Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Комп'ютерно-інтегровані методи контролю гнучких друкованих плат
(тема)

Виконав: студент 2 курсу, гр. КІТПВм-19-1
Назаренко В.С.
(прізвище, ініціали)

Спеціальність 151 Автоматизація
та комп'ютерно-інтегровані технології

освітньої програми Комп'ютерно-інтегровані
Технологічні процеси і виробництва
(код і повна назва напряму)

Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник доц. Боцман І. В.
(посада, прізвище, ініціали)

Допускається до захисту
зав. кафедри _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп’ютеризованих технологій
Кафедра	Комп’ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми	освітньо-професійна
Освітня програма	Комп’ютерно-інтегровані технологічні процеси і виробництва

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Назаренку Владиславу Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Комп'ютерно-інтегровані методи контролю гнучких друкованих плат
затверджена наказом по університету від 02.11. 2020 р. № 1511 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 07.12. 2020 р.
3. Вихідні дані до роботи 3.1 Об’єкт дослідження – гнучкі друковані плати розміром до 150 мм × 150 мм
3.2 Автоматичний рентген-апарат
3.3 Щільність зображення не менше ніж 10 ліній на міліметр
4. Перелік питань, які потрібно опрацювати у роботі
4.1 Вступ
4.2 Аналіз предметної області
4.3 Розробка методу тестування гнучких друкованих плат
4.4 Розробка системи розпізнавання дефектів
4.5 Розробка програмних рішень для реалізації системи розпізнавання дефектів
4.6 Проведення експерименту та аналіз результатів
4.7 Дослідження заходів з охорони праці на виробництві
4.8 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Боцман І. В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Аналіз технічного завдання*	01.09.2020 р.	Виконав
2	*Аналіз предметної області*	07.09.2020 р.	Виконав
3	*Розробка методу тестування*		
	гнучких друкованих плат	16.09.2020 р.	Виконав
4	*Розробка системи розпізнавання*		
	дефектів	29.09.2020 р.	Виконав
5	*Розробка програмних рішень*		
	для реалізації системи		
	розпізнавання дефектів	21.11.2020 р.	Виконав
6	*Проведення експерименту*		
	та аналіз результатів	25.11.2020 р.	Виконав
7	*Оформлення пояснювальної записки*	29.11.2020 р.	Виконав
8	*Подання роботи до ДЕК*	07.12.2020 р.	Виконав

Дата видачі завдання

Студент _____ Назаренко В.С.

(підпис) (прізвище, ініціали)

Керівник роботи _____ Боцман І.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 94 с., 68 рис., 4 табл., 42 джерела.

ГНУЧКА ДРУКОВАНА ПЛАТА, КОМП'ЮТЕРНИЙ ЗІР, НЕЙРОННА МЕРЕЖА, КОНТРОЛЬ, PYTHON, TENSORFLOW, OPENCV

Об'єкт дослідження – процес контролю гнучких друкованих плат.

Предмет дослідження – гнучкі друковані плати.

Методи дослідження – теоретичні дослідження, практична реалізація, методи машинного навчання.

Мета магістерської атестаційної роботи – розробка комп'ютерно-інтегрованого методу контролю гнучких друкованих плат.

У магістерській атестаційній роботі проведено огляд та аналіз особливостей існуючих методів контролю гнучких друкованих плат, виконаний вибір та обґрунтування оснащення для контролю гнучких друкованих плат.

Розроблено новий автоматизований комп'ютерно-інтегрований метод контролю якості гнучких друкованих плат, який використовує для аналізу нейронні мережі.

Розроблені модель згорткової нейронної мережі, класифікатор і система пошуку дефектів на базі нейронної мережі. Досліджені основні характеристики розроблюваних програмних рішень.

Результати магістерської атестаційної роботи були опубліковані у збірнику студентських наукових статей «Автоматизація та приладобудування».

ABSTRACT

Explanatory note consists of 94 p., 68 figures, 4 tables., 42 sources.

FLEX PRINTED CIRCUIT BOARD, COMPUTER VISION, NEURAL NETWORK, CONTROL, PYTHON, TENSORFLOW, OPENCV

The object of research – процес контролю гнучких друкованих плат.

The subject of research – flex printed circuit boards.

Research methods – theoretical research, practical implementation, machine learning methods.

The purpose of the master's certification work is development of computer-integrated method of flex printed circuit board control.

The master's attestation work analyzes the features of existing flex printed circuit board testing methods. Equipment for flex printed circuit boards control was chosen and chose was justified.

New automated computer-integrated method for flex printed circuit boards control with use of neural networks for analysis was developed.

The model of convolution neural network, classifier and defect search system based on neural network were developed.

The main characteristics of developed solutions were analyzed.

The results of the master's certification work were tested in the ADED journal.

ЗМІСТ

Перелік скорочень ……..8
Вступ.....9
1 Аналіз предметної області.....11
1.1 Тестування вхідного матеріалу.....12
1.2 Тестування виготовлених гнучких друкованих плат.....17
1.3 Вимоги до тестування розмірів.....19
1.4 Вимоги до тестування фізичних параметрів.....22
1.5 Вимоги до цільності конструкції.....24
1.6 Висновки до розділу 1.....26
2 Розробка методу тестування гнучких друкованих плат.....27
2.1 Вибір типу тестування ГДП.....27
2.2 Вибір методу автоматичного аналізу результатів тестування.....31
2.3 Згорткова нейронна мережа.....33
2.4 Вибір архітектури системи розпізнавання зображень.....43
2.5 Висновки до розділу 2.....45
3 Розробка системи розпізнавання зображень.....46
3.1 Вибір мови програмування.....46
3.2 Вибір робочого оточення.....47
3.3 Вибір системи контролю версій.....49
3.4 Розробка програми-класифікатора зображень.....50
3.5 Розробка нейронної мережі для розпізнавання дефектів.....59
3.6 Висновки до розділу 3.....76
4 Аналіз ефективності роботи розробленої програми.....77
4.1 Швидкість навчання.....77
4.2 Точність розпізнавання.....80
4.3 Висновки до розділу 4.....82
5 Охорона праці.....83

Висновки.....86
Перелік джерел посилання.....88
Додаток А Демонстраційний матеріал.....93

ПЕРЕЛІК СКОРОЧЕНЬ

ГДП – гнучка друкована плата;

ДП – друкована плата;

ЗНМ – згорткова нейронна мережа;

ЕОМ – електронно-обчислювальна машина;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ПОС – припій олов’яно-свинцевий;

ТД – технологічна документація;

ТЗ – технічне завдання;

ТКЛР – температурний коефіцієнт лінійного розширення;

ТП – технологічний процес;

API – Application Programming Interface.

ВСТУП

Дослідження та тестування є однією з важливіших і найскладніших частин будь-якого виробництва. Тестування надає зворотній зв’язок виробнику та дозволяє оптимізувати якість вихідного продукту та водночас знижувати вартість виробництва. Тестування готових виробів може бути витратним як в матеріальному плані, так і з точки зору тривалості. Але воно дозволяє утримувати якість виробів на достатньому рівні, не втрачати зайві ресурси та матеріали.

Гнучка друкована плата (ГДП) – це один чи більше шарів діелектрика, на якому сформований хоча б одне струмопровідне коло. Вона використовується для з’єднання окремих електронних елементів або вузлів в єдиний діючий пристрій.

ГДП відрізняються від звичайних склотекстолітових друкованих плат (ДП) тим, що можуть згинатись без ушкоджень для них. Це дозволяє виконувати монтаж електричних схем у труднодоступних місцях, збільшувати щільність монтажу в електронних блоках і використовувати ГДП як гнучкі з’єднувачі.

На сьогодні неможливо зробити сучасний компактний електронний виріб без використання ГДП.

Взагалі ГДП використовують у автомобільній електроніці, у споживчих пристроях, у медичній апаратурі, у телекомунікаціях, в електронно-обчислювальних машинах (ЕОМ), у приладобудуванні, у військовій та космічній апаратурі.

Існує багато причин для використання гнучких друкованих плат в тому числі і як з’єднуючих елементів в електронних приладах.

За допомогою ГДП можна збільшити щільність монтажу та як слід зменшити розміри пристрою. Також, гнучкі друковані плати значно легші за аналогічні жорсткі на 70-75 %. Основний конкурент ГДП серед з’єднуючих елементів – це шлейф з мідних проводів. ГДП легша, краще розсіює тепло та

виключає людський фактор під час складання. Також зменшується час та вартість складання вузла.

Одна з основних проблем під час тестування ГДП – це людський фактор. Дуже часто результати випробувань відстежуються та аналізуються операторами, що вносить фактор помилки.

До основних напрямків вирішення проблем у процесі тестування ГДП можна віднести:

– розробку більш специфічних і точних методів тестування;

– розробку методу виключення людського фактору.

Проблема підвищення якості виробу та зниження витрат на виробництво та тестування ГДП – складна та важлива, тому тема атестаційної роботи є актуальною.

Об’єкт дослідження – процес контролю гнучких друкованих плат.

Предмет дослідження – гнучкі друковані плати.

Методи дослідження – теоретичні дослідження, практична реалізація, методи машинного навчання.

Мета магістерської атестаційної роботи – розробка комп’ютерно-інтегрованого методу контролю гнучких друкованих плат.

Для досягнення поставленої мети необхідно:

– провести огляд і аналіз особливостей існуючих методів контролю гнучких друкованих плат;

– виконати вибір та обґрунтування оснащення для контролю ГДП;

– розробити новий метод контролю гнучких друкованих плат;

– розробити математичну модель процесу тестування та аналізу результатів;

– розробити програмний модуль для автоматизації контролю якості гнучких друкованих плат за розробленим методом;

– провести експериментальні дослідження для оцінки створеної системи;

– оформити пояснювальну записку згідно з рекомендаціями [1] та за вимогами ДСТУ 3008:2015 [2] та згідно з положеннями [3–7].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Тестування будь-якого продукту для того, щоб бути економічно обґрунтованим, як правило виконується на базі зразків. Це також стосується і гнучких друкованих плат. З гнучкими друкованими платами існує два рівні дослідження та тестування – тестування вхідного матеріалу та тестування готових зразків гнучких друкованих плат [8].

Вхідні матеріали аналізуються та тестуються для того, щоб запевнитись у тому, що вони будуть відповідати вимогам, які висуваються індустрією. Таке тестування надає певний рівень впевненості, що тільки відповідні до вимог продукти будуть використані для створення під час виробництва, щоб виключити його вплив на потенційні дефекти готових виробів.

Також вже виготовлений виріб тестують, щоб переконатись, що він відповідає вимогам, які висуваються специфікацією продукту, та бути впевненими, що фінальний виріб буде надійно виконувати свої функції протягом усього терміну служби.

Зразкові варіанти продукту, що передаються на тестування, не відрізняються від масових зразків, які будуть випускатись на виробничій лінії.

Виконані на одній лінії вироби мають бути практично ідентичними. Будь-які відхилення виробничого процесу постійно відстежуються та тестування виконується на статистично дійсній базі для того, щоб переконатись, що виробнича лінія продовжує випускати прийнятний продукт [8].

Тестування не слід плутати з аналізом. Аналіз –це найчастіше за все процес сортування для того, щоб відділити якісні зразки від поганих чи потенційно поганих.

Аналіз дозволяє створити своєрідний зворотній зв’язок для специфічної для виробника інформації стосовно загального стану виробництва. Аналіз також має використовуватися для того, щоб оцінювати якість продукту та те, чи вона відповідає вимогам [8].

1.1 Тестування вхідного матеріалу

Вхідні матеріли тестуються для того, щоб переконатись у тому, що готовий продукт не буде обмеженим у будь-якому плані через неоптимальну конструкцію чи якість матеріалу.

Для того, щоб забезпечити надійність виробу, який виготовлюється, тестуються вхідні матеріали, які використовують у гнучких друкованих платах, такі як [9]:

- базові матеріали;
- клейові речовини;
- фольга струмопровідного шару;
- допоміжні речовини.

Якість матеріалу оцінюється за наступними категоріями властивостей:

- фізичні;
- хімічні;
- електричні;
- середовищні.

Ці параметри в цілому являють собою спектр потенційних зон занепокоєння якістю продукту.

1.1.1 Тестування фізичних властивостей

Фізичні характеристики вхідного матеріалу перевіряються для того, щоб переконатись, що у матеріалу є в наявності мінімально необхідні фізичні властивості відповідно до вказаних вимог. Серед фізичних тестів можна виділити наступні [9]:

- тестування на розтяг і подовження;
- міцність на розрив;
- міцність на поширення розриву;
- гнучкість за низьких температур;

– стабільність розміру;
– міцність на відрив;
– наявність випарних речовин.

Таке розповсюджене механічне тестування як тестування на розтяг та подовження виконується для того, щоб визначити значення двох ключових фізичних властивостей гнучкого базового матеріалу, а саме граничну міцність на розрив та максимальний ступінь розтягу гнучкого базового матеріалу до моменту відмови. Схематичне зображення тестування на розтяг і подовження наведено на рисунку 1.1.

Ці два параметри є дуже важливими для прийняття рішення, чи підходить матеріал для конкретного застосування. Також це дуже важливі дані у процесі моделювання ГДП [10].

Рисунок 1.1 – Схематичне зображення тестування ГДП на розтяг і подовження

Таке випробування як тест на розрив використовується для того, щоб визначити, наскільки велика сила потрібна, щоб ініціювати процес розриву матеріалу. Це вимірювання міцності базового гнучкого матеріалу, одного з найважливіших факторів для гнучких схем, маючи на увазі доволі делікатну природу тонких гнучких матеріалів.

Випробування на поширення розриву – це ще одне дуже важливе вимірювання, яке залежить від міцності матеріалу. Це випробування розроблене для того, щоб вимірювати силу, яка необхідна для того, щоб розширити розрив матеріалу, який вже є. Це вимірювання дає вкрай важливу інформацію про остатню надійність продукту. Чим нижчим буде це значення, тим важливіше додавання технічних рішень, які запобігають розриву, під час розробки ГДП [10].

Вимірювання гнучкості за низьких температур виконується для того, щоб переконатися в стійкості матеріалу до крихкості та можливого розпадання за умов із низькими температурами, з якими можуть стикнутися під час експлуатації готові прилади на базі ГДП.

Характеристика стабільності розмірів має дуже важливу роль як для проектувальників, так і для виробника. Випробування на стабільність розмірів підтверджує, що гнучкий матеріал відповідає вимогам щодо усадки або розширення після нанесення металічної фольги. Чим кращою є стабільність матеріалу, тим простіше передбачити місцезнаходження об’єктів друкованої плати після виробництва.

Вимірювання міцності на відрив виконується для того, щоб переконатись у мінімальній міцності зв’язку між елементами ГДП.
Як правило, вимірювання проводиться між металічною фольгою та гнучкою основою, але воно також може проводитися між будь-якими іншими шарами та їх комбінаціями. Тести проводяться в різних умовах для того, щоб переконатись, що ця характеристика залишається незмінною.

Вимірювання вмісту випарних речовин дозволяє отримати дані про масову частину випарних речовин у гнучких ламінатах. Випарні речовини – це потенційне джерело відшарування та розпаду ламінату [11].

1.1.2 Тестування хімічних властивостей

Вимоги до хімічних параметрів гнучких друкованих плат дуже обмежені. Є два базових хімічних параметри, а саме:

– хімічна стійкість;

– горючість.

Вимірювання хімічної стійкості виконується для того, щоб переконатись у тому, що ламінат не отримає ушкоджень під час дії на нього різноманітних хімікатів, які часто використовуються під час виготовлення та використання гнучких друкованих плат [12].

Вимірювання горючості виконується для того, щоб визначити, який мінімальний рівень кисню необхідний для підтримання згорання ламінату.

1.1.3 Тестування електричних властивостей

Електричні характеристики гнучких друкованих плат дуже сильно залежать від електричних характеристик базового матеріалу. Найбільш важливі ці параметри для високовольтних, високочастотних аналогових і високошвидкісних цифрових схем. Електричні характеристики визначаються за допомогою тестування прикладів матеріалу, визначаються характеристики матеріалу за наступними параметрами [12]:

– діелектрична константа (Dk);
– коефіцієнт розсіювання;
– діелектрична міцність;
– поверхневий та об’ємний опір.

Визначення діелектричної константи є дуже важливим для подальшого проектування ГДП, воно визначає якість роботи готового модуля на визначених частотах.

Коефіцієнт розсіювання матеріалу визначає відношення між ємністю матеріалу та його провідністю. Коли вимірюється на визначеній частоті, цей параметр – ще одна дуже важлива характеристика для проектувальників друкованих плат. Набирання вологи також збільшує коефіцієнт розсіювання, що впливає на вірогідність втрати сигналів – чим більшою є частота, тим вищим буде ризик.

Діелектрична міцність матеріалу визначається для того, щоб отримати дані про мінімальну напругу пробою у даному матеріалі. Цей параметр є дуже важливим для високовольтних схем і пристроїв, що працюють на великих висотах, схильних до виникнення дугових розрядів.

Поверхневий та об'ємний опір визначається для того, щоб отримати мінімальні значення поверхневого та об'ємного опору матеріалу в теплих та вологих умовах, які можуть змінити електричні характеристики схеми. Це вимірювання визначає схильність матеріалу до витоку току у визначених умовах.

1.1.4 Тестування середовищних властивостей

Тестування середовищних властивостей здійснюється для того, щоб оцінити певні фізичні та електричні параметри матеріалу, які можуть змінюватись у результаті впливу середовища на матеріал впродовж життєвого циклу продукту. Зазвичай проводяться тестування матеріалу на [13]:

- поглинання вологи;
- опір у вологих умовах;
- стійкість до впливу грибків.

Поглинання вологи – це параметр, який визначає, як багато вологи базовий матеріал поглине в умовах з підвищеною вологістю. Поглинання вологи безпосередньо впливає на електричні характеристики виробу. Також поглинання вологи може вплинути на готову гнучку збірку, тому що поглинання зайвої вологи може призвести до вибухового випаровування вологи та, як наслідок, розшарування ламінату.

Тестування матеріалу на опір у вологих умовах проводиться для того, щоб визначити вплив перебування у вологому середовищі на електричні властивості матеріалу та переконатись у тому, що ці властивості не погіршуються більше за допустимий рівень [13].

Тест на стійкість до впливу грибків виконується для того, щоб визначити, чи є матеріал поживною речовиною для грибків і мікроорганізмів, які можуть призвести до деградації електричних або фізичних властивостей матеріалу.

1.2 Тестування виготовлених гнучких друкованих плат

Тестування завершених гнучких друкованих плат багато в чому повторює тестування вхідного матеріалу. Це виконується для того, щоб переконатись, що процес виробництва не призвів до деградації властивостей матеріалів. Як і в разі сировини, вимоги до випробувань були встановлені в декількох ключових аспектах, щоб переконатися, що продукт придатний за прямим призначенням.

Основні категорії тестування готової ГДП містять [14]:

- візуальну інспекцію;
- перевірку здатності металевого покриття до паяння;
- контроль фізичних розмірів;
- контроль фізичних властивостей;
- контроль цільності конструкції;
- перевірку якості перехідних отворів;
- контроль середовищних властивостей;
- тестування електричних параметрів.

Візуальна інспекція виконується як частина процесу тестування для того, щоб переконатись у відсутності ряду дефектів, які можуть бути виявлені візуально, наприклад [14]:

- розтікання припою;
- пустоти в металізованих отворах;
- дефекти обробки.

Присутність розтікання припою на контактних майданчиках чи доріжках на відстані від відкритих частин фольги під зовнішнім покриттям може свідчити про неякісну ламінацію верхнього захисного шару або занадто довге

знаходження плати у розплавленому припої. Приклад такого дефекту вказаний на рис. 1.2 [14]. Цей дефект зазвичай не є критичним і вважається прийнятним, якщо в технічному завданні не вказано інше.

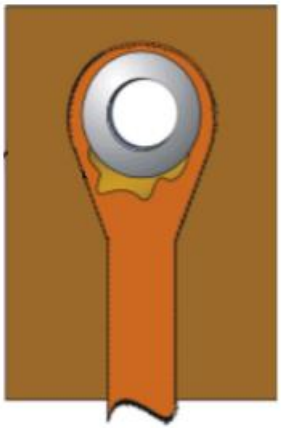

Рисунок 1.2 – Приклад розтікання припою під захисним шаром плати

Пустоти в металізованих отворах неприпустимих розмірів чи кількості (рис. 1.3) можуть призвести до можливих проблем із паянням чи надійністю металізованого отвору. Кільцеві пустоти зазвичай є приводом для відмови на вихідному контролі [15].

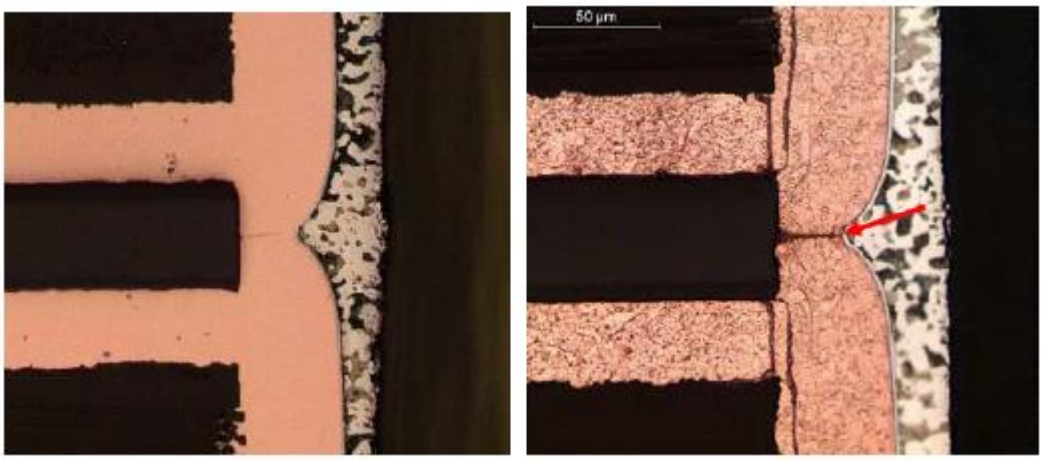

Рисунок 1.3 – Мікрофотографія пустоти в металізованому отворі

Дефекти обробки як категорія покривають різноманітні характеристики, такі як жирові забруднення, вигини, бруд, відбитки пальців, зморшки та складки та будь-які інші можливі похибки виробника плати [15].

1.3 Вимоги до тестування розмірів

Вимірювання розмірів готової ГДП виконуються для того, щоб переконатись в тому, що вони точно відповідають заданим параметрам, тому що точність розмірів друкованої плати зазвичай дуже важлива для якості роботи електричного кола пристрою та його загальної надійності. Далі буде описаний перелік основних ключових моментів для дослідження та вимірювання.

1.3.1 Визначення мінімального розміру кільцевого контактного майданчика

Недостатня площа кільцевого контактного майданчика може вплинути як на якість паяння, так і на надійність з’єднання. Вимоги до розміру кільцевого контактного майданчика помітно відрізняються для металізованих отворів та отворів без металізації. Контактні майданчики без металізації на односторонніх гнучких друкованих платах потребують набагато більшої площі для того, щоб забезпечити надійне з’єднання. Виходячи з цього, вимірюється розмір кільцевого контактного майданчика для того, щоб переконатись, що він відповідає вимогам [15].

1.3.2 Точність шаблону отворів

Точність шаблону отворів перевіряється для того, щоб впевнитись, що отвори знаходяться у правильних місцях. Невірно виконані отвори перетворюються на велику проблему на етапі складання електричного кола, особливо якщо воно відбувається на автоматизованих або автоматичних лініях. Також правильне місцезнаходження отворів важливе для цілей монтажу друкованого модулю в пристроях.

1.3.3 Точність шаблону захисного шару

Точність шаблону захисного шару має схожий вплив на якість друкованої плати, як і мінімальний розмір кільцевого контактного майданчика, тому що неспівпадіння захисного шару та контактного майданчика зменшує площу нанесення припою та призводить до зниження надійності з'єднання. Особливо чутливі до цього дефекту односторонні плати.

1.3.4 Залишки клейового шару на контактних майданчиках

Залишки клею на контактних майданчиках доволі часто з'являються під час виробництва в результаті ламінації захисного полімерного шару через те, що під час цієї операції клей наноситься під тиском та за високої температури. З більш сучасними покриттями, які наносяться за допомогою фотошаблонів, такі дефекти не зустрічаються [15].

1.3.5 Потрапляння повітря під захисний шар

Потрапляння повітря під захисний шар ГДП вздовж струмопровідних доріжок може виникати в щільних схемах або в разі використання товстих доріжок. Зазвичай не є критичним дефектом, але якщо дефект доходить до краю плати, то через капілярний ефект друкована плата буде набирати вологу та невдовзі почнеться процес корозії доріжок, які контактують із дефектом. Тому повітряні кармани обмежують використання плат із цими дефектами у вологих умовах.

1.3.6 Потрапляння сторонніх включень

Сторонні включення іноді присутні або у вхідному матеріалі, або під захисним шаром готової плати, чи обидва випадки одночасно [16]. Такі дефекти, як правило, не є приводом для відмови на контролі якості, якщо сторонні включення не знаходяться на місцях критичних загинів, не замикають доріжки та не зменшують відстань між доріжками нижче від зазначеного у технічному завданні рівня.

Перераховані дефекти представлені на рисунку 1.4.

Рисунок 1.4 – Види візуальних дефектів гнучких друкованих плат

1.3.7 Розміри отворів

Розміри отворів перевіряються для того, щоб переконатись у їх відповідності кресленикам та для того, щоб не допустити потенційні проблеми під час складання [16]. Якщо отвори занадто малі, можуть виникати проблеми з розміщенням елементів із наскрізним монтажем. Занадто великі отвори можуть заважати формуванню достатньо надійного паяного з’єднання. У разі розбіжності розмірів монтажних отворів з кресленниками може виникнути проблема з коректним розміщенням модуля.

1.3.8 Розміщення провідників

Завершений шаблон струмопровідних доріжок на гнучкій друкованій платі має точно відображувати кресленик [16].

1.3.9 Ширина провідників і відстань між ними

Ширина провідників і відстань між ними можуть бути критичними в ряді випадків і мають відповідати мінімальним значенням для наданих потужності та напруги. Ці параметри мають точно відповідати технічному завданню та наданим кресленикам.

1.4 Вимоги до тестування фізичних параметрів

Тестування фізичних параметрів гнучких друкованих плат – це ключовий етап контролю якості [17]. Воно допомагає визначити придатність продукту до його ймовірного використання. Ці тести виконують для того, щоб переконатись у якості та надійності готового виробу. Тестування гнучкості є особливо важливими, якщо ГДП буде використовуватись у пристроях, в яких вона буде постійно згибатись і знаходитись у стані руху.

1.4.1 Тест на адгезію покриття

Проводиться тест на адгезію металічного покриття до мідної фольги струмопровідного шару. Недостатньо міцний зв’язок призведе до відшарування покриття та можливої відмови через втрату контакту або коротке замикання.

1.4.2 Тест на міцність контактних майданчиків

Міцність зв’язку контактних майданчиків перевіряється для того, щоб переконатись у тому, що ГДП зможе витримати складання та ремонт без зайвих ушкоджень.

Контактний майданчик має витримати як мінімум 5 циклів запаювання та розпаювання згідно із загальною специфікацією.

1.4.3 Тест на міцність зв’язку контактних доріжок із базовим матеріалом

Цей тест виконується для того, щоб переконатись в тому, що процес виробництва плати не знизив міцність зв’язку фольги з базовим матеріалом до

неприпустимого рівня. Умови тесту в цілому ідентичні з такими тестами для вхідного матеріалу [17].

1.4.4 Тест на гнучкість

Тест на гнучкість виконується для того, щоб переконатись що плата може без проблем приймати форму, задану у вимогах до друкованого модуля та складальному кресленику без розшарування та перелому провідників. Інформація, необхідна для правильного тесту на гнучкість, включає в себе [17]:

– місцезнаходження загину;

– радіус загину;

– кут загину;

– напрям загину;

– кількість циклів згину.

1.4.5 Стійкість до згинів

Цей тип тестування є найважливішим для застосування ГДП в умовах із постійним згинанням плати. Апаратура для тестування може варіюватись у залежності від вимог замовника. Основні види пристроїв для тестування гнучких друкованих плат на стійкість до згину представлені на рисунку 1.5 [17].

1.5 Вимоги до цільності конструкції

Цілісність конструкції ГДП, як правило, оцінюється візуально, за допомогою мікроскопу. Оцінка виконується до та після термічного тестування [18]. Цей тип оцінки виконується для того, щоб виявити присутність мікроскопічних дефектів, котрі можуть вплинути на надійність готового електричного кола. Інспекція до та після термічного тестування виконується для того, щоб переконатись, що якість і надійність не погіршуються через термічний вплив під час паяння та ремонтних операцій [19].

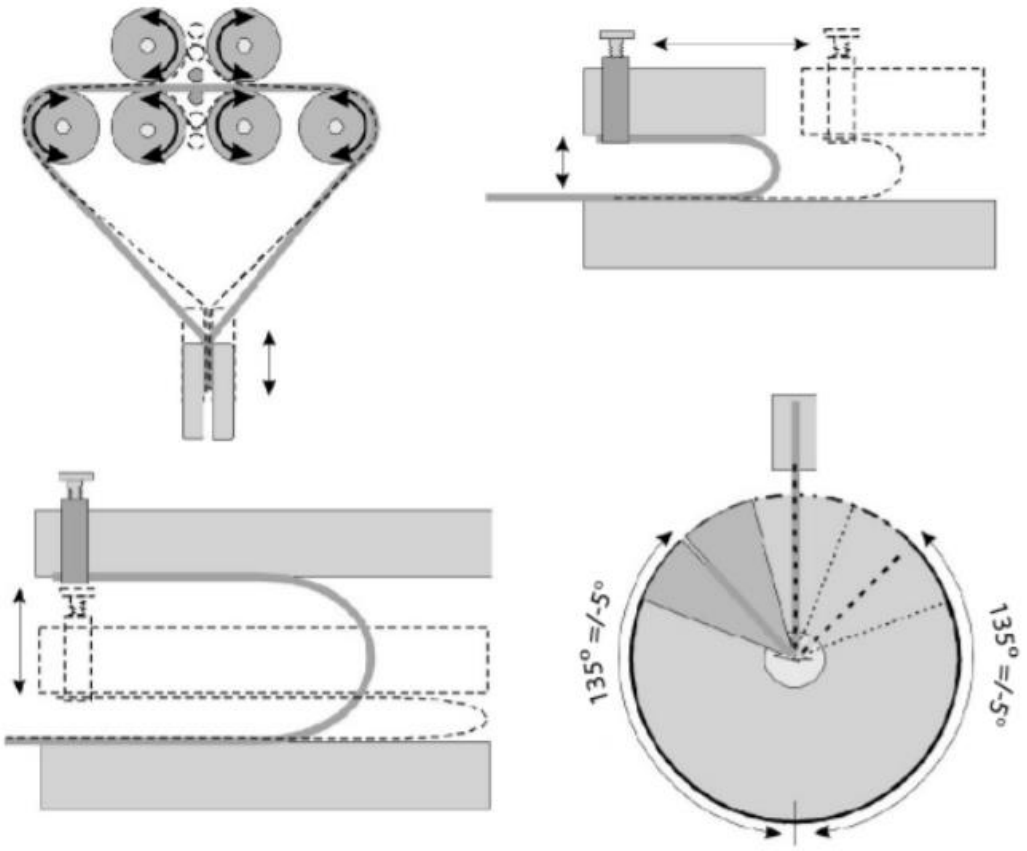

Рисунок 1.5 – Основні типи пристроїв для тестування гнучких друкованих плат на стійкість до згину

Далі будуть описані основні пункти, за якими виконуються дослідження цільності конструкції гнучких друкованих плат, які проводяться до термічного тестування.

1.5.1 Дослідження плат на предмет піднятих майданчиків

Друкована плата перевіряється на присутність піднятих майданчиків для того, щоб переконатись, що вони мають достатню адгезію. Зазвичай відшарування мідних майданчиків і доріжок від основи плати трапляється як результат впливу на неї високих температур, через різність теплових коефіцієнтів лінійного розширення полімерної основи та мідних доріжок [19].

1.5.2 Дослідження плат на цілісність металізації

Цілість металізації оцінюється для того, щоб впевнитись, що якість, однорідність і товщина металізації відповідають специфікації [20], та що відсутні тріщини в металізації отворів – для того, щоб переконатись у її надійності.

1.5.3 Термічний стрес

Цей тест виконується для того, щоб визначити, чи можуть металізовані отвори витримати процес складання та ремонту пристрою. Сам тест являє собою занурення плати у розплавлений припій на 10 секунд за температури 289 °C [21]. Це дає більше термічне навантаження, ніж звичайне складання, та дає більш чітку картину того, як плата витримує високі температури.

1.5.4 Симуляція ремонту

Тест на симуляцію ремонту являє собою запаювання та розпаювання елементу із наскрізним монтажем у металізовані отвори п’ять разів [22]. Цей тест виконується для того, щоб симулювати ремонт готового пристрою та оцінити його вплив на цілісність металізованих отворів [23]. Тест має виконуватись кваліфікованим оператором, тому що в іншому разі можуть бути завдані помітні ушкодження. Всі можливі дефекти, які можуть виявити термічні тести, відображені на рисунку 1.6 [24].

Рисунок 1.6 – Дефекти, які утворюються через температурний вплив

1.5.5 Тест на обриви та короткі замикання

Електричні кола мають бути перевірені на правильність роботи, відсутність коротких замикань та обривів доріжок [25]. Цей тест допомагає переконатись у тому, що плата відповідає проекту та буде правильно працювати. Умови тесту, такі як напруга, що вимагається, мінімальній опір між провідниками, максимальний опір провідників мають бути вказані в головному кресленику.

1.5.6 Тестування на електричний опір ізоляції

Тестування на електричний опір ізоляції повторює таке для вхідного матеріалу та дозволяє переконатись в тому, що під час виробництва цей параметр не став гіршим за проектний.

1.5.7 Тестування на протидію вологості

Під час цього тестування плати циклічно підлягають впливу гарячого та вологого повітря (від 80 % відносної вологості за температури 25 ºС до 98 % відносної вологості за температури у 65 ºС) [26]. Тест виконується для того, щоб переконатись у тому, що не виникає ніяких руйнуючих плату ефектів та що не погіршується ізоляція.

1.6 Висновки до розділу 1

У першому розділі розглянуто основні типи тестування ГДП, їх особливості, вплив різних факторів на їх проведення.

Були описані технології, методи та умови проведення тестування ГДП на предмет механічної стійкості, хімічної стійкості, електричної міцності, стійкості до зовнішніх впливів.

Виділені основні проблеми та недоліки тих чи інших методів тестування ДП. Були розглянуті основні типи дефектів ДП, в яких умовах вони можуть з’являтись та на які параметри готового виробу вони впливають.

2 РОЗРОБКА МЕТОДУ ТЕСТУВАННЯ ГНУЧКИХ ДРУКОВАНИХ ПЛАТ

2.1 Вибір типу тестування ГДП

Серед багатьох методів тестування ГДП найбільш поширені тести, у результаті яких проводиться візуальна інспекція друкованих плат, які тестуються.

Але навіть за допомогою спеціалізованих пристроїв візуальний огляд не завжди дає повну картину отриманих виробом ушкоджень під час проходження тестів.

Розроблюваний метод тестування дозволить оцінювати результати тестів багатошарових гнучких друкованих плат. Для того, щоб оцінити специфіку контролю багатошарових ГДП, треба розглянути структуру типової ГДП. Схематичне зображення ГДП представлене на рисунку 2.1 [26].

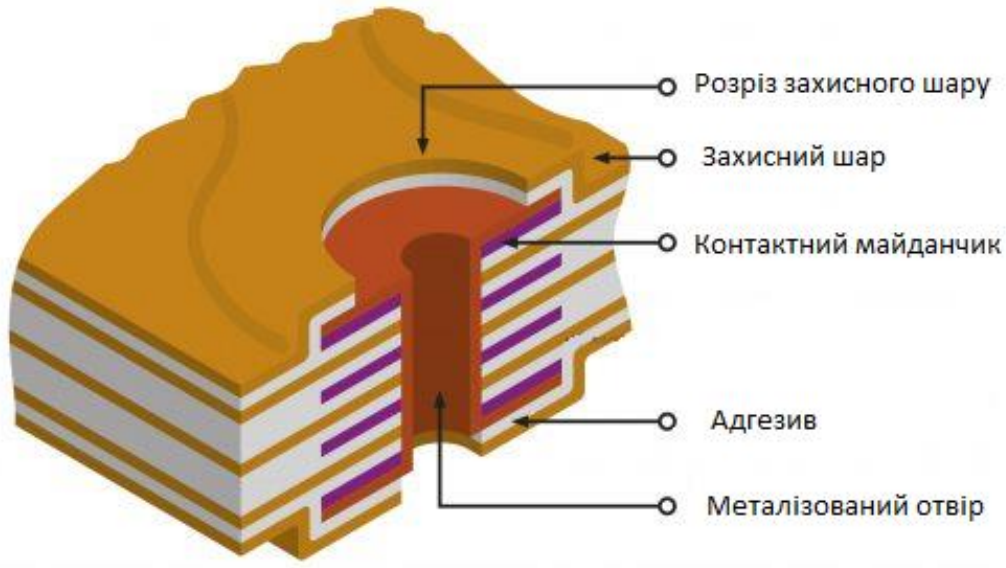

Рисунок 2.1 – Схематичне зображення структури багатошарової ГДП

Як можна побачити, більша частина потенційних точок відмови знаходиться всередині багатошарової структури та є недосяжною для будь-якого візуального огляду.

Для аналізу внутрішніх шарів можна використовувати рентген-апарат, який буде за допомогою рентгенівського випромінювання просвічувати наскрізь всі шари ДП. Рентгенівське випромінювання отримують за допомогою електровакуумного приладу – рентгенівської трубки. З технічної точки зору, рентгенівська трубка – це електронно-вакуумний діод із анодом з тугоплавкого металу та з вікном для виходу рентгенівського випромінювання. Принцип роботи цього пристрою базується на виникненні так званого тормозного випромінювання, коли електрони дуже високих енергій (як правило, не менше за 10 кеВ) різко зупиняються під час зіткнення з матеріалом. Близько 1 % енергії випромінювання виходить у вигляді тормозного рентгенівського випромінювання, все інше – у вигляді тепла. Для роботи такі пристрої потребують напруги не менш за 60 кВ.

Структура рентгенівської трубки представлена на рисунку 2.2.

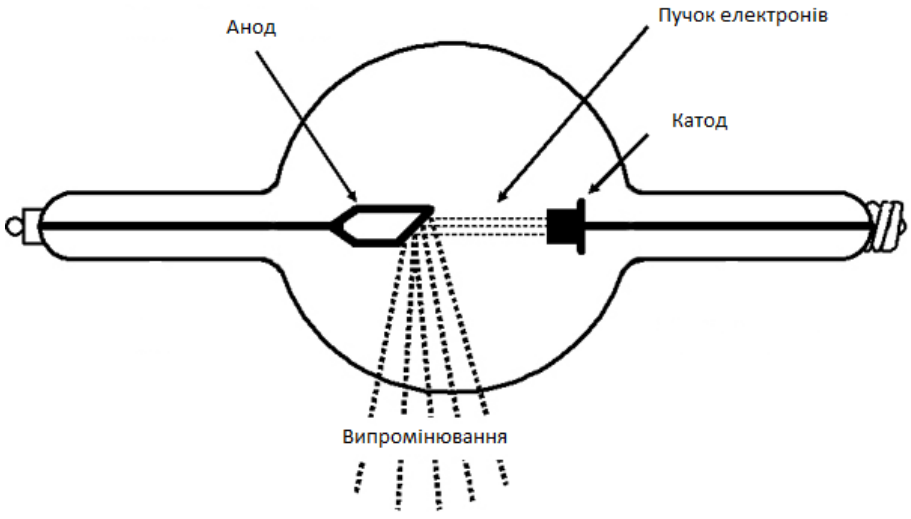

Рисунок 2.2 – Структура рентгенівської трубки

Рентгенографія матеріалу базується на тому, що різні матеріали мають різну ступінь пропускання рентгенівського випромінювання та, виходячи з

цього, відбивають різні тіні у рентгенівському спектрі. Таким чином, на пристрої, що приймає рентгенівське випромінювання, утворюється двомірне зображення, яке являє собою суму зображень всіх шарів досліджуваного предмету. Існує два основних типи приймачів рентгенівського випромінювання: аналогові та цифрові. Аналогові використовують рентген-флуоресцентні екрани та плівку. Під час проходження через досліджуваний об’єкт рентгенівське випромінювання потрапляє на плівку та на флуоресцентний екран за нею. Екран засвічує плівку й отримується зображення ослабленого досліджуваним об’єктом рентгенівського випромінювання. Цей метод потребує проявлення плівки, довгої експозиції, не може бути автоматизований і потребує присутності персоналу. Через це він зовсім не підходить для контролю ГДП [27].

Цифрові приймачі рентгенівського випромінювання базуються на так званих сцинтиляторах і фототранзисторах. Сцинтилятор – це пристрій, який перетворює рентгенівське випромінювання в оптичне у видимому спектрі.

Далі світлове випромінювання реєструється фототранзисторами та за допомогою відповідних схем формується зображення. Схематичне зображення конструкції електронної матриці приймача рентгенівського випромінювання наведено на рисунку 2.3.

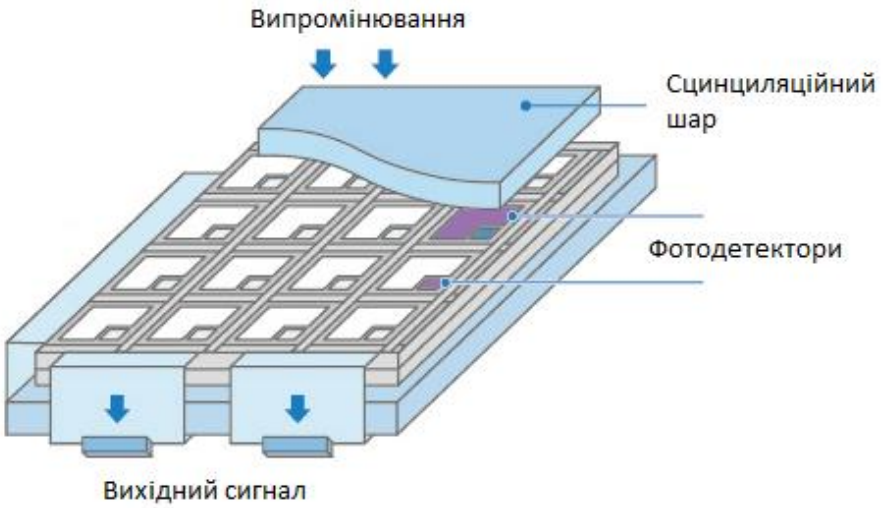

Рисунок 2.3 – Структура матриці приймача рентгенівського випромінювання

Роздільна здатність матриць сучасних електронних приймачів рентгенівського випромінювання досягає 0,1 мм на лінію, тому такий метод прийому сигналу підходить для тестування ГДП. Також із використанням такого типу приймачів можлива повна автоматизація процесу, що виключає людський фактор і вплив іонізуючого випромінювання на працівників, а швидкість роботи дозволяє виконувати навіть вихідний контроль кожного виробу з лінії за допомогою такого пристрою.

Виходячи з того, що готове зображення потрапляє безпосередньо на керуючий комп’ютер, є можливість автоматизації процесу аналізу зображення. Це дозволить значно прискорити процес аналізу та виключити людський фактор. Приклад готового рентгенівського зображення друкованої плати можна побачити на рисунку 2.4 [27].

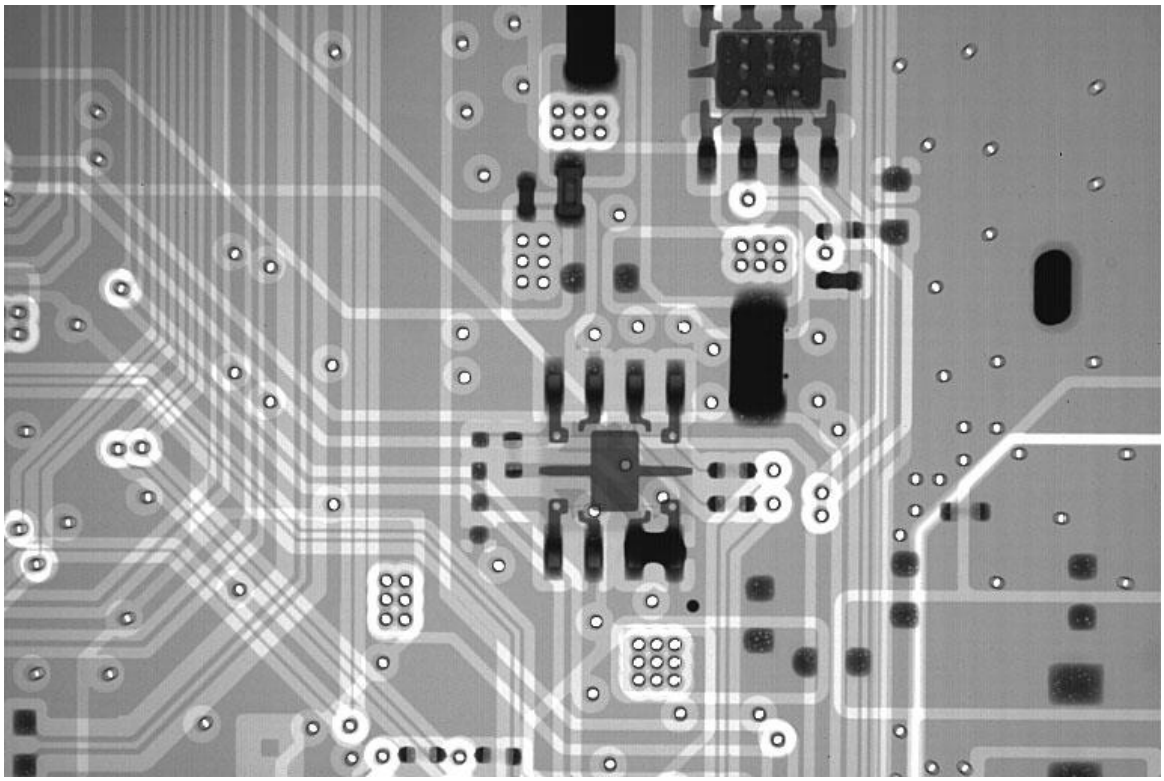

Рисунок 2.4 – Приклад рентгенівського знімку багатошарової друкованої плати

2.2 Вибір методу автоматичного аналізу результатів тестування

Маючи на увазі вимоги до швидкості та точності розпізнавання дефектів, може бути використано два методи аналізу зображення – штучні нейронні мережі або класифікатор, який буде виділяти контури фактичного зображення друкованих провідників та підготовлені проектні ескізи, яким має відповідати готовий виріб.

Класифікатор базується на принципі виділення контурів заздалегідь бінаризованого зображення та порівнює його із заданим вхідним зображенням. Якщо зображення не відповідає заданому, виріб маркується як дефектний і відправляється на додаткову експертизу.

Із використанням штучної нейронної мережі принцип аналізу кардинально відрізняється. За її допомогою виділяються дефекти на платі без порівняння зі зразковим шаблоном.

Штучна нейронна мережа – це набір алгоритмів, який починає розпізнавати взаємовідносини у наборі даних за допомогою процесу, який за своїм принципом дії схожий із тим, як працює мозок людини. У цьому сенсі нейронні мережі – це система нейронів, живих чи штучних. Нейронні мережі можуть адаптуватись до зміни вхідних даних і можуть видавати оптимальний результат без потреби у зміні вихідного критерію [28].

Штучні нейронні мережі складаються з пов’язаних між собою вузлів, які також називають нейронами. Кожний штучний нейрон має багато входів і створює один вихідний сигнал, який може відправити на численні входи інших штучних нейронів [28]. На входах можуть бути будь-які зовнішні дані чи вихідні сигнали з інших нейронів.

Вихідні дані з фінальних, вихідних нейронів виконують завдання, такі як, наприклад, розпізнавання об’єктів на зображенні. Структура штучного нейрона наведена на рисунку 2.4.

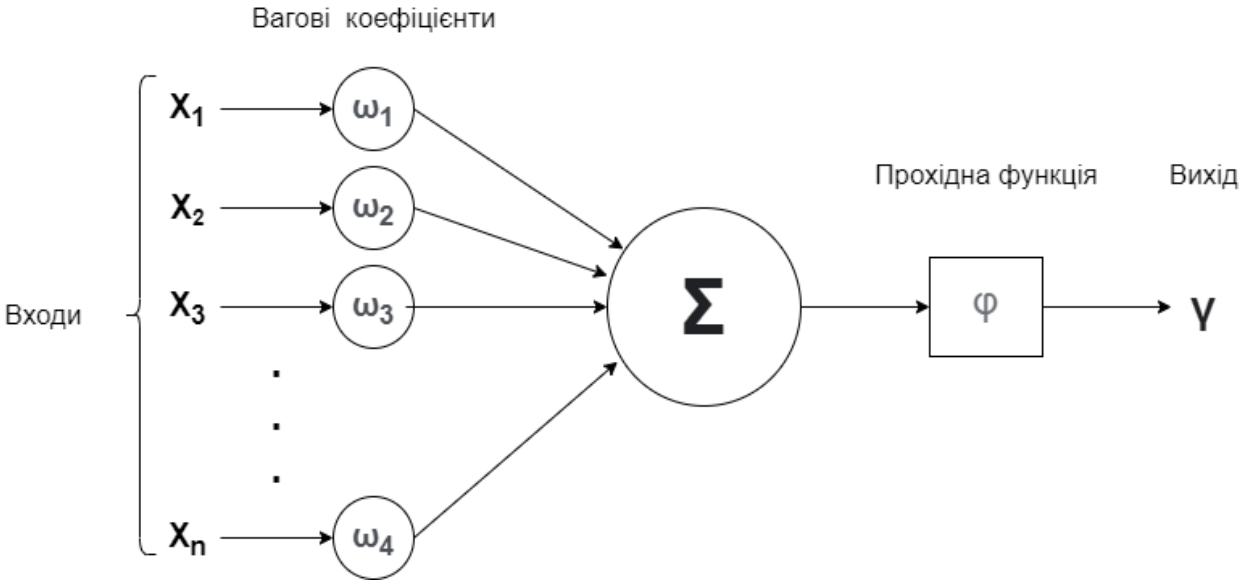

Рисунок 2.4 – Структурна схема штучного нейрона

Вихідний сигнал отримується у вигляді суми вхідних сигналів із ваговими коефіцієнтами та у результаті виконання прохідної функції за формулою (2.1):

$$\varphi = \sum_{i=1}^{m} x_i w_i \,, \tag{2.1}$$

де m – кількість входів нейрона;

x_i – значення і-го входу;

w_i – вага і-го входу.

За допомогою з’єднання таких вузлів утворюють штучні нейронні мережі. Вони використовуються в усіх сучасних технологіях, пов’язаних зі штучним інтелектом. Штучні нейронні мережі можуть використовуватись у дуже широкому спектрі задач, таких як обробка текстів, зображень і звуку.

Штучні нейрони базуються на принципі порівняння вхідних сигналів зі вказаною функцією [29].

Зазвичай у штучних нейронних мережах використовують як прохідну (активаційну) функцію ступінчасту функцію, лінійну та сигмоїду, формула (2.2), через їх реалістичність відносно ступінчастої та лінійної функції:

$$f(x) = \frac{1}{1 - e^{-ax}}. \tag{2.2}$$

За умови збільшення керуючого параметру сигмоїда наближається до ступінчастої функції, а за умови зменшення – до лінійної. Приклад сигмоїди представлений на рисунку 2.5 [29].

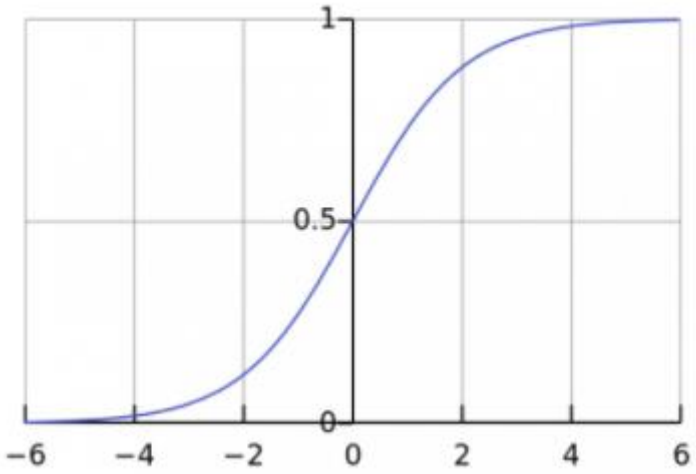

Рисунок 2.5 – Приклад графіку сигмоїди

Для навчання подібних нейронних мереж потрібна велика кількість вхідних даних із відомими результатами, які має отримати мережа.

2.3 Згорткова нейронна мережа

Найкращі результати у галузі розпізнавання та класифікації візуальних об’єктів показує згорткова нейронна мережа, яка є логічним розвитком таких архітектур нейронних мереж, як когнітронна та неокогнітронна [30]. Згорткові нейронні мережі забезпечують відносну стійкість до зміни масштабу, поворотів, зміщень та інших спотворень зображення. Згорткова нейронна мережа бере вхідне зображення, обробляє зображення та класифікує його між певними категоріями. Апаратно-програмний комплекс, який використовує нейронні

мережі, представляє вхідне зображення як масив пікселів, що залежить від роздільної здатності зображення [30]. Наприклад, кольорове зображення розміром 100 × 100 пікселів буде являти собою масив 100 × 100 × 3, а таке ж чорно-біле зображення буде являти собою матрицю 100 × 100 × 1. Структурна схема матриці кольорового зображення представлена на рисунку 2.6.

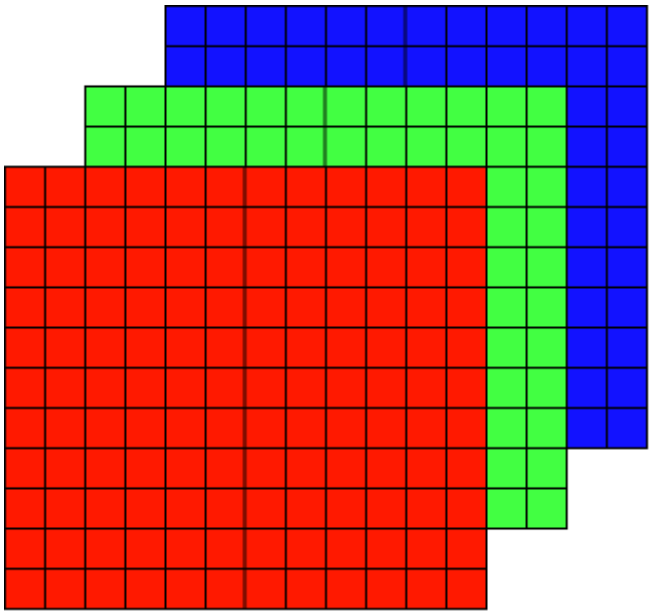

Рисунок 2.6 – Схема RGB-матриці кольорового зображення

З технічної точки зору, звичайні нейронні мережі не можуть повноцінно масштабуватись для того, щоб працювати з повнорозмірними зображеннями. Наприклад, у датасеті CIFAR-10 всі зображення являють собою масиви 32 × 32 × 3 (32 пікселі у ширину, 32 пікселі у висоту, 3 кольори), та один повністю зв’язаний нейрон вупершому скритому шарі звичайної нейронної мережі буде мати 32 × 32 × 3 = 3072 вагових коефіцієнти.

Ця кількість ще залишається можливою для обробки, але структура повністю зв’язаних нейронних мереж не може бути ефективно масштабованою до більших обсягів вхідних даних.

Кольорове зображення з більш прийнятною роздільною здатністю, наприклад 1200 × 1200 призведе до нейронів, які мають 1200 × 1200 × 3 = 4320000 вагових коефіцієнтів. Більш того, для підвищення швидкості обробки

знадобиться велика кількість таких нейронів, що призведе до необмеженого зростання зв’язків та буде потребувати дуже великих ресурсів для виконання завдання. Згорткові нейронні мережі проектуються маючи на увазі той факт, що на вході будуть саме зображення, тому їх архітектура будується належним чином. А саме, на відміну від звичайних нейронних мереж, шари згорткової нейронної мережі мають нейрони, розташовані у трьох вимірах: у висоту, у довжину та у ширину.

Згорткові нейронні мережі об’єднують у собі три архітектурні ідеї для забезпечення інваріантності до зміни масштабу, поворотів та інших спотворень, а саме [31]:

– локальні рецепторні поля;
– загальні синаптичні коефіцієнти;
– ієрархічна організація з просторовими підвибірками.

На даний момент згорткові нейронні мережі та їх модифікації вважаються найкращими алгоритмами знаходження об’єктів.

Згорткова нейронна мережа складається з різних видів шарів, а саме зі загорткових (convolutional) шарів, субдискретизуючих (subsampling) шарів і звичайної нейронної мережі, персептрона. Топологію такої мережі можна побачити на рисунку 2.7 [32].

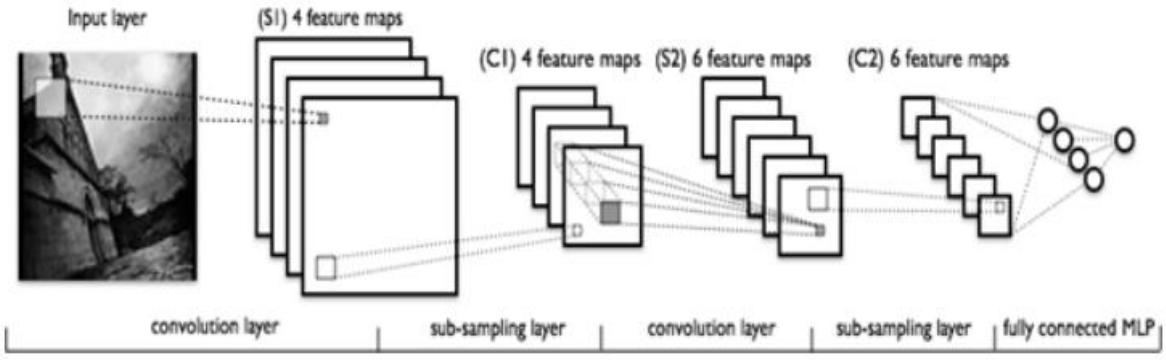

Рисунок 2.7 – Структура згорткової нейронної мережі

Перший тип шарів згорткової нейронної мережі – це згортковий шар. Згортковий шар виділяє елементи у вхідному зображенні. Цей шар називається згортковим через його принцип роботи.

Згортковий шар базується на принципі так званої «згортки» матриці зображення. Під час цього процесу виділяються та запам’ятовуються відношення між пікселями за допомогою вивчення особливостей зображення за допомогою використання невеликих квадратів вхідних даних. З технічної точки зору, згортка зображення являє собою математичну операцію між зображенням і фільтром, який ще називають кернелом, а саме суму добутків найближчих пікселів вхідного зображення на матрицю вагових коефіцієнтів кернела, видаючи на виході двомірну матрицю з коефіцієнтами відповідності зображення фільтру (рисунок 2.8)

Важливо відзначити, що кількість каналів фільтра має співпадати з кількістю каналів вхідного зображення.

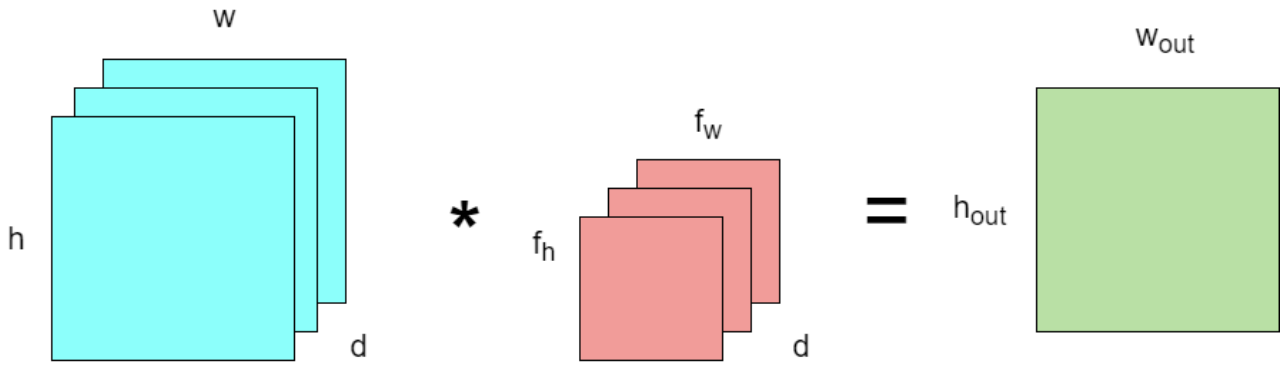

Рисунок 2.8 – Схематичне зображення процесу згортки кольорового зображення

У процесі згортки особливості на зображені знаходяться за допомогою послідовного проходження фільтра (кернела) по матриці зображення із визначеним кроком (страйдом).

Страйд – це кількість пікселів, на яку зміщується кернел за один крок. Якщо страйд дорівнює одному, то під час згортки за один крок кернел зміщується на один піксель [32].

Принцип згортки чорно-білого зображення за допомогою кернела, який визначає нахилені вліво лінії, зі страйдом у 1 піксель покроково відорбражено на рисунку 2.9.

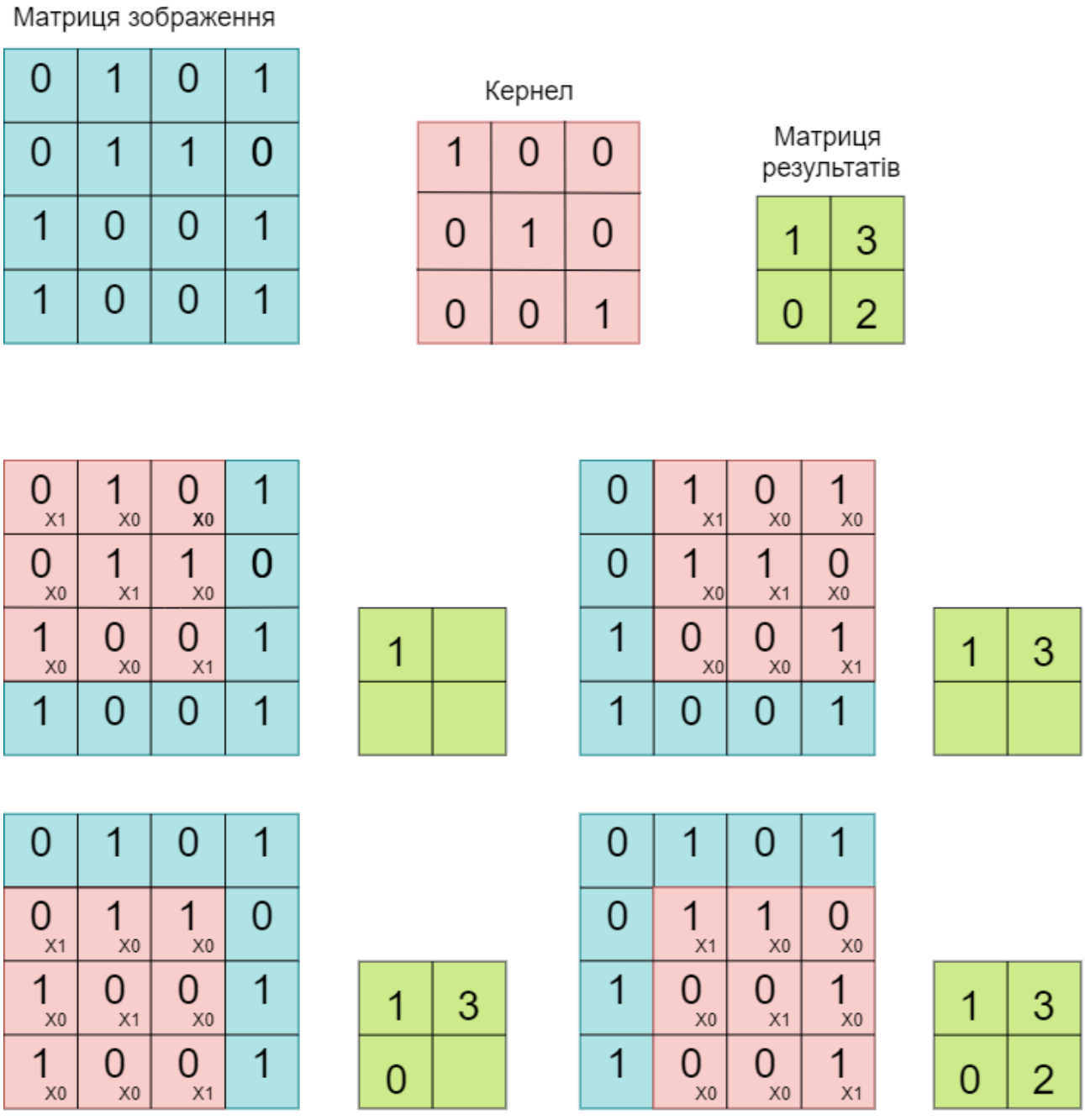

Рисунок 2.9 – Схематичне зображення процесу згортки матриці чорно-білого зображення розміром 5 × 5 пікселів із кернелом 3 × 3 і страйдом 1

На цьому зображенні добре видно, що за допомогою виставлених на кернелі вагових коефіцієнтів ця функція знаходить на матриці вхідного

зображення визначені особливості, у даному прикладі матриця фільтру схематично відображала кернел, який виділяє на зображенні нахилені вліво лінії.

Розмір вхідної матриці зазвичай є значно більшим, але розмір у матриці зображення у прикладі є мінімально достатнім для того, щоб відобразити принцип згортки зображення. Взагалі, існує цілий ряд спеціалізованих фільтрів, котрі за використання під час згортки будуть певним чином впливати на результат, наприклад знаходити краї елементів, підвищувати та знижувати різкість зображення. Основні типи кернелів і їх вплив на зображення після згортки зображено на рисунку 2.10 [32].

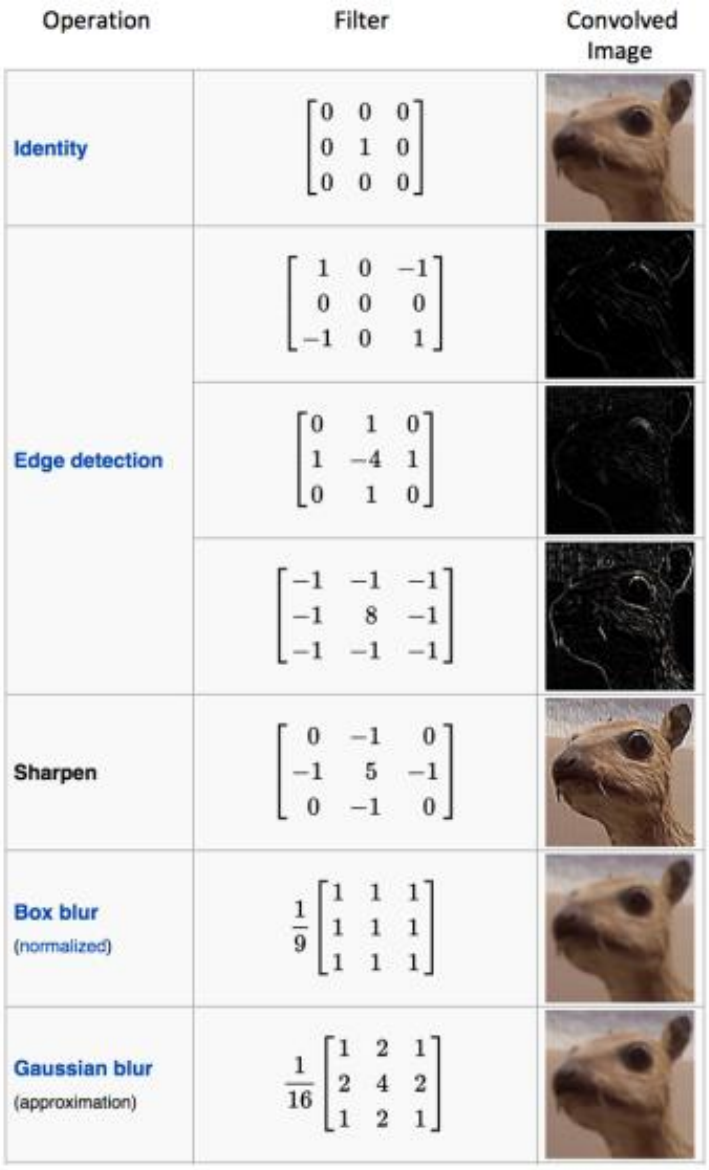

Рисунок 2.10 – Основні типи кернелів

Також необхідними елементами для коректного розпізнавання елементів є нелінійність і зміщення.

Нелінійність і зміщення у кернелі дозволяють виконати підбір вагових коефіцієнтів більш ефективно. Зміщення являє собою фіксоване значення, яке додається до параметрів фільтра. Нелінійність – це активаційна функція. Завдяки її використанню результат згортки отримує різні спотворення, які дозволяють нейронній мережі більш чітко виділяти елементи зображення.

У згорткових нейронних мережах як активаційна функція дуже часто використовується так звана ReLu (Rectified Linear Unit), графік якої зображений на рисунку 2.11.

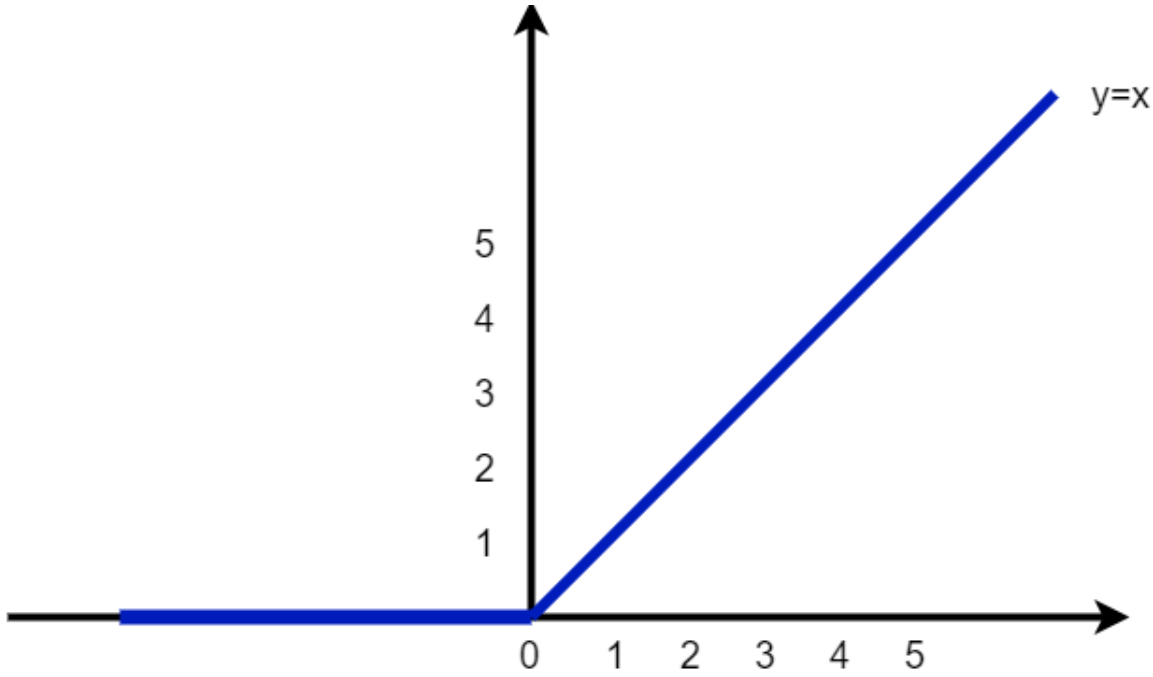

Рисунок 2.11 – Графік лінійної активаційної функції

Під час використання даної функції всі вхідні значення, що дорівнюють нулю або менші за нуль, обертаються в нуль, а значення, вищі за нуль, залишаються незмінними. Приклад результату використання такої активаційної функції відображено на рисунку 2.12.

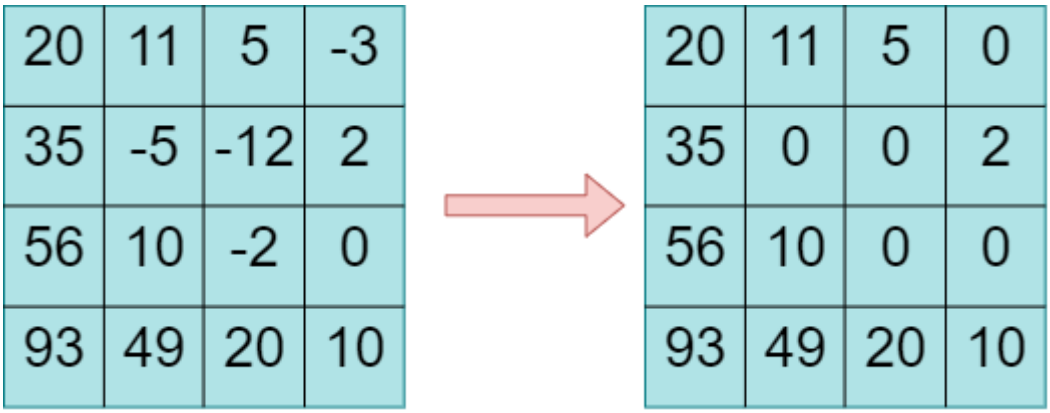

Рисунок 2.12 – Приклад перетворення матриці результатів
за допомогою лінійної активаційної функції

Зазвичай у згорткових нейронних мережах використовується більш ніж один фільтр. У цьому разі вихідні результати кожного з них збираються вздовж визначеної осі, що у результаті дає тривимірну таблицю оброблених даних.

Також для того, щоб пришвидшити процес навчання, та для того, щоб зменшити навантаження на обчислювальну техніку, виконується так званий даунсемплінг вхідних і проміжних даних. Найбільш розповсюджений алгоритм даунсемплінга називається max pooling, або максимальне об’єднання. Операція максимального об’єднання являє собою процес дуже східний до процесу згортування, але з важливими відмінностями. З програмної точки зору, процес максимального об’єднання – це прохід по зображенню фільтра, який знаходить найбільший піксель у своєму полі зору. Страйд цього фільтру може бути відмінним від одиниці та залежить від того, як сильно треба зменшити об’єм даних.

Завдяки алгоритму max pooling зменшується кількість пікселів, що сприяє зменшенню математичних операцій, які потрібно виконати обчислювальній машині, та виходячи з цього, економії обчислювальних ресурсів.

Операція максимального об’єднання також зменшує несуттєві деталі зображення та концентрує нейронну мережу на дійсно важливих елементах зображення, що дозволяє максимально уникнути такого негативного моменту як

перенавчання нейронної мережі. Схематичне зображення принципу роботи максимального об’єднання наведено на рисунку 2.13.

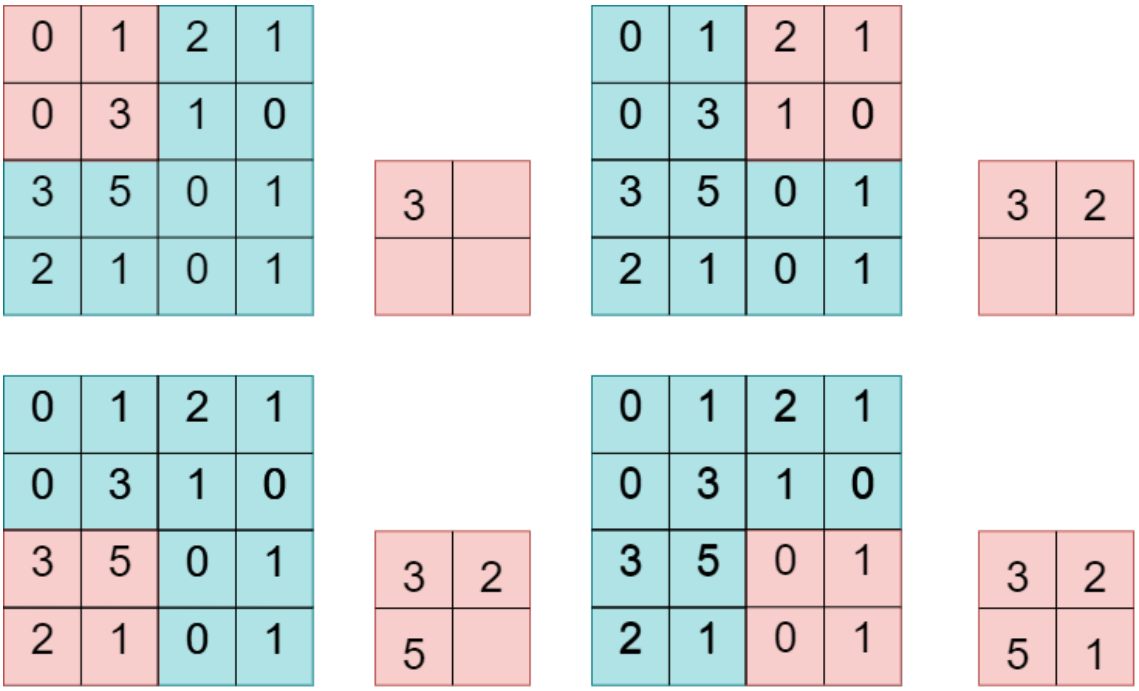

Рисунок 2.13 – Зображення роботи циклу максимального об’єднання

Після ряду згорткових шарів і блоку даунсемплінга матриця результатів обчислення має бути перетворена у єдиний вектор, побудований із послідовно розташованих даних кожного ряду матриці (рисунок 2.14).

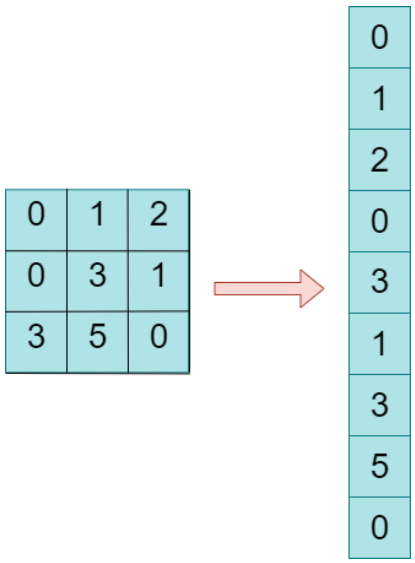

Рисунок 2.14 – Приклад перетворення матриці зображення на вектор

Цей вихідний вектор значень далі передається у так званий повнозв’язний шар, який працює подібно до звичайних нейронних мереж. Кожний елемент ряду вихідних значень перемножується на ваговий коефіцієнт, далі ці добутки сумуються та після цього перетворюються за допомогою прохідної функції. Ця послідовність відтворюється на рисунку 2.4, де зображена структура штучного нейрона. Треба зазначити, що повнозв’язний шар має дуже складну конструкцію через велику кількість зв’язків, яка є добутком перемноження кількості вхідних значень на кількість нейронів. Наприклад, якщо передати 1000 значень у повнозв’язний шар, який нараховує у своєму складі 150 нейронів, то кількість зв’язків дорівнюватиме 1000 × 150 = 150000.

Схематичне зображення повнозв’язного шару представлене на рисунку 2.15 [32].

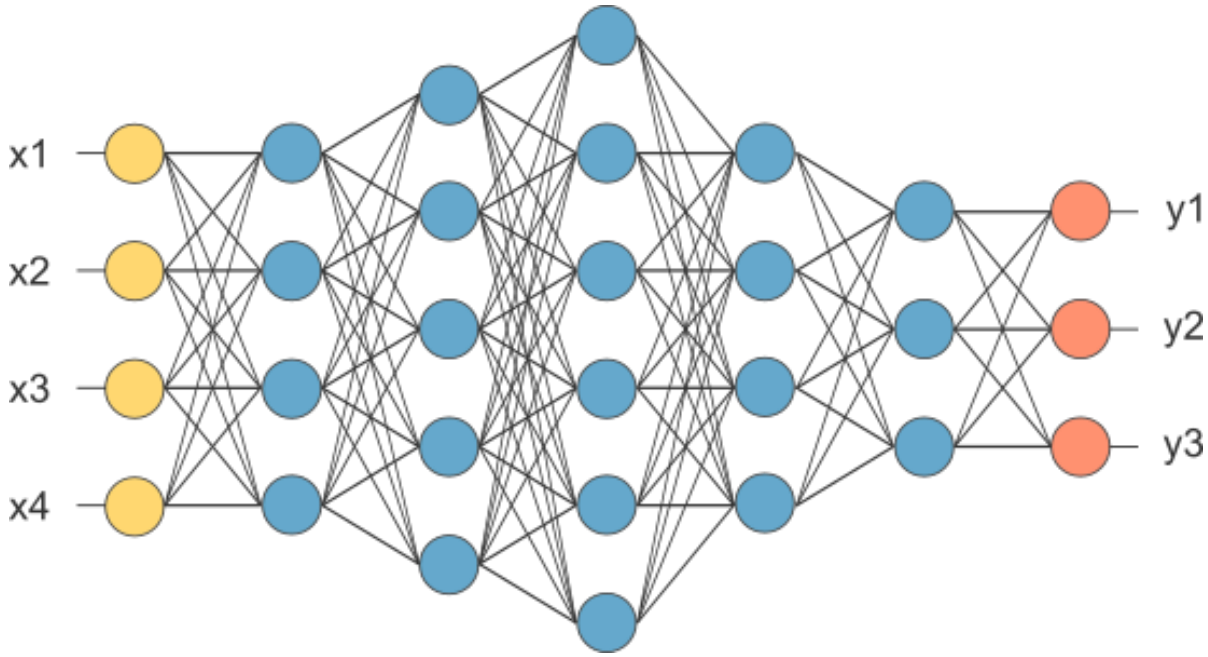

Рисунок 2.15 – Схематичне зображення повнозв’язного щару згорткової нейронної мережі

За допомогою повнозв’язного шару особливості зображень комбінуються для того, щоб утворити модель. Далі ці дані потрапляють у вихідний шар, у якому визначається належність вихідної моделі деякому образу. Ця операція

виконується за допомогою прохідної функції, наприклад сигмоїди, графік якої був представлений на рисунку 2.5.

Для того, щоб отримати результат, кількість нейронів у вихідному шарі має відповідати кількості класів. Структурна схема згорткової нейронної мережі представлена на рисунку 2.16 [32].

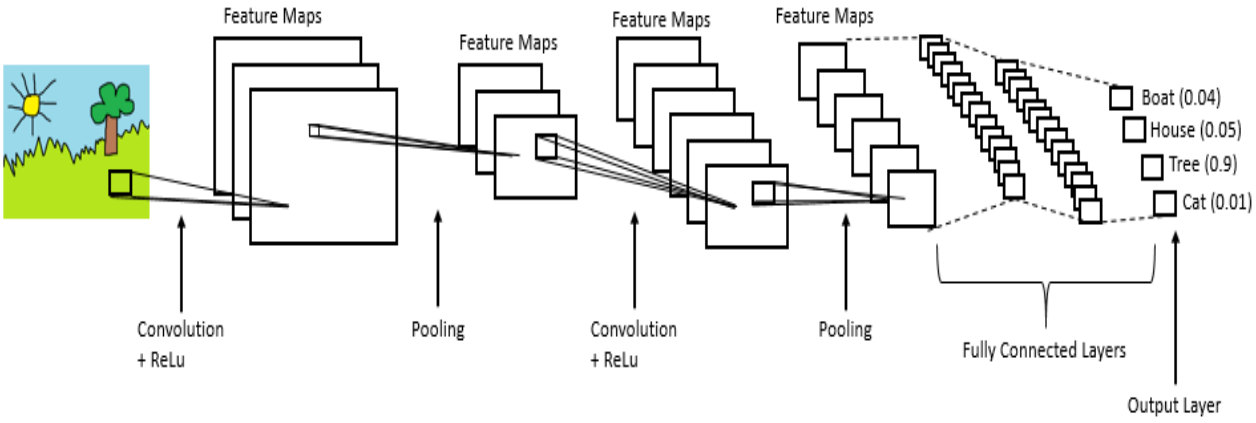

Рисунок 2.16 – Структура згорткової нейронної мережі

2.4 Вибір архітектури системи розпізнавання зображень

Виходячи з матеріалів, які були оглянуті, було прийняте рішення не використовувати ніяких готових рішень і розробити нейронну мережу, використовуючи відкриті бібліотеки, такі як Google TensorFlow.

Також для порівняння слід розробити простий класифікатор зображень, який буде порівнювати вхідне зображення зі зразковим і знаходити різницю. Під час розробки також будуть використані відкриті бібліотеки OpenCV та NumPy.

TensorFlow – це платформа для машинного навчання з відкритим вихідним кодом. Він має всебічну, гнучку систему інструментів, бібліотек та інших ресурсів, що надає можливість для дослідників і розробників будувати та розгортати програми, які базуються на машинному навчанні [33].

TensorFlow був розроблений інженерами компанії Google у рамках організації Google’s Machine Intelligence Research для того, щоб впровадити машинне навчання та дослідження глибоких нейронних мереж.

TensorFlow надає стабільні програмні інтерфейси для таких мов програмування, як C++ та Python, та зворотно сумісний із рядом інших мов програмування.

OpenCV – це програмна бібліотека для систем комп’ютерного зору та систем машинного навчання з відкритим вихідним кодом. OpenCV був створений для того, щоб впровадити загальну інфраструктуру для використання комп’ютерного зору та для того, щоб прискорити використання машинного навчання у комерційних продуктах. Ця бібліотека має у своєму складі більш ніж 2500 оптимізованих алгоритмів, які включають у себе різноманітні набори для систем комп’ютерного зору та для машинного навчання. Ці алгоритми можуть бути використані для того, щоб розпізнавати обличчя, ідентифікувати об’єкти, класифікувати рухи людей на відео, слідкувати за рухами камери та рухомих об’єктів у кадрі, виділяти тривимірні моделі об’єктів [34].

Бібліотека OpenCV має програмні інтерфейси для C++, Python, Matlab і підтримує більшість сучасних операційних систем. Бібліотека написана на мові C++.

NumPy – це програмна бібліотека з відкритим вихідним кодом, створена для математичних обчислень на мові Python [35].

Основна проблема інтерпретованих мов програмування, таких як, наприклад, Python, полягає у швидкості роботи алгоритмів, яка є значно нижчою за швидкість таких алгоритмів на компільованих мовах програмування, наприклад, C/C++. Бібліотека NumPy вирішує цю проблему за допомогою реалізації алгоритмів, які оптимізовані для роботи з багатомірними масивами.

2.5 Висновки до розділу 2

У другому розділі на базі аналізу існуючих методів контролю ГДП запропонований новий метод контролю якості ГДП, який поєднує в собі використання рентген-апарату та аналіз рентгенограм за допомогою спеціальних класифікаторів на базі порівняння зображень і класифікаторів, які використовують для аналізу нейронні мережі.

Були розглянуті згорткові нейронні мережі, основні їх елементи, шари, типи активаційних функцій та їх вплив на проміжні результати. Обрана та визначена архітектура програм, які будуть розроблені для реалізації запропонованого методу контролю ГДП.

3 РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

Як зазначалося раніше, для порівняння та більш всебічного та точного аналізу, крім самої нейронної мережі, буде розроблена програма, яка аналізує зображення за допомогою порівняння вхідного зображення зі зразковим. Іншими словами, це буде класифікатор зображень, який буде знаходити різницю між двома зображеннями, виділяти її, показувати її розташування та підраховувати її площу. Даний програмний комплекс буде порівняний з нейронною мережею та буде прийняте рішення відносно застосовності й ефективності таких програм.

3.1 Вибір мови програмування

На сьогодні у сфері системи машинного навчання та комп'ютерного зору використовуються дві основні мови програмування, а саме C++ та Python.

Python відрізняється відносною простотою програмування, не має суворої типізації та саме для мови Python є найбільша кількість необхідних бібліотек.

Інтерпретатори Python існують і підтримуються практично для будь-яких сучасних архітектур процесорів, таких як AMD64, IA64, IA32, ARM64, PowerPC, RISC, CISC, MISC, VLIW [36].

Python – це інтерпретована мова програмування, тому вона не потребує компіляції коду, що дозволяє більш зручно розробляти програмний продукт, втрачаючи менше часу на пошук та усунення недоліків кодової бази.

Архітектура мови Python має на увазі повну динамічну типізацію й автоматичне керування пам'яттю. Це дозволяє писати більш просту кодову базу, але може призводити до ситуацій, в яких утворюються витоки пам'яті, які доволі складно знайти та усунути. Також через свою специфіку Python працює відносно повільніше за той же C++, але за допомогою використання зовнішніх бібліотек на C/C++ можна наблизитись за швидкістю до компільованих мов програмування.

C++ – це компільована мова програмування, яка базується на синтаксисі мови C, але структурно більш комплексна та створювалася для об’єктно-орієнтованого програмування. З архітектурної точки зору, C++ має сувору статичну типізацію та має можливість безпосередньої роботи з регістрами пам’яті. Таким чином, C++ має риси як низькорівневої мови програмування, так і високорівневої.

Для розробки необхідних програмних модулів була обрана мова програмування Python через велику кількість зручних бібліотек і фреймворків. Відставання у швидкості і даному разі не є критичним.

3.2 Вибір робочого оточення

Розробка програмного продукту буде виконуватись на комп’ютері під керуванням операційної системи Linux, що певним чином звужує ряд середовищ розробки через те, що кросплатформових рішень існує значно менше, ніж рішень, розроблених для операційної системи Microsoft Windows.

На даний момент існує два основних кросплатформових рішення, у яких є можливість повноцінної розробки програм на мові Python, а саме Microsoft Visual Studio Code та JetBrains PyCharm.

Microsoft Visual Studio Code – це середовище розробки з відкритим вихідним кодом, створене для розробки на більшості популярних мов програмування, серед плюсів можна відзначити те, що він розповсюджується за безкоштовною ліцензією та за допомогою безкоштовних додатків функціонал цього середовища може бути розширеним для роботи зі специфічними мовами програмування, фреймворками та файлами. Також серед плюсів легковісність середовища, що також дуже важливо через те, що обчислювальний ресурс обчислювальної машини, на якій буде виконуватись розробка програмного продукту, дуже обмежений.

Серед недоліків даного середовища можна відзначити значну кількість недопрацьованих елементів, місцями нестабільну роботу, низьку інтеграцію з

системами контролю версій, незручний емулятор терміналу та примітивний налагоджувач коду. Інтерфейс користувача Microsoft Visual Studio Code представлений на скриншоті на рисунку 3.1.

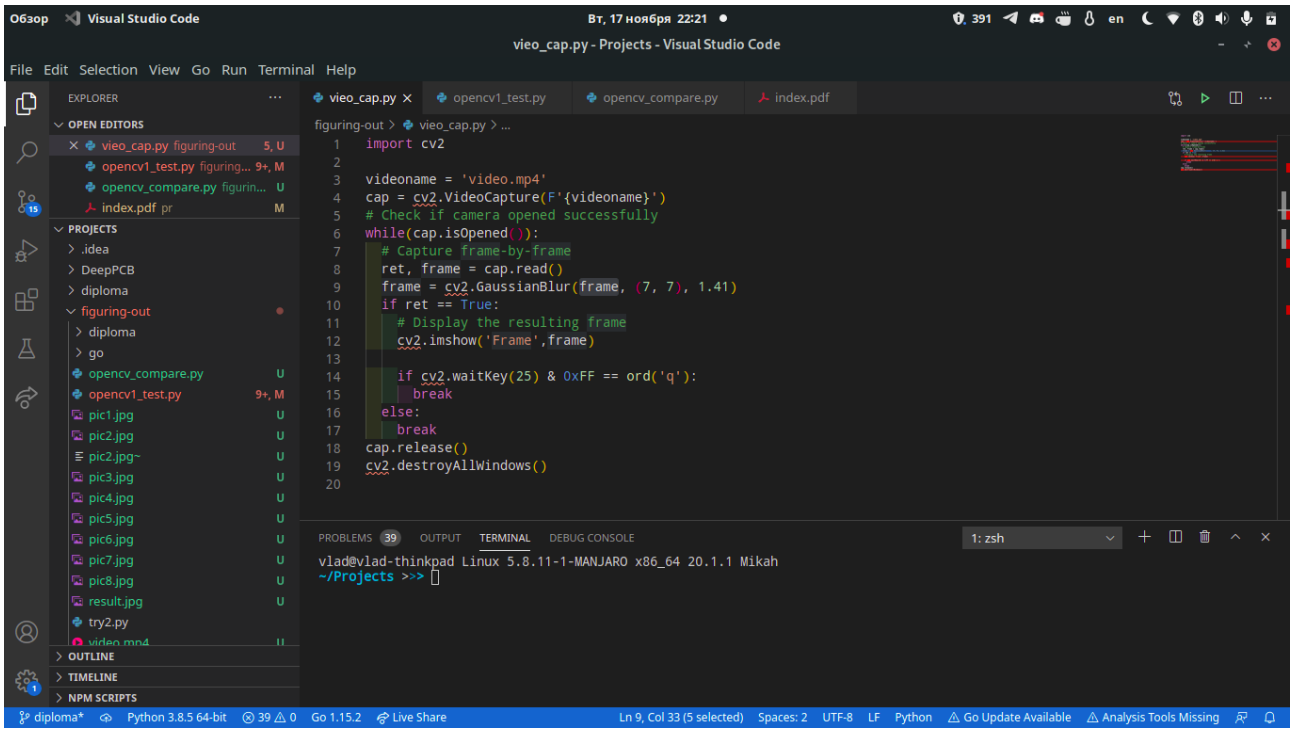

Рисунок 3.1 – Інтерфейс користувача MS VSCode

JetBrains Pycharm – це середовище розробки, спеціально спроектоване для розробки програм на мові програмування Python, розроблене компанією JetBrains. Розповсюджується платно та має закритий вихідний код. Pycharm має безкоштовну версію, PyCharm Community Edition, яка розповсюджується безкоштовно та значно поступається у функціоналі повноцінному PyCharm.

Pycharm має дуже широкий функціонал для аналізу та відлагодження коду, підтримує веб-розробку на Django, має інструменти для юніт-тестів. Також це середовище має дуже обширну бібліотеку модулів для підтримки різноманітних фреймворків, інструментів і типів файлів. Також Pycharm має дуже зручний термінал і глибоку інтеграцію з системами контролю версій, такими як Git.

Серед недоліків Pycharm можна відзначити високу вартість приватної ліцензії та високі вимоги до апаратної платформи, на якій це середовище

використовується. Безкоштовна версія поступається за функціоналом навіть не спеціалізованому на мові програмування Python Microsoft Visual Studio Code.

Скріншот інтерфейсу користувача PyCharm представлений на рисунку 3.2.

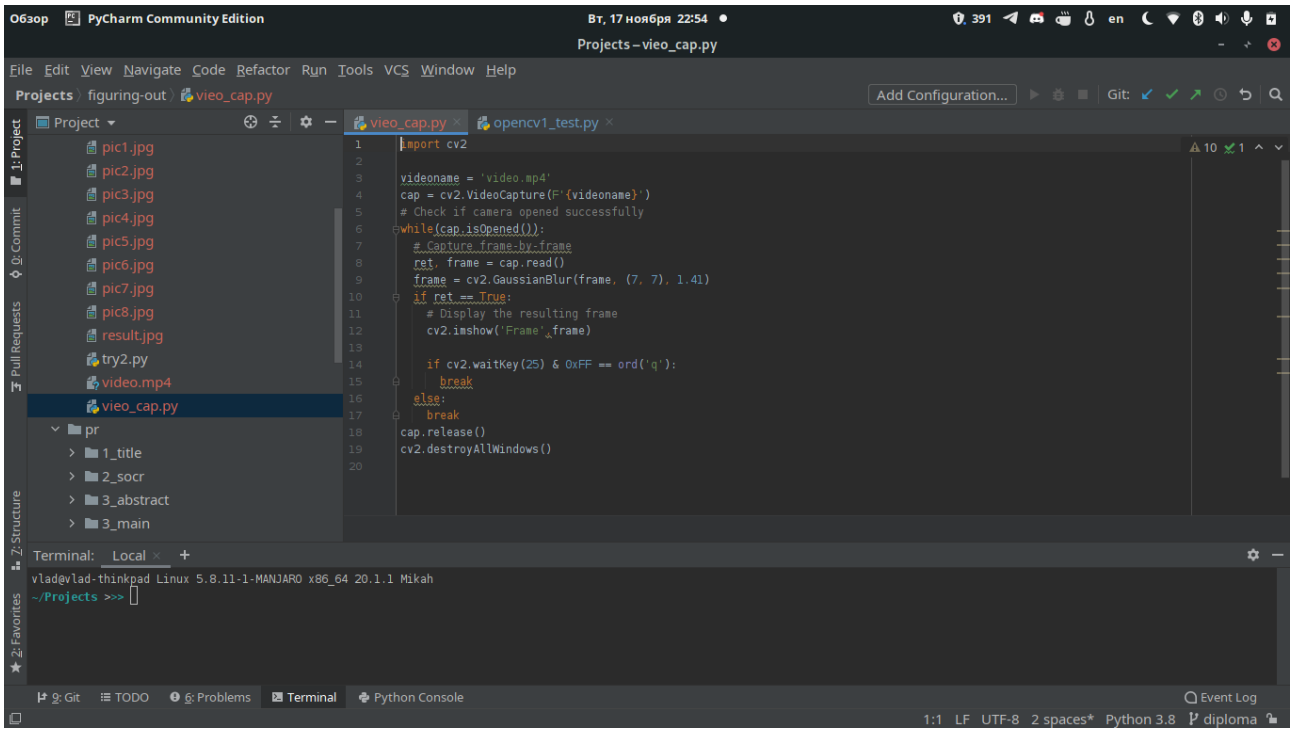

Рисунок 3.2 – Інтерфейс користувача середовища PyCharm

Таким чином, було прийняте рішення використовувати Microsoft Visual Studio Code через те, що він має відкритий вихідний код і створює значно менше навантаження на обмежений апаратний ресурс обчислювальної машини.

3.3 Вибір системи контролю версій

Як система контролю версій була обрана розподілена система Git, яка була розроблена Лінусом Торвальдсом для ведення розробки ядра Linux.

Git написаний на мові C та постачається у комплекті практично з будь-якими десктопними дистрибутивами Linux. Як віддалений репозиторій під час розробки буде використовуватись Github.

3.4 Розробка програми-класифікатора зображень

Розроблювана програма повинна мати можливість зчитувати два зображення – зображення-приклад, у даному випадку шаблон струмопровідного шару друкованої плати, та зображення, яке отримується з фактичної плати у результаті рентгенограми.

Спочатку треба отримати вхідне зображення та відділити тільки важливі елементи. Якщо мати на увазі те, що вхідні зображення – це рентгенівські знімки, то на зображенні буде багато тіней і нечітких границь. Для того, щоб виділити необхідні межі струмопровідних доріжок, требо перевести отримане зображення у чорно-білий формат і провести бінаризацію.

Бінаризація – це базовий метод підготовки зображення. Він полягає у перетворенні зображення за допомогою ступінчастої функції, графік якої представлено на рисунку 3.3.

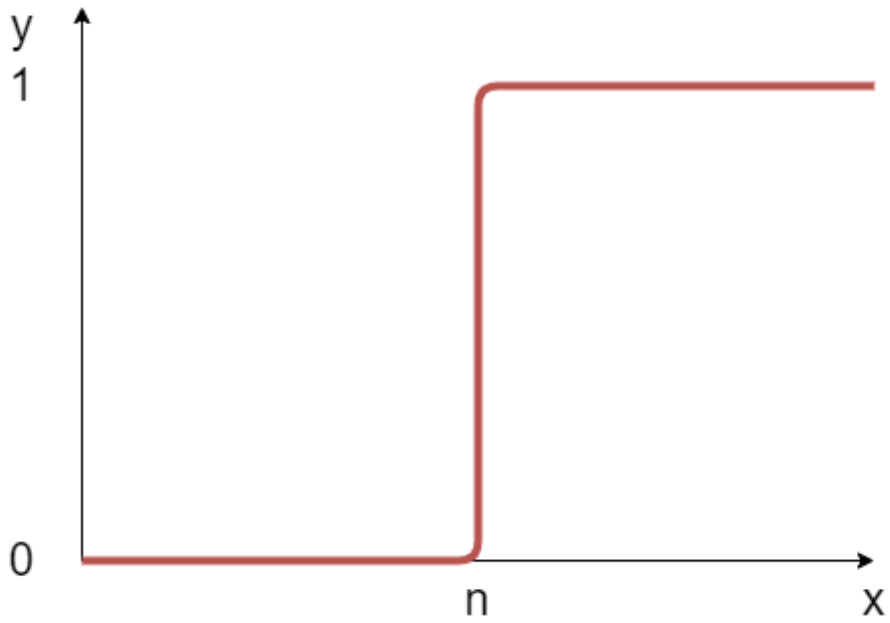

Рисунок 3.3 – Графік функції бінаризації зображення

Основний параметр такої функції – це порогове значення (threshold), позначене літерою n на рисунку 3.3. Якщо значення білого на окремому пікселі

є більшим за це порогове значення, то піксель приймає значення 1, якщо нижчим, то 0. Відповідно піксель стає або чорним або білим.

У бібліотеці OpenCV така функція називається cv2.threshold() та приймає значення мінімального та максимального порогу, типу порогу та повертає зображення після бінаризації.

Для тестування було обране зображення, яке добре імітує рентгенівський знімок тому, що має струмопровідні доріжки зі змазаними границями. Тестове зображення представлене на рисунку 3.4.

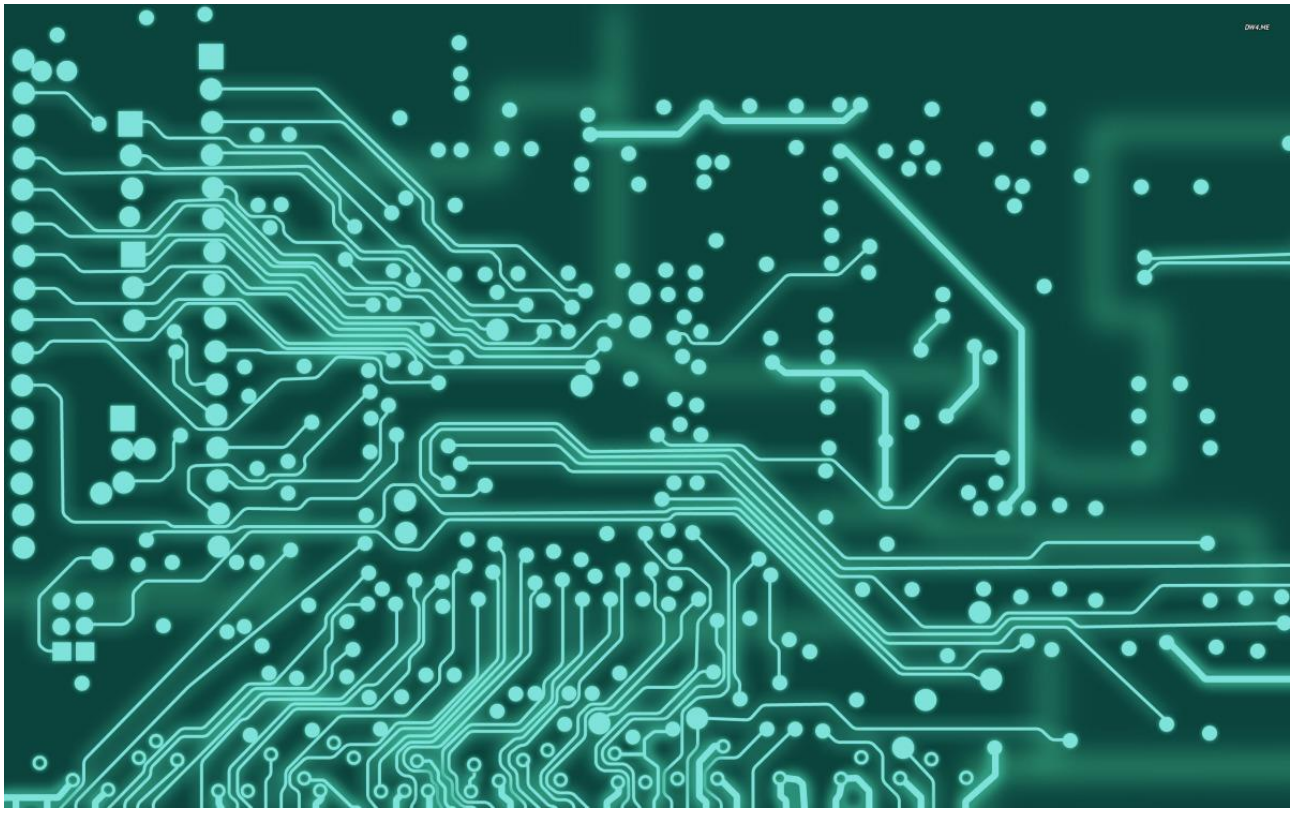

Рисунок 3.4 – Тестове зображення

Далі був написаний код обробника зображення, а саме для зчитування зображення, трансформації зображення у чорно-біле та інверсії кольорів.

Після цього за допомогою функції cv2.threshold() була проведена бінаризація зображення з використанням параметру cv2.THRESH_BINARY. Експериментальним методом було підібрано оптимальне порогове значення.

Результат роботи функції cv2.threshold() з різними пороговими значеннями можна побачити на рисунку 3.5.

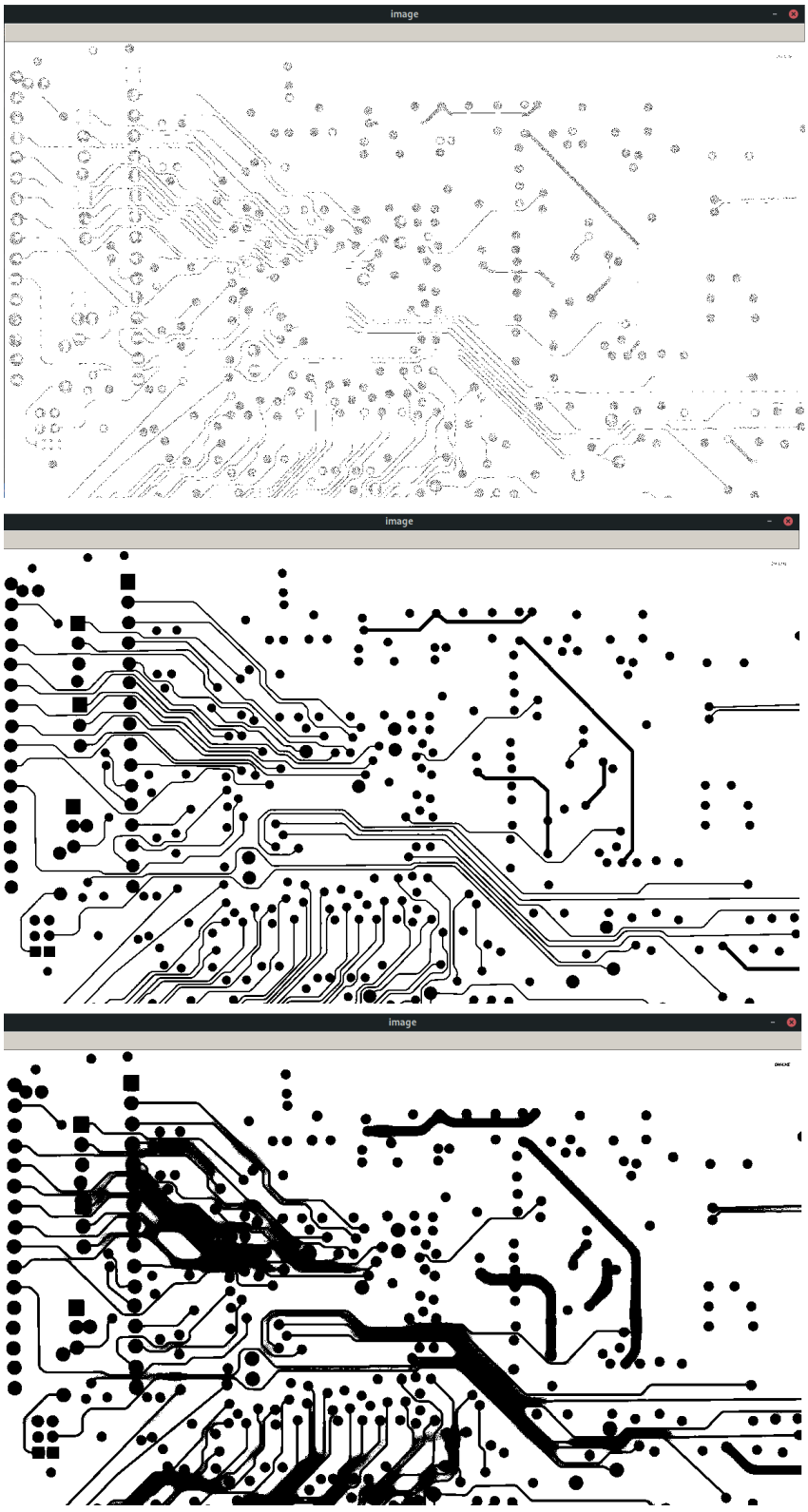

Рисунок 3.5 – Результати бінаризації з різними пороговими значеннями

Програмний код, який зчитує два зображення, перетворює їх на чорно-білі та бінаризує з експериментально підібраним порогом, представлений на рисунку 3.5.

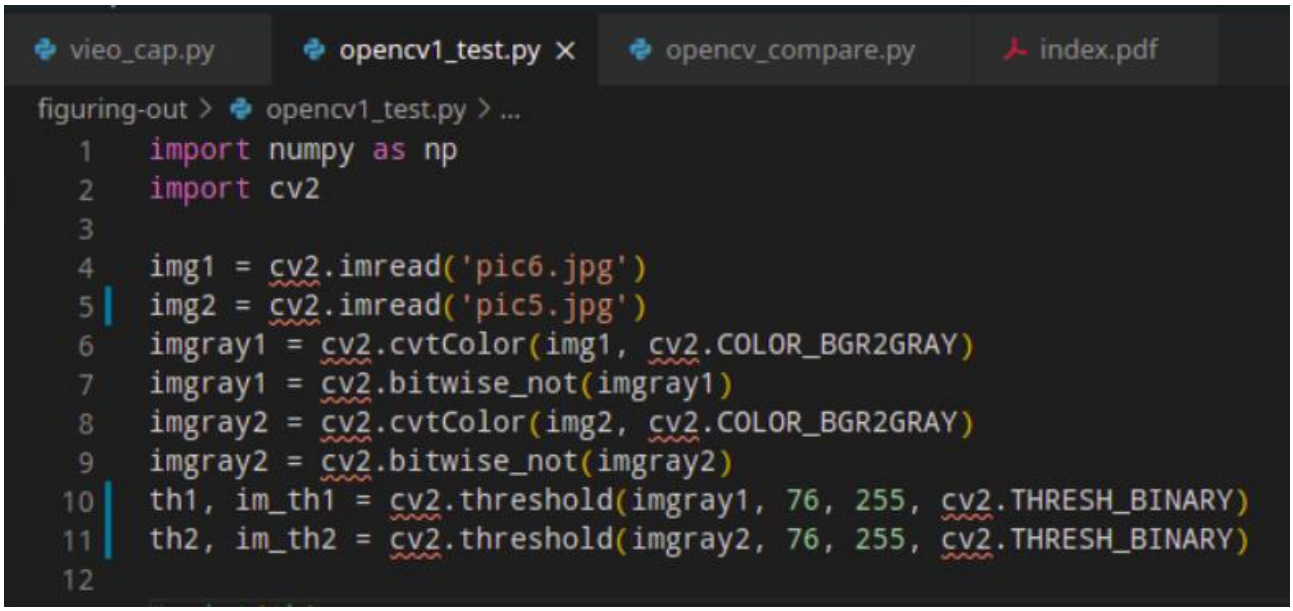

Рисунок 3.5 – Програмний код для бінарізації зображень

Далі треба знайти границі елементів на зображенні. Для знаходження границь об’єктів у бібліотеці OpenCV є функція cv2.findContours().

Функція cv2.findContours() [37] приймає на вході три аргументи, а саме:

– вхідне зображення;

– режим отримання контуру;

– метод апроксимації контурів.

На виході ця функція видає контури та ієрархію. Змінна з контурами передає список масивів у форматі мови Python, який утримує у собі всі контури, котрі були знайдені на зображенні. Кожний контур – це NumPy масив, який складається з координат контурів по осях *x* та *y*. Слід відзначити такий аргумент, як метод апроксимації контурів. Якщо передати такий параметр, як cv.CHAIN_APPROX_NONE, то всі координати контурів зберігаються у пам’яті.

У разі використання параметру CHAIN_APPROX_SIMPLE, координати контурів спрощуються тільки до опорних точок. Таким чином, якщо взяти

зображення прямокутника та знайти його контур за допомогою обох параметрів, то у разі використання параметру cv.CHAIN_APPROX_NONE, функція cv2.findContours() поверне повний контур прямокутника, а з параметром CHAIN_APPROX_SIMPLE функція cv2.findContours() поверне тільки опорні точки контуру. Приклад використання обох параметрів на бінаризованому зображенні прямокутника наведено на рисунку 3.6 [12].

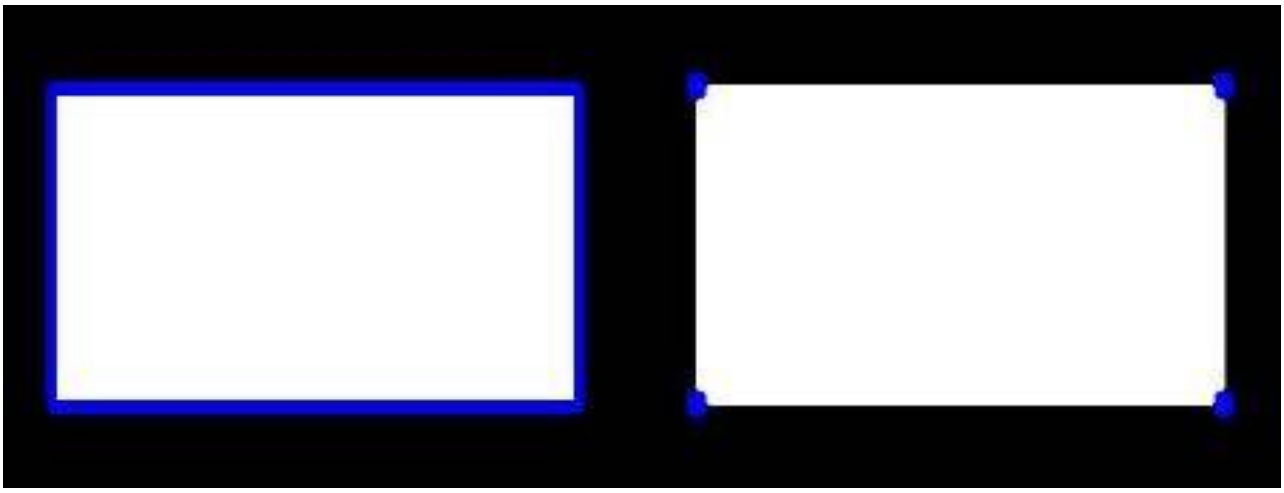

Рисунок 3.6 – Приклад різних типів апроксимації

Як можна побачити на даному зображенні, другий метод створює значно меншу кількість координат та, виходячи з цього, масив координат контуру займає значно меншу кількість пам’яті.

Але цей метод зовсім не підходить для знаходження контурів складних зображень, таких як рентгенограма друкованої плати.

Для того, щоб виводити контури регіонів, які були знайдені на вхідному зображенні, використовується функція cv2.drawContours. У неї передається п’ять аргументів:

- вхідне зображення;
- масив координат контуру;
- індекс контурів (які контури відображувати);
- колір контуру;
- товщина контуру у пікселях.

Нижче наведений приклад реалізації виділення контурів (рисунок 3.7).

```
15
16
17  contours1, hierarchy1 = cv2.findContours(im_th1, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
18  contours2, hierarchy2 = cv2.findContours(im_th2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
19
20
21  dc1 = cv2.drawContours(imgray1, contours1, -1, (0,0,255), 1)
22  dc2 = cv2.drawContours(imgray2, contours2, -1, (0,0,255), 1)
23
```

Рисунок 3.7 – Реалізація знаходження та накладення контурів на зображення

Результат виконання функцій знаходження та накладення контурів можна побачити на рисунку 3.8.

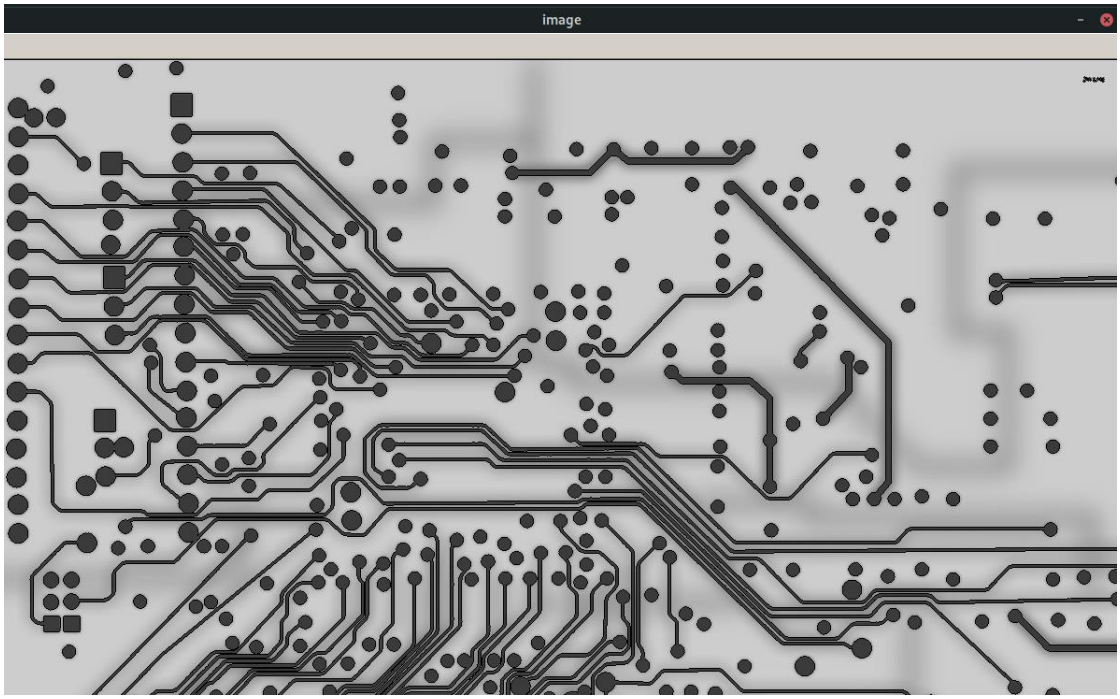

Рисунок 3.8 – Результат виконання представленого коду

Далі для коректної роботи класифікатора слід розробити функцію знаходження площі елементів. Це необхідно для того, щоб оцінити різницю між елементами двох зображень більш точно. Також надалі завдяки цій функції можна буде обчислювати реальні розміри елементів зображення, таких як

струмопровідні доріжки та різноманітні дефекти. Для цього необхідно знати роздільну здатність матриці приймача рентгенівського випромінювання. Знаючи її розмір і роздільну здатність матриці, можна обчислити її щільність пікселів. Тоді, знаючи цей параметр, можна перевести площу виділеного регіону в пікселях в площу у мм^2.

Для знаходження різниці між двома зображеннями слід скористатись математичними операціями над зображеннями, які надаються бібліотекою NumPy. Для того, щоб знаходити додаткові елементи та елементи, які зникли відносно зразкового зображення, слід послідовно провести операції віднімання спочатку фактичного зображення від зразкового, а потім зразкового від фактичного. Далі, ці результати слід скласти. Таким чином, утворюється зображення, складене з елементів і регіонів, які відрізняються між зображеннями. Важливо відзначити, що для виконання математичних операцій між двома зображеннями, представленими як матриці, треба, щоб ці зображення мали суворо однакову роздільну здатність.

Програмний код, який виділяє на зображенні контури, проводить операцію пошуку різниці між зображеннями та у разі наявності різниці виводить всі відмінності в окремий файл, зображений на рисунку 3.9.

```
17  contours1, hierarchy1 = cv2.findContours(im_th1, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
18  contours2, hierarchy2 = cv2.findContours(im_th2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
19
20
21  dc1 = cv2.drawContours(imgray1, contours1, -1, (0,0,255), 1)
22  dc2 = cv2.drawContours(imgray2, contours2, -1, (0,0,255), 1)
23
24  diff1 = cv2.subtract(dc1, dc2)
25  diff2 = cv2.subtract(dc2, dc1)
26  result = cv2.add(diff1,diff2)
27  endresult = not np.any(result)
28  if endresult is True:
29      print("same")
30  else:
31      cv2.imwrite('result.jpg', result)
32      print ("not same")
```

Рисунок 3.9 – Приклад програмного коду для порівняння двох зображень

Вхідні зображення та результат виконання представленого вище програмного коду наведено на рисунку 3.10.

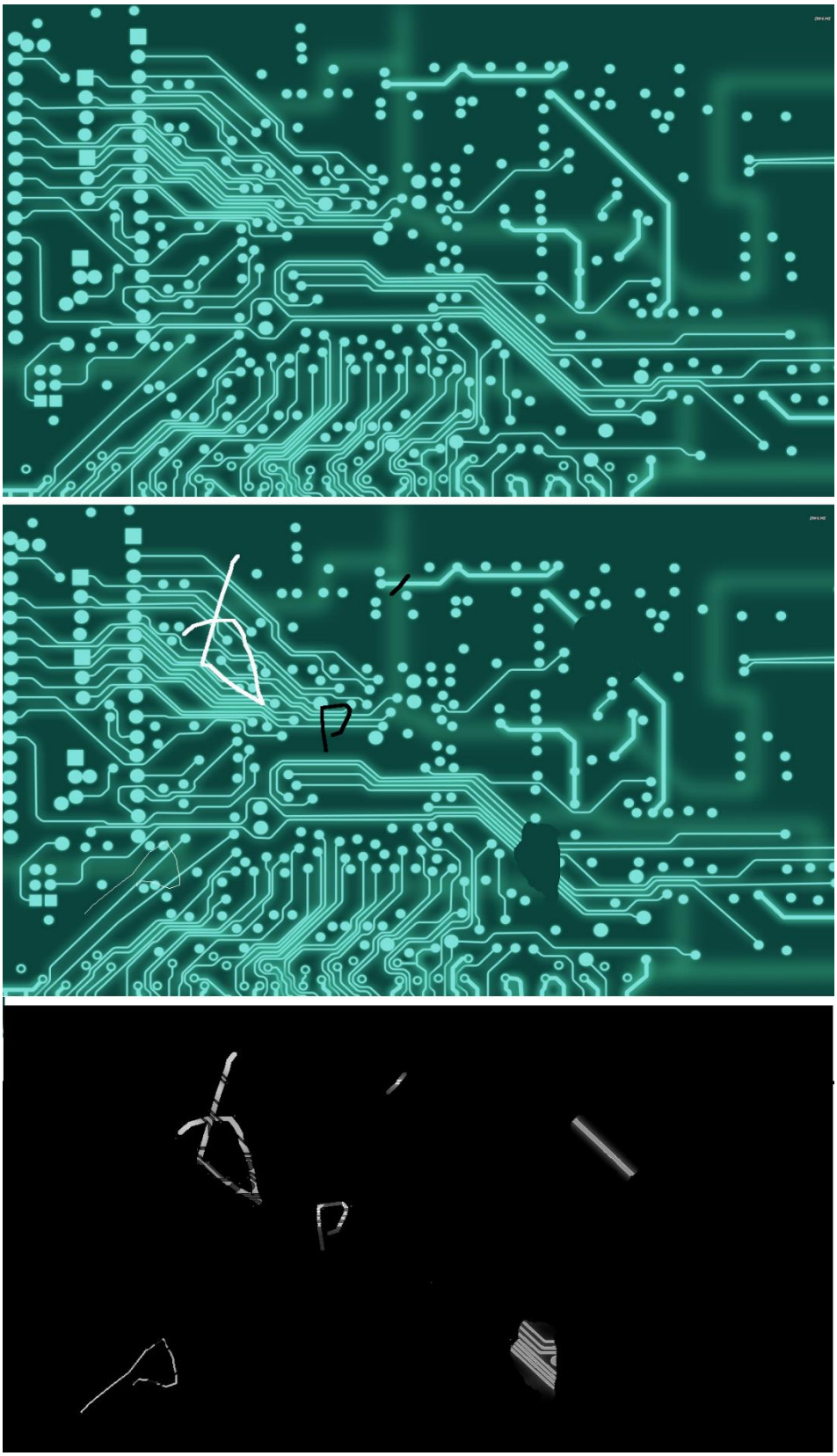

Рисунок 3.10 – Результат порівняння двох зображень

Також була реалізована функція підрахування площі дефектів, яка спирається на розмір матриці рентген-апарату та її роздільну здатність. Маючи на увазі, що розміри матриці та зображення є незмінними, можна підрахувати кількість пікселів у квадратному сантиметрі зображення та розділити на неї кількість пікселів дефекту. Це дозволить достатньо точно знати, яку площу займає той чи інший дефект. Програмний код цього модулю представлений на рисунку 3.11.

```
44
45  for contour in contours_result:
46      if cv2.contourArea(contour) > 0:
47          contour_sum_area = cv2.contourArea(contour)
48
49  print(contour_sum_area)
50
51  img = cv2.imread(tested_image)
52
53  h, w, c = img.shape
54  print('width:  ', w)
55  print('height: ', h)
56
57  pixel_in_cm_width = w / sensor_width
58  pixel_in_cm_height = h / sensor_height
59
60  if pixel_in_cm_width == pixel_in_cm_height:
61      pixel_in_sq_cm = pixel_in_cm_height * pixel_in_cm_width
62
63  print (contour_sum_area / pixel_in_sq_cm)
```

Рисунок 3.11 – Програмний код для підрахунку площі дефекту

У результаті була розроблена програма, яка приймає на вході два зображення – шаблонне та те, яке тестується, – і параметри рентгенівської матриці. На виході вона повертає зображення з математичною сумою всіх дефектів і підраховує їх площу (рисунок 3.12).

```
~/Projects/figuring-out >>> python3 opencv1_test.py
not same
159.0
width:   1200
height:  750
100.0
100.0
0.0159
```

Рисунок 3.12 – Результат роботи програми

3.5 Розробка нейронної мережі для розпізнавання дефектів

Для розпізнавання дефектів необхідні дані для навчання нейронної мережі. Такими даними для згорткової нейронної мережі є зображення елементів, які потрібно розпізнавати. З набору таких зображень і текстових анотацій формується так званий датасет.

Після пошуків не було знайдено готового датасету чи набору зображень дефектів, тому вручну був набраний пул зображень, на яких є ті чи інші набори струмопровідних доріжок рідних форм і розмірів. За допомогою програмного комплексу для обробки зображень GNU Image Manipulation Program були нанесені імітації різних дефектів, таких як включення зайвої міді, розриви, тріщини у струмопровідних доріжках і короткі замикання. Приклад таких зображень наведено на рисунку 3.13.

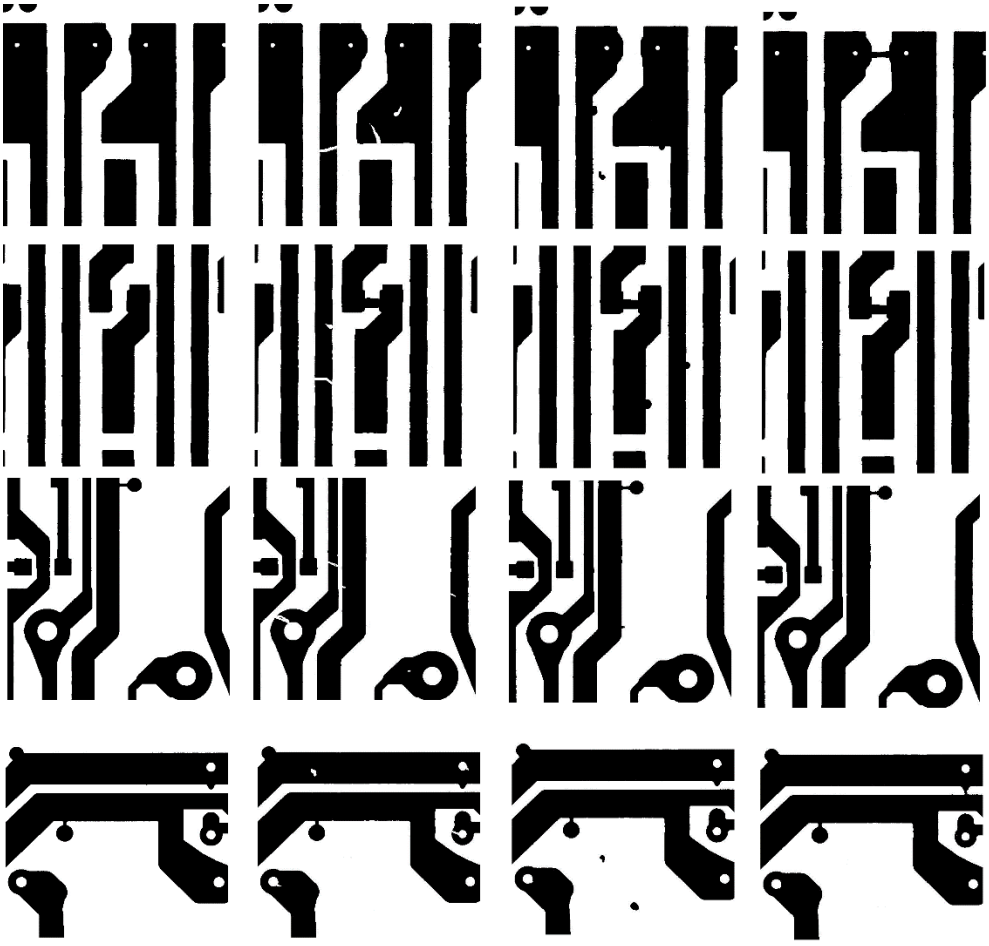

Рисунок 3.13 – Приклад зображень для розроблюваного датасету

Як можна побачити з рисунку 3.13, спочатку береться вихідне бінаризоване зображення без дефектів і далі наносяться дефекти рідних типів:

– пустоти струмопровідного шару;

– зайві включення;

– короткі замикання.

Таким чином, був створений набір з 20000 зображень.

Далі слід перетворити набір зображень у повноцінний датасет. Для цього була написана програма мовою Python.

Спочатку треба зчитати зображення з жорсткого диску та привести їх до єдиної форми, розміру та кольорового простору. Це робиться через те, що не всі зображення на вході мають ідеально однаковий розмір. Програмний код, який виконує ці дії, представлений на рисунку 3.14.

figuring-out > dataset.py > ...

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import cv2
4  import os
5  import random
6  import pickle
7
8  DATADIR = "/home/vlad/ProjectsPCBData/group1"
9  CATEGORIES = ["fine", "void", "short","foreign"]
10 for category in CATEGORIES:
11     path = os.path.join(DATADIR, category)
12     for img in os.listdir(path):
13         img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
14         plt.imshow(img_array, cmap="gray")
15         #plt.show()
16         break
17 IMG_SIZE = 100
18 new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
19 plt.imshow(new_array, cmap = "gray")
20 #plt.show()
21
22 training_data = []
23
```

Рисунок 3.14 – Програмний код трансформації зображень

З першого по шостий рядок у проєкт імпортуються всі необхідні бібліотеки для роботи з масивами, графіками, файловою системою та для трансферу даних.

Далі вказується робоча папка із зображеннями для датасету, визначаються категорії та створюються масиви зображень. Для перевірки вихідного розміру зображень у циклі додається функція plt.show(), яка відображає приведені до єдиного кольорового простору зображення. Проміжний результат роботи циклу представлений на рисунку 3.15.

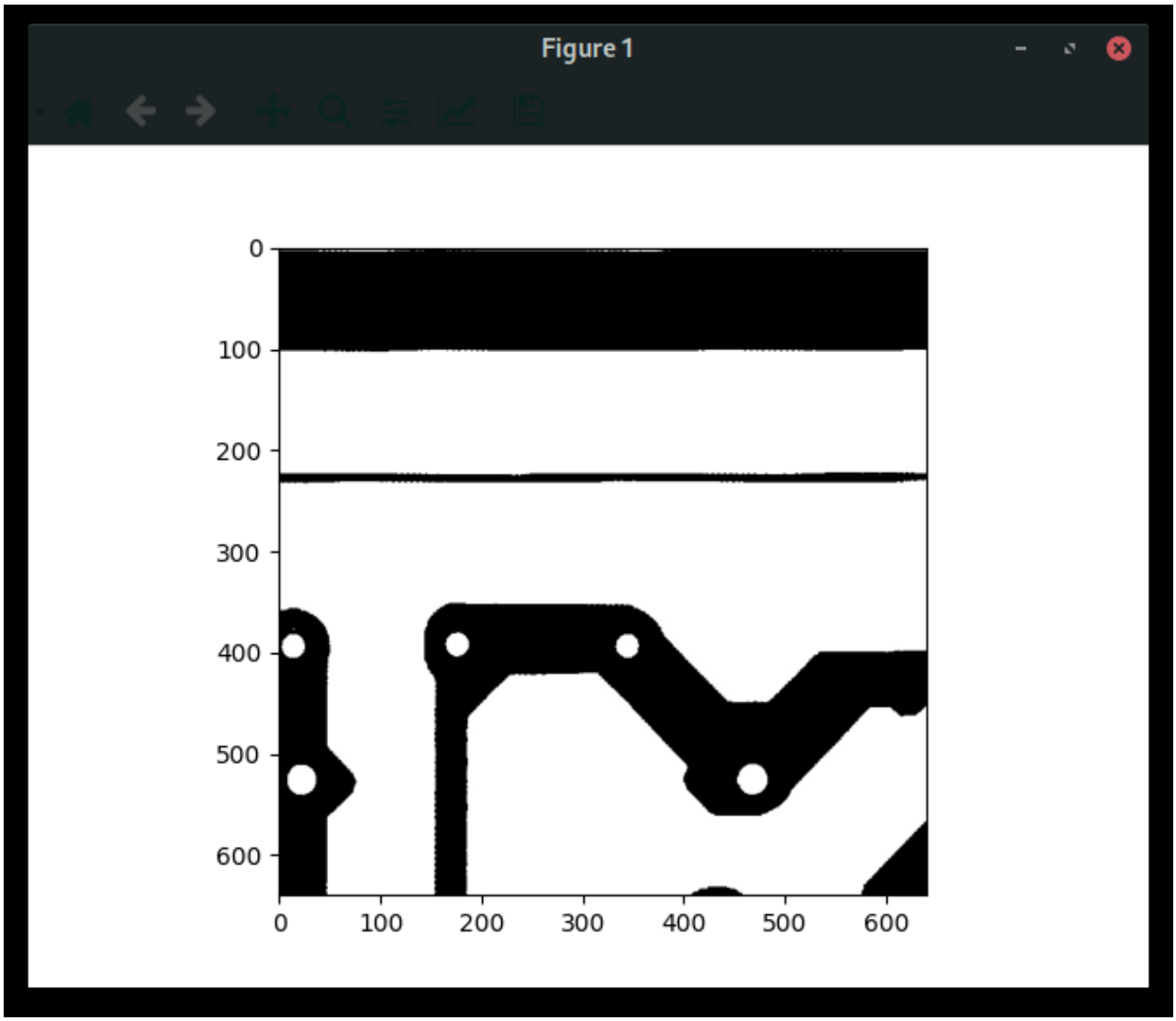

Рисунок 3.15 – Результат роботи циклу

Далі, за допомогою функції cv2.resize(), зображення приводяться до єдиного розміру, який вибирається згідно з потребами. Зображення повинні займати як можна менше місця у пам’яті через те, що через структуру проєкту датасет передається у нейронну мережу за допомогою використання буферу в

оперативній пам’яті хост-машини, яка сильно обмежена. Також чим зображення менше, тим швидше воно оброблюється нейронною мережею, але слід не забувати, що зображення має бути достатньо чітким для того, щоб на ньому можна будо виділити дефекти та розпізнати, в іншому разі процес навчання нейронної мережі буде неефективним. Після ряду тестів було вирішено, що оптимальний розмір зображень розроблюваного датасету становить від 100 × 100 пікселів до 250 × 250 пікселів. Результат функції трансформації розміру зображення наведено на рисунку 3.16.

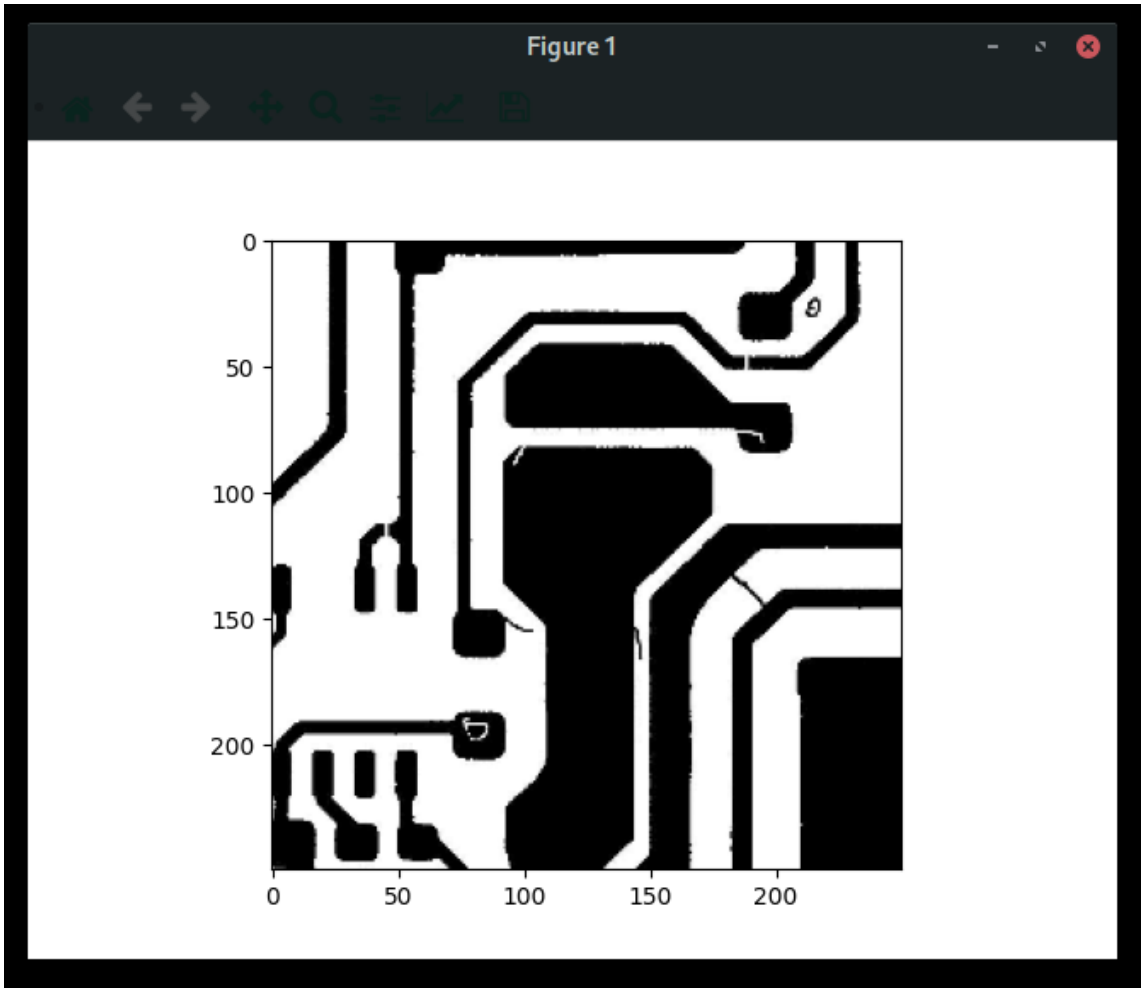

Рисунок 3.16 – Результат роботи функції трансформації розмірів зображень

Після того, як зображення зчитані, приведені к єдиному розміру та кольоровому простору та трансформовані у масиви даних, треба перетворити їх

на дані для навчання нейронної мережі. Перш за все, треба перемішати між собою дані для навчання. Через те, що зображення з однаковими типами дефектів знаходяться в окремих папках і папки зчитуються програмою послідовно, виникне проблема із різноманітністю даних. Якщо у процесі навчання передавати на вхід нейронної мережі послідовність навчальних даних із незмінним типом об’єкту, який потрібно розпізнати, навчання нейронної мережі стає неефективним. У результаті нейронна мережа буде навчатись класифікувати всі вхідні зображення як один тип дефекту. Для того, щоб цього уникнути, треба скористатись функцією бібліотеки numpy під назвою np.shuffle(). Вона перемішує індекси елементів масиву та створює однорідний масив різноманітних даних.

Далі створюються два нових масиви даних, один з тренувальними даними, другий з інформацією про об’єкт, що зображений на даному рисунку. Після цього масиви тренувальних зображень треба перетворити з двомірних на одномірні для прискорення обробки першим шаром нейронної мережі в умовах жорсткого обмеження доступних обчислювальних ресурсів.

Програмний код, який забезпечує виконання описаних операцій з масивами, представлено на рисунку 3.17.

```
24  def create_training_data():
25      for category in CATEGORIES:
26          class_num = CATEGORIES.index(category)
27          path = os.path.join(DATADIR, category)
28          for img in os.listdir(path):
29              img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
30              new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
31              training_data.append([new_array, class_num])
32  create_training_data()
33  print(len(training_data))
34  random.shuffle(training_data)
35  X = []
36  Y = []
37  for features, label in training_data:
38      X.append(features)
39      Y.append(label)
40  X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
```

Рисунок 3.17 – Підготовка даних для навчання

Після обробки тренувальних даних їх необхідно експортувати. Для цього був використаний модуль Pickle. Цей модуль використовує потужний алгоритм для серіалізації та десеріалізації даних. Цей модуль трансформує об'єкт у потік байтів, які можна записати в файл і зчитати. Цей модуль був обраний через простоту використання та швидкість роботи. Програмний код для експорту даних датасету та тестування роботи цієї функції з результатами, виведеними в емулятор терміналу, наведені на рисунку 3.18.

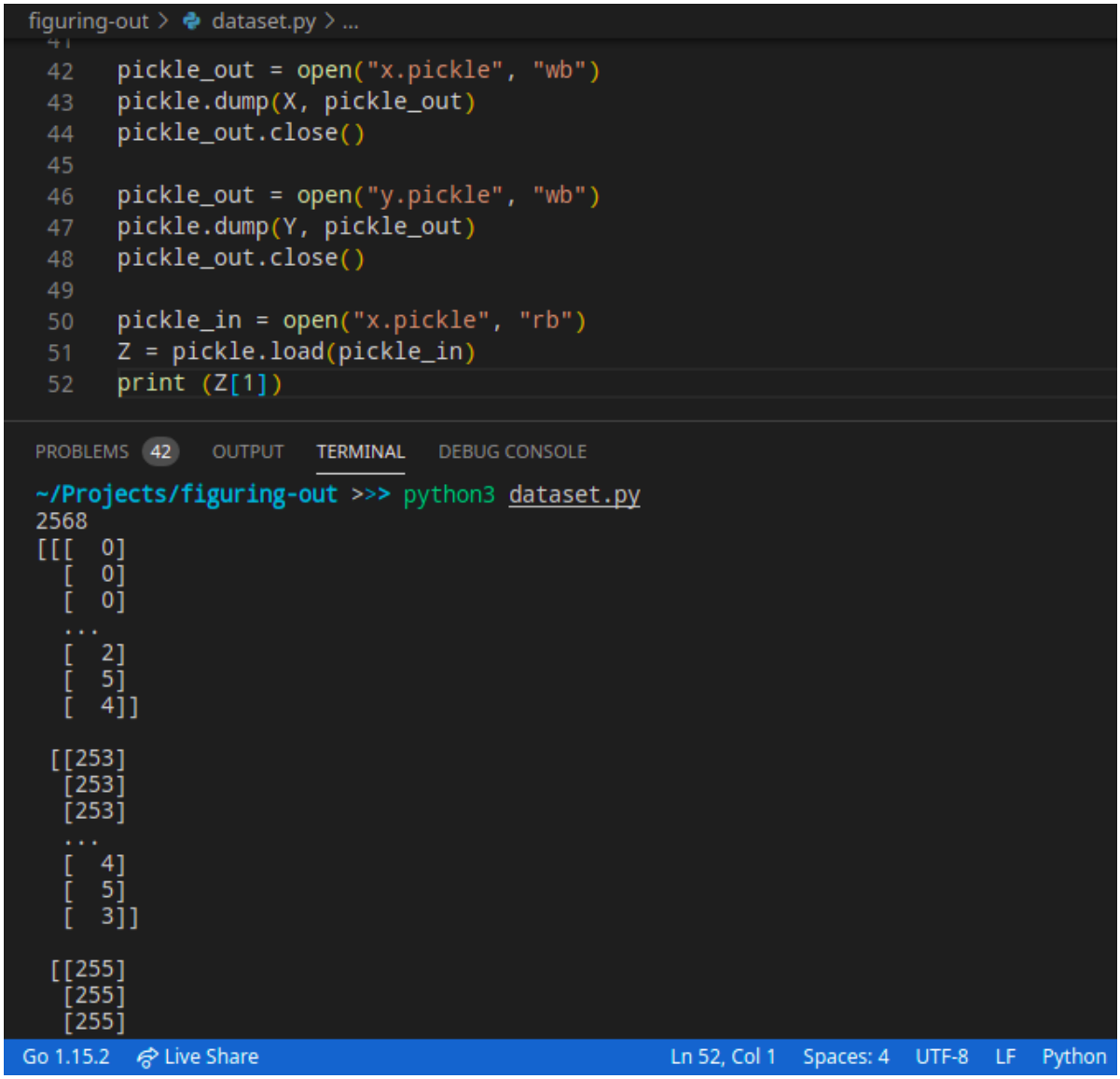

Рисунок 3.18 – Результат роботи генератора датасетів

Наступним кроком після створення та перевірки датасету буде створення та навчання нейронної мережі. Як зазначалось раніше, для розробки моделі нейронної мережі буде використана бібліотека Tensorflow [38].

Перш за все необхідно додати до проєкту всі необхідні бібліотеки та модулі (рисунок 3.19).

figuring-out > convnet_tf.py > ...

```
1  import numpy as np
2  import tensorflow as tf
3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
5  import pickle
6
```

Рисунок 3.19 – Ініціалізація проєкту

Раніше під час створення датасету він був експортований за допомогою бібліотеки pickle. Для того, щоб завантажити експортований датасет у проєкт, була використана конструкція, зображена на рисунку 3.20.

```
6
7   pickle_in = open("x.pickle" ,"rb")
8   X = pickle.load(pickle_in)
9   pickle_in = open("y.pickle" ,"rb")
10  Y = pickle.load(pickle_in)
11
12  X = np.array(X/255.0)
13  Y = np.array(Y)
14
```

Рисунок 3.20 – Імпортування датасету у проєкт

Також на цьому сніппеті виконана оптимізація вхідних даних. Знаючи, що кожен піксель на зображеннях має значення кольору від 0 до 255, можна розділити матриці зображень на 255 та отримати значення кольору від 0 до 1, що допоможе дещо прискорити процес навчання нейронної мережі.

Розроблювана модель нейронної мережі буде мати чотири скритих шари – 2 згорткових і 2 об’єднуючих. Як активаційна функція у скритих шарах нейронної мережі буде використовуватись функція Rectified Linear Unit – як

функція, яка найчастіше використовується у таких випадках. Для вихідного шару буде використовуватись сигмоїда (рисунок 2.5).

Програмний код, який створює основну структуру моделі нейронної мережі, наведено на рисунку 3.21.

```
14
15  model = Sequential()
16  model.add(Conv2D(64, (3,3), input_shape = X.shape[1:]))
17  model.add(Activation("relu"))
18  model.add(MaxPooling2D(pool_size=(2,2)))
19
20  model.add(Conv2D(64, (3,3)))
21  model.add(Activation("relu"))
22  model.add(MaxPooling2D(pool_size=(2,2)))
23
24  model.add(Flatten())
25  model.add(Dense(64))
26  model.add(Activation('sigmoid'))
27
```

Рисунок 3.21 – Структура моделі нейронної мережі

У цьому коді для додавання основних елементів нейронної мережі використовується метод model.add, за допомогою якого додаються раніше імпортовані модулі та визначаються їх параметри, такі як кількість нейронів, розмір згортки, типи активаційних функцій.

Після визначення шарів, які будуть включені у модель, та їх параметрів модель необхідно скомпілювати, навчити й експортувати. Програмний код, який виконує ці дії, показаний на рисунку 3.22.

```
27
28  model.compile(loss="crossentropy", optimizer="adam", metrics='accuracy')
29
30  model.fit(X, Y, batch_size=2, epochs = 5, validation_split= 0.1)
31
32  model.save('flexpcb_detect.model')
```

Рисунок 3.22 – Компіляція та навчання моделі

Для компіляції моделі нейронної мережі викликається метод model.compile(), в який передається тип підрахування втрат, тип оптимізатора та тип метрик навчання. Значення цих параметрів були обрані згідно з рекомендаціями в офіційній документації бібліотеки TensorFlow.

Після процесу компіляції виконується процес навчання нейронної мережі за допомогою виклику методу model.fit(), в який передається набір даних для навчання, анотації до них, розмір одного пакета зображень для циклу (batch_size), кількість епох навчання та точність валідації. Останній параметр був взятий з офіційної документації.

Якщо процес навчання проходить успішно, то за допомогою методу model.save() готова модель зберігається. Метод приймає на вході рядок, який буде назвою моделі.

Під час навчання моделі виникли проблеми з пам’яттю, якої було недостатньо для завантаження датасету, тому він був зменшений до 1000 зображень.

Виведення в емулятор терміналу з процесом навчання представлений на рисунку 3.23.

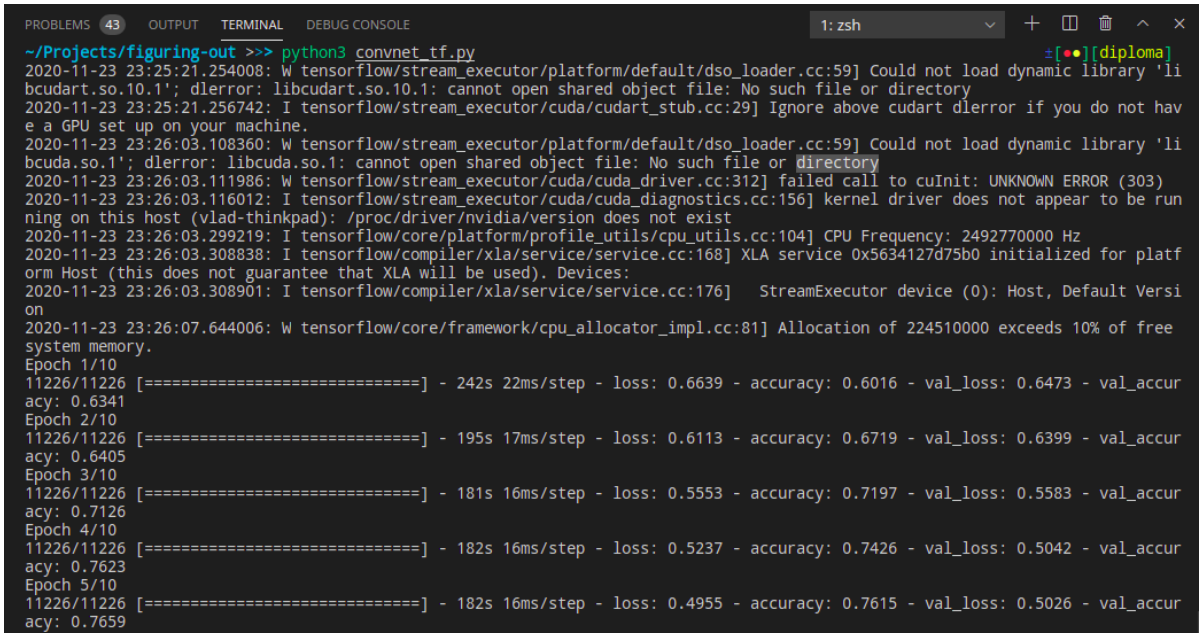

Рисунок 3.23 – Процес навчання моделі

Таким чином, за 5 епох навчання вдалося досягти точності розпізнавання у 76,59 %, що є успішним результатом для датасету зі складними зображеннями та анотаціями класів дефектів без їх місцезнаходження на зображеннях. Можливе збільшення точності розпізнавання за допомогою збільшення кількості епох навчання, але у даному випадку значне збільшення епох неможливе через обмеження пам’яті й обчислювальних можливостей машини, на якій проводяться розробка нейронної мережі та її дослідження. Характеристики хост-машини представлені на рисунку 3.24.

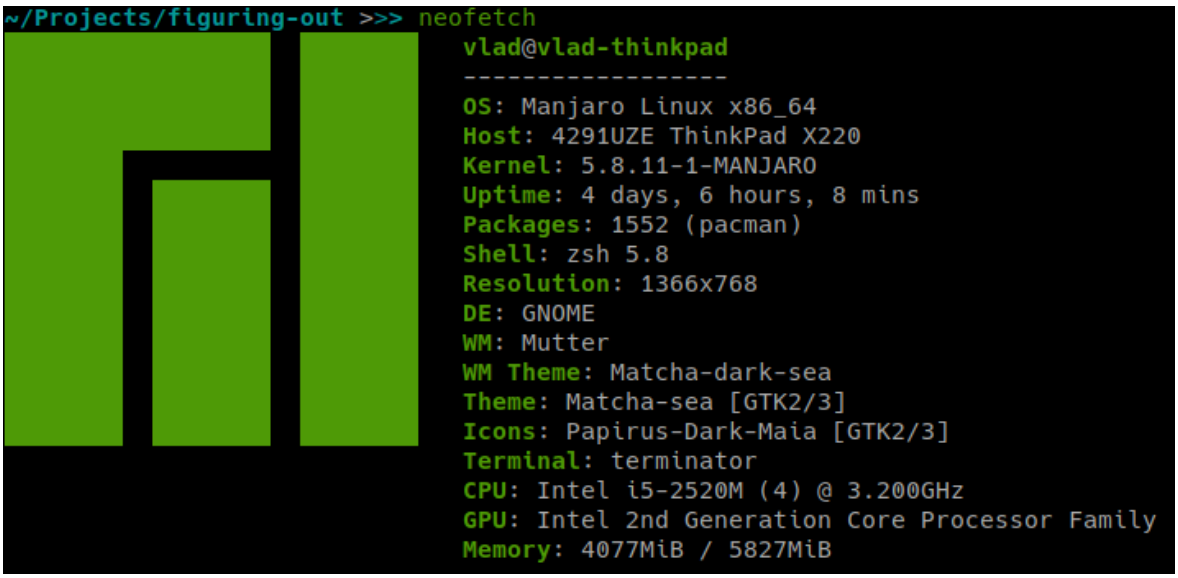

Рисунок 3.24 – Характеристики хост-машини

Для перевірки роботи моделі необхідно протестувати її на зовнішньому зображенні, яке не було використано для тренування й оцінки точності під час тренування. Для цього був написаний програмний код, який приймає на вході зображення згідно з означеним шляхом і виконує над ним наступні перетворення:

– зчитує зображення в одноканальному режимі;
– створює cv2 масив даних;
– змінює розмір масиву до прийнятого у процесі тренування моделі.

Далі завантажується раніше експортована модель, яка пройшла процес навчання, та виконується передбачення за допомогою функції model.predict(), в яку передається раніше підготовлене зображення. Далі було виведене передбачення, спочатку у необробленому вигляді, а далі у вигляді категорії. Приклад програмного коду зображений на рисунку 3.25.

```
import cv2
import tensorflow as tf

CATEGORIES = ["fine", "void", "short", "foreign"]

def prepare(filepath):
    IMG_SIZE = 100
    img_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    return new_array.reshape(-1, IMG_SIZE, IMG_SIZE, 1)

    model = tf.keras.models.load_model("flexpcb_detect.model")

prediction = model.predict([prepare('1.jpg')])
print(prediction)
print(CATEGORIES[int(prediction[0][0])])
```

Рисунок 3.25 – Програмний код модуля передбачення

Для перевірки роботи моделі було взяте зображення, наведене на рисунку 3.26.

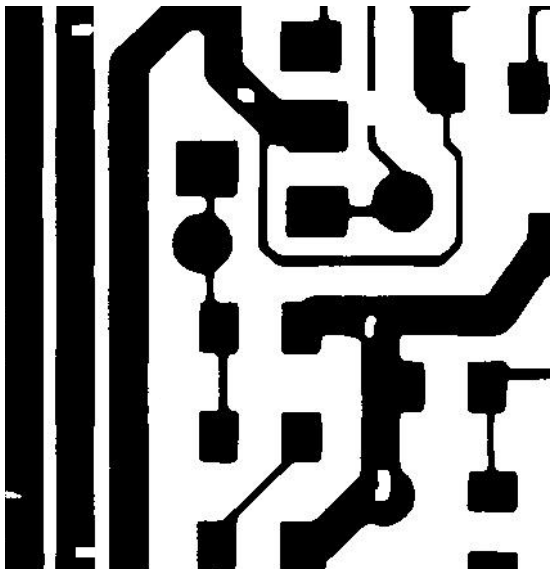

Рисунок 3.26 – Тестове зображення

На тестовому зображені нанесені тільки дефекти типу «пустоти». Далі запускається модуль передбачення (рисунок 3.27).

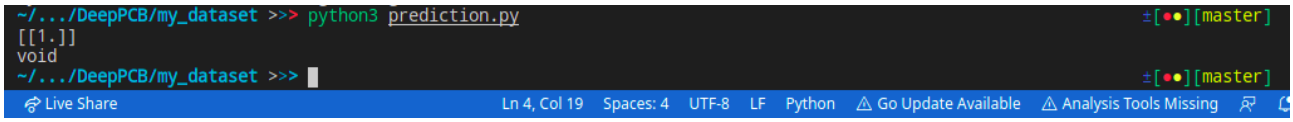

Рисунок 3.27 – Результат роботи модуля передбачення

У результаті роботи програми можна побачити, що вона повертає масив з одним значенням, яке відповідає другому елементу масиву класів. За допомогою другої функції виведення виводиться відповідний номеру елемента клас. Аналогічний тест був проведений для дефектів типу «коротке замикання». Вхідне зображення та результат роботи програми наведено на рисунку 3.28.

Рисунок 3.28 – Перевірка передбачень для дефекту класу «коротке замикання»

Розроблювана модель може класифікувати зображення за типами дефектів, які на них знаходяться, але якщо на зображенні у наявності більш ніж один тип дефекту, ця модель стає малоефективною. Для правильної роботи системи необхідно реалізувати схему розпізнавання різних об’єктів на вхідному зображенні.

Для реалізації паралельного розпізнавання елементів на зображенні був використаний фреймворк TensorFlow Object Detection API.

TensorFlow Object Detection API – це фреймворк із відкритим вихідним кодом, побудований на базі TensorFlow, створений для розробки, складання та розгортки моделей для розпізнавання об’єктів [39].

Для використання TensorFlow Object Detection API треба встановити всі необхідні залежності, а саме:

- pillow;
- lxml;
- matplotlib;
- tf_slim;
- scipy.

Залежності встановлюються у вигляді пакетів за допомогою пакетного менеджеру Python pip. Команда виглядає як pip install <назва пакету>.

Через те, що для розробки моделі був використаний фреймворк keras, що входить до ядра Tensorflow, на якому побудований фреймворк TensorFlow Object Detection API, з’являється можливість імпортувати у нього готову навчену модель ЗНМ, котра була розроблена раніше.

Через те, що далі буде реалізоване розпізнавання дефектів на зображенні, клас «fine», який використовувався у процесі навчання моделі, втрачає свою актуальність. Таким чином, датасет був доповнений зображеннями з типом дефекта «розширення» та додаванням у модель ЗНМ відповідного класу «excess». Також був змінений механізм передавання даних про зображення, якщо раніше зображення у датасеті анотувались згідно з папкою, в якій вони

знаходились, програма сама створювала анотації у .csv форматі, то тепер анотації створюються у ручному режимі за допомогою програми LabelImg.

Приклад створення анотованого зображення для датасету представлений на рисунку 3.29.

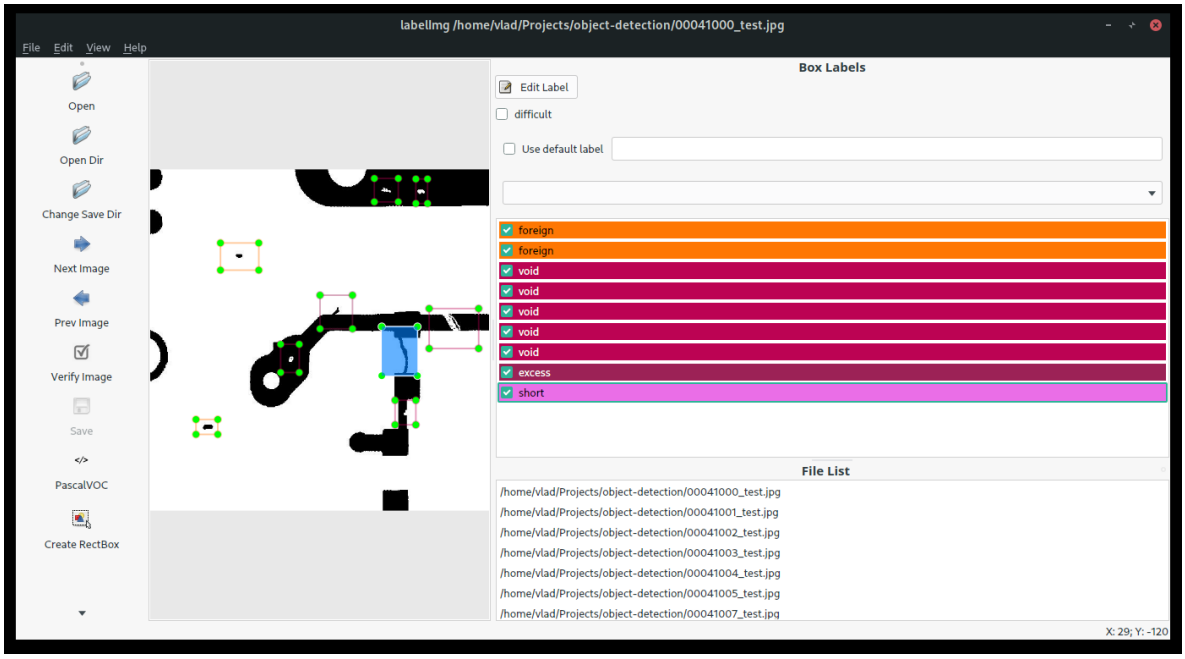

Рисунок 3.29 – Процес анотування зображень

У результаті створюється структура даних, яка складається з двох папок. В одній знаходяться зображення, а у другій знаходяться анотації до зображень у форматі .xml (рисунок 3.30).

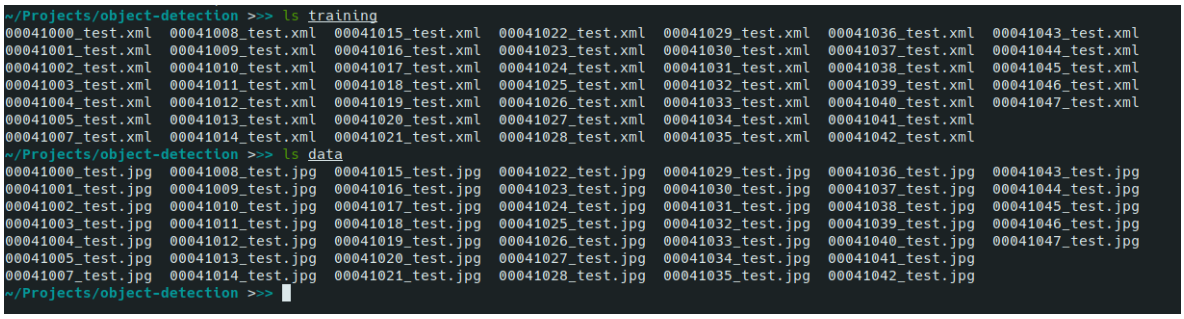

Рисунок 3.30 – Структура файлів датасету

Перенавчання ЗНМ виконувалось ідентично до начального та було описане раніше.

Перш за все необхідно розгорнути TensorFlow Object detection API. За допомогою офіційної документації [39] було створено наступний скрипт (рисунок 3.31).

```
import os
import subprocess

print("Checking if the environment is already setup ...")
c = subprocess.call(["python" ,"object_detection/builders/model_builder_test.py"],
                    stdout=open(os.devnull, 'wb')
                    )
if c == 0:
    print("environment is already setup")
    print("The Tensorflow Object detection API is ready to use!")

else:
    print('Environment is not setup already. Setting up a fresh environment...')
    curr_work_dir = os.getcwd()
    # compile the protobuf files
    cmd = "protoc object_detection/protos/*.proto --python_out=."
    os.system(cmd)
    print("Protobuf files are compiled")
    # adding environment variable to ~/.bashrc file
    home = os.path.expanduser("~")
    with open(os.path.join(home,".bashrc"),"a") as f:
        f.write("\n")
        f.write("# For Tensorflow object detection API")
        f.write("\n")
        f.write("export PYTHONPATH=$PYTHONPATH:{}:{}/slim".format(curr_work_dir,curr_work_dir))
    print("Python path added to ./bashrc file")

    print("The environment is setup. From a new terminal try running the same script to verify")
```

PROBLEMS 50 OUTPUT TERMINAL DEBUG CONSOLE 1: zsh

Рисунок 3.31 – Скрипт для налаштування робочого оточення TensorFlow Object Detection API

Цей скрипт перевіряє, чи було оточення раніше налаштоване, за допомогою виклику субпроцесу builders/model.builder/test, який стає доступний завдяки змінній оточення $PYTHONPATH, у котру додається TensorFlow Data API у разі його встановлення. Якщо субпроцес викликається та виконується, до терміналу видаються повідомлення, що оточення вже встановлене та готове до роботи, в іншому разі виконується встановлення API згідно із кроками, описаними в документації. У результаті роботи скрипту робоче оточення встановлюється та налаштовується. Результат роботи скрипту наведено на рисунку 3.32.

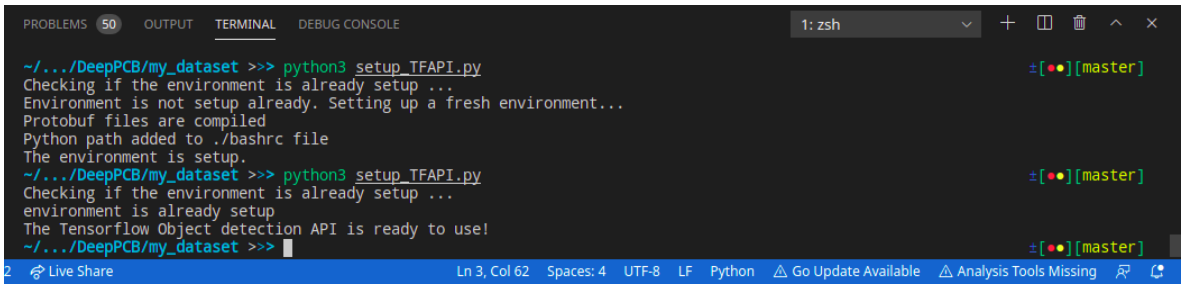

Рисунок 3.32 – Результат роботи скрипта

Для трансформації навченої моделі ЗНМ keras до моделі, сумісної з TensorFlow Data API, був адаптований наступний скрипт з неофіційної документації (рисунок 3.33) [40].

```
import tensorflow as tf
from tf.keras.models import model_from_json


def keras_ckpt_to_tf_ckpt(keras_model_json_path,
                          keras_weights_path,
                          tf_ckpt_save_path="./tf_ckpoint.ckpt"):

    with tf.Session() as sess:
        json_file = open('/home/vlad/Projects/flexpcb_model/')
        loaded_model_json = json_file.read()
        json_file.close()
        model = model_from_json(loaded_model_json)
        #load weights
        model.load_weights('/home/vlad/Projects/flexpcb_model/model.ckpt')
        print('loaded keras model from disk')
        saver = tf.train.Saver()
        saver.save(sess,'/home/vlad/Projects/flexpcb_model/tf_model.ckpt')
        print("Tensorflow checkpoint is saved in {}".format(tf_ckpt_save_path))
```

Рисунок 3.33 – Скрипт для перетворення моделі ЗНМ

TensorFlow Object detection API включає до свого складу так званий Region Proposal Network (RPN), який являє собою систему накладання регіонів на зображення. Тому відпадає необхідність у розробці програмного коду для виділення регіонів зображення з передбаченнями. Для тестуваня роботи системи залишається тільки додати до конфігурації системи розпізнавання об’єктів шлях до зображень. Як зображення для тестування були використані зображення з датасету, які не були використані для навчання або для оцінки ефективності

навчання. Для запуску розпізнавання у директорії з моделлю треба виконати наступну команду (рисунок 3.34).

```
~/.../DeepPCB/my_dataset >>> python3 export_inference_graph.py \
    --input_type image_tensor \
    --pipeline_config_path training/flexpcb_model.config \
    --trained_checkpoint_prefix training/model.ckpt-10856 \
    --output_directory flexpcb_graph
```

Рисунок 3.34 – Команда запуску моделі ЗНМ

Результати роботи згорткової нейронної мережі наведено на рисунку 3.35.

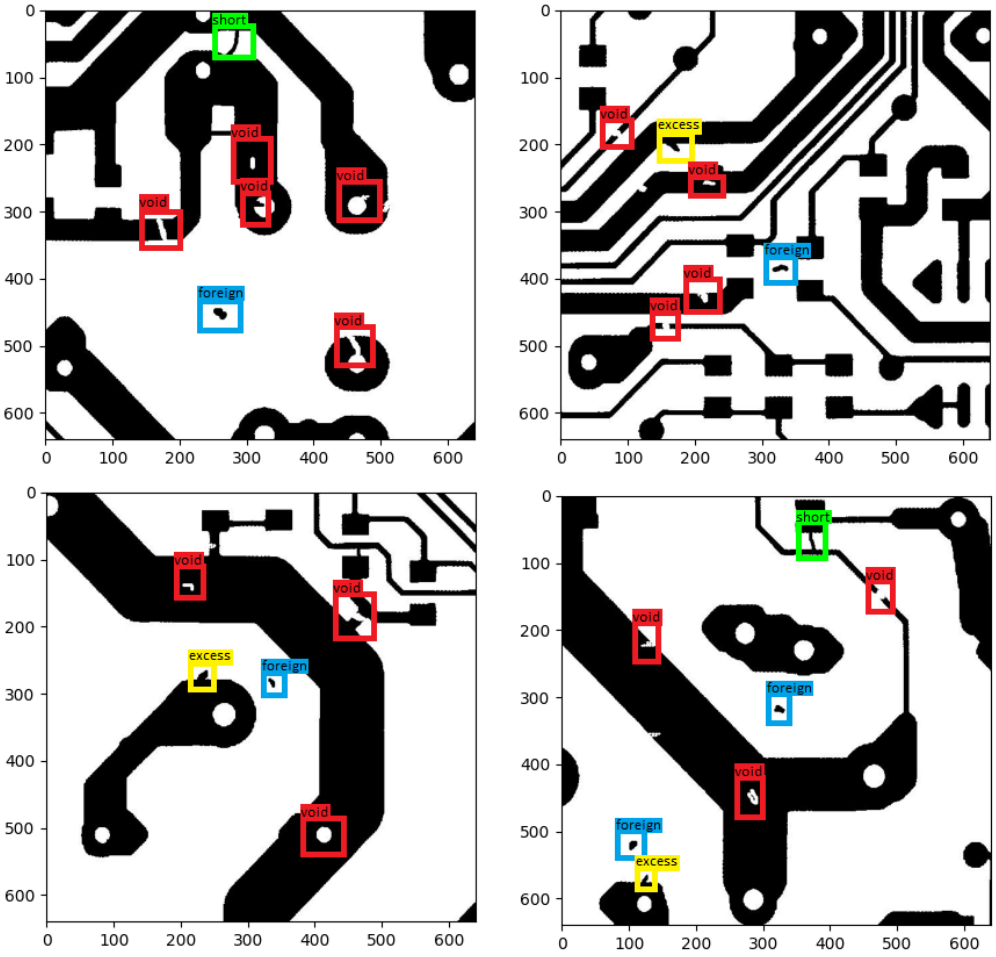

Рисунок 3.34 – Результат роботи розробленої ЗНМ

3.6 Висновки до розділу 3

У третьому розділі були розглянуті основні фреймворки, бібліотеки та інструменти для розробки систем комп’ютерного зору та згорткових нейронних мереж.

За допомогою OpenCV був розроблений класифікатор зображень, який порівнює вхідне й еталонне зображення, генерує файл з їх математичною різницею та підраховує сумарну площу різниці.

За допомогою фреймворку TensorFlow була розроблена модель згорткової нейронної мережі, яка може класифікувати зображення за типом дефектів, також був підготовлений та анотований датасет для її навчання.

За допомогою TensorFlow Object Detection API розроблена модель ЗНМ була адаптована для розпізнавання множинних об’єктів на зображенні та виконано тестування роботи програми.

4 АНАЛІЗ ЕФЕКТИВНОСТІ РОБОТИ РОЗРОБЛЕНОЇ ПРОГРАМИ

Для оцінки ефективності роботи моделі згорткової нейронної мережі необхідно оцінити ряд параметрів, а саме:

- швидкість навчання;
- точність розпізнавання.

4.1 Швидкість навчання

Навчання розроблюваної моделі ЗНМ було виконане за допомогою датасету з зображень, які трансформувались до розміру 100 × 100 пікселів, та тривало 3045 епох. Кожна епоха тривала від 143 до 268 секунд.

Таким чином, навчання моделі зайняло близько 90 годин. Навчання проводилось до моменту помітного підвищення коефіцієнту втрат, який свідчить про перенавчання моделі ЗНМ.

Для розрахунку коефіцієнту втрат під час розробки була використана категоріальна крос-ентропійна функція втрат, яка описується формулою (4.1):

$$H(x, y) = \sum_i x_i \log \frac{1}{y} = \sum_i x_i \ \ln y_i, \tag{4.1}$$

де H – коефіцієнт втрат;

x – бажана відповідь ЗНМ;

y – фактична відповідь ЗНМ.

Ця функція повертає речове число, що являє собою певне значення, яке характеризує точність розпізнавання категорії.

Для підрахунку загального коефіцієнта втрат береться середнє значення між крос-ентропійними функціями втрат для кожної з категорій. Далі за допомогою файлу з логами навчання був побудований графік залежності

коефіцієнту втрат від кількості епох навчання. Графік зображений на рисунку 4.1.

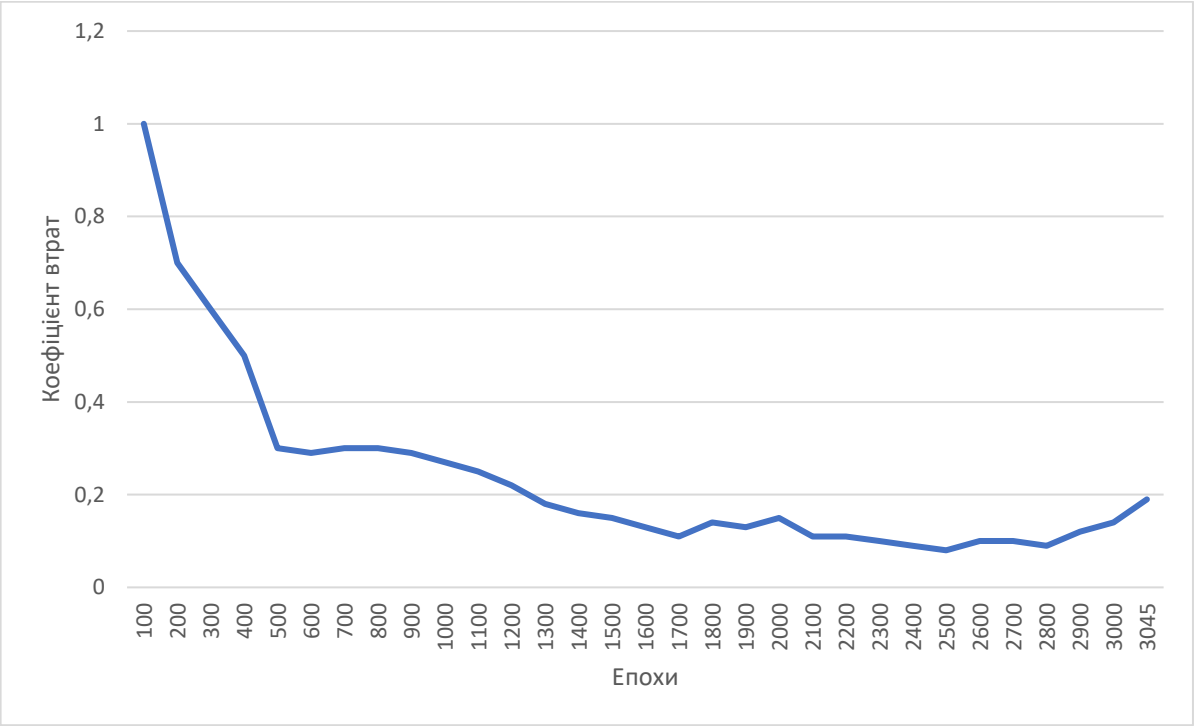

Рисунок 4.1 – Графік залежності коефіцієнту втрат від кількості епох навчання

На графіку можна побачити, що після певного моменту під час навчання коефіцієнт втрат перестає знижуватись, а далі починає зростати. Такий ефект називається перенавчанням моделі згорткової нейронної мережі. Він характеризується зниженням відносної точності розпізнавання тестових даних відносно даних для навчання. На цьому моменті навчання моделі було припинене.

Далі було досліджено залежність швидкості навчання та коефіцієнту втрат від розмірів зображення у датасеті. Для досліджень було обрано чотири розміри зображень у датасеті, а саме:

– 50 × 50 пікселів;
– 100 × 100 пікселів;
– 150 × 150 пікселів;

– 200 × 200 пікселів.

Було проведено по 10 епох навчання моделі з тестовим датасетом, який складався з зображень означених розмірів. Під час дослідження контролювались швидкість завершення однієї епохи навчання та коефіцієнт втрат. Результати дослідження представлені у таблиці 4.1 та таблиці 4.2.

Таблиця 4.1 – Залежність тривалості однієї епохи навчання від розміру зображення

Епохи	Розмір зображень			
	50 × 50 пікселів	100 × 100 пікселів	150 × 150 пікселів	200 × 200 пікселів
	Тривалість однієї епохи навчання, с			
1	73	174	378	1187
2	103	176	525	950
3	82	211	377	1045
4	121	255	569	869
5	112	202	43	914
6	95	268	547	1292
7	74	204	563	1014
8	85	149	387	1234
9	103	255	483	1271
10	73	143	396	1186

Таблиця 4.2 – Залежність коефіцієнту втрат від розміру зображення

Епохи	Розмір зображень			
	50 × 50 пікселів	100 × 100 пікселів	150 × 150 пікселів	200 × 200 пікселів
	Коефіцієнт втрат			
1	0,9161	0,6473	0,6023	0,592
2	0,8932	0,6399	0,595	0,5901
3	0,89	0,5583	0,5464	0,5876
4	0,8853	0,5042	0,5079	0,5841
5	0,8812	0,5026	0,4992	0,5107
6	0,8801	0,5001	0,497	0,4953
7	0,8709	0,499	0,4851	0,4804
8	0,8673	0,4983	0,4794	0,4625
9	0,8635	0,4969	0,4706	0,451
10	0,8591	0,4967	0,4681	0,4496

Як видно з таблиць, збільшення зображень у датасеті підвищує час на навчання та зменшує коефіцієнт втрат. На основі отриманих даних було прийняте рішення, що за відношенням витраченого часу до результатів оптимальним є розмір зображень у 100 × 100 пікселів.

4.2 Точність розпізнавання

Точність розпізнавання елементів, або пам’ять нейронної мережі – це найважливіший показник ефективності моделі. Пам’ять нейронної мережі характеризує точність визначення елементу на зображенні. Нижче наведено графік отриманих результатів оцінки пам’яті нейронної мережі для чотирьох класів елементів на зображенні (рисунок 4.2).

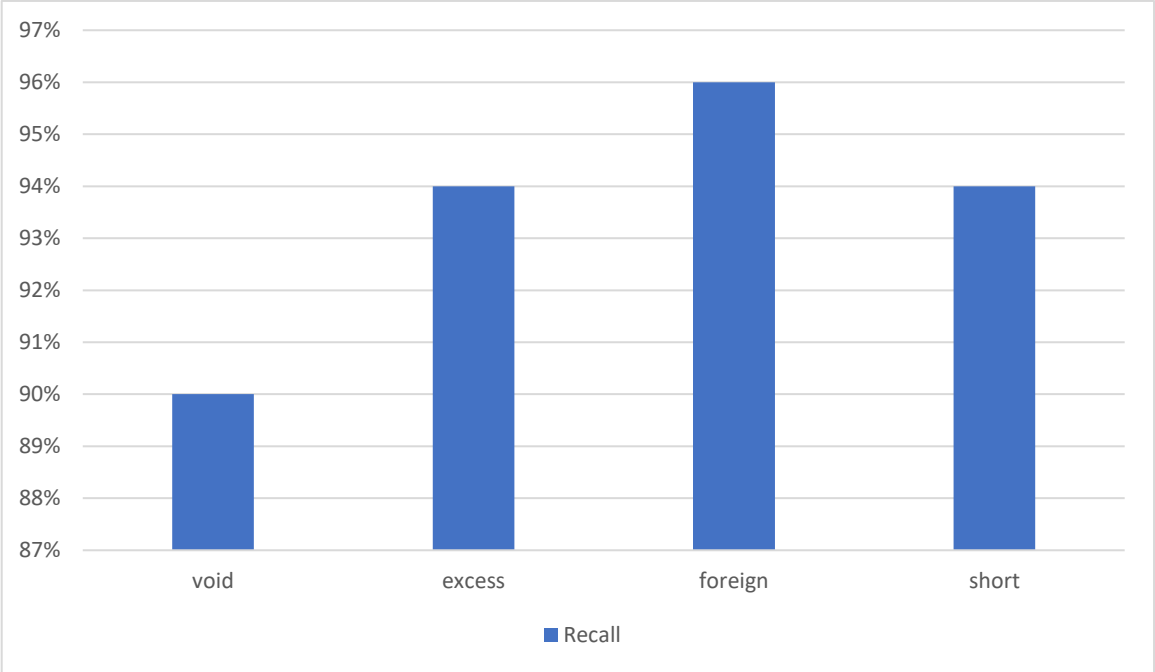

Рисунок 4.2 – Графік залежності точності розпізнавання від типу об’єкту

Середня точність роботи розроблюваної ЗНМ складає 93,5 %, що є задовільним результатом. З представленого вище графіку можна побачити, що точність розпізнавання різних типів об’єктів відрізняється. Це є результатом різної кількості тих чи інших дефектів на зображеннях у датасеті

(незбалансованість датасету) та деякої схожості ряду дефектів з елементами ДП, які не є дефектами (наприклад, замкнуті пустоти у доріжках та перехідні отвори). Далі як точність (пам’ять) нейронної мережі буде розглядатись середнє значення для всіх типів елементів.

Для оцінки реальної точності розпізнавання об’єктів був взятий набір з 10 бінаризованих тестових зображень, на яких нанесені дефекти. Усі дефекти були перераховані вручну та розсортовані по класах і далі ці зображення були передані у розроблювану модель ЗНМ, результати якої були записані та порівняні із вихідними у таблиці 4.3.

Таблиця 4.3 – Тестування якості розпізнавання дефектів

Клас дефекту	Номер зображення																			
	1		2		3		4		5		6		7		8		9		10	
	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат	Тест	Результат
void	6	6	9	7	3	3	5	5	4	4	7	7	2	2	8	7	0	0	6	6
excess	2	2	3	3	3	3	1	1	9	9	5	5	5	5	4	4	8	7	7	7
short	1	1	5	2	6	6	0	1	0	0	9	9	4	4	8	9	8	8	5	5
foreign	9	9	0	0	1	1	5	5	8	7	2	2	6	7	4	4	2	2	4	4
Сума	18	18	17	12	13	13	11	12	21	20	23	23	17	18	24	24	18	17	22	22

Як можна побачити, середня точність виділення об’єктів не нижча за 90 %, що є задовільним результатом.

У порівнянні з класифікатором, який був розроблений за допомогою OpenCV і порівнював вхідне зображення з еталонним, розроблена модель ЗНМ є більш універсальною, не потребує еталонного зображення та може з достатньою точністю класифікувати типи дефектів. Але, у свою чергу, класифікатор порівнює зображення значно швидше за згорткову нейронну мережу та використовує менше обчислювальних ресурсів.

Найоптимальнішим рішенням буде використовувати базовий класифікатор для пошуку виробів із дефектами у режимі тригера. Якщо класифікатор знаходить розбіжності еталонного та фактичного зображення, фактичне зображення передається до API, який за допомогою розробленої моделі ЗНМ знаходить і класифікує дефект.

4.3 Висновки до розділу 4

У четвертому розділі були проаналізовані основні параметри ефективності моделей згорткових нейронних мереж і була досліджена розроблювана модель.

Побудовано графік залежності коефіцієнту втрат від тривалості навчання моделі. Також проаналізовані пам’ять розроблюваної моделі ЗНМ і точність знаходження елементів на зображеннях відповідно до їх класів.

Виконано порівняння обох розроблених програмних рішень і на базі порівняння був запропонований оптимальний сценарій їх використання.

5 ОХОРОНА ПРАЦІ

До роботи на персональному комп’ютері долучаються особи, які пройшли ввідний інструктаж з охорони праці у службі охорони праці, первинний інструктаж з охорони праці на робочому місці та які вивчили інструкцію з охорони праці під час виконання робіт на комп’ютері.

Робота на комп’ютері вважається роботою з підвищеним рівнем небезпечності згідно з переліком робот з підвищеною небезпечністю, затвердженого наказом Госкомітету України з нагляду за охороною праці від 30.11.93 № 123 [41].

Площа приміщення, яка виділяється для обладнання одного робочого місця з персональною електронно-обчислювальною машиною (ПЕОМ), має бути не менше 6 $м^2$, а об’єм – не менше 20 $м^3$.

Робочі місця з ПЕОМ мають розміщуватися на відстані не менше 1 м від стіни з вікнами, так, щоб природне освітлення падало збоку, бажано зліва [41].

Відстань між бічними поверхнями моніторів не може бути менше за 1,2 м. Відстань між задньою стінкою одного монітору та екраном іншого має бути не менше за 2,5 м.

Прохід між рядами робочих місць має бути не меншим за 1 м. Конструкція робочого місця з ПЕОМ має забезпечувати оптимальне розміщення на робочій поверхні документів, монітору, системного блоку, принтера, клавіатури, і т. д. Системний блок і монітор встановлюються на основному робочому столі, як правило, з лівого боку.

Оптимальні розміри робочого столу, на якому розміщується ПЕОМ, мають становити: висота – 725 мм; довжина – 1400–1600 мм; ширина – 800 мм [41].

Конструкція робочого місця з ПЕОМ має забезпечувати підтримання оптимальної робочої пози (під час роботи сидячи): ступні ніг – на підлозі або підставці для ніг; стегна – горизонтально; передпліччя – вертикально; лікті – під кутом 70-90° до вертикальної площини; зап’ястя зігнуті під кутом не більше 20°

відносно горизонтальної площини; нахил голови – 15-20° відносно вертикальної площини [42].

Конструкція крісла користувача ПЕОМ має створювати умови для підтримки корпусу користувача ПЕОМ у оптимальному положенні зі збереженням природних вигинів хребта, а також має забезпечувати зниження статичного напруження м'язів шийно-плечової області спини та не має ускладнювати робочих рухів.

Крісло користувача ПЕОМ має включати такі основні елементи як підлокітники, спинку, сидіння, а також додатковий елемент – підставку для ніг.

Робоче сидіння користувача ПЕОМ має бути підйомно-поворотним: таким, що регулюється за висотою, кутом нахилу сидіння та спинки, за відстанню спинки до переднього краю сидіння, висотою підлокітників.

Покриття сидіння, спинки, підлокітників має бути виготовлено з вологовідштовхувального, такого, що не електризується, повітропроникного, пом'якшення та нековзкого матеріалу [42].

Принтер розміщується у зручному для користувача місці, так, щоб максимальна відстань до нього не перевищувала довжину витягнутої руки.

Розташування екрана дисплея у робочій зоні має передбачати зручність зорового спостереження у вертикальній площині під кутом +30° або -30° від нормальної лінії погляду користувача ПЕОМ.

Дисплей має бути обладнаний поворотною майданчиком, що дозволяє переміщувати його у горизонтальній і вертикальній площинах у межах +30° або -30° (праворуч-ліворуч), змінювати кут нахилу екрану до 10–15°.

Рівень освітленості на робочому столі має бути у межах 300-500 лк.

Рівні звуку й еквівалентні рівні звуку на робочому місці користувача ПЕОМ не мають перевищувати 50 дБА [42].

Параметри мікроклімату для приміщень, у яких розміщені персональні комп'ютери, мають бути наступними. У холодні періоди року: температура повітря 22–24 °С; швидкість руху повітря 0,1 м/с; відносна вологість повітря

60–40 %. Температура повітря може коливатися у межах 21-25 °С за збереження інших параметрів мікроклімату.

У теплі періоди року: температура повітря 23–25 °С; швидкість руху повітря 0,1–0,2 м/с; відносна вологість повітря 60–40 %. Температура повітря може коливатися у межах 22–26 °С за збереження інших параметрів мікроклімату.

Чистота повітря у робочій зоні має відповідати ГОСТ 12.1.005: наявність озону не має перевищувати 0,1 мг/м^3; наявність оксидів азоту не має перевищувати 5 мг/м^3; наявність пилу не має перевищувати 4 мг/м^3 [42].

ВИСНОВКИ

У ході виконання даної атестаційної роботи були розглянуті основні види тестування ГДП, їх особливості та вплив різних факторів на їх проведення. Проаналізовані процеси проведення тестування вхідних матеріалів і готових виробів. Розглянуті режими проведення тестування на електричну стійкість, протидію вологості, механічну стійкість, стійкість до ремонту, стійкість на вплив грибків і мікроорганізмів.

У результаті проведеного анвлізу були виділені основні проблеми та недоліки тих чи інших методів тестування ГДП.

Проаналізовано основні типи дефектів ГДП, умови, у яких вони можуть з’являтись, як їх виявляють та на які параметри готового виробу вони впливають.

У результаті аналізу існуючих методів контролю ГДП запропонований новий метод контролю якості ГДП, який поєднує в собі використання рентген-апарату й аналіз результатів за допомогою сучасних програмно-технічних рішень на базі нейронних мереж і систем комп’ютерного зору.

Розглянуті згорткові нейронні мережі, основні їх елементи, шари, типи активаційних функцій і їх вплив на проміжні результати роботи системи.

Визначена та розроблена архітектура програмних рішень, які будуть створені для реалізації запропонованого методу контролю ГДП.

Розглянуті основні фреймворки, бібліотеки та інструменти для розробки систем комп’ютерного зору та ЗНМ.

За допомогою бібліотеки OpenCV було розроблено класифікатор зображень, який базується на простій логіці, порівнює вхідне й еталонне зображення, генерує файл із їх математичною різницею та підраховує сумарну площу різниці на базі даних про щільність пікселів матриці рентген-апарату.

За допомогою фреймворку TensorFlow розроблено модель ЗНМ, яка може класифікувати зображення за типом дефектів, також був підготовлений та анотований датасет для її навчання.

За допомогою фреймворку TensorFlow Object Detection API розроблено модель ЗНМ була адаптована для розпізнавання множинних об’єктів на зображенні та виконано тестування роботи програми.

Виконано оцінку основних параметрів ефективності моделей ЗНМ і досліджена розроблювана модель.

Був визначений оптимальний розмір перетворення зображень для навчання моделі, а саме 100 × 100 пікселів, за умови використання якого тривалість однієї епохи навчання складає від 143 до 268 с , та коефіцієнт втрат в результаті проведення 10 епох складає 0,4967.

За результатами порівняння обох розроблених програмних рішень запропоновано оптимальний сценарій їх використання, а саме використання простого класифікатора на OpenCV як трігера, який в разі неспівпадіння вхідного зображення з еталонним, буде автоматично ініціювати перевірку вхідного зображення за допомогою нейронної мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з «Розробки й оформлення магістерської атестаційної роботи» для студентів другого (магістерського) рівня вищої освіти галузі знань 15 Автоматизація та приладобудування за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології освітні програми: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, В. В. Косенко, В. В. Євсєєв. – Харків: ХНУРЕ, 2019. – 55 с.

2. ДСТУ 3008: 2015 Інформація та документація. Звіти у сфері науки і техніки. труктура і правила оформлення. – К.: ДП «УкрНДНЦ», 2016. – 31 с.

3. Основи наукових досліджень: [навч. посіб.] / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала. – Кривий Ріг : КК НАУ, 2017. – 344 с.

4. Положення про організацію освітнього процесу в ХНУРЕ [Електронний ресурс] / Режим доступу: https://nure.ua/polozhennya-pro-organizatsiyu-osvitnogo-protsesu-v-hnure. – 29.08.2020. – Назва з екрану.

5. Положення про протидію академічному плагіату в ХНУРЕ [Електронний ресурс] / Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/-Polozhennya-pro-protidiyu-akademichnomu-plagiatu-v-HNURE----290-vid-28.04.2017.pdf. – 29.08.2020. – Назва з екрану.

6. Положення про роботу екзаменаційних комісій ХНУРЕ [Електронний ресурс] / Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/ Polozhennya-pro-poryadok-stvorennya-ta-organizatsiyu-roboti-ekzamenatsiynih-komisiy....pdf. – 29.08.2020. – Назва з екрану.

7. Положення про авторське право в ХНУРЕ [Електронний ресурс] / Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/ Polozhennya-pro-avtorske-pravo-v-HNURE.pdf. – 29.08.2020. – Назва з екрану.

8. Аналіз комп’ютерно-інтегрованих методів контролю гнучких друкованих плат. / В.С. Назаренко. // Автоматизація та приладобудування («Automation Dev. Electron. Devices» ADED-2020). – 2020. – С. 266–270.

9. IPC Guidelines and basic for flex PCBs [Електронний ресурс]. Режим доступу: https://www.protoexpress.com/blog/ipc-guidelines-and-testing-for-flex-pcbs. – 28.09.2020. – Назва з екрану.

10. Types of PCB | Different Types of Printed Circuit Board (PCB) [Електронний ресурс]. Режим доступу: http://www.electronicsandyou.com/blog/types-of-pcb-different-types-of-printed-circuit-board-pcb.html. – 18.10.2020. – Назва з екрану.

11. Brooks Douglas. Signal Integrity Issues and Printed Circuit Board Design/ Brooks Douglas // Prentice Hall PRT, 2003. – PP. 231–233.

12. Anaya Vardya and David Lackey The printed circuit board designer guide to… Flex and rigid-flex fundamentals /Anaya Vardya та David Lackey // LoozBooks, 2018. – 53 p.

13. Brooks Douglas. Signal Integrity Issues and Printed Circuit Board Design/ Brooks Douglas // Prentice Hall PRT, 2003. – PP. 103-109.

14. Bending, forming and Flexing Printed Circuits [Електронний ресурс]. Режим доступу: https://docplayer.net/11986077-Bending-forming-and-flexing-printed-circuits.html. – 20.10.2020. – Назва з екрану.

15. Structural integrity and the challenges of rigid flex pcb design [Електронний ресурс]. Режим доступу: https://resources.altium.com/p/structural-integrity-and-the-challenges-of-rigid-flex-pcb-design. – 20.10.2020. – Назва з екрану.

16. Flexible circuit technology and its applications [Електронний ресурс]. Режим доступу: https://www.lboro.ac.uk/microsites/mechman/research/ipm-ktn/pdf/Technology_review/flexible-circuit-technology-and-its-applications.pdf. – 28.10.2020. – Назва з екрану.

17. Technical Specifications [Електронний ресурс]. Режим доступу: https://www.airborn.com/products/flexible-circuits/flex-technical-specifications 02.11.2020. – Назва з екрану.

18. Rigid-flex design [Електронний ресурс]. Режим доступу: https://www.altium.com/documentation/altium-designer/rigid-flex-design-ad. – 05.11.2020. – Назва з екрану.

19. PCB thermal dimensional Stability [Електронний ресурс]. Режим доступу: https://www.sfcircuits.com/pcb-school/pcb-thermal-dimensional-stability 09.11.2020. – Назва з екрану.

20. James M. Kirkpatrick. Electronic Drafting And Printed Circuit Board Design / James M. Kirkpatrick // CENGAGE LEARNING, INC 1993. – PP. 51–57.

21. Thomas Stearns McGraw. Flexible Printed Circuitry 1st Edition/ Thomas Stearns McGraw// Hill Education 1995. – PP. 179–192.

22. Flex and rigid-flex pcb design fundamentals [Електронний ресурс]. Режим доступу: https://ieee.li/pdf/essay/flex_and_rigid-flex_fundamentals.pdf. 15.11.2020. – Назва з екрану.

23. Design guide for flexible PCB [Електронний ресурс]. Режим доступу: http://www.stormcircuit.net/design-guide-for-flexible-pcb.html. – 18.11.2020. – Назва з екрану.

24. The Via Issue [Електронний ресурс]. Режим доступу: https://iconnect007.uberflip.com/i/131107491-design007-nov2020. – 18.11.2020. – Назва з екрану.

25. Balancing the electrical and mechanical requirements for flexible circuits [Електронний ресурс]. Режим доступу: https://docplayer.net/17798292-Balancing-the-electrical-and-mechanical-requirements-of-flexible-circuits-mark-finstad-applications-engineering-manager-minco.html. – 21.11.2020. – Назва з екрану.

26. Navigating the Complex World of Flex Circuit Assembly [Електронний ресурс]. Режим доступу: http://magazines007.com/pdf/SMT-Mar2017.pdf. – 21.11.2020. – Назва з екрану.

27. X-ray inspection in PCB assembly [Електронний ресурс]. Режим доступу: https://www.pcbcart.com/article/content/Xray-inspection-in-PCB-assembly.html. – 25.11.2020. – Назва з екрану.

28. How Do Convolutional Layers Work in Deep Learning Neural Networks? [Електронний ресурс]. Режим доступу: https://machinelearningmastery.com/-convolutional-layers-for-deep-learning-neural-networks/. – 26.11.2020. – Назва з екрану.

29. ML Practicum: Image Classification [Електронний ресурс]. Режим доступу: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks. – 26.11.2020. – Назва з екрану.

30. Understanding of Convolutional Neural Network (CNN) – Deep Learning [Електронний ресурс]. Режим доступу: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. – 27.11.2020. – Назва з екрану.

31. CS231n: Convolutional Neural Networks for Visual Recognition [Електронний ресурс]. Режим доступу: http://cs231n.stanford.edu/. – 27.11.2020. – Назва з екрану.

32. Convolutional Neural Network Architecture: Forging Pathways to the Future [Електронний ресурс]. Режим доступу: https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/. – 27.11.2020. – Назва з екрану.

33. TensorFlow Official Repository [Електронний ресурс]. Режим доступу: https://github.com/tensorflow/tensorflow. – 28.11.2020. – Назва з екрану.

34. OpenCV Official Docs [Електронний ресурс]. Режим доступу: https://opencv.org/about/. – 28.11.2020. – Назва з екрану.

35. NumPy official Docs [Електронний ресурс]. Режим доступу: https://numpy.org/. – 28.11.2020. – Назва з екрану.

36. The PyPA recommended tool for installing Python packages. [Електронний ресурс]. Режим доступу: https://pypi.org/project/pip/. – 29.11.2020. – Назва з екрану.

37. Contours: Getting Started [Електронний ресурс]. Режим доступу: https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html. – 29.11.2020. – Назва з екрану.

38. Module: tf [Електронний ресурс]. Режим доступу: https://www.tensorflow.org/api_docs/python/tf. – 30.11.2020. – Назва з екрану.

39. TensorFlow Object Detection API [Електронний ресурс]. Режим доступу: https://github.com/tensorflow/models/tree/master/research/object_detection. – 30.11.2020. – Назва з екрану.

40. Integrating Keras with Tensorflow Object Detection API [Електронний ресурс]. Режим доступу: https://abhijit-2592.github.io/Keras-with-TFODAPI/. – 30.11.2020. – Назва з екрану.

41. Охрана труда при выполнении работ на компьютере [Електронний ресурс]. Режим доступу: https://ohranatrud-ua.ru/spravochnaya-informatsiya/2272-okhrana-truda-pri-vypolnenii-rabot-na-kompyutere.html. – 01.12.2020. – Назва з екрану.

42. Инструкция по охране труда при работе с персональными компьютерами [Електронний ресурс]. Режим доступу: https://businessforecast.by/partners/646/instrukcija-po-ohrane-truda-pri-rabote-29/. – 01.12.2020. – Назва з екрану.