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)

Розробка автоматизованого методу контролю друкованих плат
(тема)

Виконав: студент 2 курсу, гр. КІТПВм-19-1
Васільєв В. А.
(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 Васільєву Владиславу Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка автоматизованого методу контролю друкованих плат

затверджена наказом по університету від 02.11. 2020 р. № 1511 Ст

2. Термін подання студентом роботи до екзаменаційної комісії . . 2020 р.

3. Вихідні дані до роботи 3.1 Необхідна точність методу – 80 %.

3.2 Розмір контрольованих друкованих плат – 150 мм · 150 мм.

3.3 Якість зображення – 1024 · 1024 пікселів.

3.4 Реалізувати можливість використання програми на багатоядерному ПК.

3.5 Забезпечити можливість інтеграції нейромержі у інші програми.

4. Перелік питань, які потрібно опрацювати у роботі

4.1 Вступ

4.2 Аналіз існуючих методів контролю друкованих плат

4.3 Вибір параметрів обраного методу контролю друкованих плат

4.4 Розробка моделі обраного методу контролю друкованих плат

4.5 Розробка автоматизованої системи контролю друкованих плат

4.6 Проведення експерименту та аналіз результатів

4.7 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 14 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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Аналіз технічного завдання*	01.09.2020 р.	Виконав
2	*Аналіз сучасних методів контролю*		Виконав
	друкованих плат	10.09.2020 р.	
3	*Вибір параметрів обраного методу*		Виконав
	контролю друкованих плат	17.09.2020 р.	
4	*Розробка моделі*	30.09.2020 р.	Виконав
5	*Розробка автоматизованої*		Виконав
	системи для контролю друкованих		
	плат	20.11.2020 р.	
6	*Проведення експерименту та аналіз*		Виконав
	результатів	23.11.2020 р.	
7	*Оформлення пояснювальної записки*	30.11.2020 р.	Виконав
8	*Подання роботи до ДЕК*	07.12.2020 р.	Виконав

Дата видачі завдання

Студент _____ Васільєв В. А

(підпис) (прізвище, ініціали)

Керівник роботи _____ Боцман І. В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 81 с., 43 рис., 8 табл., 40 джерел.

ДРУКОВАНА ПЛАТА, ЗОБРАЖЕННЯ, КОНТРОЛЬ, МАШИННЕ НАВЧАННЯ, НЕЙРОМЕРЕЖА

Об’єкт дослідження – процес контролю друкованих плат.

Предмет дослідження – друкована плата з нанесеним провідниковим рисунком.

Методи дослідження – теоретичні дослідження, порівняльний аналіз, методи машинного навчання.

Мета магістерської атестаційної роботи – автоматизація процесу візуального контролю друкованих плат.

У магістерській атестаційній роботі проведено аналіз існуючих методів контролю друкованих плат на виробництві, визначені основні відмінності існуючих методів, їх переваги та недоліки, обрано оптимальний метод контролю друкованих плат. У другому розділі розраховано необхідні параметри нейромережі для створення автоматизованого методу контролю.

У третьому розділі створено нейромережу, яка використовується для візуального контролю друкованих плат.

У четвертому розділі досліджено основні можливості нейромережі з визначення дефектів та її ефективність.

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

ABSTRACT

Explanatory note: 81 p., 43 figures, 8 tables, 40 sources.

CONTROL, IMAGE, MACHINE LEARNING, NEURAL NETWORK, PRINTED CIRCUIT BOARD

The object of research – the process of control of printed circuit boards.

The subject of research – a printed circuit board with a printed conductor pattern

Research methods – theoretical research, comparison analysis, machine learning methods.

The purpose of the master's certification work is to automate the process of visual inspection of printed circuit boards.

The master's attestation work analyzes the existing methods of control of printed circuit boards in production, identifies the main differences of existing methods, their advantages and disadvantages, chose the optimal method of control of printed circuit boards. The second section calculates the necessary neural network parameters to create an automated control method.

In the third section, based on the results obtained in the second section, a neural network was created, which is used for visual control of printed circuit boards.

The fourth section explores the main capabilities of the neural network to identify defects and its effectiveness.

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

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CLI – Command Line Interface. Інтерфейс командного рядка;

ConvNet – Convolutional Neural Networks;

OpenCV – Open Source Computer Vision Library;

АЛТ – автоматизований лазерний тест;

АОК – автоматизований оптичний контроль;

ДП – друкована плата;

РВК – ручний візуальний контроль;

СКВ – система контролю версій;

УФ – ультрафіолет.

ВСТУП

Останнім часом електроніка охоплює все більше та більше різноманітних сфер життя. Але для виготовлення якісної продукції необхідні високоточні методи тестування її якості на виробництві. Одним з важливих завдань сучасного приладобудування є тестування друкованих плат (ДП) на виробничий брак.

Серед багатьох видів контролю, які використовуються на виробництві ДП, можна виділити оптичний і візуальний контроль. Такі види контролю призначені для перевірки якості ДП ще до встановлення на неї компонентів, що дозволяє знайти проблему на виробництві як можна раніше. Також такі види контролю дозволяють виявити проблемні ділянки виробництва або похибки, закладені ще на етапі проектування. Для проектування таких систем висуваються основні вимоги:

– зменшення людського фактору;

– зменшення вартості контролю;

– спрощення переналаштування засобів контролю.

Неефективний контроль на етапі виготовлення ДП може призвести до значних витрат на етапі тестування готового пристрою, створеного на основі неякісної ДП.

Системи машинного навчання мають велику кількість переваг над іншими методами програмних алгоритмів, а саме:

– схожість у роботі з людським мисленням, що дає можливість використання таких програм без заздалегідь зазначеного еталону;

– можливість використання отриманих раніше системою знань на нових виробах, тобто відпадає необхідність переналаштування на кожний окремий випадок;

– можливість продовження навчання під час роботи для збільшення ефективності контролю та зменшення похибок;

– ефективність роботи таких систем вища ніж за умови використання звичайних алгоритмів;

– відсутність людського фактору.

Таким чином, тема роботи є актуальною, оскільки на поточний час існує необхідність більш сучасних та універсальних методів контролю ДП. Основний курс розвитку науки – спрощення праці людини, зменшення людського фактору та збільшення продуктивності виробництва.

Об’єкт дослідження – процес контролю друкованих плат.

Предмет дослідження – друкована плата з нанесеним провідниковим рисунком.

Методи дослідження – теоретичні дослідження, порівняльний аналіз, методи машинного навчання.

Мета магістерської атестаційної роботи – автоматизація процесу візуального контролю друкованих плат.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести огляд існуючих методів контролю ДП;

– провести аналіз методів контролю ДП на виробництві;

– провести вибір та обґрунтування методу контролю ДП;

– розробити математичну модель методу контролю ДП;

– розробити програмний модуль для контролю ДП;

– провести експериментальне дослідження для оцінки створеної системи;

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

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи контролю якості друкованих плат

На виробництві для контролю якості друкованих плат використовуються внутрішньосхемні та функціональні методи контролю.

Функціональний метод контролю дозволяє перевіряти якість готового виробу на основі ДП. Основними завданнями функціонального контролю ДП є:

– електричне тестування готового виробу в межах максимальних і мінімальних режимів напруги;

– емуляція електричних навантажень на готовий виріб;

– вимірювання параметрів вхідних і вихідних сигналів;

– збір та обробка статистичної інформації [8].

Як правило, для проведення функціонального тестування ще під час проектування ДП закладаються крайові контакти для під’єднання тестерів.

Внутрішньосхемний контроль призначено для тестування окремих елементів готового виробу, таких як ДП, електронні компоненти, мікросхеми, встановленні у виробі тощо. Цей вид контролю поділяється на:

– цифровий контроль: перевірка цифрових мікросхем на відповідність таблиць істинності;

– аналоговий контроль: перевірка елементів виробу на короткі замикання або обриви, перевірка елементів на наявність заводського браку, перевірка номіналів встановлених елементів, перевірка наявності та правильності встановлення елементів виробу.

Вибір конкретного обладнання залежить від можливостей та обраного методу контролю конкретного виробника. Основними методом для контролю ДП на виробництві є електричний та візуальний контроль.

Електричний контроль дозволяє перевірити ДП на пошкодження або заводський брак із великою точністю, але вимагає великих грошових або часових затрат, також такі методи не завжди є універсальними і потребують переналаштування на кожний окремий вид ДП [9]. Електричний контроль може базуватися на наступних методах:

– метод кліпс і пробників – універсальний і недорогий метод, але він потребує високих часових затрат і вимагає високої кваліфікації співробітників;

– метод адаптера – вимагає виготовлення тестового адаптера для кожного виробу, що потребує закладання такої можливості ще на етапі проектування, і такий адаптер не буде універсальним;

– метод рухомих зондів – не вимагає високих затрат на переналаштування, але накладає жорсткі умови на проектування виробів і має невисоку продуктивність.

Більшість систем контролю залежать від певного виду випромінювання світла, такого як видиме світло, лазери та рентген. Вони отримують дані за допомогою обробки зображень у режимі реального часу для виявлення та вимірювання дефектів ДП.

Хоча цим автоматизованим інспекційним системам не потрібно фізично контактувати з випробовуваною друкованою платою, вони демонструють високу повторюваність та ефективно усувають суб'єктивність при огляді та вимірюванні дефектів. Системи перевірки випробувань структурних процесів можуть залучати різні методи контролю, залежно від складності складання друкованої плати [10]. Вони можуть включати:

– ручний візуальний контроль (РВК);

– автоматизований оптичний контроль (АОК);

– автоматизований лазерний тест (АЛТ);

– рентгенівська флюороскопічна система;

– система рентгенівського ламінування [11].

Візуальний контроль може виконуватися як і комп’ютером (АОК) у автоматичному режимі, так і людиною (РВК). У автоматичному режимі кожна окрема ДП порівнюється із еталонним зразком і виявляються розбіжності у провідниковому рисунку або захисному шарі. Такий метод теж потребує переналаштування на кожну окрему плату, але є можливість використання одночасно декількох еталонів із можливістю автоматично визначати необхідний еталон за допомогою спеціальних маркерів.

Візуальний контроль дає можливість швидко визначити помилки, які виникли під час виготовлення, наприклад:

– механічне пошкодження ДП – відколи, які можуть бути викликані дефектом сировини або необережним поводженням із готовим виробом;

– дефекти виготовлення монтажних отворів, наприклад заповнення отвору припоєм або пальною маскою, що унеможливить подальшу роботу із цим виробом;

– зміщення рисунку на ДП, що може унеможливити подальшу роботу з виробом;

– порушення геометрії контактних майданчиків [12].

Найпростіша форма перевірки друкованих плат – РВК. Для проведення такого типу випробувань працівник дивиться на плату неозброєним оком або за допомогою збільшення. Працівники порівнюють плату з проектною документацією, щоб переконатися, що всі технічні характеристики були дотримані. Тип дефектів, які вони шукають, варіюється залежно від виду плати, яку вони перевіряють, та компонентів на ній.

Може бути корисним виконувати РВК майже після кожного кроку в процесі виробництва ДП, включаючи збірку. Персонал перевіряє майже всі аспекти плати та шукає різні загальні дефекти кожного з них.

Типовий візуальний контрольний список інспекції друкованих плат може містити такі пункти:

– перевірка товщини плати;

– перевірка габаритів плати;

– перевірка струмопровідних малюнків;

– перевірка плати на наявність дефектів [13].

РВК має ряд переваг перед іншими видами перевірки. Завдяки своїй простоті він недорогий – вам не потрібно будь-яке спеціалізоване обладнання, крім лінз або мікроскопів [10]. Але такий метод контролю має багато недоліків, таких як:

– велика ймовірність людської похибки;

– невелика швидкість контролю;

– швидка втомлюваність людини і як наслідок – збільшення ймовірності похибки [14].

Тому візуальний контроль слід використовувати разом з іншими методами контролю.

1.2 Системи автоматизованого візуального контролю ДП

Системи візуального та оптичного контролю використовуються у автоматизованому або автоматичному режимі. Використання таких систем дозволяє зменшити людський вплив на контроль ДП, зменшити рівень браку та затрати на виготовлення готового виробу.

Такі системи можуть працювати як у автоматизованому режимі, де людина бере участь у всіх етапах тестування, а також у етапі прийняття рішень, так і у автоматичному режимі за умови встановлення на конвеєр. Також такі системи можуть мати УФ-лампи для дослідження ДП в ультрафіолетовому спектрі, лазерні установки для дослідження товщини виробу, апарат для зчитування штрих-кодів для визначення необхідної програми тестування тощо [15].

Автоматизовані системи потребують високої кваліфікації персоналу та мають високу вартість, але вони дають найвищу точність контролю та дозволяють порівнювати велику кількість параметрів із еталонним зразком, наприклад:

– наявність необхідних елементів на ДП;
– полярність встановлених елементів;
– механічні пошкодження;
– якість нанесення захисних покриттів;
– інспекція отворів;
– інспекція монтажу компонентів на контактні місця або у отворах [16].

Але такі системи непрактично використовувати у штучному або малосерійному виробництві через їхню високу вартість і необхідність налаштування на кожний окремий виріб. Приклад автоматизованої системи контролю ДП наведено на рис. 1.1.

Рисунок 1.1 – Приклад автоматизованої системи оптичного контролю ДП SMT IS-C [17]

Для використання у штучному або малосерійному виробництві існують менш дорогі апарати, але вони дають гірший результат і потребують більше людського втручання, приклад такої системи наведено на рис. 1.2.

Рисунок 1.2 – Приклад автоматизованої системи оптичного контролю ДП SMT IS-S [17]

1.3 Штучні нейронні мережі

Одним із можливих варіантів автоматизації методів візуального контролю є використання штучних нейронних мереж для класифікації об’єктів на зображенні.

Штучні нейронні мережі – це математична модель, яка заснована на принципах будови людського мозку. Така модель може використовуватися для великої кількості задач, які не піддаються автоматизації іншими методами, наприклад:

- розпізнавання образів на зображенні або відео;
- приймання рішень із керування системою на основі вхідних даних;
- кластеризація – розбиття вхідних множин на заздалегідь невідомі класи;
- прогнозування результатів на основі історичних даних;
- предикативне стиснення даних для зменшення розміру даних;
- аналіз існуючих даних для вирахування певних метрик;
- оптимізація параметрів на основі історичних даних [18].

Штучні нейронні мережі складаються з великої кількості пов’язаних між собою вузлів, які також називають нейронами через те, що вони схожі за будовою із нейронами у людському мозку (рис. 1.3), кожен з яких зв’язаний входами із одним або декількома нейронами та видає результат простих математичних операцій у вихід, який у свою чергу зв’язаний із одним або декількома нейронами.

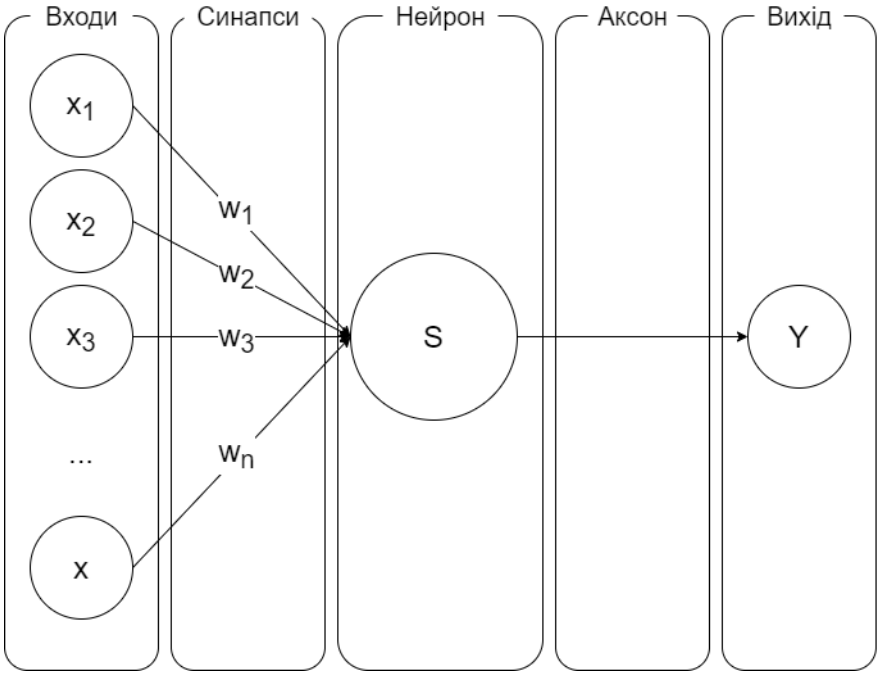


Рисунок 1.3 – Структура нейрона

Вихідний сигнал кожного вузла отримується за допомогою виконання цих операцій і параметрів, які записані всередині цього вузла за формулою (1.1)

$$S = \sum_{i=1}^{n} x_i w_i \,, \tag{1.1}$$

де n – кількість входів нейрона;

x_i – значення і-го входу;

w_i – вага і-го синапсу.

За допомогою з’єднання цих вузлів і точного встановлення параметрів кожного із них можливо створювати складні програми, які можуть виконувати функції, що практично неможливо задавати у вигляді програмних алгоритмів, також такі мережі можуть навчатися для більш точного виконання поставленої задачі.

Штучні нейронні мережі використовуються у найсучасніших досягненнях зі створення штучного інтелекту. Вони можуть використовуватися у різноманітних задачах, наприклад, визначення голосу, визначення зображення, робототехніка, відновлення частин зображення або контроль якості виробів [19].

Штучні нейрони працюють за допомогою порівняння отриманих сигналів із функцією, яка задана у його параметрах. Вихідний сигнал аксона визначається за формулою (1.2).

$$Y = f(S), \tag{1.2}$$

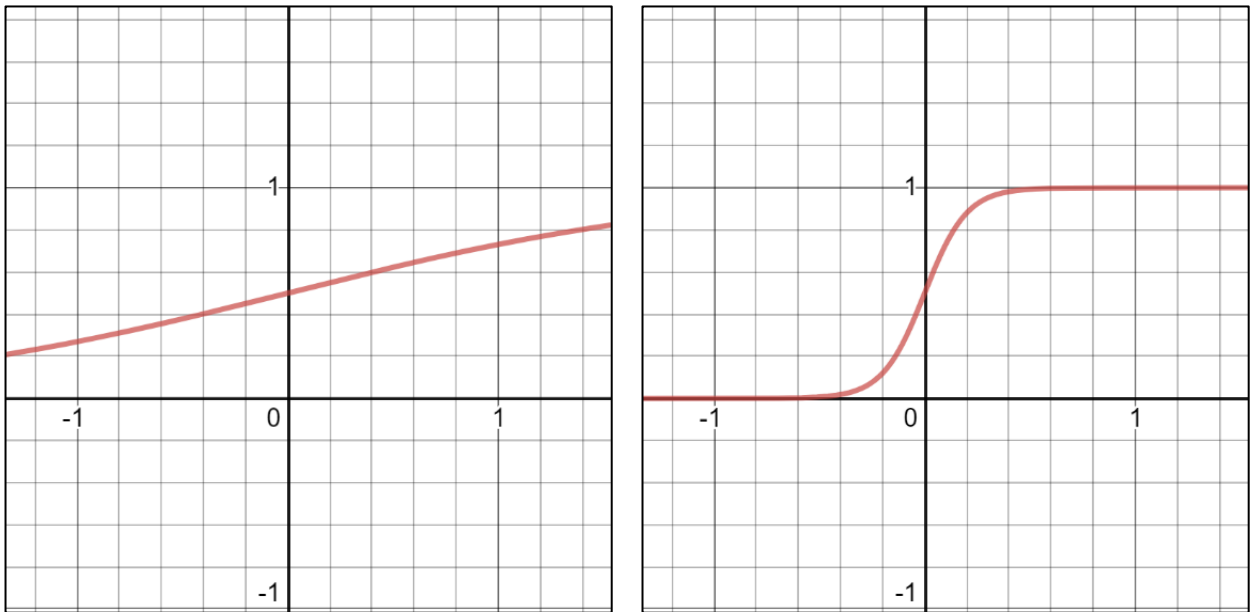
де f – активаційна функція.

Зазвичай як активаційну функцію використовують сигмоїду, формула (1.3), через те, що дана функція може бути диференційована на всій осі абсцис і має просту похідну, формула (1.4) [20]:

$$f(x) = \frac{1}{1 - e^{-ax}}, \tag{1.3}$$

$$f'(x) = af(x)\big(1 - f(x)\big). \tag{1.4}$$

За умови зменшення параметру a сигмоїда стає більш пологою, а за умови збільшення – наближається до ступінчастої функції (рис. 1.4).



а б

а) графік сигмоїди при $a = 1$; б) графік сигмоїди при $a = 10$

Рисунок 1.4 – Графік сигмоїди у залежності від параметра a

Тобто кожна штучна нейронна мережа складається з певної кількості з’єднаних між собою штучних нейронів, серед яких є вхідний шар, що приймає вхідні параметри, вихідний шар, який видає отриманий результат, а також прихований шар, який виконує математичні перетворення над вхідними даними. Вихідний ы вхідний шари існують у одиничному форматі, але прихованих шарів може бути декілька, і вони можуть дуже суттєво відрізнятися за будовою та кількістю нейронів у залежності від поставленого перед нейромережею завдання.

Нейрони у нейромережі можуть бути як із повним з’єднанням, тобто кожен із нейронів у одному шарі з’єднано із кожним нейроном у наступному шарі, так і із частковим з’єднанням, коли кожен нейрон у певному шарі з’єднано лише із певною кількістю нейронів у наступному шарі (рис. 1.5) [21].

Зазвичай у складі нейромережі можна побачити обидва види нейронів, наприклад нейрони із повним з’єднанням зазвичай встановлюються у останньому шарі нейронів перед отриманням вихідного значення.

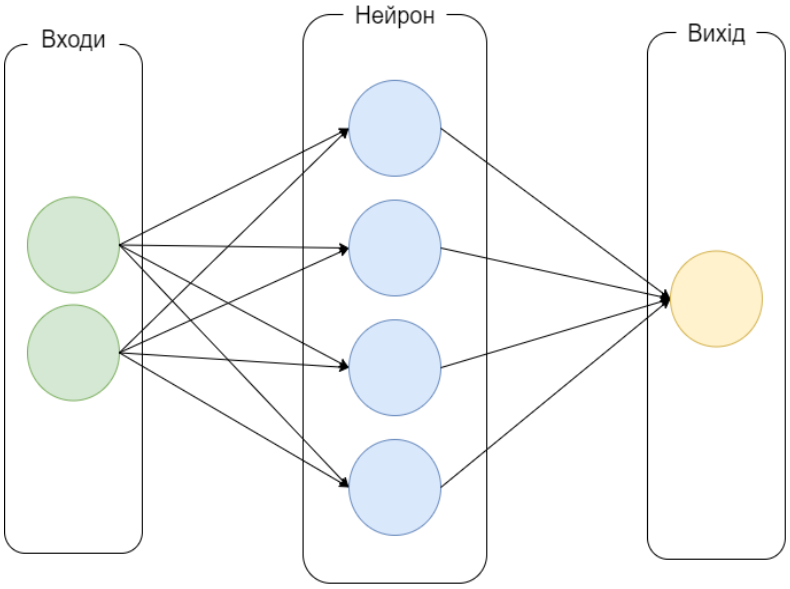


Рисунок 1.5 – Структура нейромережі

Для навчання таких мереж необхідна відносно велика кількість вхідних даних і набір тестових даних, у яких заздалегідь відомий результат, котрий необхідно отримати. Таке навчання потребує втручання людини, тобто результати, які отримуються у ході роботи такої мережі, порівнюються з результатами, які отримані у ході людського мислення. Для підналаштування параметрів нейронів можуть використовуватися декілька методів. Наприклад, генетичний алгоритм, який використовує алгоритм створення нащадків від штучних нейронних мереж із найкращими показниками, змішуючи їх параметри та додаючи можливість мутацій [22].

Також одним із найпоширеніших методів є метод зворотного поширення помилки. Даний метод полягає у поширенні сигналів помилки від виходів мережі до її входів, тобто у напрямку, зворотному від нормального режиму роботи [23]. Навчання за таким алгоритмом можна уявити як задачу оптимізації. Для навчання потрібно, щоб було визначено функцію оцінки, яка, як правило, залежить від вхідних сигналів мережі та неявно – від всіх її параметрів. Одним із найпоширеніших прикладів функції оцінювання є функція суми квадратів відстаней від вихідних сигналів мережі до їх необхідних значень, формула (1.5):

$$H = \frac{1}{2} \sum_{\tau \in out} (Z(\tau) - Z^*(\tau))^2, \tag{1.5}$$

де $Z(\tau)$ – необхідне значення вихідного сигналу.

Незважаючи на те, що даний алгоритм може бути дуже успішним, він має свої недоліки, які можуть призвести до навчання, яке буде займати дні або тижні, або взагалі можна не отримати бажаний результат:

– у процесі навчання параметри внаслідок корекції можуть стати завеликими, що може призвести до паралічу мережі;

– неправильно вибраний розмір кроку може призвести до завеликої помилки або довгого часу навчання мережі [24].

1.4 Згорткова нейронна мережа

Для розпізнавання візуальних образів було створено згорткові нейронні мережі (ConvNet). Їх принцип роботи будується на схемі з’єднання нейронів у зоровій корі тварин. Окремі нейрони реагують лише на невелику область зору, а їхня кількість дозволяє переконатися, що вони покривають усю необхідну область зору. Такі мережі можуть використовуватися у великій кількості програм для:

– розпізнавання зображення або відео;

– аналізу та класифікації зображення;

– відновлення частин зображення або відео.

Архітектура таких мереж будується на знанні, що вхідною інформацією є зображення, що дозволяє значно зменшити необхідну кількість необхідних розрахунків. У звичайних нейромережах кількість параметрів у першому шарі нейронів під час обробки зображень може досягати дуже великих значень, наприклад, для зображення $32 \text{ пікселя} \cdot 32 \text{ пікселя} \cdot 3 \text{ кольори}$ необхідно $32 \cdot 32 \cdot 3 = 3072$ параметри на кожен повністю підключений нейрон для першого шару, якщо розмір зображення складає

200 пікселів · 200 пікселів · 3 кольори, то необхідно мати $200 \cdot 200 \cdot 3 = 120{,}000$ параметрів на кожен повністю підключений нейрон. Для виконання складних завдань таких нейронів може бути декілька, що може складатися у програму, яка потребує багато ресурсів і часу для виконання розрахунків.

Тому для вирішення такої проблеми було обрано архітектуру побудови шарів у трьох вимірах, що дозволяє зменшити кількість необхідних з'єднань (рис. 1.6).

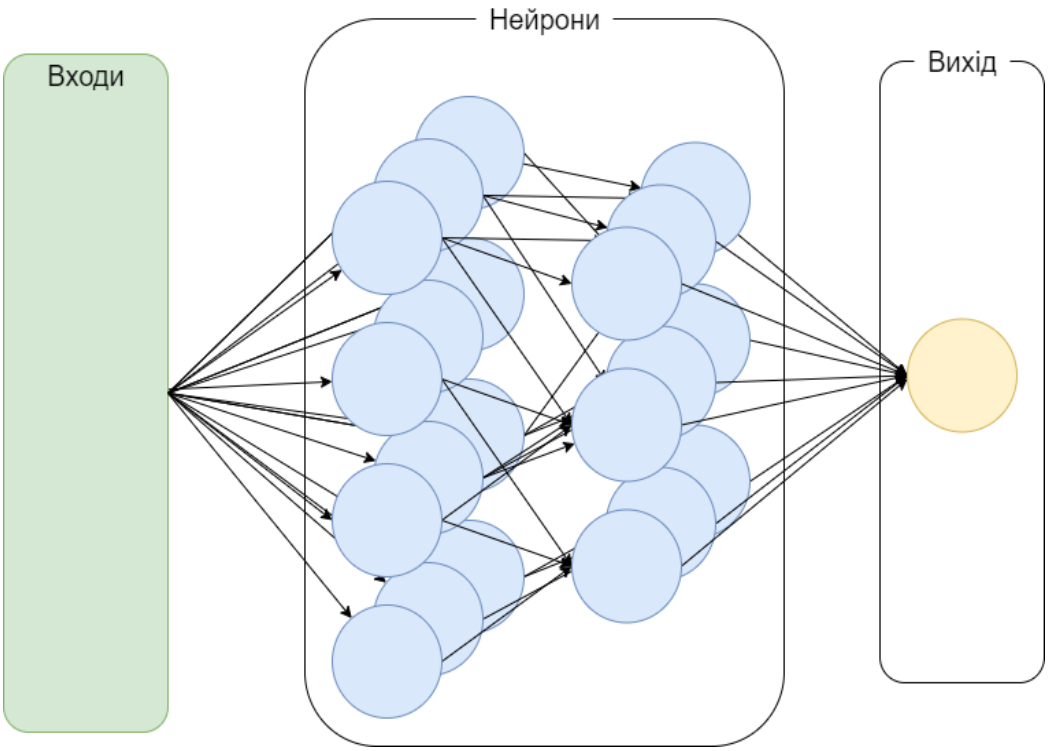


Рисунок 1.6 – Тривимірна побудова шарів нейромережі

Така тривимірна побудова дозволяє з'єднувати нейрони у кожному шарі лише з невеликою кількістю нейронів у попередньому шарі із вихідним шаром малого розміру, наприклад, для класифікації об'єкта на зображенні вихідний шар може складатися з $1 \cdot 1 \cdot$ кількість категорій для класифікації нейронів.

Архітектура такої мережі складається з наступних типів шарів (рис. 1.7):

– вхідний – зберігає сирі дані з пікселів на зображенні, розмір шару для зображення розміром 32 · 32 пікселів – 32 · 32 · 3 нейронів;

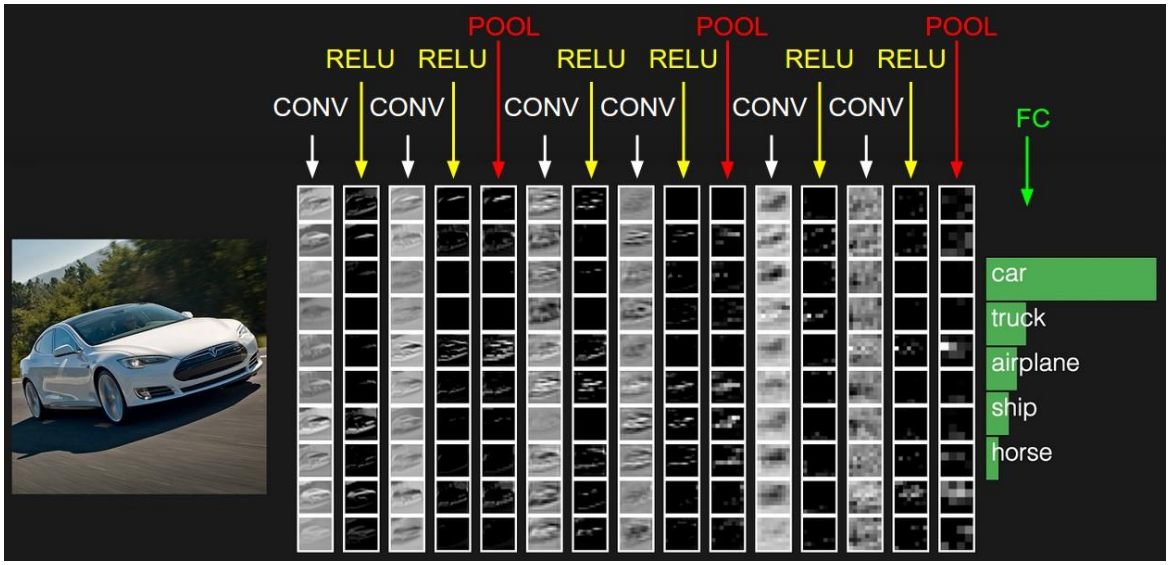


Рисунок 1.7 – Приклад ConvNetмережі [25]

– CONV – розраховує вихідні показники нейронів, що підключені до невеликої області вхідних нейронів, видає скалярний добуток між параметрами та вхідними параметрами, розмір даного шару збільшується до 32 · 32 · 12. Цей шар складається з великої кількості малих фільтрів, які можуть навчатися. Кожен фільтр має невеликі розміри, але може простягатися крізь увесь обсяг вхідних даних. Наприклад, типовий фільтр у ConvNet може бути розміром 5 · 5 · 3, під час роботи даного шару переміщуються фільтри по вхідних даних і отримується скалярний добуток між ними та фільтром. Під час такого переміщення отримується двовимірна таблиця активаторів, яка дає результат на кожну окрему позицію [26];

– RELU – шар активаційних функцій, розмір шару залишається незмінним від розміру шару, який знаходиться перед ним. Основним завданням такого шару нейронів є згладжування результату, отриманого на минулому кроці, зазвичай

для згладжування використовується активаційна функція, яка залишає лише позитивну частину аргументу, котрий було отримано нейроном [27];

– POOL – шар, який призначено для зменшення кількості нейронів, наприклад $16 \cdot 16 \cdot 12$. Таке зменшення досягається за допомогою використання алгоритмів проріджування даних, основними алгоритмами є розрахунок середнього значення та використання максимального значення;

– FC – вихідний шар нейромережі, завдання якого розрахувати результат роботи класифікатора, кожен із нейронів цього шару підключено до усіх нейронів попереднього.

Таким чином, ConvNet мережі перетворюють значення пікселів вхідного зображення на ваги класифікації. Також не всі шари є параметризованими, наприклад RELU/POOL шари мають фіксовану функцію, яка не має параметрів. У підсумку:

– ConvNet мережа – це простий приклад мережі, що обробляє зображення та видає класифікацію у результат;

– ConvNet складається із декількох визначених типів шарів CONV/FC/RELU/POOL;

– кожен шар приймає тривимірну інформацію та перетворює її у тривимірний вихід;

– параметри не є обов’язковими у кожному із шарів [25].

1.5 Мережа YOLOv4

Через те, що ConvNet використовується в основному для класифікації одного об’єкта на зображенні, на її основі було створено YOLOv3 мережу. Основними перевагами цієї мережі є:

– можливість локалізації класифікатора – тобто класифікація малих зон на зображенні, а не всього зображення повністю;

– менша кількість даних для навчання – архітектура мережі дозволяє проводити навчання із малою кількістю тестових даних на відміну від ConvNet класифікатора, де необхідно мати тисячі або десятки тисяч зображень, що може бути використано в областях, де неможливо отримати велику кількість тестових даних;

– швидка робота готової мережі – навчена мережа може визначати об’єкти на зображенні менше ніж за секунду;

– архітектура мережі дозволяє використовувати її для класифікації об’єктів на відео у реальному часі;

– можливість використання мережі для будь-якої мети завдяки вільній ліцензії;

– можливість створення на базі нейромережі динамічної бібліотеки для подальшого використання у програмах на її основі;

– велика кількість інструментарію, який поставляється разом із нейромережею і спрощує навчання та подальшу роботу з нейромережею.

Архітектура даної мережі базується на архітектурі ConvNet і має назву Darknet-53 [28]. Дана архітектура складається з 53 шарів нейронів, які виконують перетворення над зображенням для того, щоб класифікувати об’єкти на ньому. Також однією із особливостей даної архітектури є метод послідовного розбиття зображення на дрібніші зображення для класифікації зменшених частин зображення одночасно із цілим зображенням. Таке розбиття дає можливість використовувати нейромережу із зображеннями різного розміру та різного рівня деталей із результатами одного навчання без необхідності перенавчання. Отримані класифікатори регіонів складаються для отримання регіону всього об’єкта. Приклад такої системи зі списком шарів і перетворень, які виникають у цих шарах, наведено на рис. 1.8 [29].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Рисунок 1.8 – Архітектура YOLOv4 мережі [29]

На даному прикладі можна бачити збільшення кількості фільтрів від 32 до 1024 в останніх шарах. Зі збільшенням кількості фільтрів зменшується розмір кожного з них.

Такі мережі використовують алгоритм послідовного зменшення області пошуку особливостей до тих пір, поки для даної області не буде визначено класифікатор об’єкта. Такий алгоритм, на відміну від інших, дозволяє зменшити час пошуку об’єктів, якщо вони мають великі розміри, та може використовуватися для багатьох різноманітних прикладів:

– автоматизація засобів пересування;

– сканування земельних об’єктів;

– класифікація клітин організму;

– знаходження пошкоджень або відмінностей від заданого еталону [29].

Також такі мережі можуть використовуватися під час класифікації матеріалів великої щільності завдяки великій точності або у галузях, де інколи

неможливо отримати великі набори тренувальних даних. Також дана мережа може бути використана за необхідності для класифікації рухомих об’єктів за допомогою відеопотоку, але варіант із розпізнаванням об’єктів на фотографіях є переважним, оскільки він дає більш точний результат через можливість точно контролювати положення об’єкта у кадрі, що може попередити отримання хибно позитивних або хибно негативних результатів.

1.6 Бібліотека комп’ютерного зору OpenCV

Для того, щоб працювати з зображеннями для перетворень більш ефективно, можливо використовувати бібліотеку із відкритим похідним кодом під назвою OpenCV (Open Source Computer Vision Library). Ця бібліотека написана на нативній мові програмування C++ та має прошарки сумісності для усіх найбільш популярних мов програмування, таких як Python або Java, існує підтримка системи MATLAB, також у бібліотеку додано підтримку усіх сучасних операційних систем, таких як Linux, MacOS, Windows, Android та iOS, що дозволяє створювати програми, які будуть працювати на усіх платформах без необхідності внесення змін до програмного коду [30]. У програму бібліотеки закладено підтримку сучасних наборів команд процесора, таких як MMX та SSE, і підтримку бібліотеки паралельних розрахунків за допомогою графічних прискорювачів CUDA та відкритого аналогу OpenCL. Також використання такої бібліотеки стає можливим завдяки використанню ліцензії BSD, що дозволяє використовувати програму або її похідні у будь-яких намірах без необхідності розкриття похідного коду у публічному просторі [31].

Бібліотека OpenCV, окрім можливостей комп’ютерного зору, включає у себе велику кількість функцій для перетворення зображень, а також має підтримку усіх найрозповсюдженіших форматів зображення. Також у складі бібліотеки знаходяться можливості роботи із окремими кадрами відеофайлу або потокового відео з мережі Інтернет, що дозволяє використовувати дану мережу із камерами, які віддалені від комп’ютера, де запущена програма [32].

Також бібліотека OpenCV включає у себе можливості машинного навчання, які дозволяють використовувати її для знаходження обличь, об’єктів на зображенні або для класифікації об’єктів на зображенні, але, наприклад, на відміну від нейромережі Yolo, такі можливості представлені у базовому варіанті та мають меншу швидкість роботи, що може бути критичним для її використання як програми для контролю ДП.

Окремо слід визначити вбудовані можливості слідкування за об’єктами у двовимірному та тривимірному просторах, що дозволяє отримувати просторову інформацію щодо місцезнаходження об’єкту дослідження відносно границь кадру або отримати можливість викликати певний код, коли об’єкт досягнув певної мети. Також серед вбудованих можливостей є спеціальні функції, необхідні для отримання окремих деталей зображення або цілих об’єктів за допомогою їх геометричних характеристик.

Бібліотека розвивається зусиллями громадськості та за підтримки великих світових компаній, таких як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda та Toyota, окрім того, що ці компанії використовують дану бібліотеку у великій кількості своїх продуктів, вони також допомагають за допомогою написання коду та у вигляді фінансової підтримки [31].

Також одним із найбільших переваг даної бібліотеки є гарна документація, яка покриває усі стабільні можливості бібліотеки, велика кількість прикладів використання, які розповсюджуються разом із бібліотекою.

1.7 Висновки до розділу 1

У першому розділі проаналізовано літературу за темою контролю ДП. Зокрема, проаналізовано поширені методи контролю, а саме функціональний і внутрішньосхемний, було проаналізовано методи електричного контролю, методи оптичного та методи візуального контролю ДП. За основу досліджуваного методу обрано метод ручного візуального контролю як один із

найбільш універсальних, але попри це, такий, який має ряд недоліків, одним із прикладів яких може слугуватижити людський фактор.

Тому на основі проаналізованої інформації було вирішено створювати комп’ютерну програму для візуального контролю ДП за допомогою нейромереж, таке рішення дозволить залишити гарну універсальність методу без необхідності переналаштування на кожний окремий продукт завдяки схожості алгоритму роботи нейромережі із тим, як працює людське мислення, а також позбавити його недоліків, які можуть походити від людського фактору.

Таким чином, беручи за основу використання нейромереж, проаналізовано літературу за темою, проаналізовано механізми роботи нейромереж і зокрема досліджено сімейство згорткових нейромереж, які можуть бути використані для пошуку та класифікації об’єктів на зображенні.

Як основу для побудови нейромережі обрано нейромережу під назвою YoloV4, яка побудована на основі архітектури DarkNet. Використання даної архітектури дозволяє нейромережі працювати із великою точністю за великої швидкості передачі зображень, або навіть під час використання із відео.

Також досліджено основні можливості бібліотеки комп’ютерного зору OpenCV, досліджено її переваги та недоліки у порівнянні із бібліотекою YoloV4. Описано переваги використання даної бібліотеки як бібліотеки комп’ютерного зору.

2 РОЗРОБКА ОСНОВНИХ ПАРАМЕТРІВ НЕЙРОМЕРЕЖІ

2.1 Перетворення зображень для використання у процесі навчання та роботи нейромережі

Пошук дефектів на ДП потребує її зображення великого розміру, зазвичай ці розміри близькі або більші за квадрат зі стороною у 4096 пікселів із трьома бітами кольору, приклад наведено на рис. 2.1.

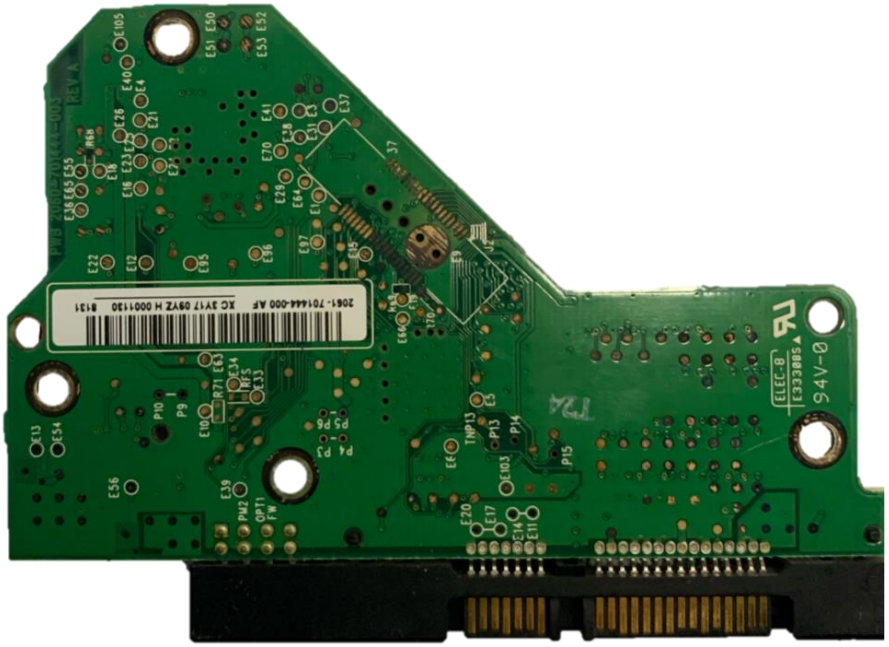


Рисунок 2.1 – Приклад фотографії ДП, розміри 4032 x 3850 пікселі

Але робота з такими зображеннями може потребувати багато обчислювальних ресурсів, тому зазвичай такі зображення переводяться з повнокольорового кольорового простору до градієнтів сірого та розбиваються на декілька зображень меншого розміру, які тестуються окремо, але через використання мережі YOLOv4 необхідність розбиття зображення відпадає.

Для перетворення у градієнти сірого необхідно кожен піксель зображення перетворити за формулою (2.1):

$$C = 0{,}21 \cdot R + 0{,}71 \cdot G + 0{,}07 \cdot B, \quad (2.1)$$

де R – значення червоного кольору у діапазоні 0…255;

G – значення зеленого кольору у діапазоні 0…255;

B – значення синього кольору у діапазоні 0…255.

Така формула необхідна через те, що кольоровий простір RGB створений з урахуванням особливостей роботи фото- та відеотехніки, тому використання формули середнього значення видає неоптимальний результат. Після виконання такого перетворення отримуємо результат, наведений на рис. 2.2.

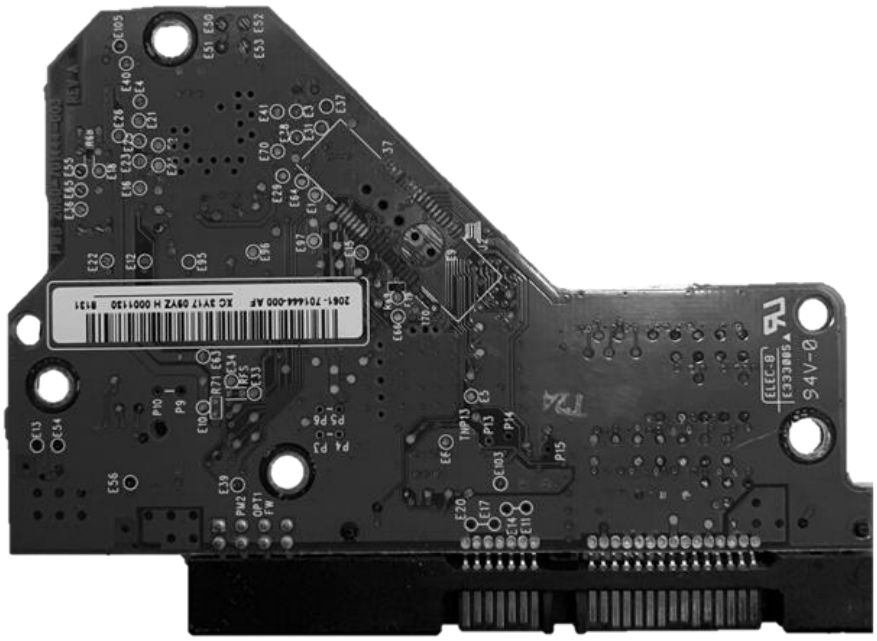


Рисунок 2.2 – Результат перетворення кольору у градації сірого

Також для більшої зручності та для зменшення необхідних ресурсів комп’ютера можливо перевести зображення у бінарний формат за допомогою формули (2.2)

$$C_{out} = \begin{cases} 1, C_{in} < a \\ 0, C_{in} \geq a' \end{cases} \quad (2.2)$$

де a – певний поріг у діапазоні 0…255;

C_{in} – вхідне значення кольору;

C_{out} – вихідне значення кольору у чорно-білому форматі.

Така бінаризація даних дозволяє уникнути похибки, яка може виникати через неспівпадіння освітлення на самих тестових даних або на реальних прикладах, що може викликати хибно негативні або хибно позитивні результати. Хоча й існують декілька різних методів для боротьби із хибними спрацюваннями, але бінаризація є одним із найпростіших варіантів для використання під час контролю ДП.

Після таких перетворень і зменшення зображень до частин отримуємо результат, наведений на рис. 2.3.

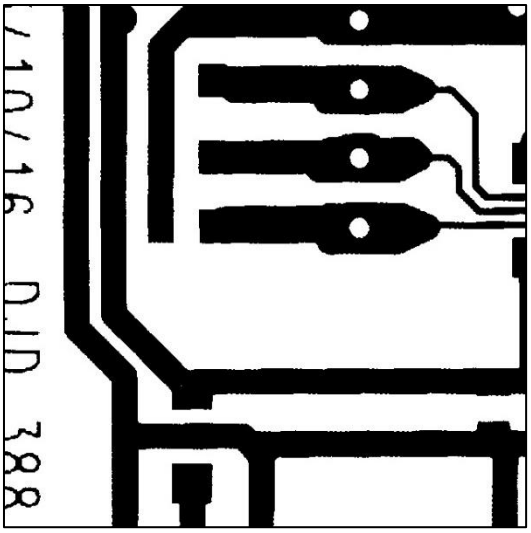


Рисунок 2.3 – Приклад частини зображення після виконання перетворень

2.2 Створення набору даних для навчання та тестування

Для проведення навчання необхідно мати достатній набір даних, який буде покривати усі можливі види дефектів, бажано, щоб кожне зображення зберігалось у всіх можливих варіантах повороту та віддзеркалено за всіма можливими осями. Також для кожного зображення у наборі даних треба скласти опис із зазначенням усіх дефектів, які зображені на ньому. Для таких описів

використовується текстовий файл у форматі .txt, у якому задані координати та тип дефекту, приклад наведено на рис. 2.4.

466 441 493 470 3
454 300 493 396 2
331 248 364 283 4
221 314 253 350 4

Рисунок 2.4 – Приклад файлу з описом дефектів

Дані задані у форматі $x_1 y_1 x_2\, y_2\, d$, де $x_1 y_1$ – це лівий верхній кут прямокутника, $x_2 y_2$ – правий нижній кут прямокутника, d – один із типів дефектів, які наведено у таблиці 2.1.

Таблиця 2.1 – Типи дефектів

Номер дефекту	Тип дефекту
1	Розрив
2	Коротке замикання
3	Недостатня кількість металізації
4	Шорсткість провідникового рисунку
5	Чужорідні вкраплення
6	Дефекти отворів

Для збільшення швидкості навчання та зменшення вірогідності похибки було обрано 1500 зображень із тестовими даними, які являють собою частини сканованих зображень у чорно-білому варіанті, де білий колір відведено на області без мідного шару, чорний колір відведено для областей, покритих мідним шаром.

Кожне зображення представлено у двох форматах – із дефектом і без дефекту, у розмірі $640 \cdot 640$ пікселів, на кожному зображенні знаходиться від 3 до 12 дефектів. Зі всього набору зображень 1000 було відведено для навчання,

500 було відведено для тестових даних [33]. Розподіл тестових даних наведено на рис. 2.5.

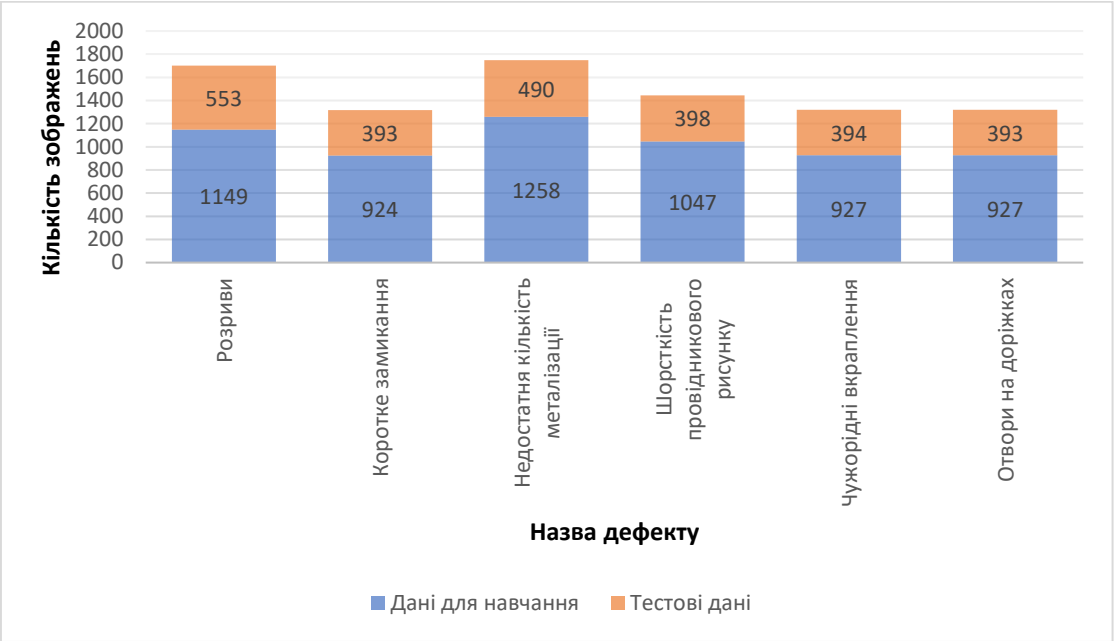


Рисунок 2.5 – Розподіл даних для навчання нейромережі

2.3 Функціональна модель нейрона

У процесі роботи нейрон перетворює вхідні сигнали на вихідний сигнал, який зазвичай лежить у діапазоні від 0 до 1. Таке перетворення виконується завдяки підсумуванню сигналів, які надходять на входи, з урахуванням вагових коефіцієнтів і передачі такого сигналу на подальшу активаційну функцію, яка в свою чергу розраховує вихідний сигнал. Модель нейрона представлено на рис. 2.6.

На цій моделі цифрами позначено:

1 – вектор вхідних параметрів – вектор чисел, які подаються на вхід нейрона, та вектор вагових коефіцієнтів;

2 – суматор – функціональний блок, який підсумовує усі вхідні числа після врахування вагових коефіцієнтів;

3 – активаційна функція – математична функція, що визначає число на виході нейрона;

4 – вихід – з'єднаний з іншими нейронами канал для передачі вихідного значення.

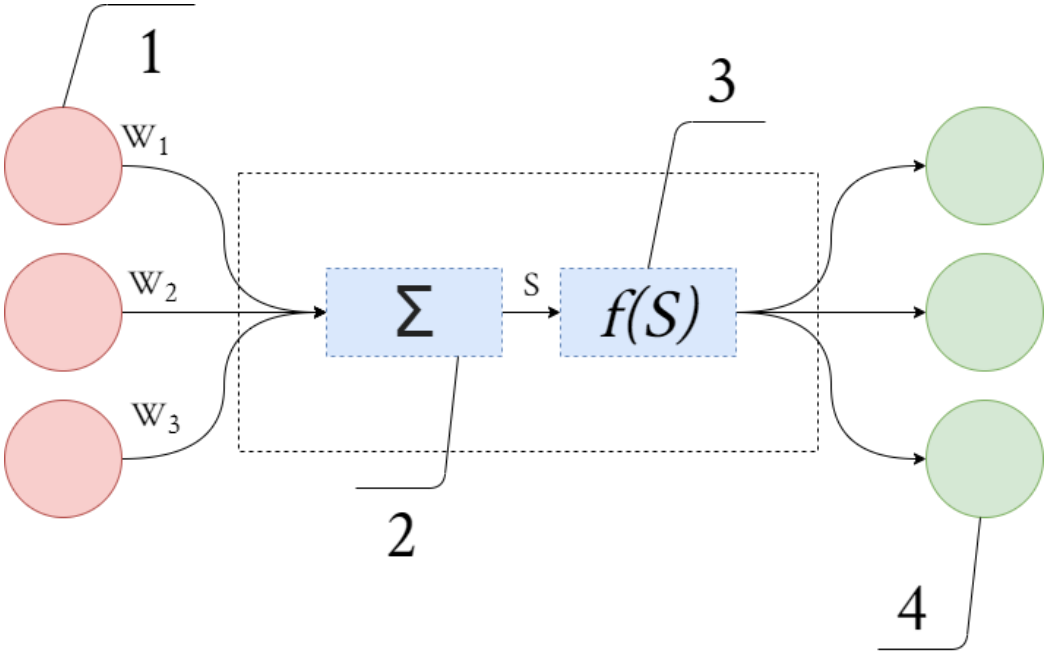


Рисунок 2.6 – Модель нейрона

Тобто нейрон можна представити як декілька функцій (2.3) та (2.4)

$$S = \sum_{i=1}^{n} w_i k_i, \tag{2.3}$$

де n – кількість входів;

w_i – і-й вхід;

k_i – коефіцієнт і-го входу;

$$Y = f(S). \tag{2.4}$$

2.4 Архітектура роботи нейромережі YOLOv4

У даній архітектурі нейромережі виходи розбиваються на три різні сітки, розмір яких являє собою 8 · 8, 16 · 16 та 32 · 32 квадрати. Тобто для зображення розміром 640 · 640 пікселів буде створено три сітки:

– $640 \div 8 = 80$;

– $640 \div 16 = 40$;

– $640 \div 32 = 20$.

Але вихідні значення є не просто числами, а векторами, розмір яких визначається за формулою (2.5).

$$S = B \cdot (5 + C), \tag{2.5}$$

де B – кількість регіонів у комірці сітки;

C – кількість класів, які ми визначаємо.

Число 5 береться з розрахунку того, що для кожного регіону вираховується вірогідність знаходження там об’єкта та 4 координати – x, y та довжина з висотою.

У нашій мережі використовується 6 класів об’єктів і для кожної комірки сітки передбачається 3 регіони, тобто ми можемо розрахувати розмірність вектора для кожної комірки сітки:

$$S = 3 \cdot (5 + 6) = 33.$$

Після отримання результатів необхідно отримати вірогідність того, що у певному регіоні знаходиться певний об’єкт, що розраховується за формулою (2.6):

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}, \tag{2.6}$$

де x – вектор, для якого використовується перетворення;

K – розмірність вхідного вектора.

Після такого перетворення ми отримуємо вектор, у якому кожне значення знаходиться у межах від 0 до 1, і сума усіх значень складає 1.

Для отримання координат регіонів використаємо формули (2.7-2.10):

$$\mathrm{x} = \sigma(\hat{\mathrm{x}}) + \mathrm{c_x}, \tag{2.7}$$
$$\mathrm{y} = \sigma(\hat{\mathrm{y}}) + \mathrm{c_y}, \tag{2.8}$$
$$\mathrm{w} = \mathrm{p_w} e^{\hat{\mathrm{w}}}, \tag{2.9}$$
$$\mathrm{h} = \mathrm{p_h} e^{\hat{\mathrm{h}}} \tag{2.10}$$

де $\hat{x}$, $\hat{y}$, $\hat{w}$, $\hat{h}$ – координати чотирьох кутів регіону;

c_x, c_y – відступ від лівого верхнього кута;

$\sigma(x)$ – функція сигмоїди;

p_w, p_h – координати якорів.

Тобто враховуючи вказану інформацію ми можемо отримати регіони інтересів із процентним значенням можливих дефектів.

Також для навчання нейромережі нам необхідно знати максимальну кількість кроків навчання, тому що за необмеженої кількості кроків можливо отримати перенавчену нейромережу, яка може визначати дефекти на тестових даних майже зі стовідсотковою точністю, але не може визначати дефекти на реальних даних. Також кількість кроків не може бути меншою ніж 6000, тому ми можемо розрахувати кількість кроків за формулою (2.11):

$$n = C \cdot 2000 = 6 \cdot 2000 = 12000, \tag{2.11}$$

де C – кількість класів, які необхідно використати.

Також необхідно визначити мінімальну та максимальну точки, де може знаходитися найкращий результат навчання нейромережі, вони складають 80 % та 90 % від максимальної кількості кроків відповідно, формула (2.12-2.13):

$$n_{min} = n \cdot 0{,}8 = 9600, \quad (2.12)$$
$$n_{max} = n \cdot 0{,}9 = 10800 \quad (2.13)$$

де n – максимальна кількість кроків навчання.

Використовуючи отримані дані ми можемо приступити до створення нейромережі.

2.5 Висновки до розділу 2

У другому розділі було обрано необхідний формат зображень та описано перетворення, які необхідно виконати для збільшення точності нейромережі, а саме перетворення зображення із RGB кольорового спектру до режиму градацій сірого, що дозволяє зменшити розмір зображень через можливість використання 8-бітного зображення замість 24-бітного, що в свою чергу призводить до збільшення швидкості роботи нейромережі. А також застосовуватиметься додаткове перетворення у бінарне зображення за допомогою порогових функцій, що дозволяє позбавитися від впливу нерівномірності освітлення.

Також було обрано датасет, необхідний для навчання та тестування нейромережі, який складається з 1000 навчальних зображень і 500 тестових зображень розміром 640 · 640 пікселів. Датасет складається із частин зображень реальних ДП, які було перетворено у бінарний формат, де білий колір позначає відсутність мідного шару, а чорний колір позначає його наявність. На цих зображеннях за допомогою графічних редакторів було нанесено дефекти. Описано необхідні перетворення формату запису дефектів через те, що нейромережа вимагає певний формат.

Було описано модель нейронів, з яких складається нейромережа, а також наведено опис алгоритму роботи нейромережі у цілому.

Також у даному розділі розраховано необхідні параметри нейромережі, які будуть оптимальними для використання із обраним датасетом, враховуючи кількість класифікаторів дефектів. Серед таких параметрів було розраховано необхідну кількість фільтрів у шарах нейромережі, максимально необхідну кількість ітерацій навчання нейромережі та можливі кількості ітерацій, у яких буде знаходитися найліпший результат.

3 РОЗРОБКА СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ДЕФЕКТІВ НА ЗОБРАЖЕННЯХ ДРУКОВАНИХ ПЛАТ

3.1 Розробка програми для бінаризації зображення

Для того, щоб збільшити точність і швидкість нейромережі, а також позбавитися від проблем, які можуть виникати через нерівномірність освітлення об’єкту, який тестується, було прийнято рішення про створення програми, основним завданням якої буде бінаризація зображення, тобто переведення його у формат, за якого існує усього два кольори – чорний і білий.

Для створення такої програми було обрано мову програмування Python завдяки деяким перевагам, а саме таким, як:

– простота використання;

– велика кількість бібліотек для роботи із зображеннями;

– можливість підключення нейромережі як сторонньої бібліотеки.

Як бібліотеку для обробки зображень було обрано бібліотеку комп’ютерного зору OpenCV, і також для виведення проміжного результату обрано бібліотеку Matplotlib (рис. 3.1).

```
# Імпорт бібліотеки для роботи із зображенням
import cv2 as cv
# Імпорт бібліотеки яка використовується для виведення зображення на екран
from matplotlib import pyplot as plt
```

Рисунок 3.1 – Підключення сторонніх бібліотек

Після підключення усіх необхідних бібліотек у першу чергу необхідно зчитати зображення з будь-якого з доступних джерел, наприклад, жорсткого диску комп’ютера або з мережевого джерела у вигляді посилання на зображення, яке зберігається у мережі Інтернет (рис. 3.2).

Рисунок 3.2 – Похідне зображення друкованої плати [34]

Наступним кроком необхідно перевести зображення з кольорового простору Blue Green Red (BGR) у кольоровий простір Red Green Blue (RGB), такий крок спростить розуміння зображення людиною через те, що у кольоровому просторі BGR кольори не співпадають із реальними.

Наступним кроком є використання розмивання Гауса, що дозволяє позбавитися від артефактів, які було отримано у процесі створення зображення за допомогою сканера або камери. Також таке розмиття дозволяє позбавитися від невеликих дефектів, які не впливають на якість виробу, але можуть ввести нейромережу в оману, наприклад, невеликі частки, які потрапили у матеріал, або невеликі розбіжності відтінків матеріалу, які можуть нашкодити під час бінаризації зображення.

Наступним кроком є перетворення зображення з кольорового до режиму градацій сірого, такий крок дозволить більш ефективно та точно працювати із зображенням на наступних кроках завдяки усуненню кольорів (рис. 3.3).

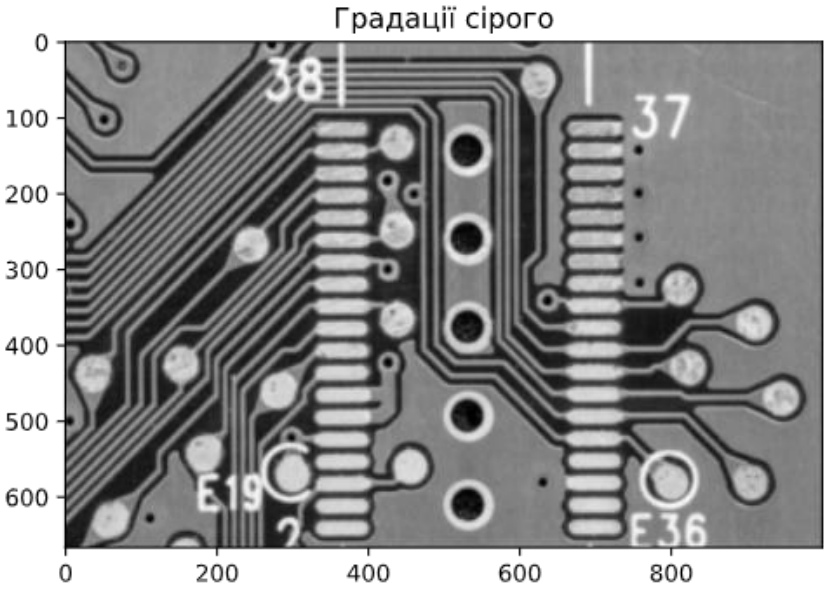


Рисунок 3.3 – Зображення у градаціях сірого

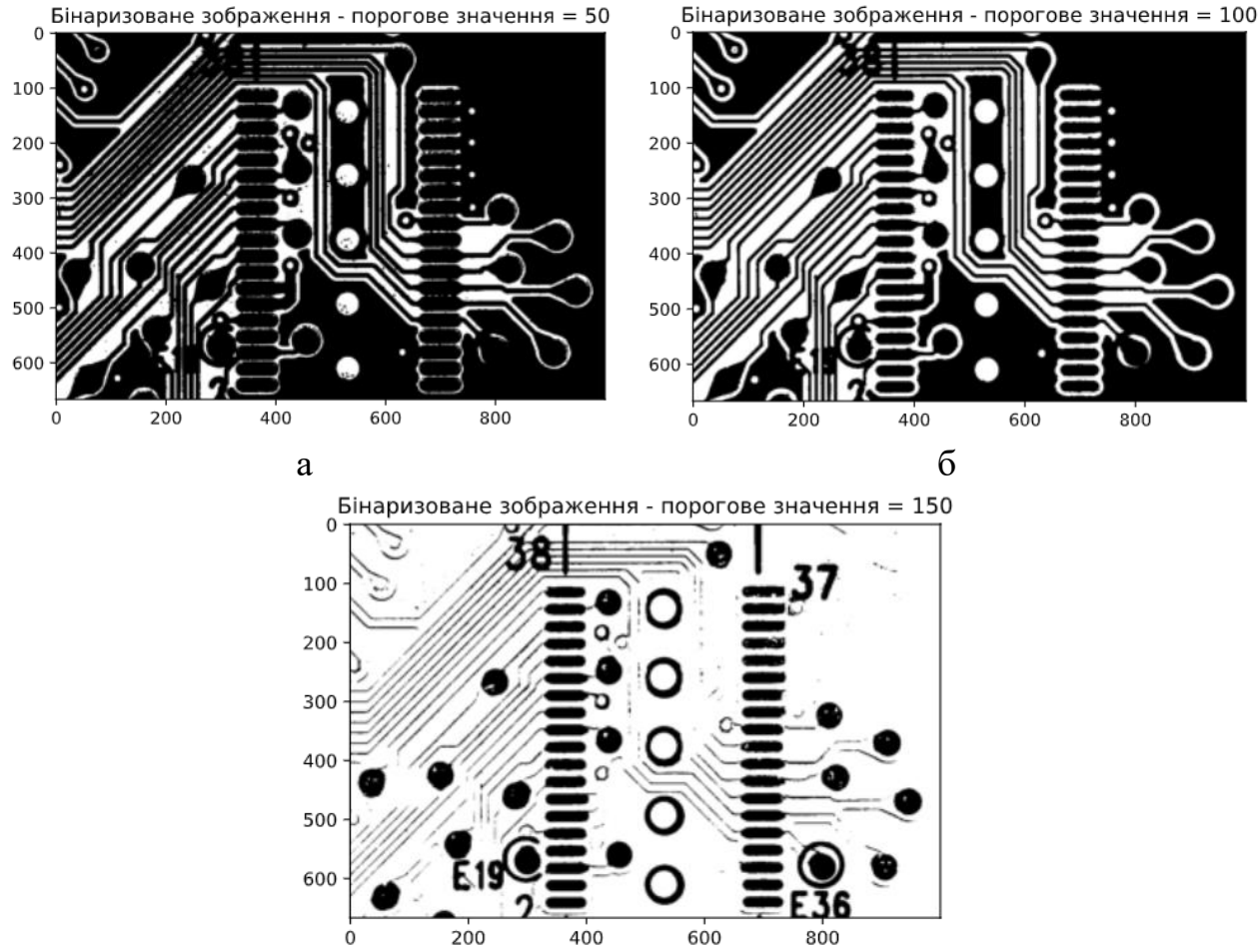
Код, який було використано для таких перетворень, наведено на рис. 3.4.

```
# Зчитування зображеня з файлу
img = cv.imread("pcb.jpg")
# Зміна кольорового простору із BGR у RGB
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
# Застосування розмивання гауса
img = cv.GaussianBlur(img,(5,5),0)
# Перетворення кольорового простору із RGB у градації сірого
grey = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
```

Рисунок 3.4 – Код для перетворень над зображенням

Отримавши після усіх перетворень зображення у режимі градацій сірого ми можемо починати працювати із ним для створення бінаризованого зображення. Для бінаризації найважливішим параметром є порогове значення у діапазоні від 0 до 255, у чорно-білому зображенні таке значення позначає межу, за якою буде розділено зображення на чорний і білий кольори. Таке порогове значення зазвичай знаходиться у ручному режимі та потребує налаштування лише один раз для кожного конкретного випадку. Параметр порогового значення залежить від можливостей камери або сканеру, рівномірності освітлення між

різними тестуваннями та також залежить від кольорів похідного зображення. У даному випадку було протестовано три порогових значення у 50, 100 та 150 і порівняно результати, результати бінаризації наведено на рис. 3.5.



а б

в

а) бінаризоване зображення із пороговим значенням 50;

б) бінаризоване зображення із пороговим значенням 100;

в) бінаризоване зображення із пороговим значенням 150

Рисунок 3.5 – Приклад бінаризованих зображень із різними пороговими значеннями

Таким чином, ми можемо вважати, що найліпше порогове значення у даному випадку дорівнює 100. Код програми для бінаризації зображення наведено на рис. 3.6.

```
ret,th50 = cv.threshold(grey,50,255,cv.THRESH_BINARY_INV)
ret,th100 = cv.threshold(grey,100,255,cv.THRESH_BINARY_INV)
ret,th150 = cv.threshold(grey,150,255,cv.THRESH_BINARY_INV)
```

Рисунок 3.6 – Код програми бінаризації зображень.

3.2 Отримання та адаптація початкового коду

Завдяки тому, що було прийняте рішення будувати нейромережу на основі готової архітектури Darknet, необхідно отримати початковий код програми. Архітектура Darknet побудована на основі мови програмування C та опублікована на сайті GitHub, який являє себою веб-інтерфейс для системи контролю версій (СКВ) під назвою git. Для роботи з даною СКВ існує інтерфейс командного рядка (CLI) під назвою git.

Також для простоти та швидкості роботи було обрано операційну систему GNU/Linux на базі дистрибутиву Ubuntu 20.04.1 LTS (рис. 3.7).

```
root@DESKTOP-1PUTVIN:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.1 LTS
Release:        20.04
Codename:       focal
```

Рисунок 3.7 – Інформація про дистрибутив GNU/Linux

Використовуючи CLI git клонуємо початковий код програми з сайту GitHub [35] на комп’ютер для подальшого використання (рис. 3.8).

```
root@DESKTOP-1PUTVIN:~# git clone https://github.com/AlexeyAB/darknet
Cloning into 'darknet'...
remote: Enumerating objects: 14370, done.
remote: Total 14370 (delta 0), reused 0 (delta 0), pack-reused 14370
Receiving objects: 100% (14370/14370), 13.09 MiB | 1.96 MiB/s, done.
Resolving deltas: 100% (9771/9771), done.
Updating files: 100% (2021/2021), done.
```

Рисунок 3.8 – Клонування похідного коду програми Darknet

Таке клонування дозволяє отримати копію похідного коду програми з усіма останніми змінами, які було написано автором програми. Використання похідного коду програми є можливим через те, що автор оригіналу обрав дозвільну ліцензію під назвою «YOLO License» для розповсюдження похідного коду програми. Дана ліцензія дозволяє використання програми або будь-яких її частин для будь-якої мети.

Після отримання похідного коду програми до файлової системи комп'ютера ми можемо починати працювати із отриманими даними та створювати необхідну нейромережу для розпізнавання дефектів друкованої плати. Також окрім похідного коду програмного забезпечення у репозиторії з програмою зберігаються:

– сторонні бібліотеки, необхідні для роботи нейромережі;

– Cmake файли для компіляції програми як на операційній системі GNU/Linux, так і на операційній системі Windows;

– Make файл для компіляції на операційній системі GNU/Linux;

– тестові дані для перевірки скомпільованої програми;

– приклади конфігурацій нейромережі під різноманітні датасети та нейромережі на основі архітектури Darknet;

– набір утиліт, які можуть бути використані з нейромережею;

– заздалегідь натреновані файли вагових коефіцієнтів для різних типів нейромереж, які використовуються для початку навчання;

– приклади використання динамічної бібліотеки нейромережі у мовах програмування C та C++;

– датасет для створення програми розпізнавання тексту на зображенні.

Файлова структура даних, які було отримано із системи контролю версій, наведена на рис. 3.10.

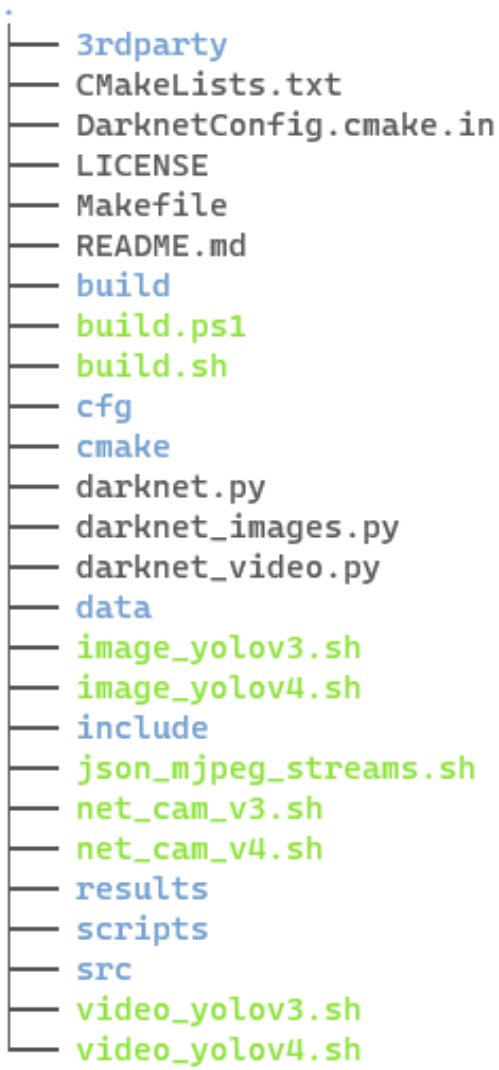


Рисунок 3.10 – Файлова структура нейромережі

Через те, що було обрано операційну систему GNU/Linux, найліпшим варіантом для компіляції програми буде використання утиліти з пакету програм GNU під назвою make. Конфігурація параметрів компіляції, а також усі необхідні кроки виконання компіляції описані у файлі «Makefile». Параметри компіляції зазначаються за допомогою булевих значень або бінарних прапорів із значенням 0 або 1, які розташовані на початку файлу «Makefile». Основні параметри системи компіляції make наведено у таблиці 3.1.

Таблиця 3.1 – Параметри системи компіляції make

Назва параметру	Значення за умовчанням	Опис параметру
1	2	3
GPU	0	Використовувати прискорення за допомогою графічної карти із використанням бібліотеки для виконання математичних розрахунків на графічній карті CUDA. Працює тільки на деяких сучасних відеокартах компанії Nvidia.
CUDNN	0	Використовувати прискорення за допомогою апаратного прискорення штучного інтелекту із використання бібліотеки cuDNN. Працює тільки на сучасних відеокартах компанії Nvidia.
CUDNN_HALF	0	Використовувати Tensor ядра, які додані у професійних відеокартах компанії Nvidia Titan або серіях відеокарт RTX 2XXX та RTX 3XXX. Дає прискорення приблизно у два рази у порівнянні із іншими методами прискорення.
OPENCV	0	Компіляція із підтримкою бібліотеки OpenCV 4.x/3.x/2.4.x для можливості використання нейромережі для класифікації об’єктів на відео, потоку відео з камер або використання потокового відео з мережі Інтернет.
DEBUG	0	Компіляція із підтримкою відлагоджувальної інформації. Не рекомендується використовувати у фінальній версії програми через великий вплив на швидкість роботи.
OPENMP	0	Компіляція із підтримкою багатоядерних систем, що дозволяє прискорити роботу на таких системах, якщо не використовується прискорення за допомогою графічної карти.

Продовження таблиці 3.1

1	2	3
LIBSO	0	Додаткова компіляція програми нейромережі у форматі динамічної бібліотеки, що дозволяє підключати її та використовувати у інших програмах або мовах програмування.
ZED_CAMERA	0	Компіляція програми із підтримкою ZED-3D камери.
AVX	0	Використання додаткового набору інструкцій процесора AVX. Дозволяє прискорити роботу нейромережі на процесорі, але не всі процесори підтримують такий набір команд.
USE_CPP	0	Використовувати для компіляції набор утиліт cpp замість набору утиліт g++.

Враховуючи особливості обчислювальної техніки, яку було використано для навчання, було виставлено наступні параметри компіляції:

– OPENCV=1 – параметр, який вмикає можливості використання разом із бібліотекою OPENCV;

– OPENMP=1 – параметр, який вмикає підтримку багатоядерних систем. Ввімкнуто через те, що комп’ютер, на якому було виконано навчання нейромережі, має багатоядерний процесор;

– LIBSO=1 – параметр, який визначає необхідність додаткової компіляції програми як динамічної бібліотеки у форматі «so» або «dll» у залежності від операційної системи.

Компіляція програми виконується за допомогою CLI make, яка поставляється у наборі програм «build-essentials» зі стандартного репозиторію Ubuntu. Після виконання програми ми отримуємо два файли «darknet» та «libdarknet.so», інформація про які наведена на рис. 3.11.

```
darknet:       ELF 64-bit LSB shared object, x86-64, versio
n 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux
-x86-64.so.2, BuildID[sha1]=746bd02a643c58100f86cbb88c5e9df
f84acd4a0, for GNU/Linux 3.2.0, with debug_info, not stripp
ed
libdarknet.so: ELF 64-bit LSB shared object, x86-64, versio
n 1 (GNU/Linux), dynamically linked, BuildID[sha1]=3cff990b
e79b0f26012a4f09dd1bb07aac9dca4b, with debug_info, not stri
pped
```

Рисунок 3.11 – Інформація про файли, отримані у процесі компіляції

3.3 Підготовка датасету для навчання нейромережі

Для навчання нейромережі було обрано датасет, який складається з 1500 парних зображень, котрі складають тільки маленьку частку від цілої ДП. Розмір кожного з зображень складає 640 на 640 пікселів. Також до кожного зображення прикладається текстовий документ з описом дефектів. Але формат опису дефектів у датасеті не співпадає з форматом опису, необхідним для навчання нейромережі Darknet. Порівняння форматів наведено на рис. 3.12.

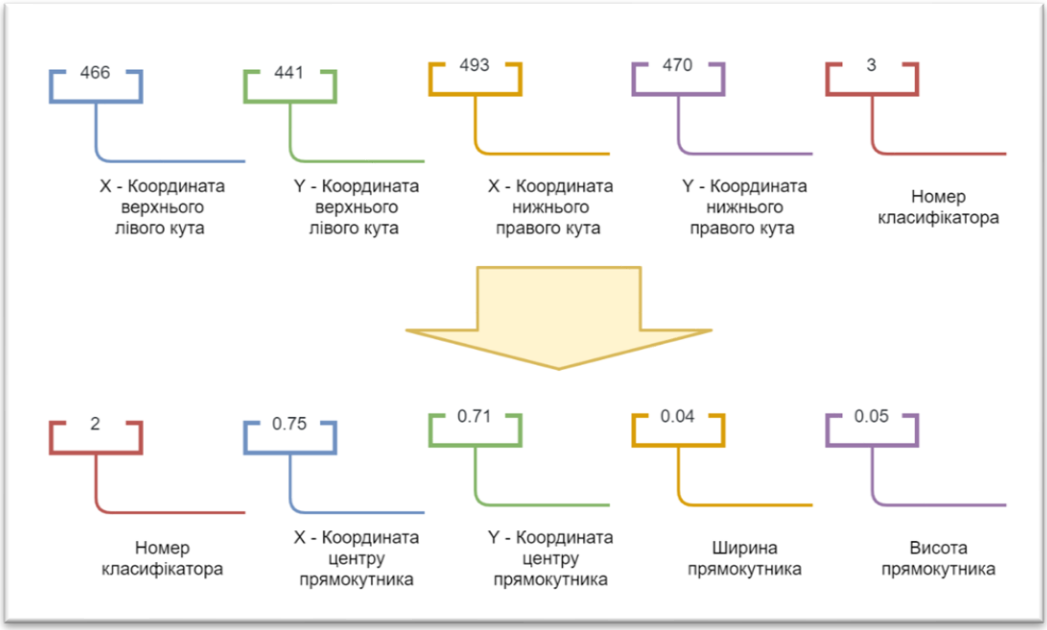


Рисунок 3.12 – Порівняння форматів опису дефектів на датасеті

Основними їх відмінностями є:

– різниця у нумеруванні класифікаторів: у датасеті номер класифікатора починається з 1, нейромережа очікує, що початковий номер класифікатора буде починатися з 0;

– різниця у задаванні координат регіонів: у датасеті координати регіонів дефектів задаються за допомогою абсолютних координат у пікселях, починаючи від верхнього лівого кута та формуються за допомогою зазначення верхнього лівого та нижнього правого кутів регіону, нейромережа очікує координати у процентному форматі від розмірів зображення та формується з процентної координати центра та проценту висоти та ширини регіону.

Таким чином, використання такого датасету напряму є неможливим і необхідне перетворення текстових документів з описом дефектів. Для такого перетворення можна використати формули (3.1-3.4):

$$x_{center} = \left(\frac{x_1 + x_2}{2}\right)/640, \tag{3.1}$$

$$y_{center} = \left(\frac{y_1 + y_2}{2}\right)/640 \tag{3.2}$$

де x_{center}, y_{center} – процентні координати центрів регіонів дефектів;

x_1, y_1 – координати лівого верхнього кута регіону;

x_2, y_2 – координати правого нижнього кута регіону.

$$w = \frac{x_2 - x_1}{640}, \tag{3.3}$$

$$h = \frac{y_2 - y_1}{640}, \tag{3.4}$$

де w, h – ширина та висота регіону з дефектом віпдопідно;

x_1, y_1 – координати лівого верхнього кута регіону;

x_2, y_2 – координати правого нижнього кута регіону.

Таким чином, для підготовки датасету у першу чергу необхідно отримати похідний код за допомогою CLI git з сайту GitHub [33]. Отримавши похідний код нам необхідно створити програму для його перетворення у формат, який підходить для роботи у нейромережі deepnet. Для створення такої програми було обрано мову програмування Python, код програми наведено на рис. 3.13.

```
1 from shutil import copyfile
2
3 for infile, outdir in [("PCBData/trainval.txt","out"),
  ("PCBData/test.txt","out_test")]:
4   with open(infile,"r") as f:
5     with open(f"{outdir}/train.txt", "w+") as out:
6       for line in f:
7         data = line.rstrip("\n").split(" ")
8         with open(f"PCBData/{data[1]}", 'r') as txt:
9           with open(f"{outdir}/{data[1].split('/')
  [2]}", "w+") as out:
10            for l in txt:
11              d = l.split(" ")
12              out.write(f"{int(d[4])-1} {((int(d[2]) +
  int(d[0]))/2)/640} {((int(d[3]) + int(d[1]))/2)/640}
  {((int(d[2]) - int(d[0])))/640} {((int(d[3]) -
  int(d[1])))/640}\n")
13
    copyfile(f"PCBData/{data[0].replace('.jpg','_test.jpg
  ')}",f"{outdir}/{data[0].split('/')[2]}")
14        out.write(f"{data[0].split('/')[2]}\n")
```

Рисунок 3.13 – Лістинг коду програми перетворення датасету

Після виконання такої програми у кореневій директорії датасету ми отримуємо дві нові директорії:

– директорія out із даними, готовими для навчання неромережі;

– директорія out_test із даними, готовими для тестування нейромережі.

3.4 Тренування нейромережі

Отримавши необхідний датасет, який підготовлено до використання у нейромережі, ми можемо починати тренування нейромережі. У першу чергу необхідно обрати базу, на основі якої буде виконуватися навчання, доступні файли налаштувань, які можна використовувати як базові, наведено на рис. 3.14.

```
yolov2-tiny-voc.cfg
yolov2-tiny.cfg
yolov2-voc.cfg
yolov2.cfg
yolov3-openimages.cfg
yolov3-spp.cfg
yolov3-tiny-prn.cfg
yolov3-tiny.cfg
yolov3-tiny_3l.cfg
yolov3-tiny_obj.cfg
yolov3-tiny_occlusion_track.cfg
yolov3-tiny_xnor.cfg
yolov3-voc.cfg
yolov3-voc.yolov3-giou-40.cfg
yolov3.cfg
yolov3.coco-giou-12.cfg
yolov3_5l.cfg
yolov4-custom.cfg
yolov4-tiny-3l.cfg
yolov4-tiny-custom.cfg
yolov4-tiny.cfg
yolov4-tiny_contrastive.cfg
yolov4.cfg
```

Рисунок 3.14 – Базові файли конфігурацій для нейромережі

Дані файли відрізняються як і поколінням нейромережі – останнє покоління під номером 4, так і тим, що ця нейромережа є повноцінною (yolov4.cfg, yolov4-custom.cfg) або зменшеною (yolov4-tiny.cfg). Для найбільшої точності отриманих результатів, що є найважливішим показником для контролю ДП, найліпше використовувати нейромережу останнього покоління (4) та з повноцінною кількістю нейронів. Тобто за основу було обрано файл налаштувань «yolov4-custom.cfg» і на його основі було створено файл під назвою «pcb.cfg».

Наступним кроком є завантаження заздалегідь натренованих вагових коефіцієнтів для прихованих шарів, які знаходяться всередині нейромережі. Такий крок дозволяє значно зменшити час, необхідний на навчання нейромережі, порівняно із іншими підходами, наприклад генерацією випадкових коефіцієнтів. Тому було обрано заздалегідь натреновані коефіцієнти для 4 повного покоління нейромережі.

Наступним кроком є адаптація файлу налаштувань під параметри класифікаторів, необхідних для виконання завдання. Було виконано наступні кроки:

– змінено кількість зображень, які одночасно використовуються під час тренування: стандартне значення – 1, змінене значення – 64;

– змінено кількість регіонів, на які буде розбито тренувальне зображення: стандартне значення – 1, змінене значення – 16;

– змінено максимальну кількість кроків тестування за формулою (2.9): стандартне значення – 500200, змінене значення – 12000;

– змінено максимальні та мінімальні точки кроків тестування, де може знаходитися найліпший результат за формулою (2.10): стандартні значення – 400000, 450000, змінені значення – 9600, 10800;

– змінено розмір мережі на число, кратне 32: стандартне значення – width=416 height=416, змінене значення – width=640 height=640;

– у кожному з блоків налаштувань «[yolo]» змінено кількість класів на необхідну кількість – 6;

– у кожному з блоків налаштувань «[yolo]» змінено кількість фільтрів на необхідну кількість, отриману за формулою (2.6) – 33;

– додано параметр random=1, який збільшує і точність готової нейромережі, і час навчання. Ефект збільшення точності досягається завдяки випадковій зміні масштабу зображень від похідних;

– змінено параметри layers на 23 та stride на 4, які дозволяють класифікувати об’єкти невеликого розміру, але цей параметр збільшує час, необхідний на навчання.

Наступним кроком є створення лістингу класифікаторів у вигляді файлу під назвою «pcb.names», у якому записані усі класифікатори, кожен знаходиться на новому рядку (рис. 3.15). Назви класифікаторів мають бути написані латиницею.

```
1  open
2  short
3  mousebite
4  spur
5  copper
6  pin-hole
```

Рисунок 3.15 – Файл із лістингом класифікаторів

Далі необхідно створити файл конфігурацій, який вказує на місцезнаходження усіх необхідних файлів для навчання нейромережі, значення усіх змінних наведено у таблиці 3.2.

Таблиця 3.2 – Змінні файлу конфігурації

Назва змінної	Опис змінної
classes	Кількість класифікаторів
train	Місцезнаходження текстового файлу з даними для навчання
valid	Місцезнаходження текстового файлу з даними для тестування
names	Місцезнаходження текстового файлу зі списком класифікаторів
backup	Місцезнаходження директорії для зберігання резервних копій вагових коефіцієнтів нейромережі

Таким чином ми можемо створити необхідний файл, підставивши необхідні значення (рис. 3.16).

```
classes = 6
train  = data/train.txt
valid  = data/test.txt
names = data/pcb.names
backup = backup/
```

Рисунок 3.16 – Файл конфігурацій місць знаходження

Записавши усі необхідні параметри нейромережі, ми можемо приступати до навчання.

Процес навчання запускається за допомогою передачі параметрів, які вказують на місцезнаходження усіх файлів конфігурації та початкового файлу з коефіцієнтами відносно до місцезнаходження скомпільованої програми або у вигляді абсолютного шляху. Приклад запуску програми наведено на рис. 3.17.

```
./darknet detector train data/pcb.data cfg/pcb.cfg
 cfg/yolov4.conv.137 -dont_show
```

Рисунок 3.17 – Команда запуску нейромережі для навчання

Після запуску необхідно почекати до тих пір, доки у даних, які виводяться до консолі користувача, середня похибка перестане змінюватися зовсім або зміни будуть незначними. Коли похибка перестала змінюватися, ми зупиняємо програму навчання та починаємо перевіряти резервні копії вагових коефіцієнтів, які робилися кожні 100 ітерацій.

Така перевірка необхідна, аби не виникало проблеми перенавчання, коли нейромережа майже ідеально працює на навчальних даних, але у процесі роботи із реальними даними видає невірні результати. Тому необхідно знайти точку ранньої зупинки (рис. 3.18).

3.5 Використання нейромережі у мові програмування Python

Після того, як було отримано вагові коефіцієнти нейромережі, і через те, що нейромережу було скомпільовано із підтримкою динамічної бібліотеки, ми можемо використати отримані дані у мові програмування Python.

У зв’язку з тим, що динамічна бібліотека нейромережі написана на мові програмування C, необхідно створити прошарок сумісності із використанням вбудованої бібліотеки CDLL.

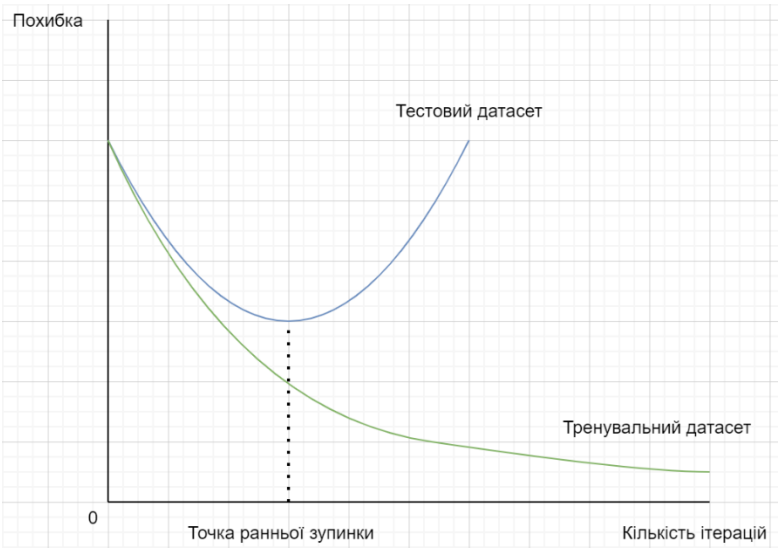


Рисунок 3.18 – Графік похибки навчання із точкою ранньої зупинки

Бібліотека CDLL дозволяє використовувати код, який було написано у мовах програмування C та C++ у похідному коді Python програм, але у даної технології є недоліки у вигляді відсутності розумних підказок у інтегрованих середовищах розробника та невеликої швидкості роботи на відміну від нативного Python коду, що зменшує переваги від використання мов програмування C та C++.

По-перше необхідно створити класи у коді Python, у яких за допомогою бібліотеки ctypes зазначено таку саму структуру, як і у динамічній бібліотеці, список класів наведено на рис. 3.19. Приклад одного із класів наведено на рис. 3.20.

```
class BOX(Structure):…

class DETECTION(Structure):…

class DETNUMPAIR(Structure):…

class IMAGE(Structure):…

class METADATA(Structure):…
```

Рисунок 3.19 – Список необхідних класів у мові програмування Python

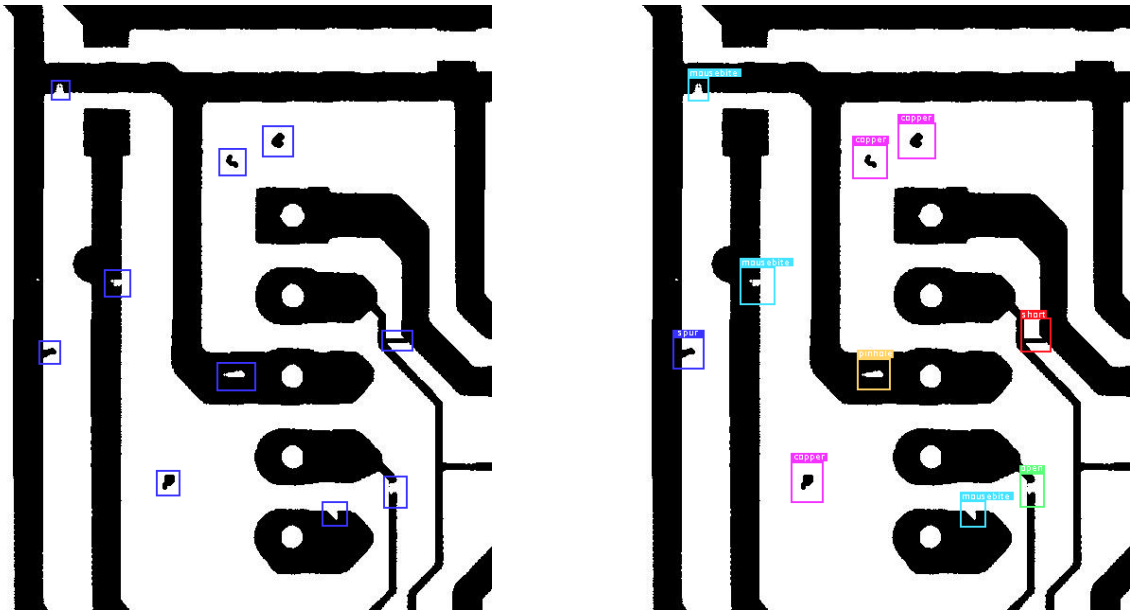
```
class DETECTION(Structure):
    _fields_ = [("bbox", BOX),
                ("classes", c_int),
                ("prob", POINTER(c_float)),
                ("mask", POINTER(c_float)),
                ("objectness", c_float),
                ("sort_class", c_int),
                ("uc", POINTER(c_float)),
                ("points", c_int),
                ("embeddings", POINTER(c_float)),
                ("embedding_size", c_int),
                ("sim", c_float),
                ("track_id", c_int)]
```

Рисунок 3.20 – Приклад класу DETECTION

Наступним кроком є створення усіх необхідних функцій у мові програмування Python і зв’язування їх із функціями з динамічної бібліотеки, що дозволить використовувати їх за необхідністю.

Також необхідно створити деякі допоміжні функції, наприклад, зчитування нейромережі у оперативну пам’ять за допомогою файлів конфігурації та вагових коефіцієнтів, або методи, які дозволяють знаходити або малювати знайдені області. Для малювання об’єктів на зображенні використовується можливості роботи з зображенням від бібліотеки комп’ютерного зору OpenCV. Колір для кожного з типів класифікаторів обирається випадково у кольоровому просторі RGB після кожного запуску бібліотеки.

Приклад результату малювання областей, знайдених нейромережею, у порівнянні із вручну нанесеними областями у графічному редакторі наведено на рис. 3.21.



а б

а) області нанесені відповідно до датасету;

б) області нанесені нейромережею

Рисунок 3.21 – Приклад малювання областей нейромережею

Таким чином, використання такої самописної бібліотеки дозволяє використовувати нейромережу у мові програмування Python, що розширює можливості її використання для розробника.

Для інтеграції нейромережі у код достатньо отримати екземпляр нейромережі за допомогою функції load_network, яка отримує усі файли конфігурацій, і за допомогою функції detect_image отримати регіони, у яких знаходяться класифікатори, передавши об’єкт нейромережі, імена класів і зображення як параметри.

Приклад простої програми, яка знаходить об’єкти на зображенні, наведено на рис. 3.22.

```
# Підключення усіх необхідних бібліотек
from network import load_network, detect_image
import cv2 as cv
# Зчитування зображення
img = cv.imread("pcb.png")
# Створення пустого масиву з іменами класів
class_names = []
# Зчитування імен класів із файла
with open("cfg/pcb.names") as f:
    for line in f:
        class_names.append(line)
# Завантажування нейромережі у програму
network = load_network("cfg/pcb.cfg", "cfg/pcb.data", "pcb.
weight")
# Знаходження об'єктів на зображені
detections = detect_image(network, class_names, img)
# Виведення знайдених об'єктів у текстовому вигляді
print("\nЗнайдені дефекти:")
for label, confidence, bbox in detections:
    x, y, w, h = bbox
    print("{}: {}%    (X координата лівого вернього кута: {:.
    0f}   Y координата правого верхнього кута:  {:.0f}
    Ширина:   {:.0f}   Висота:  {:.0f})".format(label,
    confidence, x, y, w, h))
```

Рисунок 3.22 – Приклад простої програми для знаходження об’єктів на зображенні за допомогою нейромережі

3.6 Висновки до розділу 3

У третьому розділі розроблено програму, яка використовується для перетворень над зображеннями, такими як переведення зображення до кольорового простору градацій сірого, що дозволяє позбавитися від проблем, які виникають від наявності кольору на отриманому зображенні іта спрощують подальшу роботу із зображеннями, а також бінаризація зображення, що дозволяє уникати проблем, які можуть виникати від нерівномірності освітлення та сприяє пришвидшенню роботи нейромережі завдяки зменшенню розмірності вхідної матриці до бінарного значення на кожен піксель зображення.

Також розпочато розробку нейромережі для контролю ДП на виробництві. Через те, що за основу було обрано нейромережу YOLOv4, її похідний код або точніше похідний код її архітектури під назвою DarkNet було отримано з мережі Інтернет, зокрема з системи контролю версій GitHub.

Код нейромережі було модифіковано і включено до нього необхідні функції, що дозволяють оптимально використовувати її на доступному апаратному забезпечені. А саме було включено підтримку багатопотокових процесорів, підтримку можливостей бібліотеки OpenCV і можливість створення окремої динамічної бібліотеки, що дозволяє використовувати нейромережу у інших програмах, написаних на мовах програмування C/C++.

Також було розроблено програму, необхідну для перетворення датасету у формат, який буде зрозумілим для нейромережі на відміну від формату, який поставляється разом із обраним датасетом, як мову програмування для такої програми обрано Python через її простоту використання, можливість використання у декількох операційних системах і швидкість написання невеликих програм.

Окрім того, після усіх виконаних перетворень і отримання готової програми нейромережі було створено необхідні файли конфігурацій з урахуванням розрахованих раніше значень та особливостей датасету. Також у налаштуваннях було обрано опції, які збільшують точність готової нейромережі, але потребують більше часу для навчання.

Після створення усіх необхідних файлів було запущено нейромережу у режим навчання із подальшою її зупинкою після того, як показники точності перестали змінюватися, це потрібно для того, щоб запобігти проблем перенавчання. Окрім того, для знаходження можливої точки ранньої зупинки було проведено тестування певної кількості резервних копій вагових коефіцієнтів, які було створено у ході виконання програми.

Також було створено файл Python-бібліотеки із усіма базовими функціями, необхідними для роботи з нейромережею, такими як завантаження нейромережі у пам’ять комп’ютера або знаходження об’єктів на зображенні. Така бібліотека працює як прошарок сумісності між мовами програмування C та Python і основана на технології CDLL, що дозволяє використовувати нейромережу у Python-коді. Також наведено приклад простої програми, яка виводить результат пошуку дефектів ДП на екран.

4 ПЕРЕВІРКА РЕЗУЛЬТАТІВ РОБОТИ НЕЙРОМЕРЕЖІ

4.1 Знаходження точки ранньої зупинки

Для того, щоб запобігти перенавчанню, ми можемо за допомогою тестового датасету знайти вагові коефіцієнти, у яких найменше значення похибки. Похибкою у даному випадку можна вважати відхилення знайденого регіону від регіону, який був розмічений вручну (рис. 4.1).

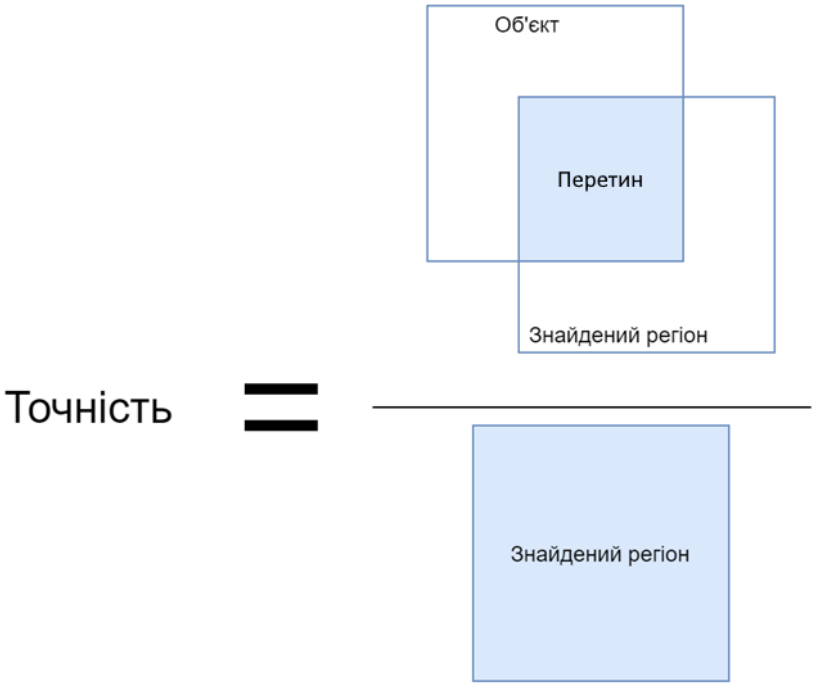


Рисунок 4.1 – Розрахунок точності нейромережі

Таким чином, використовуючи вбудовані можливості нейромережі DarkNet, а також знаючи те, що зміна похибки закінчилася на 9105 ітерації, ми можемо запустити перевірку резервних копій. Для більшої впевненості візьмемо за нижню границю 7100 ітерацію та побудуємо графік точності кожної з резервних копій (рис. 4.2).

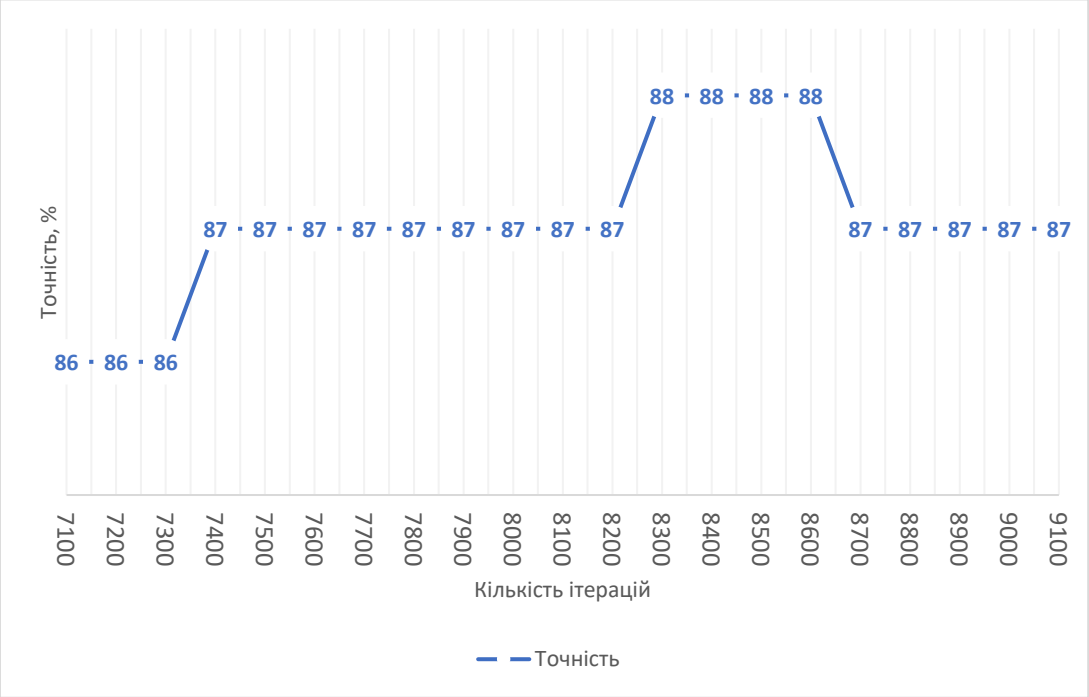


Рисунок 4.2 – Графік точності навченого датасету

За графіком ми можемо бачити, що найліпший результат на тестових даних знаходиться у діапазоні від 8300 ітерацій до 8600 включно, тобто таким чином ми можемо вважати, що найкращій результат видаватиме мережа, яка була навчена за допомогою 8500 ітерацій, і подальша робота буде проводитися із даними ваговими коефіцієнтами.

4.2 Тестування швидкості роботи нейромережі

Після того, як нейромережа була навчена, а також було обрано вагові коефіцієнти, які дають найліпший результат, ми можемо приступати до тестування нейромережі на працездатність і на можливість використання її для контролю ДП.

По-перше, необхідно перевірити швидкість роботи нейромережі, час, затрачений на обробку кожного з зображень, має бути якомога меншим, бажано аби він був меншим за 1 секунду. Для перевірки швидкості роботи обрано 10 тестових зображень, які було перевірено за допомогою нейромережі, кожне зображення було перевірено 4 рази і результат занесено у таблицю 4.1.

Таблиця 4.1 – Швидкість перевірки зображень

Номер зображення	1 ітерація, мс	2 ітерація, мс	3 ітерація, мс	4 ітерація, мс	Середній час, мс
1	71	60	77	111	79
2	86	84	94	85	87
3	104	71	85	113	93
4	117	84	66	62	82
5	113	62	87	80	85
6	60	83	92	67	75
7	102	70	65	79	79
8	96	95	91	111	98
9	88	106	103	106	100
10	115	71	106	69	90

За отриманими результатами побудувано графік середнього часу обробки зображення (рис. 4.3).

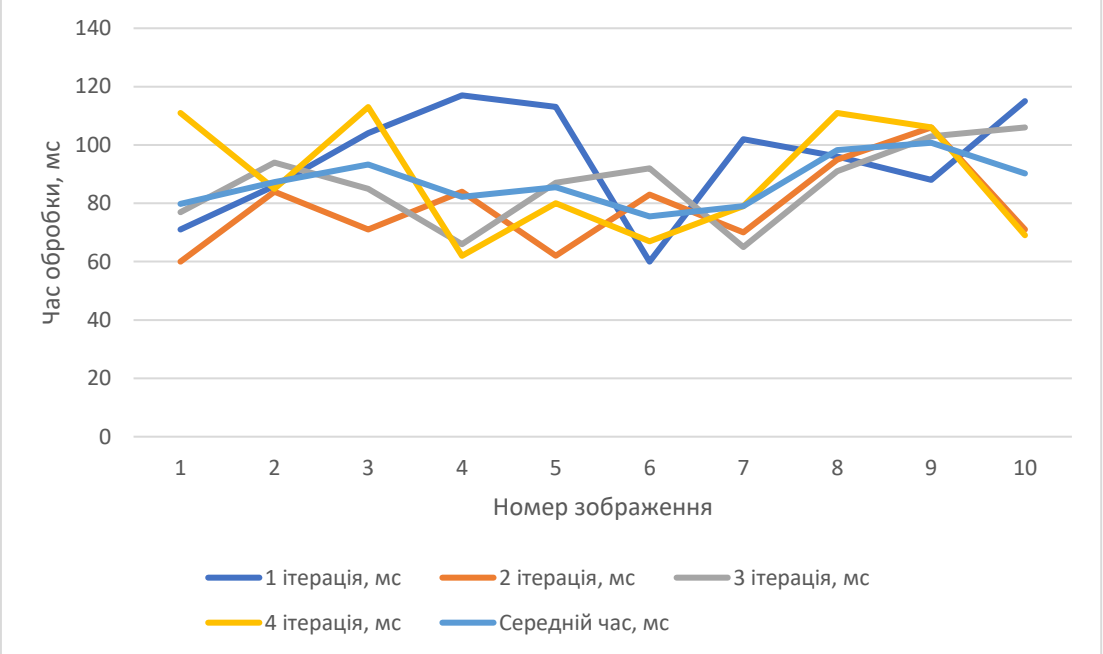


Рисунок 4.3 – Графік часу обробки зображення

Таким чином, швидкість обробки зображень складає від 60 мс до 120 мс, що дозволяє обробляти від 10 до 15 зображень на секунду, що є доволі гарним результатом і підходить для використання у процесі контролю ДП на виробництві у режимі реального часу.

Також необхідно перевірити вплив розміру зображення на швидкість обробки, тому що дана нейромережа повинна мати можливість використовуватися із зображеннями з великою роздільною здатністю, зроблених цифровою камерою, або із зображеннями, отриманими за допомогою сканування. Для такої перевірки на основі існуючих зображень було створено 10 зображень із роздільною здатністю у 1280 · 1280 пікселів і перевірено за тим же методом, що і зображення меншого розміру, результати перевірки наведено у таблиці 4.2.

Таблиця 4.2 – Швидкість перевірки зображень більшого розміру

Номер зображення	1 ітерація, мс	2 ітерація, мс	3 ітерація, мс	4 ітерація, мс	Середній час, мс
1	107	93	137	126	115
2	81	114	105	147	111
3	125	127	82	153	121
4	130	108	114	141	123
5	87	108	91	90	94
6	105	157	147	90	124
7	88	106	120	133	111
8	130	101	125	138	123
9	136	97	127	112	118
10	157	153	120	141	143

Таким чином, час обробки збільшеного зображення знаходиться у діапазоні від 94 мс до 143 мс і дозволяє обробляти зображення із швидкістю 7–10 кадрів на секунду.

За отриманими результатами побудовано графіки порівняння результатів швидкості тестування оригінального зображення із зображенням збільшеного розміру, як параметр для порівняння буде використано середній час знаходження дефектів на зображені (рис. 4.4).

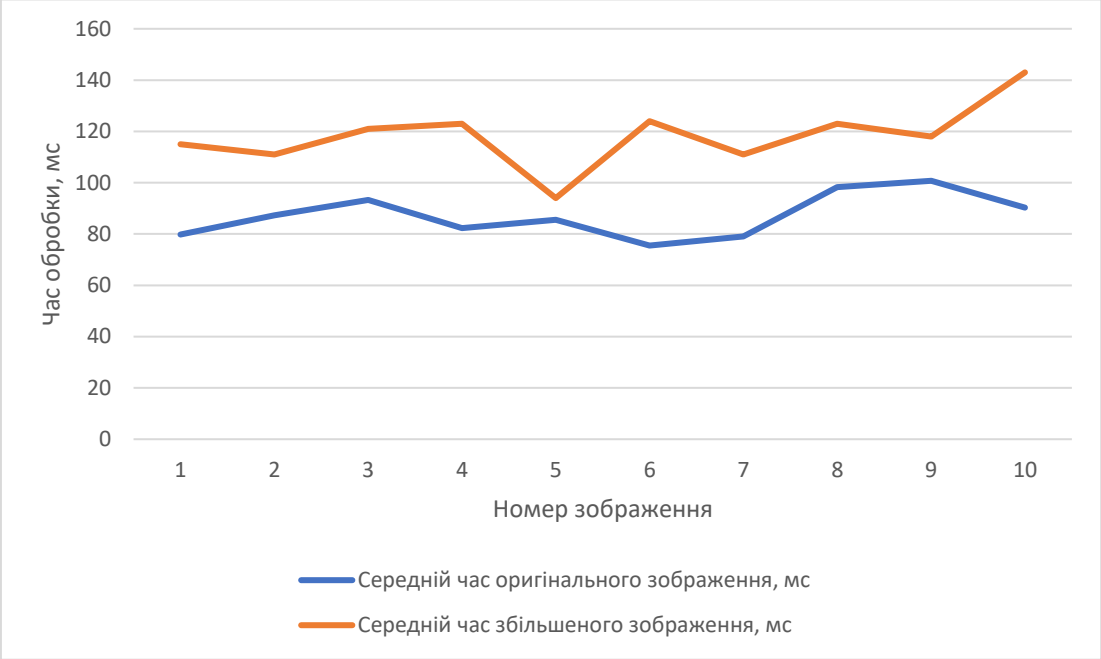


Рисунок 4.4. – Графік порівняння часу оброки зображень

Таким чином, ми можемо зробити висновки, що швидкість роботи залежить від розміру зображення, але збільшення площі зображення збільшило час не лінійно, що означає можливість використання даної нейромережі із зображеннями з високою роздільною здатністю або зображеннями, отриманими за допомогою сканування. Це дозволяє використовувати створену нейромережу для контролю великого різноманіття ДП, а також використовувати її навіть за великих обсягів виробництва.

4.3 Перевірка точності нейромережі

Також необхідно перевірити нейромережу на точність. Для цього було обрано наступний формат експерименту:

– обрано 100 випадкових зображень із тестового датасету;

– зображення обрано із тим розрахунком, щоб кількість кожного з дефектів дорівнювала 100 одиниць;

– кожне із зображень о оброблене за допомогою нейромережі та отримано результат;

– підраховано відхилення від заздалегідь зазначених значень.

Таким чином, було отримано точність для кожного з видів дефекту, графік точності визначення дефектів ДП наведено на рис. 4.5.

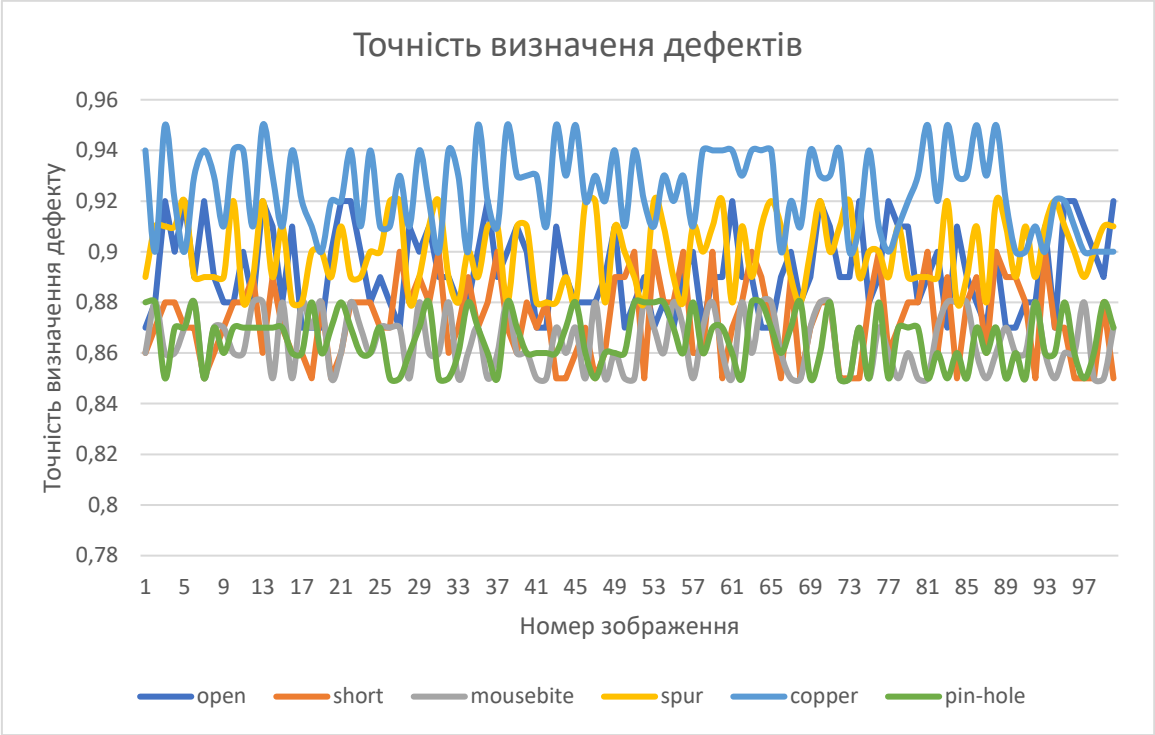


Рисунок 4.5 – Графік точності визначення дефектів нейромережею

Виходячи з даних показників ми можемо дослідити більш детальніші результати точності нейромережі, ніж ті, які було отримано під час розрахунку точки ранньої зупинки, через те, що тоді було досліджено загальну точність без урахування окремих видів дефектів. А також можна зробити висновки про можливість використання даної нейромережі на виробництві. Отримані показники точності створеної неймромережі наведено у таблиці 4.3.

Таблиця 4.3 – Точність нейромережі

Клас дефекту	open	short	mousebite	spur	copper	pin-hole	Загальна точність
Точність, %	89	87	86	89	92	86	88

Дані, отримані у цьому тестуванні, збігаються із даними, отриманими у тесті, проведеному для знаходження точки ранньої зупинки.

Також із отриманих даних ми можемо зробити декілька висновків:

– нейромережа найліпше справляється зі знаходженням зайвих ділянок міді на ДП;

– найгірша точність нейромережі проявляється у випадках, коли потрібно визначити зайві отвори або нерівності провідникового рисунку;

– розпізнавання розривів, коротких замикань і зайвої міді на провідниковому рисунку дають гарний результат із високою точністю.

Тобто така нейромережа краще визначає зайві ділянки мідного шару, ніж його відсутність на струмопровідному рисунку. Гарна точність у випадках відсутності мідного шару може бути пояснена нерівними краями провідника на відміну від більш чітких країв провідника.

4.4 Дослідження точності нейромережі в залежності від куту нахилу зображення

Попри те, що ми отримали гарні результати, як тестові дані було використано зображення з рівними лініями, але на виробництві не завжди можна отримати такі зображення, і часто отримані зображення ДП будуть мати кут нахилу відносно від границь кадру, факторами, які можуть впливати на кут повороту зображення:

– неправильно встановлений сканер або камера;

– відсутність механізму для вирівнювання виробів;

– людський фактор, якщо тестування проводиться у автоматизованому режимі;

– провідниковий рисунок нестандартної форми, зазвичай використовується через естетичні вподобання.

Також на кут нахилу можуть впливати геометричні особливості конкретного типу ДП, який буде розміщено у виробах із складною геометрією корпусу.

Тому для того, аби впевнитися, що нейромережа залишається готовою для використання для контролю ДП і вплив на точність у залежності від кута нахилу зображення є мінімальним, необхідно провести додаткові дослідження.

Таке дослідження було проведено на оригінальному датасеті, який було модифіковано, та для проведення дослідження виконано наступні кроки:

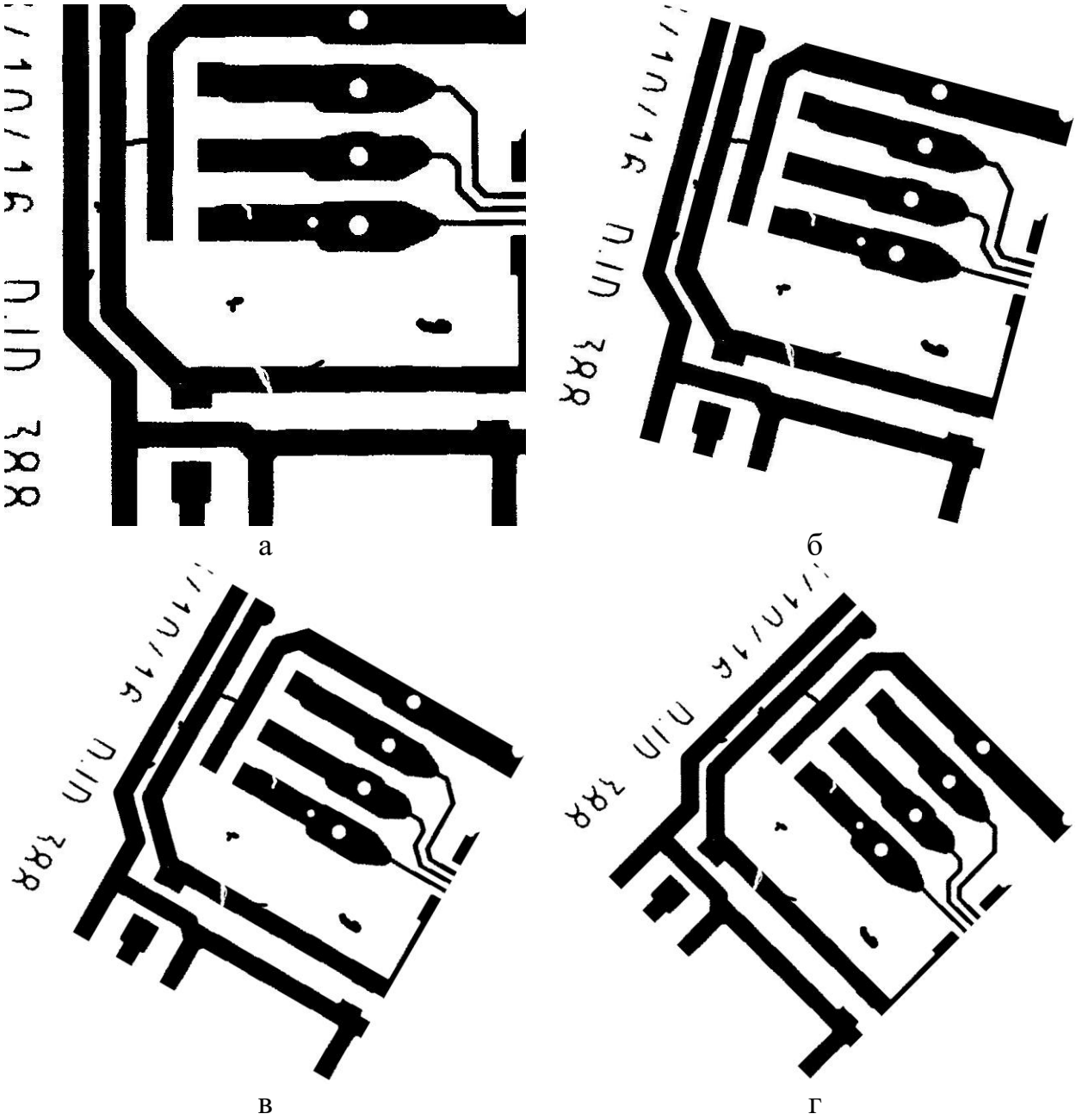
– за допомогою програми під назвою imagemagick було розгорнуто тестові зображення на 15°, 30° та 45° за годинниковою стрілкою, зі збільшенням розміру зображення та без обрізання країв похідного зображення;

– зображення було обрано таким чином, аби кожен із дефектів зустрічався на них рівно 100 разів і мав різноманітні варіанти прояву на друкованій платі;

– також через те, що за умови повороту зображень виникають негативні області, на яких немає провідникового рисунку, програму тестування було модифіковано для урахування регіону, у якому знаходиться провідниковий рисунок. Зображення та отримані результати поза межами такого регіону не впливають на розрахунок точності;

– розмір регіону було зменшено на 2 пікселі відносно розмірів похідного зображення, щоб запобігти похибці через артефакти алгоритмів згладжування країв, які використовуються у процесі перетворення зображень через те, що на фотографіях такі перетворення допомагають досягти найліпшого результату.

Приклад розгорнутих зображень наведено на рис. 4.6.



а б в г

а) похідне зображення; б) зображення, розвернуте на 15°;
в) зображення, розвернуте на 30°; г) зображення, розвернуте на 45°

Рисунок 4.6 – Приклад перетворених зображень

Також список дефектів було перевизначено заново для кожного з перетворених зображень через те, що виконання тригонометричних перетворень до існуючих списків є неможливим через особливості запису дефектів у нейромережі yolo, а також через те, що розмір зображень було змінено під час повороту.

Тобто, враховуючи такі зміни у зображені, було змінено алгоритм визначення точності нейромережі. Регіон дефекту було зменшено, для визначення, що дефект було класифіковано, замість розрахунку загальної площі між визначеним регіоном і регіоном, заданим у текстовому файлі, було обрано алгоритм, за якого співпадінням буде вважатися площа перетину регіонів відносно до заданого вручну регіону. Такий алгоритм дозволяє більш точно визначати наявність дефекту, а не точність його місцезнаходження.

Графік середньої точності знаходження дефектів у залежності від нахилу зображення наведено на рис 4.7 та середні значення точності наведено у таблиці 4.4.

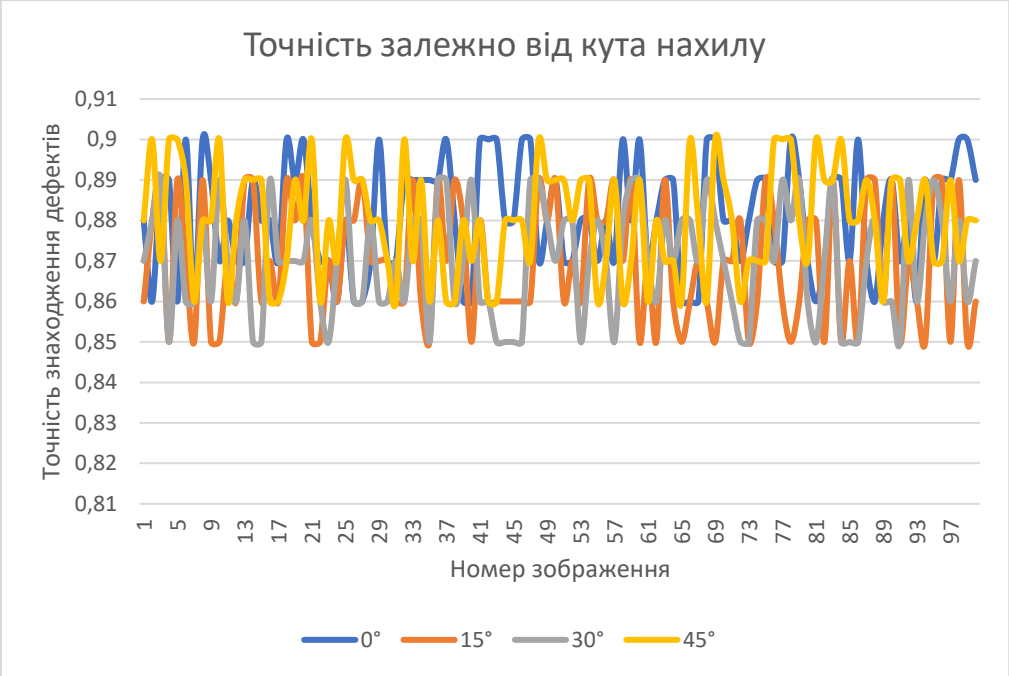


Рисунок 4.7 – Точність нейромережі у залежності від кута нахилу зображення

Таблиця 4.4 – Середні значення точності у залежності від кута нахилу зображення

Кут нахилу	0°	15°	30°	45°	Середнє значення
Точність, %	88	86	87	88	87,5

Таким чином, враховуючи отримані результати, можна вважати, що точність нейромережі залежить від кута нахилу зображення. Але це може бути більш обумовлене артефактами або алгоритмами згладжування, використаними під час створення тестових даних, ніж неточністю навченої нейромережі.

4.5 Висновки до розділу 4

У четвертому розділі за допомогою тестування обрано найбільш точні вагові коефіцієнти та точку ранньої зупинки навчання. За допомогою таких вагових коефіцієнтів визначено швидкість роботи нейромережі, а саме кількість часу, який нейромережа витрачає на класифікацію об’єктів на одному зображенні. Також досліджено вплив розміру зображення на час, який нейромережа витрачає на класифікацію об’єктів. У результаті вплив на час роботи від розміру зображення було підтверджено.

Також було досліджено точність знаходження кожного окремого з видів дефектів за допомогою тестового датасету, у якому кожен із дефектів зустрічається 100 разів. У результаті виявлено, що деякі з дефектів нейромережа визначає більш точно, ніж інші, наприклад, нестача міді у провідниковому рисунку має меншу точність знаходження, а наприклад, окремі ділянки мідного шару, які не під’єднано до жодного з провідників, вона знаходить більш точно.

Окрім того, досліджено вірогідний вплив кута повороту зображення на точність знаходження дефектів за допомогою проведення додаткових досліджень на створеному з цією конкретною метою датасеті та було виявлено, що кут нахилу впливає на точність, і за певних кутів точність може знижуватися на декілька відсотків, але не для всіх типів дефектів, наприклад, точність знаходження чужорідних вкраплень не залежить від кута нахилу зображення. Але невідомо, чи дані результати викликані особливостями нейромережі, або неточностями, які виникли у процесі перетворення зображень із тестового датасету за допомогою програми imagemagick.

5 ОХОРОНА ПРАЦІ

5.1 Вимоги до промислового приміщення

Для забезпечення нормальних умов праці у промисловому приміщені згідно з ДСН 3.3.6.042-99 [36] необхідно, щоб у приміщенні підтримувалися певні показники:

– температура повітря;

– відносна вологість повітря;

– швидкість руху повітря;

– інтенсивність інфрачервоного опромінення;

– температура поверхні.

Окрім того, необхідні показники залежать від пори року та поділяються на показники у теплий період і холодний період, оптимальні показники для робіт у приміщені середньої важкості наведено у таблиці 5.1.

Таблиця 5.1 – Оптимальні показники мікроклімату у приміщені

Період року	Температура повітря, °С	Відносна вологість у приміщені	Швидкість руху повітря, м/с
Холодний період року	19–21	60–40	0,2
Теплий період року	21–23	60–40	0,3

Також через те, що робота дослідника передбачає роботу за персональним комп’ютером згідно з ДБН В.2.5-28-2006 [37] приміщення, у якому він працює, повинно мати наступні параметри:

– мати як природне, так штучне освітлення;

– природне освітлення рекомендовано заводити з лівого боку робочого місця;

– яскравість штучного освітлення під час роботи з монітором має складати 300–500 лк;

– коефіцієнт природного освітлення має складати ≥ 1,5 %.

Також згідно із ДСН 3.3.6.037-99 [38] шум у приміщенні, де працює дослідник, не має перевищувати 50 дБ. Одним із прикладів, як досягти такого рівню шуму, є використання звукопоглинальних матеріалів.

5.2 Електробезпека у промисловому приміщенні

Через те, що виробнича апаратура вимагає підключення до електричної мережі, згідно з ДСТУ Б В.2.5-82:2016[39] для захисту від ураження електричним струмом або короткого замикання необхідно впровадити деякі заходи:

– встановити автоматичні вимикачі живлення;

– використовувати подвійну або посилену ізоляцію у разі використання високої напруги;

– використання заземлення апаратури, у якій така можливість є спочатку або є можливість внесення змін.

5.3 Пожежна безпека у промисловому приміщені

Згідно з ДСТУ 8828:2019 [40] для запобігання пожежі на підприємстві необхідно, аби виконувалися наступні умови:

– запобігання утворенню горючого середовища та запобігання утворенню джерел запалювання у горючому середовищі;

– розміщення пожежонебезпечного устаткування в окремих приміщеннях;

– механізація й автоматизація технологічних процесів, пов’язаних із обігом горючих речовин;

– захист технологічного обладнання, яке працює з горючими речовинами, від пошкоджень;

– встановлення систем пожежної безпеки та автоматичних систем пожежогасіння;

– встановлення вогнегасників на всій площі виробництва.

ВИСНОВКИ

У першому розділі було проведено аналіз літератури за темою контролю друкованих плат. Проаналізовано особливості існуючих методів контролю, їх переваги та недоліки. За результатами аналізу обрано систему контролю друкованих плат на основі візуального методу. Було досліджено існуючі аналоги такої системи. Для визначення дефектів було вирішено використовувати сучасне рішення на основі згорткових нейромереж, а саме нейромережі YoloV4 на основі архітектури DarkNet.

У другому розділі описано необхідні перетворення, які необхідно провести для зображень, а саме переведення зображення у формат градацій сірого із подальшою бінаризацією, яка необхідна, щоб усунути вплив освітлення на подальшу точність визначення дефектів. Також було описано кроки для створення датасету, необхідного для навчання нейромережі. Описано вимоги до датасету, обрано існуючий датасет, який було взято за основу для створення датасету у форматі, який зрозумілий для нейромережі. Також розраховано основні параметри нейромережі.

На основі проаналізованих даних створено нейромережу на основі нейромережі YoloV4, яка базується на архітектурі DarkNet. Параметри нейромережі було модифіковано для підвищення ефективності під час використання для контролю ДП. Створену нейромережу було навчено на основі модифікованого датасету, у процесі навчання визначено момент, за якого похибка перестала зменшуватися. Після завершення навчання обрано набір вагових коефіцієнтів, який дає найбільшу точність.

Після того, як було знайдено оптимальні параметри навчання, було перевірено швидкість роботи нейромережі, а саме було визначено час, що необхідно витратити на перевірку одного зображення. Також було перевірено точність визначення різноманітних дефектів окремо та визначено, що нейромережа ліпше визначає деякі з дефектів, ніж інші. Також перевірено вплив

повороту зображення друкованої плати на точність нейромережі, за малих кутів точність зменшується, але це може бути також обумовлено особливостями алгоритмів, які були використанні під час створення повернутих зображень.

Розроблена система відрізняться від існуючих більшою універсальністю, як і у процесі використання на виробництві, так і у процесі впровадження через те, що хоча й система вимагає використання високопродуктивного апаратного забезпечення, невеликий час обробки кожного із зображень дозволяє використовувати апаратні ресурси у режимі спільного користування на декількох етапах виробництва або на сусідніх виробничих лініях.

Також хоча дана система і показує гарний результат, вона має простір для вдосконалення. Одним із прикладів такого вдосконалення може бути розширення датасету із додатковими зображеннями або дефектами, специфічними для певного етапу виробництва ДП. Також можлива адаптація нейромережі до обладнання, яке використовується на виробництві, наприклад, до оптичного обладнання у вигляді сканерів або камер.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Невлюдов І.Ш. 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 технол / І.Ш. Невлюдов, В.В. Косенко, В.В. Євсєєв. – Харьков: ХНУРЕ, 2019. – 55 с.

2. ДСТУ 3008-15. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К. Держстандарт України, 2017. – 29 с.

3. Основи наукових досліджень / І.Ш. Невлюдов [та ін.]. – навч. посі. – Кривий Ріг: КК НАУ, 2017. – 344 с.

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

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

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

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

8. Медведев А.М. Надежность и контроль качества печатного монтажа. / А.М. Медведев // Радио и связь. – 1986.

9. Городов Владимир Александрович. Электрический контроль печатных плат и узлов / Городов Владимир Александрович // ЭЛЕКТРОНИКА:

Наука, Технология, Бизнес. – 2004. – Vol 7. – С. 68-71.

10. Васільєв В. А. Автоматизовані методи контролю друкованих плат / В.А. Васільєв // Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2020). – 2020. – С. 150-154.

11. Карпов С.В. Проблемы контроля многослойных печатных плат. / С.В. Карпов // Радиотехника. – 2003.

12. Kingatua A. PCB Inspection and Testing Techniques [Електронний ресурс]. – Режим доступу: www/ URL: https://medium.com/supplyframe-hardware/pcb-inspection-and-testing- techniques-30631a885109.

13. Городов В.А. Методы электрического контроля печатных плат [Електронний ресурс]. – Режим доступу: www/ URL: https://www.tech-e.ru/2005_1_68.php.

14. Визуальный и оптический контроль печатных плат - JUTZE [Електронний ресурс]. – Режим доступу: www/ URL: https://jutze.ru/stati/vizualnyi-i-opticheskii-kontrol-pechatnykh-plat/ 20.10.2020.

15. Визуально оптический контроль - цель, применение, технология, приборы | Speranza [Електронний ресурс]. – Режим доступу: www/ URL: https://speranza-ua.com/news/vizualno-opticheskij-metod-kontrolya/ 02.12.2020.

16. INSPECTION METHODS FOR PCB ASSEMBLY [Електронний ресурс]. – Режим доступу: www/ URL: https://rushpcb.com/inspection-methods-for-pcb-assembly/.

17. Оптическая инспекция печатных плат и финишной сборки [Електронний ресурс]. – Режим доступу: www/ URL: https://www.smt-prof.com.ua/оптическая-инспекция-печатных-плат-3/ 20.10.2020.

18. What is an artificial neural network? Here’s everything you need to know | Digital Trends [Електронний ресурс]. – Режим доступу: www/ URL: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/ 02.12.2020.

19. Mihajlovic I. Artificial Neural Networks in Practice | Towards Data Science [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/artificial-neural-networks-in-practice-c950c4be47ee 02.12.2020.

20. 7 Types of Activation Functions in Neural Networks: How to Choose?

[Електронний ресурс]. – Режим доступу: www/ URL: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ 02.12.2020.

21. Artificial Neural Network | Brilliant Math & Science [Електронний ресурс]. – Режим доступу: www/ URL: https://brilliant.org/wiki/artificial-neural-network/ 20.10.2020.

22. Commonly Used Machine Learning Algorithms | Data Science [Електронний ресурс]. – Режим доступу: www/ URL: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ 02.12.2020.

23. Kostadinov S. Understanding Backpropagation Algorithm | Towards Data Science [Електронний ресурс]. – Режим доступу: www/ URL: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd 02.12.2020.

24. Batsuuri T. Back-Propagation Algorithm I. Definitions, Concepts, Algorithms with… [Електронний ресурс]. – Режим доступу: www/ URL: https://medium.com/swlh/back-propagation-algorithm-85c65e6fc359 20.10.2020.

25. CS231n Convolutional Neural Networks for Visual Recognition [Електронний ресурс]. – Режим доступу: www/ URL: https://cs231n.github.io/convolutional-networks/ 21.10.2020.

26. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science [Електронний ресурс]. – Режим доступу: www/ URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 20.10.2020.

27. A Gentle Introduction to the Rectified Linear Unit (ReLU) [Електронний ресурс]. – Режим доступу: www/ URL: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ 02.12.2020.

28. Darknet: Open Source Neural Networks in C [Електронний ресурс]. – Режим доступу: www/ URL: https://pjreddie.com/darknet/ 02.12.2020.

29. Redmon J. YOLOv3: An Incremental Improvement / J. Redmon, A. Farhadi.

30. Шпаргалка по OpenCV — Python [Електронний ресурс]. – Режим доступу: www/ URL: https://tproger.ru/translations/opencv-python-guide/ 02.12.2020.

31. About - OpenCV [Електронний ресурс]. – Режим доступу: www/ URL: https://opencv.org/about/ 23.11.2020.

32. Realtime Computer Vision with OpenCV / K. Pulli [та ін.]. – 2012. – 9 с.

33. tangsanli5201/DeepPCB: A PCB defect dataset. [Електронний ресурс]. – Режим доступу: www/ URL: https://github.com/tangsanli5201/DeepPCB 22.10.2020.

34. 7 Tips for Choosing the Best PCB Via Option | Tempo [Електронний ресурс]. – Режим доступу: www/ URL: https://www.tempoautomation.com/blog/7-tips-for-choosing-the-best-pcb-via-option/ 19.11.2020.

35. AlexeyAB/darknet: YOLOv4 - Neural Networks for Object Detection (Windows and Linux version of Darknet) [Електронний ресурс]. – Режим доступу: www/ URL: https://github.com/AlexeyAB/darknet 09.11.2020.

36. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 від 01.12.1999. – 1999. – 10 с.

37. Мінбуд України. Інженерне обладнання будинків і споруд ПРИРОДНЕ І ШТУЧНЕ ОСВІТЛЕННЯ / Мінбуд України. – 2006.

38. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 від 01.12.1999 № 37 [Електронний ресурс]. – Режим доступу: www/ URL: https://zakon.rada.gov.ua/rada/show/va037282-99#Text 25.11.2020.

39. ДП «УкрНДНЦ». ДСТУ Б В.2.5-82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом / ДП «УкрНДНЦ». – Київ, 2016. – 109 с.

40. ДП «УкрНДНЦ». ДСТУ 8828:2019 - ПОЖЕЖНА БЕЗПЕКА / ДП «УкрНДНЦ». – 2019. – 87 с.