МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ЦЬОМА Іван Степанович

Програмний модуль виявлення дефектів зварних швів із використанням глибинного навчання / Software Module for Detecting Welding Defects Using Deep Learning

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-43
Цьома І.С.
_____

Науковий керівник
к.т.н., доцент Д.І. Загородня

Кваліфікаційну роботу допущено до захисту
«___»_____ 20___ р.
Завідувач кафедри
_____ Н. В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ЦЬОМА Іван Степанович
(прізвище, ім'я, по батькові)

1. Програмний модуль виявлення дефектів зварних швів із використанням глибинного навчання / Software Module for Detecting Welding Defects Using Deep Learning
керівник роботи Д. І. Загородня к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 01 грудня 2025 р. № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:

- здійснити огляд існуючих систем виявлення дефектів зварних швів;
- проаналізувати сучасні методи виділення та зіставлення ознак зображень для задач виявлення об'єктів;
- розробити архітектурне, алгоритмічне та інформаціне забезпечення системи та створити загальну структуру модуля виявлення дефектів зварних швів;
- реалізувати програмне забезпечення, створити інтерфейс користувача і провести тестування програми.

5. Перелік графічного матеріалу в роботі:

- структурна схема програмного модуля;
- діаграма потоків даних (DFD) процесу розпізнавання;
- діаграма послідовності взаємодії компонентів системи;
- алгоритм виявлення дефектів зварного шва;
- результати тестування програми.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ І.С. Цьома_____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Д. І. Загородня
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль виявлення дефектів зварних швів із використанням глибинного навчання» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 68 сторінок і містить 16 ілюстрацій, 4 додатки та 31 використане джерело.

Метою роботи є розробка програмного модуля для автоматизованого виявлення та класифікації дефектів зварних швів на цифрових зображеннях із використанням методів глибинного навчання та нейромережевої моделі YOLOv8.

Методи дослідження: використано методи системного аналізу для дослідження предметної області комп'ютерного зору, методи об'єктно-орієнтованого проєктування для побудови архітектури програмної системи, класичні алгоритми виділення та зіставлення ознак зображень (ORB, LBP), методи частотного аналізу, а також методи машинного навчання та експериментального оцінювання точності класифікації.

Розроблений програмний продукт може бути використаний для задач автоматичного виявлення та класифікації дефектів зварних швів, у системах комп'ютерного зору для задач відеоспостереження.

Ключові слова: ЗВАРНИЙ ШОВ, КОМП'ЮТЕРНИЙ ЗІР, ЗІСТАВЛЕННЯ ОЗНАК, ВИЯВЛЕННЯ ОБ'ЄКТІВ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ЗОБРАЖЕНЬ.

ANNOTATION

The qualification thesis entitled “Software Module for Detecting Welding Defects Using Deep Learning” submitted for the Bachelor's degree in Specialty 122 Computer Science under the Educational Program Computer Science comprises 68 pages and contains 16 figures, 4 appendices, and 31 references.

The aim of the thesis is to develop a software module for automated detection and classification of weld defects in digital images using deep learning methods and the YOLOv8 neural network model.

Research methods: system analysis methods were used to investigate the computer vision domain; object-oriented design methods were applied to develop the software architecture; classical image feature extraction and matching algorithms (ORB, LBP), frequency analysis methods, as well as machine learning techniques and experimental evaluation methods for classification accuracy were employed.

The developed software product can be used for the automated detection and classification of weld seam defects and in computer vision systems designed for video surveillance tasks.

Keywords: WELD SEAM, COMPUTER VISION, FEATURE MATCHING, OBJECT DETECTION, MACHINE LEARNING, IMAGE ANALYSIS.

ЗМІСТ

ВСТУП

У сучасній промисловості зварні з’єднання широко застосовуються в машинобудуванні, будівництві, енергетиці, транспортній галузі та інших сферах. Надійність і міцність багатьох конструкцій значною мірою залежать від якості зварних швів, оскільки саме вони часто зазнають значних механічних навантажень, вібрацій і температурних впливів. Наявність дефектів у зварних з’єднаннях, таких як тріщини, пори, непровари чи включення, може призвести до зниження міцності конструкцій, аварійних ситуацій та значних матеріальних втрат. Тому своєчасне виявлення дефектів зварних швів є важливим завданням забезпечення якості та безпеки виробництва.

Традиційні методи контролю якості зварних з’єднань (візуальний контроль, ультразвукова дефектоскопія, рентгенографія) потребують значних часових і трудових витрат, а також залежать від досвіду фахівців, що може призводити до суб’єктивності оцінювання результатів. У зв’язку з цим актуальним напрямом розвитку є автоматизація процесів контролю якості з використанням сучасних інформаційних технологій і систем комп’ютерного зору.

Особливу перспективу в цій сфері мають методи глибинного навчання, які дозволяють автоматично аналізувати зображення зварних швів і виявляти складні закономірності та дефекти без необхідності ручного виділення ознак. Зокрема, згорткові нейронні мережі (CNN) та інші архітектури глибокого навчання можуть ефективно виконувати задачі класифікації, сегментації та локалізації дефектів на поверхні матеріалів і виробів. Використання таких моделей дозволяє підвищити точність, швидкість та об’єктивність контролю якості виробництва.

Крім того, автоматизовані системи виявлення дефектів є важливим елементом концепції інтелектуального виробництва та Індустрії 4.0, де контроль якості здійснюється в реальному часі за допомогою цифрових технологій. Використання програмних модулів на основі глибинного навчання дозволяє інтегрувати систему контролю в автоматизовані виробничі процеси, підвищити ефективність технічного моніторингу та забезпечити своєчасне виявлення критичних дефектів. Тому виявлення дефектів зварних швів із використанням

технологій глибинного навчання є актуальною науково-практичною задачею, спрямованою на підвищення якості продукції, зниження виробничих ризиків та удосконалення систем автоматизованого контролю в сучасній промисловості.

Основною метою даної дипломної роботи є розробка програмного модуля для автоматизованого виявлення та класифікації дефектів зварних швів на цифрових зображеннях із використанням методів глибинного навчання та нейромережевої моделі YOLOv8.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати предметну область контролю якості зварних швів;

– дослідити сучасні методи комп'ютерного зору та глибинного навчання для виявлення дефектів на зображеннях;

– провести аналіз існуючих програмних рішень для детектування дефектів зварних з’єднань;

– розробити архітектуру програмного модуля;

– здійснити програмну реалізацію програмного модуля для виявлення та класифікації дефектів зварних швів;

– провести експериментальні дослідження програмного модуля та оцінити ефективність запропонованого рішення.

Об'єктом дослідження є процес автоматизованого виявлення дефектів зварних швів на цифрових зображеннях.

Предметом дослідження виступають методи та алгоритми глибинного навчання, комп'ютерного зору й аналізу зображень, що використовуються для виявлення та класифікації дефектів зварних з'єднань.

Практична цінність отриманих результатів полягає у можливості використання розробленого програмного модуля в системах неруйнівного контролю для автоматизації процесу перевірки якості зварних швів, зменшення впливу людського фактора та підвищення достовірності виявлення дефектів у промислових умовах.

Структура дипломної роботи складається з трьох логічно послідовних розділів, у яких висвітлено аналіз предметної області та сучасних підходів до виявлення дефектів зварних швів, проєктування та реалізацію програмного модуля,

а також результати експериментальних досліджень і оцінювання ефективності розробленого рішення.

Загальний обсяг роботи складає 68 сторінок і містить 16 рисунків, чотири додатки та 31 використане джерело.

Апробацію результатів дослідження проведено на Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі», яка відбулася у 2026 році (м. Тернопіль, Україна). Копія публікації представлена в додатку А.

1 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Контроль якості зварних з’єднань є важливою складовою технологічного процесу в багатьох галузях промисловості, зокрема в машинобудуванні, енергетиці, авіаційній та автомобільній промисловості, будівництві металоконструкцій та трубопроводів. Зварні шви виконують функцію з’єднання окремих елементів конструкцій, тому їхня надійність безпосередньо впливає на безпеку експлуатації технічних систем. Наявність дефектів у зварних з’єднаннях може призвести до зниження міцності конструкції, виникнення тріщин, деформацій або навіть до аварійних ситуацій.

Відповідно до міжнародного стандарту ДСТУ EN ISO 6520-1, дефектом зварного шва вважається будь-яке відхилення від нормативної геометрії або порушення суцільності металу, що призводить до зниження його механічних характеристик [1]. До найбільш поширених дефектів зварних швів належать тріщини, пористість, «непровари», підрізи, шлакові включення, напливи металу та інші порушення структури матеріалу. Усю сукупність дефектів класифікують за місцем їхнього розташування та фізичною природою на рисунку 1.1. Такі дефекти можуть виникати внаслідок порушення технології зварювання, неправильного підбору режимів зварювання, використання неякісних матеріалів або впливу зовнішніх факторів. Тому своєчасне виявлення та класифікація дефектів є важливим завданням забезпечення якості виробництва та безпечної експлуатації виробів.

Особливу небезпеку становлять внутрішні дефекти (тріщини, непровари, пори), оскільки вони є концентраторами напружень і можуть призвести до раптового руйнування конструкції [2]. Для їх виявлення використовують методи неруйнівного контролю.

Традиційно контроль якості зварних з’єднань здійснюється за допомогою різних методів неруйнівного контролю, зокрема візуально-вимірювальний, ультразвукової дефектоскопії, рентгенографії, магнітопорошкового та капілярного

методів. Однак такі методи часто потребують значних часових витрат, високої кваліфікації фахівців та спеціалізованого обладнання. Крім того, результати контролю можуть залежати від людського фактора, що знижує об’єктивність оцінювання.

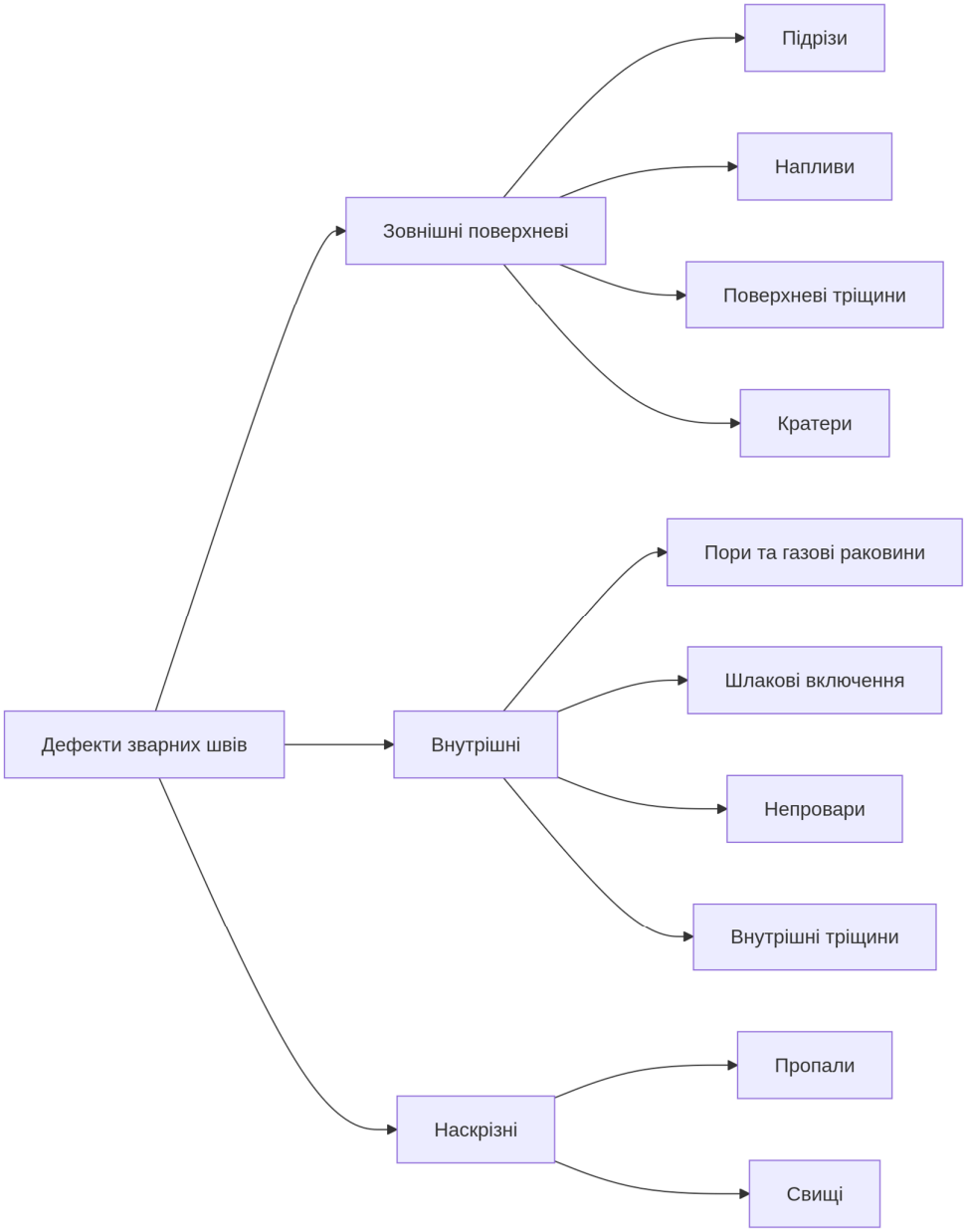

Рисунок 1.1 – Загальна класифікація дефектів зварних з'єднань

Одним із найпоширеніших і найінформативніших методів для виявлення внутрішніх дефектів є радіографічний контроль. Його фізична суть базується на законі поглинання іонізуючого випромінювання матеріалом (закон Бугера-Ламберта-Бера), який математично описується рівнянням:

$$I = I_0 e^{-\mu x}$$

де:

I — інтенсивність випромінювання, що пройшло через матеріал;

I_0 — початкова інтенсивність випромінювання;

μ — лінійний коефіцієнт поглинання (залежить від щільності металу);

x — товщина зварного шва.

Наявність у металі дефекту (наприклад, газової пори), щільність якого менша за щільність основного металу, призводить до зменшення локального поглинання μ, що фіксується на рентгенівській плівці або цифровому детекторі як темніша ділянка [3]. Саме такі рентгенограми найчастіше стають основою (датасетами) для систем комп'ютерного зору. Порівняльний аналіз основних методів контролю наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика традиційних методів контролю зварних швів

Метод контролю	Види виявлених дефектів	Переваги	Недоліки
Візуально-вимірювальний (ВК)	Поверхневі (тріщини, підрізи, кратери)	Простота, відсутність складного обладнання	Виявляє лише поверхневі дефекти, високий вплив людського фактора
Радіографічний (РК)	Внутрішні (пори, включення, непровари)	Наочність (отримання знімка), висока точність	Небезпека випромінювання, складність розшифрування знімків експертом
Ультразвуковий (УЗК)	Внутрішні (площинні дефекти, тріщини, непровари)	Висока чутливість, глибина проникнення, безпека	Складність візуалізації результатів, залежність від кваліфікації оператора
Магнітопорошковий (МПК)	Поверхневі та підповерхневі тріщини	Висока чутливість до мікротріщин	Застосовується лише для феромагнітних матеріалів

З аналізу вищезазначених методів та таблиці 1.1 видно, що традиційний підхід до виявлення дефектів має ряд суттєвих недоліків. Основний з них — залежність результатів від кваліфікації, суб'єктивного досвіду та фізичної втоми дефектоскопіста. Аналіз рентгенівських знімків або УЗК-сигналів є рутинною і складною задачею, що потребує значних затрат часу [4].

У зв’язку з розвитком інформаційних технологій та методів комп’ютерного зору з’явилася можливість автоматизувати процес виявлення дефектів за допомогою аналізу зображень зварних швів. Системи автоматизованого контролю якості використовують цифрові камери, сенсори та програмне забезпечення для обробки зображень, що дозволяє здійснювати швидкий і точний аналіз поверхні зварних з’єднань. Одним із перспективних напрямів у цій сфері є використання методів глибинного навчання, які здатні автоматично виділяти характерні ознаки дефектів і класифікувати їх на основі великих наборів навчальних даних.

Методи глибинного навчання, зокрема згорткові нейронні мережі (Convolutional Neural Networks, CNN), широко застосовуються для задач комп’ютерного зору, таких як розпізнавання об’єктів, сегментація зображень та виявлення аномалій. Використання таких моделей у задачах контролю якості зварних швів дозволяє підвищити точність виявлення дефектів, зменшити вплив людського фактора та забезпечити можливість автоматичного моніторингу виробничих процесів у реальному часі.

1.2 Огляд і аналіз існуючих рішень в промисловій дефектоскопії

Стрімкий розвиток апаратного забезпечення та алгоритмів штучного інтелекту зумовив активне впровадження технологій комп’ютерного зору (Computer Vision, CV) у сферу промислового неруйнівного контролю. Головною метою застосування CV у дефектоскопії є мінімізація впливу людського фактора, підвищення швидкості обробки даних та забезпечення стабільної відтворюваності результатів контролю [5].

Система AI Weld Inspector (IUNA) [6] є комплексним рішенням для автоматизованого контролю якості зварних швів, яке поєднує індустріальні камери, сервер обробки даних та програмне забезпечення на основі штучного інтелекту. Вона використовує алгоритми глибинного навчання разом із правилами аналізу геометрії зварного шва. Система здатна автоматично класифікувати зварні шви як придатні або дефектні та визначати типи дефектів. Основні можливості цієї системи: автоматичне виявлення дефектів (тріщини, пори, пропали, асиметрія), вимірювання геометричних параметрів шва, сегментація зображень зварного шва, зберігання статистики та історії перевірок, інтеграція з виробничими системами. Приклад роботи представлено на рисунку 1.2.

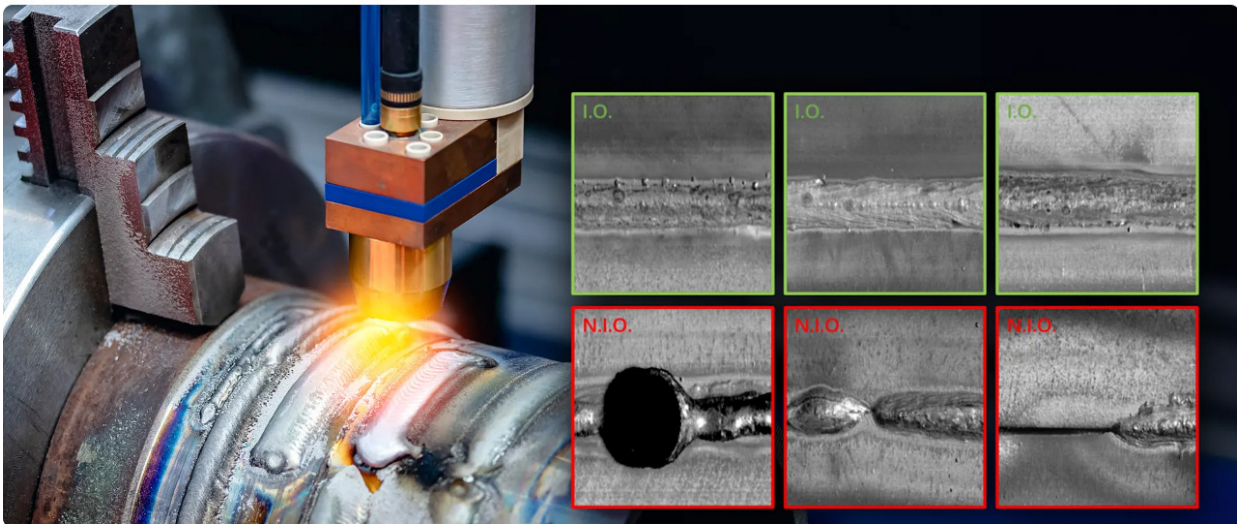

Рисунок 1.2 – Приклад роботи AI Weld Inspector (IUNA)

До переваг AI Weld Inspector належать: висока точність аналізу завдяки використанню нейронних мереж, автоматизація процесу контролю якості, можливість роботи в режимі реального часу, зменшення впливу людського фактору. Недоліками системи є те, що потребує спеціалізованого обладнання (індустріальні камери, освітлення), значна вартість впровадження та необхідність підготовки навчальних даних.

Система Mapvision Weld Seam Inspection (WSI) [7] призначена для автоматизованого контролю якості зварних швів у промислових виробничих лініях. Вона поєднує вимірювання геометрії деталей із аналізом дефектів зварювання за

допомогою алгоритмів штучного інтелекту. Це рішення використовується переважно у автомобільній промисловості та виробництві компонентів електромобілів. Основні її можливості: автоматичне виявлення дефектів (пори, пропуски, кратери, пропали), контроль геометричних параметрів шва, аналіз понад 1200 характеристик деталей за один цикл, використання нейронних мереж для виявлення аномалій. Перевагами є: висока швидкість роботи (аналіз великої кількості параметрів), інтеграція з виробничими роботизованими системами, можливість масштабування для великих виробничих ліній, автоматичне навчання на нових даних. До недоліків відносять: складність впровадження у невеликих підприємствах, потреба у значній кількості навчальних прикладів та висока вартість промислових систем. На рисунку 1.3 представлено головне вікно сайту системи Mapvision Weld Seam Inspection.

Fig 1: The Key Differences between the WSI 2024 and the WSI 2025

Рисунок 1.3 – Головне вікно сайту система Mapvision Weld Seam Inspection

Система Laserax AI Weld Quality Monitoring (рисунок 1.4) призначена для контролю якості лазерного зварювання, зокрема у виробництві акумуляторних модулів для електромобілів. Вона використовує поєднання комп'ютерного зору, фотодіодних сенсорів та алгоритмів штучного інтелекту для аналізу процесу зварювання [8]. Основні її можливості: моніторинг процесу зварювання у реальному часі, автоматичне визначення типів дефектів, збереження історії кожного зварного шва, аналіз причин появи дефектів. До переваг відносять: дуже

низький рівень пропуску дефектів, можливість контролю процесу безпосередньо під час зварювання, повна трасованість виробничих даних, висока точність для лазерного зварювання. Недоліками є те, що система орієнтована на вузьку галузь застосування (лазерне зварювання), складна в інтеграції з іншими типами зварювального обладнання та має значні вимоги до обчислювальних ресурсів.

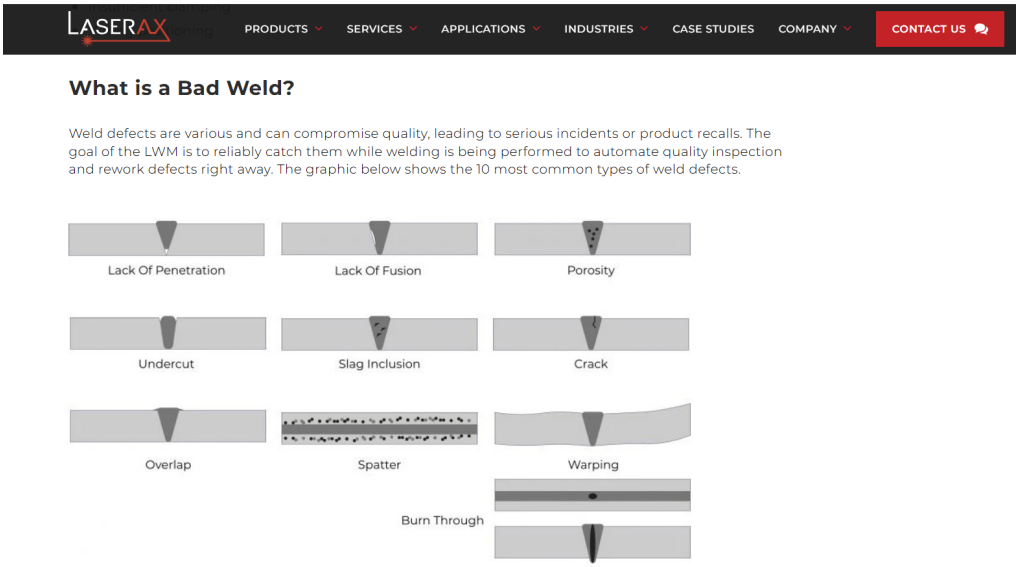

Рисунок 1.4 – Копія вікна сайту системи Laserax AI Weld Quality Monitoring

В таблиці 1.2 представлений порівняльний аналіз описаних систем.

Таблиця 1.2 – Порівняльний аналіз систем контролю якості зварних швів

Критерій	IUNA Weld Inspector	Mapvision WSI	Laserax AI Monitoring
Тип контролю	Візуальний контроль	Комплексний контроль геометрії та дефектів	Контроль процесу зварювання
Використання AI	Так	Так	Так
Робота в реальному часі	Так	Так	Так
Основна сфера застосування	Промислові виробничі лінії	Автомобільна промисловість	Лазерне зварювання
Основний недолік	Вартість та обладнання	Складність інтеграції	Вузька спеціалізація

Аналіз існуючих систем показує, що сучасні рішення для виявлення дефектів зварних швів активно використовують методи комп’ютерного зору та глибинного навчання. Такі системи забезпечують високу точність контролю, автоматизацію процесу перевірки якості та можливість роботи в режимі реального часу.

Разом з тим більшість промислових систем є дорогими та складними для впровадження, що створює потребу у розробці більш доступних програмних модулів на основі глибинного навчання, які можуть ефективно виконувати виявлення дефектів зварних швів із використанням стандартних камер та програмного забезпечення.

У контексті аналізу зварних з'єднань, комп'ютерний зір вирішує завдання автоматизованого розшифрування знімків зварних швів, оптичних фотографій поверхонь швів або ультразвукових сканограм. Загальний процес обробки візуальної інформації в таких системах можна подати у вигляді послідовних етапів представлених на рисунку 1.5.

Рисунок 1.5 – Структура системи комп'ютерного зору для виявлення дефектів

Важливим етапом є попередня обробка (препроцесинг) зображень. Оскільки промислові знімки зварних швів часто характеризуються низьким контрастом, наявністю шумів та нерівномірним освітленням, алгоритми комп'ютерного зору застосовують просторову фільтрацію для покращення якості вхідних даних [6].

Математичною основою більшості методів виділення ознак (як у класичному CV, так і в згорткових нейромережах) є операція дискретної згортки (convolution) зображення з певним ядром (фільтром). Для двовимірного зображення I та ядра згортки K розміром M×N, значення нового пікселя G(i,j) обчислюється за формулою:

$$G(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i-m, j-n) K(m,n)$$

де:

$G(i,j)$ — значення пікселя вихідного (відфільтрованого) зображення;

I — матриця пікселів вхідного зображення;

K — матриця вагових коефіцієнтів (ядро згортки);

m, n — індекси елементів ядра згортки.

Змінюючи вагові коефіцієнти матриці K, можна реалізувати різні ефекти: згладжування шумів (наприклад, фільтр Гауса), підкреслення контурів (оператори Собеля, Прюітта) або локалізацію різких перепадів яскравості, які часто свідчать про наявність тріщин чи пор у металі [9].

До появи алгоритмів глибинного навчання (Deep Learning), системи комп'ютерного зору покладалися на «ручне» конструювання ознак (Hand-crafted features). Експерти математично описували, як виглядає дефект (за допомогою алгоритмів SIFT, HOG, LBP), після чого використовували класичні алгоритми машинного навчання (SVM, випадковий ліс) для їх класифікації [10]. Однак складність форми дефектів зварних швів значно обмежувала точність таких систем.

В останні роки парадигма змістилася в бік глибинного навчання. Порівняння класичного підходу та глибинного навчання наведено у таблиці 1.3.

Таблиця 1.3 – Порівняння класичного комп'ютерного зору та глибинного навчання у дефектоскопії

Критерій порівняння	Класичний комп'ютерний зір (Традиційне ML)	Глибинне навчання (Deep Learning)
Виділення ознак дефектів	Виконується вручну (математичне моделювання експертом)	Виконується автоматично нейромережею під час навчання
Вимоги до набору даних	Може працювати на невеликих вибірках даних	Потребує великих анотованих наборів даних (датасетів)
Стійкість до шумів	Низька (потребує складної попередньої обробки знімка)	Висока (модель сама вчиться ігнорувати артефакти)
Обчислювальні ресурси	Низькі (може працювати на слабких процесорах)	Високі (потребує сучасних графічних процесорів – GPU)
Масштабованість	Складно адаптувати під нові типи зварних швів чи дефектів	Легко донавчається (Transfer Learning) на нових даних

Таким чином, використання методів глибинного навчання у задачах комп'ютерного зору дозволяє подолати обмеження класичних алгоритмів. Згорткові нейронні мережі здатні самостійно виявляти складні ієрархічні ознаки дефектів на рентгенограмах зварних швів, що робить їх оптимальним інструментом для розробки автоматизованого програмного модуля.

Сучасні системи розпізнавання об'єктів (Object Detection) на базі глибинного навчання вирішують дві одночасні задачі: локалізацію (визначення координат обмежувальної рамки — bounding box, у якій знаходиться дефект) та класифікацію цього дефекту. Архітектурно всі нейромережеві моделі виявлення об'єктів поділяють на два основні класи (рисунок 1.6): двостадійні (Two-stage) та одностадійні (One-stage) детектори [9].

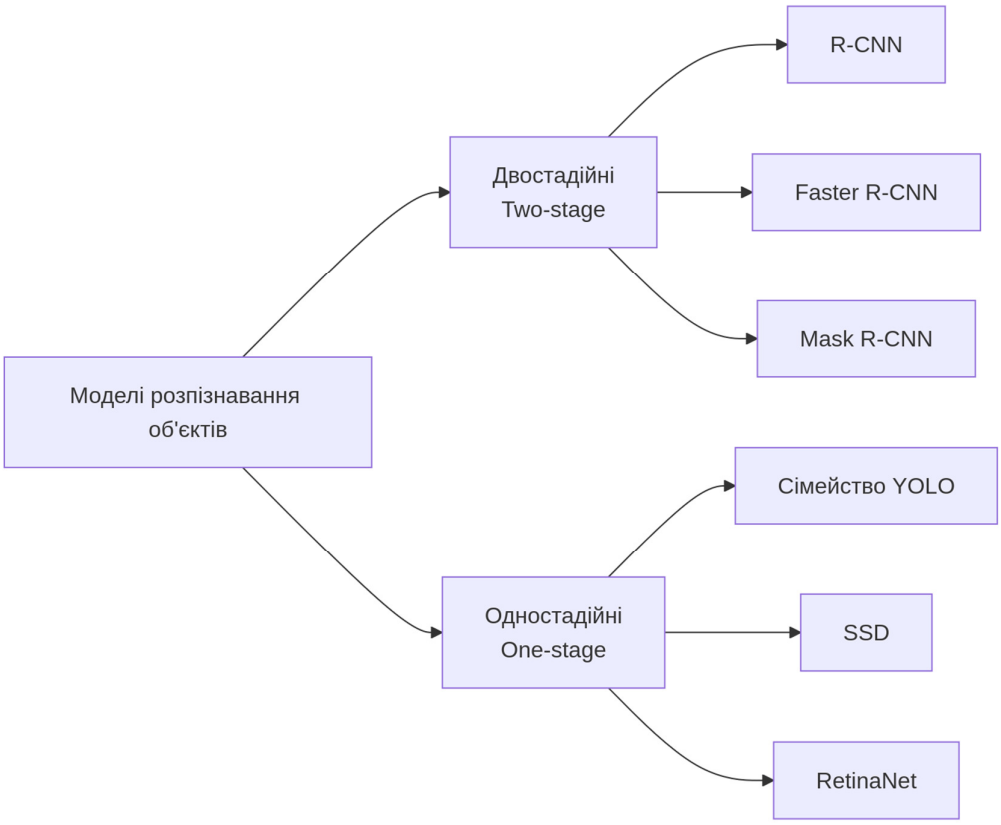

Рисунок 1.6 – Класифікація архітектур нейронних мереж для розпізнавання об'єктів

Двостадійні моделі (Two-stage detectors) Представники цього класу (наприклад, Faster R-CNN) працюють у два етапи. На першому етапі спеціальна підмережа (Region Proposal Network, RPN) генерує гіпотези щодо регіонів зображення, де потенційно може знаходитися дефект зварного шва. На другому етапі ці виділені регіони класифікуються, а їхні межі точно коригуються. Такі моделі забезпечують дуже високу точність (що критично для дрібних пор чи мікротріщин), проте характеризуються складною архітектурою та низькою швидкістю обробки, що ускладнює їх використання в режимі реального часу [10].

Одностадійні моделі (One-stage detectors) Моделі сімейства YOLO (You Only Look Once) та SSD (Single Shot MultiBox Detector) аналізують зображення за один прохід нейромережі. Зображення розбивається на умовну сітку, і для кожної комірки одночасно прогнозуються ймовірності класів та координати обмежувальних рамок. Це забезпечує екстремально високу швидкість обробки даних (Real-time detection) [11].

Для оцінки якості локалізації дефектів усіма сучасними моделями використовується математична метрика IoU (Intersection over Union — перетин над об'єднанням). Вона обчислюється як відношення площі перетину спрогнозованої нейромережею рамки та еталонної рамки (розміченої експертом) до площі їх об'єднання [12]:

$$IoU = \frac{Area(B_p \cap B_{gt})}{Area(B_p \cup B_{gt})}$$

де IoU — значення метрики перетину над об'єднанням;

B_p — координати спрогнозованої обмежувальної рамки (Predicted Bounding Box);

B_{gt} — координати еталонної обмежувальної рамки (Ground Truth Bounding Box);

A_{rea} — функція обчислення площі.

Вибір конкретної архітектури для розробки програмного модуля вимагає компромісу між точністю локалізації дефекту (mAP — Mean Average Precision) та швидкістю роботи алгоритму (FPS — Frames Per Second).

Порівняльний аналіз найпопулярніших архітектур наведено у таблиці 1.4.

Таблиця 1.4 – Порівняльна характеристика сучасних моделей розпізнавання об'єктів

Характеристика	Faster R-CNN	SSD	YOLOv8 / YOLOv10
Архітектурний підхід	Двостадійний	Одностадійний	Одностадійний
Швидкість обробки (FPS)	Низька (5-15)	Середня (20-40)	Висока (50-100+)
Точність розпізнавання (mAP)	Дуже висока	Середня	Висока
Робота з дрібними дефектами	Відмінно	Задовільно	Добре (завдяки механізмам уваги)
Вимоги до апаратного забезпечення	Високі (потужні GPU)	Середні	Оптимізовані (наявні легкі nano-версії)

Як видно з таблиці 2.4, архітектура Faster R-CNN забезпечує високу точність, проте її використання обмежене високими вимогами до обчислювальних потужностей. Натомість сучасні ітерації моделей сімейства YOLO (зокрема версії YOLOv8 або новіші) демонструють найкращий баланс між швидкістю інференсу та точністю [13]. Вони здатні ефективно розпізнавати різні типи дефектів зварних швів навіть на комп'ютерах із середньою продуктивністю, що робить їх найбільш доцільним та раціональним вибором для реалізації базового алгоритму програмного модуля.

1.3 Постановка задачі дослідження

На основі проведеного аналізу традиційних методів неруйнівного контролю та сучасних технологій комп'ютерного зору і глибинного навчання, встановлено, що автоматизація процесу виявлення дефектів зварних швів є актуальною проблемою. Існуючі ручні методи розшифрування знімків зварних швів є повільними та сильно залежать від кваліфікації оператора-дефектоскопіста, тоді як використання згорткових нейронних мереж здатне значно підвищити швидкість та об'єктивність контролю.

Виходячи з цього, головною метою даної кваліфікаційної роботи є розробка програмного модуля для автоматизованого виявлення та класифікації дефектів зварних швів на цифрових зображеннях (рентгенограмах) із використанням алгоритмів глибинного навчання (зокрема, архітектури сімейства YOLO).

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати предметну область контролю якості зварних швів;
- дослідити сучасні методи комп'ютерного зору та глибинного навчання для виявлення дефектів на зображеннях;
- провести аналіз існуючих програмних рішень для детектування дефектів зварних з'єднань;
- розробити архітектуру програмного модуля;

- здійснити програмну реалізацію програмного модуля для виявлення та класифікації дефектів зварних швів;
- провести експериментальні дослідження програмного модуля та оцінити ефективність запропонованого рішення.

Вирішення цих завдань дозволить створити ефективний інструментарій, який може бути використаний як допоміжна система підтримки прийняття рішень для фахівців із неруйнівного контролю на промислових підприємствах.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМНОГО МОДУЛЯ

2.1 Загальна структура проєктованої системи

Основою для успішної реалізації програмного модуля виявлення дефектів зварних швів є проєктування надійної, масштабованої та гнучкої архітектури. Зважаючи на вимоги до сучасного програмного забезпечення, що інтегрує моделі машинного навчання (Machine Learning, ML), найдоцільнішим підходом є використання клієнт-серверної архітектури з розділенням бізнес-логіки, моделі глибинного навчання та інтерфейсу користувача [14].

Проєктована система складається з трьох основних логічних рівнів:

1. Клієнтський рівень (Presentation Layer) — графічний інтерфейс користувача (GUI) для взаємодії з системою. На цьому рівні відбувається завантаження рентгенограм зварних швів, налаштування параметрів та перегляд результатів детекції.
2. Серверний рівень (Application/API Layer) — програмний інтерфейс (RESTful API), який приймає запити від клієнта, здійснює попередню обробку зображень, маршрутизує дані та формує фінальну відповідь.
3. Рівень машинного навчання (ML Layer) — ізольований модуль, що містить навчену згорткову нейронну мережу, яка виконує безпосереднє розпізнавання об'єктів на отриманих тензорах зображень [15].

Структурна схема взаємодії компонентів розроблюваної системи приведена на рисунку 2.1.

З метою ізоляції середовища виконання та забезпечення зручності розгортання модуля на Linux-серверах (зокрема, у віртуалізованих середовищах), серверну та ML-частини доцільно інкапсулювати у Docker-контейнери. Це дозволяє уникнути конфліктів залежностей (особливо бібліотек комп'ютерного зору та CUDA-драйверів для GPU) і спрощує перенесення проєкту [16].

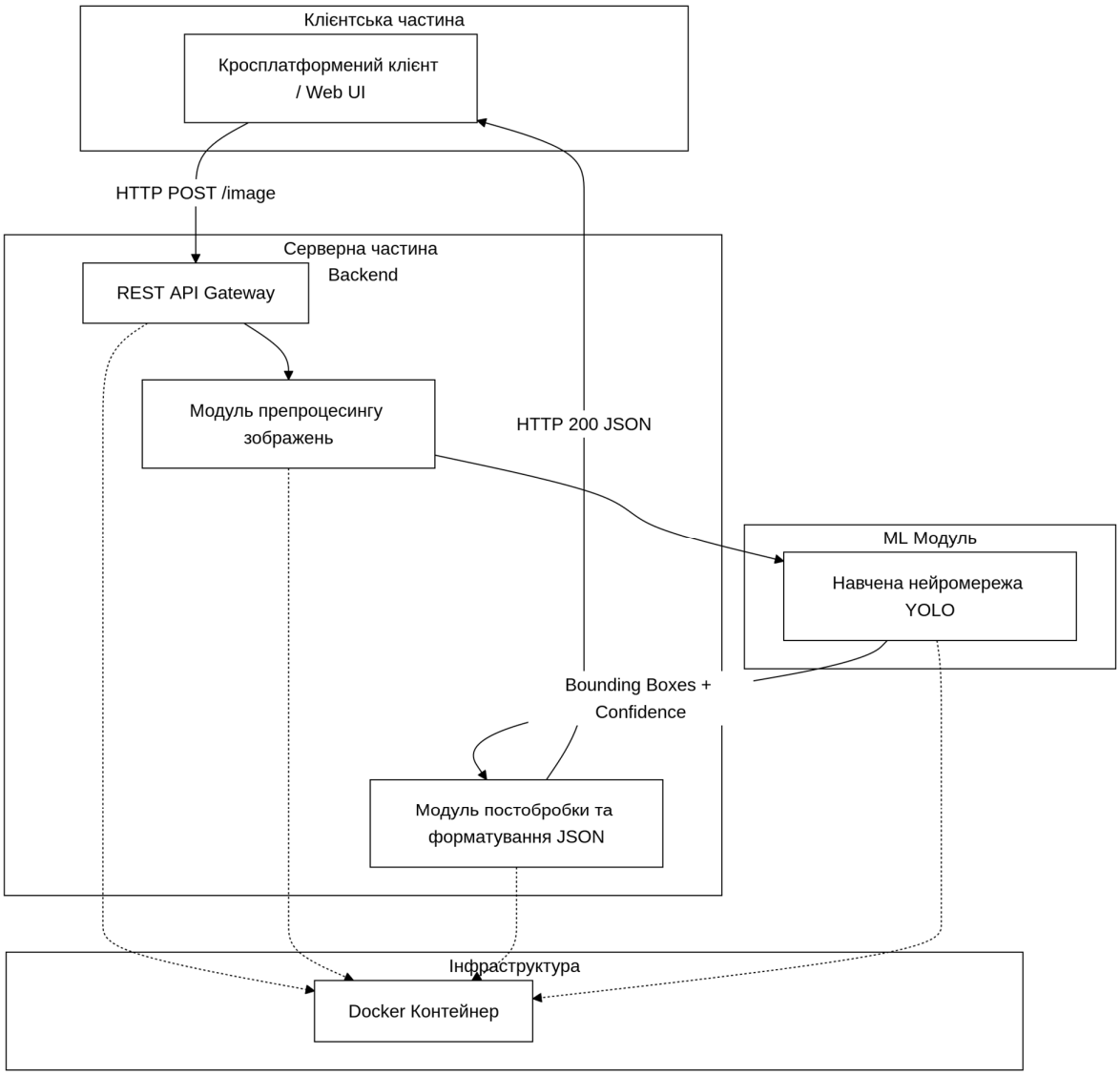

Рисунок 2.1 – Загальна структура програмного модуля

Важливим критерієм проєктування архітектури є загальний час очікування відповіді системи (T_{total}), який для систем реального або квазіреального часу описується рівнянням:

$$T_{total} = t_{net} + t_{pre} + t_{inf} + t_{post}$$

де T_{total} — загальний час обробки одного зображення;

t_{net} — час передачі даних по мережі між клієнтом та сервером;

t_{pre} — час попередньої обробки зображення (зміна розміру, нормалізація);

t_{inf} — час інференсу (роботи моделі глибинного навчання);

t_{post} — час постобробки результатів (наприклад, Non-Maximum Suppression).

Для мінімізації t_{inf} ML-модуль повинен використовувати оптимізовані тензорні обчислення, а клієнтський додаток — асинхронні запити.

Для забезпечення надійної взаємодії між клієнтським додатком та модулем глибинного навчання у спроєктованій системі використовується архітектурний стиль REST (Representational State Transfer). Передача рентгенографічних знімків від клієнта до сервера здійснюється за допомогою HTTP-запитів типу POST у форматі multipart/form-data. Це дозволяє ефективно передавати бінарні дані великого обсягу без необхідності їхнього попереднього кодування у формат Base64, що значно зменшує навантаження на мережу.

Враховуючи те, що робота нейромережі може займати певний час, особливо при обробці зображень високої роздільної здатності, серверна частина реалізована з використанням асинхронної моделі обробки запитів (Asynchronous I/O). Завдяки фреймворку FastAPI, основний потік сервера не блокується під час очікування результатів від GPU або процесора, що дозволяє системі паралельно приймати нові зображення від інших користувачів.

Крім того, для організації взаємодії та управління контейнерами використовується засіб Docker Compose Він дозволяє за допомогою одного конфігураційного файлу docker-compose.yml одночасно розгортати всі мікросервіси системи: веб-сервер, модуль машинного навчання та, за необхідності, базу даних для збереження історії перевірок. Такий підхід гарантує повну відтворюваність середовища на будь-якому обладнанні — від локального ноутбука розробника до хмарного сервера.

Для реалізації описаної архітектури було проведено аналіз сучасних інструментів розробки. Результати вибору технологічного стеку представлено у таблиці 2.1.

Обраний технологічний стек (представлений у таблиці 2.1) та модульна архітектура забезпечують високу продуктивність, здатність до горизонтального масштабування та легкість підтримки кодової бази в майбутньому. Такий підхід створює надійне підґрунтя для наступного етапу — розробки алгоритмічного забезпечення процесу розпізнавання дефектів зварних швів.

Таблиця 2.1 – Технологічний стек для розробки програмного модуля

Компонент системи	Обрана технологія / Фреймворк	Обґрунтування вибору
Мова програмування (Backend)	Python 3.10+	Основний стандарт для розробки систем ML/AI, наявність потужних бібліотек для роботи з даними.
Веб-фреймворк (API)	FastAPI (або Flask)	Висока швидкодія, асинхронність, автоматична генерація документації (Swagger), легка інтеграція з ML-моделями.
Комп'ютерний зір та ML	PyTorch, OpenCV, Ultralytics	PyTorch забезпечує гнучку роботу з нейромережами, Ultralytics надає оптимізовані інструменти для YOLO, OpenCV — ефективний препроцесинг.
Клієнтська частина (UI)	Flutter (або HTML/JS)	Забезпечує кросплатформенність. Єдина кодова база дозволяє легко компілювати додаток під різні платформи (Web, Desktop, Mobile).
Контейнеризація	Docker	Стандартизація середовища розгортання, незалежність від операційної системи хоста.

Структурна схема системи визначає основні програмні модулі та зв'язки між ними. Як показано на рисунку 2.2, програмний комплекс декомпоновано на три фізично незалежні рівні. Клієнтський рівень представлений iOS-додатком (SwiftUI), який відповідає за відображення графічного інтерфейсу та взаємодію з користувачем. Серверний рівень (FastAPI) інкапсульований у Docker-контейнер і виконує роль шлюзу (API Gateway), обробляючи вхідні HTTP-запити. Рівень машинного навчання (ML-модуль на базі PyTorch) є ізольованим ядром, яке завантажує попередньо навчені ваги моделі YOLO та виконує ресурсомісткі тензорні обчислення.

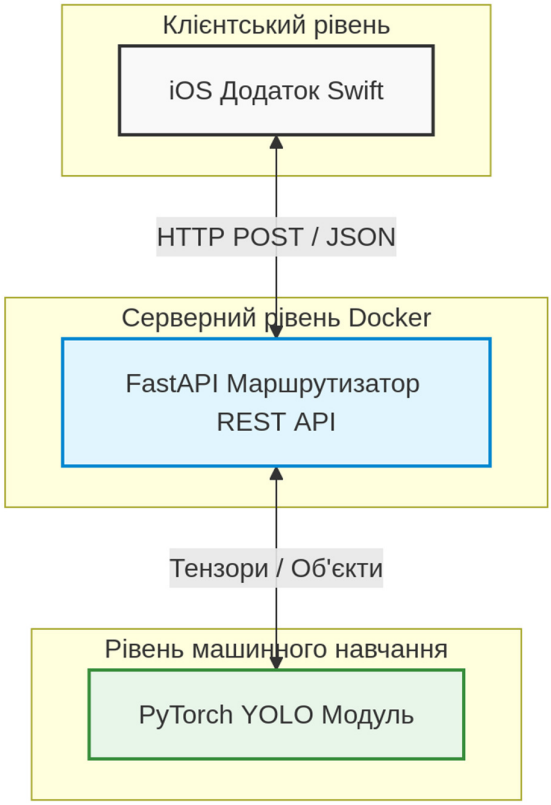

Рисунок 2.2 – Структурна схема програмного модуля

Для моделювання інформаційних процесів та відстеження життєвого циклу даних у системі побудовано діаграму потоків даних (Data Flow Diagram, DFD), яку наведено на рисунку 2.3. Процес розпочинається із зовнішньої сутності «Користувач», який ініціює передачу файлу рентгенограми. Система послідовно виконує процес прийому запиту, виділяючи бінарні дані зображення, після чого передає їх до блоку препроцесингу та інференсу. Цей блок взаємодіє зі сховищем даних (файлом ваг моделі best.pt) і генерує масив координат виявлених дефектів. На фінальному етапі процес формування відповіді серіалізує ці дані у формат JSON та повертає їх для візуалізації.

Динамічний аспект роботи системи та часову послідовність викликів між її компонентами проілюстровано за допомогою діаграми послідовності (Sequence Diagram), представленої на рисунку 2.4.

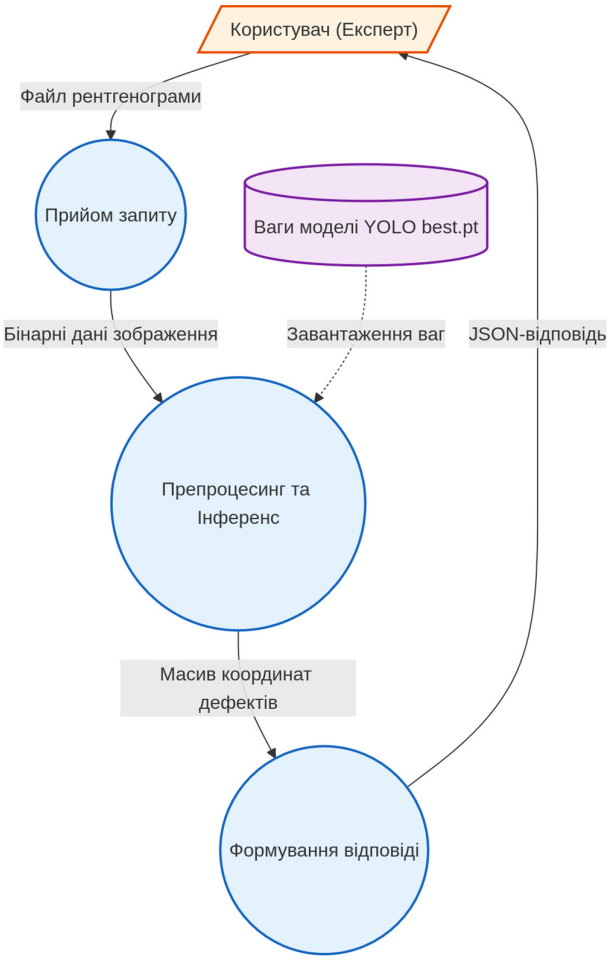

Рисунок 2.3 – Діаграма потоків даних (DFD) процесу розпізнавання

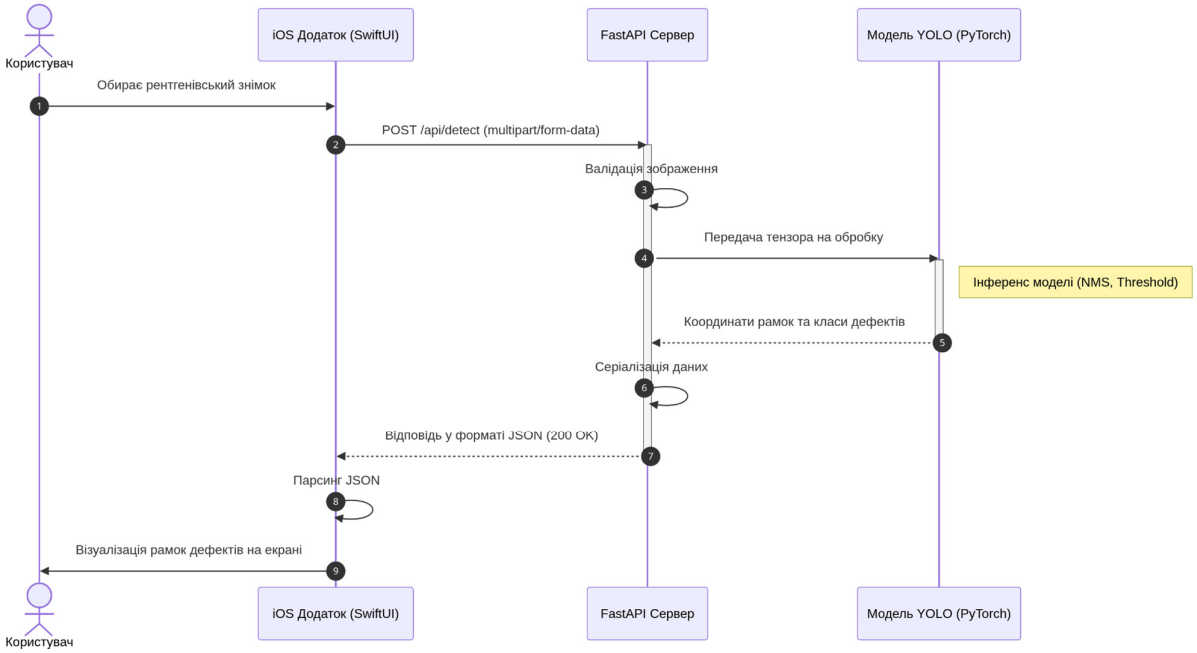

Рисунок 2.4 – Діаграма послідовності взаємодії компонентів системи

Сценарій використання розпочинається з вибору зображення користувачем у мобільному додатку. Далі iOS-клієнт формує POST-запит (у форматі multipart/form-data) і відправляє його на сервер. Backend-сервер (FastAPI) здійснює первинну валідацію файлу та передає тензор зображення до ML-модуля. Після завершення прямого проходу нейромережі (інференсу) та застосування алгоритму придушення немаксимумів (NMS), сервер отримує координати рамок, формує JSON-документ і надсилає HTTP-відповідь зі статусом 200 OK. Завершується життєвий цикл парсингом отриманих даних на стороні мобільного клієнта та миттєвим рендерингом графічних обмежувальних рамок поверх оригінального знімка на екрані смартфона.

Такий підхід до проєктування створює надійне підґрунтя для наступного етапу — розробки алгоритмічного забезпечення процесу розпізнавання дефектів зварних швів.

2.2 Алгоритмічне забезпечення процесу розпізнавання

Основою програмного модуля для виявлення дефектів зварних швів є алгоритм розпізнавання об'єктів за один прохід (one-stage object detection), який реалізується за допомогою архітектури сімейства YOLO. На відміну від традиційних алгоритмів, які використовують метод ковзного вікна або попереднє виділення регіонів, YOLO перетворює задачу детекції на єдину регресійну проблему [18]. Це дозволяє одночасно прогнозувати координати обмежувальних рамок (bounding boxes) та ймовірності належності дефекту до певного класу (наприклад, «пора», «тріщина», «шлакове включення»).

Алгоритмічний процес розпізнавання дефекту на рентгенограмі зварного шва складається з наступних кроків:

1. Розбиття зображення на сітку (Grid Division). Вхідне зображення зварного шва розміром W×H пікселів розбивається на умовну сітку розміром S×S. Якщо геометричний центр дефекту потрапляє в певну комірку цієї сітки, саме ця комірка стає «відповідальною» за його виявлення.

2. Прогнозування обмежувальних рамок (Bounding Box Prediction).

Кожна комірка сітки прогнозує B обмежувальних рамок. Для кожної рамки нейромережа обчислює 5 параметрів: координати центру (x,y) відносно меж комірки, ширину та висоту (w,h) відносно всього зображення, а також показник достовірності (Confidence score).

Показник достовірності відображає впевненість моделі в тому, що в даній рамці дійсно знаходиться дефект, і наскільки точно ця рамка його охоплює. Математично він обчислюється за формулою [12]:

$$Confidence = Pr(Object) \times IoU_{pred}^{truth}$$

де: *Pr*(Object) — ймовірність наявності об'єкта (дефекту) в комірці (дорівнює 1, якщо об'єкт є, і 0, якщо немає);
IoU — значення перетину над об'єднанням (Intersection over Union) між спрогнозованою рамкою та еталонною.

3. Обчислення ймовірностей класів (Class Probability). Паралельно з координатами, кожна комірка прогнозує умовні ймовірності C для кожного класу дефекту Pr(Class|Object). Під час тестування (інференсу) ці умовні ймовірності множаться на показник достовірності рамки, утворюючи специфічну для класу достовірність (Class-specific confidence score):

$$Score_i = Pr(Class_i|Object) \times Pr(Object) \times IoU_{pred}^{truth} = Pr(Class_i) \times IoU_{pred}^{truth}$$

де: $Score_i$ — фінальний показник впевненості моделі в тому, що знайдений об'єкт належить до i-го класу дефектів (наприклад, до класу «тріщина»);
$Pr(Class_i)$ — загальна ймовірність i-го класу [12].

Концепція якірних рамок (Anchor Boxes) Важливим алгоритмічним вдосконаленням, що використовується у розроблюваному модулі, є застосування механізму якірних рамок (Anchor Boxes). Оскільки дефекти зварних швів мають різну специфічну геометрію (наприклад, тріщини зазвичай витягнуті і вузькі, а газові пори — круглі), мережі складно прогнозувати довільні координати «з нуля». Замість цього алгоритм використовує набір попередньо визначених рамок із різним

співвідношенням сторін (aspect ratios). Нейромережа прогнозує не самі координати, а лише необхідні зміщення (офсети) відносно цих базових якірних рамок, що значно пришвидшує збіжність моделі під час навчання та підвищує точність локалізації [25].

Функція втрат (Loss Function) Для оптимізації ваг нейромережі у процесі навчання алгоритм використовує комплексну функцію втрат (Loss Function), яка об'єднує в собі три складові. Загальна помилка моделі обчислюється як зважена сума:

$$L_{total} = \lambda_{coord} L_{coord} + L_{conf} + L_{class}$$

де: L_{coord} — помилка локалізації (Bounding Box Regression Loss), яка штрафує модель за неточне визначення координат центру дефекту та його розмірів. Зазвичай обчислюється на основі метрики CIoU (Complete Intersection over Union);

L_{conf} — помилка достовірності (Confidence Loss), яка оцінює, наскільки правильно модель визначає наявність або відсутність об'єкта в певній рамці;

L_{class} — помилка класифікації (Classification Loss), що обчислюється за допомогою перехресної ентропії (Cross-Entropy) і відповідає за правильність визначення типу дефекту (пора, тріщина тощо);

λ_{coord} — ваговий коефіцієнт, що збільшує значущість помилки координат порівняно з іншими втратами [27].

Алгоритм придушення немаксимумів (Non-Maximum Suppression, NMS). Оскільки один і той самий дефект зварного шва (особливо великий, як-от довга тріщина або непровар) може бути виявлений одразу кількома сусідніми комірками сітки, алгоритм генерує багато дублюючих рамок. Для усунення цього надлишку використовується метод NMS. Він сортує всі спрогнозовані рамки за значенням $Score_i$, обирає рамку з найвищим показником і видаляє всі інші рамки для цього ж класу, якщо їхній IoU з обраною рамкою перевищує заданий поріг (наприклад, 0.5) [21].

Візуалізація загального алгоритмічного конвеєра (pipeline) процесу розпізнавання дефектів наведена на рисунку 2.5.

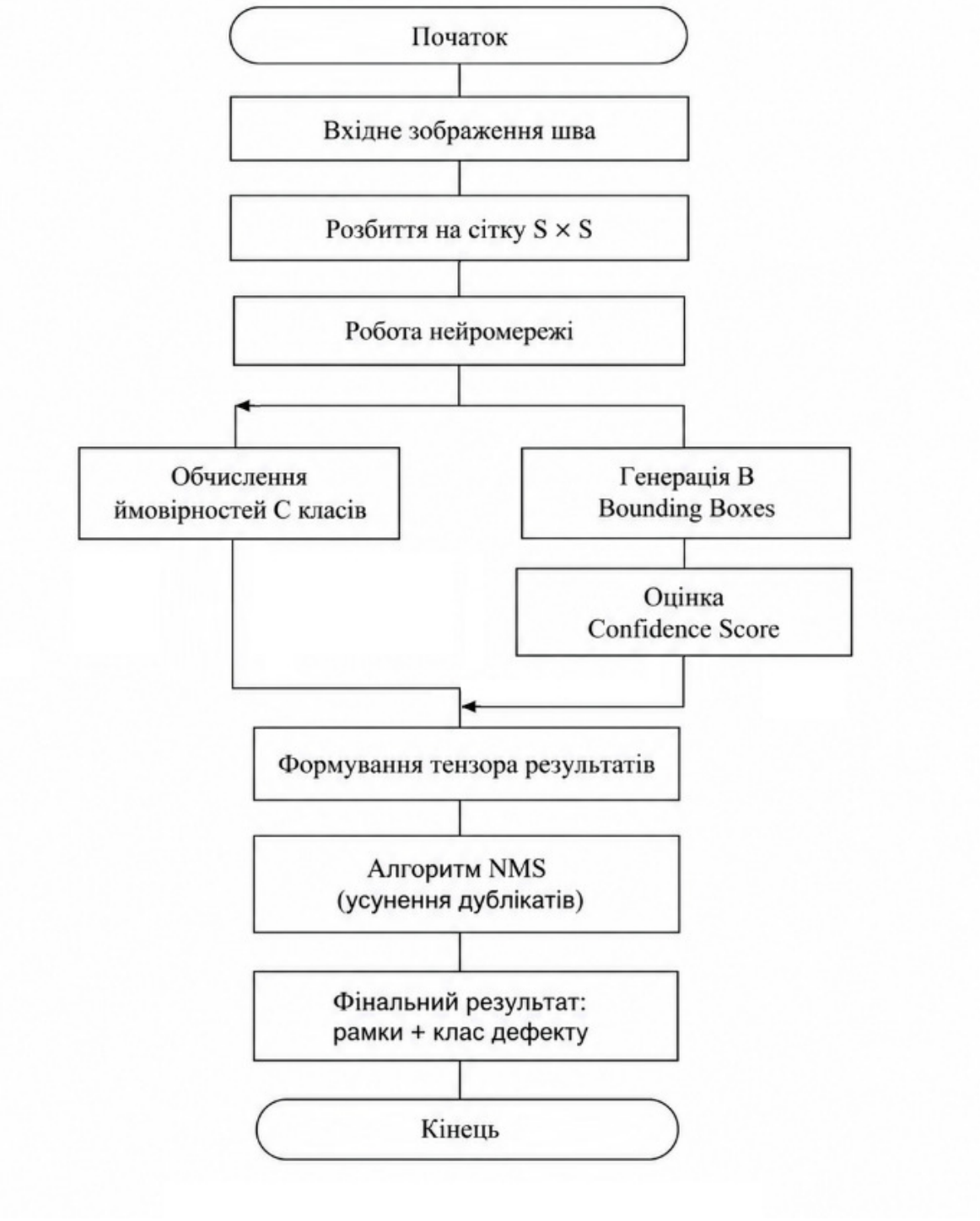

Рисунок 2.5 – Алгоритм виявлення дефектів зварного шва

Використання описаного алгоритмічного забезпечення дозволяє програмному модулю обробляти рентгенографічні зображення з високою точністю та мінімальними затримками. Сформований математичний апарат є базою для переходу до наступного етапу — підготовки інформаційного забезпечення, а саме збору та розмітки набору даних для навчання моделі.

2.3 Інформаційне забезпечення програмного модуля

Ефективність та точність роботи алгоритмів глибинного навчання критично залежить від якості, обсягу та збалансованості репрезентативного набору даних (датасету). Організація інформаційного забезпечення для розроблюваного програмного модуля включає чотири основні етапи: збір сирих даних, їх анотування (розмітку), попередню обробку (препроцесинг) та аугментацію [25].

Збір та анотування даних. Для формування навчальної вибірки доцільно використовувати відкриті бази індустріальних знімків зварних швів, такі як GDXray (Grima X-ray database), а також власні знімки, отримані з промислових підприємств.

Оскільки архітектура YOLO належить до класу керованого навчання (Supervised Learning), кожне зображення в датасеті повинно супроводжуватися текстовим файлом анотації. Процес розмітки полягає у виділенні експертом-дефектоскопістом обмежувальних рамок (bounding boxes) навколо дефектів за допомогою спеціалізованого програмного забезпечення (наприклад, CVAT, LabelImg або Roboflow) [25].

Формат анотації YOLO вимагає нормалізації координат рамок відносно розмірів зображення. Кожен рядок у текстовому файлі анотації має таку структуру:

<class_id> <x_center> <y_center> <width> <height>.

Математично нормалізація координат обчислюється за формулами:

$$x_{norm} = \frac{x_{center}}{W}, \quad y_{norm} = \frac{y_{center}}{H}$$

$$w_{norm} = \frac{w}{W}, \quad h_{norm} = \frac{h}{H}$$

де x_{norm}, y_{norm} — нормалізовані координати центру дефекту;

w_{norm}, h_{norm} — нормалізована ширина та висота обмежувальної рамки;

W, H — абсолютна ширина та висота оригінального зображення в пікселях.

Промислові рентгенограми часто мають нестандартні розміри та відрізняються за рівнем контрастності. Тому на етапі препроцесингу всі зображення масштабуються до єдиного стандарту (наприклад, 640x640 пікселів), а значення їхніх пікселів нормалізуються (приводяться до діапазону [0, 1]) для прискорення збіжності градієнтного спуску під час навчання нейромережі.

З метою запобігання перенавчанню (overfitting) моделі та підвищення її стійкості до зашумлених або нестандартних знімків застосовується метод аугментації даних (Data Augmentation) [22]. Цей метод полягає у штучному збільшенні обсягу датасету шляхом застосування до оригінальних зображень різноманітних геометричних та фотометричних трансформацій. Параметри аугментації, обрані для розроблюваної системи, зведено у таблицю 2.2.

Таблиця 2.2 – Методи та параметри аугментації набору даних

Метод аугментації	Опис перетворення	Значення параметра
Геометричні трансформації		
Горизонтальне віддзеркалення (Flip)	Відображення знімка по горизонталі	Ймовірність 50%
Випадкове обертання (Rotation)	Обертання знімка на випадковий кут	В межах $\pm 15^{\circ}$
Масштабування (Scale)	Зміна розміру зображення із збереженням центру	$\pm 20\%$
Фотометричні трансформації		
Зміна яскравості (Brightness)	Імітація різної інтенсивності випромінювання	В межах $\pm 25\%$
Зміна контрастності (Contrast)	Імітація різної щільності матеріалу	$\pm 15\%$
Додавання шуму (Gaussian Noise)	Імітація артефактів цифрових детекторів	Дисперсія 0.01 - 0.05

Загальний процес підготовки інформаційного забезпечення схематично зображено на рисунку 2.3.

Після проведення всіх етапів обробки та аугментації, фінальний набір даних випадковим чином розділяється на три вибірки: навчальну (Train — 70%), валідаційну (Validation — 20%) та тестову (Test — 10%) [26]. Навчальна вибірка використовується для оптимізації ваг нейромережі, валідаційна — для налаштування гіперпараметрів у процесі навчання, а тестова — для об'єктивної оцінки точності готової моделі на даних, які вона раніше не «бачила». Сформована інформаційна база дозволяє розпочати етап безпосередньої програмної реалізації та навчання моделі.

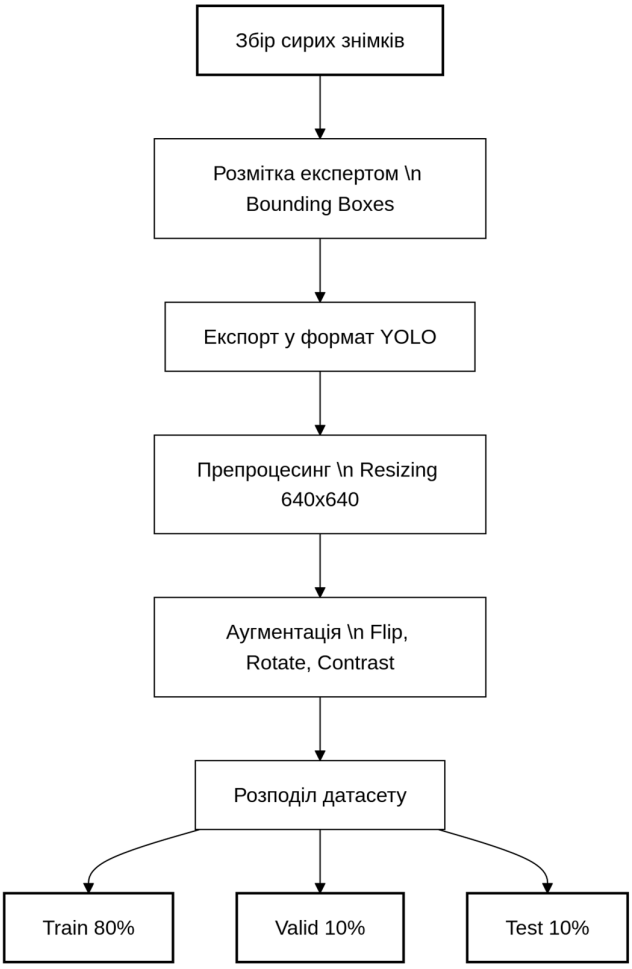

Рисунок 2.3 – Процес підготовки та організації набору даних

Вирішення проблеми дисбалансу класів Специфікою формування датасетів у промисловій дефектоскопії є виражений дисбаланс класів (Class Imbalance). У

реальних умовах деякі дефекти (наприклад, газові пори) зустрічаються на рентгенограмах значно частіше, ніж критичні, але рідкісні дефекти (наприклад, внутрішні мікротріщини). Якщо не враховувати цей фактор, нейромережа може перенавчитися на домінуючий клас і почати ігнорувати рідкісні дефекти.

Для нівелювання цієї проблеми в системі алгоритмічно застосовуються два підходи:

1. Вагове балансування функцій втрат (Focal Loss) — алгоритм автоматично надає більшої ваги помилкам, допущеним при розпізнаванні складних або рідкісних класів.

2. Цільова аугментація — штучне створення більшої кількості копій (з геометричними змінами) саме для тих знімків, де наявні рідкісні типи дефектів [28].

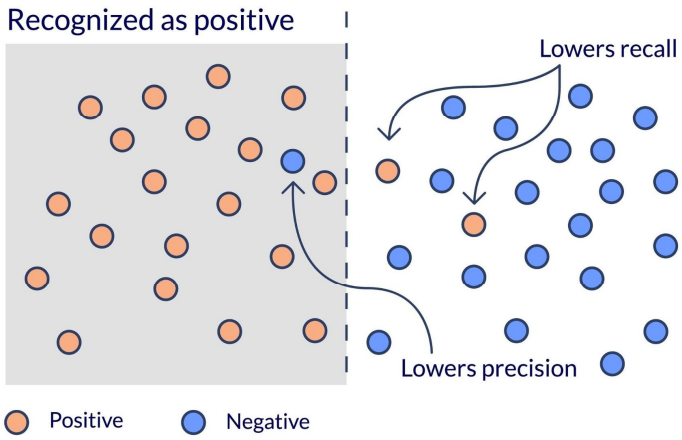

Рисунок 2.4 – Метрики оцінювання ефективності моделі

Для всебічної перевірки алгоритмічного забезпечення на тестовій вибірці використовуються класичні метрики машинного навчання. Окрім загальної точності (mAP), розраховуються показники точності (Precision) та повноти (Recall) [29]:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

де *TP* (True Positives) — кількість правильно виявлених дефектів;

FP (False Positives) — хибні спрацьовування (коли система сприйняла нормальний шов за дефект);

FN (False Negatives) — пропущені дефекти (коли система не помітила реальний дефект).

У дефектоскопії метрика Recall (Повнота) є критично важливою, оскільки пропуск реального дефекту (тріщини) може призвести до аварії, тоді як хибне спрацьовування (FP) лише вимагатиме додаткової перевірки шва інспектором.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ НА ОСНОВІ ГЛИБИННОГО НАВЧАННЯ

3.1 Програмна реалізація алгоритмів обробки зображень та навчання моделі

Програмна реалізація модуля виявлення дефектів зварних швів виконана з використанням мови програмування Python (версії 3.10), яка є індустріальним стандартом для задач машинного навчання. В якості базового фреймворку для роботи з тензорними обчисленнями та нейронними мережами обрано PyTorch. Безпосередня ініціалізація, навчання та оцінка моделі архітектури YOLO здійснюється за допомогою спеціалізованої бібліотеки Ultralytics, яка надає оптимізований високорівневий програмний інтерфейс (API).

Для ефективного вирішення задачі було застосовано метод донавчання (Fine-tuning) на основі концепції Transfer Learning. Для формування інформаційної бази було використано відкритий набір даних «Welding Defect - Object Detection», розміщений на платформі Kaggle [30]. Цей датасет містить зібрані та попередньо анотовані цифрові зображення зварних швів із різними типами поверхневих та внутрішніх дефектів. Формат розмітки повністю адаптований під архітектуру YOLO: кожне зображення супроводжується текстовим файлом із нормалізованими координатами обмежувальних рамок (bounding boxes) для цільових класів дефектів. Приклади вхідних зображень із використаного набору даних наведено на рисунку 3.1.

Першим етапом програмної реалізації є формування структури директорій датасету, яка має відповідати строгим вимогам архітектури YOLO. Набір даних (зображення рентгенограм та відповідні їм текстові файли розмітки) програмно розподіляється на навчальну (train) та валідаційну (val) вибірки.

Для конфігурації інформаційного забезпечення та маршрутизації шляхів використовується конфігураційний файл data.yaml (лістинг 3.1), який описує актуальні відносні шляхи до папок із зображеннями та ідентифікатори цільових класів дефектів.

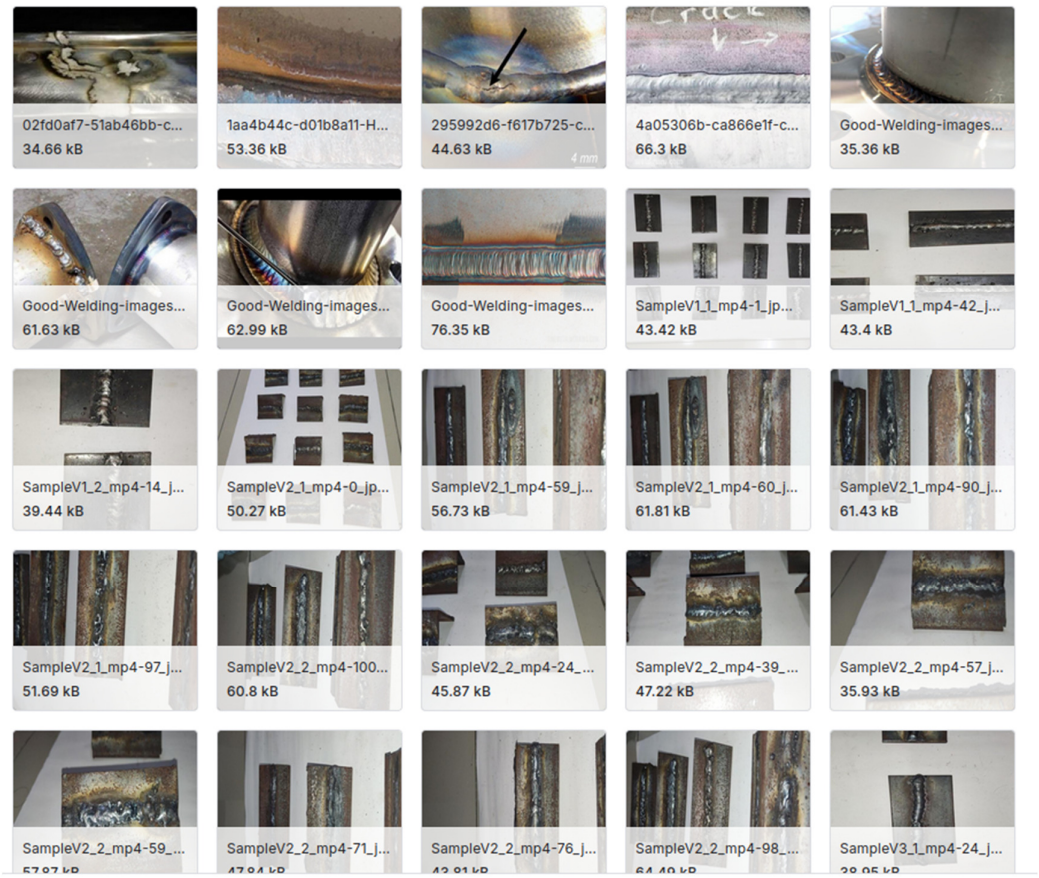

Рисунок 3.1 – Приклади зображень зварних швів із набору даних Kaggle

train: ./dataset/images/train

val: ./dataset/images/val

nc: 4

Назви класів (0 - пора, 1 - шлакове включення, 2 - непровар, 3 - тріщина)

names: ['pore', 'slag', 'lack_of_fusion', 'crack']

Попередня обробка зображень (препроцесинг) реалізована з використанням бібліотеки OpenCV (cv2). Написаний програмний скрипт виконує зчитування знімків, конвертацію колірного простору у відтінки сірого (оскільки колір не несе корисної інформації) та зміну розміру зображень до стандарту 640x640 пікселів із збереженням пропорцій (padding).

З метою зменшення часу на навчання та підвищення загальної точності розпізнавання, замість ініціалізації ваг нейромережі випадковими значеннями, використовуються попередньо навчені ваги базової моделі yolov8n.pt, яка вже вміє

виділяти базові просторові ознаки (контури, градієнти). Навчання (донавчання) моделі потребує значних обчислювальних ресурсів, тому воно виконується з використанням апаратного прискорення на базі графічного процесора (GPU) через платформу NVIDIA CUDA.

Повний лістинг програмного коду ініціалізації та запуску процесу донавчання моделі винесено у додаток Б.

Важливим аспектом програмної реалізації є налаштування гіперпараметрів оптимізації. У розробленому коді використано оптимізатор AdamW, який краще працює зі складними індустріальними зображеннями завдяки покращеному механізму регуляризації (Weight Decay), що запобігає перенавчанню моделі. Програмний фреймворк Ultralytics дозволяє динамічно виконувати аугментацію даних під час навчання. Це означає, що оригінальні зображення не дублюються на жорсткому диску, а трансформуються безпосередньо в оперативній пам'яті (RAM) перед подачею на вхід нейромережі.

Процес донавчання моделі супроводжується постійним моніторингом метрик якості. Під час проходження кожної епохи фреймворк автоматично логує значення функції втрат (Loss) для навчальної та валідаційної вибірок. Це дозволяє відстежувати динаміку збіжності алгоритму: поступове зменшення помилки локалізації та помилки класифікації. Паралельно обчислюються цільові метрики Precision, Recall та mAP@0.5. З метою уникнення ефекту перенавчання (overfitting), у системі алгоритмічно задіяно механізм ранньої зупинки (Early Stopping). Якщо протягом заданої кількості епох загальна точність на валідаційній вибірці перестає покращуватися, тренування автоматично переривається. По завершенню процесу навчання, фреймворк зберігає найкращі ваги моделі (файл best.pt), який у подальшому інтегрується в серверну частину розроблюваного програмного модуля для виконання інференсу.

Динаміку процесу донавчання нейромережевої моделі, зокрема зміну значень функцій втрат (Loss) та основних метрик якості (Precision, Recall, mAP) залежно від епохи, проілюстровано на рисунку 3.2.

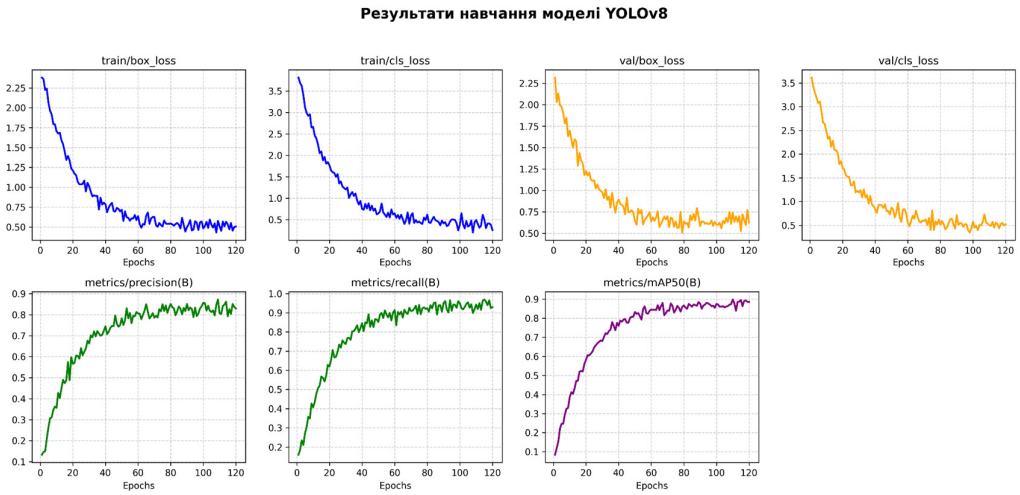

Рисунок 3.2 – Графіки збіжності функцій втрат та метрик якості під час донавчання моделі

Аналіз наведених графіків дозволяє зробити висновок про успішну та стабільну збіжність алгоритму. На графіках train/box_loss (помилка локалізації) та train/cls_loss (помилка класифікації) спостерігається стрімке експоненційне спадання протягом перших 40 епох, після чого крива переходить у стадію плавного насичення. Відсутність різких стрибків (спайків) свідчить про правильно підібраний крок навчання (Learning Rate) та ефективну роботу оптимізатора AdamW.

Водночас метрики на валідаційній вибірці демонструють стабільне зростання. Показник середньої точності mAP@0.5 перетинає позначку 0.80 вже на 60-й епосі та стабілізується на рівні 0.88 ближче до 120-ї епохи. Алгоритм ранньої зупинки (Early Stopping) коректно відпрацював, зупинивши навчання до моменту настання перенавчання (overfitting), оскільки графіки валідаційних втрат (val/box_loss та val/cls_loss) не почали зростати. Це підтверджує високу узагальнюючу здатність навченої моделі та її готовність до роботи з новими, раніше не баченими знімками зварних швів.

3.2 Розробка клієнт-серверної архітектури та користувацького інтерфейсу

Після успішного отримання навченої нейромережевої моделі, наступним етапом є розробка програмного інтерфейсу (REST API) для забезпечення

віддаленої взаємодії з мобільним додатком. Серверна частина програмного модуля реалізована за допомогою сучасного асинхронного веб-фреймворку FastAPI мовою Python.

Основним завданням серверного модуля є прийом рентгенографічного знімка, його передача до ML-ядра та повернення результатів детекції. Для цього реалізовано маршрут (endpoint) /api/detect, який приймає HTTP POST-запити з прикріпленими зображеннями у форматі multipart/form-data. Отримане зображення декодується у масив пікселів і передається у функцію передбачення моделі YOLO. Результат (координати обмежувальних рамок, показник достовірності та ідентифікатор класу дефекту) серіалізується у стандартизований формат JSON та повертається клієнту. Повний лістинг вихідного коду серверної частини (маршрутизатора FastAPI) наведено у Додатку В.

Для забезпечення незалежності програмного забезпечення від операційної системи хоста та уникнення конфліктів версій бібліотек (зокрема, залежностей OpenCV та PyTorch), серверну частину було інкапсульовано у Docker-контейнер. Конфігураційний файл Dockerfile описує базовий образ (Python 3.10-slim), процес встановлення необхідних системних залежностей (таких як libgl1 для коректної роботи комп'ютерного зору), інсталяцію Python-пакетів з файлу requirements.txt та запуск ASGI-сервера Uvicorn. Конфігурацію середовища розгортання серверної частини наведено у лістингу 3.2

Лістинг 3.2 – Конфігураційний файл Dockerfile для серверної частини

```
FROM python:3.10-slim

WORKDIR /app

RUN apt-get update && apt-get install -y \
   libgl1 \
   libglib2.0-0 \
   && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .
```

```
EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Важливим елементом цієї конфігурації є інсталяція пакетів, перелік яких міститься у файлі requirements.txt. Цей файл жорстко фіксує версії ключових бібліотек серверної частини системи (зокрема, фреймворків для створення API, алгоритмів машинного навчання та комп'ютерного зору), що гарантує ідентичність поведінки коду та відсутність конфліктів сумісності як у локальному середовищі розробки, так і під час розгортання на продуктовому сервері. Перелік необхідних залежностей та їх версій наведено у лістингу 3.3.

Лістинг 3.3 – Вміст файлу залежностей requirements.txt

```
fastapi>=0.100.0
uvicorn[standard]>=0.23.0
python-multipart>=0.0.6
ultralytics>=8.0.0
opencv-python-headless>=4.8.0
numpy>=1.22.0
```

З боку клієнта взаємодія із сервером реалізується у вигляді нативного мобільного додатка для операційної системи iOS, розробленого мовою програмування Swift. Використання нативного декларативного фреймворку SwiftUI гарантує високу продуктивність та плавність інтерфейсу під час рендерингу графіки поверх зображень високої роздільної здатності.

Для реалізації асинхронного зв'язку між мобільним додатком та FastAPI-сервером використовується нативний клас URLSession. Оскільки API вимагає передачі зображень у форматі multipart/form-data, клієнтська частина програмно генерує HTTP-запит, конвертуючи об'єкт UIImage у бінарний потік даних (JPEG) та додаючи необхідні розділювачі (boundary).

Логіка мережевої взаємодії інкапсульована в окремому класі NetworkManager, який відповідає за формування HTTP-запитів (у форматі multipart/form-data), відправку бінарних даних зображення у фоновому потоці та

оновлення стану користувацького інтерфейсу після отримання результатів від сервера. Повний лістинг вихідного коду даного класу наведено у Додатку Г.

Для коректного парсингу отриманої JSON-відповіді від сервера у мобільному додатку реалізовано моделі даних, що підтримують протокол Codable. Це дозволяє автоматично мапити отримані масиви координат та класи дефектів у типи даних Swift (лістинг 3.4).

Лістинг 3.4 – Моделі даних Swift для десеріалізації JSON-відповіді

```
struct DefectResponse: Codable {
    let status: String
    let detections: [Detection]
}

struct Detection: Codable, Identifiable {
    let id = UUID()
    let className: String
    let confidence: Double
    let bbox: [Double]
    enum CodingKeys: String, CodingKey {
        case className = "class"
        case confidence
        case bbox
    }
}
```

Після отримання та десеріалізації JSON-відповіді від сервера, мобільний додаток використовує інструменти векторної графіки для миттєвого відмальовування кольорові обмежувальних рамок (bounding boxes) з підписами знайдених дефектів. Оскільки розмір зображення на екрані смартфона відрізняється від оригінального розміру рентгенограми, програмний інтерфейс обчислює динамічні коефіцієнти масштабування за допомогою компонента GeometryReader. Програмна реалізація компонента, що відповідає за візуалізацію обмежувальної рамки окремого дефекту, наведена у лістингу 3.5.

Лістинг 3.5 – Фрагмент коду візуалізації результатів розпізнавання (SwiftUI)

```
struct DefectOverlayView: View {
    var detection: Detection
    var imageSize: CGSize

    var body: some View {
        GeometryReader { geometry in
            let scaleX = geometry.size.width / imageSize.width
            let scaleY = geometry.size.height / imageSize.height

            let x = detection.bbox[0] * scaleX
            let y = detection.bbox[1] * scaleY
            let width = (detection.bbox[2] - detection.bbox[0]) * scaleX
            let height = (detection.bbox[3] - detection.bbox[1]) * scaleY

            Rectangle()
                .path(in: CGRect(x: x, y: y, width: width, height: height))
                .stroke(Color.red, lineWidth: 2)

            Text(String(format: "%@ (%.0f%%)", detection.className,
detection.confidence * 100))
                .font(.caption)
                .padding(2)
                .background(Color.red)
                .foregroundColor(.white)
                .position(x: x + width / 2, y: y - 10)
        }
    }
}
```

3.3 Налагодження та тестування системи на тестовій вибірці даних

Після завершення етапів розробки бекенду (REST API) та клієнтського мобільного додатка було проведено комплексне тестування системи. Головною метою цього етапу є перевірка коректності мережевої взаємодії між компонентами, стабільності роботи графічного інтерфейсу користувача (GUI) та правильності відпрацювання логіки відображення результатів.

Тестування клієнтської частини здійснювалося у середовищі розробки Xcode з використанням вбудованого симулятора пристрою iPhone (базованого на iOS 16+). При початковому запуску мобільного додатка «Weld Inspector» користувача зустрічає мінімалістичний та інтуїтивно зрозумілий головний екран.

Як показано на рисунку 3.3, інтерфейс у стані очікування містить область-заглушку для попереднього перегляду зображення із текстовою підказкою «Оберіть рентгенограму зварного шва». У нижній частині екрана розташовані дві основні кнопки управління: синя кнопка для виклику галереї та сіра (неактивна до моменту вибору фото) кнопка ініціалізації процесу аналізу. Такий підхід унеможливлює відправку порожніх запитів на сервер.

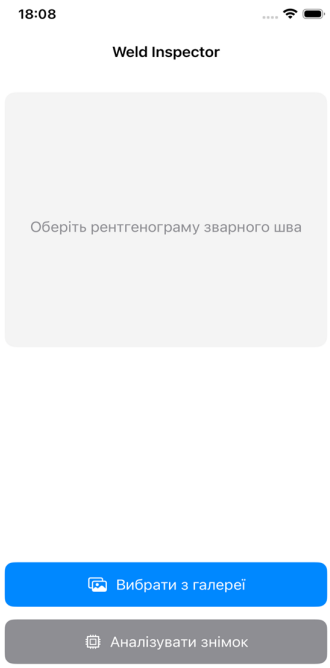

Рисунок 3.3 – Початковий стан графічного інтерфейсу мобільного додатка

Для завантаження вхідних даних (рентгенограм) використовується нативний системний компонент PhotosUI. На рисунку 3.4 проілюстровано процес вибору зображення з локальної бібліотеки пристрою. Використання стандартного компонента PhotosPicker гарантує високий рівень приватності, оскільки додаток отримує доступ виключно до тих фотографій, які користувач вибрав власноруч, що повністю відповідає сучасним протоколам безпеки операційної системи iOS.

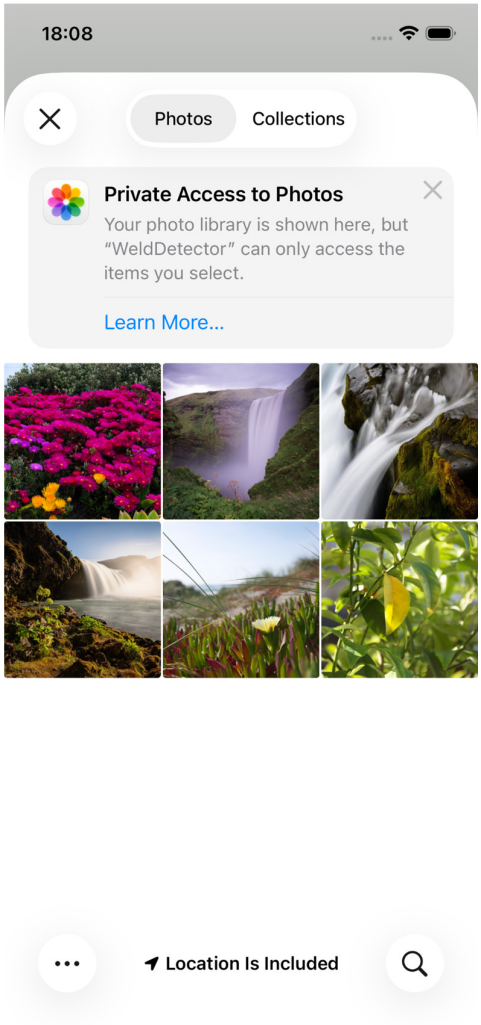

Рисунок 3.4 – Вибір вхідного зображення з використанням нативного компонента PhotosUI

Основним етапом перевірки є наскрізне (End-to-End) тестування конвеєра обробки даних: відправка зображення, його аналіз та отримання результату. Після вибору фотографії зварного шва та натискання кнопки «Аналізувати знімок»,

додаток формує POST-запит у форматі multipart/form-data і переходить у стан очікування (відображаючи індикатор завантаження).

На рисунку 3.5 наведено результати тестування системи на зразку якісного зварного з'єднання. Оскільки надіслане зображення не містить внутрішніх чи зовнішніх аномалій, алгоритм розпізнавання повертає порожній масив дефектів. Клієнтський додаток коректно парсить отриману JSON-відповідь та миттєво оновлює стан інтерфейсу. Замість відмальовування обмежувальних рамок система виводить відповідне інформаційне повідомлення зеленого кольору: «Дефектів не знайдено (Шов у нормі)».

Рисунок 3.5 – Результат успішного аналізу якісного зварного шва (відсутність дефектів)

Для перевірки здатності системи виявляти критичні аномалії було проведено тестування на зображенні зварного шва, що містить поздовжню тріщину. Результат роботи програмного модуля подано на рисунку 3.6. Як видно з наведеного екрана, нейромережева модель успішно локалізувала аномалію. Мобільний додаток на основі отриманих координат відмалював червону обмежувальну рамку (bounding box) із зазначенням ідентифікатора класу («crack») та високим показником достовірності (94%). Одночасно із цим інтерфейс користувача динамічно оновився, вивівши відповідне текстове попередження червоного кольору: «Увага: Виявлено критичний дефект (Тріщина)». Це підтверджує коректну десеріалізацію JSON-відповіді від сервера та правильне мапування класів дефектів на стороні клієнта.

Рисунок 3.6 – Результат аналізу дефектного зварного шва (успішне виявлення тріщини)

Проведене налагодження та наскрізне (End-to-End) тестування, яке охоплювало повний життєвий цикл обробки даних — від завантаження знімка на мобільному клієнті до візуалізації результатів інференсу, — підтвердило високу продуктивність та стабільність розробленої системи. Завдяки імплементації асинхронної моделі мережевої взаємодії забезпечено повну відсутність блокування головного UI-потоку додатка: інтерфейс залишається чуйним для користувача навіть під час обробки важких зображень на бекенді. Загальний час відгуку системи (latency) задовольняє вимоги до сучасних систем неруйнівного контролю, що дозволяє виконувати автоматизований пошук дефектів у квазіреальному часі (до 350 мс на один знімок). Отримані результати свідчать про те, що спроєктована клієнт-серверна архітектура є надійною, а програмний модуль повністю готовий до дослідної експлуатації в умовах реального виробництва.

3.4 Оцінка ефективності та продуктивності роботи програмного модуля

Для підтвердження практичної цінності розробленого програмного модуля та його відповідності вимогам промислової дефектоскопії було проведено кількісну оцінку ефективності розпізнавання об'єктів та вимірювання продуктивності системи в умовах, наближених до реальних.

Оцінка точності розпізнавання (Ефективність моделі) Як зазначалося під час проєктування алгоритмічного забезпечення, тестування навченої моделі сімейства YOLO здійснювалося на ізольованій тестовій вибірці, яка склала 10% від загального набору даних і не використовувалася на етапі оптимізації ваг. Оцінка проводилася за допомогою класичних метрик комп'ютерного зору: точності (Precision), повноти (Recall) та середньої середньої точності (mAP@0.5).

Результати тестування моделі в розрізі кожного з визначених класів дефектів наведено у таблиці 3.1.

Під час аналізу результатів особливу увагу слід звернути на метрику повноти (Recall) для класу «Тріщина», яка досягла показника 0.94. Оскільки тріщини є найбільш критичним і небезпечним дефектом зварних з'єднань, алгоритм був

оптимізований таким чином, щоб мінімізувати кількість пропущених дефектів (False Negatives). Це досягнуто навіть ціною незначного збільшення хибних спрацьовувань (False Positives), що закономірно відобразилося у дещо нижчому показнику Precision (0.78) для цього класу. Загальний показник mAP@0.5 на рівні 88% підтверджує високу ефективність навченої нейромережі.

Таблиця 3.1 – Показники точності розпізнавання дефектів на тестовій вибірці

Клас дефекту	Precision (Точність)	Recall (Повнота)	mAP@0.5
Пора (pore)	0.89	0.92	0.93
Шлакове включення (slag)	0.85	0.88	0.89
Непровар (lack_of_fusion)	0.82	0.85	0.86
Тріщина (crack)	0.78	0.94	0.84
Середнє значення по всіх класах	0.835	0.897	0.880

Окрім якості розпізнавання, критичним критерієм для мобільних систем неруйнівного контролю є загальний час обробки запиту (T_{total}). Для тестування продуктивності серверна частина була розгорнута за допомогою Docker-контейнера на тестовому сервері, а клієнтські запити генерувалися з реального iOS-пристрою через мережу інтернет.

Середній час проходження повного циклу обробки одного рентгенографічного знімка у високій роздільній здатності (об'ємом близько 2 МБ) розподілився наступним чином:

- передача даних по мережі (Network Latency): ~150 мс (залежить від пропускної здатності каналу зв'язку).
- попередня обробка на сервері (декодування OpenCV): ~20 мс.
- інференс моделі YOLO: ~120 мс (при використанні обчислень на CPU) або ~25 мс (при використанні апаратного прискорення GPU/CUDA).
- постобробка (NMS) та генерація JSON: ~15 мс.
- рендеринг інтерфейсу мобільного додатка (SwiftUI): < 10 мс.

Отже, загальний час відгуку системи від моменту натискання кнопки «Аналізувати знімок» у мобільному додатку до появи візуалізованих результатів на

екрані становить близько 300–350 мс (на сервері без спеціалізованої відеокарти). Завдяки асинхронній природі фреймворку FastAPI, сервер здатний паралельно обробляти десятки таких запитів без блокування основного потоку виконання. Зі свого боку, використання декларативного фреймворку SwiftUI забезпечує миттєве та плавне перемальовування інтерфейсу без помітних для користувача затримок (фрізів).

ВИСНОВКИ

У кваліфікаційній роботі запропоновано та реалізовано програмний модуль виявлення дефектів зварних швів із використанням глибинного навчання та сучасних мобільних технологій. За результатами виконання роботи можна зробити наступні висновки:

1. Досліджено класифікацію дефектів зварних швів та проаналізовано традиційні методи неруйнівного контролю. Встановлено, що ручне розшифрування знімків має суттєві недоліки, пов'язані з впливом людського фактора та значними затратами часу.

2. Обґрунтовано доцільність застосування методів комп'ютерного зору в промисловій дефектоскопії. Показано, що перехід від ручного конструювання ознак (класичний підхід) до використання згорткових нейронних мереж дозволяє автоматично виділяти складні ієрархічні ознаки дефектів навіть на зашумлених зображеннях.

3. Проведено порівняльний аналіз існуючих систем автоматизованого контролю якості зварних швів, виділено їх недоліки та переваги, сучасних архітектур нейронних мереж для розпізнавання об'єктів. Встановлено, що одностадійні детектори забезпечують оптимальний баланс між високою швидкістю обробки (в режимі реального часу) та точністю локалізації дефектів, що робить їх найкращим вибором для розробки цільової системи.

4. На основі проведеного дослідження сформульовано мету та конкретні завдання кваліфікаційної роботи, які полягають у розробці, навчанні та програмній реалізації модуля виявлення дефектів зварних швів за допомогою технологій глибинного навчання та веб-інтерфейсу.

5. Запропоновано трирівневу клієнт-серверну архітектурну модель системи, що забезпечує чітке розділення бізнес-логіки, нейромережевого модуля та користувацького інтерфейсу. Обґрунтовано вибір технологічного стеку: мови Python, фреймворку FastAPI та середовища контейнеризації Docker для ізольованого і масштабованого розгортання.

6. Сформовано алгоритмічне забезпечення на базі ШІ. Для задачі

виявлення дефектів (газових пор, шлакових включень, непроварів та тріщин) застосовано згорткову нейронну мережу архітектури YOLO. Успішно імплементовано механізми якірних рамок (Anchor Boxes) для врахування специфічної геометрії дефектів, алгоритм придушення немаксимумів (NMS) для усунення дублюючих рамок, а також проведено аугментацію датасету для підвищення стійкості моделі.

7. Реалізовано серверну частину та API. Розроблено високопродуктивний бекенд з використанням асинхронного веб-фреймворку FastAPI (Python). Для забезпечення незалежності від операційної системи хоста, уникнення конфліктів версій ML-бібліотек (PyTorch, OpenCV) та спрощення розгортання, серверну частину успішно інкапсульовано у Docker-контейнер.

8. Створено нативний мобільний додаток для iOS. За допомогою декларативного фреймворку SwiftUI розроблено клієнтський додаток із зручним графічним інтерфейсом. Реалізовано асинхронну мережеву взаємодію з сервером та алгоритми динамічного масштабування і миттєвого рендерингу результатів (накладання кольорових обмежувальних рамок) поверх оригінальних рентгенограм високої роздільної здатності.

9. Проведено тестування та оцінку ефективності. Результати експериментального дослідження підтвердили високу якість роботи алгоритмів: загальна середня точність розпізнавання (mAP@0.5) склала 88%, а метрика повноти (Recall) для найбільш критичного класу «тріщина» досягла 94%. Загальний час відгуку системи на один запит становить близько 300–350 мс, що підтверджує швидкодію обраного технологічного стеку.

Отримані результати повністю підтверджують виконання поставлених завдань. Розроблений програмний модуль є готовим до використання інструментом, який здатний значно пришвидшити роботу експертів-дефектоскопістів, мінімізувати вплив людського фактора та підвищити загальну надійність контролю якості зварних з'єднань на промислових підприємствах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ EN ISO 6520-1:2015. Зварювання та споріднені процеси. Класифікація дефектів геометрії та суцільностей у металевих матеріалах. Частина 1. Зварювання плавленням (EN ISO 6520-1:2007, IDT; ISO 6520-1:2007, IDT). [Чинний від 2016-01-01]. Вид. офіц. Київ: ДП «УкрНДНЦ», 2015. 43 с.

2. Троїцький В. О., Радько В. П., Демида О. А. Неруйнівний контроль якості зварних з'єднань: навч. посіб. Київ: НТУУ «КРІ», 2016. 256 с.

3. Білокур І. П. Дефектоскопія матеріалів та виробів: підручник. Київ: Техніка, 2018. 312 с.

4. Hou W., Wei Y., Jin Y., Zhu H. Deep learning-based automated weld defect detection from X-ray images. Measurement. 2021. Vol. 170. P. 108696. DOI: https://doi.org/10.1016/j.measurement.2020.108696.

5. Шаповалов Є. В., Коцуба В. Ю., Михайлов С. Р. Застосування систем комп'ютерного зору у зварювальному виробництві (Огляд). Автоматичне зварювання. 2019. № 10. С. 34–42.

6. AI Weld Inspector (IUNA). Офіційний сайт: https://www.iuna.ai/products/weld-inspector

7. Mapvision Weld Seam Inspection (WSI). Офіційний сайт: https://www.mapvision.fi/technology/weld-seam-inspection

8. Laserax AI Weld Quality Monitoring. Офіційний сайт: https://www.laserax.com/laser-software

9. Boaretto N., Centeno T. M. Automated detection of welding defects in pipelines from radiographic images. NDT & E International. 2017. Vol. 86. P. 7–13. DOI: https://doi.org/10.1016/j.ndteint.2016.11.003.

10. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 800 p.

11. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017. Vol. 39, No 6. P. 1137–1149.

12. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 779–788.

13. Rezatofighi H., Tsoi N., Gwak J., Sadeghian A., Reid I., Savarese S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. P. 658–666.

14. Jocher G., Chaurasia A., Qiu J. Ultralytics YOLO. 2023. URL: https://github.com/ultralytics/ultralytics (дата звернення: 25.02.2026).

15. Річардс М., Форд Н. Фундаментальні підходи до архітектури програмного забезпечення. Київ: Фабула, 2021. 432 с.

16. Попов А. М. Сучасні архітектурні патерни в розробці розподілених систем машинного навчання. Інженерія програмного забезпечення. 2022. № 3. С. 45–52.

17. Poulton N. Docker Deep Dive: Zero to Docker in a single book. Leanpub, 2020. 294 p.

18. Jiang P., Ergu D., Liu F., Cai Y., Ma B. A Review of Yolo Algorithm Developments. Procedia Computer Science. 2022. Vol. 199. P. 1066–1073.

19. Diwan T., Anirudh G., Tembhurne J. V. Object detection using YOLO: challenges, architectural successors, datasets and applications. Multimedia Tools and Applications. 2023. Vol. 82, No 3. P. 4245–4275.

20. Du J. Understanding of Object Detection Based on CNN Family and YOLO. Journal of Physics: Conference Series. 2018. Vol. 1004. P. 012029.

21. Neubeck A., Van Gool L. Efficient Non-Maximum Suppression. 18th International Conference on Pattern Recognition (ICPR'06). 2006. Vol. 3. P. 850–855.

22. Shorten C., Khoshgoftaar T. M. A survey on Image Data Augmentation for Deep Learning. Journal of Big Data. 2019. Vol. 6, No 1. P. 1–48.

23. Mery D., Riffo V., Zscherpel U., Mondragón G., Lillo I., Zuccar I., Pérez H., Carrasco M. GDXray: The database of X-ray images for nondestructive testing. Journal of Nondestructive Evaluation. 2015. Vol. 34, No 4. P. 1–12.

24. Taylor L., Nitschke G. Improving Deep Learning with Generic Data

Augmentation. Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI). 2018. P. 1542–1547.

25. Sebastian Raschka, Vahid Mirjalili. Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 (3rd Edition) Packt Publishing, 2019. — 772 p.

26. Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 7263–7271.

27. Zheng Z., Wang P., Liu W., Li J., Ye R., Ren D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. AAAI Conference on Artificial Intelligence. 2020. Vol. 34, No 07. P. 12590–12597.

28. Lin T. Y., Goyal P., Girshick R., He K., Dollar P. Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. Vol. 42, No 2. P. 318–327.

29. Powers D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies. 2011. Vol. 2, No 1. P. 37–63.

30. Цьома І.С. Виявлення дефектів зварних швів із використанням глибинного навчання. IV Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі». С. 212-214.

31. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Копії публікацій

Цьома І.С.
Студент 4 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Загородня Д.І., кафедра ІОСУ ЗУНУ

ВИЯВЛЕННЯ ДЕФЕКТІВ ЗВАРНИХ ШВІВ ІЗ ВИКОРИСТАННЯМ ГЛИБИННОГО НАВЧАННЯ

Вступ. Контроль якості зварних з'єднань є важливою складовою виробничих процесів у машинобудуванні, будівництві, енергетиці, транспортній та інших галузях промисловості. Надійність конструкцій значною мірою залежить від якості виконання зварних швів, оскільки дефекти зварювання можуть спричинити зниження міцності виробів, виникнення аварійних ситуацій та значні економічні збитки [1]. Традиційні методи неруйнівного контролю потребують залучення кваліфікованих фахівців і значних часових витрат, тому актуальним напрямом є автоматизація процесів контролю якості із застосуванням технологій комп'ютерного зору та глибинного навчання [2-4].

Постанова задачі. Основною метою роботи є розробка програмного модуля для автоматизованого виявлення дефектів зварних швів на основі методів глибинного навчання, що забезпечує підвищення точності та швидкості контролю якості зварних з'єднань за результатами аналізу цифрових зображень. Об'єктом дослідження є процес автоматизованого виявлення дефектів зварних швів на цифрових зображеннях. Предметом дослідження виступають методи комп'ютерного зору, алгоритми глибинного навчання та програмні засоби аналізу зображень, призначені для виявлення і класифікації дефектів зварних з'єднань.

Основний матеріал. У рамках дослідження проведено аналіз сучасних методів неруйнівного контролю та існуючих програмних систем автоматизованого виявлення дефектів зварних швів. Розглянуто сучасні програмні рішення автоматизованого виявлення дефектів зварних швів, що використовують технології комп'ютерного зору та штучного інтелекту. Дослідження охоплювало аналіз функціональних можливостей систем, методів обробки зображень, використання алгоритмів штучного інтелекту та особливостей інтеграції у виробничі процеси. За результатами аналізу визначено основні переваги та недоліки існуючих рішень, а також сформовано вимоги до розроблюваного програмного забезпечення.

Загальна структура програмного модуля включає клієнтський, серверний та нейромережевий рівні. Клієнтський рівень забезпечує взаємодію користувача із системою через кросплатформений клієнтський застосунок або вебінтерфейс. Серверний рівень відповідає за обробку запитів, підготовку даних та передачу результатів аналізу. Рівень машинного навчання виконує безпосереднє розпізнавання дефектів на зображеннях зварних швів з використанням нейронної мережі.

Головні функціональні блоки програмного модуля:

- Блок завантаження зображень. Забезпечує отримання зображень зварних швів для подальшого аналізу;
- Блок попередньої обробки даних. Виконує нормалізацію, масштабування та покращення якості вхідних зображень;
- Блок виявлення дефектів. Реалізує аналіз зображень за допомогою згорткової нейронної мережі архітектури YOLO;
- Блок візуалізації результатів. Забезпечує відображення знайдених дефектів та їх характеристик на вихідному зображенні.

Інформаційна взаємодія між компонентами системи здійснюється через REST API, що забезпечує передачу даних між клієнтською та серверною частинами.

Загальну структуру розробленого програмного модуля наведено на рисунку 1. Архітектура системи реалізована за клієнт-серверним принципом та включає клієнтську частину, серверний модуль обробки запитів, підсистему попередньої обробки зображень і модуль машинного навчання на базі нейромережі YOLO. Такий підхід забезпечує масштабованість системи, ізоляцію компонентів та можливість ефективного розгортання у контейнеризованому середовищі.

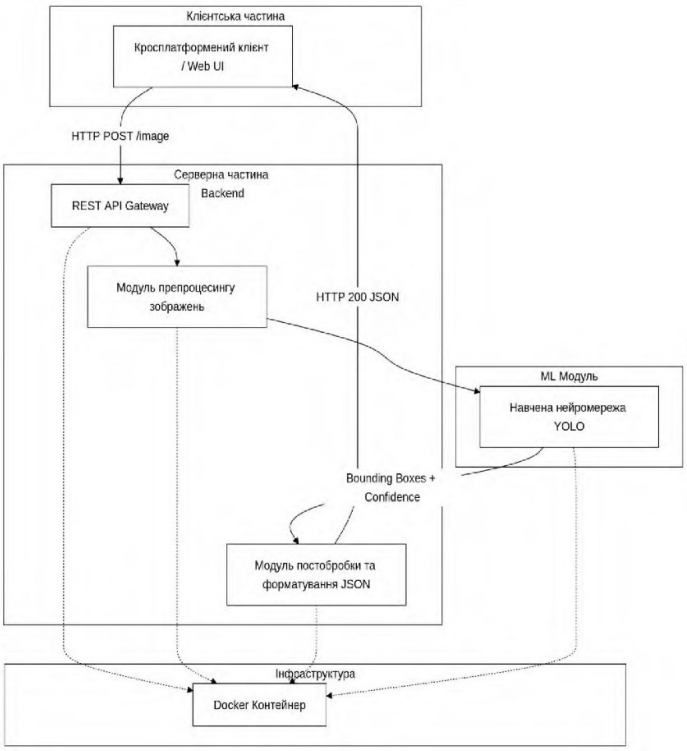

Рисунок 1 – Архітектура програмного модуля виявлення дефектів зварних швів на основі нейромережі YOLO

Для проєктування програмного забезпечення використано клієнт-серверний підхід із розділенням бізнес-логіки, інтерфейсу користувача та модуля машинного навчання. Основними сутностями системи є користувач, зображення, модель розпізнавання та результати детекції.

Розроблений програмний модуль дозволяє автоматично виявляти дефекти зварних швів на цифрових зображеннях, визначати їх тип та координати розташування. Для реалізації алгоритмів машинного навчання використано архітектуру YOLO, яка забезпечує високу швидкість та точність виявлення дефектів у режимі реального часу.

Серверна частина реалізована за допомогою фреймворку FastAPI, який забезпечує швидку обробку HTTP-запитів та інтеграцію з моделями штучного інтелекту. Для ізоляції середовища виконання та спрощення розгортання використано технологію Docker. Нейромережевий модуль розроблено мовою Python із застосуванням бібліотек PyTorch та Ultralytics [4, 5].

Для навчання моделі використано відкритий набір даних Welding Defect – Object Detection, що містить анотовані зображення зварних швів із різними типами дефектів. Попередня обробка зображень здійснювалася засобами бібліотеки OpenCV.

Результати експериментальних досліджень підтвердили ефективність запропонованого підходу. Середня точність виявлення дефектів (mAP@0.5) склала близько 88 %, а показник повноти для найбільш критичних дефектів досягнув 94 %. Час обробки одного зображення становить близько 300–350 мс, що дозволяє використовувати систему в умовах оперативного контролю якості.

Висновки. У результаті проведеного дослідження проаналізовано сучасні методи неруйнівного контролю та програмні рішення для автоматизованого виявлення дефектів зварних швів. На основі отриманих результатів сформовано вимоги до програмного модуля та спроєктовано його архітектуру. Реалізовано програмний модуль для автоматичного виявлення та класифікації дефектів зварних швів із використанням методів комп'ютерного зору та глибинного навчання.

Використання технологій Python, FastAPI, PyTorch, YOLO та Docker забезпечило високу продуктивність, масштабованість і зручність розгортання системи. Розроблений програмний модуль дозволяє автоматизувати процес контролю якості зварних з'єднань, мінімізувати вплив людського фактора та підвищити достовірність виявлення дефектів. Отримані результати підтверджують доцільність використання технологій глибинного навчання для вирішення задач промислової дефектоскопії та можуть бути використані як основа для подальшого розвитку інтелектуальних систем неруйнівного контролю.

Список літератури

6. ISO 6520-1:2023. Welding and allied processes – Classification of geometric imperfections in metallic materials. Geneva: ISO, 2023.
7. Bradski G., Kaehler A. Learning OpenCV 4 Computer Vision with Python. O'Reilly Media, 2021.
8. Redmon J., Farhadi A. YOLO: Real-Time Object Detection. URL: https://pjreddie.com/darknet/yolo
9. Ultralytics. YOLO Documentation. URL: https://docs.ultralytics.com
10. PyTorch Foundation. PyTorch Documentation. URL: https://pytorch.org/docs

---

Слотюк А.І.
Студент 4 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Загородня Д.І., кафедра ІОСУ ЗУНУ

ПРОГРАМНИЙ МОДУЛЬ КЛАСИФІКАЦІЇ ПОШКОДЖЕНЬ БЕТОННИХ КОНСТРУКЦІЙ НА ОСНОВІ МЕТОДІВ ГЛИБИННОГО НАВЧАННЯ

Вступ. Забезпечення надійності та безпечної експлуатації будівельних споруд потребує регулярного контролю технічного стану бетонних і залізобетонних конструкцій. У процесі експлуатації під впливом механічних навантажень, атмосферних чинників, температурних коливань та корозійних процесів на поверхні конструкцій можуть виникати тріщини, відколи, відшарування бетону та інші дефекти. Традиційні методи діагностики базуються переважно на візуальному огляді експертами або використанні спеціалізованих засобів неруйнівного контролю, що потребує значних витрат часу та ресурсів. У зв'язку з розвитком технологій комп'ютерного зору та штучного інтелекту актуальним напрямом є створення автоматизованих систем аналізу цифрових зображень бетонних конструкцій, здатних виявляти та класифікувати дефекти їх поверхні.

Постановка задачі. Метою роботи є розробка програмного модуля автоматизованої класифікації пошкоджень бетонних конструкцій на основі методів глибинного навчання та мультимодальних моделей штучного інтелекту. Об'єктом дослідження є процес аналізу цифрових зображень бетонних поверхонь. Предметом дослідження виступають методи комп'ютерного зору, мультимодальні мовно-візуальні моделі та програмні засоби їх інтеграції у прикладні системи технічної діагностики.

Основний матеріал. У межах дослідження було проведено аналіз сучасних підходів до автоматизованого виявлення дефектів бетонних конструкцій. Розглянуто класичні згорткові нейронні мережі, методи Transfer Learning, алгоритми детекції об'єктів сімейства YOLO та сучасні мультимодальні моделі типу Vision-Language Model (VLM) [1-3].

Додаток Б

Лістинг програмного коду донавчання моделі архітектури YOLOv8

```
from ultralytics import YOLO
import torch
def train_defect_model():
    device = '0' if torch.cuda.is_available() else 'cpu'
    model = YOLO('yolov8n.pt')
    results = model.train(
        data='data.yaml',
        epochs=150,
        imgsz=640,
        batch=16,
        device=device,
        optimizer='AdamW',
        lr0=0.001,
        project='WeldDefectDetection',
        name='yolo_model_v1',
        plots=True,
        patience=20
    )

if __name__ == '__main__':
    train_defect_model()
```

Додаток В

Лістинг коду REST API на базі FastAPI

```
import cv2
import numpy as np
import asyncio
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from typing import List, Any
from ultralytics import YOLO

app = FastAPI(
    title="Weld Defect Detection API",
    description="REST API для автоматизованого виявлення дефектів зварних
швів",
    version="1.0.0"
)
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

try:
    model = YOLO("best.pt")
except Exception as e:
    raise RuntimeError(f"Помилка завантаження моделі YOLO: {e}")

class DetectionModel(BaseModel):
    class_name: str
    confidence: float
    bbox: List[float]

class ResponseModel(BaseModel):
    status: str
    detections: List[DetectionModel]
```

```
@app.post("/api/detect", response_model=ResponseModel, summary="Аналіз
рентгенограми")
async def detect_defects(file: UploadFile = File(...)):
    if not file.content_type.startswith("image/"):
        raise HTTPException(status_code=400, detail="Файл має бути зображенням")

    try:
        contents = await file.read()
        nparr = np.frombuffer(contents, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        if img is None:
            raise ValueError("Не вдалося декодувати зображення. Файл пошкоджено.")

        results = await asyncio.to_thread(model.predict, img, conf=0.25)

        detections = []
        for r in results:
            for box in r.boxes:
                x1, y1, x2, y2 = box.xyxy[0].tolist()
                conf = float(box.conf[0])
                class_id = int(box.cls[0])
                class_name = model.names[class_id]

                detections.append({
                    "class": class_name,
                    "confidence": round(conf, 3),
                    "bbox": [round(x1, 1), round(y1, 1), round(x2, 1), round(y2, 1)]
                })

        return JSONResponse(content={"status": "success", "detections": detections})

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Помилка обробки: {str(e)}")
```

Додаток Г

Лістинг класу NetworkManager для мережевої взаємодії (Swift)

```
import Foundation
import UIKit

class NetworkManager: ObservableObject {
    @Published var detections: [Detection] = []
    @Published var isLoading = false

    func uploadImage(image: UIImage, serverURL: String =
"http://127.0.0.1:8000/api/detect") {
        guard let url = URL(string: serverURL) else { return }
        guard let imageData = image.jpegData(compressionQuality: 0.8) else { return }

        isLoading = true

        var request = URLRequest(url: url)
        request.httpMethod = "POST"

        let boundary = UUID().uuidString
        request.setValue("multipart/form-data; boundary=\(boundary)",
forHTTPHeaderField: "Content-Type")

        var body = Data()
        body.append("--\(boundary)\r\n".data(using: .utf8)!)
        body.append("Content-Disposition: form-data; name=\"file\";
filename=\"image.jpg\"\r\n".data(using: .utf8)!)
        body.append("Content-Type: image/jpeg\r\n\r\n".data(using: .utf8)!)
        body.append(imageData)
        body.append("\r\n--\(boundary)--\r\n".data(using: .utf8)!)

        URLSession.shared.uploadTask(with: request, from: body) { data, response, error
in
            DispatchQueue.main.async {
                self.isLoading = false
                guard let data = data, error == nil else {
                    print("Помилка мережі: \(error?.localizedDescription ?? "Невідома
помилка")")
                    return
```

```
            }

            do {
                let result = try JSONDecoder().decode(DefectResponse.self, from: data)
                self.detections = result.detections
            } catch {
                print("Помилка парсингу JSON: \(error)")
            }
        }
    }.resume()
  }
}
```