd .\Desktop\
PS C:\Users\Serhii\Desktop> cd .\Project_Korniichuk\
PS C:\Users\Serhii\Desktop\Project_Korniichuk> python -m venv venv
>>
PS C:\Users\Serhii\Desktop\Project_Korniichuk>
PS C:\Users\Serhii\Desktop\Project_Korniichuk> venv\Scripts\activate
>> C:\Users\Serhii\Desktop\Project_Korniichuk>
(venv) PS C:\Users\Serhii\Desktop\Project_Korniichuk> cd .\pothole_dataset_v8\
(venv) PS C:\Users\Serhii\Desktop\Project_Korniichuk\pothole_dataset_v8> cd ..
(venv) PS C:\Users\Serhii\Desktop\Project_Korniichuk> pip install ultralytics
Collecting ultralytics
  Downloading ultralytics-8.2.27-py3-none-any.whl (779 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 779.6/779.6 kB 4.1 MB/s eta 0:00:00
Collecting thop>=0.1.1
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
Collecting requests>=2.23.0
  Downloading requests-2.32.3-py3-none-any.whl (64 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.9/64.9 kB ? eta 0:00:00
Collecting pyyaml>=5.3.1
  Downloading PyYAML-6.0.1-cp310-cp310-win_amd64.whl (145 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 145.3/145.3 kB 9.0 MB/s eta 0:00:00
Collecting tqdm>=4.64.0
  Downloading tqdm-4.66.4-py3-none-any.whl (78 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.3/78.3 kB 4.5 MB/s eta 0:00:00
Collecting psutil

```

**Рисунок 3.5** – Створення віртуального середовища та встановлення Ultralytics

Задля покращення роботи та досягнення низки інших переваг було прийнято рішення використовувати YOLOv8 разом з пакетом Ultralytics [19]. Цей вибір обґрунтований значними покращеннями в точності, швидкості та ефективності обробки даних, які надає новітня версія YOLO.

Ultralytics – це компанія, яка займається розробкою інструментів для комп'ютерного зору, серед яких основним продуктом є YOLO (You Only Look Once). Використання YOLOv8 з пакету Ultralytics має низку переваг, які роблять його ідеальним вибором для багатьох проектів в галузі комп'ютерного зору. Ось чому варто використовувати саме Ultralytics YOLOv8:

Перелічимо особливості та переваги Ultralytics YOLOv8.

1. Інтегрована екосистема:

- платформа дозволяє легко маркувати та сортувати дані за допомогою Ultralytics Hub та Roboflow. Це забезпечує зручне управління даними, які є основою для навчання моделей;

- підтримка зручних інструментів для тренування моделей, включаючи ноутбуки на Paperspace, що робить процес навчання гнучким і доступним;
  - включення систем логування, таких як Comet і ClearML, що дозволяє відстежувати і аналізувати процес навчання.
2. Зручність у розгортанні:
    - можливість легко розгортати моделі на різних платформах, зокрема на Neural Magic, що спрощує інтеграцію моделей у реальні додатки;
    - широкий спектр форматів для експорту моделей, включаючи TensorFlow, ONNX, CoreML, та інші, що робить модель сумісною з багатьма різними системами.
  3. Висока точність та швидкість роботи, оптимізація для різних апаратних платформ, від потужних серверів до мобільних пристроїв.
  4. Докладна документація та активна підтримка розробників та спільноти користувачів, що забезпечує швидке вирішення проблем і доступ до новітніх технологій та оновлень.

Для відстеження різних параметрів і моніторингу тренувальних процесів було застосовано ClearML. Це рішення виявилось надзвичайно важливим з кількох причин.

1. Відстеження різних параметрів під час тренування моделей, такі як точність, втрата, швидкість навчання тощо та їх візуалізація. Це допомагає швидко оцінити продуктивність моделі та вносити необхідні корективи.
2. Управління експериментами: ClearML дозволяє легко організовувати та порівнювати різні експерименти, зберігаючи всі версії моделей, конфігурацій та гіперпараметрів, що дозволяє визначити оптимальні конфігурації та параметри моделі.

3. Автоматизація процесів:
  - використання ClearML API дозволяє автоматизувати процеси відстеження та моніторингу, що знижує ризик помилок та підвищує ефективність роботи;
  - інтеграція з API ClearML дозволяє створювати скрипти, які автоматично запускають експерименти, збирають дані та генерують звіти.
4. ClearML дозволяє запускати експерименти на віддалених серверах і кластерах, забезпечуючи повний контроль над процесом виконання з будь-якого місця.
5. Відтворюваність:
  - ClearML автоматично зберігає конфігурації середовища, включаючи версії бібліотек і залежностей, що забезпечує повну репродуктивність експериментів;
  - контроль версій повернутися до попередніх версій експериментів для аналізу та порівняння.

Використання ClearML у поєднанні з YOLOv8 та пакетом Ultralytics значно покращило процес тренування моделей завдяки можливості відстеження параметрів у реальному часі, організації та порівняння експериментів, а також автоматизації рутинних завдань. Це дозволило досягти високої ефективності та точності у реалізації проекту.

### **3.4 Представлення результатів навчання моделі**

За результатами навчання моделі можна провести аналіз графіків зображених на рис.3.6 - рис.3.9 та виділити основні результати навчання, перелічені далі.

1. Загальний прогрес: дані показують, що з кожною епохою (від 1 до 50) помилки на тренувальному наборі даних зменшуються, що свідчить про покращення моделі.
2. Точність та повнота: метрики точності та повноти також покращуються з часом, що вказує на здатність моделі краще класифікувати дані.
3. mAP (Mean Average Precision): значення mAP50(B) та mAP50-95(B) зростають, що є індикатором загальної ефективності моделі у виявленні об'єктів.
4. Валідація: помилки на валідаційному наборі даних також зменшуються, хоча і не так стрімко, як на тренувальному наборі, що може вказувати на потребу подальшої оптимізації або уникнення перенавчання.

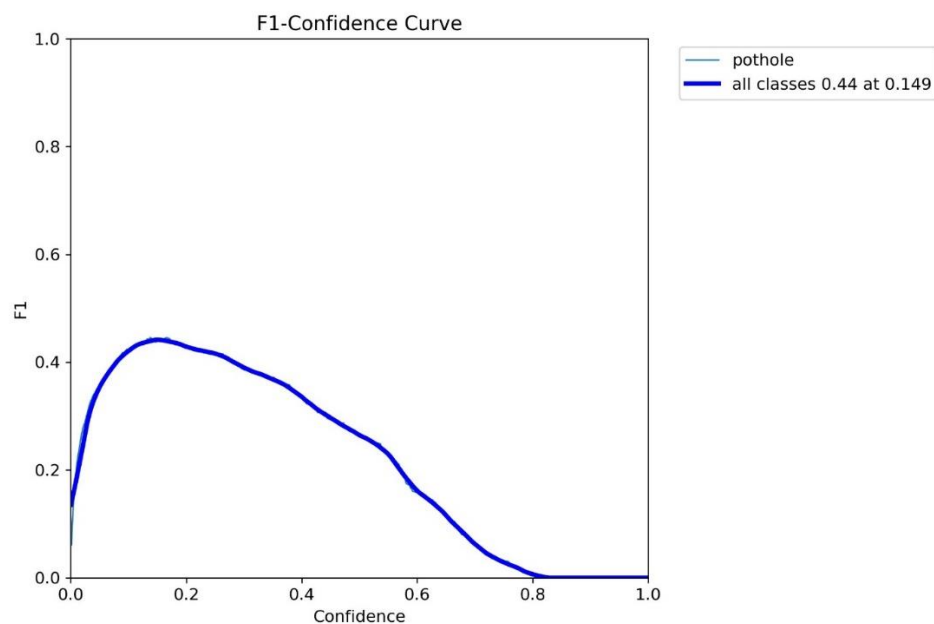


**Рисунок 3.6** – Графік кривої F1-впевненості з результатів

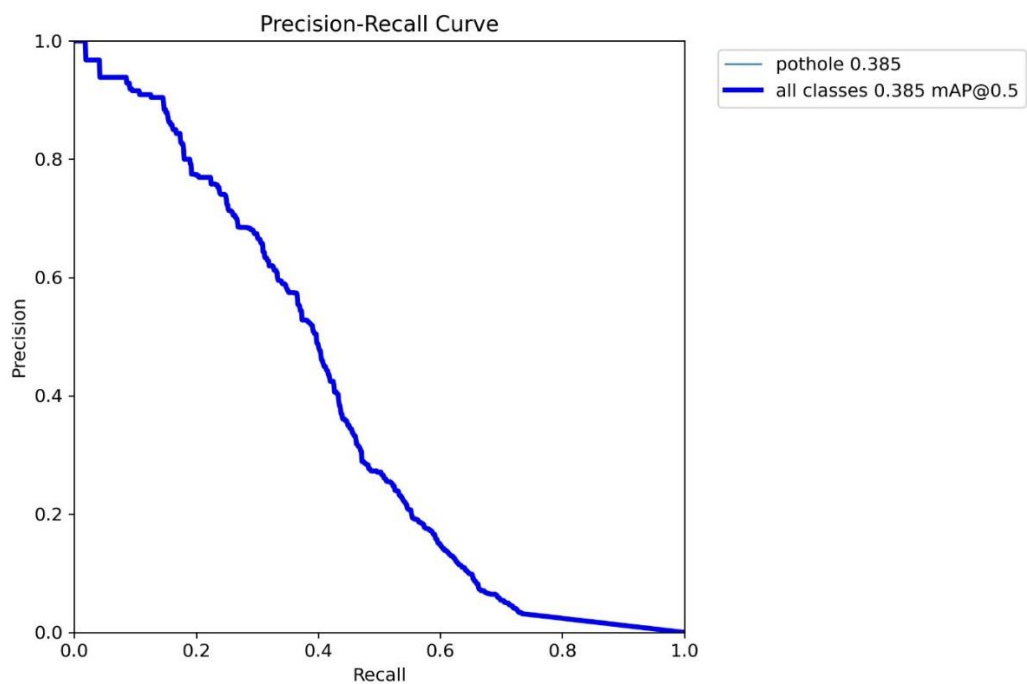


**Рисунок 3.7** – Графік PR-кривої з результатів

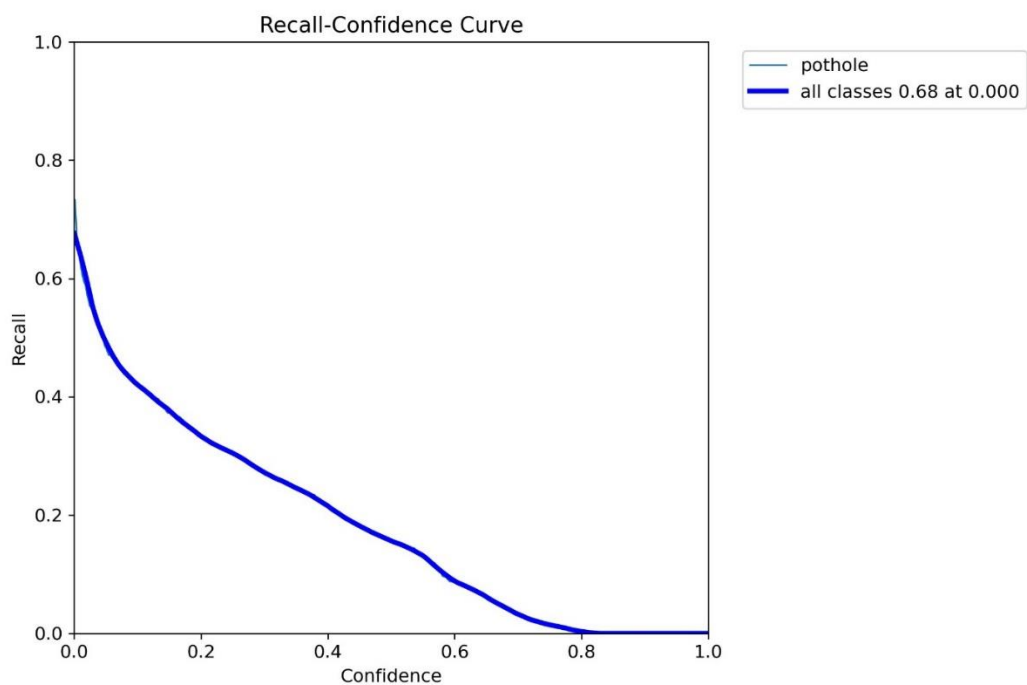


**Рисунок 3.8** – Графік кривої відношення точності та впевненості

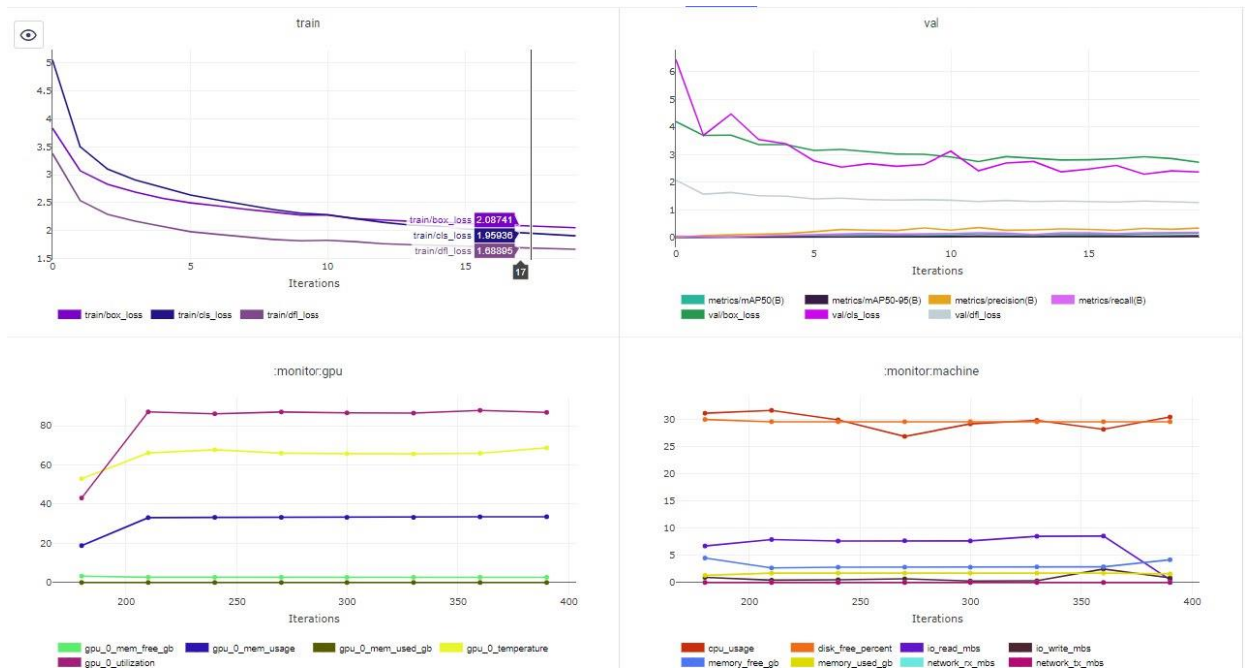


**Рисунок 3.9** – Основні метрики системи під час тренування набору

Також відобразимо результати роботи моделі для різних типів доріг на рис. 3.10 - рис.3.12.



**Рисунок 3.10** – Результат роботи для асфальтного покриття за умови низької роздільної здатності відео

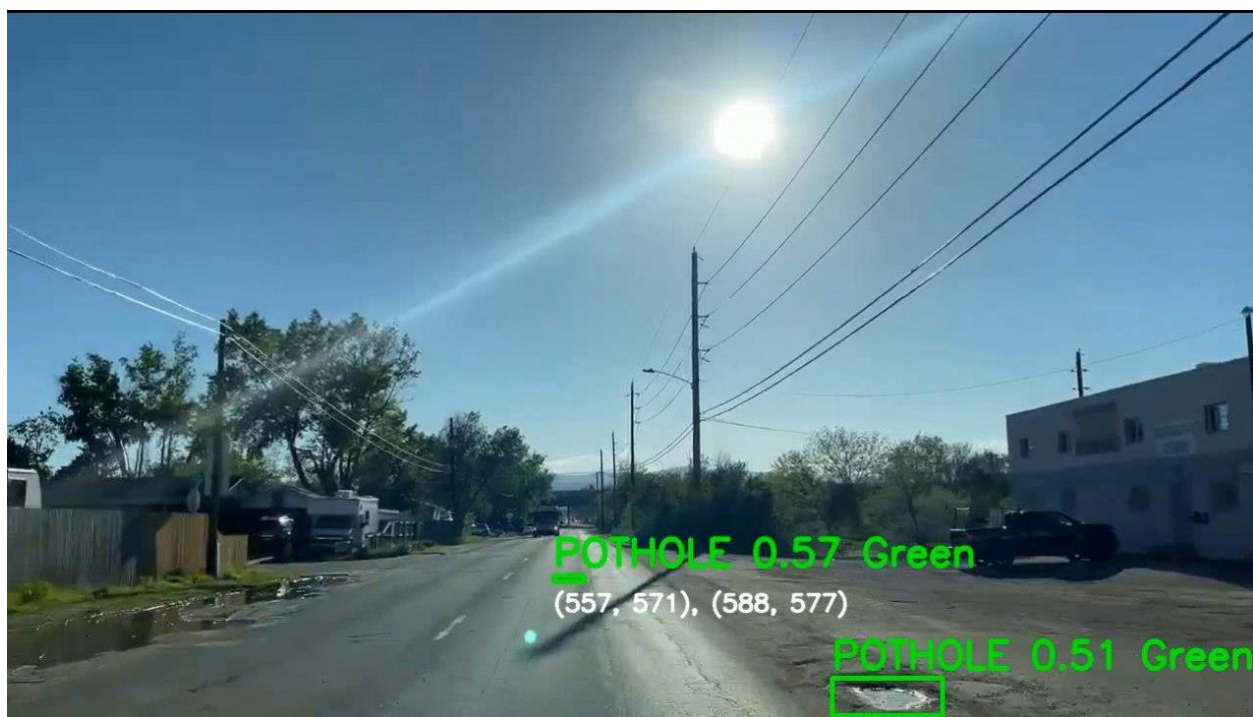


Рисунок 3.11 – Результат роботи для бетонного покриття



Рисунок 3.12 – Результат роботи для комбінованого покриття

Розпізнавання дефектів на асфальтному покритті показало найкращі результати. Це обумовлено кількома ключовими факторами. По-перше, більша кількість зображень була використана для тренування моделі, що забезпечило широкий спектр даних для навчання нейронної мережі. По-друге, під час відеозйомки відсутні коливання камери, що значно покращило якість зображень і зменшило можливість помилок при розпізнаванні. По-третє, загальна висока якість зображень сприяла точнішому визначенню дефектів асфальтного покриття.

Детекція дефектів на бетонному покритті показала трохи гірші результати. Основною причиною цього є менша кількість даних, доступних для тренування моделі. Обмеженість набору даних зменшує можливості нейронної мережі ефективно вивчати різноманітні типи дефектів, характерних для бетонних доріг.

Незважаючи на складність детекції дефектів на ґрунтових дорогах, модель показала достойні результати. Цей тип доріг має свою специфіку, і хоча кількість та якість зображень для тренування була меншою, модель змогла адекватно адаптуватися до різних умов і забезпечити задовільний рівень точності. Це свідчить про потенціал нейронних мереж у розпізнаванні дефектів на різних типах дорожнього покриття навіть за наявності певних обмежень у даних.

Після детального аналізу графіків, ми можемо сформулювати висновки щодо чотирьох ключових метрик на рис.3.9. На графіках тренувань видно, що значення втрат значно знижуються протягом перших 10 ітерацій, після чого стабілізуються, що свідчить про ефективне навчання моделі. На графіку валідації видно, що значення mAP50 та mAP50-95 поступово покращуються, тоді як показники precision і recall залишаються відносно стабільними після початкового покращення. Моніторинг GPU показує стабільне використання пам'яті та високий рівень завантаження (приблизно 80%), що вказує на інтенсивне використання графічного процесора під час тренувань. Моніторинг системних ресурсів показує стабільне використання процесора

(CPU) і пам'яті (RAM) протягом всього тренування, із невеликими коливаннями у використанні мережевих ресурсів та дискових операцій.

### **Висновки до розділу 3**

У третьому розділі було описано реалізацію програмної частини системи автоматичного розпізнавання дефектів дорожнього покриття за допомогою нейронних мереж. Основна увага в розділі приділяється аналізу вхідних наборів даних та їх підготовці до обробки нейронною мережею, так як був створений по'єднаний із різних джерел набір даних. Також було обрано модель нейронної мережі YOLOv8, пояснено переваги, особливості, архітектуру. Також було детально описано використаний набір інструментів для розробки програмного забезпечення, пояснено важивість та переваги їх використання. В решті було візуалізовано результати роботи моделі та на їхній основі зроблені висновки.

## РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на класифікації та розпізнаванні дефектів дорожнього покриття з використанням нейронних мереж. Реалізація цієї задачі зможе сприяти вирішенню таких важливих задач як забезпечення безпеки на дорозі, оптимізації роботи дорожніх служб, контролю якості виконаних робіт тощо.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

В даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

## 4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування програми на більшості операційних систем;
- можливість зручного масштабування та обслуговування;
- зручність та зрозумілість для користувача;
- швидкість виконання програми та ефективне використання ресурсів;
- мінімальні витрати на впровадження програмного продукту.

## 4.2 Обґрунтування функцій програмного продукту

Головна функція  $F_0$  – розробка можливого програмного продукту, який використовує методи класифікації та розпізнавання дефектів дорожнього покриття з використанням нейронних мереж.

Беручи за основу цю функцію, можна виділити наступні:

- $F_1$  – вибір мови програмування;
- $F_2$  – вибір моделей класифікації та розпізнавання;
- $F_3$  – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція  $F_1$ : Python, C++. Функція  $F_2$ : вибір готових переднавчених моделей класифікації та розпізнавання, власна розробка моделей. Функція  $F_3$ : Visual Studio Code, Sublime Text.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

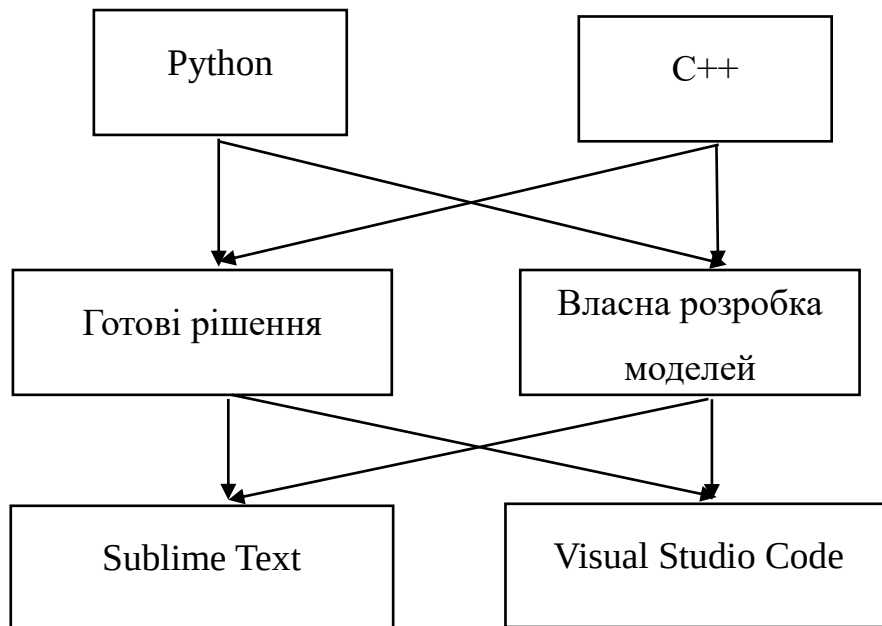


**Рисунок 4.1** – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Таблиця 4.1 – Позитивно-негативна матриця

| Функції | Варіанти реалізації | Переваги                                                                               | Недоліки                                                                |
|---------|---------------------|----------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| $F_1$   | <i>A</i>            | Простота у використанні, великий набір бібліотек для різних сфер                       | Відсутність контролю помилок під час розробки, повільна швидкодія       |
|         | <i>B</i>            | Контроль помилок під час розробки, велика швидкодія                                    | Складність у вивченні, використанні, значно менший арсенал бібліотек    |
| $F_2$   | <i>A</i>            | Швидкість впровадження, передчасно відомі потенційні результати                        | Неможливість внесення змін, можлива не найкраща оптимальність           |
|         | <i>B</i>            | Можливість внести авторські зміни, коригування будови для отримання оптимальності      | Довгострокова реалізація, передчасно не відомо чи успішна будова моделі |
| $F_3$   | <i>A</i>            | Великий набір функціоналу для налаштування під себе, функціонал для тестування програм | Більша ресурсоемність, складніший інтерфейс                             |
|         | <i>B</i>            | Простота, легкість програми                                                            | Повний доступ за оплатою, менший набір функціоналу                      |

Функція  $F_1$ .

Обираємо варіант А, так як простота, швидкість і великий набір реалізованого функціоналу у рамках цієї роботи є більш вигідним.

Функція  $F_2$ .

Обидва варіанти мають вагомі переваги, саме тому використання варіанту А так і варіанту Б для розробки можливе.

Функція  $F_3$ .

Обираємо варіант А, так як розробка програмного забезпечення у рамках цієї роботи може потребувати більше функціоналу від середи розробки програмного забезпечення.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

$$F_1a - F_2a - F_3a,$$

$$F_1a - F_2б - F_3a.$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

### 4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об'єм пам'яті для навчання та збереження даних;
- X3 – час навчання обраних моделей;
- X4 – потенційний об'єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

**Таблиця 4.2** – Основні параметри програмного продукту

| <i>Назва<br/>Параметра</i>         | <i>Умовні<br/>позначення</i> | <i>Одиниці<br/>виміру</i> | <i>Значення параметра</i> |                |              |
|------------------------------------|------------------------------|---------------------------|---------------------------|----------------|--------------|
|                                    |                              |                           | <i>гірші</i>              | <i>середні</i> | <i>кращі</i> |
| Швидкодія мови програмування       | X1                           | оп/мс                     | 10                        | 30             | 50           |
| Об'єм пам'яті                      | X2                           | Мб                        | 1400                      | 800            | 500          |
| Час навчання обраних моделей       | X3                           | хв                        | 300                       | 180            | 90           |
| Потенційний об'єм програмного коду | X4                           | кількість рядків коду     | 1200                      | 700            | 400          |

#### 4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, що спеціалізується на класифікації та розпізнаванні дефектів дорожнього покриття з використанням нейронних мереж.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведено у таблиці 4.3.

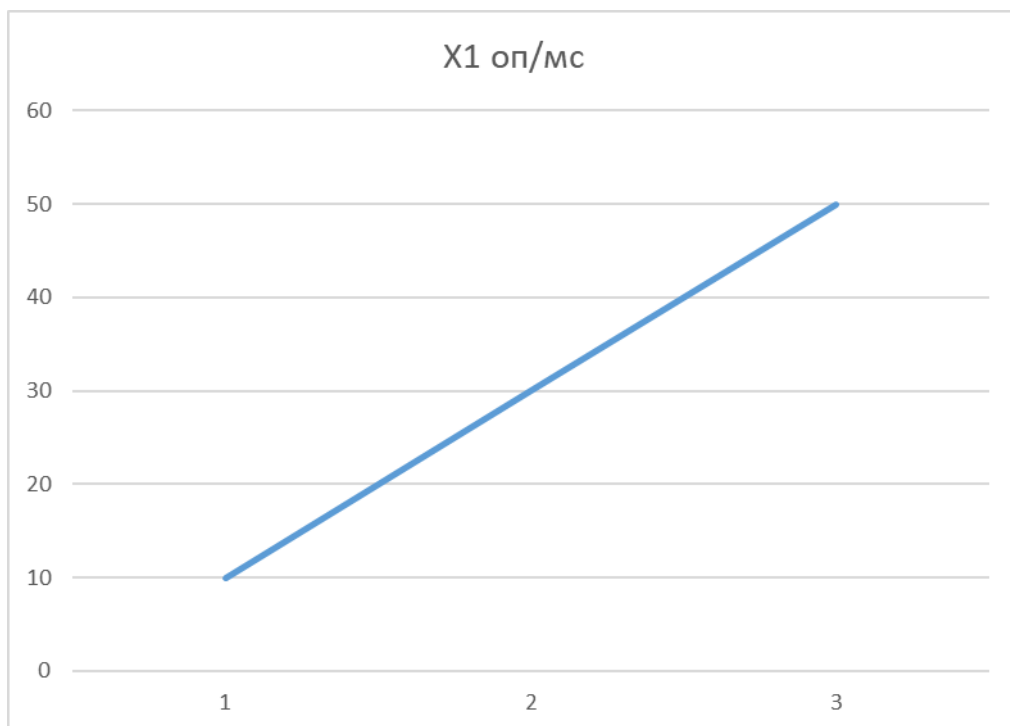


**Рисунок 4.2** – X1, швидкодія мови програмування

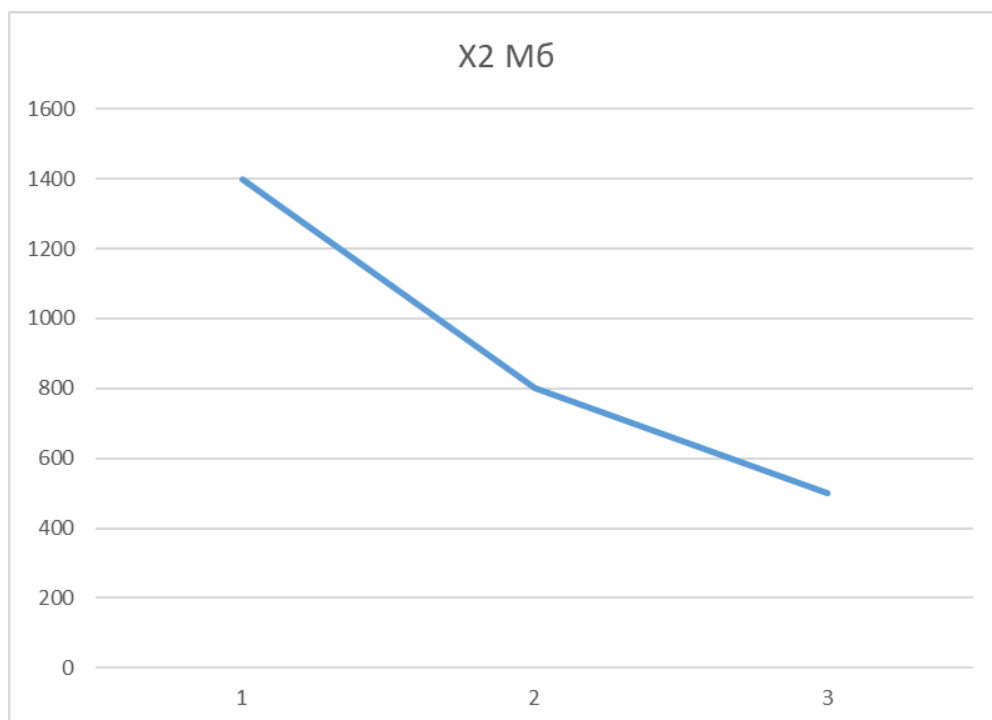


**Рисунок 4.3** – X2, об'єм використаної пам'яті

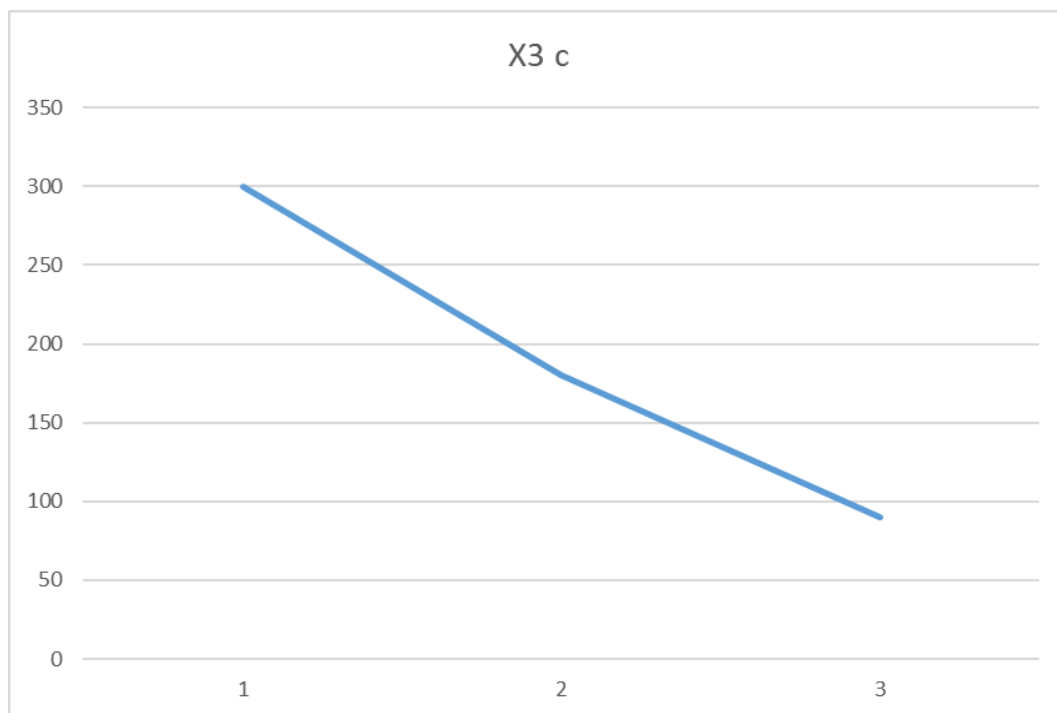


**Рисунок 4.4 – X3, час навчання обраних алгоритмів**

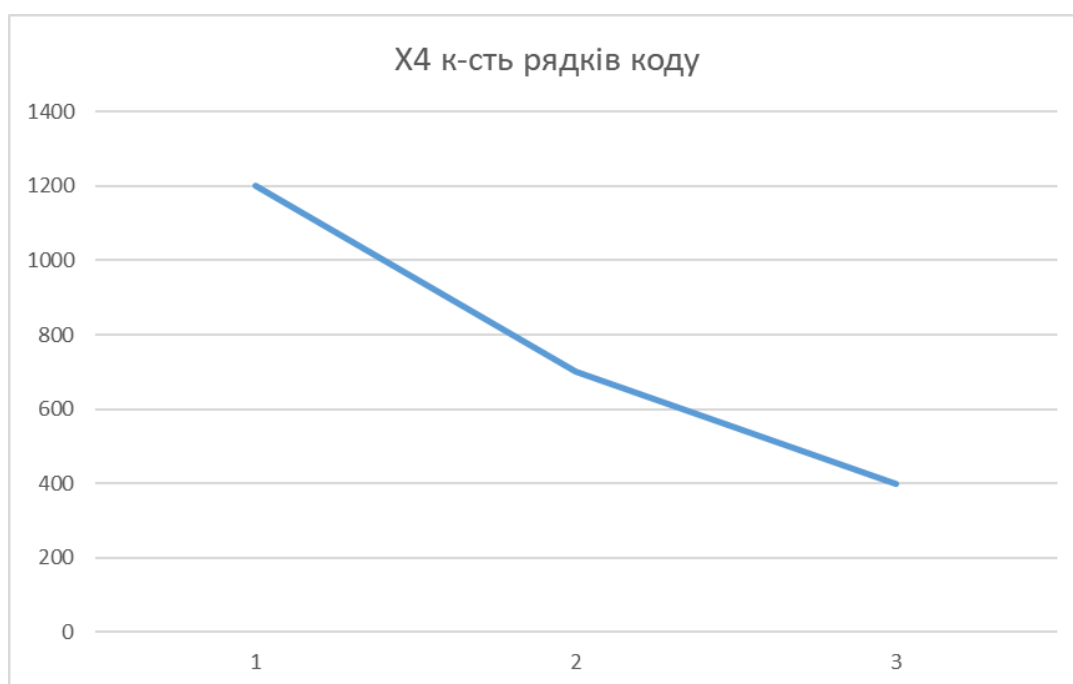


**Рисунок 4.5 – X4, потенційний об'єм програмного коду**

**Таблиця 4.3** – Результати ранжування параметрів

| Позначення параметра | Назва параметра                    | Одиниці виміру        | Ранг параметра за оцінкою експерта |    |    |    |    |    |    | Сума рангів $R_i$ | Відхилення $\Delta_i$ | $\Delta_i^2$ |
|----------------------|------------------------------------|-----------------------|------------------------------------|----|----|----|----|----|----|-------------------|-----------------------|--------------|
|                      |                                    |                       | 1                                  | 2  | 3  | 4  | 5  | 6  | 7  |                   |                       |              |
| $X_1$                | Швидкодія мови програмування       | Оп/мс                 | 2                                  | 1  | 1  | 2  | 1  | 1  | 2  | 10                | -7,5                  | 56,25        |
| $X_2$                | Об'єм пам'яті                      | Мб                    | 1                                  | 2  | 2  | 1  | 2  | 2  | 1  | 11                | -6,5                  | 42,25        |
| $X_3$                | Час навчання моделі                | с                     | 3                                  | 3  | 4  | 4  | 3  | 4  | 3  | 24                | 6,5                   | 42,25        |
| $X_4$                | Потенційний об'єм програмного коду | Кількість рядків коду | 4                                  | 4  | 3  | 3  | 4  | 3  | 4  | 25                | 7,5                   | 56,25        |
|                      | Разом                              |                       | 10                                 | 10 | 10 | 10 | 10 | 10 | 10 | 70                | 0                     | 197          |

Для перевірки ступеню достовірності експертних оцінок, визначимо наступні параметри.

Сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70,$$

де  $N$  – число експертів,

$n$  – кількість параметрів;

Середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17,5.$$

Відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

Сума відхилень по всіх параметрах повинна дорівнювати 0.

Загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 153.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2(4^3-4)} = 0,804 > W_k = 0,67.$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

**Таблиця 4.4** – Попарне порівняння параметрів

| Параметри | Експерти |   |   |   |   |   |   | Кінцева оцінка | Числове значення |
|-----------|----------|---|---|---|---|---|---|----------------|------------------|
|           | 1        | 2 | 3 | 4 | 5 | 6 | 7 |                |                  |
| X1 і X2   | <        | > | > | < | > | > | < | >              | 1.5              |
| X1 і X3   | >        | > | > | > | > | > | > | >              | 1.5              |
| X1 і X4   | >        | > | > | > | > | > | > | >              | 1.5              |
| X2 і X3   | >        | > | > | > | > | > | > | >              | 1.5              |
| X2 і X4   | >        | > | > | > | > | > | > | >              | 1.5              |
| X3 і X4   | >        | > | < | < | > | < | > | >              | 1.5              |

Числове значення, що визначає ступінь переваги  $i$ -го параметра над  $j$ -тим,  $a_{ij}$  визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю  $A = \|a_{ij}\|$ .

Для кожного параметра зробимо розрахунок вагомості  $K_{bi}$  за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i},$$

$$b_i = \sum_{j=1}^N a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). Обрахунки буде занесено у таблицю 4.5. На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{bi} = \frac{b'_i}{\sum_{i=1}^n b'_i},$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j.$$

**Таблиця 4.5** – Розрахунок вагомості параметрів

| Пара-<br>метри<br>$x_i$ | Параметри $x_j$ |     |     |     | Перша<br>ітер. |          | Друга ітер. |            | Третя ітер. |            | Четверта<br>ітер. |            |
|-------------------------|-----------------|-----|-----|-----|----------------|----------|-------------|------------|-------------|------------|-------------------|------------|
|                         | X1              | X2  | X3  | X4  | $b_i$          | $K_{bi}$ | $b_i^1$     | $K_{bi}^1$ | $b_i^2$     | $K_{bi}^2$ | $b_i^3$           | $K_{bi}^3$ |
| X1                      | 1               | 1.5 | 1.5 | 1.5 | 5.5            | 0.3437   | 30.25       | 0.432      | 166.37      | 0.526      | 915               | 0.62       |
| X2                      | 0.5             | 1   | 1.5 | 1.5 | 4.5            | 0.2812   | 20.25       | 0.289      | 91.12       | 0.289      | 410               | 0.273      |
| X3                      | 0.5             | 0.5 | 1   | 1.5 | 3.5            | 0.2188   | 12.25       | 0.175      | 42.88       | 0.136      | 150               | 0.01       |
| X4                      | 0.5             | 0.5 | 0.5 | 1   | 2.5            | 0.1563   | 6.25        | 0.089      | 15.63       | 0.049      | 39                | 0.025      |
| Всього:                 |                 |     |     |     | 16             | 1        | 70          | 1          | 316         | 1          | 1514              | 1          |

#### 4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів  $X_2$  (Об'єм пам'яті),  $X_3$  (час навчання моделей) та  $X_4$  (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра  $X_1$  (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j},$$

де  $n$  – кількість параметрів;

$K_{ei}$  – коефіцієнт вагомості  $i$ -го параметра;

$B_i$  – оцінка  $i$ -го параметра в балах.

**Таблиця 4.6** – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

| Основні функції | Варіант реалізації функції | Параметри | Абсолютне значення параметра | Бальна оцінка параметра | Коефіцієнт вагомості параметра | Коефіцієнт рівня якості |
|-----------------|----------------------------|-----------|------------------------------|-------------------------|--------------------------------|-------------------------|
| F1              | A                          | X1        | 20                           | 8                       | 0.21                           | 1.68                    |
| F2              | A                          | X2        | 700                          | 13                      | 0.342                          | 4.446                   |
| F2              | Б                          | X3        | 100                          | 8                       | 0.21                           | 1.68                    |
| F3              | A                          | X4        | 550                          | 9                       | 0.237                          | 2.13                    |

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \dots + K_{\text{ТУ}}[F_{zk}].$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1.68 + 4.446 + 2.13 = 8.256,$$

$$K_{K2} = 1.68 + 1.68 + 2.13 = 5.49.$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

#### **4.6 Економічний аналіз варіантів розробки ПП**

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

- розробка проекту програмного продукту;
- розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_P \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М},$$

де  $T_P$  – це трудомісткість розробки ПП;

$K_{\Pi}$  – поправочний коефіцієнт;

$K_{СК}$  – коефіцієнт на складність вхідної інформації;

$K_M$  – коефіцієнт рівня мови програмування;

$K_{СТ}$  – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$  – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює:  $T_P = 50$  людино-днів.

Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання:  $K_{\Pi} = 1.3$ . Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1:  $K_{СК} = 1$ . Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта  $K_{СТ} = 0.9$ . Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 50 \cdot 1.3 \cdot 0.9 = 58.5 \text{ людино-дні.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто  $T_P = 30$  людино-днів,  $K_{\Pi} = 0.8$ ,  $K_{СК} = 1$ ,  $K_{СТ} = 0.9$ :

$$T_2 = 30 \cdot 0.8 \cdot 0.9 = 21.6 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (58.5 + 21.6 + 6 + 21.6) \cdot 8 = 861.6 \text{ людино-годин,}$$

$$T_{II} = (58.5 + 21.6 + 4.1 + 21.6) \cdot 8 = 846.4 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант I.

В розробці беруть участь два програмісти з окладом 24000 грн., один аналітик в області даних з окладом 22000. Визначимо середню зарплату за годину за формулою:

$$C_q = \frac{M}{T_m \cdot t} \text{ грн,}$$

де  $M$  – це місячний оклад працівників;

$T_m$  – кількість робочих днів тиждень;

$t$  – кількість робочих годин в день.

$$C_q = \frac{24000 + 24000 + 22000}{3 \cdot 21 \cdot 8} = 138.88 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{зп} = C_q \cdot T_i \cdot K_d,$$

де  $C_q$  – це величина погодинної оплати праці програміста;

$T_i$  – трудомісткість відповідного завдання;

$K_d$  – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I.} \quad C_{3\Pi} = 138.88 \cdot 861.6 \cdot 1.2 = 143590.8 \text{ грн},$$

$$\text{II.} \quad C_{3\Pi} = 138.88 \cdot 846.4 \cdot 1.2 = 141057.6 \text{ грн}.$$

Відрахування на єдиний соціальний внесок становить 22%:

$$\text{I.} \quad C_{\text{ВІД}} = C_{3\Pi} \cdot 0.22 = 143590.8 \cdot 0.22 = 31589.98 \text{ грн},$$

$$\text{II.} \quad C_{\text{ВІД}} = C_{3\Pi} \cdot 0.22 = 141057.6 \cdot 0.22 = 31032.7 \text{ грн}.$$

Тепер визначимо витрати на оплату однієї машино-години ( $C_M$ ).

Так як одна ЕОМ обслуговує одного програміста з окладом 24000 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 24000 \cdot 0.2 = 57600 \text{ грн}.$$

З урахуванням додаткової заробітної плати:

$$C_{3\Pi} = C_{\Gamma} \cdot (1 + K_3) = 57600 \cdot (1 + 0.2) = 69120 \text{ грн}.$$

Відрахування на соціальний внесок:

$$C_{\text{ВІД}} = C_{3\Pi} \cdot 0.22 = 69120 \cdot 0.22 = 15206.4 \text{ грн}.$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 22000 грн.

$$C_A = K_{\text{ТМ}} \cdot K_A \cdot C_{\text{ПР}} = 1.2 \cdot 0.25 \cdot 22000 = 6600 \text{ грн},$$

де  $K_{\text{ТМ}}$  – це коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$K_A$  – річна норма амортизації;

$\Pi_{\text{ПР}}$  – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{\text{ТМ}} \cdot \Pi_{\text{ПР}} \cdot K_P = 1.2 \cdot 22000 \cdot 0.08 = 2112 \text{ грн},$$

де  $K_P$  – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$\begin{aligned} T_{\text{ЕФ}} &= (D_K - D_B - D_C - D_P) \cdot t_z \cdot K_B = (365 - 104 - 12 - 14) \cdot 8 \cdot 0.4 = \\ &= 752 \text{ години.} \end{aligned}$$

де  $D_K$  – це календарна кількість днів у році;

$D_B, D_C$  – відповідно кількість вихідних та святкових днів;

$D_P$  –: кількість днів планових ремонтів устаткування;

$t$  – кількість робочих годин в день;

$K_B$  – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_C \cdot K_3 \cdot \Pi_{\text{ЕН}} = 752 \cdot 0.2 \cdot 0.4 \cdot 5.23 = 314.63 \text{ грн},$$

де  $N_C$  – середньо-споживча потужність приладу;

$K_3$  – коефіцієнтом зайнятості приладу;

$\Pi_{\text{ЕН}}$  – тариф за 1 кВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = \Pi_{\text{ПР}} \cdot 0.67 = 22000 \cdot 0.67 = 14740 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}}.$$

$$C_{\text{ЕКС}} = 69120 + 15206.4 + 6600 + 2112 + 314.63 + 14740 = 105981.03 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 105981.0 / 752 = 140.93 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_{\text{М}} = C_{\text{М-Г}} \cdot T,$$

$$\text{I. } C_{\text{М}} = 140.93 \cdot 861.6 = 121425.3 \text{ грн,}$$

$$\text{II. } C_{\text{М}} = 140.93 \cdot 846.4 = 119283.15 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0.67,$$

$$\text{I. } C_{\text{Н}} = 143590.8 \cdot 0.67 = 96205.83 \text{ грн,}$$

$$\text{II. } C_{\text{Н}} = 141057.6 \cdot 0.67 = 94508.59 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}}.$$

$$\text{I. } C_{\text{ПП}} = 143590.8 + 31589.98 + 121425.3 + 96205.83 = 392811.9 \text{ грн.}$$

$$\text{II. } C_{\text{III}} = 141057.6 + 31032.7 + 119283.15 + 94508.59 = 385881.44 \text{ грн.}$$

#### 4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{\text{К}j} / C_{\text{Ф}j},$$

$$K_{\text{ТЕР}1} = 8.256 / 392811.9 = 2.1 \cdot 10^{-5},$$

$$K_{\text{ТЕР}2} = 5.49 / 385881.44 = 1.42 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня  $K_{\text{ТЕР}1} = 2.1 \cdot 10^{-5}$ .

Із виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості  $K_{\text{ТЕР}} = 2.1 \cdot 10^{-5}$ .

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- використання готових алгоритмів класифікації та розпізнавання;
- середовище розробки – Visual Studio Code.

Цей варіант виконання програмного комплексу дає користувачу простоту розробки, передбачуваність результатів та швидкість розробки.

## **Висновки до розділу 4**

У четвертому розділі було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту. За результатами аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують, на основі чого було обрано варіант реалізації програмного продукту: мова програмування Python, використання готових моделей для класифікації та розпізнавання, використання середовища програмування Visual Studio Code.

## ВИСНОВКИ

У цій дипломній роботі було виконано дослідження використання нейронних мереж для автоматичного розпізнавання дефектів дорожнього покриття.

В першому розділі було розглянуто різні підходи до ідентифікації дефектів дорожнього покриття, зокрема за допомогою комп'ютерного зору та нейронних мереж. Проведено аналіз існуючих методів виявлення дефектів, за результатами якого встановлено, що згорткові нейронні мережі є найбільш ефективними для вирішення цієї задачі завдяки їх здатності автоматично витягувати релевантні ознаки з зображень.

У другому розділі було детально досліджено архітектуру згорткових нейронних мереж, зокрема шари згортки, активації, агрегації та повнозв'язний шар. Розглянуто різні функції активації та методи агрегації, а також найпопулярніші оптимізатори для тренування нейронних мереж. Проведено порівняння основних моделей.

У третьому розділі було обрано модель YOLO для розробки програмного продукту, зроблено детальний огляд та описано причини такого вибору. Описано процес збору навчальних даних, навчання моделі та перевірки її результатів на тестових зразках. Представлено алгоритм роботи програмного забезпечення для системи розпізнавання та описано процес його функціонування.

У четвертому розділі було проведено функціонально-вартісний аналіз програмного забезпечення, за результатами якого обрано оптимальний варіант реалізації програмного продукту. Зроблено висновок про ефективність використання готових реалізацій нейронних мереж для розпізнавання дефектів дорожнього покриття, використання мови програмування Python та середовища розробки Visual Studio Code.

В результаті роботи було створено систему для автоматичного розпізнавання дефектів дорожнього покриття, яка може використовуватися для моніторингу стану доріг та своєчасного проведення ремонтних робіт. Подальша робота полягає у вдосконаленні системи для збільшення точності розпізнавання, розширення області її застосування та спрощення налаштувань для досягнення необхідних результатів.

Для подальшого вдосконалення системи рекомендується оптимізувати модель для зменшення часу обробки зображень, впровадити додаткові методи аугментації даних для підвищення стійкості моделі, а також розширити функціонал системи для виявлення нових типів дефектів дорожнього покриття. Все це сприятиме покращенню якості обслуговування доріг та підвищенню безпеки дорожнього руху, що підкреслює великий потенціал даної системи.

## ПЕРЕЛІК ПОСИЛАНЬ

1. S. Zhang and Akram Alhelyani, “Maintenance technologies for roads and its prioritization,” Zenodo (CERN European Organization for Nuclear Research), Oct. 2022, doi: <https://doi.org/10.5281/zenodo.7264113>.
2. Hamzeh Zakeri, Fereidoon Moghadas Nejad, and A. H. Gandomi, Automation and Computational Intelligence for Road Maintenance and Management. John Wiley & Sons, 2022.
3. F. Alvi, “Computer Vision and Image Processing: Understanding the Distinction and Interconnection,” OpenCV, Dec. 13, 2023. <https://opencv.org/blog/computer-vision-and-image-processing/>.
4. A. Olafenwa (she/her), “Image Segmentation With 5 Lines Of Code,” Medium, Nov. 25, 2020. <https://towardsdatascience.com/image-segmentation-with-six-lines-of-code-acb870a462e8>.
5. D. Shah, “Object Tracking,” Medium, Apr. 29, 2020. <https://medium.com/visionwizard/object-tracking-675d7a33e687>
6. Y. Poirier-Ginter and J.-F. Lalonde, “Robust Unsupervised StyleGAN Image Restoration,” Jun. 2023. [Online]. Available: <https://arxiv.org/pdf/2302.06733>.
7. S. Bigdeli and M. Zwicker, “Image Restoration using Autoencoding Priors,” Apr. 2017. Available: <https://arxiv.org/pdf/1703.09964>.
8. Терейковський І.А., Бушуєв. Д.А, та Терейковська Л.О., “ШТУЧНІ НЕЙРОННІ МЕРЕЖІ: БАЗОВІ ПОЛОЖЕННЯ,” 2022. Available: <https://ela.kpi.ua/server/api/core/bitstreams/9fee52b6-83fc-4e99-8541-c2767f634c7c/content>.
9. K. O'shea and R. Nash, “An Introduction to Convolutional Neural Networks,” Dec. 2015. Available: <https://arxiv.org/pdf/1511.08458>.
10. IBM, “What are Convolutional Neural Networks? | IBM,” [www.ibm.com](http://www.ibm.com), 2023. <https://www.ibm.com/topics/convolutional-neural-networks>.

11. P. Baheti, “12 Types of Neural Networks Activation Functions: How to Choose?” [www.v7labs.com](http://www.v7labs.com), Mar. 08, 2022.  
<https://www.v7labs.com/blog/neural-networks-activation-functions>.
12. N. Malviya, “Evaluation of Object Detection Models — Flops, FPS, Latency, Params, Size, Memory, Storage, mAP...” Medium, May 23, 2024.  
<https://medium.com/@nikitamalviya/evaluation-of-object-detection-models-flops-fps-latency-params-size-memory-storage-map-8dc9c7763cfe>.
13. Z. Kelta, “YOLO Object Detection Explained: A Beginner’s Guide,” [www.datacamp.com](http://www.datacamp.com), Sep. 2022. <https://www.datacamp.com/blog/yolo-object-detection-explained>
14. J. Kim and J. Cho, “RGDiNet: Efficient Onboard Object Detection with Faster R-CNN for Air-to-Ground Surveillance,” *Sensors*, vol. 21, no. 5, p. 1677, Mar. 2021, doi: <https://doi.org/10.3390/s21051677>.
15. J. Jiang, H. Xu, S.-C. Zhang, and Y. Fang, “Object Detection Algorithm Based on Multiheaded Attention,” vol. 9, no. 9, pp. 1829–1829, May 2019, doi: <https://doi.org/10.3390/app9091829>.
16. A. Gautam and S. Singh, “Neural style transfer combined with EfficientDet for thermal surveillance” *The Visual Computer*, Aug. 2021, doi: <https://doi.org/10.1007/s00371-021-02284-2>.
17. CVAT, “Documentation” CVAT. <https://docs.cvat.ai/docs/>.
18. G. Jocher, A. Chaurasia, and J. Qiu, “YOLOv8 by Ultralytics,” GitHub, Jan. 01, 2023. <https://github.com/ultralytics/ultralytics>.

## ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

train\_model.py

```
from clearml import Task
from ultralytics import YOLO
import multiprocessing

def main():
    # Step 1: Creating a ClearML Task
    task = Task.init(project_name="my_project_n", task_name="my_yolov8_task")

    # Load a model
    model = YOLO("yolov8s.yaml") # build a new model from scratch

    # Use the model
    results = model.train(
        data="pothole_v8.yaml",
        imgsz=1280,
        epochs=50,
        batch=8,
        name='yolov8n_v8_50e_n'
    )

if __name__ == '__main__':
    multiprocessing.freeze_support()
    main()
```

video\_app.py

```
import os
import cv2
from ultralytics import YOLO
from tkinter import Tk, Button, Label, filedialog, messagebox

# Define the video processing function
def process_video(input_path, output_path, model_path):
    cap = cv2.VideoCapture(input_path)
    ret, frame = cap.read()
    H, W, _ = frame.shape
    out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'mp4v'),
int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

    model = YOLO(model_path) # load a custom model
    threshold = 0.5
    area_threshold_yellow = 5000
```

```

area_threshold_red = 20000

def classify_rectangle(area):
    if area < area_threshold_yellow:
        return (0, 255, 0), "Green"
    elif area < area_threshold_red:
        return (0, 255, 255), "Yellow"
    else:
        return (0, 0, 255), "Red"

while ret:
    results = model(frame)[0]

    for result in results.bboxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result
        if score > threshold:
            area = (x2 - x1) * (y2 - y1)
            color, color_label = classify_rectangle(area)
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)),
color, 4)
            label = f"{results.names[int(class_id)].upper()} {score:.2f}
{color_label}"
            cv2.putText(frame, label, (int(x1), int(y1 - 10)),
cv2.FONT_HERSHEY_SIMPLEX, 1.3, color, 3, cv2.LINE_AA)
            coords = f"({int(x1)}, {int(y1)}), ({int(x2)}, {int(y2)})"
            cv2.putText(frame, coords, (int(x1), int(y2) + 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2, cv2.LINE_AA)
            print(f"Detected {results.names[int(class_id)]} at [{int(x1)},
{int(y1)}, {int(x2)}, {int(y2)}] with confidence {score:.2f} and area {area}
({color_label})")

    out.write(frame)
    ret, frame = cap.read()

cap.release()
out.release()

# Define the GUI application
class VideoApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Video Detection App")
        self.label = Label(root, text="Select a video to process")
        self.label.pack(pady=20)
        self.load_button = Button(root, text="Load Video",
command=self.load_video)
        self.load_button.pack(pady=10)
        self.process_button = Button(root, text="Process Video",
command=self.process_video)
        self.process_button.pack(pady=10)

```

```

        self.save_button = Button(root, text="Save Video",
command=self.save_video)
        self.save_button.pack(pady=10)
        self.video_path = ""
        self.output_path = ""

    def load_video(self):
        self.video_path = filedialog.askopenfilename(filetypes=[("MP4 files",
 "*.mp4")])
        if self.video_path:
            self.label.config(text=f"Loaded video:
{os.path.basename(self.video_path)}")

    def process_video(self):
        if not self.video_path:
            messagebox.showwarning("Warning", "Please load a video first")
            return
        self.output_path = os.path.join(os.path.dirname(self.video_path),
f"{os.path.splitext(os.path.basename(self.video_path))[0]}_out.mp4")
        model_path = os.path.join('.', 'runs', 'detect', 'yolov8n_v8_50e14',
'weights', 'last.pt')

        process_video(self.video_path, self.output_path, model_path)
        messagebox.showinfo("Info", f"Video processed successfully. Saved as:
{self.output_path}")

    def save_video(self):
        if not self.output_path:
            messagebox.showwarning("Warning", "No processed video to save")
            return
        save_path = filedialog.asksaveasfilename(defaultextension=".mp4",
filetypes=[("MP4 files", "*.mp4")])
        if save_path:
            os.rename(self.output_path, save_path)
            messagebox.showinfo("Info", f"Video saved successfully as:
{save_path}")

if __name__ == "__main__":
    root = Tk()
    app = VideoApp(root)
    root.mainloop()

```

## remove\_img\_without\_obj.py

```

"""
Script to remove images without objects from the validation set.
"""

import os

```

```

import cv2

TXT_ROOT = os.path.join('valid', 'labels')
IMG_ROOT = os.path.join('valid', 'images')

all_txt_files = os.listdir(TXT_ROOT)

zero_counter = 0
txt_without_obj = []
for i, file_name in enumerate(all_txt_files):
    file_path = os.path.join(TXT_ROOT, file_name)
    with open(file_path, 'r') as f:
        lines = f.readlines()
        if len(lines) == 0:
            zero_counter += 1
            txt_without_obj.append(file_name.split('.')[0])
    f.close()

all_img_files = os.listdir(IMG_ROOT)
zero_img_counter = 0
for i, file_name in enumerate(all_img_files):
    file_path = os.path.join(IMG_ROOT, file_name)
    if '.'.join(file_name.split('.')[::-1]) in txt_without_obj:
        zero_img_counter += 1
        # image = cv2.imread(os.path.join(IMG_ROOT, file_name))
        # cv2.imshow('Image', image)
        # cv2.waitKey(0)
        os.remove(os.path.join(IMG_ROOT, file_name))
        os.remove(os.path.join(TXT_ROOT, '.'.join(file_name.split('.')[::-1])[:-1])+'.txt'))

print(zero_img_counter)

```