

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повна назва кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр
(ступінь вищої освіти)

на тему МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ДІАГНОСТУВАННЯ
ДРУКОВАНИХ ПЛАТ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ
METHODS AND SOFTWARE TOOLS FOR DIAGNOSTICS OF PRINTED
CIRCUIT BOARDS BASED ON NEURAL NETWORKS

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного
забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Хохлова В.С.
(прізвище та ініціали)

Керівник Субботін С.О.
(прізвище та ініціали)

Рецензент Гофман Є.О.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет Інститут інформатики та радіoeлектроніки,
факультет комп'ютерних наук і технологій
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Хохлової Варвари Сергіївни
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Методи та програмні засоби діагностування друкованих плат на основі нейронних мереж. Methods and Software Tools for Diagnostics of Printed Circuit Boards Based on Neural Networks
 керівник проєкту (роботи) Субботін Сергій Олександрович, д.т.н., професор,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301
2. Строк подання студентом проєкту (роботи) 15 грудня 2020 року
3. Вихідні дані до проєкту (роботи) база даних, яка складається із зображення чипів різних типів, зроблених на виробництві та розмічених у блакитно-червоні кольори, перелік можливих дефектів чипів, що зображені на фотографіях, кількість зображень кожного типу чипу.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Розробка програми. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Економіко-організаційна частина. 7. Охорона праці та безпека у надзвичайних ситуаціях. 8. Додатки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Субботін С.О., професор		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека у надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Калініна М.В., асистент		

7. Дата видачі завдання “4” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Тестування програми та експериментальне дослідження програмного забезпечення.	9 тижні	Розділ 5
7	Розробка економіко-організаційної частини.	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки у надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14-15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис)

Хохлова В.С.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Субботін С.О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
136 с., 40 табл., 52 рис., 2 дод., 43 джерела.

АНАЛІЗ ЗОБРАЖЕНЬ, АОІ, ДІАГНОСТУВАННЯ ДРУКОВАНИХ ПЛАТ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, PYTHON, TENSORFLOW.

Об'єкт дослідження – процес обробки та класифікації зображень.

Предмет дослідження – методи автоматизованої оптичної інспекції друкованих плат на виробництві.

Мета роботи – дослідити та програмно реалізувати методи автоматизованої оптичної інспекції задля підвищення швидкості та точності діагностування друкованих плат.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, автоматизована оптична інспекція, згорткові нейронні мережі, використання бібліотеки TensorFlow для роботи з нейронними мережами, gRPC фреймворку та графічної бібліотеки matplotlib, мова програмування Python 3.8, середовище розробки програм Jupyter Notebook, персональний комп'ютер з процесором Intel Core i3, оперативною пам'яттю 8 Гб, 23" монітором, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи Linux Ubuntu Mate 18.04.

Результат. Створено програмний застосунок, який діагностує друковані плати на етапі виробництва та сигналізує про дефекти виробництва.

Висновки. Розроблено програмне забезпечення для діагностування друкованих плат, яке дозволяє покращити якість та швидкість виробництва.

Галузь використання – радіоелектроніка та приладобудування.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 136 pages, 40 tables, 52 figures, 2 appendixes, 43 sources.

IMAGE ANALYSIS, AOI, PRINTED CIRCUIT BOARD DIAGNOSTICS, MACHINE LEARNING, NEURAL NETWORKS, PYTHON, TENSORFLOW.

Object of study is the process of image analysis and classification.

Subject of study are methods of automated optical inspection of printed circuit boards in production.

The purpose of the work is research and software implementation of automated optical inspection methods to increase the speed and accuracy of printed circuit board diagnostics.

Materials, methods, and tools: structured and object-oriented programming, automated optical inspection, convolutional neural networks, neural networks library TensorFlow, gRPC framework and matplotlib graphic library, Python 3.8 programming language, development environment Jupyter Notebook, personal computer with an Intel Core i3 processor, a 8 GB RAM, a 23'' monitor, a keyboard and a mouse, that runs under the control of the Linux Ubuntu Mate 18.04 operating system.

Results. Application that diagnose printed circuit boards in production and alert about production defects is implemented.

Conclusions. A software system for the printed circuit board diagnostics that improve production quality and speed was developed.

Field of use is radio electronics and instrument making.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження.....	10
1.1 Характеристика предметної області.....	10
1.2 Формальна постановка задачі.....	15
1.3 Дослідження існуючих методів вирішення проблеми.....	15
1.4 Висновки за розділом.....	18
2 Матеріали і методи.....	20
2.1 Штучні нейронні мережі.....	20
2.2 Аналіз об'єктів на зображеннях.....	28
2.3 Розроблений метод аналізу об'єктів на зображеннях.....	31
2.4 Висновки за розділом.....	35
3 Опис програми.....	36
3.1 Логічна структура програми.....	36
3.2 Програмні засоби та мови програмування.....	38
3.3 Висновки за розділом.....	45
4 Розробка програми.....	46
4.1 Опис структури файлів.....	46
4.2 Опис допоміжних бібліотек, модулів та функцій.....	48
4.3 Опис системних модулів.....	49
4.4 Опис інтерфейсу gRPC.....	51
4.5 Опис даних.....	65
4.6 Висновки за розділом.....	66
5 Експлуатація, тестування та експериментальне дослідження програми.....	67
5.1 Опис експлуатації програми.....	67
5.2 Опис тестових даних.....	71
5.3 Експериментальні дослідження.....	74
5.4 Подальші покращення системи.....	80

5.5 Висновки за розділом.....	80
6 Економіко-організаційна частина.....	81
6.1 Планування розробки програмного продукту.....	81
6.2 Визначення витрат на розробку програми.....	84
6.3 Оцінка економічної ефективності.....	91
7 Охорона праці та безпека у надзвичайних ситуаціях.....	92
7.1 Аналіз потенційних небезпек.....	92
7.2 Заходи з забезпечення безпеки праці.....	93
7.3 Заходи з забезпечення виробничої санітарії та гігієни праці.....	99
7.4 Заходи з пожежної безпеки.....	101
7.5 Заходи з забезпечення безпеки у надзвичайних ситуаціях.....	104
7.6 Висновки з розділу.....	107
Висновки.....	108
Перелік джерел посилання.....	109
ДОДАТОК А Текст програми.....	113
ДОДАТОК Б Слайди презентації.....	127

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

АОІ	– автоматизована оптична інспекція;
БП	– багатошаровий перцептрон;
ВВП	– виклик віддалених процедур;
ГК	– графічний інтерфейс користувача;
ГНМ	– глибинна нейронна мережа;
ЗНМ	– згорткова нейронна мережа;
ПК	– персональний комп'ютер;
ППІ	– прикладний програмний інтерфейс;
УУІД	– універсальний унікальний ідентифікатор.

ВСТУП

За останні кілька років сучасні друковані плати були серйозно вдосконалені, стали набагато складніші. Впровадження сучасних технологій поверхневого монтажу та подальше зменшення розміру плат робить їх більш компактними. Навіть порівняно середні плати мають тисячі паяних з'єднань, і саме тут виявляється більшість проблем.

Управління якістю друкованих плат для технологічних процесів забезпечує автоматизована оптична інспекція (AOI). Вона є показником високого рівня організації виробництва та застосовується для контролю пайки чипів, нанесення пасти, монтажу всіх електронних компонентів, якості збірки друкованих плат.

Об'єктом дослідження є процес обробки та класифікації зображень.

Предметом дослідження є методи AOI друкованих плат на виробництві.

Мета роботи – дослідження та програмна реалізація методів AOI задля підвищення швидкості та точності діагностування друкованих плат.

Завдання дослідження:

- проаналізувати предметну область та існуючі методи вирішення завдання для визначення ступеня дослідження області та постановки завдань роботи;
- обрати методи обробки зображень та машинного навчання, мову програмування для програмної реалізації завдання;
- розробити алгоритм діагностування плат для формування структури розроблюваної програми;
- розробити програму для діагностування плат;
- провести експериментальне дослідження роботи програми для вирішення поставленої задачі;
- розрахувати економічні показники та охорону праці.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Характеристика предметної області

1.1.1 Введення в АОІ

АОІ – це автоматизована візуальна перевірка виготовлення друкованої плати, при якій камера автономно сканує виготовлений пристрій для діагностування катастрофічних несправностей (наприклад, відсутність компонента) та дефектів якості (наприклад, розмір або перекис форми компонента). АОІ дозволяє швидко і точно перевіряти електронні вузли і, зокрема, друковані плати, щоб гарантувати, що якість продукції, що виходить з виробничої лінії, висока, а елементи побудовані правильно і без виробничих несправностей [1].

Така перевірка зазвичай використовується у виробничому процесі, оскільки є безконтактним методом тестування. Вона впроваджується на багатьох етапах процесу виробництва, включаючи перевірку початкових плат, перевірку паяльної пасти, кріплення компонентів та інших етапах [1]. Історично системи АОІ прийнято використовувати на етапі "поствиробництва". Головним чином тому, що системи АОІ на більш пізньому етапі можуть перевіряти плати на наявність більшості типів дефектів (розміщення компонентів, пайки, відсутні припої тощо) в одному місці одразу. Після перевірки пошкоджені дошки переробляються, а інші дошки направляються на наступну стадію процесу [2].

З роками складність плат зростає. Це означає, що ручний огляд сьогодні не є прийнятним варіантом. Навіть коли це був прийнятний підхід, було зрозуміло, що він не був особливо ефективним, оскільки інспектори втомлювалися та могли легко пропустити погану або неправильну конструкцію. Оскільки зараз ринок вимагає продукцію великих обсягів та високої якості, потрібні дуже надійні та швидкі методи, щоб забезпечити ці вимоги. АОІ є важливим інструментом в стратегії тестування електроніки, що

забезпечує мінімально низькі витрати шляхом виявлення несправностей на початку виробничої лінії.

Якщо розмістити АОІ на виробничій лінії відразу після процесу припою, це допоможе виявити дефект як можна швидше. Це має ряд переваг, оскільки чим пізніше в процесі виробництва буде виявлена несправність, тим дорожче буде коштувати її усунення. Крім того, якщо діагностувати дефект на ранній стадії, то можна зупинити процес виробництва, доки ще не буду надрукована багато екземплярів плат [3].

Системи АОІ здатні виявляти різноманітні дефекти, такі як вузлики, подряпини та плями, а також більш звичні розмірні дефекти, такі як розімкнуті ланцюги, витончення припою. Вони також можуть виявити неправильні, відсутні та неправильно розміщені компоненти. Таким чином вони можуть виконувати всі візуальні перевірки, які раніше виконували оператори вручну, набагато швидше і точніше [3].

1.1.2 Введення в машинне навчання

Машинне навчання – це клас алгоритмів, які мають здатність навчатися з даних, відшукуючи в них загальні залежності та закономірності. Вони поступово покращують продуктивність у певній задачі та не програмуються явно.

Машинне навчання досліджує вивчення та побудову алгоритмів, які можуть навчатися й робити передбачення з даних [4]. Його застосовують в ряді обчислювальних задач, в яких розробка та програмування явних алгоритмів з доброю продуктивністю є складною або нездійсненною. До прикладів застосувань належать фільтрування електронної пошти, виявлення мережних зловмисників, які хочуть влаштувати витоку даних, оптичне розпізнавання символів, навчання ранжуванню та комп'ютерний зір [5].

Машинне навчання тісно пов'язане з обчислювальною статистикою, яка також зосереджується на прогнозуванні шляхом застосування

комп'ютерів. Воно має тісні зв'язки з математичною оптимізацією, яка забезпечує цю галузь методами, теорією та прикладними областями. Машинне навчання іноді об'єднують з добуванням даних, де друга підгалузь фокусується більше на розвідувальному аналізі даних, і є відомою як навчання без вчителя [6]. Машинне навчання також може бути спонтанним, і застосовуваним для навчання та встановлення базових характеристик поведінки різних суб'єктів, а потім застосовуваним для пошуку виразних аномалій.

В межах галузі аналізу даних машинне навчання є методом, який використовується для винаходження складних моделей та алгоритмів, які слугують прогнозуванню – в комерційному застосуванні це відоме як прогнозна аналітика. Ці аналітичні моделі дозволяють дослідникам, науковцям з даних, інженерам та аналітикам «виробляти надійні, повторювані рішення та результати» та розкривати «приховані розуміння» шляхом навчання з історичних співвідношень та тенденцій в даних [6].

Задачі машинного навчання, як правило, поділяють на дві широкі категорії, залежно від того, чи доступний системі, що навчається, навчальний «сигнал», або «зворотний зв'язок»:

а) навчання зі вчителем (кероване навчання): комп'ютеріві представляють приклади входів та їхніх бажаних виходів, задані «вчителем», і метою є навчання загального правила, яке відображає входи на виходи. В окремих випадках вхідний сигнал може бути доступним лише частково, або бути обмеженим особливим зворотним зв'язком:

1) напіваавтоматичне навчання: комп'ютеріві дають лише неповний тренувальний сигнал: тренувальний набір, в якому відсутні деякі (часто численні) цільові виходи;

2) активне навчання: комп'ютер може отримувати тренувальні мітки лише для обмеженого набору екземплярів, а також має оптимізувати свій вибір об'єктів для отримування міток (за інтерактивного застосування, вони можуть надаватися для мічення користувачеві);

3) навчання з підкріпленням: тренувальні дані (у вигляді винагород та покарань) надаються лише як зворотний зв'язок на дії програми в динамічному середовищі, як при керуванні автомобілем, або грі в гру з опонентом;

б) навчання без учителя (спонтанне навчання): алгоритмові навчання не дається міток, залишаючи його самому знаходити структуру в своєму вході, навчання без учителя може бути метою саме по собі (виявлення прихованих закономірностей у даних), або засобом досягнення мети (навчання ознак) [7].

1.1.3 Застосування та алгоритми машинного навчання

В залежності від бажаного виходу системи з машинним навчанням задачі, які вирішуються, можна поділити на такі типи [7]:

- класифікація;
- регресія;
- кластеризація;
- оцінка густини;
- зниження розмірності.

У класифікації входи поділяються на два або більше класів, і система-учень мусить породити модель, яка відносить небачені входи до одного або більше з цих класів. Це, як правило, намагаються розв'язувати керованим чином. Прикладом класифікації є фільтри спаму, в яких входами є повідомлення електронної пошти (або чогось іншого), а класами є «спам» та «не спам».

У регресії, також керованій задачі, входи є безперервними, а не дискретними [7].

У кластеризації набір входів повинно бути поділено на групи. На відміну від класифікації, групи не відомі заздалегідь, що зазвичай робить це завданням для спонтанного навчання.

Оцінка густини знаходить розподіл входів у деякому просторі.

Зниження розмірності спрощує входи шляхом відображення їх на простір меншої розмірності. Пов'язаною задачею є тематичне моделювання, в якому програмі надають перелік документів людською мовою, і ставлять задачу з'ясувати, які документи охоплюють подібні теми.

Існує безліч алгоритмів машинного навчання, що дозволяють вирішити певну задачу, однак не існує такого універсального алгоритму, який підходив би під всі випадки і вирішував абсолютно всі проблеми [7].

Існують наступні алгоритми машинного навчання:

- лінійна регресія;
- логістична регресія;
- дерева рішень;
- найвний Байєсовський класифікатор;
- метод k-найближчих сусідів;
- метод опорних векторів;
- випадковий ліс;
- бустинг;
- нейронні мережі.

Кожен алгоритм володіє як перевагами, так і недоліками, які можна використовувати в якості керівництва для вибору алгоритму, який найбільш підходить під ситуацію.

Зі всіх перелічених вище алгоритмів нейронні мережі мають найбільше переваг для вирішення задачі роботи. Робота з зображеннями – важлива сфера застосування технологій глибинного навчання. Глобально всі зображення зі всіх камер світу складають бібліотеку неструктурованих даних. Використавши нейронні мережі, машинне навчання і штучний інтелект, ці дані структурують і використовують для виконання різних завдань: побутових, соціальних, професійних і державних, зокрема, забезпечення безпеки. Нейронні мережі вимогливі до розміру і якості набору даних, на якому вона буде навчатися [7].

1.2 Формальна постановка задачі

Завдання системи діагностування друкованих плат полягає у тому, щоб виявити та класифікувати дефекти плат на етапі виробництва.

Вхідними даними для системи діагностування друкованих плат є:

- база даних, яка складається із зображення чипів різних типів, зроблених на виробництві та розмічених у блакитно-червоні кольори; кожне зображення без дефекту містить лише один чіп;
- перелік можливих дефектів чипів, що зображені на фотографіях;
- кількість зображень кожного типу чипу.

Вихідними даними для системи діагностування є:

- маркування зображень в залежності від наявності дефекту плати;
- маркування дефектних зображень в залежності від типу дефекту.

Для оцінювання якості результатів використовується аналіз матриці невідповідностей та кластеризація початкових зображень.

1.3 Дослідження існуючих методів вирішення проблеми

1.3.1 Виявлення дефектів друкованої плати без компонентів методом віднімання зображення за допомогою машинного зору

Розробники системи інспекції друкованих плат за допомогою машинного зору [8] пропонують застосовувати її на першому етапі виробництва, тобто після виготовлення друкованої плати без компонентів. Спочатку було порівняно стандартне зображення друкованої плати із зображенням готової друкованої плати, використовуючи простий алгоритм віднімання, який може виявити дефектні області. Типові дефекти, які можна виявити, – це розкриття, закриття, отвори тощо. Машинне бачення було реалізоване у системі та включає наступні процеси: автоматизований процес перевірки, контроль процесу, збір та індексація бази даних, моделювання системи та середовища [8].

Арифметична або логічна операція між зображеннями (віднімання) – це піксельне перетворення. Воно створює зображення, в якому кожен піксель отримує своє значення із значення пікселів з однаковими координатами в інших зображеннях. Для реалізації логіки віднімання зображень було використано Vision Assistant як інструмент розробки [8].

Vision Assistant – це інструмент для створення прототипів та тестування програм обробки зображень. У цій програмі можна створювати власні алгоритми за допомогою функції сценарію. Функція сценаріїв записує кожен крок алгоритму обробки. Після завершення алгоритму можна перевірити його на інших зображеннях, щоб переконатися, що він працює. Алгоритм записується у файл збірки, який є текстовим файлом ASCII, в якому перелічені функції обробки та відповідні параметри алгоритму, який був запрототипован у Vision Assistant [9]-[11].

Інспекція друкованої плати з використанням методу віднімання зображень виконується поетапно:

- завантаження еталонного зображення;
- буферизація еталонного зображення, щоб його можна було використовувати для операції віднімання;
- завантаження зображення, яке необхідно перевірити;
- пошук дефекту друкованої плати, віднімання двох зображень [11].

Проаналізований метод пропонує діагностувати дефекти плат за допомогою методу віднімання зображень [8].

1.3.2 Виявлення дефектів друкованої плати за допомогою обробки зображень

Автори роботи [12] використовують MATLAB як інструмент для виконання різних завдань з виявлення дефектів на друкованій платі. Проблема вирішується тут шляхом розподілу зображень на менші допоміжні зображення, пропонується вилучити частини блакитної затіненої області

зображень. Цей розподіл проводиться навіть для еталонного зображення, і обидва вони порівнюються для виявлення дефекту [12].

Під час обробки зображень автори зосередили своє дослідження на:

- куті нахилу друкованої плати;
- вертикальній відстані, на якій знімаються зображення;
- зображеннях з поганими умовами освітлення.

Обробка зображення проходить за наступним алгоритмом:

- завантаження зображення плати;
- перевірка рівня освітленості на зображенні;
- перевірка кута нахилу плати за допомогою трансформації Хафа;
- визначення меж плати за допомогою автоматичного обрізання зображення;
- порівняння зображення з еталоном [12].

Проаналізований метод пропонує діагностувати дефекти плат за допомогою розподілу зображень на менші допоміжні зображення та подальшого їх порівняння по пікселям [12].

1.3.3 Повноцінний додаток, який класифікує дефекти друкованих плат

У роботі [13] було розроблено класифікатор дефектів друкованих плат, який автоматизує завдання виявлення та класифікації дефектів.

Існує ряд проблем, які виявились при спробі автоматизувати виявлення та класифікацію дефектів. А саме:

- низький допуск на дефекти в одиницях;
- декілька типів дефектів, які потрібно врахувати;
- для автоматизованих рішень потрібна швидкість роботи (виявлення дефектів);
- навчальні дані для моделей нейронних мереж обмежені (як правило, вузькі, специфічні та невеликі);

- традиційні системи покладаються на значний досвід;
- методи, що використовуються іншими, часто приватні [13].

Був розроблений підхід, який вирішує ці проблеми у програмі, яка ефективно виявляє, витягує та передбачає типи дефектів у друкованих платах.

Було вирішено використовувати гібридний підхід для побудови класифікатора, який поєднує в собі еталонний метод та не-еталонний метод. В еталонному методі тестова плата порівнюється з еталонним зображенням шаблону, який не має дефектів, щоб аналітики могли виявити відмінності. У не-еталонному методі тестова плата перевіряється на відповідність принципам проєктування, не покладаючись на шаблонні зображення [13].

Еталонний метод є відносно простим у впровадженні, але має великі вимоги до об'єму бази даних. Не-еталонний метод не страждає від проблем із зберіганням даних, проте може обмежувати кількість виявлених дефектів.

Для розробки програми були обрані R та Python мови програмування. Яскраві частини зображення виділяються за допомогою визначення контурів об'єктів. Усі виявленні дефекти маркуються та записуються до JSON файлу. Марковані дані використовуються для навчання нейронної мережі. Для класифікації дефектів використовуються глибинне навчання та нейронні мережі [13].

Проаналізований метод пропонує діагностувати дефекти плат за допомогою порівняння зображень з еталонними та шаблонами [13].

1.4 Висновки за розділом

Аналіз предметної області, проведений у розділі 1, показав, що АОІ та машинне навчання широко застосовуються на виробництві друкованих плат та є дуже актуальними.

Аналіз існуючих рішень окреслив проблеми, які необхідно вирішити під час розробки програми:

- перевірка якості зображень з виробництва для виявлення їх особливостей та дефектів;
- обробка зображень для усунення виявлених дефектів (недостатньо світла, різний розмір зображень, розміщення більше ніж одного чипу на зображенні, хибна орієнтація чипу та інше);
- поділ зображень різних типів на навчальну та тестову вибірки;
- підбір методів машинного навчання для класифікації даних;
- класифікація зображень за наявністю дефекту та типом зображеного чипу;
- оцінка результатів класифікації.

Метою даної роботи є дослідження та програмна реалізація методів автоматизованої оптичної інспекції задля підвищення швидкості та точності діагностування друкованих плат.

2 МАТЕРІАЛИ І МЕТОДИ

Проведений аналіз теоретичних матеріалів та існуючих рішень показав, що для вирішення задачі діагностування друкованих плат використовуються наступні методи:

- порівняння контурів на зображеннях;
- порівняння колірних моделей HSI;
- порівняння зображень піксель за пікселем;
- машинне навчання;
- нейронні мережі.

Перші три методи прості у реалізації, але показують середню точність класифікації. Методи машинного навчання, у тому числі й нейронні мережі, значно складніші в програмній реалізації, але мають високу точність діагностування.

2.1 Штучні нейронні мережі

Основою всіх архітектур для відеоспостереження є аналіз, першою фазою якого буде розпізнавання зображення (об'єкта). Потім штучний інтелект за допомогою машинного навчання розпізнає дії і класифікує їх. Найбільш популярний спосіб застосування нейронних мереж – це розпізнавання зображень. Для того щоб розпізнати зображення, нейронна мережа повинна бути спершу навчена на даних. Це дуже схоже на нейронні зв'язки в людському мозку – ми володіємо певними знаннями, бачимо об'єкт, аналізуємо його і ідентифікуємо [14].

Нейронні мережі вимогливі до розміру і якості набору даних, на якому вони будуть навчатися. Важливо при навчанні навчити мережу визначати не тільки достатню кількість і значення ознак, щоб показувати хорошу точність на нових зображеннях, але й не перенавчатися, тобто, надмірно не «підлаштуватися» під навчальну вибірку зображень. Після завершення

правильного навчання нейронна мережа повинна вміти визначати образи (тих же класів), з якими вона не мала справи в процесі навчання.

Важливо враховувати, що вихідні дані для нейронної мережі повинні бути однозначні і несуперечливі, щоб не виникали ситуації, коли мережа буде видавати високі ймовірності приналежності одного об'єкта до декількох класів.

Сучасний рівень розвитку технологій і те, що сьогодні нейронні мережі застосовують в самих різних сферах, показує, що у вони мають величезні перспективи розвитку в різних областях, а їх використання є більш ніж доцільним.

Проведемо теоретичний аналіз видів нейронних мереж, для того щоб визначити, яку з них доцільно використовувати для вирішення поставленої задачі.

2.1.1 Глибинні нейронні мережі

Глибинна нейронна мережа (ГНМ) – це штучна нейронна мережа з декількома прихованими шарами вузлів між вхідним та вихідним шарами [14]. Подібно до плоских мереж, ГНМ можуть моделювати складні нелінійні відношення.

ГНМ знаходить коректний метод математичних перетворень, щоб перетворити вхідні дані у вихідні, незалежно від лінійної або нелінійної кореляції. Мережа просувається по шарам, розраховуючи ймовірність кожного виходу. Наприклад, ГНМ, яка навчена розпізнавати породи собак, пройде по заданому зображенню і вирахує ймовірність того, що собака на зображенні відноситься до певної породи. Користувач може переглянути результати і вибрати ймовірності, які повинна відображати мережа (вище певного порогу, наприклад), і повернути у мережу запропоновану мітку. Кожне математичне перетворення вважається шаром, а складні ГНМ мають багато шарів, звідси і назва «глибинні» або «глибокі» мережі [14].

ГНМ можуть моделювати складні нелінійні відносини. Архітектури ГНМ генерують композиційні моделі, в яких об'єкт виражається у вигляді багаторівневої композиції примітивів [15]. Додаткові рівні дозволяють складати елементи з більш низьких рівнів, потенційно моделюючи складні дані з меншою кількістю одиниць, ніж дрібна мережа з аналогічними показниками.

Глибока архітектура включає в себе безліч варіантів кількох основних підходів. Кожна архітектура знайшла успіх в певних областях. Не завжди можливо порівняти продуктивність декількох архітектур, якщо вони не були оцінені на одних і тих же наборах даних [15].

ГНМ, як правило, представляють собою мережі з прямим зв'язком, в яких дані передаються від вхідного рівня до вихідного рівня без зворотного зв'язку. Спочатку ГНМ створює карту віртуальних нейронів і призначає випадкові числові значення або «ваги» з'єднанням між ними. Ваги і вхідні дані множаться і повертають вихідний сигнал від 0 до 1. Якщо мережа не точно розпізнала конкретний шаблон, алгоритм буде коригувати вагові коефіцієнти. Таким чином, алгоритм може зробити певні параметри більш значущими, поки він не визначить правильні математичні маніпуляції для повної обробки даних.

Уточнення вагових коефіцієнтів може здійснюватися стохастичним найшвидшим спуском із застосуванням рівняння, приведеного на формулі (2.1):

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\delta C}{\delta w_{ij}}, \quad (2.1)$$

де w_{ij} – вектор-стовпець ваг для всіх входів нейрона j в шарі i ;

t – вектор цільових вихідних значень;

η – темп навчання;

C – функція витрат.

Вибір функції витрат залежить від таких факторів, як тип навчання (кероване, спонтанне, з підкріпленням тощо) та функції активації. Наприклад, при виконанні керованого навчання для задачі багатокласової класифікації звичайним вибором для функції активації та функції витрат є нормована експоненційна функція та функція перехресної ентропії відповідно.

З ГНМ може виникати багато проблем, якщо вони тренуються наївно. Двома поширеними проблемами є перенавчання та тривалість обчислення [15].

2.1.2 Згорткові нейронні мережі

Згорткові нейронні мережі (ЗНМ) в машинному навчанні – це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. ЗНМ використовують різновид багат шарових перцептронів (БП), розроблений так, щоб використовувати мінімального обсяг попередньої обробки [16].

Згорткові мережі взяли за основу біологічний процес, а саме схему з'єднання нейронів зорової кори тварин [16]. ЗНМ використовують порівняно мало попередньої обробки, в порівнянні з іншими алгоритмами класифікування зображень. Це означає, що мережа навчається з фільтрів, що в традиційних алгоритмах конструювали вручну. Ця незалежність у конструюванні ознак від апріорних знань та людських зусиль є великою перевагою.

Вони застосовуються в розпізнаванні зображень та відео, рекомендаційних системах [17] та обробці природної мови.

Ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів і шарів підвибірки. Структура мережі – односпрямована (без зворотних зв'язків), принципово багат шарова. Для навчання використовуються стандартні методи, найчастіше метод зворотного

поширення помилки. Функція активації нейронів – будь-яка за вибором дослідника.

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації [17].

Цей процес описують в нейронних мережах як згортку за домовленістю. З математичної точки зору він є взаємною кореляцією, ніж згорткою. Це має значення лише для індексів у матриці. ЗНМ використовують спільні ваги в згорткових шарах, що означає, що для кожного рецептивного поля шару використовується один і той же фільтр (банк ваги). Це зменшує обсяг необхідної пам'яті та поліпшує продуктивність.

ЗНМ часто застосовують у системах розпізнавання зображень. 2012 року було повідомлено про 0.23-відсотковий рівень похибки на базі даних MNIST. 2015 року багат шарова ЗНМ продемонструвала здатність з конкурентоспроможною продуктивністю помічати обличчя з великого діапазону кутів, включно із перевернутими, навіть частково закритими [17].

2.1.3 Багат шаровий перцептрон

БП є класом прямих штучних нейронних мереж. БП складається щонайменше з трьох шарів вузлів: вхідного шару, прихованого шару і вихідного шару. За винятком вхідних вузлів, кожен вузол є нейроном, який використовує нелінійну функцію активації. БП використовує алгоритм зворотного поширення помилки для навчання. Багат шаровість і нелінійна активація відрізняють БП від лінійного перцептрона [18]-[19].

Якщо БП має лінійну функцію активації у всіх нейронах, тобто лінійну функцію, яка відображає зважені входи на вихід кожного нейрона, то лінійна алгебра показує, що будь-яка кількість шарів може бути зведена до двошарової вихідної моделі. У БП деякі нейрони використовують нелінійну функцію активації [18].

У якості функції активації використовують сигмоїдальну функцію, приведену на формулі (2.2):

$$y(v_i) = (1 + e^{-v_i})^{-1}, \quad (2.2)$$

де y_i – вихідне значення i -го вузла (нейрона);

v_i – зважена сума входних зв'язків.

Вибір функції активації робиться в залежності від завдання, зручності програмної реалізації, а також алгоритму навчання.

Мережа складається з довільного числа шарів. Розрізняють вхідний шар – на який надходить вектор (зазвичай він не враховується у підрахунку шарів), вихідний шар, який видає результат розрахунків мережі і прихований шар. Вхідний шар не виконує ніяких обчислень – він просто розподіляє вектор на наступний шар і служить приймачем інформації. Зазвичай, якщо прихованих шарів більше, ніж два, то у всіх прихованих шарах однакова кількість нейронів [19].

Кожен шар розраховує нелінійне перетворення від лінійної комбінації сигналів попереднього шару. Багат шарова нейронна мережа може формувати на виході довільну багатовимірну функцію при відповідному виборі кількості шарів, діапазону зміни сигналів і параметрів нейронів. Багат шарові нейронні мережі є універсальним апроксиматором функцій [19].

За рахунок почергового розрахунку лінійних комбінацій і нелінійних перетворень досягається апроксимація довільної багатовимірної функції при відповідному виборі параметрів мережі [20].

Загальна формула (2.3), яка описує БП:

$$f(x) = F\left(\sum_{i_N} w_{i_N j_N} \dots \sum_{i_2} w_{i_2 j_2} F\left(\sum_{i_1} w_{i_1 j_1} - \theta_{j_1}\right) - \theta_{j_2} \dots - \theta_{j_N}\right), \quad (2.3)$$

де $f(x)$ – прогнозоване значення;

F – нелінійна функція активації;

w_{ijn} – вектор-стовпець ваг для всіх входів нейрона j в шарі i ;

θ – пороговий рівень даного нейрона.

Переваги БП:

- гарантоване отримання відповіді після проходу даних по шарах;
- простота у використанні;
- здатність моделювання функції будь-якого ступеня складності [20].

Недоліки методу:

- обмеження по швидкості і об'єму використовуваної пам'яті;
- складність розрахунку кількості прихованих шарів, що необхідні для вирішення даного завдання;
- складність розрахунку кількості нейронів в кожному шарі;
- можливість перенавчання;
- необхідно вибирати структуру мережі під певне завдання [20].

2.1.4 Автокодувальник

Автокодувальник – це штучна нейронна мережа, що використовується для навчання ефективних кодувань. Метою автокодувальника є навчитися представленню (кодуванню) набору даних, зазвичай задля зниження розмірності [21]. Нещодавно концепція автокодувальника стала застосовуватися ширше для навчання породжувальних моделей даних.

Автокодувальник має спеціальну архітектуру штучних нейронних мереж, що дозволяє застосовувати навчання без вчителя при використанні методу зворотного поширення помилки. Найпростіша архітектура автокодувальника – мережа прямого поширення, без зворотного зв'язку, найбільш схожа з перцептроном, містить вхідний шар, проміжний шар і вихідний шар. На відміну від перцептрону, вихідний шар автокодувальника повинен містити стільки ж нейронів, скільки і вхідний шар [21].

Основний принцип роботи і навчання мережі автокодувальника полягає у тому, щоб отримати на вихідному шарі відгук, найбільш близький до вхідного. Щоб рішення не виявилось тривіальним, на проміжний шар автокодувальника накладають обмеження: проміжний шар повинен бути або меншої розмірності, ніж вхідний і вихідний шари, або штучно обмежується кількість одночасно активних нейронів проміжного шару – розріджена активація. Ці обмеження змушують нейронну мережу шукати узагальнення і кореляцію в даних, що надходять на вхід, виконувати їх стиснення. Таким чином, нейронна мережа автоматично навчається виділяти з вхідних даних загальні ознаки, які кодуються в значеннях ваг. Так, при навчанні мережі на наборі різних вхідних зображень, вона може самостійно навчитися розпізнавати лінії і смуги під різними кутами [21].

Найчастіше автокодувальника застосовують каскадно для навчання глибоких (багатошарових) мереж. Автокодувальники застосовують для попереднього навчання глибокої мережі без вчителя. Для цього шари навчаються один за одним, починаючи з перших. До кожного нового ненавченого шару на час навчання підключається додатковий вихідний шар, що доповнює мережу до архітектури автокодувальника, після чого на вхід мережі подається набір даних для навчання. Ваги ненавченого шару і додаткового шару автокодувальника навчаються за допомогою методу зворотного поширення помилки. Потім шар автокодувальника вимикається і створюється новий, відповідний наступному ненавченому шару мережі. На вхід мережі знову подається той же набір даних, навчені перші шари мережі залишаються без змін і працюють в якості вхідних для чергового учня. Так навчання триває для всіх шарів мережі за винятком останніх. Останні шари мережі зазвичай навчаються без використання автокодувальника за допомогою того ж методу зворотного поширення помилки і на маркованих даних (навчання з учителем) [21].

Основними практичними застосуваннями автокодувальника є зменшення шуму в даних, а також зменшення розмірності багатовимірних

даних для візуалізації. З певними застереженнями щодо розмірності і розрідженості даних, автокодувальники можуть дозволяти отримувати проєкції багатовимірних даних, які виявляються кращими, ніж ті, що дає метод головних компонент або будь-який інший класичний метод [21].

2.2 Аналіз об'єктів на зображеннях

Нейронні мережі потребують великий обсяг даних для навчання та достовірної класифікації об'єктів. Також великий вплив на результати класифікації має якість зображень бази даних. У цьому підрозділі будуть проаналізовані методи аналізу об'єктів на зображеннях з метою покращення початкових даних.

2.2.1 Перетворення Хафа

Перетворення Хафа – обчислювальний алгоритм, що застосовується для параметричної ідентифікації геометричних елементів растрового зображення. Використовується в аналізі зображень, цифровій обробці зображень і комп'ютерному зорі. Призначений для пошуку об'єктів, що належать певному класу фігур, з використанням процедури голосування. Процедура голосування застосовується до простору параметрів, з якого і виходять об'єкти певного класу фігур по локального максимуму в так званому накопичувальному просторі, яке будується при обчисленні перетворення Хафа [22].

Класичний алгоритм перетворення Хафа пов'язаний з ідентифікацією прямих в зображенні, але пізніше алгоритм був розширений можливістю ідентифікації позиції довільної фігури, найчастіше еліпсів і кіл [22].

При автоматизованому аналізі цифрових зображень дуже часто виникає проблема ідентифікації простих фігур, таких як прямі, кола або еліпси. У багатьох випадках використовується алгоритм пошуку контурів як

перший етап обробки зображення для отримання точок, що знаходяться на кривій в зображенні. Однак, або через велику кількість шумів на зображенні, або через недосконалість алгоритму виявлення контурів, можуть з'явитися «втрачені» точки на кривій, також як і невеликі відхилення від ідеальної форми прямої, кола або еліпса. З цих причин часто досить складно приписати знайдені контури відповідно прямим, колам і еліпсам на зображенні. Призначення перетворення Хафа – вирішити проблему угруповання граничних точок шляхом застосування певної процедури голосування до набору параметризованих об'єктів зображення [22].

Алгоритм перетворення Хафа використовує масив, який називається акумулятором. Розмірність акумулятора дорівнює кількості невідомих параметрів простору Хафа [22].

Перетворення Хафа ефективно тільки при значній кількості «влучень» у відповідний елемент простору Хафа, тільки тоді можна з упевненістю визначити фігуру, нехтуючи фоновим шумом. Це означає, що розмір елемента не повинен бути дуже маленьким, а то деякі значення потраплять в сусідні елементи, зменшуючи видимість потрібного елемента.

Ефективність алгоритму у великій мірі обумовлена якістю вхідних даних: контури фігур на етапі попередньої обробки зображення повинні бути чітко визначені. Використання перетворення Хафа на зображеннях з великою кількістю шуму ускладнено [22].

2.2.2 Перетворення Фур'є

Перетворення Фур'є – інтегральне перетворення однієї функції дійсної змінної на іншу. Використовується для того, щоб розрахувати спектр частот для сигналів змінних у часі (як-от мова або електрична напруга) [23].

Перетворення Фур'є функції $f(t)$ математично визначається як комплексна функція $F(\omega)$, яка задається інтегралом, представлена на формулі (2.4):

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt . \quad (2.4)$$

Одним з методів, що дозволяють значно скоротити кількість обчислень, є застосування Фур'є перетворень і дискретних Фур'є перетворень для виявлення ступеню збігу двох зображень при різних зсувах їх між собою. Обчислення при цьому відбуваються одночасно для різних комбінацій зсувів зображень відносно один одного [23].

У спектрі сигналу низькочастотні компоненти показують ті частини зображення, де яскравість практично не змінюється. Високочастотні компоненти показують зміни в інтенсивності. Таким чином, якщо залишити тільки низькі частоти (низькочастотний фільтр), то після зворотного перетворення Фур'є будуть отримані лише розпливчасті плями. Якщо ж залишити тільки високі частоти (високочастотний фільтр), то залишаться в основному тільки контури предметів, так як саме на їхніх контурах різко змінюється яскравість зображення [23].

Відповідно до властивості Фур'є, «зсув у часі» при зміні положення об'єкта на зображенні, в області частот змінюється тільки фаза, таким чином, по амплітуді можна зрозуміти, що це саме цей об'єкт, нехай навіть він знаходиться в іншому місці на зображенні. Якщо ж на зображенні є безліч різних об'єктів, то їх спектри складуться, і буде неможливо дізнатися, чи був присутній на зображенні шуканий об'єкт. Зрозуміло, відповідно до властивості лінійності, можна з спектра зображення відняти спектр шуканого об'єкта. Тоді, якщо об'єкт на зображенні присутній, то він зникне з зображення. Якщо на зображенні шуканого предмета не було, то результат буде залежати від того, як багато частот займає віднімається об'єкт. А саме, або будуть невеликі перешкоди на зображенні, або зображення стане нерозбірливим [23].

2.2.3 Об'єднання даних

Об'єднання даних – це техніка попередньої обробки даних, що використовується для зменшення наслідків незначних помилок спостереження. Вихідні значення даних, які потрапляють у заданий невеликий інтервал, бункер, замінюються значенням, репрезентативним для цього інтервалу, часто центральним значенням. Це форма квантування [24].

Статистичне об'єднання даних – це спосіб групування чисел більш-менш безперервних значень у меншу кількість "бункерів". Наприклад, якщо у вас є дані про групу людей, можливо, ви захочете розкласти їх вік на меншу кількість вікових інтервалів (наприклад, групувати кожні п'ять років разом).

У контексті обробки зображень об'єднання даних – це процедура об'єднання кластера пікселів в один піксель. Таким чином, при об'єднанні 2 на 2 масив з 4 пікселів стає одним великим пікселем, зменшуючи загальну кількість пікселів [24].

Це узагальнення, хоча і пов'язане з втратою інформації, зменшує обсяг даних, що підлягають обробці, полегшуючи аналіз. Наприклад, об'єднання даних може також зменшити вплив шуму на оброблене зображення (ціною нижчої роздільної здатності) [24].

Гістограми є прикладом об'єднання даних, що використовується для спостереження базового розподілу. Зазвичай вони зображуються в одновимірному просторі та через рівні інтервали для зручності візуалізації. Об'єднання даних також використовується в машинному навчанні, щоб прискорити метод дерева прийняття рішень для контрольованої класифікації та регресії [25].

2.3 Розроблений метод аналізу об'єктів на зображеннях

Теоретичний аналіз методів аналізу зображень показав доцільність їх використання. Але кожен з методів має як переваги, так і недоліки.

Перетворення Фур'є показують кращі результати при аналізі зображень, які були приведені до чорно-білої кольорової палітри.

Перетворення Хафа та об'єднання даних потребують зображення, на яких немає різких переходів, наприклад, від початкового зображення до програмно доданого фону чорного кольору. Такі переходи будуть спотворювати результати аналізу.

Враховуючи особливості бази даних з зображеннями друкованих плат, було розроблено алгоритм аналізу орієнтації об'єктів, а саме чіпів, на зображеннях. Застосування розробленого алгоритму нормалізує дані у базі та приведе їх до загального вигляду – у нашому випадку це горизонтальна орієнтація чіпів на зображеннях. Нормалізація зображень спрощує процес знаходження закономірностей та особливостей об'єктів нейронними мережами, зменшує кількість хибних ознак вхідних даних.

На рисунку 2.1 представлена логічна схема розробленого алгоритму аналізу об'єктів.

Розроблений алгоритм складається з наступних кроків.

Крок 1. Завантаження зображень з бази даних.

Крок 2. Створення трьох груп для розподілення зображень: хибно орієнтовані, правильно орієнтовані та зображення, що потребують рішення оператора.

Крок 3. Задання правильного та хибного нахилів об'єкту на зображеннях у градусах. Якщо зображення може бути опрацьовано і вертикально, і горизонтально, то доцільно вибрати ту орієнтацію, у якій розташовано більшість зображень у базі.

Крок 4. Якщо на зображеннях є чорний фон, який був доданий програмно, для того щоб привести всі зображення до одного розміру, то цей фон потрібно прибрати. Таким чином на зображенні залишається лише сам об'єкт.

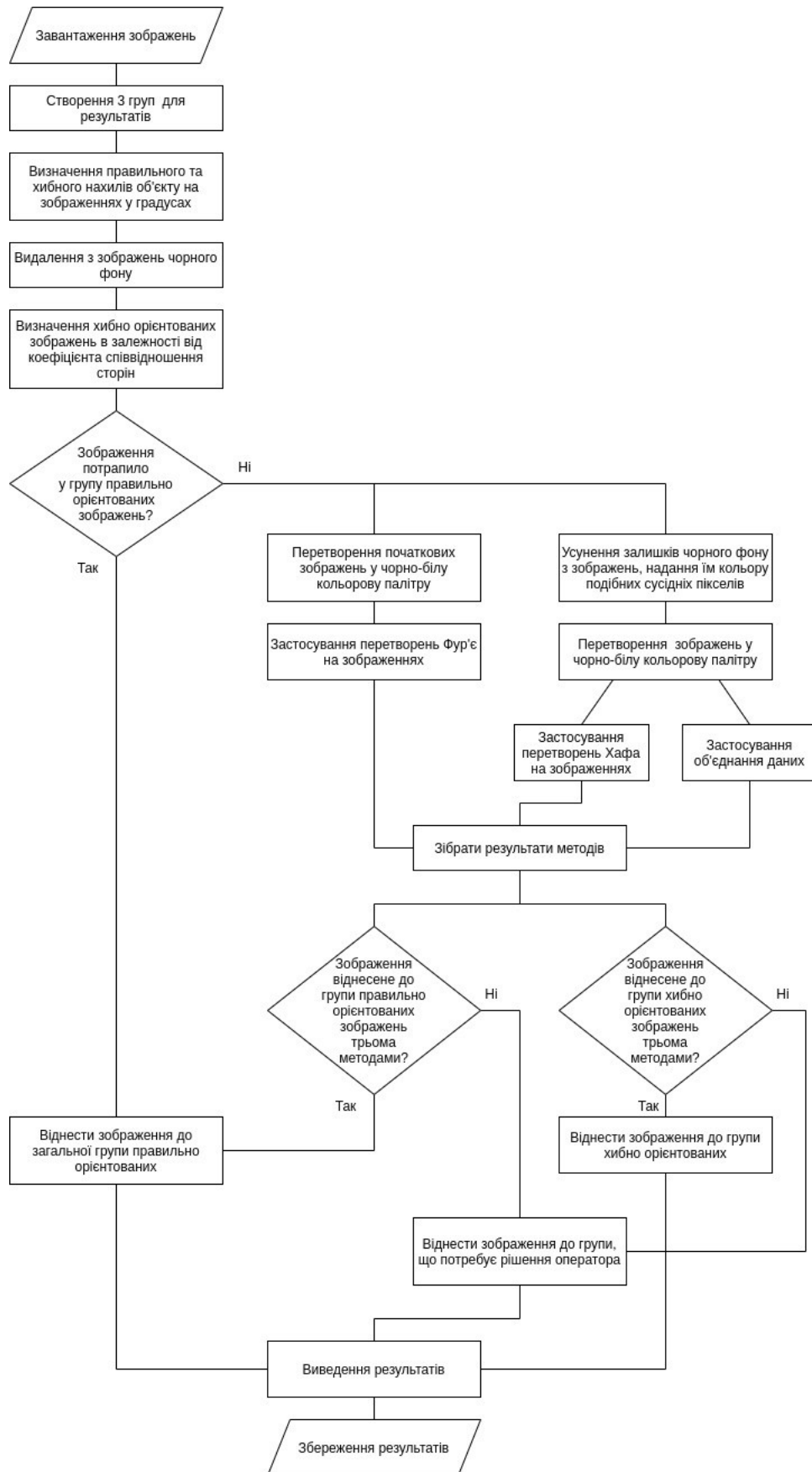


Рисунок 2.1 – Схема алгоритму аналізу об'єктів

Крок 5. В залежності від типу об'єкту на зображенні, визначити коефіцієнт правильного співвідношення сторін об'єкта, у нашому випадку чіпу. Розділити зображення на дві групи: ті, що мають коефіцієнт співвідношення сторін більший за мінімально визначений, та інші.

Крок 6. Перша група з попереднього кроку – це правильно орієнтовані зображення, занести їх до загальної групи правильно орієнтованих зображень.

Крок 7. Друга група – це зображення, що потребують подальшого дослідження.

Крок 8. Перетворити зображення другої групи у чорно-білу кольорову палітру.

Крок 9. Застосувати перетворення Фур'є на зображеннях.

Крок 10. Усунути залишки чорного фону з другої групи зображень, а саме замінити чорні пікселі на пікселі подібного до сусідніх кольори.

Крок 11. Перетворити, створені на кроці 10, зображення на чорно-білу палітру.

Крок 12. Застосувати перетворення Хафа на зображеннях кроку 11.

Крок 13. Застосувати об'єднання даних на зображеннях кроку 11.

Крок 14. Зібрати результати аналізів на кроках 9, 12, 13.

Крок 15. Якщо зображення було помічено як правильно орієнтовано всіма трьома методами, то віднести його до загальної групи правильно орієнтованих зображень.

Крок 16. Якщо зображення було помічено як хибно орієнтовано всіма трьома методами, то віднести його до загальної групи хибно орієнтованих зображень.

Крок 17. Усі зображення, що досі не потрапили ні в одну групу результатів, вважаються тими, які потребують рішення оператора. Це можуть бути зображення з об'єктами, які випадково потрапили до бази даних, або пошкоджені зображення. Якщо оператора немає, то такі зображення потрібно

видалити з бази даних, тому що вони будуть погіршувати точність нейронної мережі.

Крок 18. Вивести результати аналізу.

Крок 19. Зберегти результати аналізу у файл.

2.4 Висновки за розділом

Був проведений теоретичний аналізу різних видів нейронних мереж, а саме: ГНМ, ЗНМ, БП, автокодувальник. Аналіз показав, що для вирішення поставленої задачі діагностування дефектів друкованих плат доцільно використовувати ЗНМ.

Було проаналізовано такі методи аналізу зображень:

- перетворення Хафа;
- перетворення Фур'є;
- метод об'єднання даних.

З метою покращення початкових даних було розроблено алгоритм попереднього аналізу зображень у базі. Запропонований алгоритм використовує усі розглянуті методи аналізу зображень послідовно. Перед застосуванням кожного методу зображення попередньо оброблюються: прибирається фон, змінюється кольорова палітра, перефарбовуються пікселі чорного кольору.

3 ОПИС ПРОГРАМИ

3.1 Логічна структура програми

Для вирішення поставленої задачі було спроектовано структурну схему програми (рис. 3.1) та схему функціонування програми (рис. 3.2).

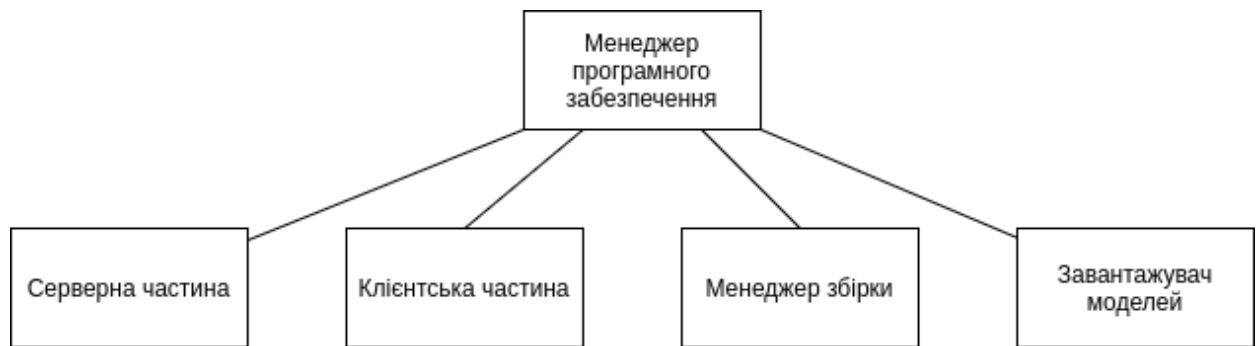


Рисунок 3.1 – Структурна схема програми

Для ознайомлення з системою нижче описані структурна схема та схема функціонування.

Система складається з 4 основних частин, які працюють разом:

- серверна частина;
- клієнтська частина;
- менеджер збірки;
- завантажувач моделей.

Серверна частина відповідає за машинне навчання, логінг та всю системну логіку.

Клієнтська частина відповідає за графічний інтерфейс користувача (ГІК) та дії користувача.

Менеджер збірки використовується для роботи з прикладним програмним інтерфейсом (ППІ).

Завантажувач моделей завантажує всі доступні та реалізовані моделі, щоб користувач мав до них доступ.

Система розрахована на 2 типи користувачів. На рисунку 3.2 показана схема функціонування програми для двох типів користувачів.



Рисунок 3.2 – Схема функціонування програми

Один користувач запускає сервер з необхідними параметрами, завантажує дані та починає тренування нейронної мережі.

Нейронна мережа класифікує зображення за класами. Якщо нейронна мережа не може віднести зображення до будь-якого з зазначених класів, тоді вона поміщає його до «ручного» класу. Це означає, що зображення вимагає

розпізнавання вручну і має бути класифіковане другим типом користувача – оператором.

Оператор запускає клієнтську частину і чекає завдання. Коли нейронна мережа класифікує зображення як «ручне», система надсилає його користувачеві. Він вирішує, до якого класу належить отримане зображення, і надсилає відповідь назад до нейронної мережі. Вона аналізує відповідь та активно донавчається.

3.2 Програмні засоби та мови програмування

Для вибору мови програмування було порівняно їх особливості (табл. 3.1).

Таблиця 3.1 – Порівняння мов програмування

Мова програмування	Python	C++	Lua	Java	R	MATLAB
Продуктивність	-	+	-	+	-	-
Універсальність	+	-	+	+	-	-
Візуалізації даних	+	-	+	+	+	+
Бібліотеки для використання нейронних мереж	+	+	+	+	+	+
Ліцензія	Безкоштовна	Безкоштовна	Безкоштовна	Безкоштовна	Безкоштовна	Платна
Простота у вивченні	+	-	+	-	-	-
Стабільність	+	+	+	+	+	+
Читабельність	+	-	+	+	+	-

Враховуючи усі вимоги до створюваної системи та її призначення, було вирішено обрати для розробки мову програмування Python 3.8 [26].

В якості середовища розробки було обрано Jupyter Notebook. Jupyter Notebook – це веб-інтерактивне обчислювальне середовище для створення документів Jupyter notebook [27].

Для реалізації віддалених запитів на клієнтському боці було використано gRPC фреймворк. gRPC – це сучасний безкоштовний фреймворк, який допомагає реалізувати RPC протокол [28].

Для серіалізації даних був використаний формат Protocol Buffers. Protocol Buffers – формат серіалізації даних, запропонований корпорацією Google, як альтернатива XML [29].

Для реалізації та використання нейронних мереж, навчання та тренування моделей нейронних мереж було обрано безкоштовну бібліотеку TensorFlow [30].

Для візуалізації результатів двовимірною 2D графікою використана графічна бібліотека на мові програмування Python matplotlib [31]. Зображення використані для перевірки та систематизації отриманих результатів.

3.2.1 Особливості мови програмування Python

Python – це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та читання коду [32].

Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій. Python підтримує кілька парадигм програмування, в тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне.

Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних.

Код в Python організовується у функції та класи, які можуть об'єднуватися в модулі, а вони в свою чергу можуть бути об'єднані в пакети.

Еталонної реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках, включаючи пропріетарні. Проєкт PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови і постановку експериментів над новими можливостями.

Python – це мова програмування, що активно розвивається, нові версії (з додаванням / зміною мовних властивостей) виходять приблизно раз в півроку. Python підтримує динамічну типізацію, тобто тип змінної визначається лише під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». В Python є вбудовані типи: булевий, рядок, Unicode-рядок, ціле число довільної точності, число з плаваючою крапкою, комплексне число і деякі інші. З колекцій в Python вбудовані: список, кортеж (незмінний список), словник, множина і інші. Всі значення є об'єктами, в тому числі функції, методи, модулі, класи. Додати новий тип можна написавши клас (class) або визначивши новий тип в модулі розширення (наприклад, написаному на мові C). Система класів підтримує спадкування (одиначне і множинне) і метапрограмування. Можливо успадкування від більшості вбудованих типів і типів розширень.

Всі об'єкти діляться на посилальні і атомарні. До атомарним відносяться int, long і деякі інші. При присвоєнні атомарних об'єктів копіюється їх значення, в той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують одне і те ж значення. Довідкові об'єкти бувають змінювані і незмінні. Наприклад, рядки і кортежі є незмінними, а списки, словники і багато інших об'єктів – змінними.

Кортеж в Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх.

3.2.2 Особливості середовища розробки Jupyter Notebook

Jupyter Notebook – це веб-програма з відкритим вихідним кодом, яка дозволяє створювати та обмінюватися документами, що містять код, рівняння, візуалізації та текст файлу. Використання включає: очищення та перетворення даних, чисельне моделювання, статистичне моделювання, візуалізацію даних, машинне навчання та багато іншого [27].

Jupyter підтримує понад 40 мов програмування, включаючи Python, R, Julia та Scala. Ноутбуками можна ділитися з іншими за допомогою електронної пошти, Dropbox, GitHub та Jupyter Notebook Viewer. Код може дати багатий інтерактивний результат: HTML, зображення, відео, LaTeX та власні типи MIME.

Jupyter Notebook дає можливість використовувати інструменти для обробки великих даних, такі як Apache Spark, від Python, R та Scala та дослідити ці самі дані за допомогою бібліотек pandas, scikit-learn, ggplot2, TensorFlow.

3.2.3 Особливості gRPC фреймворку

gRPC – це сучасний високопродуктивний фреймворк з відкритим кодом для реалізації виклику віддалених процедур (ВВП). Він може працювати в будь-якому середовищі [28].

gRPC розроблен в Google у 2015 році. Він використовує HTTP/2 для транспортування, Protocol Buffers як мову опису ППІ, а також надає такі функції, як аутентифікація, двоспрямована потокова передача та управління

поток, блокування або неблокування прив'язок, а також скасування та таймауту.

У gRPC клієнтська програма може безпосередньо викликати метод з серверної програми на іншій машині, ніби це локальний об'єкт. Як і в багатьох системах ВВП, gRPC базується на ідеї створення сервісу, якому можна вказати методи, які можна викликати віддалено з їх параметрами та значеннями, що повертаються. На стороні сервера, сервер реалізує цей ППІ і запускає сервер gRPC для обробки викликів клієнта. З боку клієнта клієнт має заглушку, яка визначає ті самі методи, що і сервер.

Клієнти та сервери gRPC можуть запускатися та розмовляти між собою в різних середовищах і можуть бути написані будь-якою з підтримуваних мов gRPC. Так, наприклад, ви можете легко створити gRPC-сервер на Java за допомогою клієнтів у Go, Python або Ruby.

gRPC має чотири типи методів обслуговування:

- одинарний;

Клієнт відправляє один запит на сервер і отримує одну відповідь назад.

- потокове передавання з сервера;

Клієнт відправляє запит на сервер і отримує потік для читання послідовності повідомлень назад. Клієнт читає з повернутого потоку, поки повідомлення не закінчаться. gRPC гарантує порядок повідомлень у межах окремого ВВП.

- потокове передавання з клієнта;

Клієнт пише послідовність повідомлень і відправляє їх на сервер за допомогою наданого потоку. Після того, як клієнт закінчить писати повідомлення, він чекає, поки сервер прочитає його і поверне свою відповідь. gRPC гарантує порядок повідомлень клієнта у межах окремого ВВП.

- двоспрямоване потокове передавання.

Обидві сторони надсилають послідовність повідомлень за допомогою потоку читання-запису. Два потоки працюють незалежно, тому клієнти та

сервери можуть читати та писати у будь-якому порядку. Порядок повідомлень у кожному потоці зберігається.

3.2.4 Особливості формату Protocol Buffers

Protocol buffers – це гнучкий, ефективний, автоматизований механізм серіалізації структурованих даних. Ви визначаєте один раз, як ви хочете, щоб ваші дані були структуровані, потім ви можете використовувати спеціально згенерований код, щоб легко зберігати та зчитувати структуровані дані з різних потоків даних та використовуючи різні мови [29].

Ви вказуєте, як потрібно структурувати інформацію, яку ви серіалізуєте, визначаючи типи повідомлень Protocol Buffers у файлах .proto. Кожне повідомлення – це невеликий логічний запис інформації, що містить ряд пар ім'я-значення.

Кожне поле в повідомленнях має бути анотоване одним з трьох модифікаторів:

- обов'язковий модифікатор означає, що значення поля є необхідним та не може бути пустим, інакше повідомлення буде вважатися "неініціалізованим";

- необов'язковий модифікатор означає, що поле може бути встановлене або не встановлене; якщо необов'язкове значення поля не встановлено, використовується значення за замовчуванням (нуль для числових типів, порожній рядок для рядків, false для логічних типів); для вбудованих повідомлень значенням за замовчуванням завжди є "екземпляр за замовчуванням" або "прототип" повідомлення, яке не має жодного з полів;

- повторний модифікатор означає, що поле може повторюватися будь-яку кількість разів (включаючи нуль); порядок повторюваних значень буде збережений у Protocol Buffers; він використовується для масивів.

3.2.5 Особливості бібліотеки TensorFlow

TensorFlow – відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення завдань побудови і тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття [33]. Застосовується як для досліджень, так і для розробки власних продуктів Google. Основний ППІ для роботи з бібліотекою реалізований для Python, також існують реалізації для R, C Sharp, C ++, Haskell, Java, Go і Swift.

Платформа використовуються в сервісах Google для розпізнавання мови, виділення облич на фотографіях, визначення схожості зображень, відсіювання спаму в електронних повідомленнях, підбору новин у Google News і організації перекладу з урахуванням смислу. Розподілені системи машинного навчання можна створювати на типовому обладнанні, завдяки вбудованій підтримці в TensorFlow рознесення обчислень на кілька процесорів.

TensorFlow пропонує кілька рівнів абстракції, можна вибрати відповідний для своїх потреб. Моделі можна створювати та тренувати за допомогою високорівневого ППІ Keras, що полегшує початок роботи з TensorFlow та машинним навчанням.

Незалежно від того, на серверах чи в Інтернеті, TensorFlow дозволяє легко навчати та розгортати свою модель, незалежно від того, яку мову чи платформу ви використовуєте.

3.2.6 Особливості бібліотеки matplotlib

matplotlib – бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою. Отримувані зображення можуть бути використані як ілюстрації в публікаціях [31].

Бібліотека `matplotlib` написана і підтримується в основному Джоном Хантером і поширюється на умовах BSD-подібної ліцензії. Зображення, які генеруються в різних форматах, можуть бути використані в інтерактивній графіці, наукових публікаціях, ГІК, веб-додатках, де потрібно будувати діаграми. В документації автор зізнається, що `matplotlib` починався з імітування графічних команд `MATLAB`, але є незалежним від нього проєктом.

Бібліотека `matplotlib` побудована на принципах об'єктно-орієнтованого програмування, але має процедурний інтерфейс `pylab`, який надає аналоги команд `MATLAB`.

3.3 Висновки за розділом

У даному розділі було розроблено структурну схему та схему функціонування програми, описано частини системи та два типи користувачів програмного забезпечення, порядок їх дій.

Було порівняно мови програмування, які надають можливість роботи з методами АОІ.

Також було обрано та проаналізовано такі програмні засоби:

- мову програмування Python 3;
- середовище розробки Jupyter Notebook;
- gRPC фреймворк для реалізації віддалених запитів на клієнтському боці;
- формат Protocol Buffers для серіалізації даних;
- бібліотеку для роботи з нейронними мережами TensorFlow;
- графічну бібліотеку `matplotlib`.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис структури файлів

Програма складається з декількох каталогів та окремих файлів. Кожен файл має своє призначення. Фізична структура файлів проекту зображена на рисунку 4.1.

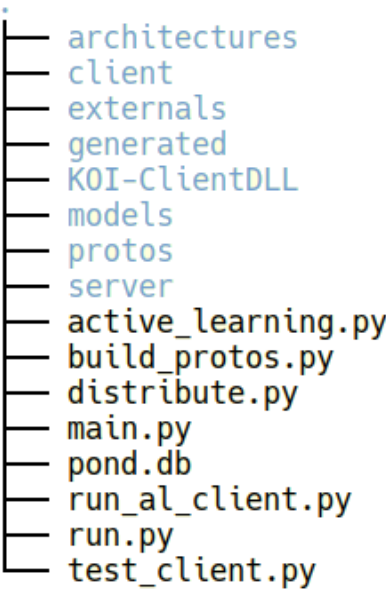


Рисунок 4.1 – Фізична структура файлів

Детальний опис файлів та їх призначення наведено у таблиці 4.1.

Таблиця 4.1 – Опис файлів проекту

Назва файлу	Опис
architectures	каталог містить допоміжні функції для активації шарів нейронної мережі, автоматичного кодування, кластеризації, загальних архітектур шарів, завантаження та управління великою кількістю навчальних файлів, бази даних, вилучення функцій, обробки міток класів, базового БП, деяких підказок numpy та алгоритмів

Продовження таблиці 4.1

Назва файлу	Опис
client	файли, що описують клієнтську частину, ГІК програми
externals	допоміжні файли C++ для ведення логів, побудови grpc, ППІ С для файлів TensorFlow
generated	файли, створені компілятором Protocol Buffers
KOI-ClientDLL	файли, що були згенеровані grpc C++ плагіном
models	файли, що реалізують доступні моделі нейронних мереж
protos	зберігає файли .proto grpc
server	файли, що реалізують серверну частину, систему та ведення логів
active_learning.py	файл, який є точкою входу для основного клієнта активного навчання системи АОІ
build_protos.py	скрипт, який створює файли grpc
main.py	файл визначає центральну точку входу для системи АОІ
pond.db	постійна база даних, яка містить усі знімки
run.py	точка входу для виконання зашифрованого розгортання системи
run_al_client.py	точка входу для виконання зашифрованого розгортання системи активного навчання
test_client.py	файл для тестування клієнтської частини системи

4.2 Опис допоміжних бібліотек, модулів та функцій

Під час реалізації програми були застосовані різні допоміжні бібліотеки, модулі та функції. Їх опис наведено у таблиці 4.2.

Таблиця 4.2 – Опис бібліотек, модулів та функцій

Назва модуля / функції	Опис
tkinter	Інструмент для створення ГІК на Python
aixlog	бібліотека ведення логів на C ++, містить лише заголовки
numpy	бібліотека для підтримки великих багатовимірних масивів і матриць, а також математичних функцій високого рівня для операцій з цими об'єктами
matplotlib	бібліотека побудови графіків для мови програмування Python та її числового математичного розширення numpy
protobuf	реалізація Protocol Buffers різними мовами
time	модуль для роботи з часом
os	модуль для роботи з операційною системою
uuid	модуль, який реалізує незмінні об'єкти – універсальні унікальні ідентифікатори (УУІД), – та містить функції для генерації УУІД версій 1, 3, 4 та 5
traceback	модуль, який реалізує стандартний інтерфейс для вилучення, форматування та друку стеків програм Python
queue	модуль, який реалізує черги для кількох виробників та споживачів
sqlite3	модуль, який надає інтерфейс SQL

4.3 Опис системних модулів

Система АОІ взаємодіє із зовнішніми модулями, такими як tkinter, ППІ С для TensorFlow, grpc, aixlog, TensorFlow, базою даних, користувачами та допоміжними бібліотеками (рис. 4.2).

Система складається з трьох основних частин: клієнтської, серверної та збірки (рис. 4.3).

Збірка – це процес компіляції файлів gRPC.

Клієнтська частина складається з клієнта АОІ та ГІК (рис. 4.4).

Серверна частина складається з трьох частин: логера АОІ, сервера АОІ та системи АОІ.

Логер АОІ допомагає слідкувати за процесами роботи системи.

Система АОІ зберігає всі зображення та моделі, організовані системою.

Сервер АОІ визначає функції gRPC для системи АОІ.

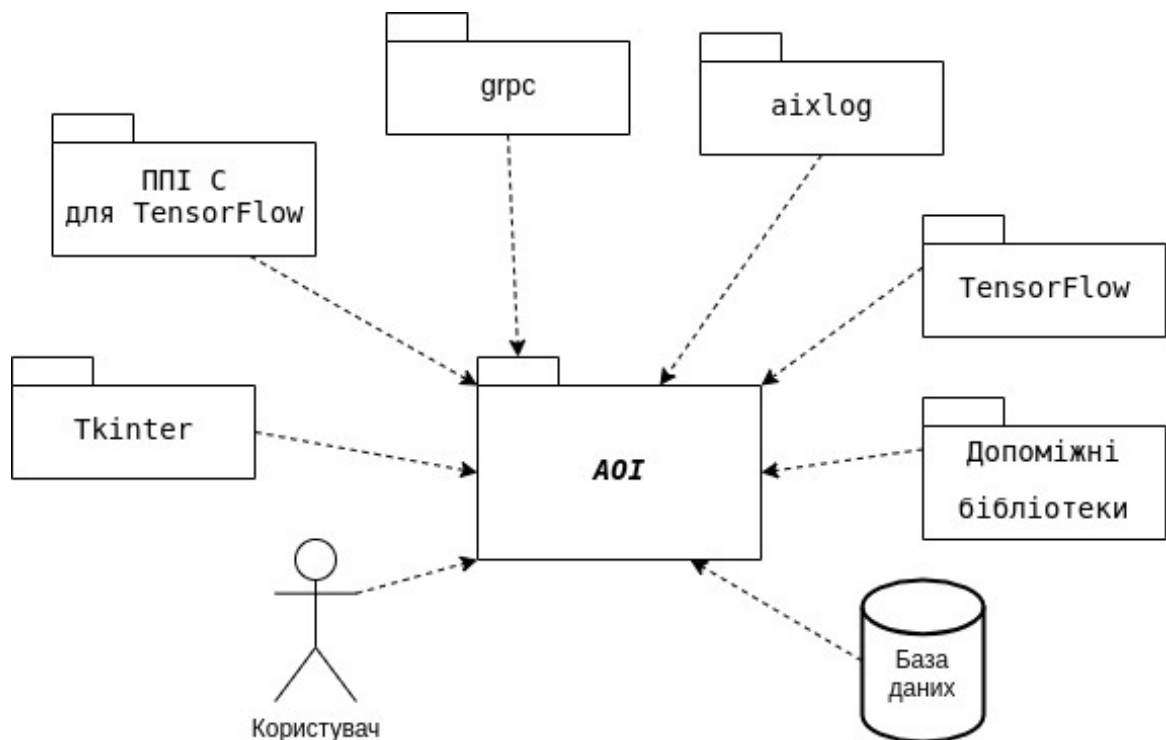


Рисунок 4.2 – Діаграма компонентів системи

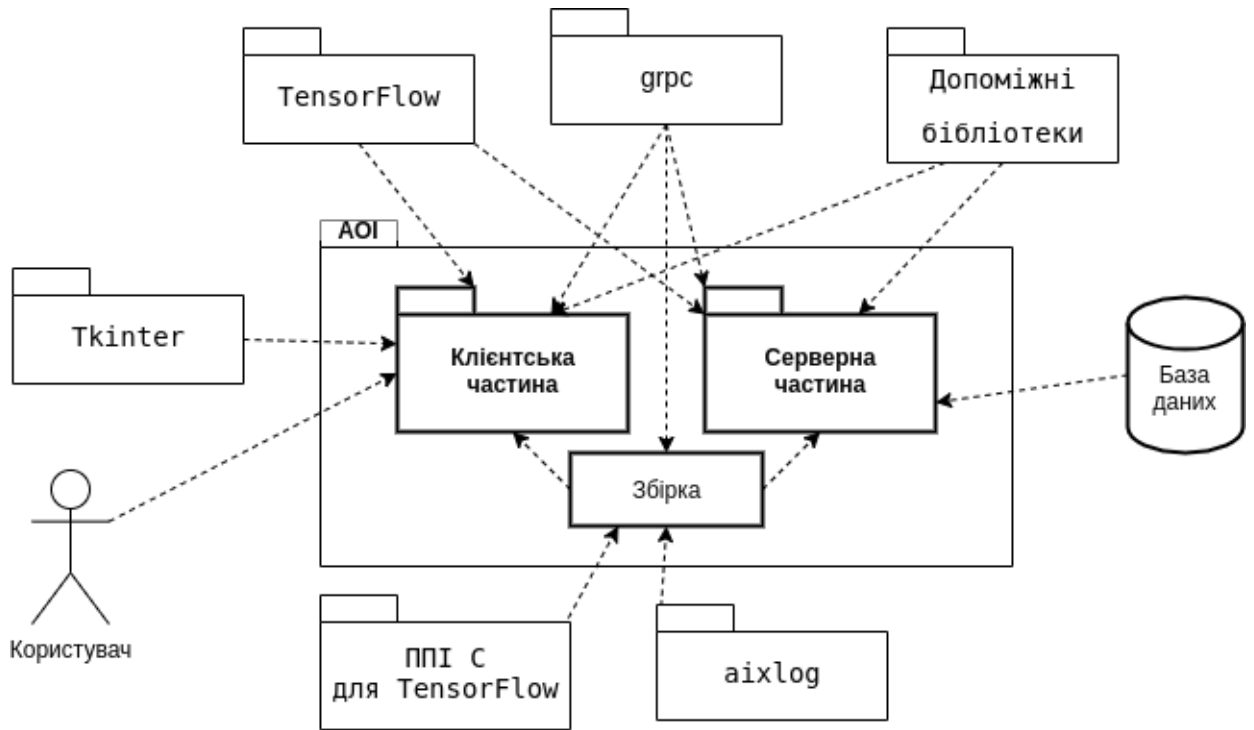


Рисунок 4.3 – Діаграма компонентів частин системи

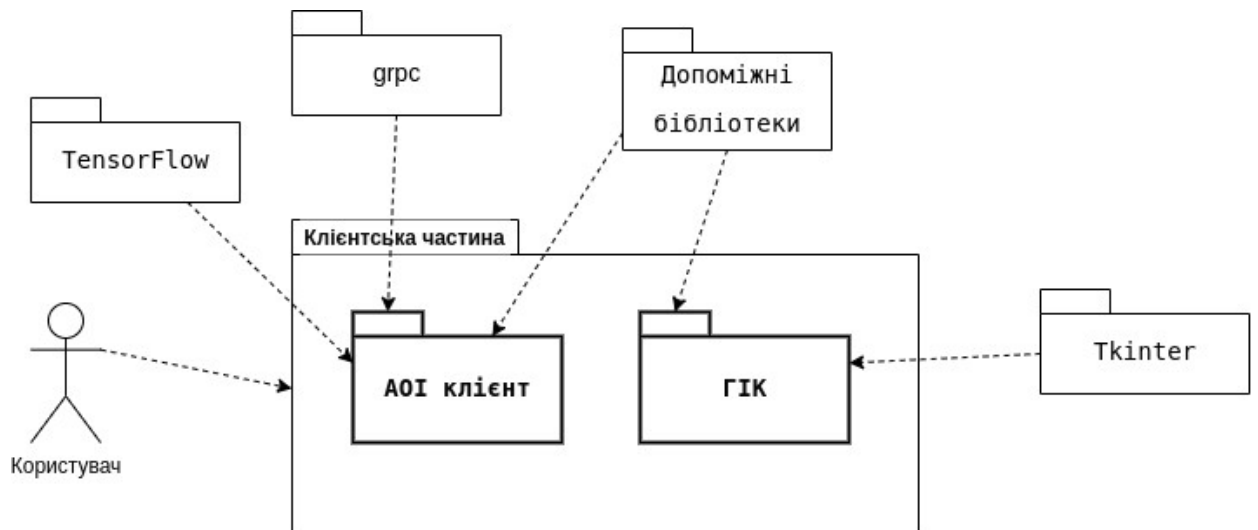


Рисунок 4.4 – Діаграма компонентів клієнтської частини системи

Діаграма компонентів серверної частини представлена на рисунку 4.5.

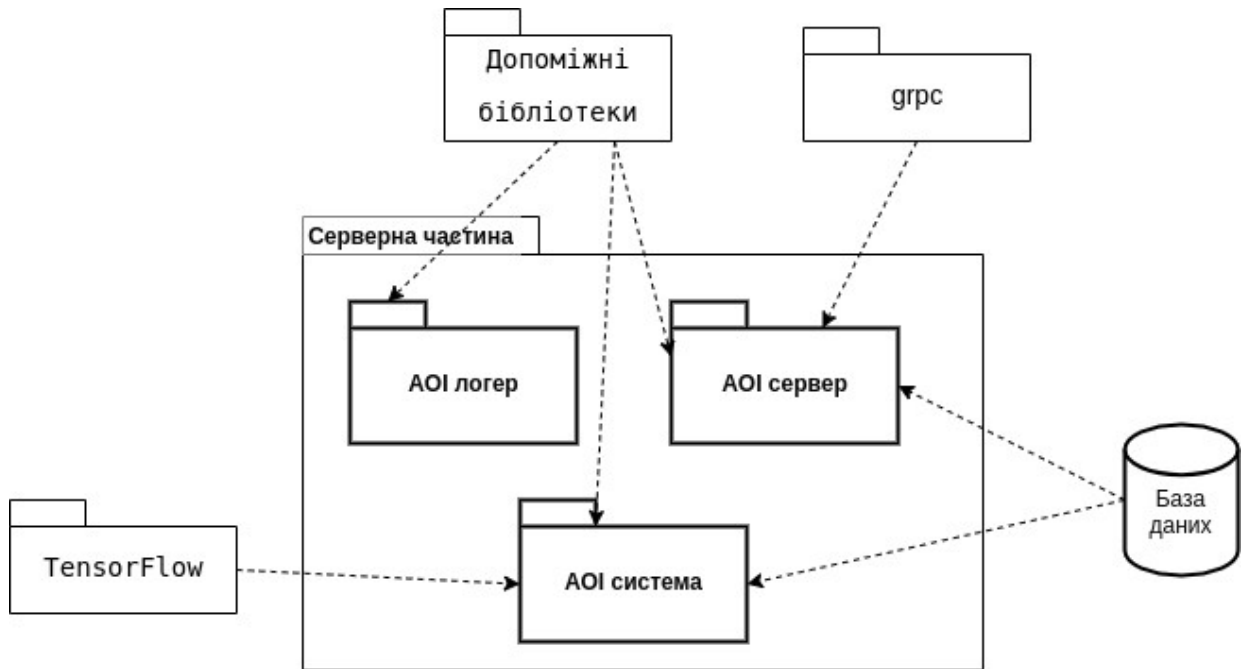


Рисунок 4.5 – Діаграма компонентів серверної частини системи

4.4 Опис інтерфейсу gRPC

Всі файли, що описують інтерфейс gRPC, який потім компілюється в цільову мову програмування, зберігаються в каталозі protos. Фізична структура підкаталогу protos наведена на рисунку 4.6. Він містить 4 файли. Усі файли описані нижче.



Рисунок 4.6 – Фізична структура підкаталогу protos

4.4.1 Типи значень

У таблиці 4.3 описані типи значень, які використовуються у gRPC, та відповідні їм значення у Python та C++ мовах програмування.

Таблиця 4.3 – Типи значень, які використовуються у gRPC

Тип proto	Опис	Python	C++
string	рядок завжди повинен містити кодований UTF-8 або 7-бітний текст ASCII	str/unicode	string
bool	має два можливих значення	boolean	bool
int32	використовує кодування змінної довжини, неефективно для кодування від’ємних чисел – якщо у вашому полі ймовірно є від’ємні значення, використовуйте замість цього sint32	int	int32
bytes	може містити будь-яку довільну послідовність байтів	str	string
float	представляє широкий динамічний діапазон числових значень за допомогою плаваючої точки	float	float

4.4.2 Опис файлу common.proto

Цей файл описує основні структурні одиниці, які використовуються як аргументи в сервісних методах, такі як context, shape, snapshot descriptor, subsample та sample meta.

Context складається з імен моделі й клієнта та використовується в таких структурах для передачі інформації: checkModelReq, checkModelResp, getSnapshotListReq, getModelReq, registerSubSamplesReq, registerSampleReq, getActiveLearningSampleInfoReq, getActiveLearningSampleReq.

Назва моделі потрібна для ідентифікації моделі, якій посилається запит.

Опис структури Context представлено в таблиці 4.4.

Таблиця 4.4 – Опис структури Context

Поле	Тип	Модифікатор
model_name	string	default
client_name	string	default

Shape включає в себе поле, у якому зберігаються розмірності об'єктів.

Shape використовується в таких структурах для реконструкції форми із запиту: SubSample, checkModelResp, registerSubSamplesReq, registerSampleReq та getActiveLearningSampleInfoResp.

Опис структури Shape представлено в таблиці 4.5.

Таблиця 4.5 – Опис структури Shape

Поле	Тип	Модифікатор
dim	int32	repeated

SnapshotDescriptor складається з назви моделі та відмітки часу.

SnapshotDescriptor використовується в структурі getSnapshotListResp для створення нового снапшоту.

Снапшот – це стан графіка в певний момент часу. У TensorFlow алгоритми машинного навчання представлені як обчислювальні графіки.

Назва моделі потрібна, щоб знати, до якої моделі застосовувати запит.

Відмітка часу зберігає час створення знімка.

Опис структури SnapshotDescriptor представлено в таблиці 4.6.

Таблиця 4.6 – Опис структури SnapshotDescriptor

Поле	Тип	Модифікатор
model_name	string	default
timestamp	string	default

SampleMeta складається з ключа і значення та використовується в структурах registerSampleReq й getActiveLearningSampleResp для реєстрації метаданих.

Ключ – це ключ у словнику, а значення – це значення, яке відповідає цьому ключу.

Опис структури SampleMeta представлений в таблиці 4.7.

Таблиця 4.7 – Опис структури SampleMeta

Поле	Тип	Модифікатор
key	string	default
value	string	default

SubSample складається з позицій запуску, зупинки та метаданих. Використовується в структурі registerSubSamplesReq.

Опис структури SubSample подано в таблиці 4.8.

Таблиця 4.8 – Опис структури SubSample

Поле	Тип	Модифікатор
start_pos	Shape	default
stop_pos	Shape	default
meta_info	SampleMeta	repeated

4.4.3 Опис файлу client.proto

Цей файл описує аргументи для роботи з моделями нейронних мереж, даними та навчанням.

Структура checkModelReq складається з Context та використовується в функції checkModel як тип запиту.

Опис структури checkModelReq представлено в таблиці 4.9.

Таблиця 4.9 – Опис структури checkModelReq

Поле	Тип	Модифікатор
ctx	Context	default

Структура checkModelResp складається з таких полів: контексту, форми, прапор наявності моделі у системі та кількості зроблених кроків. Вона використовується у функції checkModel як тип відповіді.

Поле наявності моделі у системі вказує, чи вже існує така модель у системі.

Опис структури checkModelResp представлено в таблиці 4.10.

Таблиця 4.10 – Опис структури checkModelResp

Поле	Тип	Модифікатор
ctx	Context	default
shape	Shape	Default
is_present	bool	default
steps_taken	int32	default

Структура getSnapshotListReq складається з контексту та використовується в функції getSnapshotList як тип запиту.

Опис структури getSnapshotListReq представлено в таблиці 4.11.

Таблиця 4.11 – Опис структури getSnapshotListReq

Поле	Тип	Модифікатор
ctx	Context	default

Структура getSnapshotListResp складається зі снапшотів та використовується у функції getSnapshotList як тип відповіді.

Опис структури getSnapshotListResp представлено в таблиці 4.12.

Таблиця 4.12 – Опис структури getSnapshotListResp

Поле	Тип	Модифікатор
snapshots	SnapshotDescriptor	repeated

Структура getModelReq складається з контексту та відмітки часу. Вона використовується в функції getModel як тип запиту.

Відмітка часу потрібна для отримання контрольної точки з бази даних.

Опис структури getModelReq представлено в таблиці 4.13.

Таблиця 4.13 – Опис структури getModelReq

Поле	Тип	Модифікатор
ctx	Context	default
timestamp	string	default

Структура getModelChunk складається з фрагмента, даних вводу, виводу та мітки класу. Вона використовується в функції getModel як тип потоку відповіді.

Всі поля використовуються для збереження інформації з бази даних.

Опис структури getModelChunk представлено в таблиці 4.14.

Таблиця 4.14 – Опис структури getModelChunk

Поле	Тип	Модифікатор
chunk	bytes	default
input	string	default
output	string	default
class_labels	string	default

Структура feedSampleChunk складається з закодованих даних та використовується у функції feedSample як тип потоку запиту.

Кодовані дані – це інформація про вибірку в байтах.

Опис структури feedSampleChunk представлений в таблиці 4.15.

Таблиця 4.15 – Опис структури feedSampleChunk

Поле	Тип	Модифікатор
encoded_data	bytes	default

Структура feedSampleResp складається з ідентифікатора та використовується у функції feedSample як тип відповіді.

Ідентифікатор – це УУІД для ідентифікації інформації про зразок в комп'ютерній системі. Він генерується за допомогою модуля uuid python.

Опис структури feedSampleResp представлений в таблиці 4.16.

Таблиця 4.16 – Опис структури feedSampleResp

Поле	Тип	Модифікатор
identifier	string	default

Структура registerSubSamplesReq складається з контексту, ідентифікатора, форми та підвибірки. Вона використовується в функції registerSubSample як тип запиту.

Опис структури registerSubSamplesReq представлено в таблиці 4.17.

Таблиця 4.17 – Опис структури registerSubSamplesReq

Поле	Тип	Модифікатор
ctx	Context	default
identifier	string	default
shape	Shape	default
subs	SubSample	repeated

Структура registerSubSamplesResp містить повідомлення та використовується в функції registerSubSample як тип відповіді.

Опис структури registerSubSamplesResp представлено в таблиці 4.18.

Таблиця 4.18 – Опис структури registerSubSamplesResp

Поле	Тип	Модифікатор
message	string	default

Структура registerSampleReq складається з контексту, ідентифікатору, форми та мета інформації. Вона використовується в функції registerSample як тип запиту.

Ідентифікатор використовується для ідентифікації інформації про зразок.

Опис структури registerSampleReq представлено в таблиці 4.19.

Таблиця 4.19 – Опис структури registerSampleReq

Поле	Тип	Модифікатор
ctx	Context	default
identifier	string	default
shape	Shape	default
meta_info	SampleMeta	repeated

Структура registerSampleResp містить повідомлення та використовується у функції registerSample як тип відповіді.

Опис структури registerSampleResp представлено в таблиці 4.20.

Таблиця 4.20 – Опис структури registerSampleResp

Поле	Тип	Модифікатор
message	string	default

Структура getActiveLearningSampleInfoReq складається з контексту та використовується в функції getActiveLearningSampleInfo як тип запиту.

Опис структури getActiveLearningSampleInfoReq представлений у таблиці 4.21.

Таблиця 4.21 – Опис структури `getActiveLearningSampleInfoReq`

Поле	Тип	Модифікатор
ctx	Context	default

Структура `getActiveLearningSampleInfoResp` складається з форми, зразків у черзі та підтримує активні поля навчання. Вона використовується у функції `getActiveLearningSampleInfo` як тип відповіді.

Поле підтримки активного навчання вказує, чи підтримує модель активне навчання. Зразки в черзі – це ряд активних навчальних зразків.

Опис структури `getActiveLearningSampleInfoResp` представлено в таблиці 4.22.

Таблиця 4.22 – Опис структури `getActiveLearningSampleInfoResp`

Поле	Тип	Модифікатор
<code>supports_active_learning</code>	bool	default
<code>samples_in_queue</code>	int32	default
<code>shape</code>	Shape	default

Структура `getActiveLearningSampleReq` складається з контексту та використовується в функції `getActiveLearningSample` як тип запити.

Опис структури `getActiveLearningSampleReq` представлено в таблиці 4.23.

Таблиця 4.23 – Опис структури `getActiveLearningSampleReq`

Поле	Тип	Модифікатор
ctx	Context	default

Структура `getActiveLearningSampleResp` складається з фрагмента даних та зразків у черзі. Вона використовується в функції `getActiveLearningSample` як тип потоку відповіді.

Опис структури `getActiveLearningSampleResp` представлено в таблиці 4.24.

Таблиця 4.24 – Опис структури `getActiveLearningSampleResp`

Поле	Тип	Модифікатор
<code>chunk</code>	<code>bytes</code>	<code>default</code>
<code>samples_in_queue</code>	<code>SampleMeta</code>	<code>repeated</code>

4.4.4 Опис файлу `management.proto`

Цей файл описує необхідні аргументи для роботи зі службою `watchdog` та структурними підрозділами.

Структура `wd_ModelSamples` складається з назви моделі та кількості зразків. Вона використовується в функції `wd_getSamplesResp` для передачі інформації.

Назва моделі потрібна для ідентифікації моделі, до якої застосовується запит. Кількість зразків – це кількість зображень.

Опис структури `wd_ModelSamples` представлено в таблиці 4.25.

Таблиця 4.25 – Опис структури `wd_ModelSamples`

Поле	Тип	Модифікатор
<code>model_name</code>	<code>string</code>	<code>default</code>
<code>num_samples</code>	<code>int32</code>	<code>default</code>

Структура `wd_getSamplesReq` не має полів. Вона використовується в функції `wd_getSamplesResp` як тип запиту.

Структура `wd_getSamplesResp` складається з моделей та використовується у функції `wd_getSamplesResp` як тип відповіді.

Опис структури `wd_getSamplesResp` представлено в таблиці 4.26.

Таблиця 4.26 – Опис структури wd_getSamplesResp

Поле	Тип	Модифікатор
models	wd_ModelSamples	repeated

Структура wd_option складається з імені, типу, рядка, мінімального та максимального значень, параметрів лише для зчитування та опису. Вона використовується в функціях wd_getOptionsResp, wd_setOptionsReq для передачі параметрів.

Опис структури wd_option представлено в таблиці 4.27.

Таблиця 4.27 – Опис структури wd_option

Поле	Тип	Модифікатор
opt_name	string	default
opt_type	string	default
opt_string	string	default
opt_float	float	default
opt_int	int32	default
opt_float_min	float	default
opt_float_max	float	default
opt_int_min	int32	default
opt_int_max	int32	default
opt_read_only	int32	default
opt_description	string	default

Структура wd_getOptionsReq складається з назви моделі та використовується в функції wd_getOptions як тип запиту.

Опис структури wd_getOptionsReq представлено в таблиці 4.28.

Таблиця 4.28 – Опис структури wd_getOptionsReq

Поле	Тип	Модифікатор
model_name	string	default

Структура `wd_getOptionsResp` складається з назви моделі та параметрів. Вона використовується у функції `wd_getOptions` як тип відповіді.

Опис структури `wd_getOptionsResp` представлено в таблиці 4.29.

Таблиця 4.29 – Опис структури `wd_getOptionsResp`

Поле	Тип	Модифікатор
<code>model_name</code>	<code>string</code>	<code>default</code>
<code>options</code>	<code>wd_option</code>	<code>repeated</code>

Структура `wd_setOptionsReq` складається з назви моделі та параметрів. Вона використовується у функції `wd_setOptions` як тип запиту.

Опис структури `wd_setOptionsReq` представлено в таблиці 4.30.

Таблиця 4.30 – Опис структури `wd_setOptionsReq`

Поле	Тип	Модифікатор
<code>model_name</code>	<code>string</code>	<code>default</code>
<code>options</code>	<code>wd_option</code>	<code>repeated</code>

Структура `wd_getOptionsResp` не має полів та використовується у функції `wd_setOptions` як тип відповіді.

4.4.5 Опис методів з файлу `koi.proto`

Файл `koi.proto` описує сервіси, які визначають можливості gRPC для АОІ системи.

Функції у файлі згруповані за категоріями, визначеними варіантом використання (пасивний клієнт, активний клієнт, WatchDog).

Усі функції, пов'язані з пасивними клієнтами, лише зчитують дані із системи, але ніколи не вносять та не змінюють будь-які дані.

Усі функції, пов'язані з активними клієнтами, змінюють дані.

Усі функції, пов'язані з WatchDog, використовують служби для моніторингу подій файлової системи.

Метод `checkModel` перевіряє, чи є модель з зазначеним ім'ям вже у системі. Якщо так, тоді метод записує цю інформацію у поле, отримує клас з описом цієї моделі і надсилає всю інформацію у відповіді. Якщо ні, метод повертає значення за замовчуванням, вказуючи, що модель з тим самим іменем не представлена в системі.

Категорія: ППІ пасивного клієнта.

Тип запити: `checkModelReq`.

Тип відповіді: `checkModelResp`.

Метод `getSnapshotList` отримує список усіх знімків певної моделі. Метод підключається до бази даних, дістає всі контрольні точки для цієї моделі, обробляє їх і повертає у відповіді.

Категорія: ППІ пасивного клієнта.

Тип запити: `getSnapshotListReq`.

Тип відповіді: `getSnapshotListResp`.

Метод `getModel` завантажує снапшот моделі. Снапшот моделі визначається за назвою та відміткою часу. Метод підключається до бази даних, запитує конкретні контрольні точки для цієї моделі, отримує всі дані з контрольної точки та повертає їх у відповіді.

Категорія: ППІ пасивного клієнта.

Тип запити: `getModelReq`.

Тип відповіді: `getModelChunk` (потік).

Метод `feedSample` подає зразок до стеку зразків і повертає ідентифікатор відправнику. Зразок чекає в стеку, доки клієнт не зареєструє метаінформацію або додаткові зразки для нього. Метод створює ідентифікатор за допомогою модуля `uuid python`, кодує вхідні дані, записує їх в систему і повертає створений ідентифікатор.

Категорія: ППІ активного клієнта.

Тип запити: `feedSampleChunk` (потік).

Тип відповіді: `feedSampleResp`.

Метод `registerSample` слідує за запитом `feedSample` та реєструє метаінформацію для ідентифікатора. Метод отримує ідентифікатор із запиту, відновлює метаінформацію із запиту, відновлює зразок із попередньо надісланих даних, видаляє вихідні дані зразка, додає побудований зразок та його метаінформацію до стеку зразків і відповідає клієнту.

Категорія: ППІ активного клієнта.

Тип запиту: `registerSampleReq`.

Тип відповіді: `registerSampleResp`.

Метод `getActiveLearningSampleInfo` перевіряє, чи підтримує модель активне навчання та отримує наступний об'єкт. Метод будує об'єкт відповіді, отримує ім'я моделі. Якщо модель підтримує активне навчання, метод отримує кількість зразків активного навчання для цієї моделі, розмір та форму моделі та записує цю інформацію у відповідь. Якщо модель не підтримує активне навчання, метод записує інформацію у відповідь відповідно до цього випадку: не підтримує активне навчання, кількість зразків дорівнює нулю.

Категорія: ППІ активного клієнта.

Тип запиту: `getActiveLearningSampleInfoReq`.

Тип відповіді: `getActiveLearningSampleInfoResp`.

Метод `getActiveLearningSample` отримує активну навчальну вибірку, що відповідає моделі, оновлює зразок метаінформації, отримує дані вибірки та повертає всю цю інформацію у відповіді.

Категорія: ППІ активного клієнта.

Тип запиту: `getActiveLearningSampleReq`.

Тип відповіді: `getActiveLearningSampleResp` (потік).

Метод `wd_getSamples` отримує кількість зразків для кожної моделі, яка була завантажена в систему, і надсилає інформацію у відповіді.

Категорія: WatchDog ППІ.

Тип запиту: `wd_getSamplesReq`.

Тип відповіді: wd_getSamplesResp.

Метод wd_getOptions отримує параметри моделі, визначає їх тип, записує у змінні та повертається у відповіді.

Категорія: WatchDog ППІ.

Тип запиту: wd_getOptionsReq.

Тип відповіді: wd_getOptionsResp.

Метод wd_setOptions отримує параметри конкретної моделі і повертає їх у відповіді.

Категорія: WatchDog API.

Тип запиту: wd_setOptionsReq.

Тип відповіді: wd_setOptionsResp.

4.5 Опис даних

Вхідні дані зберігаються у базі та представляють собою зображення чипів розміром 100 пікселів на 100 пікселів. Усі зображення мають мітки, що помічають наявність дефекту на зображенні та тип чипу.

Приклад вхідних даних зображено на рисунку 4.7.

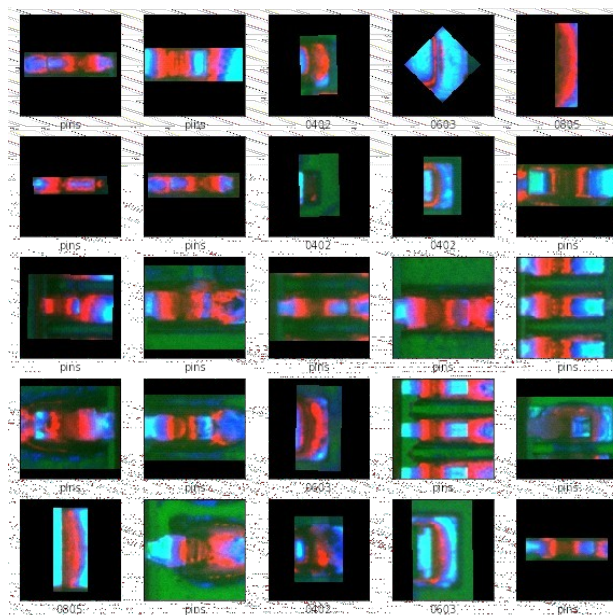


Рисунок 4.7 – Приклад вхідних даних

4.6 Висновки за розділом

У даному розділі було наведено фізичну структуру програми, описані каталоги, файли, структури та функції. Були наведені таблиці їх призначень.

Були описан повний ППІ, його синтаксис, аргументи, які він приймає, та тип значення, що повертає. Для написання функцій ППІ був використаний Protocols Buffer, то були використані притаманні для цієї мови модифікатори, що описують структури.

Під час програмної реалізації методів було використано допоміжні бібліотеки, модулі та функції, такі як `grpc`, `aiolog`, `tkinter`, `tensorflow` та інші.

Вхідні дані представляють собою зображення чипів 100*100 пікселів, а вихідними даними є результати діагностування наявності дефекту на зображенні.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис експлуатації програми

Перед початком роботи з програмою потрібно встановити на комп'ютер наступні пакети:

- python (^3.6);
- grpcio (^1.27.2);
- protobuf (^3.11.3);
- numpy (^1.18.2);
- tensorflow (1.14);
- tensorflow-datasets (1.3.1);
- grpcio-tools (^1.27.2);
- matplotlib (^3.2.1).

Щоб ініціалізувати серверну частину АОІ системи, потрібно запустити скрипт `main.py`. Сервер АОІ почне обслуговувати свій ППІ для клієнтів системи.

Розглянемо основний алгоритм запуску серверної частини.

Крок 1. Користувач запускає сервер та вказує параметри запуску.

Крок 2. Головний модуль системи оброблює передані параметри.

Крок 3. Запуск обробника ВВП.

Крок 4. Перевірка параметрів та файлу бази даних на коректність.

Крок 5. Створення об'єкту сервера АОІ.

Крок 6. Ініціалізація АОІ системи.

Крок 7. Запуск циклу перевірки зображень та збереження снапшотів.

Крок 8. Очікування переривання циклу перевірки.

Діаграма послідовності роботи програми представлена на рисунку 5.1. На ній зображено основні модулі програми та дії користувачів.

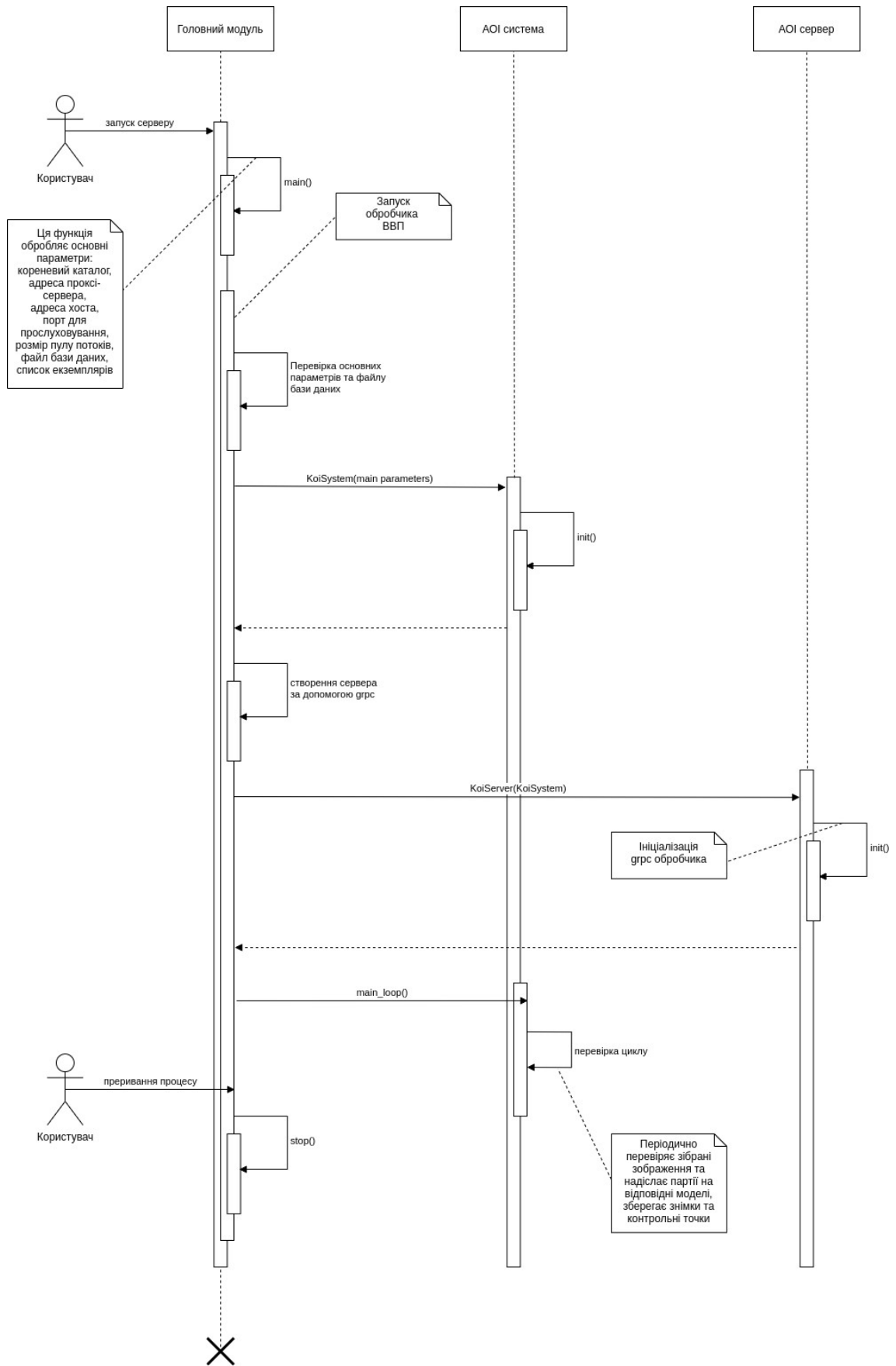


Рисунок 5.1 – Діаграма послідовності запуску програми

Сервер можна запустити з декількома параметрами (табл. 5.1).

Таблиця 5.1 – Доступні параметри сервера

Коротка команда	Повна команда	Опис	Значення за замовчуванням
-r	--root	кореневий каталог АОІ системи	каталог користувача
	--proxy	адреса проксі сервера	системний
-b	--bind	хост адреса, на якій буде запущено сервер	localhost
-p	--port	порт	50051
	--threadpool	кількість потоків	10
	--pond	файл бази даних	pond.db
	--inst	список екземплярів моделей	name1:script1; name2:script2

Якщо сервер успішно запустився, ви побачите повідомлення про успіх (рис. 5.2).

```
*****
Welcome to:
KoiSystem ver: 0.8.0
KoiServer ver: 0.6.0
*****
opening sqlite 'pond.db'...
loading model scripts...
found 2 model scripts.
building models...
ready for action
serving on port: 50051
```

Рисунок 5.2 – Успішний запуск сервера

Для демонстрації роботи ППП було створено прототип ГІК розробленої системи. Для запуску базового клієнта активного навчання системи АОІ

запустіть файл `active_learning.py`. Буде створено єдиний екземпляр `ActiveLearningApp`, який успадковується від класів `tkinter`.

Після запуску клієнта ви побачите вкладку “Classify” (рис. 5.3). Ця вкладка призначена для відображення завдань оператору. На вкладці знаходиться три кнопки, які дають оператору можливість класифікувати зображення: до 1 класу, до 2 класу, видалити.

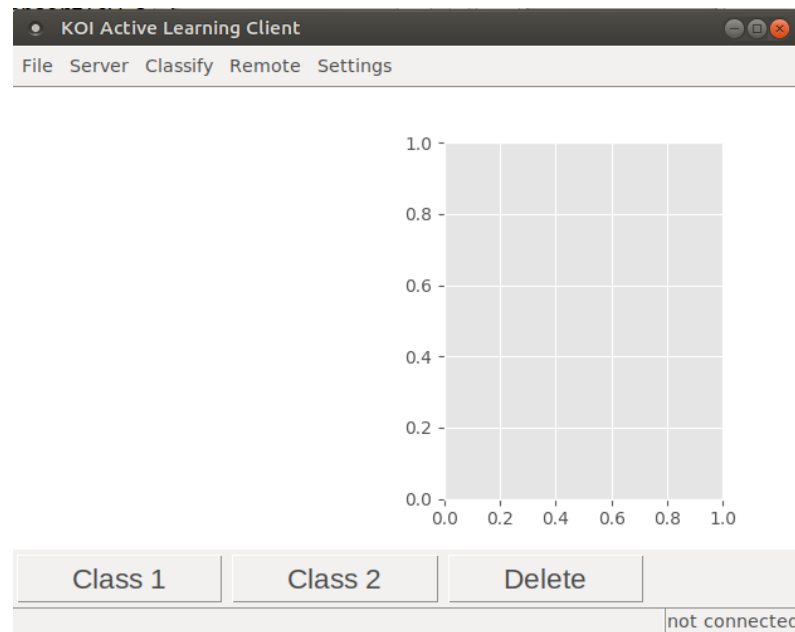


Рисунок 5.3 – Інтерфейс вкладки “Classify”

Усі поточні налаштування ви можете побачити на вкладці “Settings” (рис. 5.4). Ця вкладка призначена для зміни налаштувань сервера.

Вона дає можливість змінити адресу та порт сервера, вибрати модель, яку використовувати для навчання нейронних мереж, показує статус підключення сервера до клієнту.

Наступний блок дає можливість визначити швидкість оновлення налаштувань: автоматично, 60 сек, 10 сек, 1 сек.

Також у налаштуваннях можна визначити класи класифікації або видалити вже визначенні.

Дана клієнтська частина є прототипом – прикладом використання grps. Система дає можливість підключити усі методи, що були описані у розділі 4.

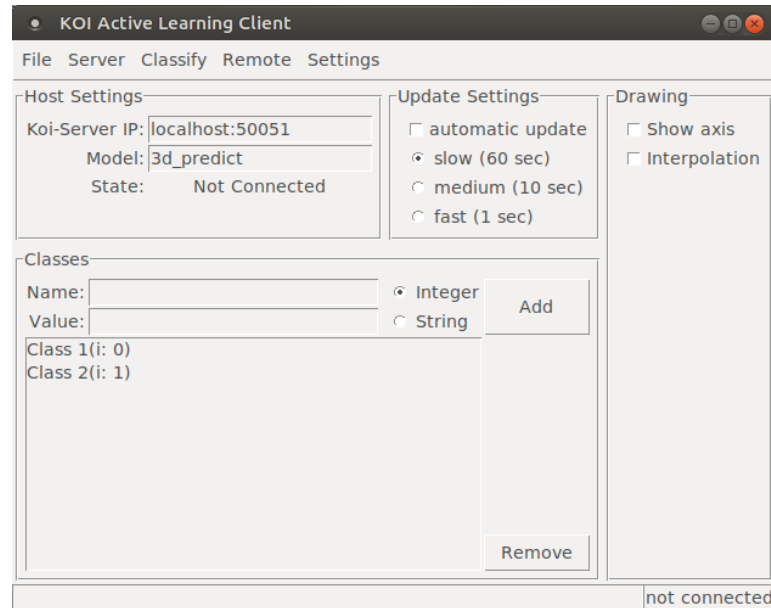


Рисунок 5.4 – Інтерфейс вкладки “Settings”

5.2 Опис тестових даних

Дослідження даних показало, що деякі зображення мають неправильну орієнтацію (рис. 5.5). Це спричинено неправильним програмуванням АОІ. Такі дані будуть погіршувати точність класифікації нейронної мережі.

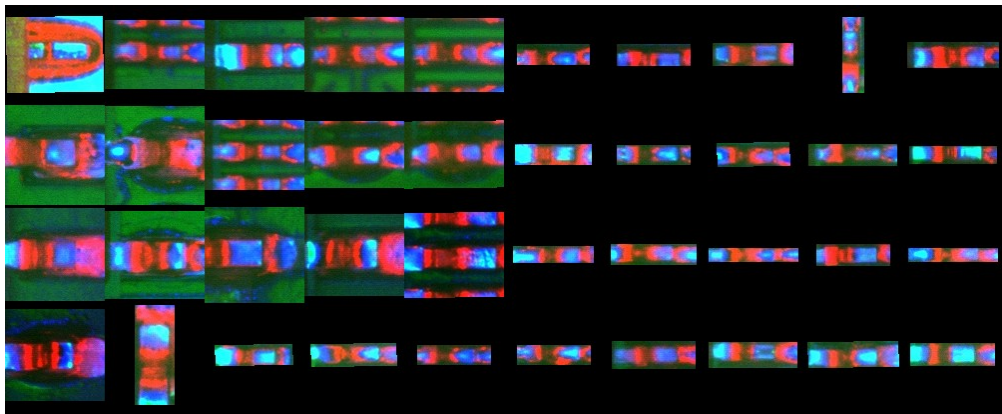


Рисунок 5.5 – Зображення з бази даних

Для поліпшення початкових даних був розроблений алгоритм аналізу зображень, описаний у розділі 2. Спочатку з усіх зображень видаляється чорний фон. Потім кожне зображення перевіряється на пропорції сторін. Кожне зображення, яке не має правильного співвідношення сторін, аналізується трьома різними методами: перетвореннями Фур'є, Хафа та об'єднанням даних. Якщо будь-який спосіб виявляє, що зображення неправильно орієнтоване, воно позначається для ручного контролю. Усі методи працюють на версії зображення у відтінках сірого, яка підкреслює червоний та синій кольорові канали зображення, і таким чином видаляє зелений фон друкованої плати.

Перетворення Фур'є працює з версією вихідного зображення у відтінках сірого, бере спектр потужності зображення і шукає найбільш значущу лінію в спектрі потужності, яка відповідає куту.

На рисунку 5.6 представлено результат перетворень Фур'є для зображення чипа під кутом 90° .

На рисунку 5.7 представлено результат перетворень Фур'є для зображення чипа під кутом 45° .

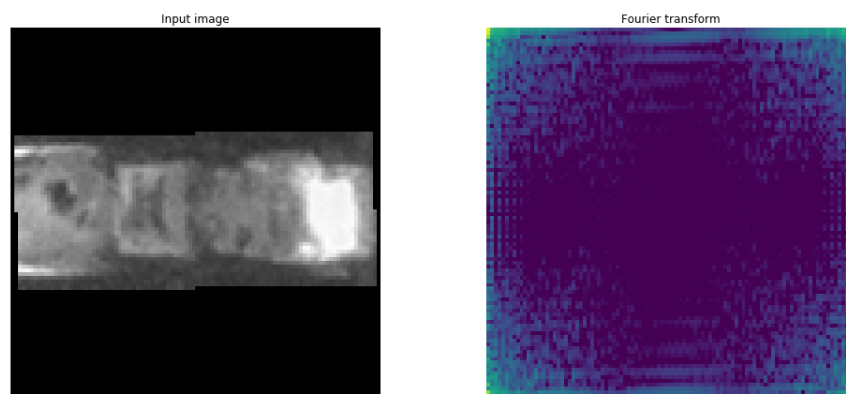


Рисунок 5.6 – Аналіз зображення під кутом 90° за допомогою перетворення Фур'є

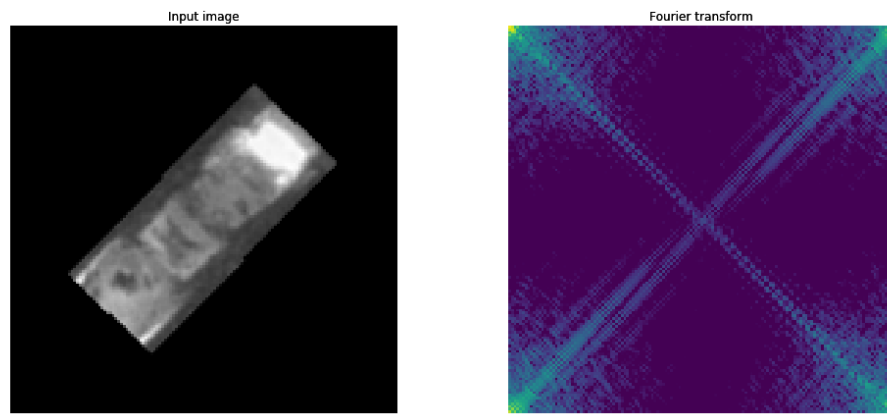


Рисунок 5.7 – Аналіз зображення під кутом 45° за допомогою перетворення Фур'є

Перетворення Хафа (рис. 5.8, 5.9) та об'єднання даних працюють з зображення у відтінках сірого без чорного фону та з зафарбованими залишками фону в кольори сусідніх пікселів.

Перетворення Хафа пов'язані з ідентифікацією прямих у зображенні. Цей метод визначає характеристики прямої не як рівняння з двома точками у просторі, а враховує кутовий коефіцієнт.

Метод об'єднання даних приймає порогові суми рядків і стовпців. Потім визначається, чи функція над горизонтальними лініями, чи функція над вертикальними лініями має вужчу ширину.

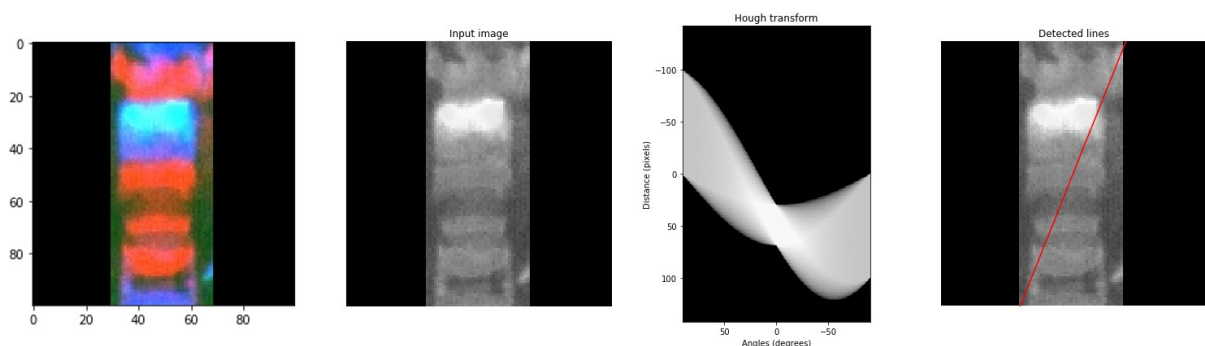


Рисунок 5.8 – Аналіз зображення під кутом 0° за допомогою перетворення Хафа

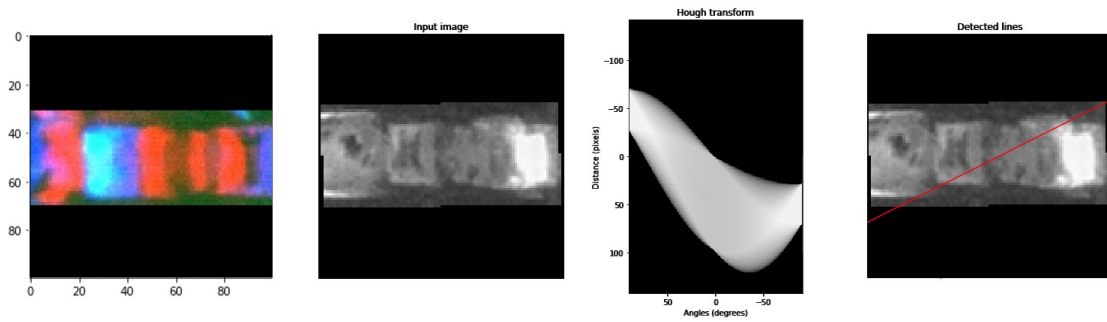


Рисунок 5.9 – Аналіз зображення під кутом 90° за допомогою перетворення Хафа

5.3 Експериментальні дослідження

Була розроблена нейронна мережа для класифікації зображень за типами ('0201', '0402', '0603', '0805', '1206', 'pins') та наявністю дефекту. Приклад зображень з навчальної вибірки представлено на рисунку 5.10.

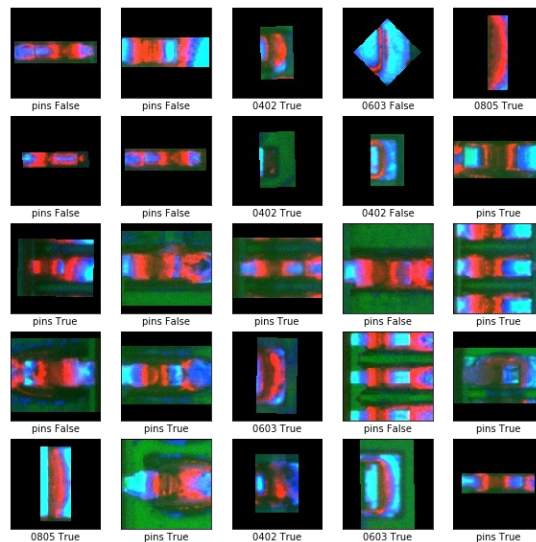


Рисунок 5.10 – Приклад зображень з навчальної вибірки

Навчальна вибірка складала 230787 зображення. Кількість шарів нейронної мережі – п'ять. Кількість ступенів навчання – п'ять. Отримана точність після навчання – 96%. Результати навчання зображені на рисунку 5.11. Тестування моделі на незнайомих даних показало 95% точності класифікування.

Точність моделі склала 94%, що є дуже високим результатом, тому було вирішено перевірити причину такої високої точності. Такий високий результат є прийнятним при класифікації картинок за типами, але при класифікації кожного типу на дефекти він є підозрілим, тому що зображення одного типу не повинні сильно відрізнятися між собою.

```

Train on 230787 samples
Epoch 1/5
230787/230787 [=====] - 144s 624us/sample - loss: 0.2884 - accuracy: 0.8855
Epoch 2/5
230787/230787 [=====] - 141s 610us/sample - loss: 0.1753 - accuracy: 0.9343
Epoch 3/5
230787/230787 [=====] - 141s 611us/sample - loss: 0.1395 - accuracy: 0.9480
Epoch 4/5
230787/230787 [=====] - 141s 610us/sample - loss: 0.1159 - accuracy: 0.9565
Epoch 5/5
230787/230787 [=====] - 141s 610us/sample - loss: 0.1005 - accuracy: 0.9627

```

Рисунок 5.11 – Результати навчання нейронної мережі

Так як початкова вибірка є незбалансованою, то в якості оцінки точності моделі була використана матриця помилок (confusion matrix), яка дозволяє подивитися відсоток помилок по кожному класу окремо (рис. 5.12). Точність класифікації складає 81-99%, що залежить від кількості тестових даних у початковій вибірці.

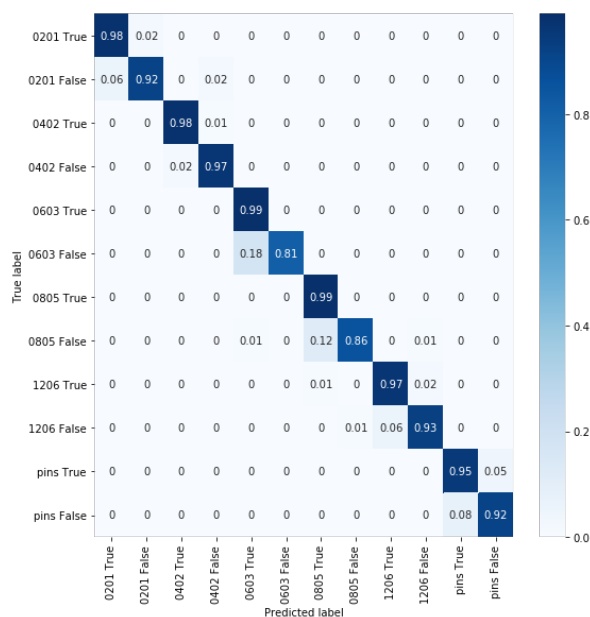


Рисунок 5.12 – Матриця помилок класифікації наявності дефекту та типу чипа

На рисунку 5.13 представлена візуалізація результатів класифікування.

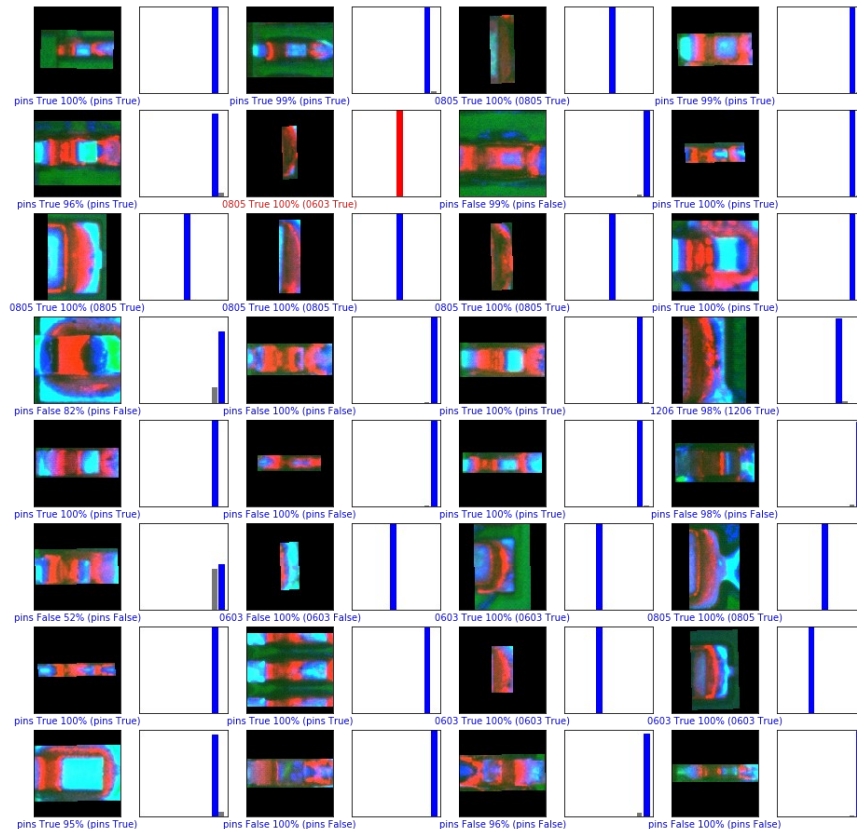


Рисунок 5.13 – Візуалізація помилок класифікації наявності дефекту та типу чипа

Також була розроблена нейронна мережа для класифікації зображень з дефектами на класи: реальний дефект або псевдодефект. На рисунку 5.14 представлено декілька зображень з початкової вибірки з позначенням класу зображення.

Так як початкова вибірка є незбалансованою, то в якості оцінки точності моделі була використана матриця помилок (confusion matrix), яка дозволяє подивитися відсоток помилок по кожному класу окремо (рис. 5.15). З матриці видно, що нейронна мережа схильна класифікувати картинки в клас "псевдодефект". Це можливо через велику кількість зображень цього типу в початковій вибірці. Точність класифікації за наявністю реального дефекту складає 0,71-100%, що залежить від кількості тестових даних у початковій вибірці.

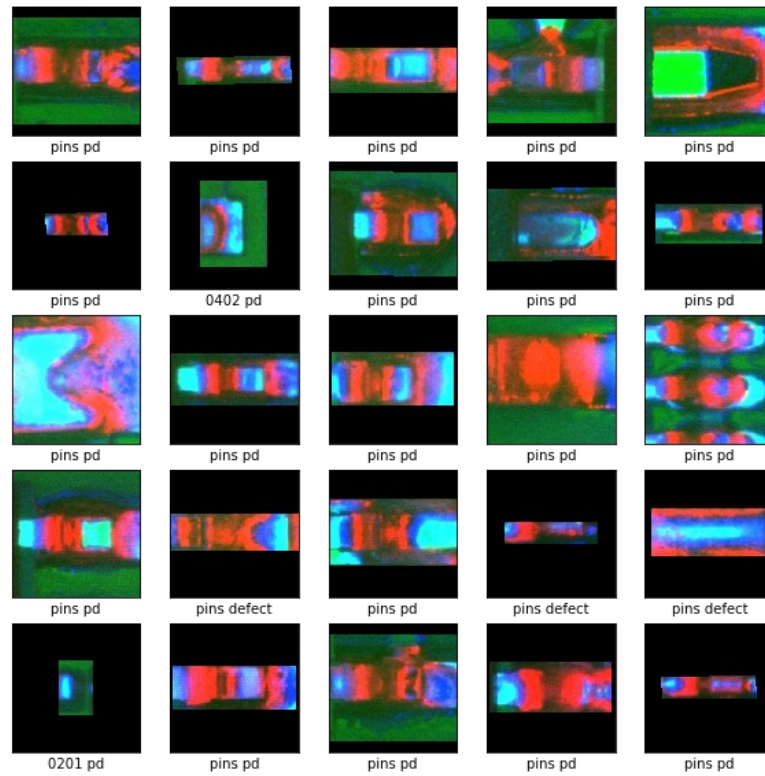


Рисунок 5.14 – Приклад зображень з навчальної вибірки

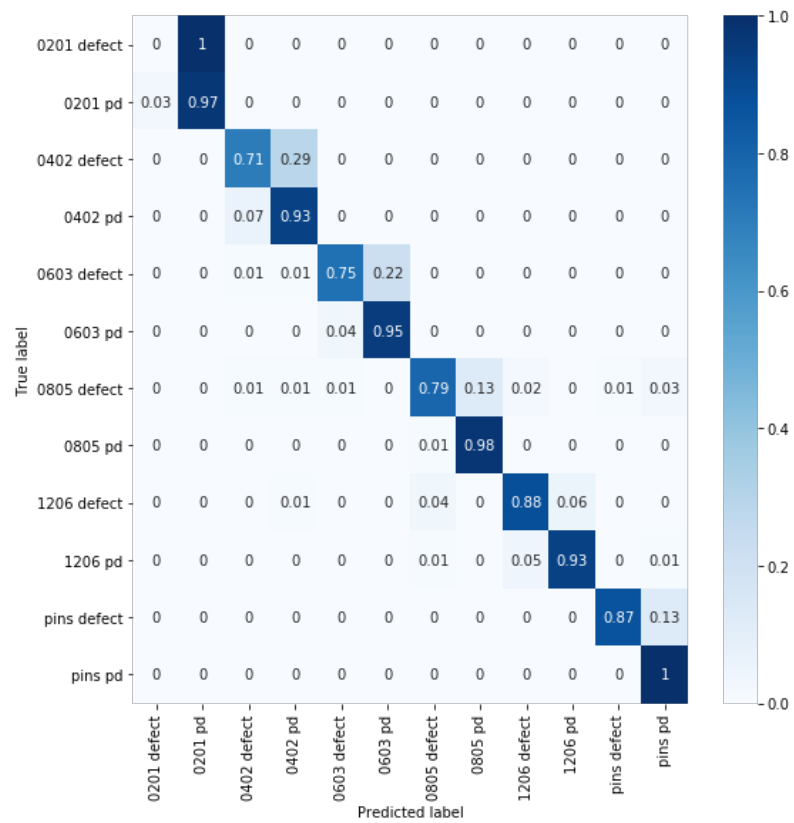


Рисунок 5.15 – Матриця помилок класифікації справжнього дефекту та псевдодефекту

Для подальшого аналізу результатів були проаналізовані маркування зображень в залежності від того, який оператор промаркував це зображення. Це дасть можливість оцінити вплив людського фактору на класифікацію нейронною мережею. На рисунках 5.16 - 5.17 представлено результати аналізу. Синій стовпець показує кількість зображень, що були опрацьовані оператором та правильно класифіковані мережею, а червоний стовпець показує кількість зображень, що нейронна мережа не змогла правильно класифікувати.

Результати показують, що рішення усіх операторів є приблизно однаковими, тож можна виключити людський фактор з можливих бар'єрів для високої точності класифікації.

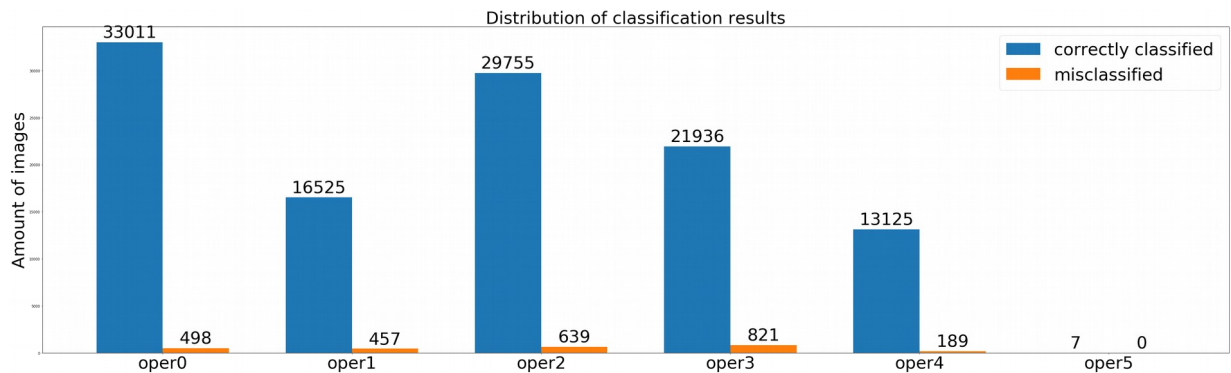


Рисунок 5.16 – Аналіз рішень оператора у абсолютних величинах

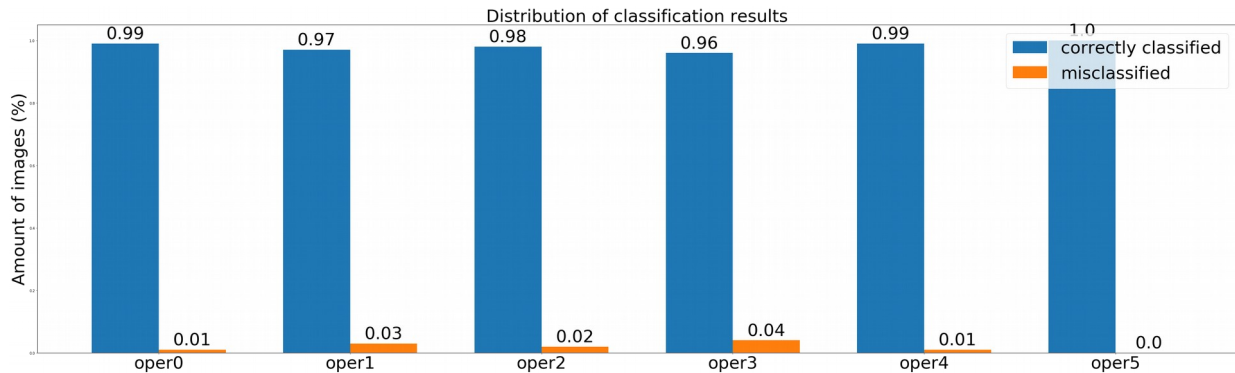


Рисунок 5.17 – Аналіз рішень оператора у відносних величинах

Для підтвердження точності класифікації була розроблена нейронна мережа для кластеризації кожного типу зображень в залежності від наявності дефекту. Результати кластеризації представлені на рисунках 5.18 та 5.19.

Можна зробити висновок, що чипи першого типу будуть класифікуватися краще, тому що вони мають більш виражені ознаки та добре поділилися на кластери.

Результати кластеризації добре показують різницю між чипами з дефектами та без: чипи без дефектів мають більше червоного кольору на зображенні у першому типі та більше блакитного кольору у другому типі. Таким чином, можна зробити висновок, що чипи і дійсно будуть класифікуватися нейронними мережами з високою точністю.

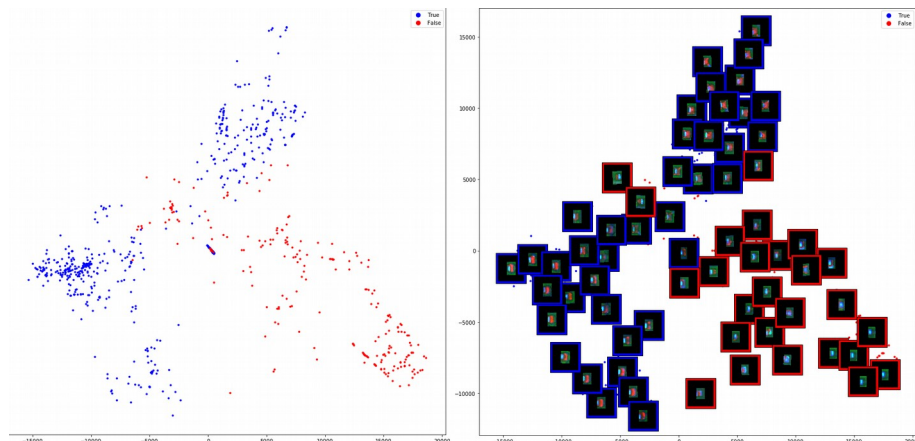


Рисунок 5.18 – Результати кластеризації першого типу чипів

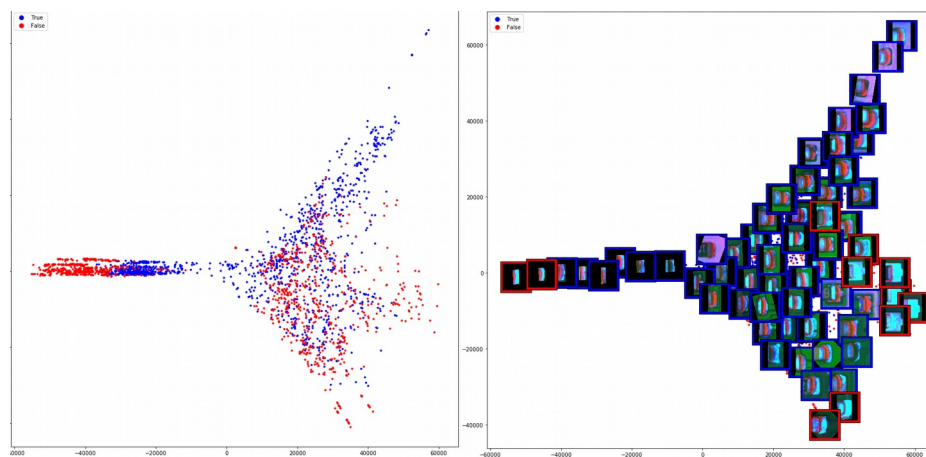


Рисунок 5.19 – Результати кластеризації другого типу чипів

5.4 Подальші покращення системи

Для системи АОІ друкованих план була розроблена серверна частина та прототип клієнтської частини. Серверна частина реалізує інтерфейс для створення моделей нейронних мереж для класифікування дефектів на платах. В даній галузі є доцільним використання активного навчання нейронних мереж, тому що галузь специфічна та потребує допомоги оператора у класифікації окремих чипів.

Таким чином, подальший етап в розробці системи – це реалізація ГІК та ППІ для активного навчання нейронних мереж.

5.5 Висновки за розділом

У розділі було наведено інструкцію по запуску серверної та клієнтської частин системи, описані доступні параметри запуску та доступні вкладки ГІКу. Були наведені рисунки успішного запуску серверної частини.

Було проведено навчання нейронної мережі на даних для навчання, що склали 80% від загального обсягу даних. Навчання проходило у п'ять кроків з одночасним визначенням точності навчання. Кожен крок навчальна та вибірка для оцінки результатів змінювалися. Після навчання було проведено тестування створеної моделі на тестовій вибірці. Дані з цієї вибірки були для навченої моделі зовсім незнайомі. Точність класифікації склала 95%.

Отримані результати були перевірені декількома методами: матриця помилок, аналіз маркування операторів, кластеризація початкових зображень.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У атестаційній роботі магістра розроблена та реалізована технологія, що дозволяє діагностувати та класифікувати дефекти друкованих плат на етапі виробництва з високим ступенем достовірності класифікації за рахунок використання нейронних мереж.

В даному розділі приводиться економічне обґрунтування створення програмного забезпечення, оцінка затрат на його створення, розрахунок економічної ефективності та терміну окупності [34].

Наявні аналоги таких систем для перевірки друкованих плат мають такі недоліки: низький показник достовірності виявлення дефектів, висока складність користування програмою, низька продуктивність, малоінформативні результати роботи. Розроблена технологія дозволить підвищити швидкодію алгоритму виявлення та класифікації дефектів на платах та суттєво покращить точність результату.

Передбачається, що розроблена технологія буде розповсюджуватися комплектами з вартістю 80000,00 грн за комплект. За результатами дослідження ринку, було виявлено 95 компаній, що можуть бути зацікавлені у придбанні розробленого комплекту. Очікується, що приблизно 10% з цих компаній придбає комплект, що дозволить продати 9 комплектів.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного забезпечення можна поділити на етапи. Для кожного етапу вказується трудомісткість, кількість виконавців та тривалість робіт. У розробці приймають участь програміст впродовж 2 місяців і консультант впродовж 0,2 місяця. Розробка програми починається 14 вересня 2020 року та має бути завершена до 13 листопада 2020 року. Тривалість робіт розраховується за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – довжина циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, люд.

Отримана інформація зведена в таблиці 6.1.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування роботи	Трудомісткість		Виконавці		Тривалість
	людино-днів	% до підсумку	спеціальність	кількість осіб	днів
1	2	3	4	5	6
1. Дослідження вимог до проєктованого продукту	2	4,2	програміст	2	1
			консультант		
2. Розробка технічного завдання	2	4,2	програміст	2	1
			консультант		
3. Проєктування загальної структури програми	4	8,3	програміст	2	2
			консультант		
4. Розробка схеми функціонування програми	3	6,2	програміст	1	3
5. Створення програмного коду	26	54,2	програміст	1	26
6. Тестування та налагодження програмного продукту	7	14,6	програміст	1	7
7. Складання програмної документації	4	8,3	програміст	1	4
Всього	48	100,0			44

За даними таблиці 6.1 був зведений стрічковий графік планування розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення термінів початку

виконання всі види роботи, а навпроти – календарний період їх виконання. Даний графік наведено в таблиці 6.2.

Необхідні початкові дані:

- час на розробку – 2 міс.;
- консультант працює до 0,2 міс. (4 дн.) із заробітною платою 9500 грн./міс;
- програміст працює 2 міс. (44 дн.) із заробітною платою 25000 грн./міс.

Останнім етапом планування розробки програмного продукту є звідний стрічковий графік розробки програмного продукту – лінійний графік Ганта (табл. 6.2), побудований за даними таблиці 6.1.

Таблиця 6.2 – Лінійний графік Ганта розробки програмного продукту

Найменування роботи	Календарний період, дні									
	14.09 - 18.09	21.09 - 25.09	28.09 - 02.10	05.10 - 09.10	12.10 - 16.10	19.10 - 23.10	26.10 - 30.10	02.11 - 06.11	09.11 - 13.11	
1. Дослідження вимог до проєктованого продукту	■									
2. Розробка технічного завдання	■									
3. Проєктування загальної структури програми	■	■								
4. Розробка схеми функціонування програми		■	■							
5. Створення програмного коду			■	■	■	■	■	■	■	
6. Тестування та налагодження програмного продукту								■	■	■
7. Складання програмної документації									■	■

Позначення: ■ – програміст; ■ – консультант.

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає наступні статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок;
- витрати на спеціальне обладнання;
- матеріали і комплектувальні вироби;
- накладні витрати;
- податки.

6.2.1 Розрахунок основної заробітної плати

Витрати по цій статті складаються з планового фонду заробітної плати усіх категорій працівників, зайнятих в розробці програмного продукту. Розрахунок заробітної плати ведеться на основі відомостей про трудомісткість (табл. 6.1). Результати розрахунків зведені в таблиці 6.3.

Таблиця 6.3 – Розрахунок основної заробітної плати

Посада виконавця	Кіль- кість, осіб	Місячний оклад, грн	Кіль- кість, міс. роботи	Премії та доплати, грн.	Середня місячна зарплата, грн.	Сумарна зарплата, грн.
Програміст	1	25000,00	2	—	25000,00	50000,00
Консультант	1	9500,00	0,2	—	9500,00	1900,00
Всього	2					51900,00

6.2.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати. Розрахунок основної та додаткової заробітної плати наведений в таблиці 6.4.

Таблиця 6.4 – Розрахунок додаткової заробітної плати

Посада виконавця	Додаткова заробітна плата, грн.	Сума основної та додаткової заробітної плати, грн.
Програміст	5000,00	55000,00
Консультант	190,00	2090,00
Всього	5190,00	57090,00

6.2.3 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок визначають у відсотковому відношенні від суми основної і додаткової зарплат, з урахуванням премій і доплат. Премії і доплати в нашому випадку дорівнюють 0,00 грн. Відрахування на соціальне страхування складають 22%. Сума відрахувань обчислюється за формулою (6.2):

$$ВІД = (ЗП_{осн} + ЗП_{дод}) \cdot 0,22, \quad (6.2)$$

де ВІД – сума відрахувань;

$ЗП_{осн}$ – основна заробітна плата, грн;

$ЗП_{дод}$ – додаткова заробітна плата, грн.

Отримуємо результат:

$$ВІД = (51900 + 5190) \cdot 0,22 = 12559,8 \text{ грн.}$$

6.2.4 Визначення витрат на матеріали

Матеріали, які знадобилися для розробки даної програми представлені в таблиці 6.5.

Таблиця 6.5 – Витратні матеріали

Матеріал	Кількість	Вартість, грн.
Папір формату А4	500 листів	100,00
CD-RW диск	1 шт.	10,00
Картридж для принтера	1 шт.	450,00

Витрати на матеріали розраховуються за формулою (6.3):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{\text{т.з.}}) - C_{io} \cdot N_{io}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектувальні вироби, грн.;

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати (0,03 - 0,1);

C_i – ціна i -го найменування матеріалу, напівфабрикату, комплектувального, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті, комплектувальному;

C_{io} – ціна поворотних відходів i -го найменування матеріалу, грн.;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів, комплектувальних.

Розрахуємо витрати на матеріали:

$$C_{io}=0; N_{io}=0; K_{m.з.}=0,05 ,$$

$$M=(1+0,05) \cdot (100+10+450)=588 \text{ грн.}$$

Разом витрати на матеріали склали 588 грн.

6.2.5 Витрати на спеціальне устаткування

У цю статтю входять витрати на оплату машинного часу ЕОМ. Витрати на оплату машинного часу ЕОМ для написання і відладки цієї програми визначають по формулі (6.4):

$$C_{mo} = P_{екс} \cdot t_{mo} , \quad (6.4)$$

де C_{mo} – витрати на оплату машинного часу, грн.;

$P_{екс}$ – експлуатаційні витрати на одну годину машинного часу, грн.;

t – машинний час ЕОМ для написання і відладки цього програмного продукту, ч.

Для розрахунку експлуатаційних витрат на одну годину машинного часу складемо кошторис витрат на зміст і експлуатацію цього устаткування.

Амортизаційні відрахування визначають по формулі (6.5).

$$A = \Phi_6 \cdot \frac{H_a}{100} , \quad (6.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки.

Балансова вартість обчислювальної техніки становить 20000,00 грн., а норма амортизаційних відрахувань – 25%. Таким чином амортизаційні відрахування становлять 5000,00 грн.

Витрати на електроенергію визначаються по формулою (6.6):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зп} \cdot Ц_e, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, ч.;

$K_{зч}$ – середній коефіцієнт завантаження за часом;

$K_{зп}$ – коефіцієнт завантаження по потужності;

$Ц_e$ – ціна одного кВт·год електроенергії, грн.

Номінальна потужність робочої машини – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,80 грн. Розраховуємо витрати:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,8 = 544,32 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{обсл} = ФЗП_p \cdot (1 + K_{від}) \cdot \frac{t_{обсл}}{\Phi_{ef.обсл}}, \quad (6.7)$$

де $ФЗП_p$ – річний фонд зарплати обслуговуючих працівників, грн.;

$K_{від}$ – коефіцієнт відрахувань на соціальне страхування і до інших фондів, %;

$t_{обсл}$ – час впродовж року, необхідний на техобслуговування ЕОМ, год/рік;

$\Phi_{ef.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік.

Машину обслуговує один працівник з основною заробітною платою 5100,00 грн./міс. Річний ефективний фонд робочого часу обслуговуючого машину працівника дорівнює 1750 год/рік. На обслуговування однієї машини витрачається по 1 годині на місяць, що в рік становить 12 годин. Річний фонд зарплати обслуговуючих працівників розраховується за формулою (6.7):

$$\begin{aligned} \Phi ЗП_p &= 5100 \cdot 12 = 61200 \text{ грн} , \\ K_{\text{від}} &= 0,22 ; \Phi_{\text{еф.обсл}} = 12 \text{ год/рік} ; t_{\text{обсл}} = 1750 \text{ год} , \\ ЗП_{\text{обсл}} &= 61200 \cdot (1 + 0,22) \cdot \frac{12}{1750} = 511,98 \text{ грн} . \end{aligned}$$

Сума витрат по статті "Поточний ремонт устаткування" дорівнює 3% від балансової вартості устаткування, що дорівнює 600,00 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Інші витрати дорівнюють $6656,3 \cdot 0,05 = 332,82$ грн.

Зведемо отримані дані в таблиці 6.6.

Таблиця 6.6 – Кошторис витрат на зміст і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація устаткування	5000,00
Експлуатація устаткування	544,32
Зарплата обслуговуючого персоналу з відрахуваннями	511,98
Поточний ремонт устаткування	600,00
Інші витрати	332,82
Всього	6989,12

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{\text{мо}} = B_{\text{екс}} t_{\text{мо}} , \quad (6.8)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн;

$V_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино-год;

$t_{\text{мо}}$ – машинний час цифровий ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію устаткування (ЕОМ)» (табл. 6.6) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу розраховуються:

$$V_{\text{екс}} = \frac{6989,12}{1800} = 3,88 \text{ грн/маш. год.}$$

Розробка і відладка програмного продукту займає 352 годин машинного часу (44 дні по 8 робочих годин), тобто $t_{\text{мо}} = 352$ годин.

Витрати на оплату машинного часу розраховуються:

$$C_{\text{мо}} = 3,88 \cdot 352 = 1365,76 \text{ грн.}$$

6.2.6 Накладні витрати

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і так далі), зміст і експлуатацію будинків. Накладні витрати включаються у вартість розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати складають 40% до основної заробітної плати розробників,

що складає 20760,00 грн. Результати калькуляції кошторисної вартості робіт по розробці програми і ціна програмного продукту наведені у таблиці 6.7.

Таблиця 6.7 – Калькуляція кошторисної вартості робіт по розробці програми і ціна програмного продукту

№	Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
1	Основна заробітна плата	51900,00	56,19
2	Додаткова заробітна плата	5190,00	5,62
3	Соціальні відрахування	12559,80	13,60
4	Витрати на матеріали	588,00	0,64
5	Витрати на оплату машинного часу	1369,28	1,48
6	Накладні витрати	20760,00	22,47
7	Разом витрати	92367,08	100,00
8	Прибуток (30%)	27710,12	—
9	Всього	120077,20	—
10	ПДВ (20%)	24015,44	—
11	Всього	144092,64	—

Таким чином, вартість проекту оцінюється в 144092,64 грн.

6.3 Оцінка економічної ефективності

Планується продаж 15 комплектів розробленого програмного забезпечення по 115000,00 гривень за комплект. Таким чином очікується отримати надходження від продажів 1725000,00 гривень. З урахуванням ризику 50% надходження від продажів складуть 862500,00 гривень, що більше вартості розробки програмного забезпечення, в якій вже була врахована рентабельність 30 відсотків.

Таким чином, розробка даного програмного забезпечення є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

В дипломній кваліфікаційній роботі магістра проводиться дослідження та програмна реалізація програми діагностування друкованих плат.

Охорона праці – це система законодавчих актів, соціально-економічних, організаційних, технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, що забезпечують безпеку збереження здоров'я та працездатності людини в процесі праці.

Основою охорони праці є системний та послідовний підхід до дослідження організації праці з точки зору безпеки, людино-машинної взаємодії, аналізу різних фізичних, хімічних, психологічних, соціальних чинників та їх вплив на організм людини.

Розробка програмного забезпечення в більшості сферах ведеться в офісному приміщенні, в якому людину оточують обчислювальні пристрої, системи зв'язку, генератори струму та інші електричні прибори, що створюють електромагнітне випромінювання. Незважаючи на розвиток технічних засобів у сфері виробництва та підходів до забезпечення якості, безпеки роботи працівників – місце праці залишається небезпечним та шкідливим для здоров'я людини.

При роботі користувача постійно або періодично діють наступні небезпечні і шкідливі чинники, що пов'язані з:

- підвищеним навантаженням на зоровий аналізатор – тривала робота з об'єктами малого розміру;
- інтелектуальним навантаженням – необхідність швидкого прийняття рішень, творча діяльність, постійне сприйняття і оцінка нової інформації, високий ступінь складності завдання;
- емоційним навантаженням – ступінь відповідальності за завдання для виконання, дефіцит часу, значимість помилки;

- монотонністю трудового процесу – багаторазове повторення одноманітних дій, тривалість зосередженого спостереження;
- гіподинамією – тривалим перебуванням оператора в одній позі без активних рухів.

Для створення сприятливих умов роботи з комп'ютером необхідна відповідність нормативно-правовим актам з охорони праці, які регламентують показники стану навколишнього середовища:

- достатню площу і об'єм приміщення, де встановлено комп'ютер;
- показники мікроклімату робочого приміщення (температура повітря, відносна вологість, та ін.);
- параметри світлового середовища робочого місця і робочої поверхні (наявність природного і штучного освітлення, освітленість екрана монітора, освітленість горизонтальної поверхні столу, відсутність прямого і відбитого сліпучого відблиску джерел світла, відсутність пульсації освітленості);
- рівень шуму і вібрації робочих місць в межах норм;
- відсутність в повітрі робочої зони шкідливих речовин.

Нижче розглядаються заходи з охорони праці, виробничої санітарії, гігієни праці та пожежної безпеки при експлуатації ЕОМ з візуальними дисплейними терміналами (ВДТ) в будівлях і приміщеннях.

7.2 Заходи з забезпечення безпеки праці

Забезпечення безпеки праці, організації оптимальних умов праці, а також захисту навколишнього середовища в умовах роботи з персональним комп'ютером (ПК) досягається шляхом дотримання норм і проведення заходів, передбачених законодавчими актами України.

ПК підключаються до мережі з напругою 220 В і частотою 50 Гц. Згідно ПУЕ приміщення відноситься до 1-ї категорії «Приміщення без підвищеної небезпеки», оскільки існує можливість торкання корпусів апаратури й батареї центрального опалення [35]. У приміщенні повинні бути

відсутні відкриті струмоведучі частини з напругою більше 12В (послідовний інтерфейс RS-232C). При виконанні правил ПУЕ враження електричним струмом можливо тільки у випадку несправності апаратури й живильних кабелів [35].

Апаратура, що перебуває під небезпечною для життя напругою, припускає забезпечення електробезпеки. Всі металеві частини обчислювальної техніки повинні бути заземлені на третій провід спеціальних триконтактних розеток, призначених для включення ПК. Струмопровідні кабелі повинні бути розташовані під дерев'яною підлогою, зверху покритою лінолеумом, або у вигляді схованої проводки в стінах, таким чином, контакт із кабелем виключається. Над кожною розеткою повинен бути попереджувальний напис «220 В». Індикаторні лампи в блоках ЕОМ сигналізують про включення блоку в мережу.

Електрична ізоляція забезпечується конструкціями пристроїв і способами їхнього підключення. У процесі експлуатації необхідно періодично робити перевірку якості ізоляції, шляхом виміру її опору. Відповідно до ПУЕ в електроустановках до 1000 В мінімальне значення опору ізоляції силових і освітлювальних електропроводок на ділянці між суміжними запобіжниками або за останніми запобіжниками між будь-яким проведенням і землею становить не менше 0,5 Ом [35].

Відповідно до ПУЕ захисне заземлення застосовується у всіх електроустановках змінного струму під напругою 380 В і вище й постійного струму 440 В і вище [36]. У приміщеннях з підвищеною небезпекою заземлення використовується в електроустановках змінного струму напругою 42 В та постійного струму 110 В. Захисному заземленню підлягають корпуси трансформаторів, оболонки кабелів, металеві огороження електроустановок відповідно до ГОСТ 12.1.030-81 «Электробезопасность. Защитное заземление, зануление» [36].

Вирівнювання потенціалів досягається шляхом устрою контурних заземлювачів. У офісних приміщеннях вирівнювання потенціалів

здійснюється з'єднанням металевих стоек ЕОМ, корпусів апаратів світильників і іншого устаткування з усіма доступними для дотику металевими конструкціями будинку, спорудження, а також з контуром-шиною захисного заземлення.

Електричний поділ мережі припускає розділення мережі на окремі електрично не пов'язані між собою ділянки за допомогою розділових трансформаторів. Підвищення електробезпеки відбувається за рахунок зменшення ємності мережі й підвищення опору ізоляції фаз мережі щодо землі.

Подвійна ізоляція забезпечує застосування крім основної, робочої ізоляції струмоведучих частин ще одного шару додаткової ізоляції, що ізолює людину від металевих неструмоведучих частин, які можуть випадково виявитися під напругою. При застосуванні електроустаткування з подвійною ізоляцією не потрібно ні заземлення, ні занулення їхніх корпусів.

В офісних приміщеннях розрядні струми статичної електрики найчастіше виникають при дотику до кожного з елементів ЕОМ. Такі розряди небезпеки для людини не представляють, але крім неприємних відчуттів вони можуть привести до виходу з ладу ЕОМ. Для зниження величини зарядів статичної електрики, що виникають, в офісному приміщенні покриття технологічних підлог варто виконувати з одношарового полівінілхлоридного антистатичного лінолеуму. Іншим методом захисту є нейтралізація заряду статичної електрики іонізованим газом. До загальних мір захисту від статичної електрики в офісі можна віднести загальне й місцеве зволоження повітря.

Внаслідок роботи за персональним комп'ютером на фізіологію людини негативно впливають електромагнітні випромінювання. Щоб зменшити наслідки впливу на людину та знизити негативні показники у робочій зоні до допустимих значень, згідно з ГОСТ 12.2.007.0–75 «Изделия электротехнические. Общие требования безопасности», вироби, які створюють електромагнітні поля, повинні мати захисні елементи [36].

Вимоги до захисних елементів повинні бути вказані в стандартах та технічних умовах на конкретні види виробів. Згідно з НПАОП 0.00–1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» [37] та ДсанПіН 3.3.2.007–98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ЕОМ» [38] рекомендовано використовувати монітор типу Samsung SyncMaster 959 NF який не потребує додаткових захисних елементів при умові, що монітор працює в режимі 120 Гц.

У розділі 4 ДсанПіН 3.3.2.007-98 передбачені гігієнічні вимоги до організації устаткування робочих місць ВДТ ЕОМ і ПЭВМ [38].

Конструкція робочого стола повинна відповідати сучасним вимогам ергономіки й забезпечувати оптимальне розміщення на робочій поверхні використовуваного устаткування (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого стола із ВДТ повинні регулюватися в межах 680...800 мм, а ширина й глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600...1400 мм, глибина – 800..1000 мм). Приймаємо робочу поверхню з розмірами 1400 на 800. Щоб уникнути відблисків на екрані вікон доцільно розмістити робоче місце торцем до вікна не ближче чим на 1 м. Площа дозволяє нам розмістити додаткові меблі для зберігання документів, магнітних дисків, полками, стелажми, тумбами та ін.

Велике значення надається ергономічним та естетичним вимогам. Цього домоглися за допомогою колірної рішення приміщення й устаткування з урахуванням вимог ГОСТ 17.2.032-78 «Рабочее место при выполнении работ сидя» – у приміщенні стіни пофарбовані фарбами пастельних тонів [37].

Під час організації робочого простору необхідно враховувати індивідуальні антропометричні параметри користувача з відповідними допусками на можливі зміни робочих поз та потребу у переміщеннях.

Площа на одного працівника регламентується СніП 2.09.04-87 «Административные и бытовые здания» [37] і становить не менше 6 м^2 . Значить у приміщенні довжиною $A = 6 \text{ м}$, шириною $B = 4 \text{ м}$, висотою $H = 3,75 \text{ м}$ загальною площею $S = 24 \text{ м}^2$ можна розмістити не більше 4-х робочих місць. Крім площі приміщення береться в увагу й обсяг приміщення, що становить не менше 20 м^3 на одне робоче місце, що передбачено ДсанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ЕОМ» [37]. Наше приміщення має об'єм в 90 м^3 , а це означає, що в нашому приміщенні можна розмістити чотири робочих місця із ВДТ ЕОМ.

Приміщення розташоване на першому поверсі виробничого корпусу, що припустимо й відповідає перерахованим вище правилам і нормам.

Серед користувачів ПК виявлено новий тип захворювання – синдром комп'ютерного стресу, який проявляється головним болем, запаленням слизової оболонки очей, підвищеною дратівливістю, млявістю і депресією.

Згідно з НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» необхідно комплексно вирішувати проблеми ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку [37].

По-перше, для збереження здоров'я працівників, необхідно дотримуватися всіх вимог до приміщень, де відбувається робота з комп'ютерами.

По-друге, необхідно подбати про правильне розміщення робочих місць як щодо безпосередньо кімнати, так і щодо всієї будівлі підприємства.

По-третє, слід забезпечити робітників сучасними комп'ютерними меблями, які були б зручні у використанні і не доставляли почуття дискомфорту та були пристосовані до особливостей людського організму.

Необхідно прагнути забезпечення всіх працівників сучасною комп'ютерною технікою (рідкокristалічні дисплеї, які можна назвати майже «зеленими» пристроями, безпечними для здоров'я людей; використання

різноманітних сучасних ергономічних клавіатур). В іншому випадку необхідно забезпечити техніку спеціальними захисними пристроями (наприклад, захисний фільтр).

Крім того, повинен бути встановлений правильний режим праці та відпочинку

Час перерв протягом робочого дня для 8-годинної зміни розподіляється наступним чином:

- для I категорії (до 20 000 тис. друкованих знаків) – 2 перерви по 15 хв. Через 2 години після початку зміни і після обідньої перерви;
- для II категорії (до 40 000 тис. друкованих знаків) – через 2 години після початку зміни і через 1,5-2 години після обідньої перерви по 15 хв. Або 10 хв., через кожну годину роботи;
- для III категорії (до 60 000 тис. друкованих знаків) – через 1,5-2 години після початку зміни і через 1,5-2 години після обідньої перерви по 20 хв., або по 15 хв. через кожну годину.

При 12-годинній зміні перерви в перші 8 годин такі ж, як і при 8-годинній зміні, протягом останніх 4 годин, незалежно від категорії і виду робіт, – кожну годину по 15 хв.

Не рекомендується працювати за ПК більше 2 годин підряд без перерви, У процесі роботи по можливості, щоб зменшити негативний вплив монотонності, слід змінювати тип і зміст діяльності.

Дотримання цих заходів призводить до того, що підвищується продуктивність праці, скорочується кількість професійних захворювань, крім це працівники знаходяться завжди в гарному настрої і не скаржаться на погане самопочуття.

Заходи мінімізації негативного впливу випромінювань ВДТ на органи зору. Відеотермінальний пристрій повинен відповідати наступним технічним вимогам:

- яскравість свічення екрана повинна бути не менше 100 кд/м²;

- мінімальний розмір світної точки – не більше 0,4 мм для монохромного дисплея і не більше 0,6 мм – для кольорового дисплея;
- контрастність зображення знаку – не менше 0,8;
- частота регенерації зображення при роботі з позитивним контрастом у режимі обробки тексту – не менше 72 Гц;
- кількість точок на рядку – не менше 640;
- низькочастотне тремтіння зображення в діапазоні 0,05-10 Гц повинно знаходитися в межах 0,1 мм;
- екран повинен мати покриття антивідблиску;
- розмір екрану повинен бути не менше 31 см по діагоналі, а висота символів на екрані – не менше 3,8 мм, при цьому відстань від очей оператора до екрану повинна бути в межах 40-80 см.

7.3 Заходи з забезпечення виробничої санітарії та гігієни праці

Для зниження шкідливого впливу комп'ютера на людину повинні дотримуватися певні вимоги до умов роботи, до робочого місця й крім того, дотримуватися режим роботи на комп'ютері.

Об'ємно-планувальні рішення будинків і приміщень для роботи з візуальними дисплейними терміналами (ВДТ) ЕОМ повинні відповідати вимогам ДсанПіН 3.3.2.007-98 «Державним санітарним правилам і нормам роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Розміщення робочих місць із ВДТ ЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено [37].

В офісному приміщенні дотримуються норми ДБН В.2.5-28-2006 «Природне та штучне освітлення» відносно штучного освітлення: освітленість робочих місць у горизонтальній площині на рівні 0,8 м від підлоги не менше 400 лк, вертикальна освітленість у площині екранів дисплея – не менше 300 лк, коефіцієнт пульсації освітленості – не більше

10%. Забезпечуються таке освітлення розташованими на стелі світильниками ЛСП з люмінесцентними лампами типу ЛБ-60 [38].

Віконні прорізи приміщень для роботи із ВДТ обладнані регульованими пристроями.

Оптимальні норми температури повітря в робочій зоні виробничих приміщень у холодний і перехідний періоди року при II категорії робіт згідно ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень – 20...23°C, відносна вологість – 60...40%, швидкість руху повітря – 0.2 м/сек. У теплий період року температура – 22...25°C, відносна вологість – 60...40%, швидкість руху повітря – 0.2 м/сек [39].

Для забезпечення вищевказаних параметрів передбачене застосування вентиляції – природної (аерації й провітрювання), механічної (загальобмінної, місцевої приточної і витяжної), кондиціювання повітря.

У приміщенні забезпечений приплив свіжого повітря, кількість якого визначається техніко-економічним розрахунком і вибором схеми системи вентиляції. Мінімальна витрата повітря визначається з розрахунку 50...60 м³/ч на одну людину, але не менш дворазового повітрообміну в годину.

Згідно з ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [39] зниження шуму, створюваного на робочих місцях внутрішніми джерелами, а також шуму, що проникає ззовні, є дуже важливим завданням. Зниження шуму в джерелі виникнення забезпечується застосуванням пружних прокладок між основою машини, приладу й опорною поверхнею. Як прокладки використовуються гума, повсть, пробка, різної конструкції амортизатори. Під настільні апарати можна підкладати м'які килимки із синтетичних матеріалів, а під ніжки столів, на яких вони встановлені, – прокладки з м'якої гуми, повсті, товщиною 6-8 мм.

Звукоізоляція огорожувальних конструкцій приміщення із ВДТ забезпечує параметри шуму, які відповідають вимогам ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [40].

Усі перелічені вище заходи проведені з метою поліпшення умов праці програміста, і, отже, сприяють підвищенню ефективності праці.

7.4 Заходи з пожежної безпеки

Заходи щодо протипожежного захисту розділяються на організаційні, експлуатаційні, технічні й режимні (спеціальні).

Організаційні заходи: навчання робітників та службовців правилам пожежної безпеки, організація пожежної охорони, проведення бесід, лекцій, видання необхідних інструкцій, плакатів і т.п.

Технічні заходи передбачають дотримання протипожежних правил і норм при устрої систем опалення, вентиляції, кондиціюванні повітря, блисковкозахисті при спорудженні будинків, установці технологічного устаткування й ін.

Експлуатаційні заходи передбачають правильну експлуатацію систем опалення, вентиляції й кондиціонування повітря, блисковкозахисту технологічних машин і устаткування, правильне утримання будинків і територій і т.п.

Режимні заходи передбачають заборону або обмеження застосування відкритого вогню в вогнебезпечних місцях, паління в невстановлених місцях, обов'язкове дотримання норм і правил при роботі з вогнебезпечними й вибухонебезпечними речовинами.

Категорія приміщення за вибухопожежобезпекою відноситься до категорії Д.

Одним з найбільш важливих завдань пожежної профілактики є захист будівельних конструкцій від руйнування й забезпечення їхньої достатньої міцності в умовах впливу високих температур при пожежі. Залежно від меж вогнестійкості будівельних конструкцій ДБН В.2.1.1-2002 «Протипожежні норми» установлені вісім ступенів вогнестійкості будинків: I, II, III, IIIa, IIIб, IV, IVa, V [41].

Враховуючи високу вартість електронного обладнання, а також категорію їх пожежної безпеки, будівлі повинні мати ступінь вогнестійкості не нижче III. Тому для виготовлення будівельних конструкцій використовують, як правило, цегла, залізобетон, скло, метал та інші негорючі матеріали. Застосування дерева – обмежено, а в разі використання його необхідно просочувати вогнезахисним составом – антипіреном або покривати вапняно-цементною, азбестоцементною, гіпсовою штукатуркою. Для запобігання розповсюдження вогню під час пожежі з однієї частини будівлі в іншу влаштовують протипожежні перешкоди у вигляді протипожежних стін, перегородок, перекриття, тощо.

Пожежна профілактика – це комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, запобігання пожежі, обмеження його розповсюдження, а також на створення умов для успішного гасіння пожежі.

Протидимний захист будівель, спрямований на незадимлюваність евакуаційних шляхів, окремих приміщень і видалення продуктів горіння в певному напрямку, є першорядним завданням пожежної профілактики.

Для виявлення початкової стадії загоряння, швидкого й точного оповіщення служби пожежної охорони про час і місце виникнення пожежі встановлені системи автоматичної пожежної сигналізації (АПС). Вони можуть самостійно пускати в хід установки пожежогасіння, коли пожежа не досягла великих розмірів. Системи АПС складаються з пожежних сповіщувачів, ліній зв'язку й прийомних пультав (станцій).

Для подачі сигналу про пожежу встановлені пожежні сповіщувачі. Залежно від контролюваного фактору, вони діляться на теплові, димові, світлові, комбіновані.

До первинних засобів гасіння пожеж, призначених для локалізації невеликих загорянь, належать пожежні цівки, внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри й т.п.

Для гасіння вогню у випадку пожежі в приміщенні площею 24 м² повинно бути 2 вуглекислотних вогнегасники ВВК-3,5, можливо також використання порошкових вогнегасників ВП-3.

На випадок пожежі передбачені шляхи евакуації робітників: проходи, проїзди, евакуаційні виходи.

Евакуаційні виходи розташовуються розосереджено, а їх кількість – не менше двох із будівлі. При кількості робітників у зміні до 50 чоловік передбачений другий (запасний вихід) на зовнішні сходи.

Коридори й проходи, призначені для евакуації, мають найменшу довжину й мінімальну кількість поворотів. На всьому протязі проходу немає порогів або проміжних щаблів.

Важливу роль у забезпеченні безпечного виходу людей грає протидимовий захист евакуаційних ходів. У будинках висотою до 9 поверхів незадимлюваність сходових кліток на час евакуації досягається їхньою ізоляцією від підвалів, горищ і поверхів. Для цього влаштовані відособлені входи в підвали, вхід на сходову клітку з поверхів здійснюють через тамбур – шлюз із підпором повітря, відокремлюють горища від сходових кліток перекриттями з негорючих матеріалів.

У таблиці 7.1 наведені відстані від найбільш віддаленого робочого місця до найближчого евакуаційного виходу.

Для відводу надлишкової теплоти служать системи вентиляції й кондиціонування повітря. Однак вони становлять додаткову пожежну небезпеку, тому що, з одного боку, вони забезпечують подачу кисню – окислювача в усі приміщення, а з іншого боку – при виникненні пожежі поширюють вогонь і продукти горіння по всіх приміщеннях і установкам, з якими зв'язані повітроводи.

Для тепло- і звуковбирної ізоляції систем вентиляції застосовують такі негорючі матеріали, як мати з мінеральної вати, скловолокна, стиліту.

Напруга до електроустановок подається по кабельних лініях, які становлять особливу пожежну небезпеку.

Одним з найбільш важливих завдань пожежної профілактики є захист будівельних конструкцій від руйнування й забезпечення їхньої достатньої міцності в умовах впливу високих температур при пожежі.

Підпільні простори під знімними підлогами розділяють неспаленими перегородками з межею вогнестійкості не менш 0,75 год. на відсіки площею не більше 250 м², комунікації через які прокладають у спеціальних обоймах із застосуванням негорючих ущільнювальних матеріалів [42].

Таблиця 7.1 – Відстані від найбільш віддаленого робочого місця до найближчого евакуаційного виходу

Об'єм приміщення, тис. м ³	Ступінь вогнестійкості будівлі	Відстань, м, при щільності людського потоку в загальному проході, люд/м ²		
		до 1	від 1 до 3	від 3 до 5
Незалежно від об'єму	I, II, III	Не обмежується		
	IV	160	95	65
	V	120	70	50

7.5 Заходи з забезпечення безпеки у надзвичайних ситуаціях

Порядок ліквідації наслідків застосування біологічних засобів.

Застосування біологічних засобів відноситься до надзвичайних ситуацій техногенного походження.

При ураженні території та населення біологічними засобами, проводиться комплекс рятувальних та інших невідкладних робіт. Сутність рятувальних та інших невідкладних робіт – це усунення безпосередньої загрози життю та здоров'ю людей, відновлення життєзабезпечення населення, запобігання або значне зменшення матеріальних збитків.

В осередку біологічного ураження організовуються та проводяться:

– бактеріологічна розвідка та індикація бактеріальних засобів;

- карантинний режим або обсервація у відповідності з рішенням старшого начальника;
- санітарна експертиза;
- контроль зараження продовольства, харчової сировини, води та фуражу, їх знезараження;
- протиепідемічні; санітарно-гігієнічні, спеціальні профілактичні, лікувально-евакуаційні, протиепізоотичні, ветеринарно-санітарні заходи, а також санітарно-роз'яснювальна робота.

В осередку біологічного ураження роботи здійснюються за рішенням старшого начальника цивільної охорони. Роботами щодо ліквідації бактеріологічного осередку керує начальник цивільної охорони об'єкта, а організацією та проведенням медичних заходів – начальник медичної служби.

При організації робіт щодо ліквідації осередку бактеріологічного (біологічного) ураження враховуються:

- здатність бактеріальних засобів спричиняти інфекційні захворювання серед людей і тварин;
- здатність деяких мікробів і токсинів тривалий час зберігатися у зовнішньому середовищі;
- наявність та тривалість інкубаційного періоду виявлення хвороби;
- складність лабораторного виявлення застосованого збудника та тривалість визначення його виду;
- небезпечність зараження особового складу формувань та необхідність застосування засобів індивідуального захисту.

У випадку виявлення ознак застосування бактеріальних засобів у район негайно висилається бактеріологічна розвідка. На підставі отриманих даних, встановлюється зона карантину або зона обсервації, намічається обсяг та послідовність проведення заходів, а також порядок використання сил та засобів для ліквідації осередку біологічного ураження. Карантинний режим встановлюють з метою недопущення розповсюдження інфекційних

захворювань за межі осередку. Ізоляційно-обмежуючі міри при обсервації менш суворі, чим при карантині.

У всіх випадках в осередку біологічного ураження одного із першочергових заходів проведення профілактичного лікування населення від особливо небезпечних інфекційних хвороб. Для цього застосовуються антибіотики широкого спектру дії та інші препарати, що забезпечують профілактичний і лікувальний ефект, а також препарати, що є в індивідуальній аптечці АІ-2.

Після того, як буде визначено вид збудника, проводиться екстрена профілактика – застосування специфічних для даного захворювання препаратів: антибіотиків, сировоток та ін., своєчасне застосування яких зменшить кількість жертв і буде сприяти прискореній ліквідації осередку ураження.

Для проведення заходів з ліквідації осередку бактеріологічного ураження залучаються в першу чергу, сили і засоби, які опинилися на території осередку, в тому числі санітарно-епідемічні станції (СЕС), ветеринарні станції, пересувні протиепідемічні загони, спеціалізовані протиепідемічні бригади, лікарні, поліклініки та інші медичні і ветеринарні установи та формування.

Якщо цих сил і засобів недостатньо, то залучаються сили і засоби медичної та інших служб цивільної охорони, які знаходяться за межами осередку. Формування загального призначення залучаються для виявлення хворих та підозрілих на захворювання та їх ізоляцію, проведення знезараження території, будівель і споруд, санітарної обробки людей, дезінфекції одягу. В зоні карантину здійснюється суворий контроль за дотриманням населенням установленого режиму поведінки, виконують інші заходи.

Інфекційних хворих госпіталізують і лікують в інфекційних лікарнях в осередку ураження, або розгортають тимчасові інфекційні стаціонари. Якщо

необхідно, то хворих з особливо небезпечними інфекціями евакуюють спеціальними групами.

Осередок біологічного ураження вважається ліквідованим після того, як з моменту виявлення останнього хворого пройде час, що дорівнює максимальному терміну інкубаційного періоду для даного типу захворювання [43]-[44].

7.6 Висновки з розділу

Під час виконання розділу «Охорона праці та безпека у надзвичайних ситуаціях» був проведений аналіз потенційних небезпек на робочому місці, в робочій зоні і у виробничому середовищі відповідно до виду робіт, що розглядаються в проєкті.

На основі аналізу потенційних небезпек були обрані найбільш ефективні заходи, спрямовані на виключення або мінімізацію негативного впливу певних потенційно небезпечних і шкідливих факторів виробничого процесу.

Були розроблені заходи щодо забезпечення виробничої санітарії та гігієни праці, виробничих факторів санітарно-гігієнічного характеру, розроблені поліпшувальні заходи, проведені необхідні розрахунки.

Здійснено розробку заходів щодо пожежної безпеки спрямованих на збереження життя персоналу, обладнання, матеріальних ресурсів в умовах виникнення загорянь, організації ліквідації вогнищ загорянь, евакуації персоналу.

Також були розроблені заходи безпеки в надзвичайних ситуаціях техногенного характеру.

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи магістра було вирішено завдання дослідження та програмної реалізації методів АОІ на основі нейронних мереж для вирішення задачі діагностування друкованих плат.

Були розглянуті та проаналізовані такі види нейронних мереж:

- багат шаровий перцептрон;
- глибинні нейронні мережі;
- згорткові нейронні мережі;
- автокодувальник.

Програма була створена на мові програмування Python у середовищі розробки Jupyter Notebook з використанням допоміжних бібліотек, таких як tensorflow, grpc та matplotlib.

Було створено та програмно реалізовано алгоритм для аналізу початкових даних (зображень), який використовує перетворення Фур'є, перетворення Хафа та метод об'єднання даних.

Даний алгоритм дозволяє попередньо проаналізувати початкові зображення та виявити ті, що значно відрізняються від решти. Якість початкових даних має значний вплив на точність отриманих результатів.

Було розроблено систему для реалізації АОІ друкованих плат з використанням нейронних мереж. Система була протестована та експериментально досліджена. Результати проведеного дослідження показали, що згорткові нейронні мережі класифікують дані з точністю 94%.

Розроблена система допоможе підвищити швидкість та точність діагностування, надасть можливість виявити дефекти виробництва на початку.

Практична цінність роботи полягає у тому, що було створено програму для діагностування друкованих плат на етапі виробництва, що дозволить підвищити якість виробництва та прискорить економічне зростання.

Перспективою подальших досліджень є розробка графічного застосунку та програмна реалізація системи активного навчання нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 3D Solder Joint Reconstruction on SMD based on 2D Images [Electronic resource]. – Access mode: https://www.academia.edu/25794300/3D_Solder_Joint_Reconstruction_on_SMD_based_on_2D_Images.
2. Automatic Optical Inspection for Surface Mounting Devices with IPC-A-610D compliance [Electronic resource]. – Access mode: https://www.academia.edu/4616712/Automatic_Optical_Inspection_for_Surface_Mounting_Devices_with_IPC_A_610D_compliance.
3. Automatic optical inspection, AOI systems [Electronic resource]. – Access mode: <https://www.electronics-notes.com/articles/test-methods/automatic-automated-test-ate/aoi-optical-inspection.php>.
4. Kohavi, R. Glossary of terms . Machine Learning [Text] / R. Kohavi, F. Provost. – Boston : Kluwer Academic Publishers, 1998. – 274 p.
5. Shai, S. Understanding Machine Learning: From Theory to Algorithms [Text] / S. Shai, B. Shai. – Cambridge : Cambridge University Press, 2014. – 449 p.
6. Friedman, J. H. Data Mining and Statistics: What's the connection?. Computing Science and Statistics [Text] / J. H. Friedman. – Stanford : Department of Statistics and Stanford Linear Accelerator Center, 1998. – 9 p.
7. Bishop, C. M. Pattern Recognition and Machine Learning [Text] / C. M. Bishop. – Singapore : Springer, 2006. – 738 p.
8. Detection of Bare PCB Defects by Image Subtraction Method using Machine Vision [Electronic resource]. – Access mode: https://www.researchgate.net/publication/265319589_Detection_of_Bare_PCB_Defects_by_Image_Subtraction_Method_using_Machine_Vision.
9. Sundaraj, K. PCB Inspection for Missing or Misaligned Components using Background Subtraction [Text] / K. Sundaraj. – Malaysia : University Malaysia Perlis School of Mechatronic Engineering, 2009. – 8 p.

10. Inspection of Bare Printed Circuit Board Using Planar Type ECT probe [Text] / [S. Yamada, M. Iwahara, Y. Fukuda et al.]. – Japan : Gunma University, 2004. – 478 p.
11. Automatic PCB Inspection Algorithms: A Survey [Text] / [M. Moganti, F. Ercal, C. H. Dagli et al.]. – Rolla : Academic Press, 1996. – 313 p.
12. PCB Fault Detection Using Image Processing [Electronic resource]. – Access mode: https://www.researchgate.net/publication/319497091_PCB_Fault_Detection_Using_Image_Processing.
13. Building an End-to-End Defect Classifier Application for Printed Circuit Boards [Electronic resource]. – Access mode: <https://towardsdatascience.com/building-an-end-to-end-deep-learning-defect-classifier-application-for-printed-circuit-board-pcb-6361b3a76232>.
14. Bengio, Y. Learning Deep Architectures for AI [Text] / Y. Bengio. – Montral : University Montral, 2009. – 55 p.
15. Szegedy, C. Deep neural networks for object detection [Text] / C. Szegedy, A. Toshev, E. Dumitru. – NY : Curran Associates Inc, 2013. – 2561 p.
16. Subject independent facial expression recognition with robust face detection using a convolutional neural network [Text] / [M. Katsuhiko, M. Yusuke, M. Matusugu et al.]. – Amsterdam : Elsevier Science Ltd, 2003. – 2643 p.
17. Bottou, L. Deep content-based music recommendation [Text] / L. Bottou, M. Ghahramani, Z. Weinberger. – NY : Curran Associates Inc, 2013. – 2561 p.
18. Rosenblatt, F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms [Text] / F. Rosenblatt. – Washington : Spartan Books, 1961. – 248 p.
19. Rumelhart, D. Learning Internal Representations by Error Propagation [Text] / D. Rumelhart, G. Hinton R. J. Williams. – Cambridge : MIT Press, 1986. – 362 p.
20. Субботін, С. О. Нейронні мережі [Текст] / С. О. Субботін, А. О. Олійник. - Запоріжжя : ЗНТУ, 2014. - 132 с.

21. Liou, C. Y. Modeling word perception using the Elman network [Text] / C. Y. Liou, J. C. Huang, W. C. Yang. – Taiwan : Elsevier, 2008. – 315 p.
22. Duda, R. O. Use of the Hough Transformation to Detect Lines and Curves in Pictures [Text] / R. O. Duda, P. E. Hart. – NY : Communications of the ACM, 1972. – 15 p.
23. Лопушанська, Г. П. Перетворення Фур'є, Лапласа: узагальнення та застосування [Текст] / Г. П. Лопушанська, А. О. Лопушанський, О. М. М'яус. - Львів : ЛНУ, 2014. - 152 с.
24. What is Binning good for? [Electronic resource]. – Access mode: <http://www.starrywonders.com/binning.html>.
25. CCD Signal-To-Noise Ratio [Electronic resource]. – Access mode: <https://www.microscopyu.com/tutorials/ccd-signal-to-noise-ratio>.
26. Python 3.8.0 [Electronic resource]. – Access mode: <https://www.python.org/downloads/release/python-380/>.
27. Project Jupyter [Electronic resource]. – Access mode: <https://jupyter.org/>.
28. gRPC [Electronic resource]. – Access mode: <https://grpc.io/>.
29. What are protocol buffers? [Electronic resource]. – Access mode: <https://developers.google.com/protocol-buffers>.
30. TensorFlow [Electronic resource]. – Access mode: <https://www.tensorflow.org/>.
31. matplotlib [Electronic resource]. – Access mode: <https://matplotlib.org/>.
32. Downey, A. Think Python How to Think Like a Computer Scientist [Text] / A. Downey. - Needham : Green Tea Press, 2015. - 292 p.
33. TensorFlow: Open source machine learning [Electronic resource]. – Access mode: https://www.youtube.com/watch?v=oZikw5k_2FM.
34. Остапенко, В. В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп'ютерні та інтелектуальні системи і

мережі» всіх форм навчання [Текст] / В. В. Остапенко, Е. М. Касьян, Є. М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с

35. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5-те, перероблене й доповнене : затв. М-вом енергетики та вугільної промисловості України 20.06.17 : введення в дію з 20.11.17. – Х. : Міненерговугілля України, 2017. – 793 с. ; 24 см. – 5000 прим.

36. Електробезпека. Захисне заземлення, занулення: ГОСТ 12.1.030-81. – [введ. 1982-07-01]. – М.: Изд-во стандартів, 1982. – 10 с.

37. Правила охорони праці під час експлуатації електронно-обчислювальних машин: НПАОП 0.00-1.28-10. Наказ Держгірпромнагляду від 26.03.2010 р. №65.

38. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машині: ДсанПіН 3.3.2.007-98. – К.: Держстандарт України, 1998. – 28 с.

39. Природне і штучне освітлення: ДБН В.2.5-28-2006. – К.: Держстандарт України, 2006. – 96 с.

40. Москальова В.М. Основи охорони праці [Текст] / В. М. Москальова. – Рівне: НУВГП, 2006. – 666 с.

41. Протипожежні норми: ДБН В.2.1.1-2002. – К.: Держстандарт України, 2002. – 42 с.

42. Бедрій Я.І. Охорона праці та пожежна безпека: навчальний посібник для студентів вищих навчальних закладів та інженерів-практиків [Текст] / Я. І. Бедрій. – Тернопіль: Навчальна книга – Богдан, 2013. – 184 с.

43. Михайлюк В. О. Цивільна безпека: навч. посібник [Текст] / В. О. Михайлюк, Б. Д. Халмурадов. – К: Центр учбової літератури, 2008. – 158 с.

44. Бедрій Я. І. Цивільний захист України: навч. посібник для студентів вищих навчальних закладів [Текст] / Я. І. Бедрій, В. Малов. – Тернопіль: Навчальна книга - Богдан, 2014. – 392 с.

ДОДАТОК А
Текст програми

A.1 Текст файла `classification_Fails_typeese.ipynb`

```
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow import keras
import tensorflow as tf
import statistics
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 0 - defect; 1 - pseudo defect
def prepare_data(data):
    X, y = [], []
    for ind, defect in enumerate(data[2]):
        X.append(data[1][ind])
        y.append(0) if defect[0] else y.append(1)

    X = np.array(X)
    y = np.array(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.17, random_state=17)
    X_train = X_train / 255.0
    X_test = X_test / 255.0
    return X_train, X_test, y_train, y_test

def fit_check_model(X_train, y_train):
    model = keras.Sequential([
        keras.layers.Conv2D(16, 3, padding='same', activation='relu',
input_shape=(100, 100 ,3)),
        keras.layers.MaxPooling2D(),
        keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
        keras.layers.MaxPooling2D(),
        keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
        keras.layers.MaxPooling2D(),
        #     keras.layers.Conv2D(128, 3, padding='same', activation='relu'),
        #     keras.layers.MaxPooling2D(),
        keras.layers.Flatten(),
        keras.layers.Dense(64, activation='relu'),
        keras.layers.Dense(2, activation='softmax')
    ])
    model.compile(
```

```

        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )
    model.fit(X_train, y_train, epochs=5)
    return model

def evaluate_model(model, X_test, y_test):
    test_loss, test_acc = model.evaluate(X_test, y_test)
    print('\nTest accuracy:', test_acc)

def make_confusion_matrix(model, X_test, y_test):
    predictions = model.predict(X_test)
    con_mat = tf.math.confusion_matrix(y_test, [np.argmax(pr) for pr in
predictions], num_classes=2).numpy()
    print('\nConfusion matrix:\n', con_mat)
    con_mat_norm = np.around(con_mat.astype('float') /
con_mat.sum(axis=1)[:, np.newaxis], decimals=2)
    print('\nConfusion matrix (normalized):\n', con_mat_norm)
    avr_conf = statistics.mean(con_mat_norm.diagonal())
    print('\nAverage accuracy:', avr_conf)
    class_names = ['defect', 'pseudo defect']
    con_mat_df = pd.DataFrame(con_mat_norm, index = class_names, columns
= class_names)
    figure = plt.figure(figsize=(10, 10))
    sns.heatmap(con_mat_df, annot=True, cmap=plt.cm.Blues)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

X_train_0201,      X_test_0201,      y_train_0201,      y_test_0201      =
prepare_data(data[0])
model_0201 = fit_check_model(X_train_0201, y_train_0201)
evaluate_model(model_0201, X_test_0201, y_test_0201)
make_confusion_matrix(model_0201, X_test_0201, y_test_0201)

```

A.2 Текст файла `aoi_dataset_preprocess.ipynb`

```

import sys
import math
import numpy as np

```

```

from matplotlib import pyplot as plt
from matplotlib import cm
from matplotlib.image import imsave
import skimage
import skimage.transform
import skimage.filters
from skimage.transform import rotate
from skimage.restoration import inpaint
from scipy.signal import find_peaks
# np.set_printoptions(threshold=sys.maxsize)

def loadData(name):
    path='E:\\datasets\\aoi_ng\\'
    return [
        np.load(path+name+'_True_images.npy'),
        np.load(path+name+'_False_images.npy'),
        np.load(path+name+'_False_fails.npy'),
    ]

def make_dict(data):
    d = {}
    for i, image in enumerate(data):
        d[i] = image
    return d

def make_dict_by_indexes(data, indexes):
    d = {}
    for ind in indexes:
        d[ind] = data[ind]
    return d

def show_data(data):
    for ind, img in data.items():
        plt.title(ind)
        plt.imshow(img)
        plt.show()

def crop_image_only_outside(cr_img, tol=0):
    # cr_img is 2D or 3D image data, tol is a tolerance
    mask = cr_img>tol
    if cr_img.ndim==3:
        mask = mask.all(2)
    m,n = mask.shape

```

```

mask0,mask1 = mask.any(0),mask.any(1)
col_start,col_end = mask0.argmax(),n-mask0[::-1].argmax()
row_start,row_end = mask1.argmax(),m-mask1[::-1].argmax()
return cr_img[row_start:row_end,col_start:col_end]

def get_mask(def_img, black_thresh=30/255.0, debug=False):
    if debug:
        plt.imshow(def_img)
        plt.show()
    bl_mask = np.copy(def_img)
    for i_r, r in enumerate(bl_mask):
        for i_c, c in enumerate(r):
            res = 1 if (c < black_thresh).all() else 0
            bl_mask[i_r][i_c] = res
    if debug:
        plt.imshow(bl_mask[:, :, 0], cmap=cm.gray)
        plt.show()
    return bl_mask[:, :, 0]

def inpaint_background(def_img, im_mask):
    tmp = def_img.copy()
    for layer in range(tmp.shape[-1]):
        tmp[np.where(im_mask)] = 0
        image_result = inpaint.inpaint_biharmonic(tmp, im_mask,
multichannel=True)
    return image_result

def take_cropped(data, debug=False):
    cropped = {}
    for ind, img in data.items():
        img = crop_image_only_outside(img, 8/255.0)
        cropped[ind] = img
        if debug:
            plt.imshow(img)
            plt.title(ind)
            plt.show()
    return cropped

def take_inpainted(data, debug=False):
    inpainted = {}
    for ind, crop in data.items():
        im_mask = get_mask(crop)
        im_paint = inpaint_background(crop, im_mask)

```

```

        inpainted[ind] = im_paint
    if debug:
        plt.imshow(im_paint)
        plt.show()
    return inpainted

def take_grayscales(data, debug=False):
    grayscales = {}
    for ind, img in data.items():
        threshold=np.maximum(img-[0.2,1,0.2],0)
        grayscale=np.dot(threshold,[1,1,1])
        grayscales[ind] = grayscale
    if debug:
        plt.title(ind)
        plt.imshow(grayscale)
        plt.show();
    return grayscales

def fourier(images, debug=False):
    ffts = {}
    for ind, img in images.items():
        fft=np.fft.fftshift(np.fft.fft2(img))
        ffts[ind] = fft
    if debug:
        plt.figure(figsize=(7,7))
        plt.title(ind)
        plt.imshow(np.abs(fft));
        plt.show();
    return ffts

def fourier_result(ffts, debug=False):
    fourier_res = {
        '90': [],
        '0': [],
        'mark': [],
    }
    for ind, fft in ffts.items():
        line_vote = []
        for x in reversed(range(1, fft.shape[1])):
            y1, y2 = 0, fft.shape[0] - 1
            x1, x2 = x, fft.shape[1] - 1 - x
            angle = math.atan2(y2-y1, x2-x1)
            vote = np.sum(np.abs(fft[skimage.draw.line(y1,x1,y2,x2)]))

```

```

        line_vote.append([angle,vote])
    for y in range(1, fft.shape[0]):
        y1, y2 = y, fft.shape[0]-1-y
        x1, x2 = 0, fft.shape[1]-1
        angle = math.atan2(y2-y1, x2-x1)
        vote = np.sum(np.abs(fft[skimage.draw.line(y1,x1,y2,x2)]))
        line_vote.append([angle,vote])
    line_vote = np.array(line_vote)

    winning_angle =
    math.degrees(line_vote[np.argmax(line_vote[:,1]),0])
    if -3 <= winning_angle < 3:
        res = '0'
    elif 87 <= winning_angle <93:
        res = '90'
    else:
        res = 'mark'

    peaks, _ = find_peaks(line_vote[:,1], prominence=1000)
    peaks = line_vote[peaks, 1]
    num_peaks = len(peaks)
    if num_peaks > 1:
        if num_peaks > 2:
            peaks.sort()
            peaks = peaks[-2:]
        # сравнить вершины относительно
        if abs(peaks[0] - peaks[1]) < 500:
            res = 'mark'
    if debug:
        plt.plot(line_vote[:,0], line_vote[:,1])
        plt.title(f"{ind}: {res} degree")
        plt.show();
    fourier_res[res].append(ind)
    return fourier_res

def take_fourier(images, debug=False):
    ffts1 = fourier(images)
    fourier_r = fourier_result(ffts1)
    if debug:
        print(fourier_r)
    return fourier_r

def binning(images, debug=False):
    sxs, sys = {}, {}

```

```

for ind, img in images.items():
    sx = np.abs(skimage.filters.sobel_h(img))
    sy = np.abs(skimage.filters.sobel_v(img))
    sxs[ind] = sx
    sys[ind] = sy
    if debug:
        plt.imshow(sx)
        plt.title(ind)
        plt.show()
        plt.imshow(sy)
        plt.title(ind)
        plt.show()
return sxs, sys

def binning_result(sxs, sys, treshold=1.3, debug=False):
    binning_res = {
        '90': [],
        '0': [],
        'mark': [],
    }
    for ind in sxs.keys():
        x = np.maximum(sxs[ind]-0.3,0)
        y = np.maximum(sys[ind]-0.3,0)
        sum_x = np.sum(x)
        sum_y = np.sum(y)
        if debug:
            plt.imshow(x)
            plt.title(f"{ind}: {sum_x}")
            plt.show()
            plt.imshow(y)
            plt.title(f"{ind}: {sum_y}")
            plt.show()
        res = 'mark'
        if sum_x > sum_y:
            sum_y = 1 if not sum_y else sum_y
            ratio = sum_x / sum_y
            if ratio > treshold:
                res = '90'
        #         print(f"{ind}-x: {sum_x / sum_y}")
    elif sum_y > sum_x:
        sum_x = 1 if not sum_x else sum_x
        ratio = sum_y / sum_x
        if ratio > treshold:

```

```

        res = '0'
    #         print(f"{ind}-y: {sum_y / sum_x}")
    if debug:
        print(f"{ind}: {res}")
        binning_res[res].append(ind)
    return binning_res

def take_binning(images, debug=False):
    sxs1, sys1 = binning(images)
    binning_r = binning_result(sxs1, sys1)
    if debug:
        print(binning_r)
    return binning_r

def houghs_trans(images, debug=False):
    houghs = {}
    for ind, img in images.items():
        h, theta, d = skimage.transform.hough_line(img)
        houghs[ind] = [h, theta, d]
        if debug:
            plt.imshow(h)
            plt.title(ind)
            plt.show()
    return houghs

def houghs_trans_result(houghs, debug=False):
    houghs_trans_res = {
        '90': [],
        '0': [],
        'mark': [],
    }
    for ind, [h, theta, d] in houghs.items():
        angle_binned = np.sum(h ** 2, axis = 0)
        winning_angle = theta[np.argmax(angle_binned)]
        if debug:
            plt.title(f"{ind}: {math.degrees(winning_angle)}")
            plt.plot(angle_binned)
            plt.show()
        if -3 <= math.degrees(winning_angle) < 3:
            res = '0'
        elif 87 <= abs(math.degrees(winning_angle)) < 93:
            res = '90'
        else:

```

```

        res = 'mark'
    if debug:
        print(f"{ind}: {res}")
    houghs_trans_res[res].append(ind)
return houghs_trans_res

def take_houghs(images, debug=False):
    houghs1 = houghs_trans(images)
    houghs_trans_r = houghs_trans_result(houghs1)
    if debug:
        print(houghs_trans_r)
    return houghs_trans_r

def check_image_rotation_result(images, by_width=True, prop=1,
debug=False):
    rotation_res = {
        '90': [],
        '0': [],
        'mark': [],
    }
    for ind, image in images.items():
        h, w, _ = image.shape
        if not by_width:
            w, h = h, w
        if h > w * prop:
            res = '90'
        elif w > h * prop:
            res = '0'
        else:
            res = 'mark'
        if debug:
            print(f"{ind}: {res}")
        rotation_res[res].append(ind)
    return rotation_res

def take_rotated(images, debug=False):
    rotation_r = check_image_rotation_result(images, by_width=False,
prop=1.3)
    if debug:
        print(rotation_r)
    return rotation_r

def save_data(path, data):

```

```

        with open(path, 'w') as f:
            f.write(', '.join([str(ind) for ind in data]))

#data = [loadData(name) for name in
['0201', '0402', '0603', '0805', '1206', 'pins']]
cur = '0201'
initial_data = loadData(cur)
plt.title('1')
plt.imshow(initial_data[1][0])
plt.show()
false_images = make_dict(initial_data[1])
true_images = make_dict(initial_data[0])
right = []
wrong = []
mark = []
right_or = '90'
wrong_or = '0'
false_cropped = take_cropped(false_images)
res = take_rotated(false_cropped)
right += res[right_or]
false_rest = make_dict_by_indexes(false_images, res[wrong_or] +
res['mark'])
false_grayscales = take_grayscales(false_rest)
res_f = take_fourier(false_grayscales)
false_cropped_rest = take_cropped(false_rest)
# false_inpainted = take_inpainted({k:false_cropped_rest[k] for k in
[1,3,5] if k in false_cropped_rest})
false_inpainted = take_inpainted(false_cropped_rest)
false_inpainted_gray = take_grayscales(false_inpainted)
res_b = take_binning(false_inpainted_gray)
res_h = take_houghs(false_inpainted_gray)
for ind, image in false_rest.items():
    if ind in res_f[wrong_or] and ind in res_b[wrong_or] and ind in
res_h[wrong_or]:
        wrong.append(ind)
    elif ind in res_f[right_or] and ind in res_b[right_or] and ind in
res_h[right_or]:
        right.append(ind)
    else:
        mark.append(ind)
print(wrong)
print(right)
print(mark)

```

```

save_data('.\\Results\\' + cur + '_False_wrong.txt', wrong)
save_data('.\\Results\\' + cur + '_False_right.txt', right)
save_data('.\\Results\\' + cur + '_False_mark.txt', mark)
save_data('.\\Results\\' + cur + '_True_wrong.txt', wrong)
save_data('.\\Results\\' + cur + '_True_right.txt', right)
save_data('.\\Results\\' + cur + '_True_mark.txt', mark)

```

A.3 Текст файла clustering_True_False.ipynb

```

%matplotlib inline
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.lines import Line2D
from matplotlib import offsetbox
from sklearn.manifold import Isomap

def loadData(i,name):
    path='E:\\datasets\\aoi_ng\\'
    return [\
        np.load(path+name+'_True_images.npy'),\
        np.load(path+name+'_False_images.npy'),\
        np.load(path+name+'_False_fails.npy')\
    ]

data = [loadData(i,name) for i,name in
enumerate(['0201', '0402', '0603', '0805', '1206', 'pins'])]

def make_model(data_t, data_f):
    reshape_t = data_t.reshape([-1, 100*100*3])
    reshape_f = data_f.reshape([-1, 100*100*3])
    reshape_all = np.concatenate((reshape_t, reshape_f), axis=0)

    model = Isomap(n_components=2)
    proj = model.fit_transform(reshape_all)
    print(proj.shape)

    return proj

def plot_components(data, proj, limit, images=None, ax=None,
thumb_frac=0.05, cmap='gray', filename=''):
    ax = ax or plt.gca()

    # proj = model.fit_transform(data)
    ax.plot(proj[:, 0][:limit], proj[:, 1][:limit], '.b')
    ax.plot(proj[:, 0][limit:], proj[:, 1][limit:], '.r')

```

```

plt.legend(handles=[Line2D([0], [0], marker='o', color='w',
label='True', markerfacecolor='b', markersize=10),Line2D([0], [0], marker='o',
color='w', label='False', markerfacecolor='r', markersize=10)])
if images is not None:
    min_dist_2 = (thumb_frac * max(proj.max(0) - proj.min(0))) ** 2
    shown_images = np.array([2 * proj.max(0)])
    for i in range(data.shape[0]):
        dist = np.sum((proj[i] - shown_images) ** 2, 1)
        if np.min(dist) < min_dist_2:
            # don't show points that are too close
            continue
        shown_images = np.vstack([shown_images, proj[i]])
        if i < limit:
            bb = dict(fc="b")
        else:
            bb = dict(fc="r")
        imagebox = offsetbox.AnnotationBbox(
            offsetbox.OffsetImage(images[i], cmap=cmap),
            proj[i], bboxprops=bb
        )
        ax.add_artist(imagebox)
    if filename:
        plt.savefig(f"clustering/{filename}.png")

proj_0201 = make_model(data[0][0], data[0][1])
all_0201 = np.concatenate((data[0][0], data[0][1]), axis=0)

proj_0402 = make_model(data[1][0][::10], data[1][1][::2])
all_0402 = np.concatenate((data[1][0][::10], data[1][1][::2]), axis=0)

proj_0603 = make_model(data[2][0][::30], data[2][1][::2])
all_0603 = np.concatenate((data[2][0][::30], data[2][1][::2]), axis=0)

proj_0805 = make_model(data[3][0][::20], data[3][1][::2])
all_0805 = np.concatenate((data[3][0][::20], data[3][1][::2]), axis=0)

proj_1206 = make_model(data[4][0][::5], data[4][1])
all_1206 = np.concatenate((data[4][0][::5], data[4][1]), axis=0)

proj_pins = make_model(data[5][0][::50], data[5][1][::30])
all_pins = np.concatenate((data[5][0][::50], data[5][1][::30]), axis=0)

# 0201

```

```

fig, ax = plt.subplots(figsize=(15, 15))
plot_components(all_0201, proj_0201, len(data[0][0]),
images=all_0201[:, ::2, ::2], filename='0201_img')

fig, ax = plt.subplots(figsize=(15, 15))
plot_components(all_0201, proj_0201, len(data[0][0]),
filename='0201_point')

# 0402
fig, ax = plt.subplots(figsize=(15, 15))
plot_components(all_0402, proj_0402, len(data[1][0][::10]),
images=all_0402[:, ::2, ::2], filename='0402_img')

fig, ax = plt.subplots(figsize=(15, 15))
plot_components(all_0402, proj_0402, len(data[1][0][::10]),
filename='0402_point')

# 0603
fig, ax = plt.subplots(figsize=(15, 15))
plot_components(all_0603, proj_0603, len(data[2][0][::30]),
images=all_0603[:, ::2, ::2], filename='0603_img')

fig, ax = plt.subplots(figsize=(15, 15))
plot_components(all_0603, proj_0603, len(data[2][0][::30]),
filename='0603_point')

```

ДОДАТОК Б
Слайди презентації

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ДІАГНОСТУВАННЯ ДРУКОВАНИХ ПЛАТ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

Виконала
ст. гр. КНТ-129м

В. С. Хохлова

Керівник
д.т.н., професор

С. О. Субботін

Рисунок Б.1 – Титульний слайд

Об'єкт, предмет та мета дослідження

► **Об'єкт дослідження:**

Процес обробки та класифікації зображень.

► **Предмет дослідження:**

Методи автоматизованої оптичної інспекції друкованих плат на виробництві.

► **Мета:**

Дослідити та програмно реалізувати методи автоматизованої оптичної інспекції задля підвищення швидкості та точності діагностування друкованих плат.

Рисунок Б.2 – Об'єкт, предмет, мета та задачі дослідження

Задачі дослідження

- 1) проаналізувати предметну область та існуючі методи вирішення завдання;
- 2) обрати методи обробки зображень та машинного навчання, мову програмування;
- 3) розробити алгоритм діагностування плат;
- 4) розробити програму для діагностування плат;
- 5) провести експериментальне дослідження роботи програми.

► 3

Рисунок Б.3 – Задачі дослідження

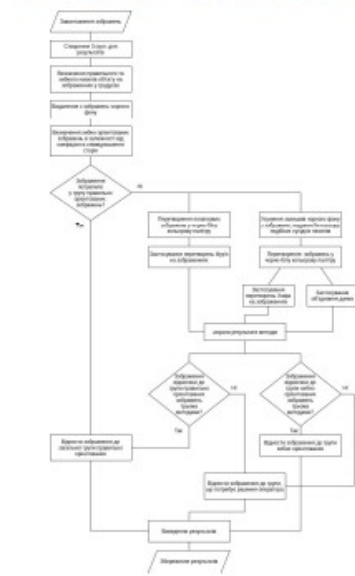
Аналіз предметної області, існуючих рішень

Методи аналізу зображень:	Існуючі рішення:
1) перетворення Хафа;	1) віднімання зображень
2) перетворення Фур'є;	для виявлення дефектних
3) об'єднання даних.	частин;
	2) розділення зображення
Види нейронних мереж:	на частини, їх порівняння
1) глибинні;	по пікселям;
2) згорткові;	3) порівняння реальних
3) багатосаровий перцептрон;	зображень з шаблонами.
4) автокодувальник.	

► 4

Рисунок Б.4 – Аналіз предметної області, існуючих рішень

Розроблений алгоритм аналізу зображень



- 1) видалення чорного фону;
- 2) визначення коефіцієнту відношення сторін;
- 3) перетворення зображень у чорно-білу кольорову палітру;
- 4) зміна кольору чорних пікселів на колір сусідніх;
- 5) перетворення Фур'є, Хафа, об'єднання даних;
- 6) аналіз отриманих результатів.

► 5

Рисунок Б.5 – Розроблений алгоритм аналізу зображень

Структурна схема програми



► 6

Рисунок Б.6 – Структурна схема програми

Схема функціонування програми

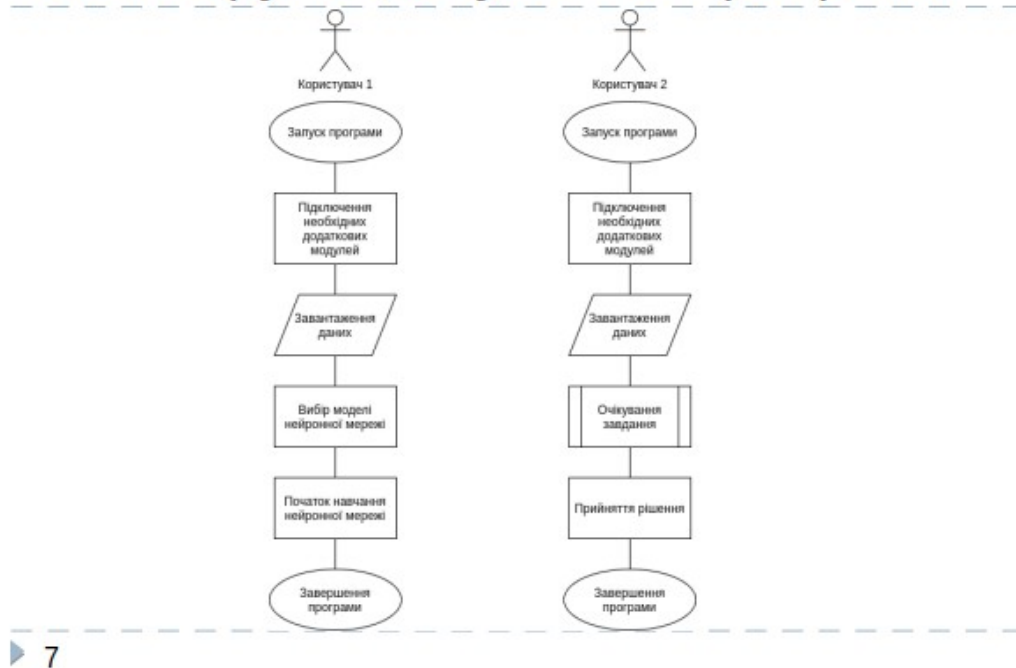


Рисунок Б.7 – Схема функціонування програми

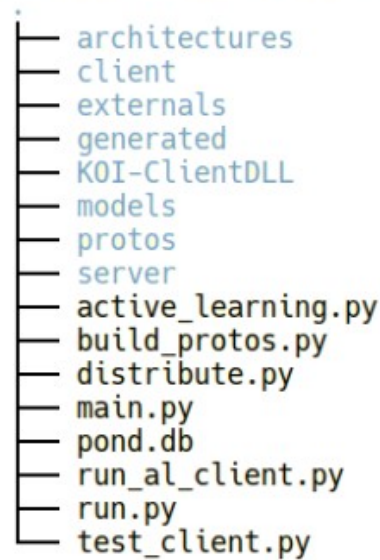
Інструменти реалізації

Обрані програмні засоби: Python 3, Jupyter Notebook, gRPC, Protocol Buffers, TensorFlow, matplotlib.

Мова програмування	Python	C++	Lua	Java	R	MATLAB
Продуктивність	-	+	-	+	-	-
Універсальність	+	-	+	+	-	-
Візуалізації даних	+	-	+	+	+	+
Бібліотеки для використання нейронних мереж	+	+	+	+	+	+
Ліцензія	Безкоштовна	Безкоштовна	Безкоштовна	Безкоштовна	Безкоштовна	Платна
Простота у вивченні	+	-	+	-	-	-
Стабільність	+	+	+	+	+	+
Читабельність	+	-	+	+	+	-

Рисунок Б.8 – Інструменти реалізації

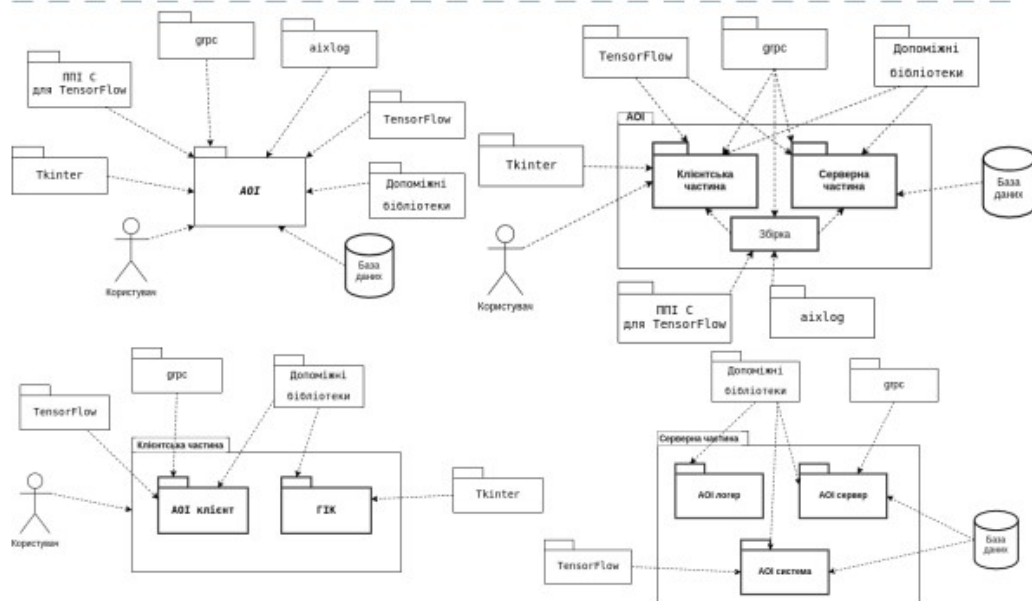
Структура файлів програми



9

Рисунок Б.9 – Структура файлів програми

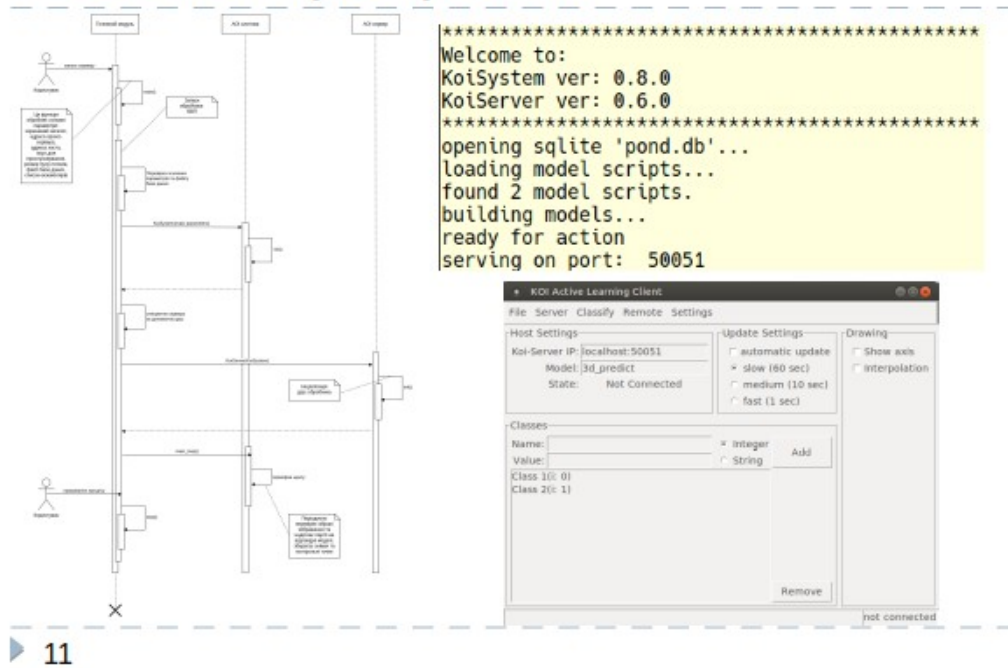
Компоненти програми



10

Рисунок Б.10 – Компоненти програми

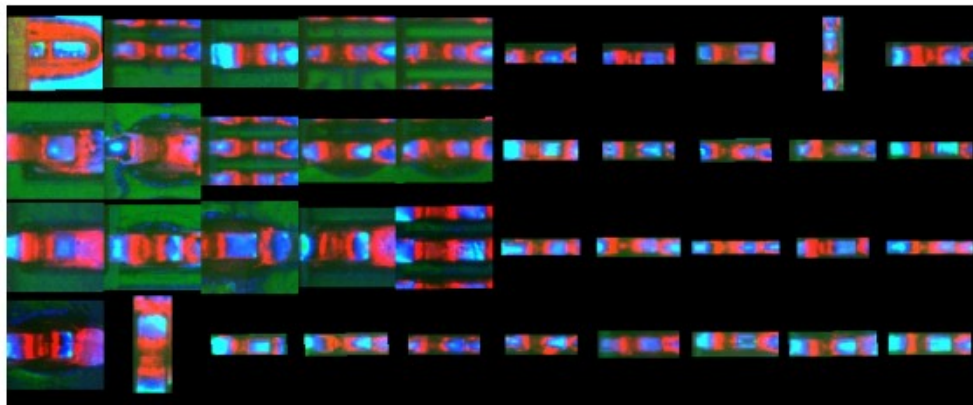
Робота програми



► 11

Рисунок Б.11 – Робота програми

Вхідні дані



► 12

Рисунок Б.12 – Вхідні дані

Обробка початкових даних

Перетворення Фур'є

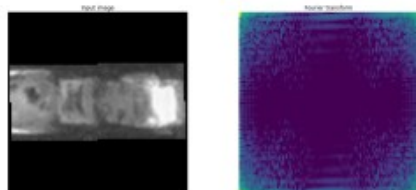


Рисунок 1

Перетворення Хафа

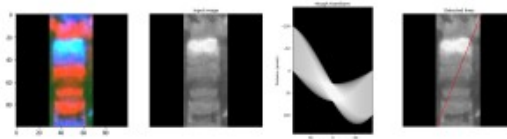


Рисунок 3

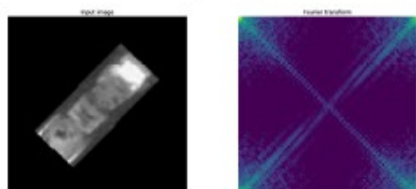


Рисунок 2

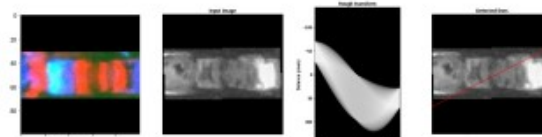


Рисунок 4

► 13

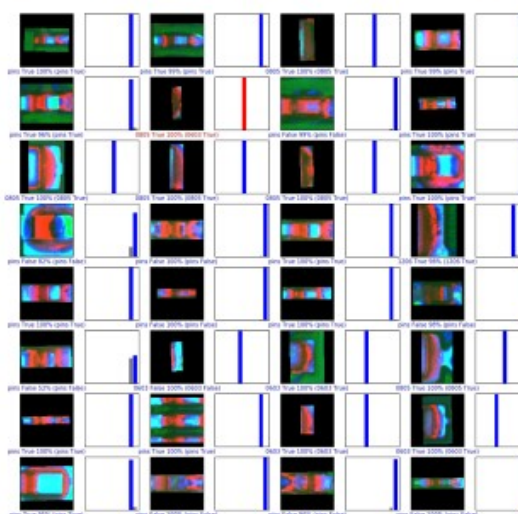
Рисунок Б.13 – Обробка початкових даних

Результати дослідження

Точність моделі - 94%.



Матриця помилок (81-99%)

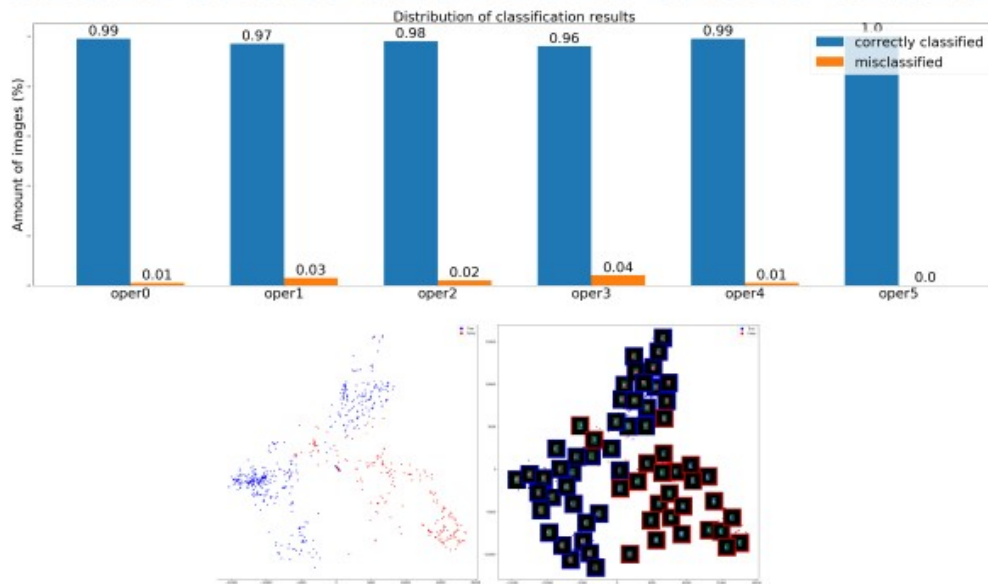


Візуалізація помилок

► 14

Рисунок Б.14 – Результати дослідження

Експериментальні дослідження



► 15

Рисунок Б.15 – Експериментальні дослідження

Висновки

- У результаті виконання магістерської дипломної роботи було вирішено актуальне завдання дослідження та програмної реалізації методів автоматизованої оптичної інспекції на основі нейронних мереж для вирішення задачі діагностування друкованих плат.
- Було створено та програмно реалізовано алгоритм для аналізу початкових даних (зображень), який використовує перетворення Фур'є, перетворення Хафа та метод об'єднання даних.
- Результати проведеного дослідження показали, що згорткові нейронні мережі класифікують дані з точністю 94%.
- Розроблена система допоможе підвищити швидкість та точність діагностування, надасть можливість виявити дефекти виробництва на початку.
- Практична цінність роботи полягає у тому, що було створено програму для діагностування друкованих плат на етапі виробництва, що дозволить підвищити якість виробництва та прискорить економічне зростання.
- Перспективою подальших досліджень є розробка графічного застосунку та програмна реалізація системи активного навчання нейронних мереж.

► 16

Рисунок Б.16 – Висновки

