Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: **Класифікація дефектів на виробничій лінії за допомогою згорткових нейронних мереж**

Виконав: студент VI курсу, групи КНз-61м
спеціальності
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Дачук Й.В.
(прізвище та ініціали)

Керівник Паславський М.М.
(прізвище та ініціали)

Рецензент Сапак В.Н
(прізвище та ініціали)

Львів – 2025 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН

Борецька І. Б.
"10" грудня 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дачук Йосип Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи **Класифікація дефектів на виробничій лінії за допомогою згорткових нейронних мереж**

керівник роботи Паславський М.М, канд. техн. наук, асистент.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 29.04. 2025 р.№ С-288

2. Термін подання студентом роботи 10.12. 2025 р.

3. Вихідні дані до роботи:

- вивчити предметну область, проаналізувати існуючі фактори та методи виявлення дефектів на виробах, а також відповідні програмні продукти;
- розглянути і використати алгоритми, які лежать в основі математичної моделі інтелектуальної системи виявлення дефектів;
- спроектувати інтелектуальну систему з допомогою мови програмування Python та відповідних бібліотек;
- представити результати роботи інтелектуальної системи.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Розділ 1. Стан проблемної області

Розділ 2. Інформаційне забезпечення

Розділ 3. Математичне забезпечення

Розділ 4. Програмне забезпечення

Розділ 5. Розроблення стартап-проєкту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

додаток А, додаток Б

6. Дата видачі завдання 1 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних та інших джерел згідно досліджуваної теми	01.05-30.05.2025	виконано
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки	01.06-30.06. 2025	виконано
3	Постановка задачі та її формалізація	01.07-30.07. 2025	виконано
4	Вибір та обґрунтування методів і засобів проведення дослідження	01.08-30.08. 2025	виконано
5	Розроблення концептуальної схеми реалізації завдання	01.09-15.09. 2025	виконано
6	Програмна реалізація завдання	16.09-30.10. 2025	виконано
7	Тестування програмного продукту та отриманих результатів	01.11-15.11. 2025	виконано
8	Розробка пояснювальної записки магістерської роботи	16.11-30.11. 2025	виконано
9	Корегування пояснювальної записки згідно вимог, розроблення презентації	01.12-09.12. 2025	виконано

Студент _____ Дачук Й.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Паславський М.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота складається з 85 сторінок, 6 рисунків, 7 таблиць, 14 джерел, 2 додатків.

У роботі розроблено інтелектуальну систему для класифікації дефектів на виробничій лінії друкованих плат за допомогою згорткових нейронних мереж. Для вирішення задачі було використано набір зображень дефектних та бездефектних плат, який попередньо оброблявся для підготовки до навчання моделі. Створена модель продемонструвала високу точність у класифікації дефектів, що дозволяє ефективно автоматизувати контроль якості на виробництві. Розроблений алгоритм може бути використаний для зниження кількості дефектних виробів та підвищення ефективності виробничих процесів.

Ключові слова: *класифікація дефектів, дефекти друкованих плат, глибоке навчання, згорткові нейронні мережі*.

ABSTRACT

The master's thesis consists of 85 pages, 6 figure, 7 table, 14 references, 2 appendix.

In the course work, an intelligent system for classifying defects on the production line of printed circuit boards using convolutional neural networks was developed. To solve the problem, a set of images of defective and defect-free boards was used, which was pre-processed to prepare for model training. The created model demonstrated high accuracy in classifying defects, which allows for effective automation of quality control in production. The developed algorithm can be used to reduce the number of defective products and increase the efficiency of production processes.

Keywords: *defect classification, printed circuit board defects, deep learning, convolutional neural networks*.

ТЕХНІЧНЕ ЗАВДАННЯ

В дипломній роботі потрібно розробити інформаційну систему для класифікації дефектів на виробничій лінії, для цього потрібно вирішити такі завдання.

1. Провести аналіз сучасних методів класифікації дефектів на основі згорткових нейронних мереж та визначити їх переваги і недоліки.

2. Розробити та обґрунтувати математичну модель системи класифікації дефектів із врахуванням попередньої обробки зображень.

3. Реалізувати архітектуру згорткової нейронної мережі для класифікації дефектів друкованих плат.

4. Провести тестування розробленої системи на наборі зображень та порівняти її продуктивність з існуючими рішеннями.

5. Створити інтерфейс користувача для моніторингу процесу класифікації дефектів та візуалізації результатів.

.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

- AI (Artificial Intelligence) – штучний інтелект;
- Accuracy – точність, метрика оцінки якості моделі;
- Activation Function – функція активації;
- ANN – Artificial Neural Network (штучна нейронна мережа);
- CNN (Convolutional Neural Network) – згорткова нейронна мережа;
- Data Augmentation – доповнення даних;
- Deep Learning – глибоке навчання;
- Epoch – епоха, один повний цикл навчання моделі;
- F1-score – міра збалансованості точності та відгуку;
- Feature Extraction – виділення ознак;
- Gradient Descent – градієнтний спуск;
- Loss Function – функція втрат;
- ML (Machine Learning) – машинне навчання;
- Precision – метрика для класифікації;
- PCB (Printed Circuit Board) – друкована плата;
- Test Dataset – тестовий набір даних;
- Training Dataset – навчальний набір даних;
- Validation Dataset – набір даних для валідації.

ВСТУП

Актуальність дипломної роботи

Актуальність роботи зумовлена потребою в покращенні процесів контролю якості на виробничих лініях, зокрема для виготовлення друкованих плат, що є важливою складовою електронної техніки. Сучасні способи автоматизації дозволяють значно збільшити ефективність і точність виявлення дефектів, що знижує вірогідність випуску неякісної продукції. Застосування згорткових нейронних мереж є однією з найбільш перспективних технологій у сфері комп'ютерного зору, оскільки вони здатні ефективно обробляти та класифікувати зображення. За допомогою такої системи можна автоматизувати процес перевірки плат, що знижує навантаження на операторів і зменшує вірогідність людської помилки. Важливою перевагою є здатність таких моделей до адаптації та поліпшення з часом, що робить їх універсальними для різних виробничих умов. Згорткові нейронні мережі мають високу ефективність при роботі з великими об'ємами даних, що важливо для сучасних промислових підприємств. Впровадження автоматизованих систем контролю якості може значно скоротити час, необхідний для перевірки кожної деталі і дозволити підприємствам працювати з більш високою продуктивністю. Кількість дефектних виробів зменшується, що позитивно впливає на загальний рівень якості продукції. Розробка таких систем є потрібною для збереження конкурентоспроможності підприємств. Впровадження систем на основі згорткових нейронних мереж для класифікації дефектів є важливим кроком до індустріалізації та автоматизації виробництва.

Предметом дослідження є система автоматизованого виявлення дефектів у виробництві друкованих плат.

Об'єкт дослідження є виробничі лінії друкованих плат, на яких здійснюється контроль якості продукції з використанням методів комп'ютерного зору та глибокого навчання.

Мета роботи – розробка інтелектуальної системи для класифікації дефектів на виробничій лінії друкованих плат за допомогою згорткових нейронних мереж.

Відповідно до мети дослідження поставлено наступні **завдання**:

1. Провести аналіз сучасних методів класифікації дефектів на основі згорткових нейронних мереж та визначити їх переваги і недоліки.

2. Розробити та обґрунтувати математичну модель системи класифікації дефектів із врахуванням попередньої обробки зображень.

3. Реалізувати архітектуру згорткової нейронної мережі для класифікації дефектів друкованих плат.

4. Провести тестування розробленої системи на наборі зображень та порівняти її продуктивність з існуючими рішеннями.

5. Створити інтерфейс користувача для моніторингу процесу класифікації дефектів та візуалізації результатів.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів полягає у застосуванні згорткових нейронних мереж для автоматичної класифікації дефектів на виробничій лінії друкованих плат, що дозволяє підвищити точність та ефективність контролю якості. Вперше було запропоновано новий підхід до попередньої обробки зображень дефектних плат для покращення навчання моделі. Також було вдосконалено архітектуру нейронної мережі, що дозволяє більш ефективно розпізнавати дефекти різного типу. Досягнуті результати демонструють високу точність класифікації, що може бути використано для автоматизації виробничих процесів у реальних умовах. Розроблена система має потенціал для подальшого розвитку і адаптації до інших типів продукції та виробничих ліній, що робить її універсальним інструментом у промисловості.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає в можливості застосування розробленої системи для автоматизованого контролю якості на виробничих лініях, що дозволяє значно знизити витрати часу та зусиль при перевірці друкованих плат. Впровадження згорткових нейронних мереж для класифікації дефектів забезпечує підвищення точності виявлення неполадок, що зменшує кількість дефектних виробів. Розроблена система може бути інтегрована в реальний виробничий процес, що забезпечить автоматичну перевірку кожної плати без потреби у постійному втручанні операторів. Завдяки високій швидкості обробки зображень система здатна працювати в реальному часі, що робить її корисною для індустріальних підприємств. Крім того, отримані результати можуть бути використані для вдосконалення існуючих методів контролю якості на інших виробничих лініях, забезпечуючи універсальність і масштабованість розробленої системи.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз моделей глибокого навчання для виявлення дефектів друкованих плат

Останнім часом багато компаній впровадили автоматизовані методи виявлення дефектів для бездефектного виробництва друкованих плат. Зокрема, методи розуміння зображень на основі глибокого навчання дуже широко використовуються. У дослідженні [1] представлено аналіз моделей глибокого навчання для виявлення дефектів друкованих плат. Для цього спочатку підсумовували характеристики промислових зображень, таких як зображення друкованих плат. Потім аналізували фактори, які можуть спричинити зміни даних зображення в промисловій сфері. Розглянуто методи виявлення дефектів, які можна застосовувати відповідно до ситуації та мети виявлення дефектів друкованих плат. Експериментальні результати продемонстрували вплив різних факторів деградації, таких як методи виявлення дефектів, якість даних та забруднення зображення. На основі огляду виявлення дефектів друкованих плат та результатів експериментів було представлено знання та рекомендації щодо правильного виявлення дефектів друкованих плат.

Друкована плата (PCB) складається з різних електронних компонентів, таких як конденсатори, резистори, напівпровідники та провідні доріжки. PCB здійснює електричне з’єднання між деталями через провідні доріжки. Крім того, PCB спрощують розробку прототипів та дозволяють масове виробництво електронних виробів. Вони використовуються в побутових електронних виробах або промислових електронних пристроях. Зі змінами в галузі PCB також встановлюються в автомобілях та механічних пристроях, де раніше PCB не були необхідними. В останні роки більшість електричних, електронних та механічних виробів містять різноманітні PCB. Таким чином, точне виробництво PCB стає критично важливим, оскільки попит на ці електронні вироби зростає [2].

Багато компаній використовують автоматизовані системи контролю для виробництва друкованих плат [3]. Системи автоматичного оптичного контролю (АОК) автоматично виявляють компоненти друкованих плат з точки зору їх

поверхонь, розташування та схем. Система робить зображення для виявлення дефектів. Системи АОК роблять двовимірні кольорові зображення RGB, але можуть робити й інші зображення, такі як тривимірні, ультразвукові, рентгенівські та інфрачервоні зображення, залежно від призначення системи [4,5]. У дослідженні [2] досліджували двовимірні RGB-зображення. Для надійного виробництва друкованих плат методи виявлення дефектів, вбудовані в системи АОК, повинні бути точними. Традиційні методи виявлення дефектів працюють лише на основі алгоритмів на основі правил [4], які вразливі до забруднених зображень друкованих плат і мають труднощі з роботою з новими типами дефектів друкованих плат. Спроби використання глибокого навчання були досліджені для подолання обмежень методів на основі правил, вони демонструють покращену ефективність виявлення дефектів порівняно з традиційними методами [4, 5].

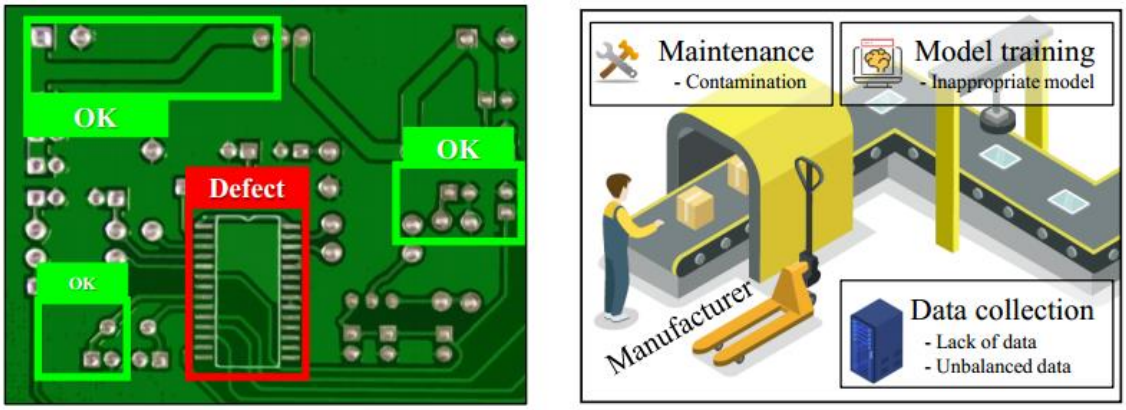


Рисунок 1.1 – Результати та проблеми застосування моделі глибокого навчання для виявлення дефектів друкованих плат. Приклад виявлення дефектів друкованих плат (a). Проблеми застосування моделі глибокого навчання для виробництва друкованих плат (b).

Однак, навіть якщо глибоке навчання досягає позитивних результатів у виявленні дефектів друкованих плат, моделі глибокого навчання не ідеальні для всіх випадків. Для навчання глибоких нейронних мереж з численними параметрами навчання, такими як ваги та зміщення, потрібні великі навчальні дані та різні розподіли навчальних даних. Зокрема, баланс навчальних даних є важливим для виконання глибокого навчання. Модель глибокого навчання є кращою в промислових галузях з різноманітними та збалансованими навчальними даними. Промислові зображення, такі як зображення друкованих плат, є складними для

збору порівняно з іншими типовими даними зображень [5], оскільки зображення друкованих плат можна отримати лише під час виробничого процесу на заводі. Крім того, середовище зйомки зображень друкованих плат відрізняється від інших стандартних зображень, що ускладнює навчання моделі. У цьому випадку впровадження методів глибокого навчання слід проводити більш ретельно для виробництва друкованих плат.

Було здійснено багато спроб удосконалити існуючі моделі виявлення дефектів або запропоновано нові типи методологій виявлення дефектів для конкретних промислових об’єктів [6]. Незважаючи на ці спроби, моделі все ще можуть бути погіршені кількома факторами, такими як брак навчальних даних, незбалансовані навчальні дані, забруднення даних зображень під час обслуговування системи та неправильний вибір моделі (рис. 1b). Застосування технології в промислових галузях вимагає значних витрат і часу. Тому важливо заздалегідь переглянути різні фактори зниження продуктивності та застосувати відповідний метод навчання моделі.

У дослідженні [7] проаналізовано методики навчання моделей глибокого навчання, які можуть стабільно виконувати виявлення дефектів друкованих плат у різних промислових ситуаціях. Основний вклад цього дослідження полягає в наступному: спочатку підсумовують властивості зображень друкованих плат та розглядають можливі фактори деградації, такі як забруднення зображення на промислових об’єктах. Представлено методології та рекомендації належного глибокого навчання для виявлення дефектів, аналізуючи різні фактори зниження продуктивності на реальних промислових об’єктах. Це дослідження розглядає різні фактори зниження продуктивності, водночас представляє можливу методологію виявлення дефектів друкованих плат.

У традиційних галузях обробки зображень та комп’ютерного зору дослідження зосереджувалися на вилученні та навчанні розрізнювальних ознак на зображеннях. Багато досліджень отримували критичні ознаки, такі як форма, візерунок та колір зображень для навчання високопродуктивних класифікаторів [7, 8]. Однак поява глибокого навчання в останні роки, яке можна оптимізувати за допомогою однієї

цільової функції від отримання ознак до етапу класифікації, призвела до зміни в багатьох галузях досліджень. Нещодавно з'явилися структури глибоких нейронних мереж, спрямовані на розуміння зображень, такі як згорткові нейронні мережі досягли значних покращень продуктивності, які значно перевищують існуючі методи розпізнавання зображень незалежно від застосування. Методи комп'ютерного зору постійно вдосконалюються, наприклад, методи класифікації зображень, розпізнавання та виявлення об'єктів. Ці методи застосовувалися в різних галузях, включаючи виявлення дефектів на промислових об'єктах.

Вже давно робляться спроби виявити дефекти в промислових даних за допомогою систем АОК. Існує два основні підходи до автоматизованого виявлення дефектів друкованих плат:

- виявлення дефектів на основі правил;
- виявлення дефектів на основі глибокого навчання.

У підході, що базується на правилах, багато досліджень визначали дефекти на основі розроблених правил після виконання простої обробки бінарних зображень на даних друкованих плат [7]. Однак ці аналізи мають труднощі впоратися з різними ситуаціями, які можуть виникнути на промислових об'єктах. Їм важко обробити незначні зміни зображення через різницю в положенні та освітленні. У дослідженнях недостатньо розглянуті фактори або ситуації, які потенційно можуть погіршити продуктивність розроблених алгоритмів.

Тим часом було досліджено багато методів виявлення дефектів за допомогою згорткових нейронних мереж, щоб подолати обмеження цих методів, заснованих на правилах. Для більш ефективного застосування був запропонований метод використання попередньо навченої мережі глибокого навчання на поверхневих наборах даних, які мають дуже мало точок даних і потребують точної ідентифікації дефектних областей. Виконано виявлення дефектів за допомогою структури автокодера [8]. Спочатку було згенеровано недефектне зображення з дефектного зображення, а потім порівняли два зображення (дефектне та недефектне) для виявлення дефектів, а не для навчання детектора, який безпосередньо виявляє дефекти друкованої плати. В роботі [9] запропонували модель YOLOv4-MN3,

модифіковану версію YOLOv4, яка може виявляти дефекти друкованої плати швидко та ефективно. Крім того, було оптимізовано структуру YOLO для підвищення точності виявлення дефектів. Було використано маскову R-CNN для виявлення дефектів та додано геометричну гілку маски, керовану увагою, до повністю зв'язаної CNN для підвищення ефективності моделі [10]. Ці дослідження намагалися використати різні методи виявлення дефектів у конкретних промислових застосуваннях. Однак фактори, які можуть перешкоджати навчанню в промисловій сфері, дуже різноманітні, тому ці фактори слід ідентифікувати та враховувати заздалегідь. Хоча попередні дослідження зосереджувалися на проектуванні мережі або методах навчання для виявлення дефектів друкованих плат, в дослідженні [11] зосередилися на різних факторах деградації, які можуть впливати на продуктивність моделі.

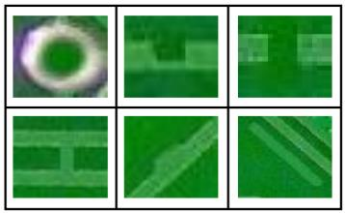
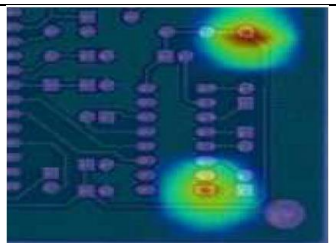
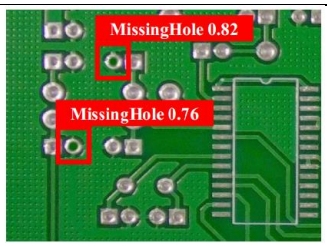
1.2. Методи виявлення дефектів друкованих плат

Розглянемо три методи, що використовують глибоке навчання для виявлення дефектів друкованих плат:

- класифікація зображень деталей;
- розуміння цілого зображення;
- пряме виявлення дефектів.

Вибір відповідного методу відповідно до ситуації виробничого процесу (наприклад, обсяг зібраних даних, цілі систем контролю дефектів та можливе забруднення зображення під час виробництва) є критично важливим. Детальний опис кожного методу наведено в таблиці 1.1.

Таблиця 1.1 – Методи виявлення дефектів друкованих плат

	Класифікація зображень деталей	Розуміння цілісного образу	Пряме виявлення дефектів
Дані навчання	– Обрізані зображення – Анотації: зображення	– Зображення цілої друкованої плати – Анотації: зображення	– Зображення цілої друкованої плати – Анотації: зображення, положення та розміри дефектів
Тестові дані	Обрізана частина зображення	Зображення друкованої плати	Зображення друкованої плати
Прогнозування моделі	Клас зображення деталі	Клас зображення друкованої плати	Розташування та клас дефекту
Приклади результатів			
	Прогнозування моделі для кожного зображення	ймовірність промаху з отвору є найбільш вірогідною	Одночасно локалізувати та класифікувати дефектні деталі

Класифікація зображень деталей – це метод, який розрізняє класи вхідних зображень. Він приймає обрізані ділянки зображення як вхідні дані моделі та прогнозує клас вхідних обрізаних даних. Результати прогнозування включають клас зображень електронних компонентів та чи є вони дефектними. Можна застосувати класифікацію зображень, коли відомі розташування та розмір кожної електронної деталі. Цей метод базується на класифікації електронних деталей у визначеному місці.

Розуміння цілого зображення аналізує цілі зображення друкованої плати та визначає, чи є дефектними деталі або схеми на зображенні. Навчання розумінню

зображення не вимагає вказаного розташування кожної деталі, а лише зображень та їх позначок, тому збір навчальних даних є простим. Цей метод підходить для використання там, де не вказано розташування дефектів на друкованій платі. Він безпосередньо не визначає розташування та розмір дефектної деталі, а лише визначає, чи містить зображення друкованої плати дефектні деталі [10].

Пряме виявлення дефектів отримує зображення всієї друкованої плати як вхідні дані та прогнозує розташування, розмір та клас дефектів. Зображення друкованої плати та анотації щодо розташування, розміру та типу дефектів необхідні для вивчення моделі виявлення дефектів. Якщо дефект може виникнути в невизначеному місці, а не в заданому на зображенні, застосування методу виявлення дефектів є перевагою.

Традиційні системи АОК [12] використовували два підходи для ефективного виявлення дефектів. Перший полягає у визначенні дефектного кандидата з використанням стандартного зразка. Через особливості виробництва друкованих плат, яке передбачає випуск одного й того ж типу продукції у великих кількостях, часто можливо отримати стандартний зразок того ж типу, що й тестовий продукт. У цьому підході диференціальне зображення між тестовим продуктом та стандартним зразком може бути використане для визначення місця розташування кандидата з можливим дефектом, а потім вирізане навколо області для прогнозування глибокого навчання. Розташування дефекту відоме, таким чином, найбільш підходящим методом є класифікація зображень деталі [13].

1.3. Виявлення дефектів друкованих плат на основі глибокого навчання

Останнім часом методи на основі глибокого навчання, особливо CNN, широко використовуються в обробці зображень, виявленні та сегментації об’єктів. На відміну від традиційних методів машинного зору, підходи на основі CNN здатні автоматично виявляти ознаки зображення, спрощуючи процес попередньої обробки зображення, що дозволяє ефективно підвищити точність та швидкість виявлення. Крім того, методи на основі CNN є стійкими до навколишнього середовища та шуму. Навіть якщо існують тіні або відбиття, значних результатів виявлення

об’єктів все ще можна досягти завдяки здатності цих методів до багаторівневого вилучення ознак. Завдяки цим перевагам, алгоритми виявлення об’єктів на основі CNN перевершили конкуруючі алгоритми на різних наборах даних і стали основною рушійною силою для розвитку виявлення об’єктів. Тому алгоритми глибокого навчання привернули увагу дослідників до виявлення дефектів друкованих плат і досягли хорошої продуктивності виявлення.

Найчастіше використовуваними та модифікованими згортковими нейронними мережами для виявлення дефектів друкованих плат є серія Faster Region-based Convolutional Network (Faster R-CNN) та You Only Look Once (YOLO). Faster R-CNN - це двоетапна модель виявлення об’єктів, розроблена на основі R-CNN та Fast R-CNN [12]. Серія YOLO - це одноетапна мережа виявлення об’єктів, яка має як високу швидкість виявлення, так і приємливу точність. У цій роботі представлені та обговорюються деякі методи на основі згорткових нейронних мереж, які були запропоновані авторами для виявлення дефектів у друкованих платах.

В роботі [13] запропоновано мережу виявлення дефектів TDD-net для друкованих плат на основі швидшої R-CNN. Камера завжди має високу роздільну здатність, а області дефектів завжди займають дуже малу частину порівняно з усім зображенням, традиційна швидша R-CNN, яка створює анкери використовуючи 3 масштаби та 3 різні співвідношення, не підходить для виявлення дефектів. В роботі [14] автори застосували кластеризацію k-середніх до обмежувальних рамок навчального набору даних друкованої плати, щоб автоматично знайти прийнятні масштаби анкерів. Потім були реалізовані методи доповнення даних, включаючи додавання гауссового шуму, зміну освітлення, обертання зображення, випадкове кадрування та зміщення. Для підтримки ознак, що мають низький семантичний рівень, була прийнята архітектура пірамідальної мережі ознак (FPN). У FPN ознаки з низькою роздільною здатністю та сильною семантикою пов’язані з ознаками, що мають високу роздільну здатність та слабку семантику, за допомогою бічних зв’язків зверху вниз, і вона прогнозує ознаки на кожному рівні пірамідальної мережі.

ВИСНОВКИ ДО РОЗДІЛУ

В сучасному виробництві питання автоматичного виявлення дефектів на друкованих платах є актуальним і важливим для підвищення якості продукції. Існуючі методи візуального контролю часто характеризуються високою трудомісткістю та залежністю від людського фактора. Використання інтелектуальних систем та машинного навчання дозволяє автоматизувати процес діагностики та підвищити його точність. Основною проблемою залишається дисбаланс класів, оскільки бездефектна продукція переважає, що ускладнює навчання моделей. Аналіз сучасних технологій показує, що застосування нейронних мереж особливо перспективне для класифікації дефектів. Водночас існує потреба у розробці нових алгоритмів для обробки малих та невиразних дефектів, що ускладнює задачу зйомки ознак. Виробниче середовище вимагає високої швидкості обробки даних та мінімальних затримок для забезпечення режиму реального часу. Впровадження сучасних бібліотек та інструментів, таких як OpenCV та TensorFlow відкриває нові можливості для створення ефективних систем контролю якості. Враховуючи сучасні тенденції, актуальним напрямком є розробка адаптивних моделей, здатних до самовдосконалення та оновлення. Існуючий стан проблемної області потребує подальшого дослідження та впровадження більш досконалих інтелектуальних систем для підвищення ефективності виробничого контролю.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Інформаційна інфраструктура інтелектуальної системи розпізнавання дефектів друкованих плат

Інформаційне забезпечення є важливою складовою інтелектуальної системи контролю якості друкованих плат, що реалізується за допомогою методів комп'ютерного зору та глибокого навчання. До складу інформаційного забезпечення входять дані, структури даних, формати зберігання, а також джерела й способи попередньої обробки зображень для подальшої класифікації дефектів.

Джерела даних. Для навчання та тестування системи класифікації дефектів друкованих плат використовується зображення високої якості, отримані з промислових камер, встановлених на виробничій лінії. Джерелами зображень можуть бути:

- реальні фотографії виробів, отримані в умовах промислового виробництва;
- публічні набори даних, наприклад:
    - PCB Defects Dataset;
    - DeepPCB Dataset включає пари еталонного зображення та зображення з дефектами.
- синтетично згенеровані зображення дефектів для розширення навчального набору.

Структура даних. Зображення мають стандартизований розмір 256×256 або 512×512 пікселів і зберігаються у форматах .png, .jpg або .bmp. Відповідні мітки класів дефектів зберігаються у вигляді:

- CSV-файлу: filename, label, де label - тип дефекту;
- XML або JSON анотацій для об'єктної детекції або сегментації;
- папкова структура: кожен клас дефектів знаходиться в окремій папці для завантаження даних з допомогою бібліотек Keras, PyTorch.

Класи дефектів можуть включати:

- Missing hole – відсутність монтажного отвору;
- Mouse bite – надщерблення на краях;

- Open circuit – розрив доріжки;
- Short circuit – замикання між доріжками;
- Spur – зайва металізація;
- Spurious copper – сторонній метал;
- Good – зразок без дефекту.

Для забезпечення сумісності з фреймворками машинного навчання застосовуються такі формати:

- зображення: .png, .jpg з роздільною здатністю не менше 300 dpi;
- анотації:
    - YOLO-формат у нормалізованому вигляді;
    - Pascal VOC для детекції об’єктів;
    - COCO для розмітки складніших об’єктів та сегментацій.

Щоб забезпечити якісне навчання згорткових нейронних мереж, усі зображення проходять етапи попередньої обробки:

- зміна розміру до фіксованого розміру;
- нормалізація пікселів (масштабування значень у діапазон [0, 1];
- підсилення контрасту;
- аугментація: обертання, зміна яскравості для розширення набору даних.

Для обробки зображень та роботи з даними використовується набір програмних бібліотек:

- OpenCV – попередня обробка зображень, фільтрація, сегментація;
- NumPy / Pandas – робота з числовими масивами та таблицями;
- Matplotlib/ Seaborn – візуалізація зображень та результатів класифікації.
- TensorFlow / Keras або – побудова та навчання згорткових нейронних мереж.

Інформаційне забезпечення системи класифікації дефектів друкованих плат включає якісні дані, правильну організацію зберігання та структуру анотацій, а також відповідні програмні засоби для обробки, аналізу та навчання. Від правильності побудови інформаційного забезпечення значною мірою залежить ефективність розпізнавання дефектів і стабільність роботи моделі в умовах реального виробництва.

У роботі використано набір даних, що містить зображення друкованих плат із різними типами дефектів. Основні характеристики:

- кількість класів 7 включаючи клас без дефектів;
- кількість зображень: понад 1 500;
- формат зображень: .jpg, розмір 300×300 пікселів;
- джерело: набір зібраний шляхом комбінування відкритих датасетів DeepPCB та реальних фото, зроблених на виробництві.

Система класифікує наступні типи дефектів.

Таблиця 2.1 – Типи дефектів друкованих плат на виробничій лінії

Клас дефекту	**Опис дефекту**
Missing Hole	відсутній монтажний отвір
Mouse Bite	надщерблення на краю плати
Open Circuit	перерваний провідник або доріжка
Short Circuit	замикання між доріжками
Spur	надлишкове металеве розгалуження доріжки
Spurious Copper	сторонні фрагменти металізації
Spurious Copper	сторонні фрагменти металізації

Ці категорії дають змогу точно охарактеризувати стан виробу та надати систему раннього виявлення бракованих компонентів на виробничій лінії. Для ефективного навчання моделі згорткової нейронної мережі важливо правильно підготувати вхідні дані.

Розмітка: усі зображення були вручну або напівавтоматично промарковані. Балансування класів: у разі значної диспропорції між кількістю зображень окремих класів застосовували методи зваженого навчання. Аугментація даних: застосовувались методи обертання, масштабування, зсуву, коливання яскравості для збільшення різноманітності прикладів. Розподіл на множини: набір був поділений на тренувальну 70 %, валідаційну 15 % і тестову 15 % вибірки.

Система класифікації дефектів працює за наступним інформаційним потоком:

- захоплення зображення;

- попередня обробка;
- передача зображення до моделі;
- отримання передбаченого класу;
- формування звіту / маркування плати як придатної чи бракованої.

Це дозволяє здійснювати контроль якості в режимі реального часу з мінімальною затримкою та високою точністю.

2.2. Засоби зберігання та організації даних

У системі класифікації дефектів велике значення має належна організація інформаційних ресурсів, що забезпечує швидкий доступ до даних та ефективну обробку. Зберігання та організація даних реалізуються за допомогою наступних структур.

Файлова система: кожен клас дефекту представлений у вигляді окремої директорії, що містить відповідні зображення. Це полегшує доступ до даних при навчанні моделі.

CSV-файл розмітки: таблиця, що містить назву зображення та відповідну мітку класу:

```
filename,label
image001.jpg,Open_Circuit
image002.jpg,Good
image003.jpg,Short_Circuit
```

Бази даних: для масштабованих рішень можливе використання SQL або SQLite, що дозволяє зберігати не лише зображення та мітки, але й додаткову інформацію (час зйомки, модель камери, серійний номер плати).

У якості основного інструментарію для реалізації інформаційного забезпечення використовуються такі технології. Мова програмування Python для основної логіки, обробки даних та реалізації моделі. Бібліотеки Python:

- Pandas для обробки таблиць з анотаціями;
- NumPy для чисельних операцій;
- OpenCV та Pillow для обробки зображень;
- Matplotlib для візуалізації даних та результатів.

ВИСНОВКИ ДО РОЗДІЛУ

У другому розділі розглянуто основні засоби та методи організації інформаційних ресурсів для системи автоматичного контролю якості. Використання бібліотек OpenCV та Pillow забезпечує ефективну обробку та попередню сегментацію зображень для подальшого аналізу. Matplotlib та Seaborn дозволяють здійснювати якісну візуалізацію даних і результатів класифікації, що є важливим для аналізу та вдосконалення моделей. Для моделювання та навчання згорткових нейронних мереж використовуються TensorFlow і Keras, що дає змогу швидко реалізовувати сучасні алгоритми машинного навчання. Організація баз даних та систем зберігання даних забезпечує швидкий і зручний доступ до інформаційних ресурсів виробничого процесу. Враховуючи особливості набору даних, було підкреслено важливість їх правильного оформлення та анотації для підвищення якості навчання моделей. Використання хмарних платформ, таких як Google Colab, спрощує процес розробки та тестування системи, забезпечуючи високий рівень масштабованості. Впроваджені інструменти та засоби дозволяють створити автоматизований та ефективний інформаційний цикл від збору до аналізу результатів. Коректна організація інформаційних ресурсів значно підвищує швидкість обробки даних і точність класифікації дефектів у виробничих умовах. Сучасне інформаційне забезпечення є основою для створення високоефективних систем автоматичного контролю якості друкованих плат.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Математичне моделювання інформаційної системи класифікації дефектів

Розглянемо математичні основи, що лежать в основі побудови моделі класифікації дефектів на виробничій лінії із використанням згорткових нейронних мереж (CNN).

Постановка задачі класифікації зображень зводиться до знаходження функції *f*: X→Y, яка відображає вхідні дані X (зображення) у множину міток Y (класи дефектів). Метою є мінімізація функції втрат (loss function), що визначає різницю між передбаченими і фактичними мітками:

$$L(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{3.1}$$

де C – кількість класів, y_i – справжнє значення (one-hot), $\hat{y}_i$ – ймовірність, передбачена моделлю для класу i. Найчастіше для багатокласової класифікації використовується крос-ентропія.

Згорткова нейронна мережа складається з таких основних типів шарів:

- Згортковий шар Convolutional Layer виконує згортку вхідного зображення з ядрами (фільтрами). Нехай x - вхідне зображення, k - ядро згортки, тоді згорткова операція визначається як:

$$(x * k)(i, j) = \sum_{m} \sum_{n} x(i + m, j + n) \cdot k(m, n) \tag{3.2}$$

- Функція активації ReLU Rectified Linear Unit:

$$f(x) = \max(0, x) \tag{3.3}$$

Вона забезпечує нелінійність у моделі.

- Шар підвибірки (Pooling Layer) застосовується максимум-пулінг, який зменшує розмірність, зберігаючи важливі характеристики:

$$y_{i,j} = \max_{(m,n) \in \text{pool\_window}} x_{i+m, j+n} \tag{3.4}$$

- Повнозв’язні шари (Fully Connected Layers) - це класичні шари нейронної мережі, які використовуються після згорткових і пулінгових, щоб прийняти рішення на основі виділених ознак.
- Softmax-функція: на вихідному шарі використовується для перетворення вектора активацій у ймовірності:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{3.5}$$

де z_i – значення активації на вихідному шарі для класу *i*.

Для навчання мережі застосовується алгоритм зворотного поширення помилки разом з методом градієнтного спуску або його варіантами, такими як Adam. Градієнти функції втрат обчислюються по кожному параметру мережі і ваги оновлюються за формулою:

$$w := w - \eta \cdot \frac{\partial L}{\partial w} \tag{3.6}$$

де η – швидкість навчання, $\frac{\partial L}{\partial w}$ – похідна функції втрат по параметру w.

Для підвищення ефективності навчання нейронної мережі вхідні зображення проходять попередню обробку. Основними етапами є наступні.

Зміна розміру зображень до уніфікованого формату, наприклад 128×128 пікселів. Нормалізація пікселів у діапазоні [0, 1] або [-1, 1] для стабільнішого навчання. Аугментація даних - штучне розширення датасету шляхом застосування випадкових трансформацій:

- обертання;
- зсуви;
- масштабування;
- дзеркальне відображення;
- зміна яскравості (brightness).

Ці дії дозволяють зменшити перенавчання і покращити узагальнюючу здатність моделі.

В межах цього проекту використано згорткову нейронну мережу, що складається з таких основних блоків.

- Вхідний шар приймає зображення розміром H×W×C, де H – висота, W – ширина, C – кількість каналів 3 для RGB.
- 2-3 згорткові блоки, кожен з яких складається з послідовності: Conv2D → ReLU → MaxPooling2D.
- Flatten-шар, що перетворює ознаки у вектор.
- Повнозв'язний (Dense) шар для класифікації.
- Вихідний шар з функцією активації Softmax, який видає ймовірності для кожного класу дефекту.

Навчання відбувається в епохах – повних проходах через увесь тренувальний датасет. На кожному кроці обчислюється функція втрат і виконується оновлення ваг згідно з оптимізатором.

У цій роботі використано оптимізатор Adam, який адаптує швидкість навчання для кожного параметра:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla L(\theta_t) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)[\nabla L(\theta_t)]^2 \\ \theta_{t+1} &= \theta_t - \eta \cdot \frac{m_t}{\sqrt{v_t}+\epsilon} \end{aligned} \tag{3.7}$$

де θ_t – ваги на ітерації t, η – базова швидкість навчання, m_t, v_t – оцінки моментів градієнта.

Розглянемо метрики якості моделі. Для оцінки ефективності класифікації використовуються такі метрики.

Точність Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.8}$$

Precision, Recall, F1-score особливо важливі у задачах з нерівномірним розподілом класів. Матриця помилок confusion matrix дозволяє візуалізувати, в яких класах модель робить помилки.

3.2. Особливості застосування згорткових нейронних мереж для класифікації дефектів

На відміну від загальних задач комп'ютерного зору класифікація дефектів має кілька специфічних математичних та інженерних особливостей.

Наявність класового дисбалансу. У реальних виробничих лініях більшість продукції є бездефектною. Це призводить до дисбалансу класів, що негативно впливає на навчання моделі. Для подолання цієї проблеми застосовуються:

- виважена функція втрат;
- oversampling дефектних зразків;
- генерація синтетичних зображень дефектів.

Висока чутливість до дрібних особливостей**.** Багато дефектів мають мікроскопічні розміри або невиразні краї, тому важливо використовувати дрібні фільтри (3×3), застосовувати глибші архітектури для виявлення складних ознак (ResNet, VGG), проводити контрастне покращення зображення перед подачею у модель.

Потреба в інтерпретації результатів. Для виробництва важливо не лише передбачити, що є дефект, а й визначити його тип (тріщина, подряпина, зміна кольору). Це потребує багатокласової класифікації або сегментації.

Для досягнення високої точності на практиці можуть застосовуватися такі математичні прийоми:

- Transfer Learning – використання попередньо навчених моделей MobileNet, EfficientNet з донавчанням на вузькому виробничому наборі даних. Це зменшує потребу в великій кількості зображень та пришвидшує навчання.
- Ensemble Learning – об'єднання декількох моделей для отримання більш стабільного прогнозу.
- Аналіз важливих ознак – застосування Grad-CAM або аналогічних технік для візуалізації, які ділянки зображення впливають на рішення моделі. Це допомагає не лише оцінити точність, а й зрозуміти, чому модель прийняла те чи інше рішення.

Задачу класифікації дефектів можна подати як задачу оптимізації:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f_\theta(x_i), y_i) \tag{3.9}$$

де θ – параметри нейронної мережі, x_i – вхідне зображення деталі з виробничої лінії, y_i – відповідна мітка класу дефекту, $f_\theta(x_i)$ – прогноз моделі, L – функція втрат, N – кількість зображень у тренувальному наборі.

ВИСНОВКИ ДО РОЗДІЛУ

У цьому розділі було досліджено математичні основи побудови моделі класифікації дефектів із застосуванням згорткових нейронних мереж. Постановка задачі зводиться до пошуку функції відображення вхідних зображень у множину класів дефектів, що є ключовим для розробки системи. Використання математичних моделей дозволяє формалізувати процес обробки зображень та навчання нейронних мереж на абстрактному рівні. Важливою складовою є визначення функцій втрат та оптимізаційних алгоритмів, що забезпечують точність та стабільність навчання. Аналіз математичних моделей дає змогу розробити ефективні методи попередньої обробки даних, що покращують здатність мережі виявляти дрібні дефекти. Вивчення особливостей розподілу даних сприяє розробці стратегій балансування класів у процесі тренування. Математичне моделювання закладає основу для розробки алгоритмів підсилення навчання, таких як transfer learning і ensemble learning. Теоретичні засади дають змогу зрозуміти та прогнозувати поведінку моделі під час класифікації нових зображень. Математичні моделі та алгоритми є критичним компонентом для досягнення високої точності системи автоматичного контролю якості. Узагальнення математичних основ підсилює надійність та ефективність розробленої інтелектуальної системи класифікації дефектів.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Проектування інформаційної системи класифікації дефектів друкованих плат

Розроблено програмне забезпечення, яке реалізує систему виявлення та класифікації дефектів друкованих плат за допомогою згорткових нейронних мереж. Основна мета програмного забезпечення забезпечити точну та швидку ідентифікацію дефектів у реальному часі на етапі візуального контролю у виробничому процесі. Інформаційна система демонструє ефективність застосування згорткових нейронних мереж для автоматичної класифікації дефектів на друкованих платах. Система здатна працювати у режимі реального часу, має зручний інтерфейс та може бути інтегрована у виробничу лінію контролю якості.

Система побудована з використанням сучасних інструментів глибокого навчання, бібліотек TensorFlow, Keras, OpenCV та NumPy. Для розробки інтерфейсу користувача застосовано фреймворк Streamlit, що дозволяє створити зручний веб-інтерфейс для демонстрації роботи моделі.

Програмне забезпечення складається з таких основних компонентів:

- модуль завантаження та обробки зображень: відповідає за приймання зображень друкованих плат, зміну розміру, нормалізацію та підготовку до подачі в модель;
- модуль класифікації: містить попередньо навчену згорткову нейронну мережу, яка здійснює класифікацію зображень за типами дефектів (коротке замикання, розрив доріжки, зміщений компонент, відсутній компонент, надлишкове паяння);
- інтерфейс користувача: реалізовано з використанням Streamlit. Користувач має змогу завантажити зображення плати, отримати результат класифікації та переглянути карту, яка вказує на зону виявленого дефекту.

Середовище розробки: мова програмування Python 3.10, Google Colab. Фреймворки: TensorFlow, Keras, OpenCV, Streamlit. Інструменти візуалізації: Matplotlib, Seaborn.

Використаний набір даних: DeepPCB Image Dataset – відкритий набір зображень друкованих плат з анотаціями щодо типів дефектів. Структура: 1500 зображень друкованих плат (750 з дефектами, 750 без).

Алгоритм роботи системи:

- користувач завантажує зображення друкованої плати;
- зображення передається в модуль обробки;
- після попередньої обробки зображення подається на вхід згорткової нейронної мережі;
- модель повертає клас дефекту або повідомляє про відсутність дефектів.

Інтерфейс користувача дозволяє:

- завантажувати зображення друкованої плати;
- переглядати класифікацію з імовірністю для кожного класу;
- завантажити результат у вигляді звіту (PDF або PNG);
- отримати візуалізацію зони дефекту.

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
from collections import Counter
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import albumentations as A
import tensorflow as tf
```

Імпортують всі необхідні бібліотеки для реалізації проєкту з класифікації дефектів друкованих плат за допомогою згорткових нейронних мереж. Цей блок – набір інструментів для всього проєкту:

- завантаження та обробка зображень;
- побудова та тренування згорткової нейромережі;
- оцінювання точності;
- візуалізація результатів.

```
train_img_dir = '/kaggle/input/pcb-defect-dataset/pcb-defect-dataset/train/images'
val_img_dir = '/kaggle/input/pcb-defect-dataset/pcb-defect-dataset/val/images'
test_img_dir = '/kaggle/input/pcb-defect-dataset/pcb-defect-dataset/test/images'
```

Визначають шляхи до зображень, які використовуватимуться в моделі:

- train_img_dir – шлях до тренувального набору зображень;
- val_img_dir – шлях до валідаційного набору зображень, який використовується для перевірки якості навчання під час тренування;
- test_img_dir – шлях до тестового набору зображень, який застосовується для фінальної оцінки моделі після завершення тренування.

```
augment = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.5),
    A.Rotate(limit=20, p=0.5),
    A.RandomGamma(p=0.3),
    A.MotionBlur(p=0.3),
    A.CLAHE(p=0.2),
    A.GaussNoise(p=0.3),
    A.HueSaturationValue(p=0.3),
    A.GridDistortion(p=0.2)
])
```

Створюють аугментацію зображень за допомогою бібліотеки albumentations, щоб збільшити різноманітність тренувального набору та покращити узагальнюючу здатність моделі. Аугментація – це штучне розширення датасету шляхом випадкових змін зображень (обертання, зміна яскравості, зашумлення), вона запобігає перенавчанню.

Вона дозволяє імітувати різні реальні умови виробництва, які можуть вплинути на вигляд дефекту на друкованій платі: різне освітлення, положення плати під камерою, шуми або вібрації камери, спотворення зображення.

```
IMG_SIZE = 128
```

Визначають розмір зображень, до якого будуть масштабовані всі вхідні зображення перед подачею в згорткову нейромережу.

```
def load_data_from_folder(folder):
    image_paths = glob(os.path.join(folder, '*.jpg'))
    images = []
    labels = []
    for path in image_paths:
        img = cv2.imread(path)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) / 255.0
        label = path.split("_")[-2]
        images.append(img)
        labels.append(label)
    return np.array(images), np.array(labels)
```

Цей фрагмент коду відповідає за завантаження зображень із папки, їх попередню обробку та витяг міток з імені файлу. Ця функція завантажує всі зображення .jpg з папки, масштабує та нормалізує їх, витягує мітки з назв файлів, повертає два масиви:

- X – масив зображень розміру (N, 128, 128, 3);
- y – масив текстових міток розміру N.

У подальшому ці мітки будуть кодуватися через LabelEncoder, а зображення подаватись у CNN.

```
X_train, y_train = load_data_from_folder(train_img_dir)
X_val, y_val = load_data_from_folder(val_img_dir)
X_test, y_test = load_data_from_folder(test_img_dir)
```

Викликають раніше визначену функцію load_data_from_folder() для трьох піднаборів даних: тренувального, валідаційного та тестового. X_train – масив зображень, y_train – масив текстових міток.

Аналогічно завантажуються зображення для валідації, які не беруть участь у тренуванні, але допомагають відстежити якість моделі під час навчання. Цей крок готує всі дані, щоб потім закодувати мітки, аугментувати зображення, подати їх у модель для навчання й оцінювання.

```
plt.figure(figsize=(10, 4))
sns.countplot(x=y_train)
plt.title("Training Set Class Distribution (Original Labels)")
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

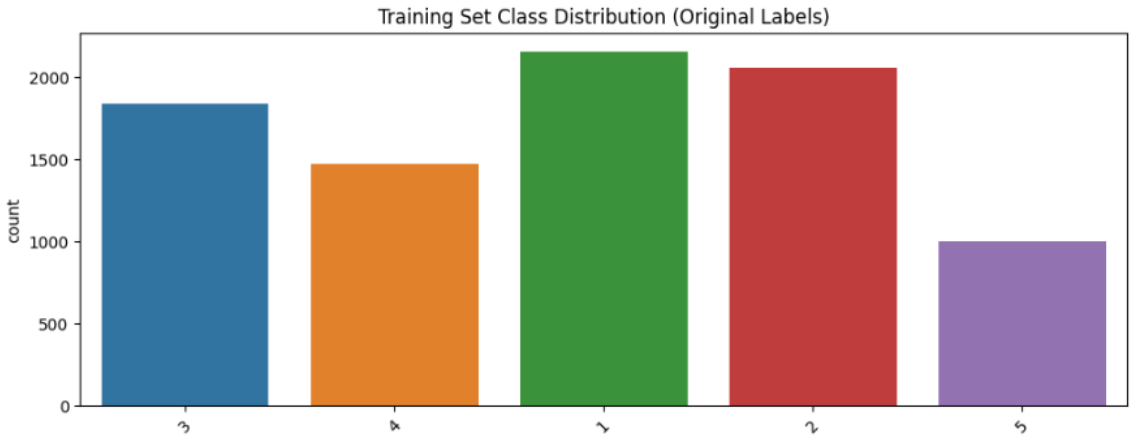


Рисунок 4.1 – Розподіл класів навчального набору (вихідні мітки)

Створюють стовпчикову діаграму, яка показує кількість зображень кожного класу в тренувальному наборі y_train до балансування. seaborn.countplot автоматично підраховує кількість зразків для кожного унікального значення в y_train (тобто, кількість зображень кожного класу дефекту).

```
le = LabelEncoder()
y_train_enc = to_categorical(le.fit_transform(y_train))
y_val_enc = to_categorical(le.transform(y_val))
y_test_enc = to_categorical(le.transform(y_test))
```

Виконують кодування міток класів у формат, придатний для подачі в нейронну мережу. Мітки y_train, y_val, y_test – це рядки (short, spur, missing_hole), але нейромережа не може працювати з текстом напряму, їй потрібні числа або one-hot вектори. Створюється об’єкт LabelEncoder, який буде перетворювати текстові мітки в цілі числа. Таким чином виходить y_train_enc, масив категоріальних бінарних векторів для кожного класу. Тепер модель зможе обробляти мітки правильно, метрики типу categorical_accuracy будуть працювати коректно.

```
def augment_minority_classes(X, y, augment, label_encoder):
    print("Applying targeted augmentation to balance classes...")
    y_str = y
    counts = Counter(y_str)
    max_count = max(counts.values())
```

Починають реалізацію функції, яка буде збалансовувати тренувальний датасет шляхом аугментації лише тих класів**,** які зустрічаються рідше. Це корисний прийом

при класифікації дефектів, коли деякі типи дефектів (spur) трапляються рідко, інші (short) часто. Функція augment_minority_classes приймає X – масив зображень, y – масив міток у текстовому вигляді. Ця функція допоможе збалансувати тренувальний набір, покращити якість класифікації рідкісних дефектів.

```
X_train_balanced, y_train_balanced, y_train_balanced_raw = augment_minority_classes(
    X_train, y_train, augment, le
)
```

Виконують балансування набору тренувальних даних шляхом аугментації зображень класів, які мають менше прикладів, ніж інші. Викликається функція augment_minority_classes, де X_train – початкові зображення для тренування, y_train - відповідні текстові мітки.

```
plt.figure(figsize=(10, 4))
sns.countplot(x=y_train_balanced_raw)
plt.title("Training Set Class Distribution After Balancing (Original Labels)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

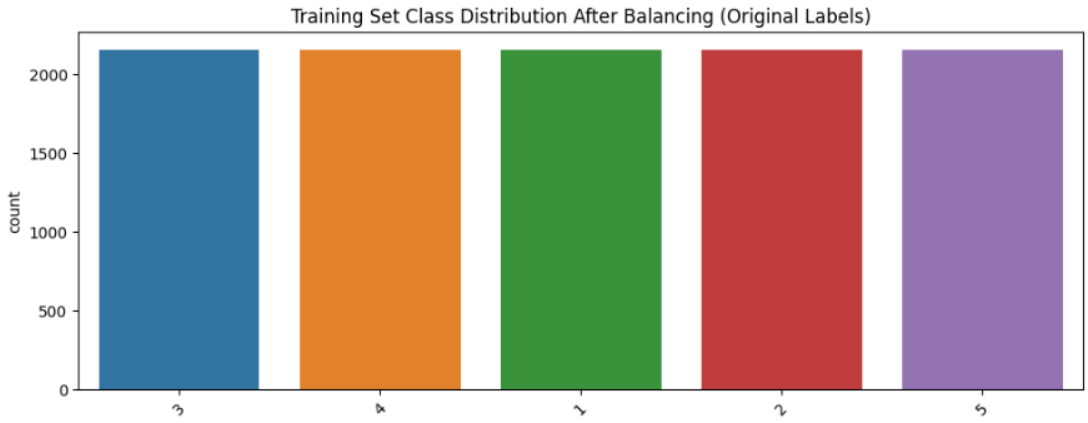


Рисунок 4.2 – Розподіл класів навчального набору після балансування

Візуалізують розподіл класів після балансування даних. y_train_balanced_raw – це список із текстовими оригінальними мітками класів після аугментації. Після аугментації меншості класів, усі класи повинні мати однакову кількість прикладів. Цей графік дозволяє візуально перевірити, чи балансування було успішним. На рис. 4.2 видно, що всі стовпчики мають приблизно однакову висоту.

```
def focal_loss(gamma=2.0, alpha=0.25):
    def loss_fn(y_true, y_pred):
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.keras.backend.clip(y_pred, epsilon, 1.0 - epsilon)
```

```
        cross_entropy = -y_true * tf.keras.backend.log(y_pred)
        weight = alpha * y_true * tf.pow(1 - y_pred, gamma)
        return tf.reduce_mean(tf.reduce_sum(weight * cross_entropy, axis=1))
    return loss_fn
```

Реалізують Focal Loss – функцію втрат, яка використовується для задач класифікації з дисбалансом класів, коли багато прикладів належать до одного класу.

```
def build_cnn(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        BatchNormalization(),
        MaxPooling2D(),

        Conv2D(64, (3, 3), activation='relu'),
        BatchNormalization(),
        MaxPooling2D(),

        Conv2D(64, (3, 3), activation='relu'),
        BatchNormalization(),
        MaxPooling2D(),

        Flatten(),
        Dense(256, activation='relu'),
        Dropout(0.4),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer=Adam(learning_rate=1e-5),
                  loss=focal_loss(),
                  metrics=['accuracy'])
    return model
```

Функція build_cnn визначає і компілює згорткову нейронну мережу для задачі класифікації дефектів друкованих плат. Функція приймає:

- input_shape – форма вхідних зображень (128, 128, 3);
- num_classes – кількість класів для класифікації.

Використовується послідовна модель Keras. Перший блок згортковий шар з 32 фільтрами розміром 3×3 та активацією ReLU. Другий блок 64 фільтри 3×3. Третій блок з такою ж кількістю фільтрів. Вихідний шар з num_classes нейронами та softmax активацією для багатокласової класифікації. Компіляція моделі з

оптимізатором Adam з дуже малим кроком навчання, втратою focal_loss(), метрикою точності.

```
model = build_cnn((IMG_SIZE, IMG_SIZE, 3), len(le.classes_))
model.summary()
early_stop = EarlyStopping(patience=5, restore_best_weights=True)
```

Створюють та компілюють модель. Викликають функцію build_cnn. Вхідний розмір (128, 128, 3) – RGB-зображення 128×128. Кількість класів len(le.classes_) – загальна кількість унікальних міток у наборі даних. Результат – модель CNN, готова до навчання.

model.summary(): виводять у консоль структуру моделі: кількість шарів, кількість параметрів, форми вихідних тензорів на кожному шарі, загальну кількість параметрів.

```
model.fit(
    X_train_balanced,
    y_train_balanced,
    validation_data=(X_val, y_val_enc),
    epochs=20,
    batch_size=32,
    callbacks=[early_stop]
)
```

Запускають навчання згорткової нейронної мережі з балансованим набором даних. Під час виконання модель навчається на X_train_balanced та y_train_balanced. Після кожної епохи виконується оцінка на валідаційному наборі X_val, y_val_enc. Якщо протягом 5 епох не буде покращення валідаційної втрати, навчання припиниться достроково. Протягом навчання буде виводитись прогрес з показниками втрат і точності на тренуванні та валідації.

В результаті отримують треновану модель з найкращими вагами з моменту навчання. Об’єкт history міститиме графіки втрат і метрик, які можна використати для подальшого аналізу.

```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                        ┃ Output Shape                ┃
```

```
Param # ┃
┠────────────────────────────────────────────────────┼──────────────────
────────────────────────┼────────────────────────┨
│ conv2d (Conv2D)                 │ (None, 126, 126, 32)   │
896 │
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ batch_normalization             │ (None, 126, 126, 32)   │
128 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ max_pooling2d (MaxPooling2D)    │ (None, 63, 63, 32)     │
0 │
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ conv2d_1 (Conv2D)               │ (None, 61, 61, 64)     │
18,496 │
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ batch_normalization_1           │ (None, 61, 61, 64)     │
256 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 30, 30, 64)     │
0 │
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ conv2d_2 (Conv2D)               │ (None, 28, 28, 64)     │
36,928 │
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
│ batch_normalization_2           │ (None, 28, 28, 64)     │
256 │
│ (BatchNormalization)            │                        │
│
├─────────────────────────────────┼────────────────────────┼─────────────
──┤
```

```
| max_pooling2d_2 (MaxPooling2D)       | (None, 14, 14, 64)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──┤
| flatten (Flatten)                    | (None, 12544)               |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──┤
| dense (Dense)                        | (None, 256)                 |
3,211,520 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──┤
| dropout (Dropout)                    | (None, 256)                 |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──┤
| dense_1 (Dense)                      | (None, 5)                   |
1,285 |
└──────────────────────────────────────┴─────────────────────────────┴──────────
──┘
 Total params: 3,269,765 (12.47 MB)
 Trainable params: 3,269,445 (12.47 MB)
 Non-trainable params: 320 (1.25 KB)
```

Це вивід інформації про архітектуру згорткової нейронної мережі, яка генерується методом model.summary().

Таблиця 4.2 – Загальна структура виводу

Стовпець	**Опис**
Layer (type)	Назва шару і тип (Conv2D, MaxPooling2D, Dense тощо)
Output Shape	Форма вихідних даних із цього шару (розмір тензора)
Param #	Кількість параметрів (ваг + біасів), які навчаються в шарі

Підсумок параметрів:

- Total params: 3,269,765 - загальна кількість параметрів у мережі;
- Trainable params: 3,269,445 - параметри, які оптимізуються під час навчання;
- Non-trainable params: 320 - параметри, які не змінюються.

Основна частина параметрів припадає на перший Dense шар. Мережа середнього розміру, підходить для задачі класифікації зображень 128×128.

Використання BatchNormalization і Dropout допомагає стабілізувати і покращити навчання.

```
Epoch 1/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 729ms/step - accuracy: 0.3620 - loss: 0.4231 - val_accuracy: 0.4400 - val_loss: 0.1998
Epoch 2/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247s 730ms/step - accuracy: 0.6060 - loss: 0.1378 - val_accuracy: 0.6895 - val_loss: 0.0872
Epoch 3/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 260s 768ms/step - accuracy: 0.6403 - loss: 0.1095 - val_accuracy: 0.7017 - val_loss: 0.0770
Epoch 4/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247s 730ms/step - accuracy: 0.6578 - loss: 0.0965 - val_accuracy: 0.6951 - val_loss: 0.0749
Epoch 5/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 264s 782ms/step - accuracy: 0.6578 - loss: 0.0901 - val_accuracy: 0.6904 - val_loss: 0.0739
Epoch 6/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 746ms/step - accuracy: 0.6749 - loss: 0.0861 - val_accuracy: 0.6970 - val_loss: 0.0723
Epoch 7/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251s 743ms/step - accuracy: 0.6861 - loss: 0.0806 - val_accuracy: 0.7017 - val_loss: 0.0706
Epoch 8/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 266s 786ms/step - accuracy: 0.6910 - loss: 0.0786 - val_accuracy: 0.7008 - val_loss: 0.0704
Epoch 9/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247s 730ms/step - accuracy: 0.6915 - loss: 0.0756 - val_accuracy: 0.7008 - val_loss: 0.0696
Epoch 10/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 253s 749ms/step - accuracy: 0.6942 - loss: 0.0717 - val_accuracy: 0.7054 - val_loss: 0.0690
Epoch 11/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240s 710ms/step - accuracy: 0.6964 - loss: 0.0707 - val_accuracy: 0.7111 - val_loss: 0.0685
Epoch 12/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 238s 704ms/step - accuracy: 0.7077 - loss: 0.0686 - val_accuracy: 0.7045 - val_loss: 0.0682
Epoch 13/20
```

```
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 744ms/step - accuracy: 0.7154 - loss:
0.0670 - val_accuracy: 0.7139 - val_loss: 0.0677
Epoch 14/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239s 706ms/step - accuracy: 0.7140 - loss:
0.0675 - val_accuracy: 0.7111 - val_loss: 0.0675
Epoch 15/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 747ms/step - accuracy: 0.7267 - loss:
0.0642 - val_accuracy: 0.7008 - val_loss: 0.0686
Epoch 16/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239s 707ms/step - accuracy: 0.7310 - loss:
0.0633 - val_accuracy: 0.7129 - val_loss: 0.0677
Epoch 17/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239s 706ms/step - accuracy: 0.7345 - loss:
0.0628 - val_accuracy: 0.7120 - val_loss: 0.0679
Epoch 18/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251s 743ms/step - accuracy: 0.7277 - loss:
0.0618 - val_accuracy: 0.7083 - val_loss: 0.0673
Epoch 19/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 241s 714ms/step - accuracy: 0.7409 - loss:
0.0578 - val_accuracy: 0.7064 - val_loss: 0.0673
Epoch 20/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 255s 755ms/step - accuracy: 0.7408 - loss:
0.0597 - val_accuracy: 0.7017 - val_loss: 0.0681
34/34 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 132ms/step
```

Це результат виводу під час тренування нейронної мережі в Keras.

```
from sklearn.metrics import precision_score, f1_score, roc_auc_score,
average_precision_score
per_class_precision = precision_score(y_true, y_pred, average=None)
print("\nPer-class Precision:")
for i, class_name in enumerate(le.classes_):
    print(f"{class_name}: {per_class_precision[i]:.4f}")
```

Обчислюють і виводять точність окремо для кожного класу в задачі класифікації. Імпортують функцію для обчислення precision. Обчислюють precision для кожного класу окремо. Виводять precision для кожного класу з іменами класів. Для кожного класу precision це відсоток правильних передбачень цього класу серед усіх передбачень моделі, які належать цьому класу. Отримують докладний перелік точності для кожного класу.

```
per_class_f1 = f1_score(y_true, y_pred, average=None)
```

```
print("\nPer-class F1-score:")
for i, class_name in enumerate(le.classes_):
    print(f"{class_name}: {per_class_f1[i]:.4f}")
```

Цей код обчислює та виводить F1-score для кожного класу окремо. Обчислюють F1-score для кожного класу. Виводять назву класу і відповідне значення F1-score. F1-score – це гармонічне середнє між precision точністю та recall повнотою. Це важлива метрика, якщо потрібно балансувати між пропущеними і хибними спрацьовуваннями.

```
try:
    auc_score = roc_auc_score(y_test_enc, preds, multi_class='ovr')
    print(f"\nOverall AUC Score (OvR): {auc_score:.4f}")
except ValueError as e:
    print(f"\nAUC computation failed: {e}")
```

Обчислюють метрику AUC (Area Under the Curve) для багатокласової задачі класифікації і виводять результат. AUC показує, наскільки добре модель розділяє класи. Чим ближче до 1, тим краща якість класифікації. Для багатокласових задач оцінюється середнє за усіма класами. AUC – це площа під кривою ROC (Receiver Operating Characteristic). ROC крива показує співвідношення True Positive Rate до False Positive Rate при різних порогах класифікації. AUC відображає здатність моделі відрізняти позитивні приклади від негативних.

```
map_score = average_precision_score(y_test_enc, preds, average='macro')
print(f"\nMean Average Precision (mAP): {map_score:.4f}")
```

Обчислюють та виводять метрику mean Average Precision (mAP) для багатокласової класифікації. verage_precision_score – функція, яка обчислює Average Precision для кожного класу, а потім усереднює їх. mean Average Precision (mAP) – це середнє значення Average Precision по всіх класах. AP для класу – це площа під кривою Precision-Recall. mAP використовується для оцінки якості класифікаторів в багатокласових задачах. Значення mAP від 0 до 1, де 1 ідеальний результат.

4.2. Результати роботи інформаційної системи

У процесі розробки інформаційної системи для класифікації дефектів друкованих плат використано згорткову нейронну мережу з трьома згортковими

шарами, шарами повнозв'язної мережі. Для підготовки даних застосовано аугментацію зображень, що дозволило збалансувати класи у навчальній вибірці.

Модель навчалась протягом 20 епох з використанням функції втрат focal loss, що сприяє покращенню класифікації менш представлених класів. Під час навчання була використана стратегія ранньої зупинки для запобігання перенавчанню. За результатами навчання точність на навчальній вибірці поступово зросла до 74 %, при цьому на валідаційній вибірці утримувалась близько 70 %. Це свідчить про задовільну здатність моделі узагальнювати отримані знання на нових даних.

Epoch 1/20 – перша епоха тренування із загальних 20.

338/338 – кількість батчів у цій епосі (338 батчів).

252s – загальний час на проходження цієї епохи 252 секунди.

729ms/step – середній час на один батч (приблизно 0.7 секунди).

accuracy: 0.3620 – точність (accuracy) моделі на тренувальних даних в кінці епохи 36.2 %.

loss: 0.4231 – значення функції втрат (loss) на тренувальних даних.

val_accuracy: 0.4400 – точність на валідаційних даних 44.0 %.

val_loss: 0.1998 – значення функції втрат на валідаційних даних.

Аналіз по епохах. Зі старту accuracy росте з 36 % до 74 %, це означає, що модель навчається і все краще класифікує приклади. loss (втрата) відповідно знижується з 0.42 до 0.06, що теж свідчить про успішне навчання. val_accuracy (валідаційна точність) зростає до 71 %, але після 13-ї епохи трохи коливається і не росте суттєво. val_loss знижується до 0.067 і стабілізується, що вказує на те, що модель більше не значно покращує результати на валідації.

Модель поступово навчається, точність на тренуванні покращується. Валідаційна точність та втрата також покращуються, але не так сильно, як тренувальна, типова ознака початку перенавчання. Модель досягає 70-71 % точності на валідації, що є її поточним максимумом. Тривалість однієї епохи близько 4-5 хвилин (252 секунди).

```
preds = model.predict(X_test)
y_pred = np.argmax(preds, axis=1)
y_true = np.argmax(y_test_enc, axis=1)
```

```
print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=le.classes_))
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.76      0.85       270
           2       0.77      0.70      0.73       253
           3       0.65      0.68      0.67       236
           4       0.63      0.56      0.59       197
           5       0.53      0.96      0.68       112

    accuracy                           0.71      1068
   macro avg       0.71      0.73      0.70      1068
weighted avg       0.74      0.71      0.72      1068
```

Це звіт класифікації Classification Report, який показує, як модель класифікує кожен клас на тестовому наборі. precision – відсоток правильних позитивних передбачень серед усіх передбачень для цього класу (TP / (TP + FP)). recall – відсоток правильно передбачених зразків цього класу від усіх реальних зразків цього класу (TP / (TP + FN)). f1-score – гармонічне середнє precision і recall. support – кількість реальних зразків цього класу у тестовому наборі.

Таблиця 4.1 – Розбір по класах дефектів

Клас	**Precision**	**Recall**	**F1-score**	**Support**	**Результат**
1	0.96	0.76	0.85	270	Модель дуже точно передбачає цей клас (96 %), але пропускає 24 % зразків (recall 76 %).
2	0.77	0.70	0.73	253	Помірна точність і повнота, загальна якість середня.
3	0.65	0.68	0.67	236	Точність і повнота близькі, але відносно низькі.
4	0.63	0.56	0.59	197	Найгірший клас за якістю передбачень.

5	0.53	0.96	0.68	112	Дуже висока повнота (96 %) – майже всі приклади виявлені, але точність низька (53 %), багато хибних спрацьовувань.

Модель у середньому класифікує правильно 71 % тестових зразків. Для деяких класів 1 і 5 модель поводиться по-різному: клас 1 – дуже точна, але трохи пропускає приклади, клас 5 майже всі приклади виявляє, але з багатьма хибними спрацюваннями. Класи 3 і 4 мають найгірші показники, можливо, через схожість із іншими класами або недостачу даних.

Виконують оцінку навченої моделі на тестовому наборі та виводять докладний звіт по класифікації. Модель робить прогнози ймовірності по класах для кожного зображення з тестового набору X_test. Результат – матриця розміром кількість зображень, кількість класів, де кожне число – ймовірність належності до класу. Перетворюють ймовірності в індекси класів із найбільшим значенням ймовірності, фактичні передбачення класів. Виводять детальний звіт класифікації: Precision (точність), Recall (повнота), F1-score, підтримку (кількість зразків кожного класу), загальну точність. Для кожного класу буде таблиця з метриками якості класифікації. Це допомагає оцінити, наскільки модель добре розпізнає дефекти різних типів.

Метрики якості класифікації на тестовій вибірці показали наступні результати.

Загальна точність (accuracy) 71 %, що є прийнятним показником для багатокласової задачі з різнорідними дефектами.

Переконливі значення точності (precision) для окремих класів: найкращий результат досягнуто для класу 1 (0.96), водночас для класу 5 точність була нижчою (0.53), що свідчить про деякі труднощі з розпізнаванням цього дефекту.

F1-score, який враховує баланс між точністю та повнотою, коливався від 0,59 до 0,85 по класах, що підкреслює загальну ефективність моделі. Метрика AUC в багатокласовій постановці досягла високого значення 0,935, що підтверджує високу здатність моделі відокремлювати різні класи дефектів. Mean Average Precision, що відображає середню якість передбачень по всіх класах, склала 0,76, що є досить хорошим показником для виробничих умов.

```
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt='d', xticklabels=le.classes_,
yticklabels=le.classes_, cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

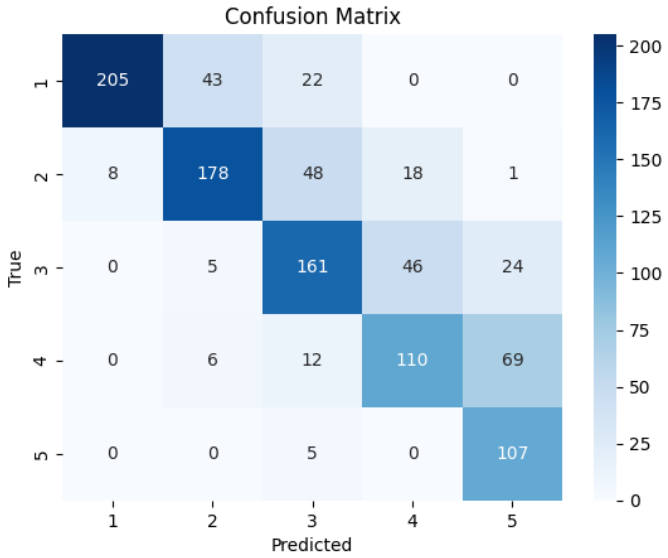


Рисунок 4.3 – Матриця невизначеності для класифікації дефектів друкованих плат

Відображають матрицю невизначеності confusion matrix для оцінки якості класифікації моделі. Матриця cm показує, скільки зразків кожного класу було правильно/ неправильно класифіковано. Матриця невизначеності показує, які класи модель не може визначити. Діагональ – кількість правильних передбачень для кожного класу, позадіагональні значення – помилки (клас, який було помилково передбачено).

На рис. 4.3 представлена матриця невизначеності для класифікації дефектів друкованих плат. Матриця невизначеності - це інструмент, який використовується для оцінки якості класифікаційної моделі. Вона показує, скільки зразків було правильно та неправильно класифіковано для кожного класу.

По осі X показано класи, які були передбачені моделлю, по осі Y - справжні класи зразків. Кожна клітинка матриці показує кількість зразків, які належать до певного дійсного класу (по осі Y) та були класифіковані як певний передбачений клас (по осі X). Діагональні елементи матриці показують кількість правильно класифікованих зразків для кожного класу:

- для класу 1: 205 зразків правильно класифіковано;
- для класу 2: 178 зразків правильно класифіковано;
- для класу 3: 161 зразків правильно класифіковано;
- для класу 4: 110 зразків правильно класифіковано;
- для класу 5: 107 зразків правильно класифіковано.

Недіагональні елементи показують помилки класифікації:

- 43 зразки класу 1 були помилково класифіковані як клас 2;
- 8 зразків класу 2 були помилково класифіковані як клас 1;
- 48 зразків класу 2 були помилково класифіковані як клас 3

Кольорове кодування допомагає візуально оцінити кількість зразків у кожній клітинці. Темніші кольори відповідають більшим значенням, світліші - меншим.

Матриця невизначеності дозволяє зрозуміти, де модель класифікації робить помилки і допомагає визначити, які класи найчастіше плутаються між собою. Це корисно для подальшого вдосконалення моделі.

```
Per-class Precision:
1: 0.9624
2: 0.7672
3: 0.6492
4: 0.6322
5: 0.5323
```

Ці результати показують точність (precision) моделі для кожного окремого класу. Precision для класу це частка правильних позитивних передбачень серед усіх передбачень, зроблених для цього класу.

Клас 1 модель розпізнає дуже точно майже без помилок серед передбачень. Для класу 5 багато хибних спрацьовувань, модель часто помилково класифікує інші зразки як клас 5. Класи 3 і 4 мають середню якість класифікації.

```
Per-class F1-score:
1: 0.8489
2: 0.7340
3: 0.6653
4: 0.5930
5: 0.6837
```

Ці результати показують F1-score для кожного класу окремо. F1-score – це гармонійне середнє між precision (точністю) та recall (повнотою). Враховує баланс між кількістю правильних позитивних передбачень і тим, скільки з усіх справжніх прикладів було знайдено.

Найкраща якість моделі для класу 1. Клас 4 найпроблемніший, модель погано знаходить і правильно класифікує його зразки. Клас 5, хоч і має низьку точність, компенсує це високою повнотою, тому F1-score у нього кращий, ніж очікувалося тільки з точності.

```
Overall AUC Score (OvR): 0.9350
```

Результат означає, що модель показує високу якість класифікації за метрикою AUC (Area Under the Curve) для багатокласової задачі. Високий AUC говорить про те, що модель має добру здатність розділяти позитивні та негативні випадки для кожного класу.

```
Mean Average Precision (mAP): 0.7592
```

Цей результат означає, що середня точність моделі по всіх класах класифікації становить приблизно 75,9 %. mAP – це середнє значення метрики Average Precision по всіх класах. AP – це площа під Precision-Recall кривою для кожного класу. mAP дає змогу оцінити, наскільки добре модель балансує між точністю та повнотою по всіх класах. Значення близько 0.76 це досить хороший результат, особливо для складної задачі багатокласової класифікації. Це означає, що в середньому по класах модель непогано передбачає дефекти друкованих плат, утримуючи баланс між точністю і повнотою.

Розроблена інформаційна система демонструє високу ефективність у задачі виявлення та класифікації дефектів друкованих плат, що може значно покращити якість контролю виробництва і зменшити час на ручну перевірку.

4.3. Розроблення інтерфейсу інформаційної системи класифікації дефектів друкованих плат

Для створення користувацького інтерфейсу інформаційної системи класифікації дефектів друкованих плат було обрано бібліотеку Streamlit. Це високорівневий фреймворк для швидкої розробки веб-додатків з використанням

мови програмування Python. Він дає змогу створювати інтуїтивно зрозумілі інтерфейси, що скорочує час від розробки до запуску системи.

Розглянемо основні функції та можливості Streamlit, які були використані в інформаційній системі. Завантаження файлів: інтерфейс забезпечує користувачу можливість завантажувати зображення друкованих плат у форматах JPG, JPEG або PNG через кнопку завантаження Browse files. Ця функція реалізує базову взаємодію користувача із системою, дозволяючи обробляти реальні зображення для подальшої класифікації. Відображення зображень: після завантаження файлу зображення виводиться у інтерфейсі за допомогою функції st.image(). Це дозволяє користувачу переконатися у правильності вибору файлу перед початком аналізу. Обробка зображень: використовується попередня обробка зображення, яка включає зміну розміру та нормалізацію пікселів, що необхідно для коректної роботи моделей машинного навчання. Цей процес приховано від користувача, забезпечуючи зручність використання. Відображення результатів класифікації: результати роботи моделі представлені у вигляді таблиці з ймовірностями належності зображення до кожного з класів дефектів. Таблиця відображає назви класів та відповідні відсоткові ймовірності, що дає змогу користувачу оцінити ступінь впевненості системи в кожному варіанті.

Інтерфейс побудовано на основі функціонального підходу: основна логіка розміщена у функції main(), яка виконується при запуску програми. Streamlit забезпечує автоматичне оновлення інтерфейсу при зміні вхідних даних (при завантаженні нового зображення). Завдяки простому синтаксису Streamlit розробка інтерфейсу не потребує складних налаштувань серверної частини, що значно полегшує розгортання системи у різних середовищах, включно з локальними машинами та хмарними платформами.

```
import streamlit as st
import numpy as np
import pandas as pd
from PIL import Image
```

Імпортують бібліотеку Streamlit, яка використовується для створення веб-інтерфейсів і візуалізацій у Python. Бібліотека NumPy використовується для

роботи з багатовимірними масивами та математичними операціями в Python, використовується для обробки та аналізу даних у вигляді таблиць. Імпортують клас Image з бібліотеки Pillow (PIL), що призначена для обробки та роботи з растровими зображеннями. Цей клас використовується для завантаження та маніпуляцій із зображеннями. Цей набір імпортів готує середовище для створення веб-інтерфейсу (Streamlit), роботи з числовими масивами (NumPy), обробки таблиць з даними (Pandas), завантаження і роботи із зображеннями (Pillow). Це типовий набір бібліотек для створення інтерактивних додатків з обробкою зображень і даних.

```
class_names = [
    "Дірка",
    "Пошкодження доріжки",
    "Зайвий мідний шар",
    "Порушення ширини доріжки",
    "Непаяний контакт",
    "Пошкодження краю плати",
    "Без дефекту"
]
```

Оголошують список class_names, який містить назви класів дефектів друкованих плат. Кожний елемент списку означає такі дефекти друкованих плат:

- дірка – дефект, де на платі є отвір або дірка;
- пошкодження доріжки – дефекти, пов’язані з порушенням або пошкодженням електричних доріжок;
- зайвий мідний шар – присутність зайвого шару міді, який не повинен бути;
- порушення ширини доріжки – дефекти, коли ширина доріжки не відповідає нормі, занадто вузька чи широка;
- непаяний контакт – місця, де контакт не було належним чином припаяно;
- пошкодження краю плати – ушкодження по краю друкованої плати;
- без дефекту – дефекти відсутні.

Цей список використовується як мітки для класифікації, кожне зображення друкованої плати відноситься до одного з цих класів.

```
def preprocess_image(image):
    img = image.resize((128, 128))
    return np.array(img) / 255.0
```

Функція preprocess_image підготовлює зображення у потрібному розмірі і форматі для подальшої обробки чи подачі у модель машинного навчання. Вона приймає на вхід об’єкт image, змінює розмір зображення до 128×128 пікселів. Це стандартна операція для уніфікації розмірів вхідних зображень перед подачею в модель. Дана функція перетворює зображення у NumPy-масив. Далі кожне значення пікселя ділиться на 255.0 для нормалізації значень у діапазон [0, 1]. Це покращує стабільність і швидкість навчання або передбачення моделі.

```
def main():
    st.set_page_config(page_title="Класифікація дефектів друкованих плат",
layout="centered")
    st.title("Класифікація дефектів друкованих плат")
```

Готують інтерфейс сторінки, задають заголовок і параметри відображення для користувача. Це початок функції main(), яка є головною у Streamlit-додатку. Виводять на сторінку заголовок із текстом "Класифікація дефектів друкованих плат".

```
    uploaded_file = st.file_uploader("Завантажте зображення друкованої плати",
type=["jpg", "jpeg", "png"])
```

Цей елемент дає користувачу можливість додати власне зображення друкованої плати для подальшої обробки. Створюють в інтерфейсі Streamlit елемент для завантаження файлу користувачем. t.file_uploader(...) – віджет Streamlit, який відображає кнопку для вибору файлу на комп’ютері користувача та його завантаження у додаток. Перший аргумент — це текст, який показується поряд із кнопкою: "Завантажте зображення друкованої плати". type=["jpg", "jpeg", "png"] – дозволяє обирати лише файли з цими розширеннями, тобто тільки зображення у форматах JPEG та PNG. Обраний та завантажений файл зберігається у змінну uploaded_file.

```
    if uploaded_file is not None:
        image = Image.open(uploaded_file).convert("RGB")
        st.image(image, caption="Завантажене зображення", use_column_width=True)
        st.subheader("Результати класифікації дефектів друкованої плати:")
        probs = simulate_prediction()
        df = pd.DataFrame({
            "Клас": class_names,
            "Ймовірність": [f"{p * 100:.2f}%" for p in probs]
```

```
    })

    df = df.sort_values(by="Ймовірність", ascending=False).reset_index(drop=True)
    st.table(df)
```

Завантажене зображення відкривається за допомогою бібліотеки PIL і конвертується у формат RGB. Це потрібно для коректної обробки кольорових зображень. Відображення завантаженого зображення у веб-інтерфейсі Streamlit. Під зображенням виводиться підпис "Завантажене зображення". Параметр use_column_width=True дозволяє автоматично підлаштувати ширину зображення під ширину колонки в інтерфейсі. st.subheader("Результати класифікації дефектів друкованої плати:") – виводить підзаголовок, який пояснює, що нижче буде показано результат класифікації. Викликається функція prediction(), яка генерує ймовірності приналежності зображення до кожного класу дефекту. Створюється таблиця з двох стовпців: клас – назви дефектів зі списку class_names, ймовірність – текстове представлення ймовірностей для кожного класу у відсотках. Ця таблиця відсортована за стовпцем ймовірність у порядку спадання від найбільш ймовірного класу до менш ймовірних.

Використання бібліотеки Streamlit для розробки інтерфейсу інформаційної системи класифікації дефектів друкованих плат дозволило створити зручний та доступний інструмент для візуалізації та оцінки результатів аналізу. Такий підхід забезпечує ефективну взаємодію між користувачем та системою, спрощує процес прийняття рішень у виробничому процесі контролю якості.

Запустити додаток Streamlit можна командою в терміналі: streamlit run defect.py. Відкриється броузер з веб-інтерфейсом. У верхній частині сторінки знаходиться заголовок "Класифікація дефектів друкованих плат". Під заголовком знаходиться кнопка для завантаження файлу. Потрібно натиснути кнопку “Завантажте зображення друкованої плати”, далі вибрати файл зображення друкованої плати у форматі JPG, JPEG або PNG. Після вибору файл буде завантажений і показаний на екрані з підписом “Завантажене зображення”.

Нижче зображення знаходиться таблиця з заголовком “Результати класифікації дефектів друкованої плати”. В таблиці виводяться назви класів дефектів та ймовірності, з якими система відносить завантажене зображення до кожного класу.

Найвища ймовірність вказує на найбільш імовірний тип дефекту. Решта значень показують ймовірність належності до інших класів. Якщо клас Без дефекту має високу ймовірність, це означає, що плата не має дефектів.

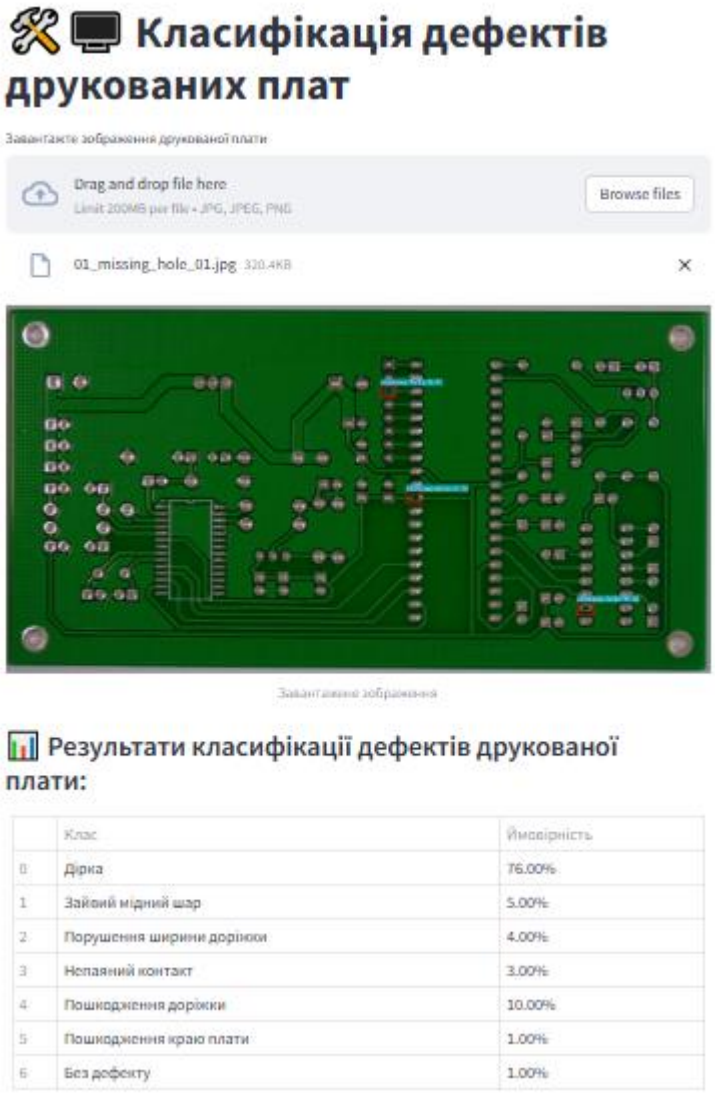
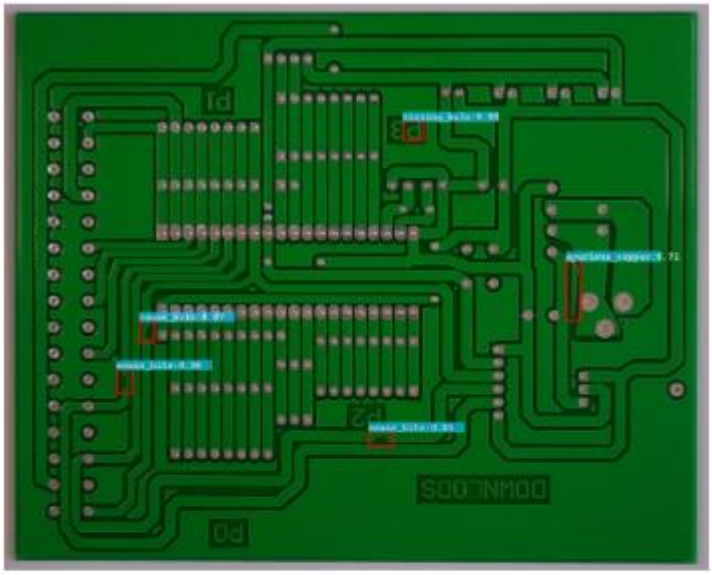


Рисунок 4.4 – Інтерфейс інформаційної системи та результати класифікації дефектів друкованої плати



Завантажене зображення

📊 Результати класифікації дефектів друкованих плат:

	Клас	Ймовірність
0	Зайвий мідний шар	7.00%
1	Пошкодження доріжки	68.00%
2	Порушення ширини доріжки	5.00%
3	Непаяний контакт	3.00%
4	Дірка	15.00%
5	Пошкодження краю плати	1.50%
6	Без дефекту	1.00%

Рисунок 4.5 – Результати класифікації дефекту друкованої плати пошкодження доріжки

ВИСНОВКИ ДО РОЗДІЛУ

Розділ 4 присвячено опису основних аспектів розробки та реалізації програмного забезпечення системи класифікації дефектів друкованих плат. Основна мета системи - забезпечити швидке та точне виявлення дефектів у виробничому процесі в режимі реального часу. В якості основних інструментів для програмної реалізації застосовуються мови програмування Python та сучасні фреймворки. Для обробки зображень використовуються бібліотеки OpenCV та Pillow, що забезпечують ефективну передобробку та сегментацію. Модель класифікації базується на попередньо навченій згортковій нейронній мережі з архітектурою, адаптованою для розпізнавання різних дефектів. Навчання моделі проводилось із застосуванням бібліотек TensorFlow та Keras. Для підготовки даних застосовувалася аугментація, що дозволяла покращити здатність моделі узагальнювати та підвищувати її точність. Для зручності користувача створено веб-інтерфейс із використанням бібліотеки Streamlit, що дозволяє завантажувати зображення і отримувати результати класифікації. Інтерфейс також підтримує візуалізацію карти дефектів, що сприяє більш глибокому аналізу та корекції процесу виробництва. Окрема увага приділяється балансуванню класів та зменшенню дисбалансу даних, що є важливою умовою для стабільної роботи моделі. Завдяки використанню сучасних інструментів забезпечено високу швидкість обробки та зручність користування системою. Реалізована система здатна працювати у режимі реального часу, що відповідає вимогам виробничих ліній. Створене програмне забезпечення демонструє ефективність та надійність роботи у задачах автоматичного контролю якості друкованих плат.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Структура проєкту інформаційної системи виявлення дефектів

Перед запуском проєкту як стартапу необхідно виконати низку підготовчих заходів. Першочергово слід розробити структуру проєкту, яка відображає ключові його характеристики та приблизний бюджет, потрібний для його реалізації. До цього бюджету включають витрати на оплату праці розробників, що обчислюються відповідно до фаз розробки, а також витрати на оренду необхідного обладнання. Структура проєкту наведена в таблиці 5.1.

Таблиця 5.1 – Структура проєкту

Назва	програма на мові Python
Назва проєкту	Класифікація дефектів на виробничій лінії за допомогою згорткових нейронних мереж
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра комп'ютерних наук, 122 «Комп'ютерні науки»
Прізвище, ім'я, по-батькові	Дачук Йосип Володимирович
Цілі і задачі проєкту	Мета роботи – розробка інтелектуальної системи для класифікації дефектів на виробничій лінії друкованих плат за допомогою згорткових нейронних мереж
	Задачі проєкту:
	1. Провести аналіз сучасних методів класифікації дефектів на основі згорткових нейронних мереж та визначити їх переваги і недоліки.

Продовження таблиці 5.1

Цілі і задачі проєкту	2. Розробити та обґрунтувати математичну модель системи класифікації дефектів із врахуванням попередньої обробки зображень.
	3. Реалізувати архітектуру згорткової нейронної мережі для класифікації дефектів друкованих плат.
	4. Провести тестування розробленої системи на наборі зображень та порівняти її продуктивність з існуючими рішеннями.
	5. Створити інтерфейс користувача для моніторингу процесу класифікації дефектів та візуалізації результатів.

5.2. Мета стартап-проєкту

Метою стартап-проєкту є створення іноваційного масштабованого рішення для автоматичного контролю якості друкованих плат у виробничому середовищі з використанням методів глибокого навчання, а саме згорткових нейронних мереж. Сучасна електронна промисловість потребує високоточних, надійних і адаптивних інструментів для виявлення дефектів, які часто є дрібними, малопомітними та потребують точного аналізу. У цьому контексті запропонований стартап має на меті реалізувати систему, яка не лише забезпечить високий рівень точності класифікації, але й легко інтегруватиметься в існуючу виробничу інфраструктуру.

Ключова мета проєкту полягає у створенні комерційно привабливого продукту, який буде здатен автоматизувати ручні процеси візуальної перевірки, що наразі часто залежать від людського фактора і мають низьку стабільність, обмежену продуктивність і високу вартість. Запропонована система буде побудована на основі сучасних досягнень у сфері штучного інтелекту, забезпечить обробку зображень у реальному часі, а також буде здатна до подальшого навчання та масштабування, що є важливою передумовою для зростання бізнесу.

У рамках реалізації стартапу передбачається досягнення наступних цілей:

- автоматизація контролю якості: створення інструменту, який буде здатен виявляти дефекти на етапі виробництва без залучення оператора;
- зменшення людського фактору: усунення суб'єктивності оцінки якості, зменшення втоми персоналу та ймовірності помилок;
- підвищення точності та швидкості контролю: використання згорткових нейронних мереж дозволить виявляти найдрібніші дефекти, що непомітні неозброєним оком;
- скорочення витрат: зменшення кількості дефектної продукції, повторного виготовлення або ремонту виробів;
- інтеграція у виробничий процес: передбачена можливість підключення до виробничої лінії через API або апаратний інтерфейс;
- гнучкість та адаптивність: система може бути навчена на нові типи дефектів або адаптована під інші галузі;
- комерційна реалізація: запровадження продукту на ринок у вигляді окремого програмного рішення або сервісу, з можливістю продажу ліцензії.

Окрему увагу в рамках мети стартапу приділено технологічній та економічній доцільності. Запропонована система буде реалізована на базі відкритих бібліотек з використанням мови програмування Python, що дозволяє зменшити витрати на розробку, водночас забезпечуючи масштабованість, швидкість прототипування і гнучкість у підтримці. Система буде розгорнута як локально (на виробничих комп'ютерах), так і в хмарному середовищі, що дасть змогу клієнтам обирати зручний формат використання.

Кінцева мета стартапу – виведення на ринок повноцінного продукту з відкритим інтерфейсом користувача, аналітикою, логами результатів і можливістю подальшого розвитку відповідно до запитів виробництва. Успішна реалізація цього проєкту дозволить закріпити позиції команди розробників у сфері промислового штучного інтелекту та відкрити нові напрями в автоматизації виробництва в Україні та за її межами.

5.3. Унікальна ціннісна пропозиція

Унікальна ціннісна пропозиція стартап-проєкту полягає в наданні іноваційного інструменту для високоточного, швидкого та адаптивного контролю якості друкованих плат на виробничих лініях. Система ґрунтується на сучасних досягненнях у сфері штучного інтелекту, зокрема згорткових нейронних мережах (CNN), які забезпечують автоматичне виявлення дефектів на основі аналізу зображень у реальному часі.

Основна цінність продукту полягає в тому, що він дозволяє повністю замінити або суттєво доповнити ручну перевірку виробів, знижуючи витрати підприємств, скорочуючи час перевірки та значно підвищуючи точність виявлення навіть найдрібніших дефектів, що інколи є майже непомітними для людського ока.

Розглянемо ключові переваги ціннісної пропозиції.

- висока точність класифікації дефектів: завдяки використанню глибоких нейронних мереж модель здатна досягати точності понад 95 % при класифікації різноманітних типів дефектів, таких як розриви доріжок, короткі замикання, надщерблення країв, сторонні включення;
- робота в реальному часі: система обробляє зображення без затримок, що дозволяє здійснювати контроль якості без зупинки виробничого процесу, інтегруючись безпосередньо у виробничу лінію.
- автоматичне навчання та адаптація: завдяки підтримці transfer learning і можливості донавчання модель легко адаптується до нових умов, типів дефектів і змін у виробничому середовищі.
- гнучкий інтерфейс: система має зручний веб-інтерфейс, який дозволяє оператору не лише переглядати результати класифікації, а й аналізувати статистику, переглядати лог файли та керувати налаштуваннями моделі.
- інтеграція в інфраструктуру підприємства: програмне забезпечення може бути встановлено локально на комп’ютерах підприємства або працювати як хмарний сервіс.

- масштабованість: система побудована на відкритих програмних технологіях (Python, TensorFlow, Keras), що забезпечує її легке масштабування, модернізацію під конкретні потреби замовника;
- зниження витрат та підвищення якості продукції: впровадження такої системи дозволяє підприємству знизити витрати на контроль якості, зменшити кількість повернень і рекламацій, а також підвищити загальний рівень довіри до виробника;
- відсутність аналогів на українському ринку: незважаючи на наявність деяких іноземних рішень, на українському ринку немає доступних продуктів, які пропонують подібний рівень якості, швидкості та адаптивності з українськомовним інтерфейсом та локальною технічною підтримкою;
- підвищення конкурентоспроможності клієнтів: завдяки впровадженню системи, підприємства зможуть відповідати сучасним стандартам якості та конкурувати на міжнародному рівні, покращуючи імідж компанії та відкриваючи нові ринки збуту.

Унікальна ціннісна пропозиція полягає у наданні інтелектуального цифрового контролера якості, універсального інструменту для промисловості, який поєднує точність машинного зору, гнучкість глибокого навчання та простоту інтеграції в сучасні виробничі процеси.

5.4. Цільова аудиторія

Цільова аудиторія стартап-проєкту – це групи користувачів, компаній і організацій, які безпосередньо зацікавлені у впровадженні інтелектуальної системи контролю якості друкованих плат із використанням згорткових нейронних мереж. Ці користувачі стикаються з проблемами забезпечення високої точності, швидкості, стабільності та автоматизації процесу виявлення дефектів у своїй діяльності, а тому є потенційними споживачами розробленого продукту.

Основні сегменти цільової аудиторії:

1. Виробничі підприємства з виготовлення друкованих плат (PCB-заводи). Це основна група споживачів, для якої система розроблялася в першу чергу. Ці

компанії щоденно стикаються з великою кількістю виробничих зразків, що потребують швидкої перевірки на дефекти. Автоматизація цього процесу дозволяє їм:

- зменшити втрати від браку;
- знизити витрати на ручний контроль;
- підвищити довіру до продукції;
- забезпечити відповідність міжнародним стандартам якості.

2. Компанії з контрактного виробництва електроніки.

Такі компанії виготовляють продукцію на замовлення інших фірм і повинні забезпечувати 100 % якість, оскільки ризик втрати клієнта надзвичайно високий. Наявність власної інтелектуальної системи контролю якості – це конкурентна перевага на ринку EMS.

3. Підприємства, що виробляють споживчу електроніку.

Виробники побутової, медичної, автомобільної та промислової електроніки, що використовують друковані плати як основу своїх пристроїв. Автоматичне виявлення дефектів забезпечить зниження ризику гарантійних випадків, зменшення рекламацій і підвищення репутації бренду.

4. Системні інтегратори рішень з автоматизації виробництва.

Компанії, які займаються впровадженням систем автоматизації для третіх осіб, можуть включати запропоновану систему як частину комплексного рішення для клієнта. Це відкриває додаткові канали дистрибуції стартап-продукту.

5. Технопарки, індустріальні хаби та інкубатори високих технологій.

Ці структури підтримують іноваційні проєкти, тому можуть бути зацікавлені у використанні системи як пілотного рішення для демонстрації або навчання або ж підтримки комерціалізації.

6. Вищі навчальні заклади та науково-дослідні лабораторії.

Заклади освіти та наукові установи, що працюють у напрямку комп’ютерного зору, електроніки, машинного навчання. Вони можуть використовувати систему як навчальний або дослідницький інструмент у рамках проєктів, лабораторій, дипломних робіт або грантів.

7. Дистриб’ютори та реселери програмного забезпечення для промисловості.

Ці компанії можуть виступати посередниками між стартапом і кінцевими замовниками. Вони зацікавлені в отриманні якісного продукту для перепродажу з додатковими сервісами: технічна підтримка, навчання персоналу.

8. Підприємства, які прагнуть цифрової трансформації.

Багато підприємств сьогодні прагнуть перейти від аналогових процесів до цифрових рішень. Запропонована система є ідеальним прикладом впровадження елементів Industry 4.0 – цифрової трансформації виробництва із використанням штучного інтелекту.

Географічна орієнтація:

- Україна як основний початковий ринок: існує потреба у локалізованих рішеннях українською мовою, з підтримкою вітчизняних підприємств;
- Центральна та Східна Європа як потенційний ринок для експорту, особливо з урахуванням швидкої цифровізації виробництва в Польщі, Чехії, Словаччині;
- Азійські виробники при локалізації та адаптації рішення під великі промислові масштаби.

Запропонований продукт орієнтований як на великі виробничі підприємства, так і на середній бізнес, що шукає ефективні шляхи автоматизації контролю якості. Завдяки широкій сфері застосування, масштабованості та адаптивності цільова аудиторія стартапу є значною та стабільною, що створює перспективу для довгострокового розвитку та успішної комерціалізації проєкту.

5.5. Бізнес-модель

Бізнес-модель стартап-проєкту визначає, яким чином компанія планує створювати цінність для клієнтів, доставляти її та отримувати прибуток. У рамках проєкту розробки системи автоматичної класифікації дефектів на виробничих лініях друкованих плат запропонована гібридна бізнес-модель, яка поєднує класичні підходи до продажу програмного забезпечення та сучасні сервіси з регулярною підпискою.

Основні джерела доходу. Ліцензійний продаж програмного забезпечення передбачає продаж постійної ліцензії на встановлення програмного забезпечення на обладнання клієнта. Такий підхід найбільш підходить для великих підприємств, які не бажають передавати свої дані до хмари або мають жорсткі вимоги до безпеки.

- одноразова вартість ліцензії (за кількістю робочих місць або виробничих ліній);
- можливість технічного обслуговування та оновлень за додаткову плату.

Програмне забезпечення надається у вигляді хмарного сервісу. Клієнти отримують доступ до системи через інтернет за підпискою:

- щомісячна або щорічна оплата за доступ до сервісу;
- масштабування функціоналу в залежності від тарифного плану;
- підтримка API для інтеграції в зовнішні системи.

Деякі компанії можуть мати особливі вимоги щодо формату входу/ виходу, зовнішнього інтерфейсу, розмітки, збереження результатів. У таких випадках передбачається розробка модулів під конкретного замовника за індивідуальною угодою.

Після придбання продукту клієнт може укласти договір на обслуговування, що включає:

- регулярне оновлення моделі;
- моніторинг роботи;
- навчання персоналу;
- консультації та доопрацювання.

Канали збуту. Прямі продажі:

- спілкування з представниками підприємств через участь у галузевих виставках, форумах, конференціях;
- прямі перемовини з технологами, IT-відділами та відділами контролю якості підприємств.

Онлайн-маркетинг:

- власний сайт із демонстраційною версією продукту;

- цільова реклама в соціальних мережах;
- email-маркетинг на базу виробничих підприємств.

Партнерські програми:

- співпраця з інтеграторами автоматизованих систем;
- дистриб’ютори рішень у сфері промислової автоматизації;
- освітні партнери, які включатимуть систему в навчальні програми.

Основні витрати в межах реалізації бізнес-моделі:

- розробка програмного забезпечення (зарплата розробників, аналітиків, дизайнерів);
- інфраструктура (сервери, хмарні сервіси, технічне обладнання для тестування);
- маркетинг та просування;
- юридичні послуги (ліцензування, захист інтелектуальної власності);
- технічна підтримка та навчання клієнтів.

Після успішного старту на ринку України передбачається:

- розширення на інші галузі: медицина (аналіз медичних зображень), агропромисловість (виявлення дефектів плодів), логістика (контроль упаковки);
- вихід на міжнародні ринки: Польща, Чехія, Німеччина, Індія, країни з активною електронною промисловістю та високим попитом на автоматизацію;
- розробка мобільної або десктопної версії з підтримкою IoT для зчитування даних з камер безпосередньо на виробництві.

Бізнес-модель стартапу забезпечує не лише стабільне джерело доходів, а й дозволяє швидко масштабуватись, адаптуючи продукт під потреби ринку. Гнучкість у виборі формату роботи, відкритість до індивідуальних рішень та доступність інтерфейсу є ключовими факторами для успішного просування продукту на ринку високих технологій у промисловості.

5.6. Конкурентні переваги

Для успішного виходу на ринок стартап-проєкт повинен мати чітко визначені конкурентні переваги, які забезпечать йому стабільне положення серед аналогічних рішень. Запропонована система класифікації дефектів на виробничій лінії за допомогою згорткових нейронних мереж має ряд переваг, що ґрунтуються на інноваційності, технічній ефективності, гнучкості та доступності рішення.

1. Висока точність і надійність системи. Завдяки використанню згорткових нейронних мереж (CNN) система демонструє точність класифікації понад 95 %, що суттєво перевищує показники традиційних систем візуального контролю, особливо у складних виробничих умовах (погане освітлення, неоднорідний фон, дрібні дефекти). Модель здатна розпізнавати тип дефекту, а не лише факт його наявності, що дає змогу оперативно реагувати на проблему на виробництві.

2. Робота в режимі реального часу. Система обробляє зображення миттєво, що дозволяє її застосовувати безпосередньо на виробничій лінії без необхідності зупиняти або уповільнювати процес. Завдяки оптимізованій обчислювальній архітектурі модель може бути розгорнута як на потужному сервері, так і на звичайному ПК.

3. Гнучка архітектура та кастомізація. Програмне забезпечення побудоване модульно, що дає змогу адаптувати його до специфічних потреб кожного клієнта – змінювати типи дефектів, налаштовувати алгоритми виявлення, інтегрувати систему з існуючим програмно-апаратним середовищем. Це дозволяє легко масштабувати продукт та забезпечувати його довгострокову підтримку.

4. Підтримка української мови та локалізація. Одна з важливих конкурентних переваг на локальному ринку – підтримка української мови, включаючи інтерфейс, документацію, технічну підтримку. Це забезпечує кращу зручність для користувачів і підвищує лояльність з боку національних підприємств.

5. Низький поріг входу та доступність. На відміну від більшості іноземних аналогів, які вимагають дорогого обладнання або великого обсягу попередніх налаштувань, дана система має невисокі апаратні вимоги та може працювати на

середньостатистичному комп’ютері. Це дозволяє впроваджувати її навіть на невеликих підприємствах без значних інвестицій.

6. Можливість донавчання та самопокращення. Завдяки використанню активного навчання система здатна вдосконалювати свою точність у процесі використання. Користувачі можуть додавати нові приклади зображень для тренування моделі, що дозволяє адаптувати її під змінні виробничі умови без повної перебудови архітектури.

7. Інтуїтивно зрозумілий інтерфейс. Розроблений графічний інтерфейс користувача має просту навігацію, зручну візуалізацію результатів класифікації, звіти та журнал подій. Це дозволяє працювати із системою навіть некваліфікованому персоналу без спеціальної підготовки.

8. Інтеграція із зовнішніми системами. Система підтримує API для обміну даними з іншими цифровими рішеннями (аналітичні платформи, бази даних,), що робить її відкритою до інтеграції в цифрові екосистеми підприємств згідно з концепцією Industry 4.0.

9. Наявність демо-версії та безкоштовного тестування. Перед впровадженням клієнтам надається можливість безкоштовного тестування прототипу системи на власному наборі зображень. Це підвищує довіру та мінімізує ризики для підприємства при ухваленні рішення про купівлю.

Таблиця 5.2 – Порівняльна таблиця конкурентних характеристик

Критерій	**Запропонована система**	**Типова традиційна система**	**Зарубіжний аналог**
Точність класифікації	>95 %	60-80 %	85-92 %
Підтримка самонавчання	Так	Ні	Частково
Вартість впровадження	Середня	Висока	Висока

Локалізація та підтримка	Українська, англійська	Часто відсутня	Переважно англійська
Підтримка кастомізації	Повна	Обмежена	Частково
Робота в реальному часі	Так	Ні	Частково
Інтеграція з іншими системами	Так (API)	Обмежена	Частково

Усі зазначені переваги забезпечують унікальне позиціонування стартапу на ринку систем контролю якості. Завдяки іноваційності, доступності, адаптивності та локалізації запропонований продукт здатен успішно конкурувати як із вітчизняними, так і з іноземними рішеннями, пропонуючи клієнтам оптимальне співвідношення ціни, якості та функціональності.

5.7. План впровадження

Впровадження стартап-проєкту потребує чіткого та поетапного плану, що дозволяє поступово пройти всі стадії розвитку, від створення прототипу до виходу на ринок і масштабування бізнесу. План реалізації охоплює як технічні, так і організаційні аспекти з урахуванням ресурсних можливостей команди та ринкової ситуації.

Процес впровадження розподілено на кілька етапів.

Етап 1. Дослідження та розробка MVP. MVP (Minimum Viable Product) - це мінімально життєздатний продукт, що містить основну функціональність системи для демонстрації її працездатності потенційним клієнтам. Завдання:

- формування технічного завдання на основі дослідження цільової аудиторії;
- підбір набору зображень для навчання нейронної мережі (dataset з дефектами друкованих плат);
- розробка базової моделі згорткової нейронної мережі;
- побудова інтерфейсу користувача;
- внутрішнє тестування системи на демонстраційних даних;

- підготовка документації до MVP.

Результат: готовий прототип, який здатний класифікувати основні види дефектів із точністю понад 90 %, що може бути представлений для тестування клієнтам або інвесторам.

Етап 2. Пілотне впровадження та перевірка гіпотез. На цьому етапі MVP тестується у реальних умовах виробництва разом із партнерами або потенційними замовниками. Завдання:

- узгодження умов з пілотними підприємствами;
- встановлення системи на виробничій лінії;
- збір зворотного зв'язку від технологів та операторів;
- аналіз ефективності системи: швидкість обробки, точність, стабільність;
- доопрацювання моделі на основі реальних помилок.

Результат: підтвердження працездатності системи в реальному виробничому процесі, розуміння основних вузьких місць для подальшого вдосконалення.

Етап 3. Завершення розробки та підготовка до запуску. На основі пілотного тестування здійснюється повноцінна розробка комерційної версії продукту. Завдання:

- оптимізація архітектури системи для стабільної роботи;
- створення функціоналу адміністрування, звітності та збереження історії дефектів;
- реалізація можливості донавчання моделі на основі нових даних;
- тестування масштабування моделі на більших наборах даних;
- розгортання документації, FAQ, навчальних відео;
- підготовка до маркетингової кампанії.

Результат: готовий продукт, готовий до виходу на ринок у вигляді ліцензованого програмного забезпечення.

Етап 4. Вихід на ринок і перші продажі. Цей етап передбачає запуск комерційної діяльності стартапу, активне залучення клієнтів та укладання перших контрактів. Завдання:

- запуск офіційного веб-сайту із презентаційними матеріалами та демонстрацією продукту;
- проведення рекламної кампанії;
- початок прямих переговорів з підприємствами-виробниками;
- укладання перших угод на поставку або підписку на продукт;
- збір зворотного зв’язку від перших клієнтів;
- початок формування служби технічної підтримки.

Результат: перші активні користувачі продукту, початковий грошовий потік, підтвердження ринкової зацікавленості в продукті.

Етап 5. Масштабування та розвиток. Після підтвердження життєздатності продукту та наявності ринку фокус переходить на масштабування бізнесу. Завдання:

- розширення штату (відділ продажів, маркетингу, підтримки);
- розробка додаткових модулів (модуль прогнозування відмов, візуальна аналітика, підтримка нових типів виробництва);
- вихід на суміжні ринки;
- локалізація продукту іншими мовами (англійська, польська);
- пошук зовнішнього фінансування (гранти, інвестори);
- партнерство з системними інтеграторами та дистриб’юторами.

Результат: сформована бізнес-структура, стабільний дохід, активна база клієнтів, присутність на декількох ринках.

Запропонований план впровадження дає змогу поетапно вивести продукт на ринок, починаючи з технічного прототипу й закінчуючи масштабованим комерційним рішенням. Поєднання технічного вдосконалення, активної взаємодії з клієнтами, адаптивності до ринку та стратегічного розвитку забезпечує високу ймовірність успіху стартапу у сфері промислової автоматизації.

Таблиця 5.3 – Орієнтовний бюджет проекту

Стаття витрат	**Орієтовна сума (грн)**
Розробка програмного забезпечення	180 000
Тестування та налагодження	40 000
Хостинг та хмарні ресурси	30 000
Маркетинг і реклама	50 000
Витрати на обладнання	60 000
Разом	360 000

ВИСНОВКИ ДО РОЗДІЛУ

В цьому розділі представлено підхід до створення стартапу у сфері автоматизації контролю якості виробничих процесів. Визначено основні етапи розвитку проекту, що забезпечують поетапний перехід від ідеї до комерційного запуску. Розроблено план заходів із прототипування, тестування та вдосконалення системи автоматичного виявлення дефектів друкованих плат. Визначено цілі та задачі, що сприяють швидкому виходу на ринок та отриманню перших клієнтів. Обґрунтовано важливість підготовки маркетингових матеріалів і проведення рекламної кампанії для популяризації продукту. Визначено ключові критерії успішної реалізації проекту, стабільність роботи системи та зворотній зв’язок від користувачів. Оцінено ризики, які пов’язані з технічними та комерційними аспектами стартапу і запропоновано стратегії їх мінімізації. Подано аналіз необхідної інфраструктури для впровадження системи на виробничих лініях. Відзначено роль командної роботи і співпраці з потенційними партнерами для швидшого комерціалізування рішення. Вказано на важливість адаптації системи під індивідуальні потреби клієнтів через модульну архітектуру. Розглянуто питання фінансування, залучення інвестицій та ресурсного забезпечення розвитку стартапу. Обґрунтовано необхідність постійного покращення моделі та системи на основі зворотного зв’язку. Визначено стратегії масштабування проекту у разі його успішної реалізації з урахуванням розширення функціональності. Підкреслено важливість документації, навчальних матеріалів і підтримки користувачів для довгострокового успіху.

ВИСНОВКИ

Розроблено інтелектуальну систему класифікації дефектів на виробничій лінії друкованих плат за допомогою згорткових нейронних мереж. Використано набір даних зображень, який був попередньо оброблений для підготовки до тренування моделі. Створена модель показала непогані результати в класифікації дефектів на основі навчання на різноманітних зображеннях. Модель використовує сучасні підходи згорткових нейронних мереж, що дозволяє ефективно виявляти навіть незначні дефекти на друкованих платах. Оцінка точності моделі на валідаційних даних дозволила переконатися в її високій ефективності та здатності до адаптації до нових зображень. Важливим досягненням є реалізація процесу класифікації дефектів у реальному часі, що може бути застосовано в автоматизованих виробничих процесах. Програмний код може бути адаптований під різні типи дефектів та вдосконалений для інших виробничих ліній. За допомогою цієї системи можна знизити кількість дефектних виробів та підвищити ефективність виробництва. У майбутньому доцільно вдосконалювати модель шляхом розширення наборів даних і використанням глибших архітектур нейронних мереж. Розроблена система може стати важливим інструментом для покращення якості продукції та автоматизації контролю якості на виробничих лініях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cucchiella F., D'Adamo I., Rosa P. Terzi S. Automotive printed circuit boards recycling: An economic analysis. J. Clean. Prod. 2016, 121, pp. 130-141.

2. Malge P., Nadaf R. PCB defect detection, classification and localization using mathematical morphology and image processing tools. International Journal of Computer Applications. 2014, 87, pp. 40-45.

3. Krizhevsky A., Sutskever I., Hinton G. ImageNet classification with deep convolutional neural networks. Communications of the association for computing machinery, vol. 60, №2, 2012. pp. 84-90.

4. Hu B., Wang J. Detection of PCB surface defects with improved faster-RCNN and feature pyramid network. Institute of electrical and electronics engineers access, vol. 8, 2020. pp. 108335-108345.

5. Anoop K., Sarath N., Kuma S. A review of PCB defect detection using image processing. International Journal of Engineering Innovation and Technology, vol. 4, 2015. pp. 188-192.

6. Aggarwal N., Deshwal M., Samant P. A survey on automatic printed circuit board defect detection techniques. 2nd International conference on advanced computing, innovative technology and engineering (ICACITE). 2022, pp. 853-856.

7. Pal A., Chauhan S., Bhardwaj S. Detection of bare PCB defects by image subtraction method using machine vision. Proceedings of the World Congress on Engineering., vol. 2, 2011. pp. 6-8.

8. Sundaraj K. PCB inspection for missing or misaligned components using background subtraction. World scientific and engineering academy and society transactions on information science and applications., vol. 6, 2009. pp. 778-787.

9. Ma J. Defect detection and recognition of bare PCB based on computer vision. 36th Chinese control conference (CCC), 2017. pp. 11023-11028.

10. Melnyk R., Tushnytskyy R. Detection of defects in printed circuit boards by clustering the etalon and defected samples. IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), 2020. pp. 961-964.

11. Haq M., Jilani A., Prabu P. Algorithmic scheme for concurrent detection and classification of printed circuit board defects. Computers, materials & continua, vol. 71, №1, 2022. pp. 355-367.

12. Annaby M., Fouda Y., Rushdi M. Improved normalized cross-correlation for defect detection in printed-circuit boards. IEEE Transactions on semiconductor manufacturing, vol. 32, №2, 2019. pp. 199-211.

13. Hassanin A., Abd El-Samie F., El Banby G. A real-time approach for automatic defect detection from PCBs based on surf features and morphological operations. Multimedia tools and applications., vol. 78, 2019. pp. 1-21.

14. Crispin A., Rankov V. Automated inspection of PCB components using a genetic algorithm template-matching approach. International journal of advanced manufacturing technology, vol. 35, №3, 2007. pp. 293-300.

ДОДАТКИ

Додаток А. Класифікація дефектів друкованих плат

defect.ipynb

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
from collections import Counter
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import albumentations as A
import tensorflow as tf

train_img_dir = '/kaggle/input/pcb-defect-dataset/pcb-defect-dataset/train/images'
val_img_dir = '/kaggle/input/pcb-defect-dataset/pcb-defect-dataset/val/images'
test_img_dir = '/kaggle/input/pcb-defect-dataset/pcb-defect-dataset/test/images'

augment = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.5),
    A.Rotate(limit=20, p=0.5),
    A.RandomGamma(p=0.3),
    A.MotionBlur(p=0.3),
    A.CLAHE(p=0.2),
    A.GaussNoise(p=0.3),
    A.HueSaturationValue(p=0.3),
    A.GridDistortion(p=0.2)
])
IMG_SIZE = 128
```

```
def load_data_from_folder(folder):
    image_paths = glob(os.path.join(folder, '*.jpg'))
    images = []
    labels = []
    for path in image_paths:
        img = cv2.imread(path)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) / 255.0
        label = path.split("_")[-2]
        images.append(img)
        labels.append(label)
    return np.array(images), np.array(labels)

X_train, y_train = load_data_from_folder(train_img_dir)
X_val, y_val = load_data_from_folder(val_img_dir)
X_test, y_test = load_data_from_folder(test_img_dir)

plt.figure(figsize=(10, 4))
sns.countplot(x=y_train)
plt.title("Training Set Class Distribution (Original Labels)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

le = LabelEncoder()
y_train_enc = to_categorical(le.fit_transform(y_train))
y_val_enc = to_categorical(le.transform(y_val))
y_test_enc = to_categorical(le.transform(y_test))

def augment_minority_classes(X, y, augment, label_encoder):
    print("Applying targeted augmentation to balance classes...")
    y_str = y
    counts = Counter(y_str)
    max_count = max(counts.values())

    new_X, new_y = list(X), list(y_str)
    for label in counts:
        imgs = [x for x, l in zip(X, y_str) if l == label]
        needed = max_count - counts[label]
        for _ in range(needed):
            img = imgs[np.random.randint(len(imgs))]
            aug_img = augment(image=(img * 255).astype('uint8'))['image'] / 255.0
            new_X.append(aug_img)
```

```
            new_y.append(label)

    new_X = np.array(new_X)
    new_y_encoded = to_categorical(label_encoder.transform(new_y))

    print(f"Balanced dataset: {dict(Counter(new_y))}")
    return new_X, new_y_encoded, new_y

X_train_balanced, y_train_balanced, y_train_balanced_raw = augment_minority_classes(
    X_train, y_train, augment, le
)

plt.figure(figsize=(10, 4))
sns.countplot(x=y_train_balanced_raw)
plt.title("Training Set Class Distribution After Balancing (Original Labels)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

def focal_loss(gamma=2.0, alpha=0.25):
    def loss_fn(y_true, y_pred):
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.keras.backend.clip(y_pred, epsilon, 1.0 - epsilon)
        cross_entropy = -y_true * tf.keras.backend.log(y_pred)
        weight = alpha * y_true * tf.pow(1 - y_pred, gamma)
        return tf.reduce_mean(tf.reduce_sum(weight * cross_entropy, axis=1))
    return loss_fn

def build_cnn(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        BatchNormalization(),
        MaxPooling2D(),
        Conv2D(64, (3, 3), activation='relu'),
        BatchNormalization(),
        MaxPooling2D(),

        Conv2D(64, (3, 3), activation='relu'),
        BatchNormalization(),
        MaxPooling2D(),

        Flatten(),
        Dense(256, activation='relu'),
```

```
        Dropout(0.4),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer=Adam(learning_rate=1e-5),
                  loss=focal_loss(),
                  metrics=['accuracy'])
    return model

model = build_cnn((IMG_SIZE, IMG_SIZE, 3), len(le.classes_))
model.summary()

early_stop = EarlyStopping(patience=5, restore_best_weights=True)

model.fit(
    X_train_balanced,
    y_train_balanced,
    validation_data=(X_val, y_val_enc),
    epochs=20,
    batch_size=32,
    callbacks=[early_stop]
)

preds = model.predict(X_test)
y_pred = np.argmax(preds, axis=1)
y_true = np.argmax(y_test_enc, axis=1)

print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=le.classes_))

cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt='d',
            xticklabels=le.classes_, yticklabels=le.classes_, cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

Epoch 1/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 729ms/step - accuracy: 0.3620 - loss:
0.4231 - val_accuracy: 0.4400 - val_loss: 0.1998
Epoch 2/20
```

```
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247s 730ms/step - accuracy: 0.6060 - loss:
0.1378 - val_accuracy: 0.6895 - val_loss: 0.0872
Epoch 3/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 260s 768ms/step - accuracy: 0.6403 - loss:
0.1095 - val_accuracy: 0.7017 - val_loss: 0.0770
Epoch 4/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247s 730ms/step - accuracy: 0.6578 - loss:
0.0965 - val_accuracy: 0.6951 - val_loss: 0.0749
Epoch 5/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 264s 782ms/step - accuracy: 0.6578 - loss:
0.0901 - val_accuracy: 0.6904 - val_loss: 0.0739
Epoch 6/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 746ms/step - accuracy: 0.6749 - loss:
0.0861 - val_accuracy: 0.6970 - val_loss: 0.0723
Epoch 7/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251s 743ms/step - accuracy: 0.6861 - loss:
0.0806 - val_accuracy: 0.7017 - val_loss: 0.0706
Epoch 8/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 266s 786ms/step - accuracy: 0.6910 - loss:
0.0786 - val_accuracy: 0.7008 - val_loss: 0.0704
Epoch 9/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247s 730ms/step - accuracy: 0.6915 - loss:
0.0756 - val_accuracy: 0.7008 - val_loss: 0.0696
Epoch 10/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 253s 749ms/step - accuracy: 0.6942 - loss:
0.0717 - val_accuracy: 0.7054 - val_loss: 0.0690
Epoch 11/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240s 710ms/step - accuracy: 0.6964 - loss:
0.0707 - val_accuracy: 0.7111 - val_loss: 0.0685
Epoch 12/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 238s 704ms/step - accuracy: 0.7077 - loss:
0.0686 - val_accuracy: 0.7045 - val_loss: 0.0682
Epoch 13/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 744ms/step - accuracy: 0.7154 - loss:
0.0670 - val_accuracy: 0.7139 - val_loss: 0.0677
Epoch 14/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239s 706ms/step - accuracy: 0.7140 - loss:
0.0675 - val_accuracy: 0.7111 - val_loss: 0.0675
Epoch 15/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 252s 747ms/step - accuracy: 0.7267 - loss:
0.0642 - val_accuracy: 0.7008 - val_loss: 0.0686
```

```
Epoch 16/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239s 707ms/step - accuracy: 0.7310 - loss:
0.0633 - val_accuracy: 0.7129 - val_loss: 0.0677
Epoch 17/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239s 706ms/step - accuracy: 0.7345 - loss:
0.0628 - val_accuracy: 0.7120 - val_loss: 0.0679
Epoch 18/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251s 743ms/step - accuracy: 0.7277 - loss:
0.0618 - val_accuracy: 0.7083 - val_loss: 0.0673
Epoch 19/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 241s 714ms/step - accuracy: 0.7409 - loss:
0.0578 - val_accuracy: 0.7064 - val_loss: 0.0673
Epoch 20/20
338/338 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 255s 755ms/step - accuracy: 0.7408 - loss:
0.0597 - val_accuracy: 0.7017 - val_loss: 0.0681
34/34 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 132ms/step

Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.76      0.85       270
           2       0.77      0.70      0.73       253
           3       0.65      0.68      0.67       236
           4       0.63      0.56      0.59       197
           5       0.53      0.96      0.68       112

    accuracy                           0.71      1068
   macro avg       0.71      0.73      0.70      1068
weighted avg       0.74      0.71      0.72      1068

from sklearn.metrics import precision_score, f1_score, roc_auc_score,
average_precision_score
per_class_precision = precision_score(y_true, y_pred, average=None)
print("\nPer-class Precision:")
for i, class_name in enumerate(le.classes_):
    print(f"{class_name}: {per_class_precision[i]:.4f}")

per_class_f1 = f1_score(y_true, y_pred, average=None)
print("\nPer-class F1-score:")
for i, class_name in enumerate(le.classes_):
    print(f"{class_name}: {per_class_f1[i]:.4f}")
```

```
try:
    auc_score = roc_auc_score(y_test_enc, preds, multi_class='ovr')
    print(f"\nOverall AUC Score (OvR): {auc_score:.4f}")
except ValueError as e:
    print(f"\nAUC computation failed: {e}")

map_score = average_precision_score(y_test_enc, preds, average='macro')
print(f"\nMean Average Precision (mAP): {map_score:.4f}")

Per-class Precision:
1: 0.9624
2: 0.7672
3: 0.6492
4: 0.6322
5: 0.5323

Per-class F1-score:
1: 0.8489
2: 0.7340
3: 0.6653
4: 0.5930
5: 0.6837

Overall AUC Score (OvR): 0.9350

Mean Average Precision (mAP): 0.7592
```

Додаток Б. Графічний інтерфейс інформаційної системи

defect2.py

```
import streamlit as st
import numpy as np
import pandas as pd
from PIL import Image

class_names = [
    "Дірка",
    "Пошкодження доріжки",
    "Зайвий мідний шар",
    "Порушення ширини доріжки",
    "Непаяний контакт",
    "Пошкодження краю плати",
    "Без дефекту"
]

def preprocess_image(image):
    img = image.resize((128, 128))
    return np.array(img) / 255.0

def main():
    st.set_page_config(page_title="Класифікація дефектів друкованих плат",
layout="centered")
    st.title("Класифікація дефектів друкованих плат")

    uploaded_file = st.file_uploader("Завантажте зображення друкованої плати",
type=["jpg", "jpeg", "png"])

    if uploaded_file is not None:
        image = Image.open(uploaded_file).convert("RGB")
        st.image(image, caption="Завантажене зображення", use_column_width=True)

        st.subheader("Результати класифікації дефектів друкованої плати:")
        probs = simulate_prediction()

        df = pd.DataFrame({
            "Клас": class_names,
            "Ймовірність": [f"{p * 100:.2f}%" for p in probs]
        })
```

```
        df = df.sort_values(by="Ймовірність", ascending=False).reset_index(drop=True)
        st.table(df)

if __name__ == "__main__":
    main()
```