DEFECT AND NON-DEFECT IMAGE CLASSIFICATION SYSTEM OF C-SAM IMAGES USING ARTIFICIAL INTELLIGENCE TECHNIQUES

DIVIEN RAJ KUMAR A/L BALA KRISHNAN

BACHELOR OF MECHATRONIC ENGINEERING

WITH HONOURS

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DEFECT OR NON-DEFECT IMAGE CLASSIFICATION SYSTEM OF C-SAM IMAGES USING ARTIFICIAL INTELLIGENCE TECHNIQUES

DIVIEN RAJ KUMAR A/L BALA KRISHNAN

A report submitted in partial

fulfilment of the requirements

for the degree of Bachelor of Mechatronics Engineering

Faculty of Electrical Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I declare that this thesis entitled "DEFECT AND NON-DEFECT IMAGE CLASSIFICATION SYSTEM OF C-SAM IMAGES USING ARTIFICIAL INTELLIGENCE TECHNIQUES“ is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.

Signature	:	[illegible]
Name	:	DIVIEN RAJ KUMAR A/L BALA KRISHNAN
Date	:	10/1/2024

APPROVAL

I hereby declare that I have checked this report entitled DEFECT AND NON-DEFECT IMAGE CLASSIFICATION SYSTEM OF C-SAM IMAGES USING ARTIFICIAL INTELLIGENCE TECHNIQUES, and in my opinion, this thesis fulfils the partial requirement to be awarded the degree of Bachelor of Mechatronics Engineering with Honours

Signature	:	[illegible]
Supervisor Name	:	DR.TARMIZI BIN AHMAD IZZUDIN
Date	:	

DEDICATIONS

I dedicate this project to my unwavering support system to God Almighty, my family, whose love, encouragement, and sacrifices have been the driving force behind my academic journey. To my parents, for instilling in me the value of education and providing endless encouragement. Thank You.

ACKNOWLEDGEMENTS

During the preparation of this report, I was in contact with many researchers, academicians, seniors, and friends. They have contributed to my understanding and thought. I want to express my appreciation to those who have helped me to finish the report. Firstly, I would like to express my most tremendous gratitude to my main project supervisor, Dr. Tarmizi Bin Ahmad Izzudin for all the guidance, patience, and advice while completing the project. His guidance is beneficial and helps me finish this report within the allocated time.

Moreover, I would also like to express my appreciation and gratitude to my panels, Dr. Mohd Rusdy Bin Yaacob and Dr. Nur Latif Azyze Bin Mohd Shaari for advice and critics. After they provided me sincere advice, I managed to improve my work and identify my mistakes, which contributed to the report completion. Besides, I am thankful to University Teknikal Malaysia Melaka (UTeM) for funding this project.

Lastly, I am deeply indebted to my beloved family and friends for their mental support and help, also their sacrifice for me during this completion of my study and final year project.

ABSTRACT

This research presents an innovative method for automating the classification system in the semiconductor industry, addressing the inefficiencies and high costs associated with manual inspection of C-SAM images. The developed image classification system leverages modern technologies such as Artificial Intelligence (AI) for object detection and TensorFlow for data recognition and pre-processing. Its primary objective is to automate the identification, determination, and classification of C-SAM images into categories of defects and non-defects. The system enhances data gathering, annotation, model selection, training, and software configuration on a Convolutional Neural Network (CNN) architecture through thorough data pre-processing and customization. This study highlights the inefficiencies of conventional neural network techniques and the adverse effects of manual inspection, emphasizing the impact on time and cost.

By automating the image classification process, the project aims to reduce the need for human labor, improve operational efficiency, and provide a user-friendly solution. The scope of the study includes offering employees' instructions on data labeling and customization, restricting the model training learning rate, and specifying the number of epochs implemented. The project's objectives are to develop an algorithm capable of differentiating between defect and non-defect images using AI techniques, evaluate the classification performance of the developed algorithm, and provide a potential solution to improve the employee experience in the semiconductor sector. Additionally, the project includes implementing the system on a Graphical User Interface (GUI) to enhance user interaction and usability.

ABSTRAK

Penyelidikan ini memperkenalkan kaedah inovatif untuk mengotomatiskan sistem klasifikasi dalam industri semikonduktor, menangani ketidaksempurnaan dan kos yang tinggi yang berkaitan dengan pemeriksaan manual imej C-SAM. Sistem pengklasifikasian imej yang dibangunkan memanfaatkan teknologi moden seperti Kecerdasan Buatan (AI) untuk pengesanan objek dan TensorFlow untuk pengiktirafan data dan pra-prosesan. Objektif utamanya ialah untuk mengotomatiskan pengenalan, penentuan, dan pengklasifikasian imej C-SAM ke dalam kategori cacat dan bukan cacat. Sistem ini meningkatkan pengumpulan data, anotasi, pemilihan model, latihan, dan konfigurasi perisian pada arsitektur Rangkaian Neural Convolutional (CNN) melalui pra-pemprosesan data yang menyeluruh dan penyesuaian.

Kajian ini menyoroti ketidaksempurnaan teknik rangkaian saraf konvensional dan kesan negatif pemeriksaan manual, menekankan kesan pada masa dan kos. Dengan mengotomatiskan proses klasifikasi imej, projek ini bertujuan untuk mengurangkan keperluan tenaga manusia, meningkatkan kecekapan operasi, dan menyediakan penyelesaian yang mudah digunakan. Kawasan kajian termasuk menawarkan arahan pekerja mengenai label dan penyesuaian data, mengehadkan kadar pembelajaran model latihan, dan menentukan bilangan era yang dijalankan. Objektif projek ini ialah untuk membangunkan algoritma yang mampu membezakan antara imej cacat dan tidak cacat menggunakan teknik AI, menilai prestasi klasifikasi algorithm yang dibangunkan, dan menyediakan penyelesaian yang berpotensi untuk meningkatkan pengalaman pekerja dalam sektor semikonduktor. Selain itu, projek ini termasuk pelaksanaan sistem pada GUI (Graphical User Interface) untuk meningkatkan interaksi pengguna dan kegunaan.

TABLE OF CONTENT

LIST OF TABLE

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CNN	-	Convolutional Neural Network
R-CNN	-	Regional Convolutional Neural Network
ML	-	Machine Learning
AI	-	Artificial Intelligence

LIST OF APPENDICES

CHAPTER 1:

INTRODUCTION

1.1 Research Background

The evaluation of precise, efficient, and various automated methods for defect recognition, classification and quality control in materials has never been critical in this generation. The era of advanced technology and non-destructive testing has shown that the consolidation of Confocal Scanning Acoustic Microscopy (C-SAM) with addition of Artificial Intelligence (AI) and Machine Learning (ML) techniques have in clutch together in immerse potential. This project outlines the development and application of an Artificial Intelligence/ Machine Learning based system to automatically identify and categorize defects in C-SAM images, and therefore, enhancing the utility of this technology for non-destructive testing.

These techniques have occasionally taken the ordinary form of roll calls, but in more intriguing instances, they have taken the form of surprises tests [1]. Most employees in the industries or companies especially in semiconductor fields put their hard work and effort of collecting and recording the scientific data of a packaging C-SAM image defects through manually. Due to the difficulties, including the manual method of identifying and classifying on which defect contains. The manual technique requires employees to recognize the classify the defects which leaves possibility for many errors. This manual method of classifying numerous defects takes lots of time and it’s difficult to determine the number of packaging defects one by one and identify the location of the defect.

Most companies and Industries in underdeveloped countries were still cultivating the method manually especially for Semiconductor and Microelectronic parts inspection. It has become crucial to be able to Identify the packaging whether it is under defect or non-defect category and able to categorize the type of defect and its location. This will aid in reducing the number of operators required for inspection of the C-SAM image defect and reducing the time taken of the inspection. The automated defect classification and identification system based on Artificial Intelligence is one of the technical advancements that is accessible to reduce the rates of human errors and reducing human effort.

Confocal Scanning Acoustic Microscopy is a cutting-edge imaging technique that provides high-resolution and non-destructive imaging capabilities. The working principle of the method is based on the sound wave reflection where the acoustic waves experience different density irregularities and with that, it addresses the focalized Ultrasound from MHz to GHz pulses to penetrate the material interface and boundaries to examine sample interior defects such as voids or delamination. Defects such as fine cracks, voids, porosity, and delamination's can be classified and recognized effectively using C-SAM than other NDT methods such as X-Ray or IR Imaging due to its high sensitivity to the elastic properties of the materials it travels through.

A defect's "Classification" or "Recognition" refers to a quality control that typically classifies the quality of defects that can be various types. In this case, classification can be stated as a identifier of the packaging defects traits by determining the optimal location of the defects through the parts of packaging. On the other hand, "identification" uses one of many defects matching to identify the type of defect which has been contaminated with the help of Artificial Intelligence. Moreover, with the implementation of AI technique, it is now possible to identify whether the packaging chips are under defect or non-defect for the purpose of identification or classification thanks to advancements in Artificial Intelligence technology.

In this research, an automatic defect classification and recognition of C-SAM images using an AI based system that addresses the challenges of C-SAM image detection throughout the industry is proposed. The system allows us to classify and identify the packaging chips that went through C-SAM method and produce the output through image. So, the image will be classified whether the chips are categorized as defect or non-defect based on acoustic waves characteristics.

1.2 Motivation

Image classification and recognition of Confocal Scanning Acoustic Microscopy or known as C-SAM plays in non-destructive testing within industries, particularly electronics manufacturing. The miniaturization of electrical components has become increasingly intricate that makes it challenging to identify defects precisely using traditional methods which is through manual inspection. Defect classification of the images through Artificial Intelligence technique plays a significant role in identifying and classifying the categories of defects and non-defects and yet differentiating the type of defects to avoid any failure of inspection or reduce production costs because defects of a product will cause the product to be damaged. Based on Figure 1.1, it stated that the process of C-SAM method using acoustic wave to interface the material and the defects that can be affected on the packaging sample.

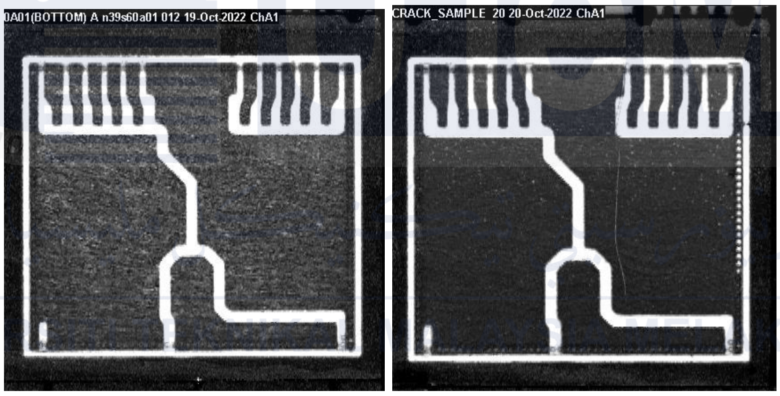

0A01(BOTTOM) A n39s60a01 012 19-Oct-2022 ChA1
CRACK_SAMPLE 20 20-Oct-2022 ChA1

Figure 1.1 The images of IC packages through Confocal Scanning Acoustic Microscopy (C-SAM) method

1.3 Problem Statement

The effectiveness of Confocal Scanning Acoustic Microscopy (C-SAM) as a high-resolution imaging technique is indisputable. However, the manual inspection and analysis of CSAM images pose significant challenges, including subjectivity, time consumption, and the potential for human error. The need for a more efficient, accurate, and automated approach to defect recognition and classification in CSAM images has become increasingly apparent.

The primary disadvantage of manual C-SAM method is that the manual inspection of C-SAM images relies on the subjective judgment of operators which leads to inconsistencies in defect recognition. Inter-operator variability can produce out the result in different interpretations of same defect images which will impact the reliability of inspection outcomes. Although it can be challenging for a operator to keep track on identifying the defects, it can also be challenging for them to produce the output file result in the given time.

CSAM images often capture intricate internal structures of materials with high resolution. The complexity of these structures can pose challenges for manual inspection, as operators need to navigate through a vast amount of information to identify potential defects. Human error becomes more likely in the presence of intricate patterns, layered structures, or subtle variations in acoustic properties that may indicate defects.

1.4 Objectives

The objective of this project are as follows:

- To Develop an algorithm that can differentiate between the defect and non-defect images using AI techniques.
- To evaluate the classification performance of developed algorithm
- To Design an Graphical User Interface for the defect recognition system which facilitates easy integration C-SAM equipment and ensuring accessibility for operators.

1.5 Scope and limitation

The project will focus on the following segments:

i. The project involves creating a defect classification and recognition system using AI techniques and Python. The system is specifically aimed at classifying and recognizing defects in C-SAM images.
ii. The dataset is trained as a machine learning model, focusing on C-SAM images.
iii. The project enhanced the system's ability to accurately and precisely recognize and classify defects under deep learning techniques.
iv. The system includes an image classification software that acts as a GUI, displaying the classified defects.

CHAPTER 2:

LITERATURE REVIEW

2.1 Introduction

There are various scholarly or research articles available in the academic literature on defect and non-defect classification system by involving Artificial Intelligence technique such as supervised and unsupervised classification and yet Convolutional Neural Network for implementing image classification and recognition system, few related works are described briefly in this research.

2.1.1 Confocal Scanning Acoustic Microscopy (C-SAM)

Confocal Scanning Acoustic Microscopy or known as C-SAM is an effective technique and non-destructive method that can find hidden flaws in biological, elastic and non-transparent hard materials. Moreover, Since the early 1980s, scanning acoustic microscopy has been a non-destructive method for inspecting faults and delamination that occur during the production process, allowing for visualization of thick-film circuits' interior architecture[2]. This method has the function of monitoring the internal features of a package IC sample and eventually determine the physical flaws and defects. SAM analyses acoustic impedance variations in integrated circuits (ICs) and related materials using ultrasonic waves outlined by Yu et.al. [3]. SAM's great potential in microelectronic packaging has been demonstrated by defect inspection[4], which found microcracks, voids, and delamination[4]

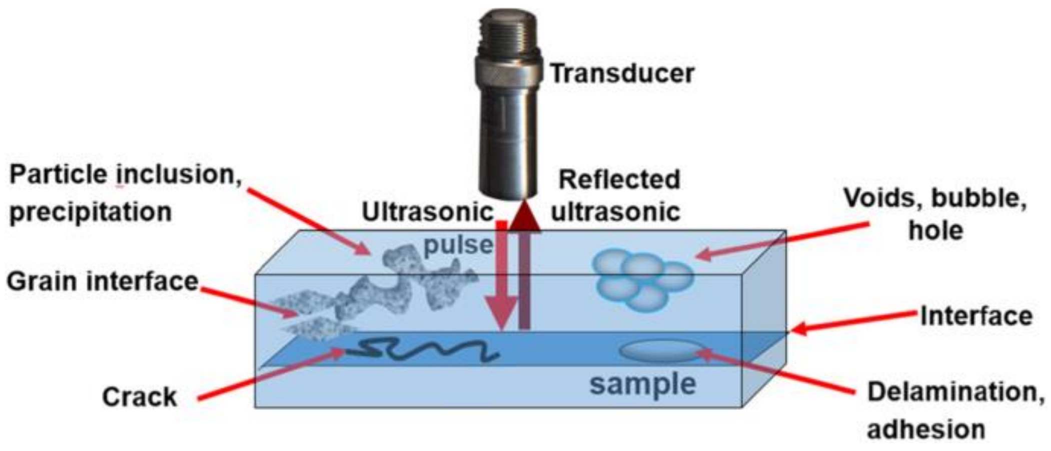


Figure 2. 1 The process of Confocal Scanning Acoustic Microscopy (C-SAM)[5]

2.1.2 Principles of Confocal Scanning Acoustic Microscopy

C-SAM technique consist of an image processor, a mechanical scanner, and an ultrasonic emitting transducer that serves as a crucial lens that transmits and concentrates the sound wave. It stated that the latter targets a small spot on the target item with an ultrasonic signal (TX)[7]. In addition to the required information, the resolution and examination depth of SAM measurements depend on the acoustic frequency, the material qualities of the transducer, and the intricacy of the structure of the specimen being tested[8]. When there is a change in the acoustic impedance, Z between the interfaces of interior materials, fractions of the incident acoustic energy by TX are backreflected.

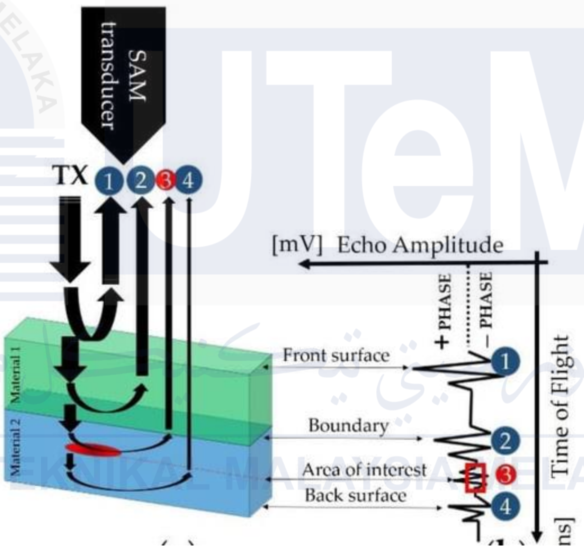


Figure 2. 2: C-SAM Transducer and frequency implementation

The acoustic waves, or ultrasonic pulses are produced via a piezoelectric transducer that consists of a radiofrequency coil and a magnet. The constituents of crystals are single lithium niobate (LiNbO3) Lead magnesium niobate-lead titanate crystals (PMNPT) quartz, piezoelectric ceramics, or single crystals for frequencies that are less than 100 MHz. The echo amplitude measures the intensity of the reflected signal, or greater brightness in the 2D image, the higher the impedance mismatch at the interface. Since entire reflection of the ultrasonic wave happens at an interface with air (Z = 0), SAM is extremely sensitive to any trapped air in the sample being tested[8]. When analysing and processing the echo signals, the locations of any particles, voids, air bubbles, delamination, or cracks are taken into account.

2.2 Defect Image Classification categories

2.2.1 Convolutional Neural Network

A Convolutional Neural Network, abbreviated as CNN or ConvNet, is a category of neural networks specifically designed for handling data with a grid-like structure, such as images. A digital image is a representation of visual information in binary form, consisting of pixels organized in a grid. Each pixel is assigned values to indicate its brightness and color. CNN method was the first suggestion for the purpose of Neural Network implementation because able to find application across various tasks and yet demonstrating excellent performance on the image classification authorized by Sharma et al. [9]. The CNN method has a special characteristic where It is capable of picking up highly ethereal features. and able to identify the objects efficiently. In comparison to detection method for image classification, the system is more systematic and convenient.

In this study by Alzubaidi et al. Alzubaidi et al., [10], an automatic detects with significant features is explained and presented. It involves embedding the image defects into an image classification system that will be detected over a algorithm trained data or model through deep learning. Such original concepts will replace the manual inspection techniques when image classification via CNN trained model will entirely become widely used and accepted in most industries. The goal of the research was to demonstrate the fundamental properties and integration options for using CNN and Deep Learning method for Detection and Classification.

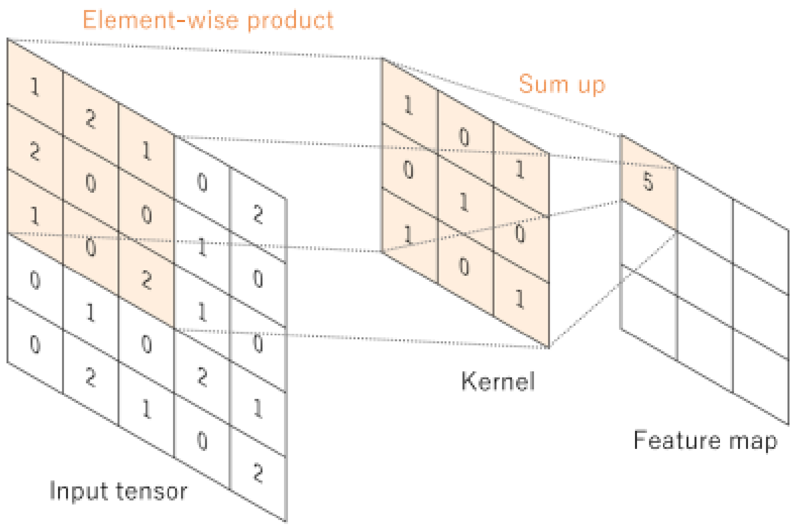


Figure 2. 3 Convolutional Layer[12]

Pooling layer has been involved and executed in the Convolutional Neural Network method by introducing Translation invariance through down sampling the feature map, these layers lessen the CNN model's tendency to overfit. The layers gradually reduce the spatial dimension of the representation to reduce the amount of network computations and parameters and the output will be in a maximized or averaged value. Maximizing combination is often implemented for an optimal function where in cutting back on computation weights or overfitting highlighted by Nguyen et al.[12]

There are various types of pooling layer that are been implemented throughout the CNN method including tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling but commonly extracted pooling layer are Max pooling and Average Pooling. Max pooling illustrates the image that is returned by max pooling is sharper than the original and keeps the most noticeable aspects of the feature map meanwhile Average pooling keeps the features map's average feature values are preserved using average pooling. It retains the essential elements of each feature in the image while smoothing it too.

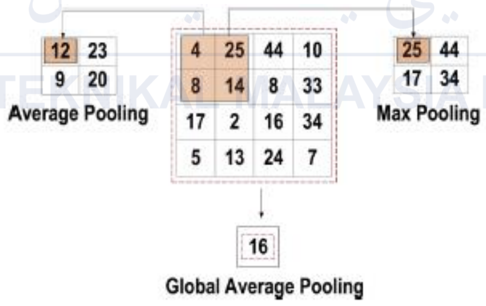


Figure 2. 4 Max Pooling and Average Pooling[13]

2.2.2 Supervised learning

Labelled input and output data are necessary for supervised machine learning in the training stage of the machine learning model lifecycle where the data set will be frequently labelled for data training for training purposes and evaluate the model. The model can be used to predict outcomes and classify new and unexplored information once it has learned the relationship between the input and output data. Supervised learning is able to utilized as a predictive model to forecast trends and future change and to classify unknown data into predetermined categories moreover able to identify objects and the characteristics that categorize them. Supervised machine learning techniques are frequently employed in the training of predictive models. It can make predictions from previously unseen data by identifying patterns in the input and output data. With only a small percentage of the labelled data needed for typical supervised learning techniques, these pre-trained big language models have demonstrated the ability to perform effectively on novel tasks and achieve state-of-the-art outcomes for a variety of NLP tasks.[14]

Supervised learning through image classification involves several steps. Initially, a labelled dataset with photos associated with matching class labels is gathered. To aid in the training and assessment of models, the dataset is subsequently split into training and testing sets. Pre-processing involves resizing photos to a standard size, normalizing pixel values, and maybe using data augmentation techniques to improve model's capacity for generalization. The layers and linkages between them are specified in the definition of the model. During training, the model minimizes a selected loss function typically categorical cross-entropy by modifying its parameters based on the labelled training data. Once training is complete, the model is evaluated on the testing set, and various classification metrics, including accuracy and precision, are calculated to assess its performance.

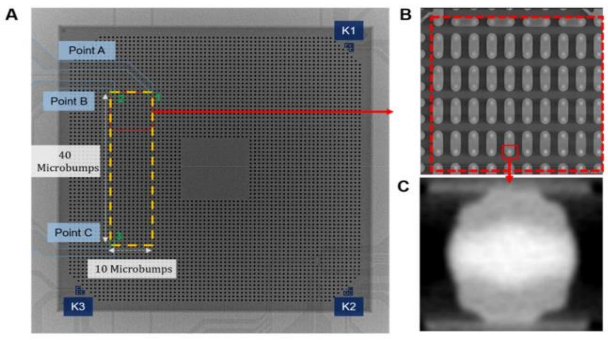


Figure 2. 5 Image Classification of SAT method through supervised learning[15]

2.3 Overview of image classification

The method of image classification involves applying labeling to an image in order to define the image into various categories. The database that has predetermined patterns that are compared with an object to categories it into the right category makes up the Image Classification system[16]. Besides that, image classification is currently becoming one of the technologies in engineering research that is expanding quickly and crossing borders [18]One of the achieving aims of image classification technology is to improvise the precision and accuracy of defect classification and recognition of Integrating Circuits through C-SAM method especially where the autonomous detect classification would be wisely used in the future.

Generally, image classification can be summed up in four steps, Pre-Processing, Detection and Extraction, Training and Classification[19]. Preprocessing is a primary component analysis where it allows atmospheric correction and noise reduction through the image classification process and yet allows the input images to be modified. The improvement of the image data includes obtaining a normalized image, enhancing contrast and obtaining grey-scale image. Next step is the Object detection and extraction which includes determining the position and other details of a moving object picture captured by a camera and, during the extraction process, predicting the trajectory of the object in question inside the image plane. Then, the dataset will undergo a training phase by choosing the specific characteristic that most accurately characterizes the pattern. Lastly, classification of the object step take place where it will categorizes detected objects into predetermined classes by using an appropriate technique that compares the picture patterns[19]

Some advanced pre-processing comes from machine learning such as edge detection, background subtraction and using the processed data in deep learning. These methods can be directly trained to detect image defects or use as a pre-processing approach. For example, Shiyang Yan et al. used GMM to subtract the background and feed the data to deep-CNN[20]. Seyed et al. use edge computation to locate the human location in the image and feed the data into the CNN model for training[21]. The process of processing digital images is shown in Figure 2.4 below. Before the input image can be processed, it must be obtained. After that, it will go through a loop that begins with feature extraction, associative storage, comparison, and knowledge-based processing. However, picture 2.5 illustrates analogue image processing.

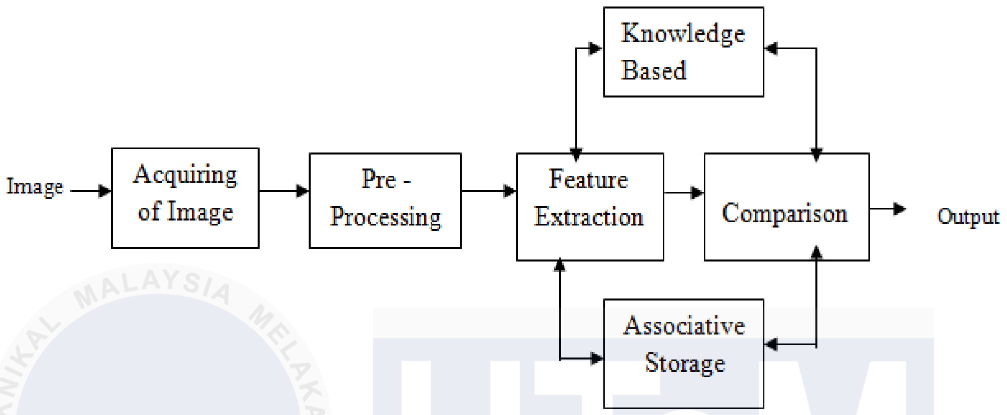


Figure 2. 6 Digital Image Processing

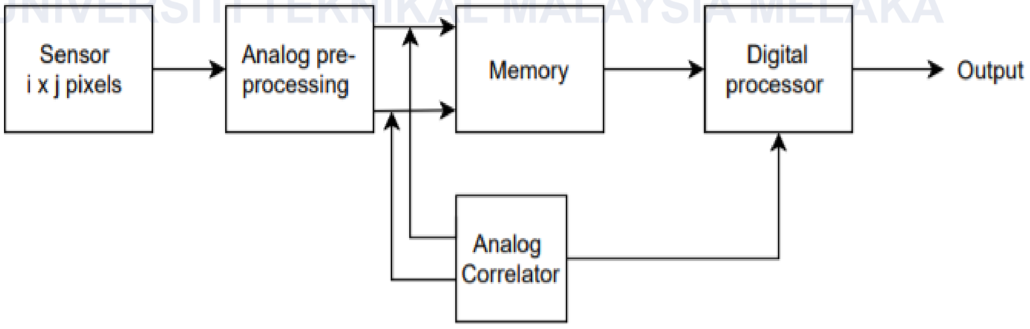


Figure 2. 7 Analogue Image Processing

2.4 Methods of Detecting and Recognizing IC packages through Image classification

This work focuses on the identification and acknowledgement of C-SAM images through classification system. However, there are few concepts that need to be highlighted such as [22] which proposed a method for detecting and classifying C- SAM.

2.5 Deep Learning Convolutional Neural Network

The convolutional neural network (CNN) was invented in recent years has been extensively employed in the image processing industry because it’s superior performance at addressing issues with image categorization and recognition and has significantly increased the precision of numerous machine learning applications. Convolutional Neural Network (CNN) is am multilayer neural network which significantly the most classical and common deep learning framework where Newman et al.[23] have proposed a new reconstruction approach based on CNN algorithm where the performance and speed are displayed while Wang et al.[24] discussed about methods about CNN model which is pretraining or fine-tuning and yet the hybrid method.

CNN consists of an input layer, convolutional layer, activation layer, pool layer, full connection layer and final output layer from the input to output. The convolutional neural network layer transmits input data and creates connections between various computational neural nodes layer by layer, the original data’s feature signals are decoded, inferred, converged and mapped to the hidden layer feature space which been stated by Sun et al[25]. The first layer that is utilized to extract various information from the input picture or photo is called the convolutional layer.

This layer performs the convolutional mathematical process between an input image and a filter with a specified MxM size. The dot product between the filter and the portions of the input image about the filter's size (MxM) is calculated by swiping the filter over the image. The output, known as the feature map, provides us with details about the image, including its edges and corners. This feature map is later supplied to additional layers so that they can identify more features from the input image. After applying, the CNN convolution layer transfers the output to the following layer.

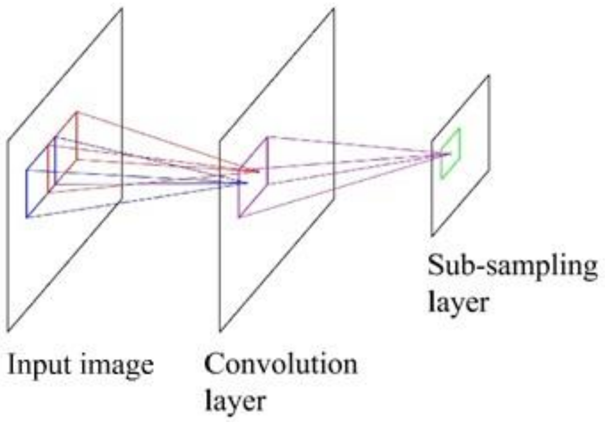


Figure 2. 8 Convolutional Layer

The next layer is followed by pooling layer which aimed to reduce the convolved feature map's size in order to save computing expenses and develop new filters based on the desired rules stated by [26]. Reducing the connections between layers and performing independent operations on every feature map is how this is accomplished. There are various kinds of pooling processes, depending on the technique employed. It essentially provides an overview of the features produced by a convolution layer. The largest element in a feature map is used in max pooling. The average of the components in an Image segment of a predetermined size is determined by average pooling. Sum Pooling computes the total sum of the components in the predefined section. Typically, the Pooling Layer acts as a link between the Fully Connected Layer and the Convolutional Layer. The convolution layer's features are generalized by this CNN model, which also aids in the networks' autonomous feature recognition. This also helps to decrease the computations within a network.

Finally, the output layer is about the fully connected layer which known as Multilayer Perceptron which is a part of an artificial neural network and comprises of many neurons connected by connecting weights[27]. The layer is often located in the network's bottom layer. It receives input from the convolutional output layer or final pooling and flattens it before sending it to the next layer. Equitable allocation of the output, which is a vector (3D matrix) created by unrolling all the values of the result produced after the last pooling or convolutional layer. This approach is simple strategy for researching high-level nonlinear combinations of a feature that the output convolutional layer represents[28].

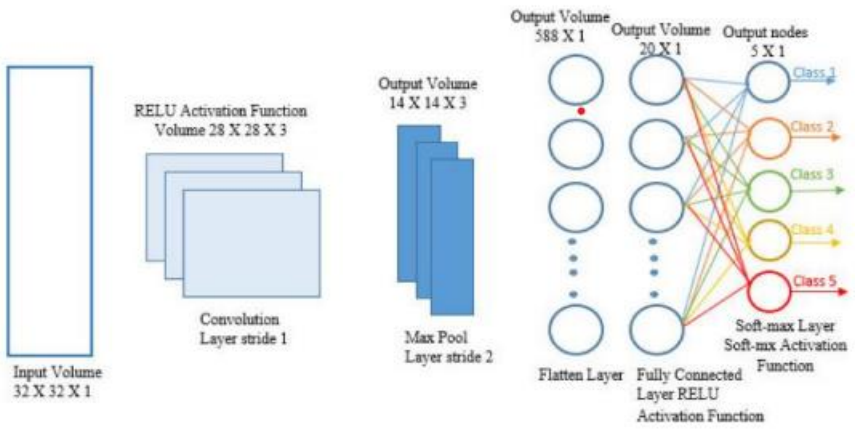

Input Volume 32 X 32 X 1
RELU Activation Function Volume 28 X 28 X 3
Convolution Layer stride 1
Output Volume 14 X 14 X 3
Max Pool Layer stride 2
Output Volume 588 X 1
Flatten Layer
Output Volume 20 X 1
Fully Connected Layer RELU Activation Function
Output nodes 5 X 1
Class 1
Class 2
Class 3
Class 4
Class 5
Soft-max Layer Soft-mx Activation Function

Figure 2. 9 Fully Connected Layer

2.6 Image Classification System using Transfer Learning CNN

Transfer Learning is a Machine Learning technique whereby a model is trained and constructed for one activity, then applied to another task that is similar to it. It describes a scenario in which knowledge gained in one context is applied to enhance optimization in a different one[29]. Transfer learning is commonly applied when there is a new dataset smaller than the original dataset used to train the pre-trained model[30].Transfer learning is being classified to three subclasses which is inductive, transductive and unsupervised which had been described by Pan et al.[31]. Moreover, transfer learning with cognitive research is able to improve performance on a new task with involvement of fine tuning of pre-trained network. The architecture parameter which mostly involved in the pre-trained model is VGG16 which is a convolutional neural network trained on 1.2 million images to sort 1000 distinct categories.

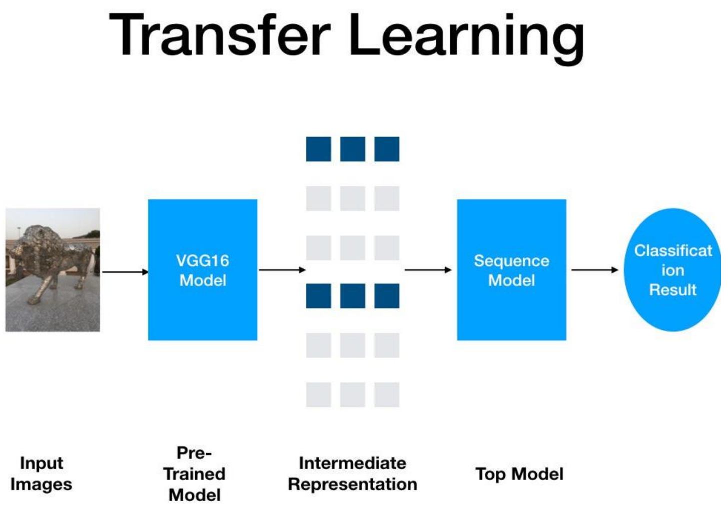


Figure 2. 10 Image Classification through Transfer Learning method

Convolutional neural network VGG16 was trained using a portion of the ImageNet dataset, which consists of more than 14 million photos divided into 22,000 categories which been stated by Simonyan et al.[32]. The ImageNet dataset that contains different variety of images of a specific model and once the previously instructed in the ImageNet datasets been imported, then, the pre-trained layers undergone different transfer learning approaches which is Feature Extraction Approach and Fine-Tuning Approach[33].

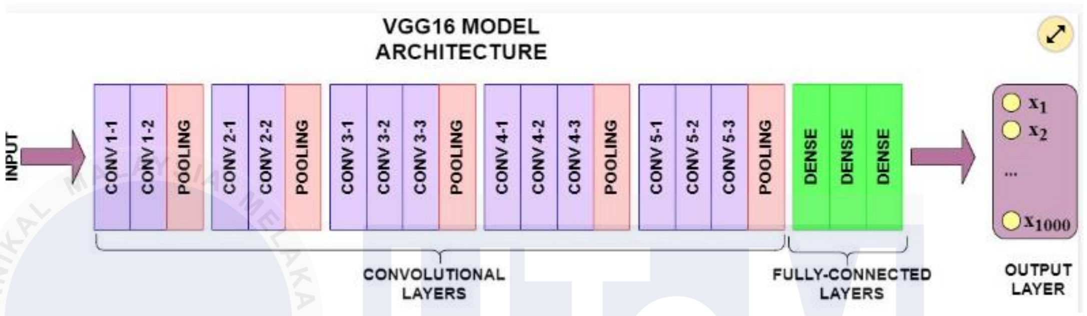


Figure 2. 11 The VGG16 Model has 16 Convolutional and Max Pooling layers, 3 Dense layers for the Fully Connected layer, and an output layer of 1,000 nodes.

Feature Extraction Approach has been implemented with the pre-trained model's architecture by creating a new dataset from the input images and importing the Convolutional and Pooling Layers instead of the Top portion of Fully Connected Layers. Then, the image dataset has been passed through the pre-trained layers for convolutional visual features extraction. The 3-Dimensional feature stack that is produced must be flattened in order for other machine learning classifiers to use it for prediction.

In order to flattened the feature, There are few extractors options for image extraction. The extractor option is stand-alone extractor where the pre-trained layers are used for image feature extraction. After that, a new dataset with no need for image processing would be created using the features that were extracted. The other feature extraction approach is Bootstrap Extractor where the pre-trained layers can be integrated by creating a new fully connected layer. This layer will be initialized with random weights, which will update via backpropagation during training.

Another approach that needs to be highlighted is Fine-Tuning Approach that involves a pre-trained model to be trained on a large dataset in terms for a general task of image classification. The aim of the approach is to enhance the model's performance on an additional, relevant task without having to train it from scratch. Fine-Tuning offers several advantages where it is able to increase the efficiency and significantly reduce the time and resource required. Furthermore, fine-tuning able to improve performance for a specific task, where a new dataset smaller than the original dataset used to train the pre-trained model. Fine – tuning approach can be entailed by selecting a pre-trained model that matches the nature of the task. Then, Once the pre-trained model has been chosen, its architecture needs to be adjusted to meet the demands of your particular assignment. This usually entails making changes to the model's upper layers. Next, freezing or unfreezing layers in the pre-trained model able to prevent from updating its weights during the fine-tuning process whereas unfreezing allows the matching layers to adjust during fine-tuning to the new data depending on the complexity of the task and yet the size of the dataset. Finally, after the architecture are being adjusted on the layers for freezing or unfreezing, The modified model are been trained to the specific dataset. Once the models are trained, fine-tuning will be applied for adjustment purposes on certain parameters that need to be iterated.

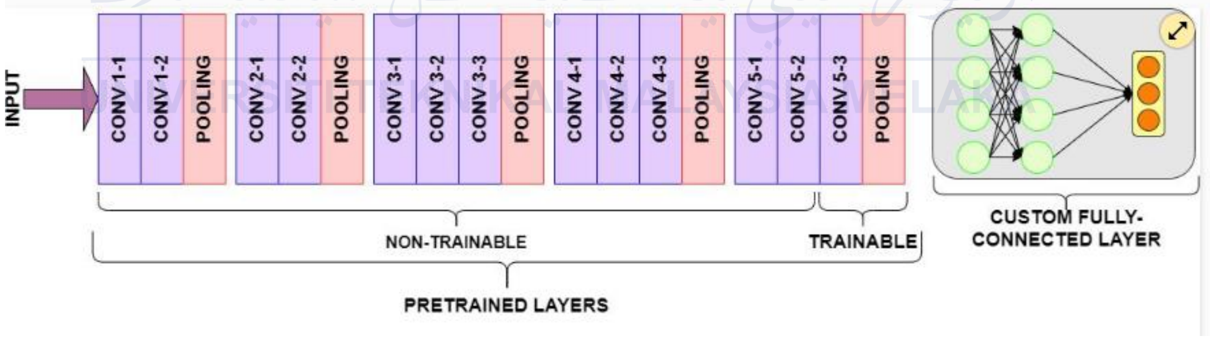


Figure 2. 12 Fine-Tuning Approach

2.7 Image Classification System involving Support Vector Machine (SVM)

Support Vector Machines (SVM) has evolved as a successful paradigm towards classification where it was first proposed by Vapnik and then the machine learning research community has shown a great deal of interest[34]. Support vector machines, or SVMs, can typically perform better than other data categorization techniques in terms of classification accuracy which will maximize the geometric margin and minimize the empirical classification error at the same time. It is predicated on the idea of locating a hyperplane in a high-dimensional space that can divide data points of several types. To improve classification performance, the distance between the support vectors and the hyperplane is maximized. The support vectors are the data points that are closest to the hyperplane.

Moreover, SVM can be stated as classification method for Image classification which has ben clarified by Jiang et al[35]. For two-class data, SVM looks for the best classification hyperplane in a space that is linearly separable. This method makes it possible to analyze the nonlinearity of samples using linear algorithms and finds the best classification hyperplane in this feature space. In the case of linear inseparability, it adds the slack variable for analysis and maps the samples in the low-dimensional input space into the high-dimensional attribute space to make it linearly separable by means of nonlinear mapping. Next, using SRM, it builds the best classification hyperplane in the attribute space to globally optimize the classifier and ensure that the sample space's overall predicted risk satisfies a certain upper bound at a given probability.

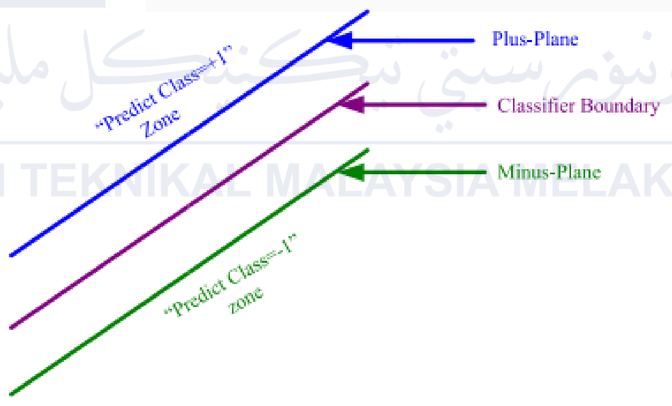


Figure 2. 13 Linear SVM

To achieve classification, SVM uses kernel functions to transfer nonlinearly separable input data from the low-dimensional space into the high-dimensional space. It then locates the best classification hyperplane in high-dimensional space once during the training purposes of SVM classifier. Usually, the SVM has a kernel function and constructs an image classifier with "one-to-one" classification combination that produces high training speed and prevent inseparable problems[36].

2.8 Image Classification System involving KNN method.

K Nearest Neighbor or known as KNN is a type of an algorithm that is based on regression and classification where the algorithm has the benefits of being simple to construct and having a quick training time whereas has the advantages of easy implementation and fast training speed. The fundamental idea behind KNN is to predict the label of a data point by looking at the 'k' nearest labeled data points and taking a majority vote for classification or an average for regression. The KNN algorithm is an instance-based, non-learning algorithm that is very straightforward and effective[37].

The sample data set in KNN algorithm are designed by corresponds each data to a label that corresponds one-to-one between data and its category which been described by Xiao et al.[38]. The KNN algorithm simply retains the whole training dataset during the training phase where the entire dataset is kept in memory in its entirely. The method is used to forecast new, unseen data items during the prediction phase where in the data input, The algorithm must identify its neighbors given a new data point for which we wish to estimate the class label for classification. The distance is determined by the separation between each data point in the training set and the input data point depending on the common distance metrics.

Then, the K data points are identified with the smallest distances to the input data point. The distance is sorted by the size, starting with the closes K values. The category with the highest frequency can be designated as a new data category by counting the number of occurrences of each kind in the initial K points, it is possible to forecast the category of the new data. If the KNN technique is applied to smaller size samples, it can easily result in incorrect classification. An imbalance in the sample can lead to the preponderance of large size samples in the sample's k neighbors.

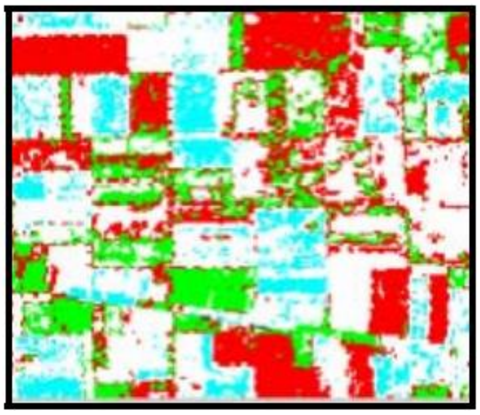


Figure 2. 14 Image classified using KNN algorithm[39]

2.9 Image classification system using Artificial Neural Network

Artificial Neural Networks employ artificial neurons to process data. Artificial neurons exchange information with one another to eventually build an activation function, a brain-like structure that makes decisions[40]. Nevertheless, the artificial neural network (ANN) is a parallel distributed processor[41] with an innate propensity to store experiential knowledge. They are able to offer appropriate remedies for issues, which are typically typified by high dimensionality noisy, complex, imprecise, inaccurate, or error-prone sensor data, non-linear relationships, and the absence of an explicit mathematical solution or algorithm. Neural networks have the advantage of allowing for the construction of a system model from the given data. Neural networks classify images by first extracting texture features and then using the backpropagation technique.

The architecture of Artificial Neural Network is characterized with feature extraction algorithm that the way that grey levels are distributed spatially within a neighborhood defines its texture. The goal of texture categorization is to designate a one of the known texture classes with an unknown example image. Scalar numbers, empirical distributions, or discrete histograms are examples of textural features. Four textural characteristics are considered in the design variance, correlation, contrast, and angular second moment. A two-dimensional dependence and texture analysis matrix is taken into consideration to capture the spatial dependence of gray-level values, which contribute to the perception of texture[43].

2.10 Region-Based Convolutional Neural Networks(R-CNN)

R-CNN is a supervised machine learning model. The initial objective of R-CNN was to take an input image and produce a set of bounding boxes as output, where each of the bounding boxes contains an object and shows the category of the object. In the R-CNN, classification and localization will take place. In the beginning, R-CNN has the issue of selecting many regions because it used selective search to extract 2000 regions from the image and is called region proposals. Selective search will generate initial sub-segmentation and use a greedy algorithm to combine the similar regions into larger and then produce a final region proposal.bIt causes the R-CNN to be difficultly implemented in real-time because it takes 47seconds for each test image[44]. However, due to the usage of the CNN model, R-CNN gets the highest mAP (mean average precision) when it initially proposes as Figure 2-15.

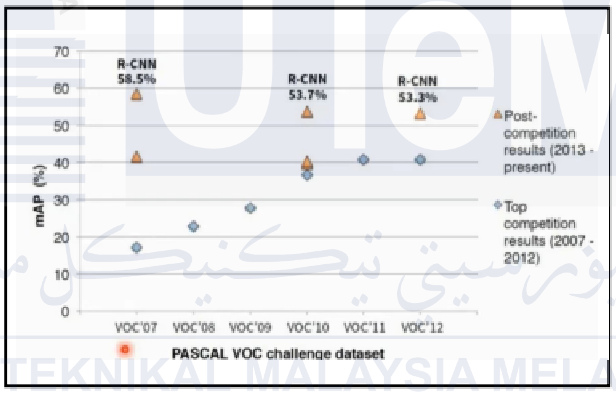


Figure 2. 15 mAP Comparison of RCNN and Another Model

Due to the problem of R-CNN, the newer version is keep proposed like Fast R-CNN and Faster R-CNN. The newer version decreases the test time significantly as Figure 2-16. For example, X. Zhao et al. proposed a pipeline classification system by using the faster-RCNN model[45]. X. Sun et al. used faster-R-CNN approach to detect the human face[46].

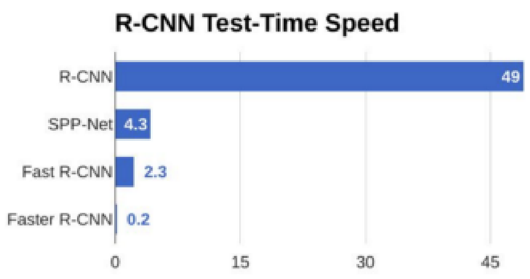


Figure 2. 16 : Speed Comparison of Different Version of R-CNN[47]

2.11 Summary

Based on the papers reviewed, seven articles from the previous five years have been chosen because they are most pertinent to the project. Most of the journals develop an Image Classification system using Deep Learning and Machine Learning due to the powerful specification. Most of the authors use CNN to train the system because CNN can automatically detect the important feature without any human supervision. Computational efficiency is one of the reasons to choose CNN. In a nutshell, most of the authors used CNN to detect and classify defect and non-defect images.

Table 2. 1 Summary of Literature Journal

Criteria	JOURNAL					
	Anton. A[26]	**Simonyan K**[32]	**Jiang H**[35]	**Wang**[38]	**Seetha M**[43]	**Srivastava S**[44]
Research Content	Proposed a Deep Learning Using Convolutional Neural Network (CNN) Method For Women's Skin Classification	Proposed a Large-Scale Image Recognition using Transfer Learning.	Proposed a study about kernel Techniques of SVM for Image Classification	Proposed an application of K-Nearest Neighbor (KNN) Algorithm for Human Action Recognition	Proposed an ANN method for Image classification	Comparative analysis of R-CNN
Method To Classify Defects	Deep Learning Convolutional Neural Network (CNN)	Transfer Learning Convolutional Neural Network (CNN)	Support Vector Machines	K-Nearest Neighbor	Artificial Neural Network	Region based CNN
Training model	Convolutional Neural Network	VGG 16 CNN	Convolutional Neural Network	Convolutional Neural Network	Artificial Neural Networks (neurons)	Lightweight Convolutional Neural Network

2.12 Gap from Literature Review

From the journal of the reviewed studies, a gap can be concluded from literature review where there are many journals related to image classification and recognition which focus on the types of architecture that need to be implemented during model training. Furthermore, an absolute method to be chosen in order to develop an accurate and precise image classification system. From these gaps, a defect or non-defect image classification system of C-SAM images using AI techniques will be developed.

Since the monitoring system is designed using classification based, the convolutional neural network method will be chosen and will be the priority. From the literature review, machine learning can be categorized into supervised and unsupervised. Supervised machine learning will be used. Supervised learning gets from literature review include SVM, CNN and KNN. CNN will be mainly used due to the image classification system. CNN architecture method will be used especially for modal selection, training, tuning and yet is proposed to take an input image and produce the classification performance as output. the method selected will be convolutional neural network because CNN is much faster when comparing with Region Proposal Network approaches (R-CNN) when classifying the object in real-time.

CHAPTER 3

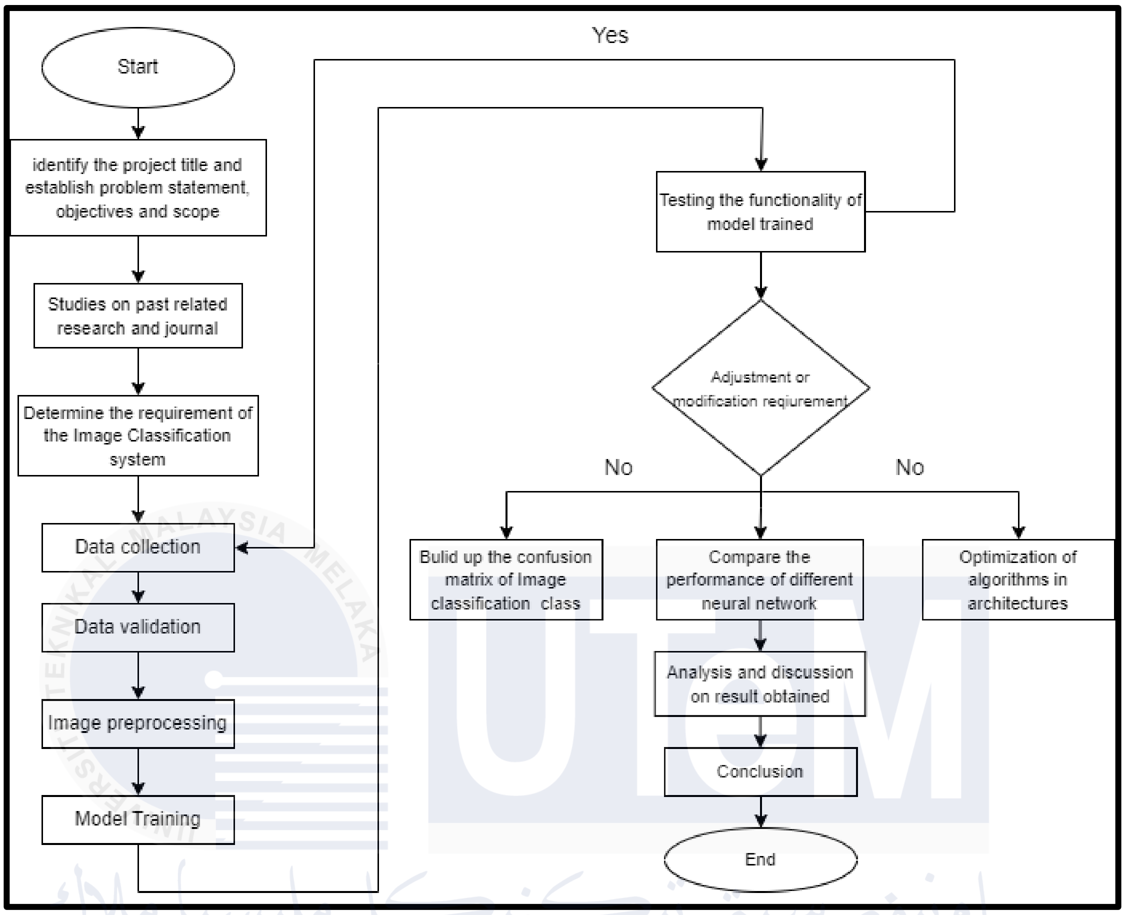
METHODOLOGY

3.1 Introduction

The chapter provides a significant and detailed overview of the methodology from the project that is being utilized to ensure the image classification and implementation of GUI has been effectively. The methods and techniques used in achieving the objectives final year project will be described. The goal of the conversation is to give participants a thorough grasp of the actions that must be taken before beginning the simulation. thorough explanations of the tactics and methods utilised in building the Defect or Non-defect Image Classification system will be looked into. Firstly, figure 3-1 will display the flowchart of the final year project overview and project process. Then, the simulation, which is the best Python software chosen will be described and the system illustration will be shown and explained. In addition, a Gantt chart will be used as a visual timetable to guarantee the project's timely completion and appropriate planning. The main aim is to successfully develop an Image Classification System for C-SAM Images by categorizing whether it's if a defect or non-defect and the Gantt chart will be essential to tracking the development of our final goal in benchmarking and maintaining project momentum, all in perfect harmony with our main goal.

3.2 Project Overview

The flowchart that illustrates the general direction of the investigation is presented in Figure 3.1. The success of the project can be ensured by using this flowchart as a reference. Based on the project's goals, two experiments were created.



Start
identify the project title and establish problem statement, objectives and scope
Studies on past related research and journal
Determine the requirement of the Image Classification system
Data collection
Data validation
Image preprocessing
Model Training
Yes
Testing the functionality of model trained
Adjustment or modification reqiurement
No
No
Bulid up the confusion matrix of Image classification class
Compare the performance of different neural network
Optimization of algorithms in architectures
Analysis and discussion on result obtained
Conclusion
End

Figure 3. 1 Project Overview Flowchart

3.3 Defect and Non-defect Image Classification System Flowchart

Figure 3.2 shows the flowchart that describes the operation of the Defect and Non-defect Image Classification System in carrying out the inspection task. This flowchart gives a clear idea of how the classification system works and ongoing process from the beginning till the end.

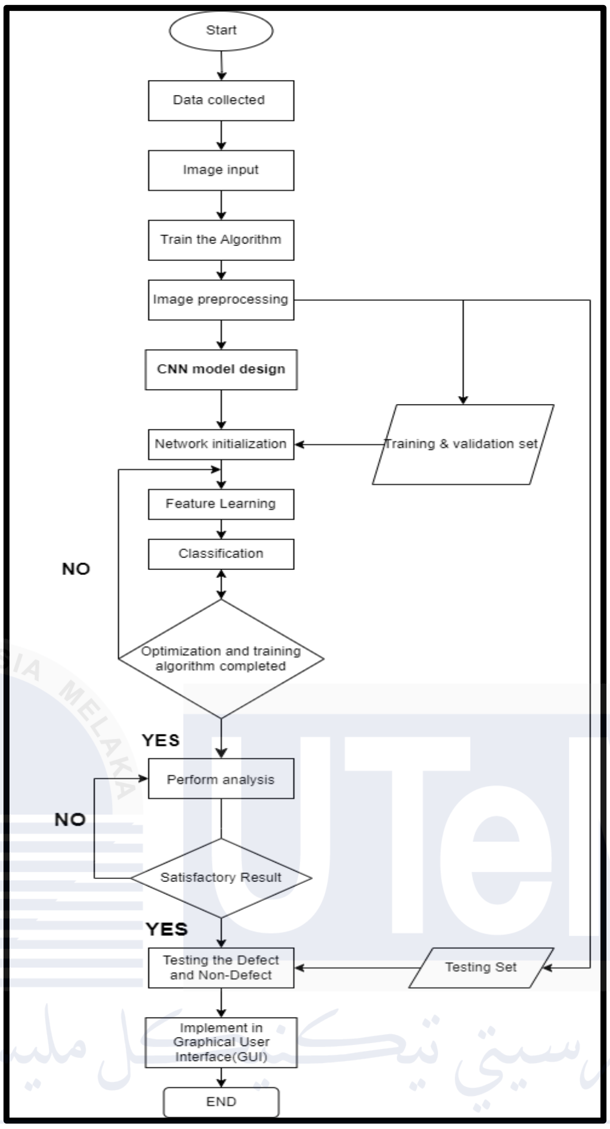


Figure 3. 2 Defect and Non-defect Image Classification System Flowchart

Figure 3.2 illustrates how the image classification system detects the defect into their categories. Firstly, the dataset of various defect and non-defect images will be labelled for data customization and separation. Then, the dataset will be trained will proper CNN architecture for pre-processing. After the images are being pre-processed, it will be fed into the trained model and the model will be tested and undergo classification performance to identify the accuracy performance. Lastly, once the classification performance reached the target, the trained model will appear in the Graphical User Interface (GUI) where the classification images will be shown as defect or non-defect.

3.4 Gantt Chart

Gantt Chart illustrates the activities scheduled for a project to be carried out smoothly so that the expected completion time can be met. Table 3-1 show the Gantt Chart for Defect and Non-defect Image Classification System using artificial intelligence.

Table 3. 1 Gantt Chart for FYP 1"

Task	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.0 Project Briefing	■	■	■	■	■	■	■	■	■					
2.0 Submission of project title				■	■	■	■	■	■	■	■			
3.0 Determine project motivation, objectives, and scope				■	■	■	■	■	■	■	■			
4.0 Research paper review					■	■	■	■	■	■	■	■	■	
5.0 Methodology													■	
6.0 Draft report submission												■	■	
7.0 Attend FYP 1 Seminar														■
8.0 Finalise Progress Report														■
9.0 Progress report submission														■

Table 3. 2 Gantt Chart for FYP 2"

Task	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.0 Simulation build up	■	■	■	■	■	■	■	■	■					
2.0 Experimental work				■	■	■	■	■	■	■	■			
3.0 Report writing				■	■	■	■	■	■	■	■			
4.0 Draft report submission					■	■	■	■	■	■	■	■	■	
5.0 Attend FYP 2 Seminar													■	
6.0 Finalise Report												■	■	
7.0 Report submission														■

3.5 Software Components and Applications

In this part, the software components used in the development of the Defect and Non-defect Image Classification System is described.

3.5.1 Tensorflow

AI technologies utilize extensive datasets in conjunction with intelligent algorithms, enabling the software to recognize patterns and acquire knowledge. Depending on the employed algorithm, the AI model can adjust to novel inputs, facilitating independent learning and growth without constant intervention from its creator, guided by the input it receives. The AI program can analyze the data to find patterns and connections between different points.

The Google Brain team created the open-source machine learning library TensorFlow. It is extensively employed in the development and training of diverse machine learning models, with a focus on deep learning. TensorFlow offers an extensive collection of tools and features that make it appropriate for a variety of uses, ranging from straightforward linear regression to intricate neural network structures.

TensorFlow is particularly well-suited for building and training neural networks. It includes high-level APIs like Keras that simplify the process of defining and training neural network models. TensorFlow also provides flexibility for control over the model architecture. Moreover, Tensorflow uses a computational graph to represent mathematical operations. In the graph, operations are represented by nodes, and the data flow between them is shown by edges.This allows for efficient parallel execution of computations and optimization.

The main reason to choose Tensorflow as the software application is because it integrates well with other popular technologies and tools commonly used in the machine learning ecosystem. This includes integration with libraries like NumPy and pandas for data manipulation, as well as compatibility with specialized hardware such as GPUs for accelerated model training.

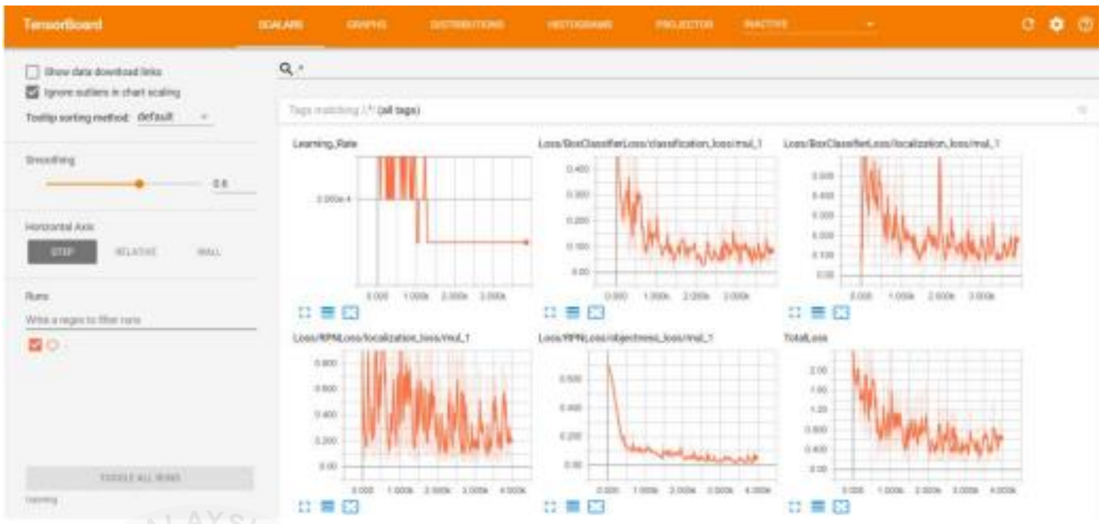


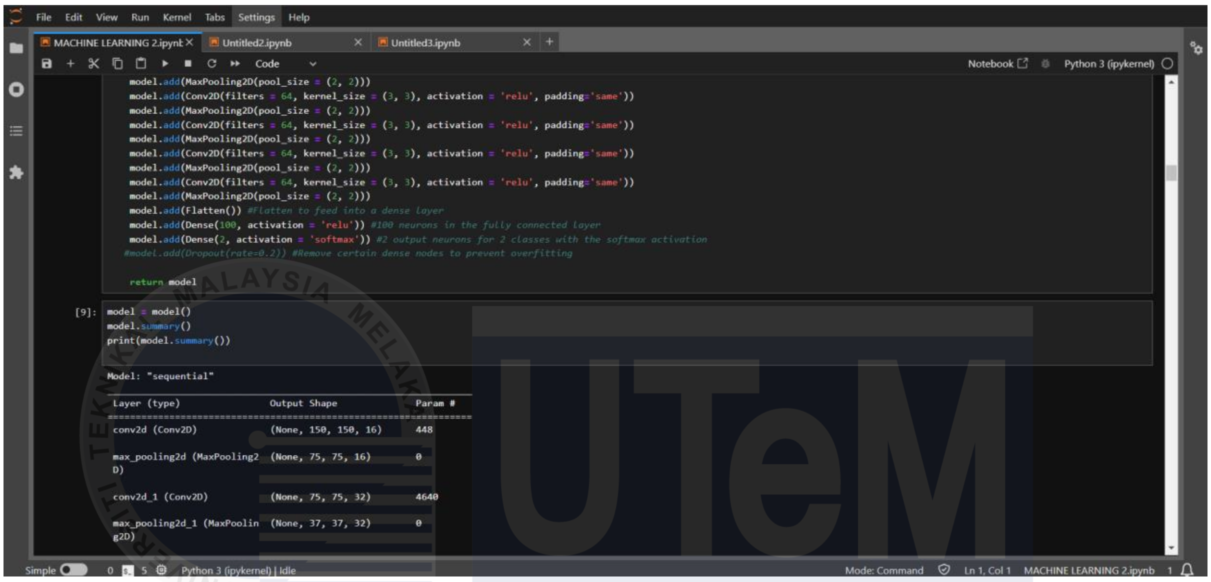
Figure 3. 3 Tensorboard for Tracking Learning Process

3.5.2 Jupyter Notebook

Jupyter Notebook is an open-source web tool that combines narrative prose, equations, live code, and visuals into documents that can be shared. This robust tool is adaptable for a variety of data science activities, supporting interactive computing and a broad range of programming languages, such as Python. One of Jupyter Notebooks' key features is that it can run code cells interactively, integrate rich media and visualizations right into the notebook, and write equations and descriptive prose in Markdown. Version management and sharing are made easier by the JSON format in which these notebooks store their documents.

Moreover, Jupyter Notebook has the functionality of facilitating iterative development and debugging by allowing users to execute code cell by cell. Because they support inline plots and graphs, which make data analysis more understandable, they are perfect for data visualization. Moreover, they provide comprehensive documentation features, which simplify the process of integrating code, visual outputs, and explanations into a single document. The reproducibility and clarity of data science initiatives are improved by this feature.

For a defect or non-defect image classification system, Jupyter notebooks come in very handy. From loading and preparing picture data to carrying out exploratory data analysis (EDA), they optimize the entire workflow. Within the notebook environment, users can create and train machine learning models, such as convolutional neural networks (CNNs). The main reason to choose Jupyter Notebooks as the python IDE because it's easier to assess model performance using a variety of metrics and see classification results and images with projected labels. The implementation of an interactive environment guarantees comprehensive documentation and easy reproducibility of each stage in the machine learning process, hence improving transparency and collaboration in image categorization projects. Jupyter Notebook is very efficient on training the model dataset even though running through PC.



```
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Flatten()) #Flatten to feed into a dense layer
    model.add(Dense(100, activation = 'relu')) #100 neurons in the fully connected layer
    model.add(Dense(2, activation = 'softmax')) #2 output neurons for 2 classes with the softmax activation
   #model.add(Dropout(rate=0.2)) #Remove certain dense nodes to prevent overfitting

    return model
```

```
[9]: model = model()
     model.summary()
     print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 16)      448

 max_pooling2d (MaxPooling2  (None, 75, 75, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 75, 75, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 37, 37, 32)        0
 g2D)
```

Figure 3. 4 Image Classification Model Training through Jupyter Notebook

3.5.3 Taipy (Web Application)

Taipy is a well-liked Python package for building interactive web applications for data science and machine learning tasks. It enables you to build web applications without requiring any prior understanding of web development languages like HTML, CSS, or JavaScript by using straightforward Python scripts. Taipy's primary characteristics are its simplicity, user-friendliness, and quick prototyping capabilities.

Taipy has the ability of enhancing the performance with caching control of graphical events, optimizing rendering by selectively updating graphical components only upon interaction. The main reason for choosing Taipy is because of the relative data performance in terms of data customization and scalability and thus, the time taken for launching the web GUI or web-based design is relatively fast compared to other interactive web applications.

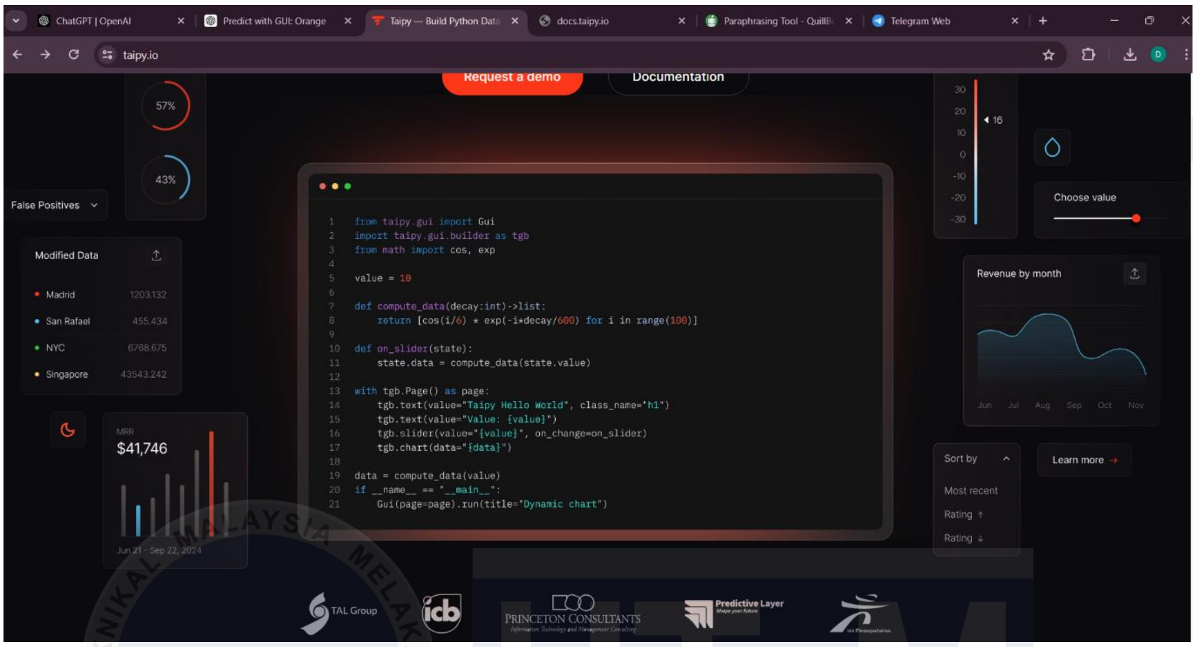
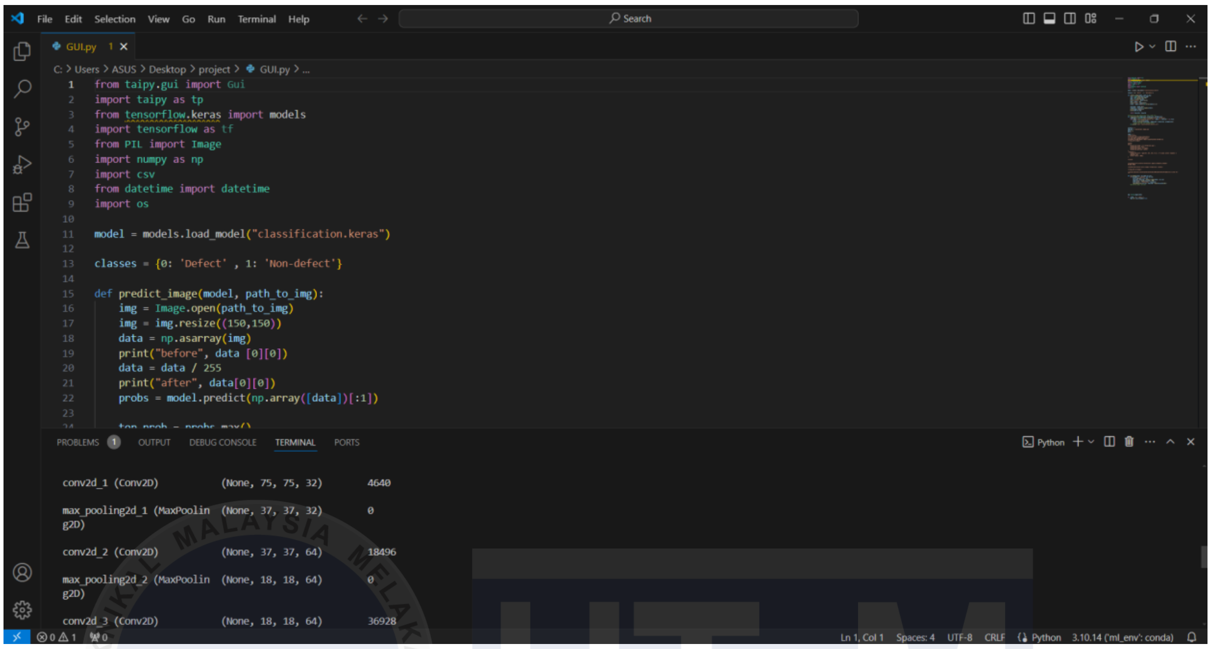

```python
from taipy.gui import Gui
import taipy.gui.builder as tgb
from math import cos, exp

value = 10

def compute_data(decay:int)->list:
    return [cos(i/6) * exp(-i*decay/600) for i in range(100)]

def on_slider(state):
    state.data = compute_data(state.value)

with tgb.Page() as page:
    tgb.text(value="Taipy Hello World", class_name="h1")
    tgb.text(value="Value: {value}")
    tgb.slider(value="{value}", on_change=on_slider)
    tgb.chart(data="{data}")

data = compute_data(value)
if __name__ == "__main__":
    Gui(page=page).run(title="Dynamic chart")
```

Figure 3. 5 Taipy Web Application

3.5.4 Microsoft Visual Code

Microsoft Visual Code is a flexible and powerful source code editor that works with a variety of operating systems, including Windows, macOS, and Linux. VS Code is a free and open-source application that has become quite popular among developers because of its powerful features and adaptability. With integrated syntax highlighting, IntelliSense for code completion, and debugging features for multiple programming languages like JavaScript, Python, Java, and C++, it provides broad language compatibility.

VS Code's user-friendly design offers a smooth coding experience that blends the functionality of an Integrated Development Environment (IDE) with the ease of use of a text editor. Its high degree of customisation also lets users tailor settings, themes, and key bindings to suit their preferred workflows. With the help of a vibrant community and frequent updates from Microsoft, Visual Studio Code keeps improving and holding its place as a top option for developers working with a variety of platforms and programming languages.


File Edit Selection View Go Run Terminal Help
Search
GUI.py 1
C: > Users > ASUS > Desktop > project > GUI.py > ...

```python
from taipy.gui import Gui
import taipy as tp
from tensorflow.keras import models
import tensorflow as tf
from PIL import Image
import numpy as np
import csv
from datetime import datetime
import os

model = models.load_model("classification.keras")

classes = {0: 'Defect' , 1: 'Non-defect'}

def predict_image(model, path_to_img):
    img = Image.open(path_to_img)
    img = img.resize((150,150))
    data = np.asarray(img)
    print("before", data [0][0])
    data = data / 255
    print("after", data[0][0])
    probs = model.predict(np.array([data])[:1])

    top_prob = probs.max()
```

```
conv2d_1 (Conv2D)            (None, 75, 75, 32)        4640

max_pooling2d_1 (MaxPoolin  (None, 37, 37, 32)        0
g2D)

conv2d_2 (Conv2D)            (None, 37, 37, 64)        18496

max_pooling2d_2 (MaxPoolin  (None, 18, 18, 64)        0
g2D)

conv2d_3 (Conv2D)            (None, 18, 18, 64)        36928
```

⊗ 0 ⚠ 1 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.14 ('ml_env': conda)

Figure 3. 6 Microsoft Visual Code IDE

3.6 System Configuration

3.6.1 Data Selection

A collection of defect and non-defect images that were received by the working staffs in Onsemi Sdn.Bhd, were being utilized in the initial phase of the research, which is the selection of data. Throughout the course of the research, the dataset, which is referred to as "C-SAM images", gets dispersed for several reasons during the research project. Specifically, 70% of the "C-SAM images" dataset is marked for training, while 20% is assigned for validation, the remaining 9.8% is set aside for experimentation.

3.6.2 Data Labelling

Data annotation is the next step after data gathering. Data annotation is the process of marking the location of each defect item in each image using TensorFlow labelling tool. Every picture is added to TensorFlow and given a unique label, such as depicted in the figure that follows. Every label has a name that matches the corresponding defect item. For example, the item 'defect' in the C-SAM image dataset is labelled as 'defect,' and the remaining elements are labelled similarly so that the model can detect them. Now that the annotation process has been finished, Tensorflow can use the dataset to train models and make predictions and also samples normalization.

3.6.3 Model Architecture (Convolutional Neural Network Deep Learning)

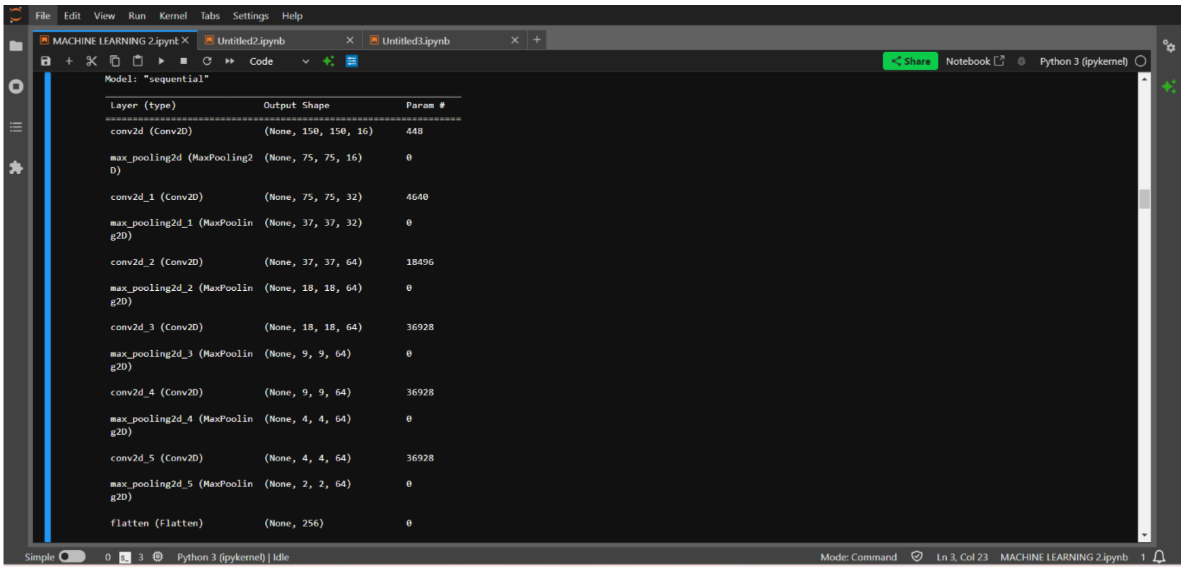
In this research, the utilized CNN-based model is trained using deep learning, a type of Convolutional Neural Network (CNN). Images serve as examples of inputs suitable for processing by CNNs, specialized neural networks designed for grid-structured inputs. CNNs are currently employed in various tasks, including photo classification, object localization, and facial recognition. CNN processes information through layered filters, with each layer capable of identifying diverse properties within the image. Neural networks consist of three distinct levels: the input layer, hidden layer, and output layer, each assigned specific functions. In the input layer, the total pixel count of the data is divided, facilitating further operations on the image data. The hidden layer, following the input layer in the layer hierarchy, receives the input values generated. Hidden layers, responsible for producing output by applying activation functions to weighted inputs, form a complex structure. The output layer, linked entirely, receives the output from hidden layers, flattens and distributes it, and classifies it into the required number of classes based on specifications. This section includes the sought-after Figure 3.7. The hidden layer would consist of Conventional layer, Max and Min layer,Pooling layer, Dense layer and Flatten layer.

The model architecture involves the input layer in which the model expects to have the shape of (150,150,3) where the spatial dimension is 150x150 and the RGB color channels are represented by 3. The images being sent to the neural network will have a resolution of 150 pixels wide by 150 pixels high. This establishes how big the input image will be overall when processed by the model. The images are commonly displayed using a combination of the three primary colours: red, green and blue (RGB). These three-color channels have intensity values associated with each pixel in the image. As a result, the input layer's '3' signifies that the model anticipates seeing three color channels in input photos.

The model architecture process involves into the Conv2D layers where these Convolutional layers responsible for extracting features from the input images. A predetermined number of filters which consist of (16, 32, 64) are applied to the input image by each Conv2D layer. Each filter learns to detect a particular pattern or feature in the input image. As the filters slide over the input image, they detect patterns like edges, textures, or shapes, depending on the patterns they were trained to recognize, and these filters multiply the input pixels and their surrounding pixels element-by-element. Feature maps, which represents the existence of characteristics or patterns in the input images, are the results of these layers.

Next layer is about the MaxPooling2D layers where these layers reduce the feature maps' spatial dimensions (width and height), which lowers the model's computational cost and prevents overfitting. A pooling technique called max pooling chooses the maximum element from a section of the feature map that is determined by the pooling kernel's size and thus controlling the number of parameters. It undergoes a downsampling process on the input feature maps. The most crucial data is preserved throughout this downsampling process, which shrinks the feature maps' spatial dimensions width and height. The functionality of MaxPooling also involving translation Invariance where it focuses on the prominent feature in the region.

Then, there's the last two bottom layers which are the Flatten and Dense layer. Flatten layer converts the 2D feature maps into a 1D vector that can be used as input into a fully connected neural network for classification. The multi-dimensional output from the preceding layers is flattened into a one-dimensional array by the Flatten layer. It accomplishes this by simply rearranging the tensor's elements into a single vector while preserving their original order. Meanwhile, on the dense layers, are known as fully connected layers that are in charge of learning patterns in the feature maps and making predictions. The first dense layer has 100 units, and the second dense layer has 2 units, indicating a binary classification task



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 16)      448

 max_pooling2d (MaxPooling2  (None, 75, 75, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 75, 75, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 37, 37, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 64)        18496

 max_pooling2d_2 (MaxPoolin  (None, 18, 18, 64)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 18, 18, 64)        36928

 max_pooling2d_3 (MaxPoolin  (None, 9, 9, 64)          0
 g2D)

 conv2d_4 (Conv2D)           (None, 9, 9, 64)          36928

 max_pooling2d_4 (MaxPoolin  (None, 4, 4, 64)          0
 g2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_5 (MaxPoolin  (None, 2, 2, 64)          0
 g2D)

 flatten (Flatten)           (None, 256)               0
```

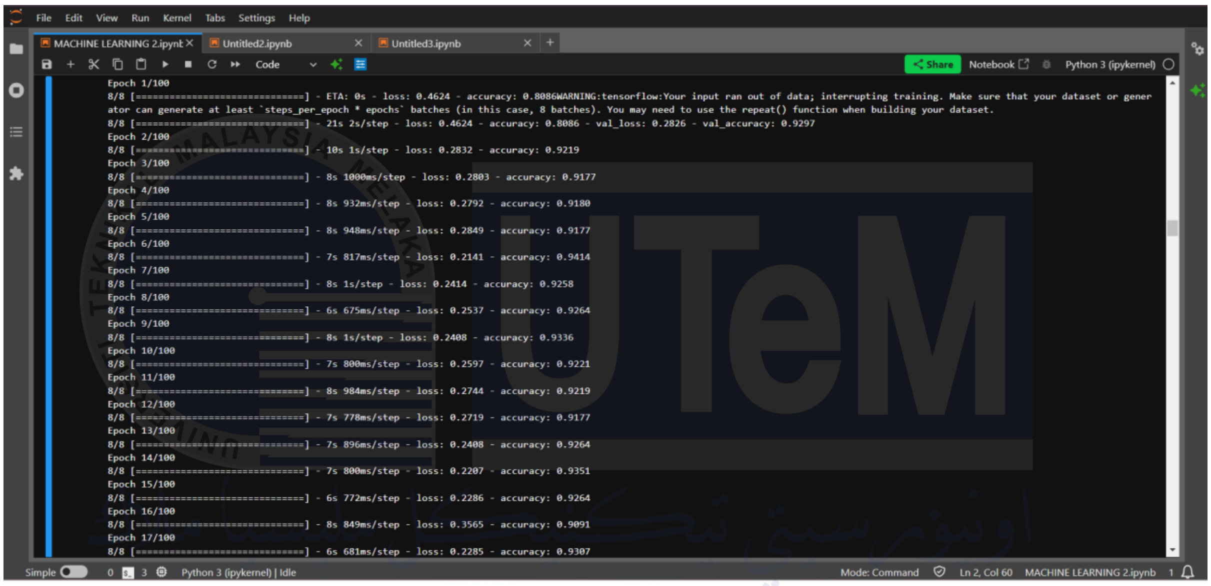
Figure 3. 7 Model sequential and layers of CNN

3.6.4 Model Compilation, Training and Validation

The model compilation, training and validation procedures become crucial steps once the Convolutional Neural Network (CNN) architecture is chosen for the defect and non-defect picture classification system. Convolutional layers are often used for feature extraction in the CNN architecture that is selected for image classification tasks, whereas fully connected layers are used for accurate classification. In order for the model to acquire the unique characteristics that define each class, it must be exposed to the labelled dataset, which includes both defective and non-defective photos, during the training phase. To minimize the selected loss function, the model iteratively modifies its internal parameters, such as weights and biases. Techniques for regularization can be used to stop overfitting. Before the model training been implemented, model compilation tooks place in order to be configured for training purposes with several important parameters to be setup which is called, adam, loss and metrics. Adam usually known as an optimization algorithm which used for training neural networks and adjusts learning rate during training while loss function is employed to quantify the discrepancy between the model's output and the actual target values. In this case, categorical crossentropy is chosen since predicting multi-class classification issues. Then, the accuracy metric is written in the coding to precisely measure the percentage of correct classified images.

Next, the model training plays an important part where the dataset will be trained completely with involved parameters in order to enhance the classification percentage achieve higher than expected and allows to predict the images precisely and accurately. The training dataset or which known as 'train_data' which consists of input images and its corresponding labels allows the dataset to specify the number of samples per gradient to be updated. The model processes each batch of 32 photos and then updates its weights. Then, epoch parameter come in place where 8 batches will be drawn in each epoch for training. The number of epochs defines how many times the complete dataset is transmitted forward and backward through the neural network. In the project, 100 epochs of training will be completed to ensure better classification percentage and performance.

The validation dataset is important for training at the same time. It functions as a separate set that the model does not see during training and is used to assess the model's generalizability and performance after each epoch. Recall, accuracy, precision, and F1 score are among the metrics that can be tracked on the validation set to help adjust hyperparameters and spot possible overfitting problems. Until the model performs satisfactorily on both the training and validation datasets, this iterative process of training and validation is continued. The model can be used in real-world applications after completing the training and validation stages successfully. In these applications, it can be highly accurate in classifying new occurrences of defective and non-defective products. The classification system's continuing dependability is further ensured by routine monitoring and possible retraining using current data.



```
Epoch 1/100
8/8 [==============================] - ETA: 0s - loss: 0.4624 - accuracy: 0.8086WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or gener
ator can generate at least `steps_per_epoch * epochs` batches (in this case, 8 batches). You may need to use the repeat() function when building your dataset.
8/8 [==============================] - 21s 2s/step - loss: 0.4624 - accuracy: 0.8086 - val_loss: 0.2826 - val_accuracy: 0.9297
Epoch 2/100
8/8 [==============================] - 10s 1s/step - loss: 0.2832 - accuracy: 0.9219
Epoch 3/100
8/8 [==============================] - 8s 1000ms/step - loss: 0.2803 - accuracy: 0.9177
Epoch 4/100
8/8 [==============================] - 8s 932ms/step - loss: 0.2792 - accuracy: 0.9180
Epoch 5/100
8/8 [==============================] - 8s 948ms/step - loss: 0.2849 - accuracy: 0.9177
Epoch 6/100
8/8 [==============================] - 7s 817ms/step - loss: 0.2141 - accuracy: 0.9414
Epoch 7/100
8/8 [==============================] - 8s 1s/step - loss: 0.2414 - accuracy: 0.9258
Epoch 8/100
8/8 [==============================] - 6s 675ms/step - loss: 0.2537 - accuracy: 0.9264
Epoch 9/100
8/8 [==============================] - 8s 1s/step - loss: 0.2408 - accuracy: 0.9336
Epoch 10/100
8/8 [==============================] - 7s 800ms/step - loss: 0.2597 - accuracy: 0.9221
Epoch 11/100
8/8 [==============================] - 8s 984ms/step - loss: 0.2744 - accuracy: 0.9219
Epoch 12/100
8/8 [==============================] - 7s 778ms/step - loss: 0.2719 - accuracy: 0.9177
Epoch 13/100
8/8 [==============================] - 7s 896ms/step - loss: 0.2408 - accuracy: 0.9264
Epoch 14/100
8/8 [==============================] - 7s 800ms/step - loss: 0.2207 - accuracy: 0.9351
Epoch 15/100
8/8 [==============================] - 6s 772ms/step - loss: 0.2286 - accuracy: 0.9264
Epoch 16/100
8/8 [==============================] - 8s 849ms/step - loss: 0.3565 - accuracy: 0.9091
Epoch 17/100
8/8 [==============================] - 6s 681ms/step - loss: 0.2285 - accuracy: 0.9307
```

Figure 3. 8 Model Training using Number of Epochs

3.6.5 Model Monitoring and Deployment

Ensuring the efficacy and dependability of an image classification system for defect and non-defect identification requires careful consideration of both model deployment and monitoring. Error analysis must be continuously monitored once the model has been trained and validated. This entails closely examining the model's misclassifications and inaccuracies throughout both training and actual use. To improve the model's overall performance, changes can be made to the training dataset, hyperparameters, or model architecture by comprehending and evaluating these errors.

Using a Graphical User Interface (GUI) to categorize photographs as output makes it easier for users to engage with the model and makes it easier to use. Individuals without considerable technical skills can utilize the system because of its user-friendly interface, which enables users to input photographs and receive predictions in real-time. Moreover, the GUI system allows for the recording and tracking of classification results. It saves the uploaded file name, along with the corresponding defect classification, in an Excel file named 'Classification_results.csv' which is saved under the project directory. This functionality facilitates comprehensive defect analysis, report generation, and ongoing quality control monitoring. It's important to note that the entire GUI system is developed using the Python programming language, in this case, Taipy was involved as a creative and interactive web design application for its faster and reliable performance.

Monitoring the model's performance after deployment is essential to sustaining its efficacy over time. Continuous monitoring entails evaluating the model's performance on fresh, untested data, spotting any drift or performance deterioration, and applying updates or retraining as necessary. Maintaining a regular monitoring of performance indicators guarantees that the model stays precise and in step with the changing properties of the input data. Figure 3.9 shows a Graphical User Interface (GUI) which is a web based design for classifying Defects and non defects images.

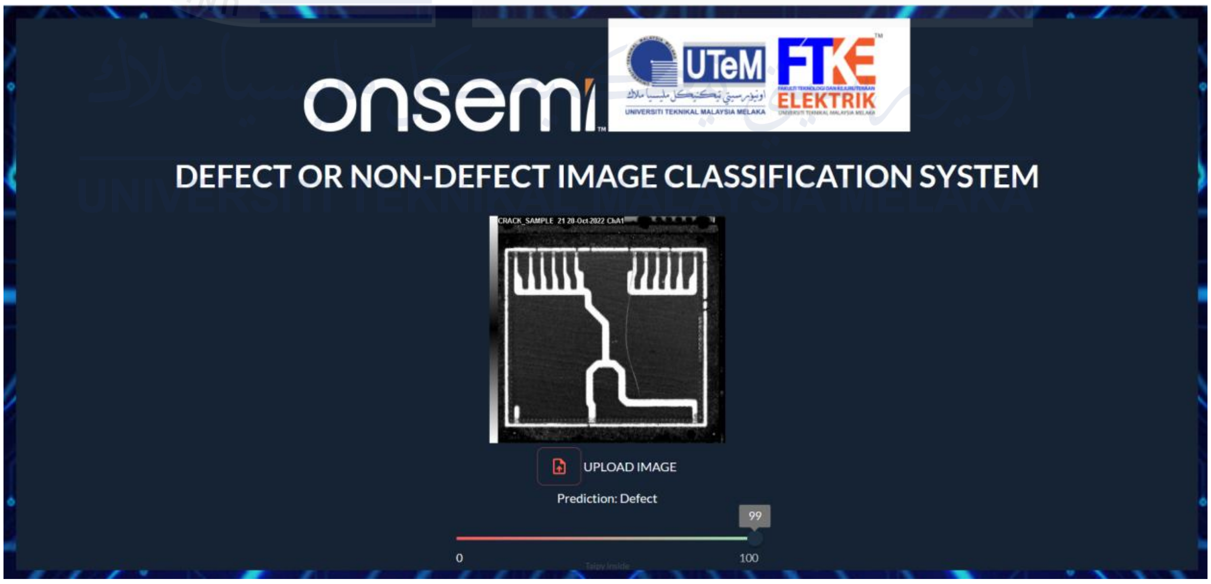


Figure 3. 9 Graphical User Interface (GUI)

3.7 Analyzing Models and Gathering Data

3.7.1 Confusion Matrix of Models

Quantitative methods to assess the detection of objects a popular evaluation statistic for assessing object classification models' effectiveness is confusion matrix of each neural network to analyze the performance of the different types of neural networks. A confusion matrix is a table used in machine learning to assess the performance of a classification model. It provides a detailed breakdown of the model's predictions, comparing them to the actual classes in the dataset.

Table 3. 3 Data for Confusion Matrix of Models

DEFECT/ NON-DEFECT			
		PREDICTED	
		NO	YES
ACTUAL	NO	TN	FP
	YES	FN	TP

- True-positive (TP): Prediction is +ve and defect/non-defect is present.
- True-negative (TN): Prediction is -ve and defect/non-defect is absent.
- False-positive (FP): Prediction is +ve but defect/non-defect is absent.
- False-negative (FN): Prediction is -ve but defect/non-defect is present.

3.7.2 Performance Testing With Different Neural Network

The performance of each neural network will be tested and compared. The test time will be tested, and accuracy, precision and recall will be calculated. Accuracy is a parameter to show how often the neural network detects and classifies correctly. Precision shows how often the neural network predicts correctly. Recall can be said as sensitivity to show how often the neural network predicts yes when the object occurs.

$$\boldsymbol{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.1)\ [48]$$

$$\boldsymbol{Precision} = \frac{TP}{TP+FP} \quad (3.2)\ [48]$$

$$\boldsymbol{Recall} = \frac{TP}{TP+FN} \quad (3.3)\ [48]$$

Table 3. 4 Data of Performance Testing

	DEFECTS				NON-DEFECTS			
TYPE OF NEURAL NETWORK	FPS	ACCURACY	PRECISION	RECALL	FPS	ACCURACY	PRECISION	RECALL
CONVOLUTIONAL NEURAL NETWORK								

3.8 Ethics and Safety of the model

In the field of artificial intelligence and machine learning, standardization is essential. One well-known international standardization organization is the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), that develops and publishes standards for a range of sectors and fields, including artificial intelligence and machine learning. Table 3.4 lists a few noteworthy projects.

Table 3. 5 ISO/IEC Relate to AI and ML

ISO/IEC JTC 1/SC 42 [48]	This branch of the AI-focused ISO/IEC Joint Technical Committee 1 (JTC 1) is responsible for this. AI-related guidelines, including like those developed by SC 42 for AI systems and their uses. They concentrate on language, data, reference designs, AI dependability, and moral dilemmas.
ISO/IEC 40500 [49]	This specification, commonly referred to as the "Web Content Accessibility Guidelines," focuses on web accessibility. (WCAG) 2.0" is significant even if it isn't a specific AI standard because AI systems and apps need to follow accessibility guidelines in order to preserve inclusiveness.

3.9 Summary

Essentially, the main purpose of the project is to construct a Defect and Non-defect Image Classification System of C-SAM images using Artificial Intelligence of Machine learning and deep learning. The project involves a methodical approach to data collection, organization, annotation, CNN-based model training using deep learning and machine learning, by classifying images. The Gantt diagram ensures that the project's prompt finish, with a focus on system integrity and safety. The software configuration includes installing necessary Python libraries and dependencies and gives priority to Tensorflow for development. The main objective of using the Tensorflow datasets in this research is to automatically develop an image classification system that classifies the C-SAM images' flaws and non-defects.

CHAPTER 4:

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, the simulation results on the network trained are presented and discussed in detail. Besides, the expected results for all the experiments are described by assuming the project works in an ideal condition.

4.2 Project Overview

Tensorflow worked well in training the model and saved through Keras file while the Graphical User Interface (GUI) was designed using Taipy as web design application and Microsoft Visual Studio as IDE. This chapter will mostly focus on the models' classification performance and including GUI performance on showing the Defect and Non-defect images.

4.2.1 Dataset Preparation

All the data is collected in png form and then converted to jpg format. All the images are being separated for system training purposes which include Test and Train. The total number of images prepared is 648 and split with 80% (518) for the train data and 20% (130) for the test data.

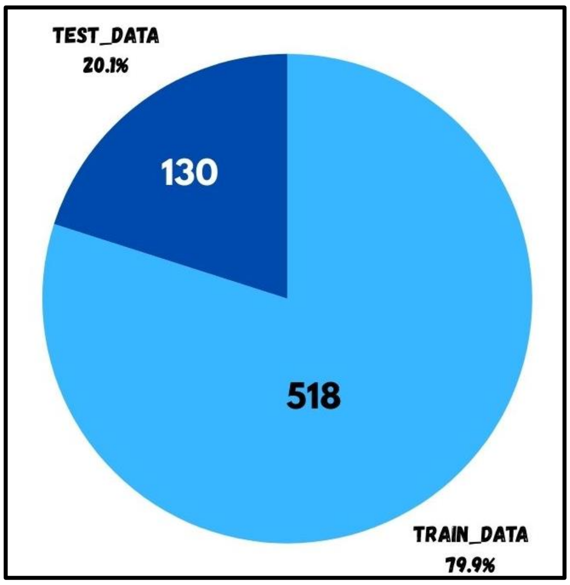
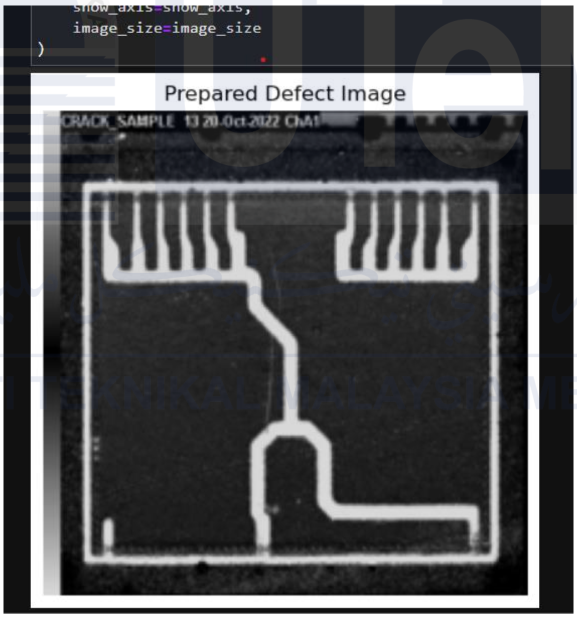


Figure 4. 1 Dataset Splitting

4.2.2 Model initialization and customization

In this project, the CNN method is employed for network training, while TensorFlow are used to train the model by using Jupyter Notebook and Google Colab. With the help of the powerful hardware accelerators known as Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs), users of Jupyter Notebook can accelerate calculations and prepare them for machine learning models at no cost. The model was initialized and customized through programming where the datasets are modified according to the scales, shear range, zoom range and image height and width. The image size was set and programmed at a measurement of (150,150) to make sure the image is clearer for prediction.



```
    show_axis=show_axis,
    image_size=image_size
)
```

Figure 4. 2 Image Labelling and customization

4.2.3 Machine Learning Model Training Monitoring

The images are feeded into the model for training purpose. Jupyter Notebook will involve tensorflow module which used to monitor the training process to check the training progress and to ensure the training is running properly. It is anticipated that 100 epochs in total will be used throughout training. Figure below shows the graph of 'Accuracy over epochs' and 'Loss over epochs'. It will provide a clearer picture of how well the model is been trained. These graphs allow us to see the precision and recall values that correlate to each epoch and help us determine which period has the highest confusion matrix value in relation to other epochs. Moreover, this data is essential for future research since it informs choices on model optimisation and finetuning.

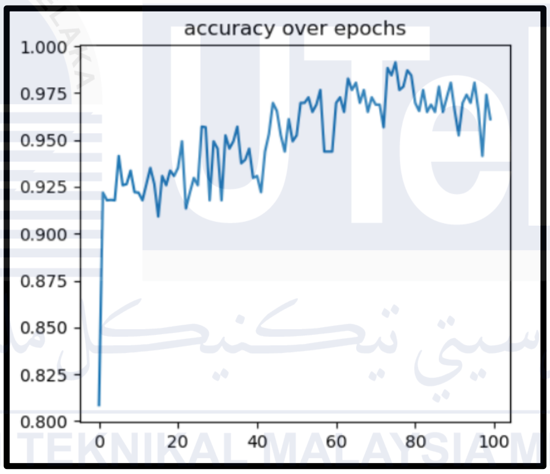


Figure 4. 3 Accuracy over Epochs

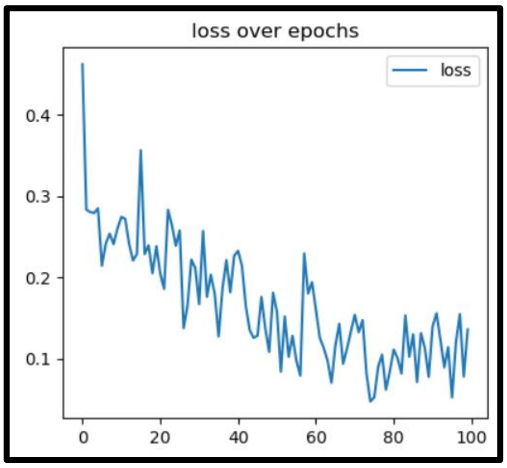


Figure 4. 4 Loss over Epochs

4.2.4 Graphical User Interface Outcome

After the model has been trained after 100 number of epochs, it is saved under a Keras file because it’s essential later to be uploaded into the GUI programming on Taipy which will be able to support the file with compatible packages involved. The GUI was design with different feature and options to make sure the GUI is interactive and convenient to search about the Defect and Non-defect images. The real-life performance on the GUI was good after testing was done. The logo and search bar were added to upload the images on the GUI. Once the images are uploaded, the prediction result will be shown on the GUI straightly without further due. This could include displaying defect labels, highlighting defect regions on the C-SAM image, or providing statistical summaries. Also, it enables users to analyze and export the results for further investigation or reporting. This involves generating data of defects in excel.

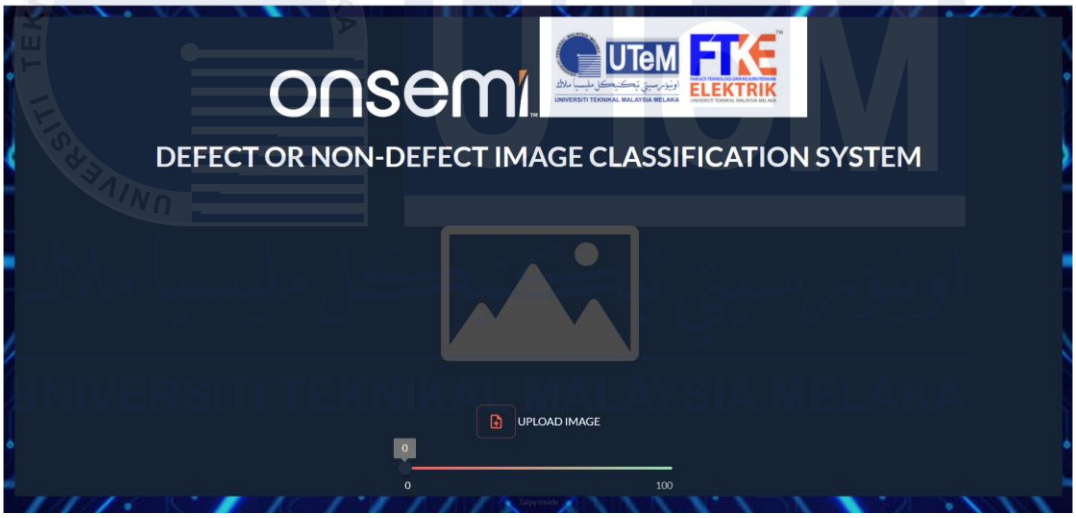


Figure 4. 5 Graphical User Interface

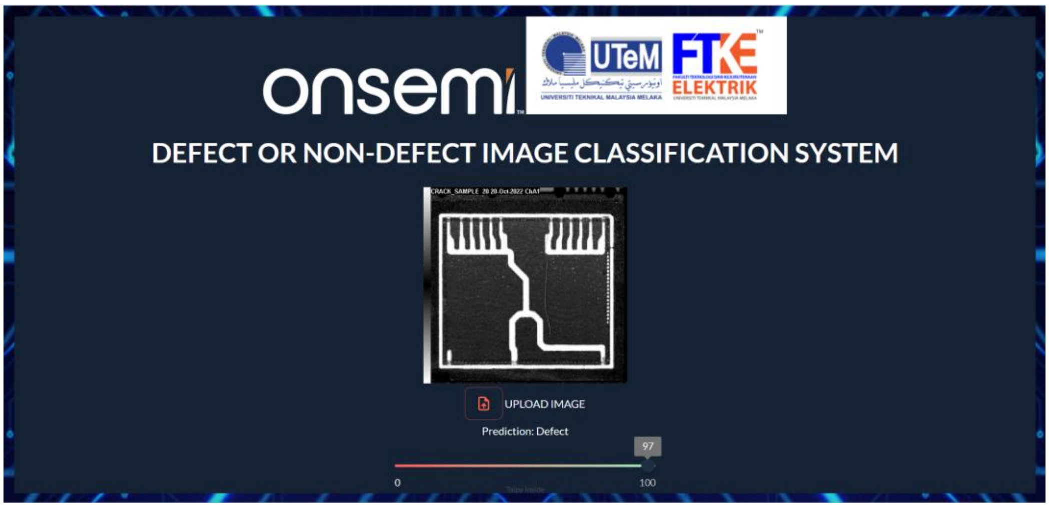
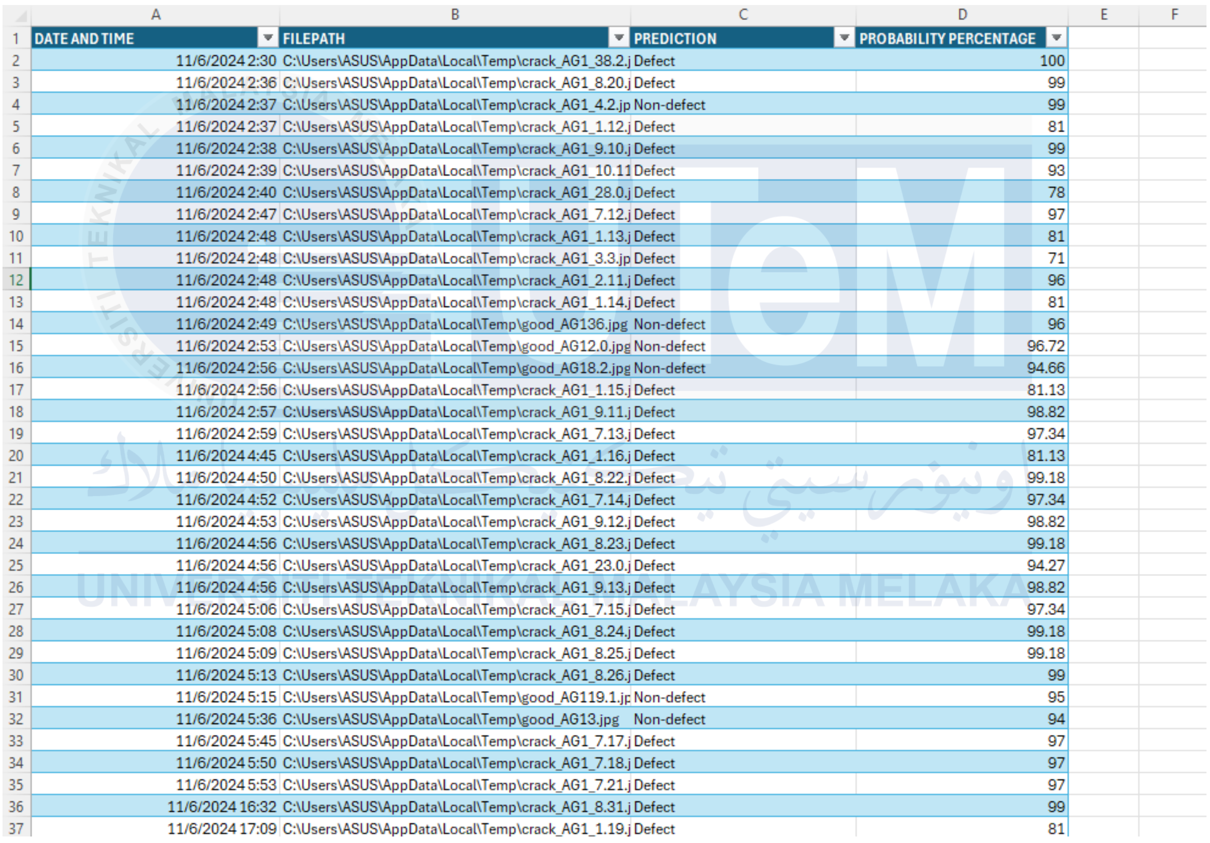


Figure 4. 6 Prediction Result On the GUI

4.2.5 Prediction Result through CSV file

A CSV excel file was created in order save and record the Defect and Non-defect images and that have been uploaded through the GUI and track the activities including the Production time progress and classification results whenever using GUI. The CSV will automatically pop up once the Image is selected and uploaded on the GUI and thus record the Activities of the scenario.



	A	B	C	D	E	F
1	DATE AND TIME	FILEPATH	PREDICTION	PROBABILITY PERCENTAGE		
2	11/6/2024 2:30	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_38.2.j	Defect	100		
3	11/6/2024 2:36	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.20.j	Defect	99		
4	11/6/2024 2:37	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_4.2.jp	Non-defect	99		
5	11/6/2024 2:37	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_1.12.j	Defect	81		
6	11/6/2024 2:38	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_9.10.j	Defect	99		
7	11/6/2024 2:39	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_10.11	Defect	93		
8	11/6/2024 2:40	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_28.0.j	Defect	78		
9	11/6/2024 2:47	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.12.j	Defect	97		
10	11/6/2024 2:48	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_1.13.j	Defect	81		
11	11/6/2024 2:48	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_3.3.jp	Defect	71		
12	11/6/2024 2:48	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_2.11.j	Defect	96		
13	11/6/2024 2:48	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_1.14.j	Defect	81		
14	11/6/2024 2:49	C:\Users\ASUS\AppData\Local\Temp\good_AG136.jpg	Non-defect	96		
15	11/6/2024 2:53	C:\Users\ASUS\AppData\Local\Temp\good_AG12.0.jpg	Non-defect	96.72		
16	11/6/2024 2:56	C:\Users\ASUS\AppData\Local\Temp\good_AG18.2.jpg	Non-defect	94.66		
17	11/6/2024 2:56	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_1.15.j	Defect	81.13		
18	11/6/2024 2:57	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_9.11.j	Defect	98.82		
19	11/6/2024 2:59	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.13.j	Defect	97.34		
20	11/6/2024 4:45	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_1.16.j	Defect	81.13		
21	11/6/2024 4:50	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.22.j	Defect	99.18		
22	11/6/2024 4:52	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.14.j	Defect	97.34		
23	11/6/2024 4:53	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_9.12.j	Defect	98.82		
24	11/6/2024 4:56	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.23.j	Defect	99.18		
25	11/6/2024 4:56	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_23.0.j	Defect	94.27		
26	11/6/2024 4:56	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_9.13.j	Defect	98.82		
27	11/6/2024 5:06	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.15.j	Defect	97.34		
28	11/6/2024 5:08	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.24.j	Defect	99.18		
29	11/6/2024 5:09	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.25.j	Defect	99.18		
30	11/6/2024 5:13	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.26.j	Defect	99		
31	11/6/2024 5:15	C:\Users\ASUS\AppData\Local\Temp\good_AG119.1.jp	Non-defect	95		
32	11/6/2024 5:36	C:\Users\ASUS\AppData\Local\Temp\good_AG13.jpg	Non-defect	94		
33	11/6/2024 5:45	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.17.j	Defect	97		
34	11/6/2024 5:50	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.18.j	Defect	97		
35	11/6/2024 5:53	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_7.21.j	Defect	97		
36	11/6/2024 16:32	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_8.31.j	Defect	99		
37	11/6/2024 17:09	C:\Users\ASUS\AppData\Local\Temp\crack_AG1_1.19.j	Defect	81		

Figure 4. 7 Classification Results on CSV file

4.3 Experimental Result

The neural network selected includes Convolutional Neural Network (CNN). All the experiments are carried out by using models with the TensorFlow form.

4.3.1 Experiment 1: Confusion Matrix of Models

128 of Defect and Non-defect images is selected randomly to test models

Table 4- 1 Confusion Matrix of Defect and Non-Defect

DEFECT/ NON-DEFECT			
		PREDICTED	
		NO	YES
ACTUAL	NO	9	31
	YES	0	88

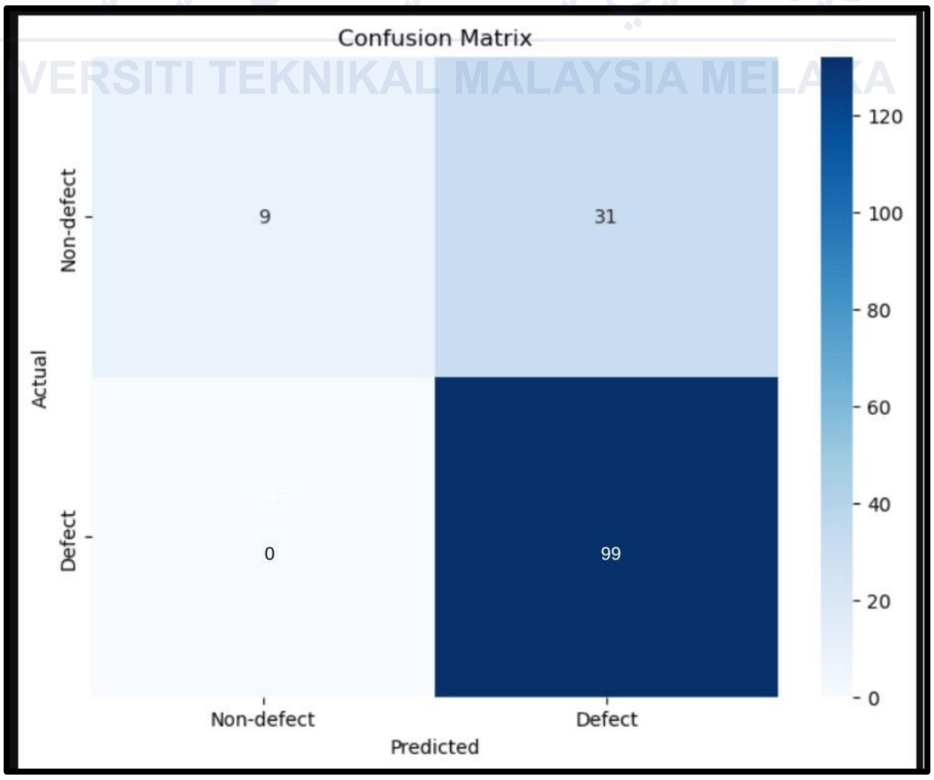


Figure 4. 8 Confusion Matrix of Models

4.3.2 Experiment 2: Performance Testing with Different Neural Network

From the Confusion matrix table, all the parameters are calculated and tabulated in Table 4-2 for comparison purposes.

Table 4- 2 Performance with Neural Network

	DEFECTS				NON-DEFECTS			
TYPE OF NEURAL NETWORK	FPS	ACCURACY	PRECISION	RECALL	FPS	ACCURACY	PRECISION	RECALL
CONVOLUTIONAL NEURAL NETWORK	79.58%	97.00%	73.95%	1.0	36.73%	97.00%	99%	0.225

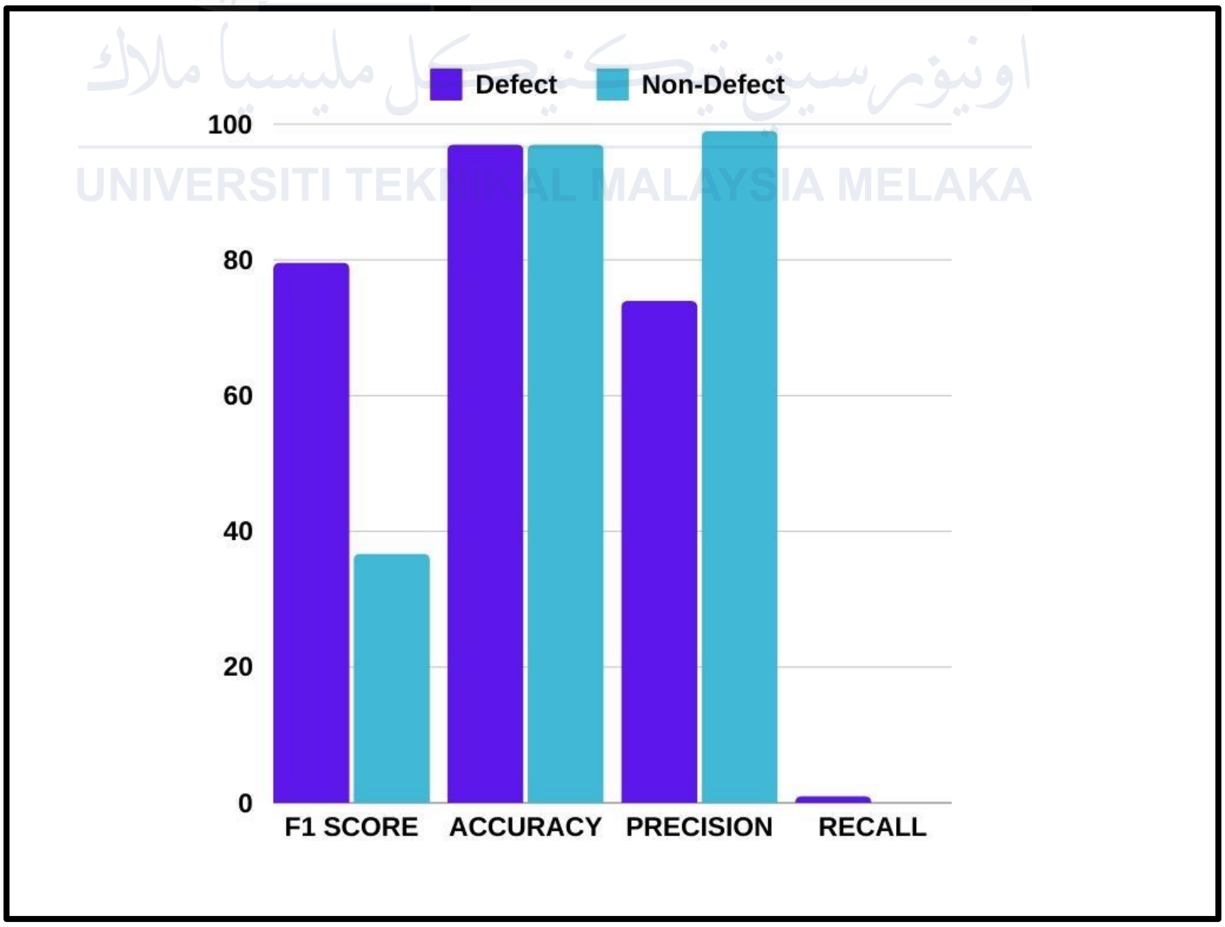


Figure 4. 9 The Performance of Neural Network Between Defect and Non-Defect

From the table 4-5, identified that the bar graph illustrates the performance two models perform for defect and non-defect detection in terms of F1 Score, Accuracy, Precision, and Recall. The non-defect model's F1 Score is roughly 40, whereas the defect model's is roughly 80, suggesting that the defect model strikes a better balance between recall and precision. The near-perfect accuracy of both models, which hovers around 100, indicates that their forecasts are generally accurate. In terms of precision, the defect model performs marginally worse at about 80, while the non-defect model scores a perfect 100. The recall metric, however, is where the biggest distinction can be found. The defect model has an almost negligible recall, meaning it fails to identify most of the actual defect cases. In stark contrast, the non-defect model excels with a perfect recall score, capturing all non-defect cases accurately.

The optimal model selection, considering this approach, is contingent upon the particular context and relative importance of several metrics. The defect model is unsuitable for circumstances where finding all defects is crucial because of its extremely poor recall, which compromises its high F1 Score and precision. However, even though the non-defect model has a lower F1 Score, its perfect accuracy, precision, and recall make it more dependable and consistent. Therefore, increasing the defect model's recall is required for applications where missing faults could have detrimental effects. As things stand, the non-defect model performs well and consistently across all assessed criteria, making it the more reliable option.

CHAPTER 5:
CONCLUSION

5.1 Conclusion

This project presents comprehensive works on the design and development of a defect and non-defect image classification system to determine and classify the categories of the images whether it's a defect or non-defect automatically. The image classification system is designed vision-based, and the neural network is used. The objective of this project was to develop an algorithm that can effectively differentiate between defect and non-defect images using AI techniques, evaluate the classification performance of the developed algorithm, and design a Graphical User Interface (GUI) for a defect recognition system that integrates seamlessly with C-SAM equipment, ensuring operator accessibility.

In order to effectively identify objects, the project started with methodical and exact steps including data collection and customization, labeling and annotation, and the use of a CNN-based model with machine learning implementation. The software configuration produced a fully integrated and optimized system, which comprised installing the required Python libraries and utilizing coding using Tensorflow. Next, the designed GUI facilitates easy integration with C-SAM equipment, providing a convenient and user-friendly interface that enhances operator accessibility and efficiency. This interface allows operators to easily interact with the system, view classification results on the CSV file, and make informed decisions based on the algorithm's outputs

Eventually, this project basically created a C-SAM Image Classification system that offers accuracy, efficiency, and seamless integration for circumstances involving Semiconductor industries. The amalgamation of hardware and software components, in addition to Tensorflow and machine learning implementation, make the classification system an extremely promising option for the workers as the system operates automatically in the semiconductor sectors. This programme lays the groundwork for upcoming innovations and advancements in the field of automated inspection and classification systems thus enhancing operational efficiency in the relevant industrial applications.

5.2 Future Works

Future research endeavors should explore alternative avenues to enhance the Defect and Non-defect image classification system. The simultaneous integration of model training and selecting the correct model architecture is a crucial element that provides an all-inclusive classification automation system. This entails fine-tuning the system's algorithms to identify different defect or non-defect images and associate them with their appropriate labeling, training and validation resulting in a seamless and efficient classification system.

In addition, future endeavors can prioritize the integration of cutting-edge technologies to account for the C-SAM images in which the system accuracy and sensitivity need to be improved by using more different data with a different model to the system. By using Auto-encoder for anomaly detection process, the system might provide a more accurate picture of the defects associated with various types of defects, improving a fair and accurate classification process.

The next developments aim to improve the Defect and Non-defect C-SAM Image Classification system by making it more intelligent, intuitive, and flexible enough to fit in with many types of semiconductor industrial sector environments. Putting these upgrades into practice would not only improve in addition to enhancing customer happiness through a quick and convenient Classification process, it will advance automated inspection and classification systems in the semiconductor industry as a whole.

REFERENCES

[1] F. Bertocci, A. Grandoni, and T. Djuric-Rissner, "Scanning Acoustic Microscopy (SAM): A Robust Method for Defect Detection during the Manufacturing Process of Ultrasound Probes for Medical Imaging," *Sensors*, vol. 19, no. 22, p. 4868, Nov. 2019, doi: 10.3390/s19224868.

[2] C. S. Tsai, C. C. Lee, and J. K. Wang, "Diagnosis of Hybrid Microelectronics using Transmission Acoustic Microscopy," in *17th International Reliability Physics Symposium*, IEEE, Apr. 1979, pp. 178–182. doi: 10.1109/IRPS.1979.362890.

[3] H. Yu, "Scanning acoustic microscopy for material evaluation," *Appl Microsc*, vol. 50, no. 1, p. 25, Dec. 2020, doi: 10.1186/s42649-020-00045-4.

[4] F. Liu, L. Su, M. Fan, J. Yin, Z. He, and X. Lu, "Using scanning acoustic microscopy and LM-BP algorithm for defect inspection of micro solder bumps," *Microelectronics Reliability*, vol. 79, pp. 166–174, Dec. 2017, doi: 10.1016/j.microrel.2017.10.029.

[5] C. S. Lee, G.-M. Zhang, D. M. Harvey, and A. Qi, "Characterization of micro-crack propagation through analysis of edge effect in acoustic microimaging of microelectronic packages," *NDT & E International*, vol. 79, pp. 1–6, Apr. 2016, doi: 10.1016/j.ndteint.2015.11.007.

[6] A. S. Dukhin and P. J. Goetz, "Fundamentals of Acoustics in Homogeneous Liquids," 2010, pp. 91–125. doi: 10.1016/S1383-7303(10)23003-X.

[7] P. Aryan, S. Sampath, and H. Sohn, "An Overview of Non-Destructive Testing Methods for Integrated Circuit Packaging Inspection," *Sensors*, vol. 18, no. 7, p. 1981, Jun. 2018, doi: 10.3390/s18071981.

[8] "2018_De Wolf_ISTFA".

[9] Andrew. Briggs and O. V. (Oleg V.) Kolosov, *Acoustic microscopy*. Oxford University Press, 2010.

[10] N. Sharma, V. Jain, and A. Mishra, “An Analysis Of Convolutional Neural Networks For Image Classification,” *Procedia Comput Sci*, vol. 132, pp. 377–384, 2018, doi: 10.1016/j.procs.2018.05.198.

[11] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[12] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights into Imaging*, vol. 9, no. 4. Springer Verlag, pp. 611–629, Aug. 01, 2018. doi: 10.1007/s13244-018-0639-9.

[13] A.-D. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee, “Distribution Padding in Convolutional Neural Networks,” in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, Sep. 2019, pp. 4275–4279. doi: 10.1109/ICIP.2019.8803537.

[14] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. doi: 10.1109/CVPR.2007.383157.

[15] S.-C. Huang, A. Pareek, M. Jensen, M. P. Lungren, S. Yeung, and A. S. Chaudhari, “Self-supervised learning for medical image classification: a systematic review and implementation guidelines,” *NPJ Digit Med*, vol. 6, no. 1, p. 74, Apr. 2023, doi: 10.1038/s41746-023-00811-0.

[16] S. Cheon, H. Lee, C. O. Kim, and S. H. Lee, “Convolutional Neural Network for Wafer Surface Defect Classification and the Detection of Unknown Defect Class,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 2, pp. 163–170, May 2019, doi: 10.1109/TSM.2019.2902657.

[17] “A Review of Image Classification Approaches and Techniques,” *International Journal of Recent Trends in Engineering and Research*, vol. 3, no. 3, pp. 1–5, Mar. 2017, doi: 10.23883/IJRTER.2017.3033.XTS7Z.

[18] P. Gavali and J. S. Banu, "Deep Convolutional Neural Network for Image Classification on CUDA Platform," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, Elsevier, 2019, pp. 99–122. doi: 10.1016/B978-0-12-816718-2.00013-0.

[19] Jianxin Wu, "Efficient HIK SVM Learning for Image Classification," *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4442–4453, Oct. 2012, doi: 10.1109/TIP.2012.2207392.

[20] H. R. Roth *et al.*, "Improving Computer-aided Detection using Convolutional Neural Networks and Random View Aggregation." [Online]. Available: https://code.google.com/p/cuda-convnet

[21] Y. Chang, "Research on de-motion blur image processing based on deep learning," *J Vis Commun Image Represent*, vol. 60, pp. 371–379, Apr. 2019, doi: 10.1016/j.jvcir.2019.02.030.

[22] Karen Hao, "What is Machine Learning."

[23] H. Wu and Z. Zhou, "Using Convolution Neural Network for Defective Image Classification of Industrial Components," *Mobile Information Systems*, vol. 2021, pp. 1–8, Sep. 2021, doi: 10.1155/2021/9092589.

[24] E. Newman, M. Kilmer, and L. Horesh, "Image classification using local tensor singular value decompositions," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, IEEE, Dec. 2017, pp. 1–5. doi: 10.1109/CAMSAP.2017.8313137.

[25] X. Wang, C. Chen, Y. Cheng, and Z. J. Wang, "Zero-Shot Image Classification Based on Deep Feature Extraction," *IEEE Trans Cogn Dev Syst*, vol. 10, no. 2, pp. 432–444, Jun. 2018, doi: 10.1109/TCDS.2016.2632178.

[26] Z. Sun, F. Li, and H. Huang, "Large Scale Image Classification Based on CNN and Parallel SVM," 2017, pp. 545–555. doi: 10.1007/978-3-319-70087-8_57.

[27] A. Anton, N. F. Nissa, A. Janiati, N. Cahya, and P. Astuti, "Application of Deep Learning Using Convolutional Neural Network (CNN) Method For Women's

Skin Classification," *Scientific Journal of Informatics*, vol. 8, no. 1, pp. 144–153, May 2021, doi: 10.15294/sji.v8i1.26888.

[28] H. Sepriandi, I. Jondri, M. Si, and U. N. Wisesty, "ANALISIS DEEP LEARNING UNTUK MENGENALI QRS KOMPLEKS PADA SINYAL ECG DENGAN METODE CNN."

[29] G. Wei, G. Li, J. Zhao, and A. He, "Development of a LeNet-5 Gas Identification CNN Structure for Electronic Noses," *Sensors*, vol. 19, no. 1, p. 217, Jan. 2019, doi: 10.3390/s19010217.

[30] Y. Gao and K. M. Mosalam, "Deep Transfer Learning for Image-Based Structural Damage Recognition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 748–768, Sep. 2018, doi: 10.1111/mice.12363.

[31] D. Larsen-Freeman, "Just learning," *Language Teaching*, vol. 50, no. 3, pp. 425–437, Jul. 2017, doi: 10.1017/S0261444817000106.

[32] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans Knowl Data Eng*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[33] K. Simonyan and A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," 2015. [Online]. Available: http://www.robots.ox.ac.uk/

[34] H. Bichri, A. Chergui, and M. Hain, "Image Classification with Transfer Learning Using a Custom Dataset: Comparative Study," *Procedia Comput Sci*, vol. 220, pp. 48–54, 2023, doi: 10.1016/j.procs.2023.03.009.

[35] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer New York, 1995. doi: 10.1007/978-1-4757-2440-0.

[36] H. Jiang, W.-K. Ching, and Z. Zheng, "Kernel Techniques in Support Vector Machines for Classification of Biological Data," *International Journal of Information Technology and Computer Science*, vol. 3, no. 2, pp. 1–8, Mar. 2011, doi: 10.5815/ijitcs.2011.02.01.

[37] B. B. Damodaran and R. R. Nidamanuri, “Assessment of the impact of dimensionality reduction methods on information classes and classifiers for hyperspectral image classification by multiple classifier system,” *Advances in Space Research*, vol. 53, no. 12, pp. 1720–1734, Jun. 2014, doi: 10.1016/j.asr.2013.11.027.

[38] P. Wang, Y. Zhang, and W. Jiang, “Application of K-Nearest Neighbor (KNN) Algorithm for Human Action Recognition,” in *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, IEEE, Jun. 2021, pp. 492–496. doi: 10.1109/IMCEC51613.2021.9482165.

[39] Q. Xiao, X. Zhong, and C. Zhong, “Application Research of KNN Algorithm Based on Clustering in Big Data Talent Demand Information Classification,” *Intern J Pattern Recognit Artif Intell*, vol. 34, no. 06, p. 2050015, Jun. 2020, doi: 10.1142/S0218001420500159.

[40] S. Sehgal, “Remotely Sensed LANDSAT Image Classification Using Neural Network Approaches,” vol. 2, pp. 43–046, [Online]. Available: www.ijera.com

[41] by Dian, “Image Classification Using Artificial Neural Networks [1].” [Online]. Available: https://www.kaggle.com/c/digit-recognizer/data#

[42] S. S. Haykin, *Neural networks : a comprehensive foundation*. Macmillan, 1994.

[43] M. Seetha, I. V Muralikrishna, B. L. Deekshatulu, B. L. Malleswari, and P. Hegde, “Artificial neural networks and other methods of image classification,” 2007. [Online]. Available: www.jatit.org

[44] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, “Comparative analysis of deep learning image detection algorithms,” *J Big Data*, vol. 8, no. 1, p. 66, Dec. 2021, doi: 10.1186/s40537-021-00434-w.

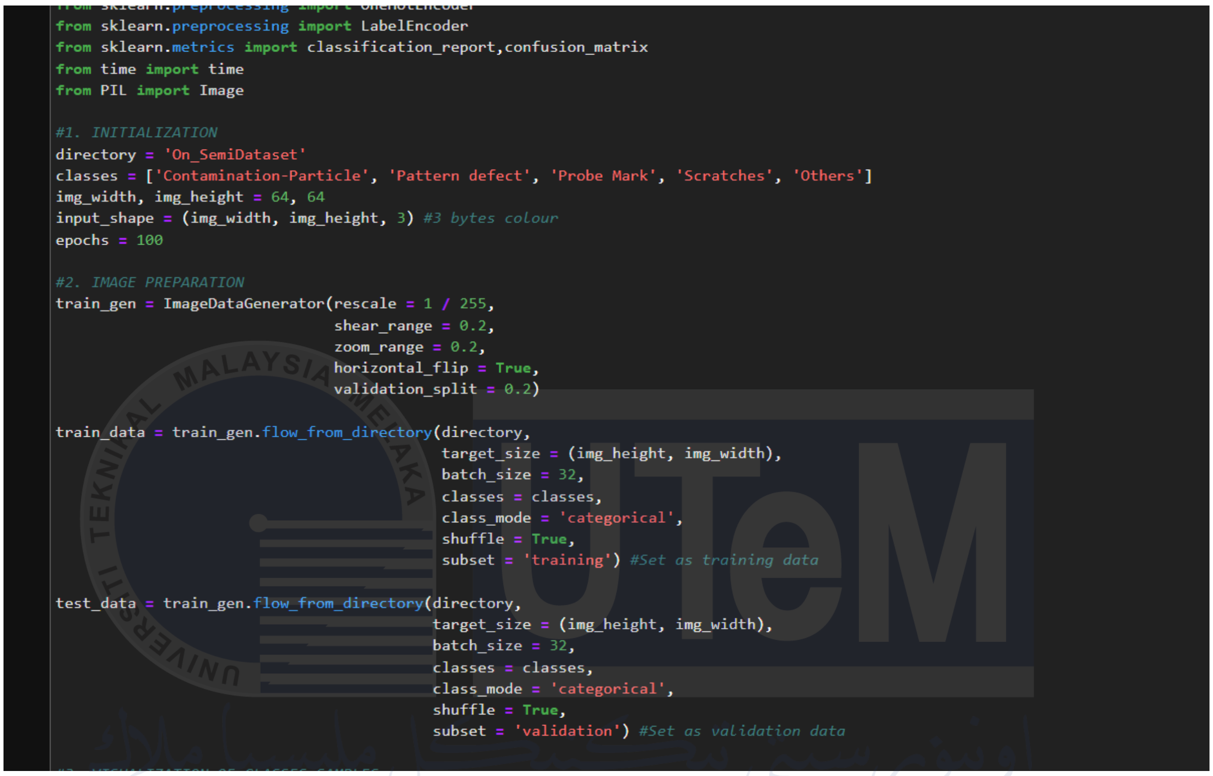
[45] B. Gašparović, J. Lerga, G. Mauša, and M. Ivašić-Kos, “Deep Learning Approach For Objects Detection in Underwater Pipeline Images,” *Applied Artificial Intelligence*, vol. 36, no. 1, Dec. 2022, doi: 10.1080/08839514.2022.2146853.

[46] X. Sun, P. Wu, and S. C. H. Hoi, “Face detection using deep learning: An improved faster RCNN approach,” *Neurocomputing*, vol. 299, pp. 42–50, Jul. 2018, doi: 10.1016/j.neucom.2018.03.030.

[47] Augustin Ador, “The Nuts and Bolts of Deep Learning Algorithms for Object Detection,” Scaling AI, Tech Blog.

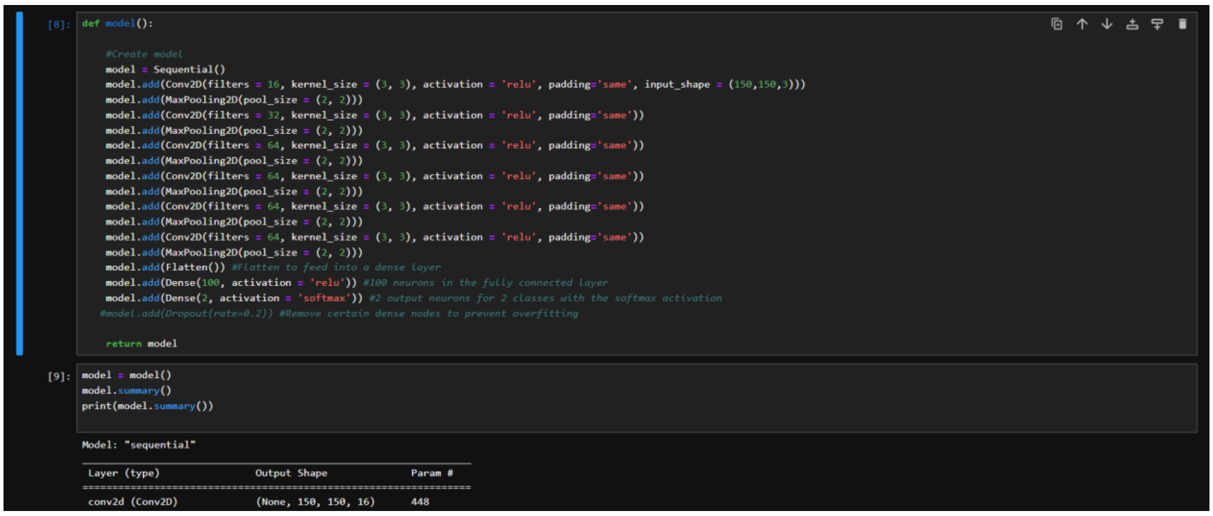
[48] Kevin Markham, “Simple guide to confusion matrix terminologyhttps://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/.”

APPENDICES



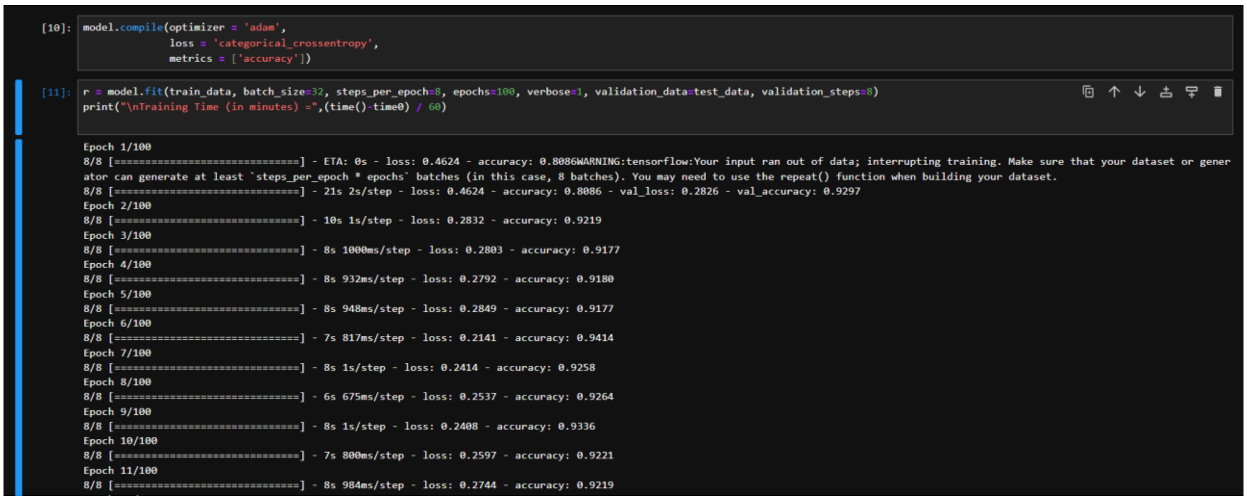
```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,confusion_matrix
from time import time
from PIL import Image

#1. INITIALIZATION
directory = 'On_SemiDataset'
classes = ['Contamination-Particle', 'Pattern defect', 'Probe Mark', 'Scratches', 'Others']
img_width, img_height = 64, 64
input_shape = (img_width, img_height, 3) #3 bytes colour
epochs = 100

#2. IMAGE PREPARATION
train_gen = ImageDataGenerator(rescale = 1 / 255,
                               shear_range = 0.2,
                               zoom_range = 0.2,
                               horizontal_flip = True,
                               validation_split = 0.2)

train_data = train_gen.flow_from_directory(directory,
                                           target_size = (img_height, img_width),
                                           batch_size = 32,
                                           classes = classes,
                                           class_mode = 'categorical',
                                           shuffle = True,
                                           subset = 'training') #Set as training data

test_data = train_gen.flow_from_directory(directory,
                                          target_size = (img_height, img_width),
                                          batch_size = 32,
                                          classes = classes,
                                          class_mode = 'categorical',
                                          shuffle = True,
                                          subset = 'validation') #Set as validation data
```

APPENDIX 1: Image Preparation



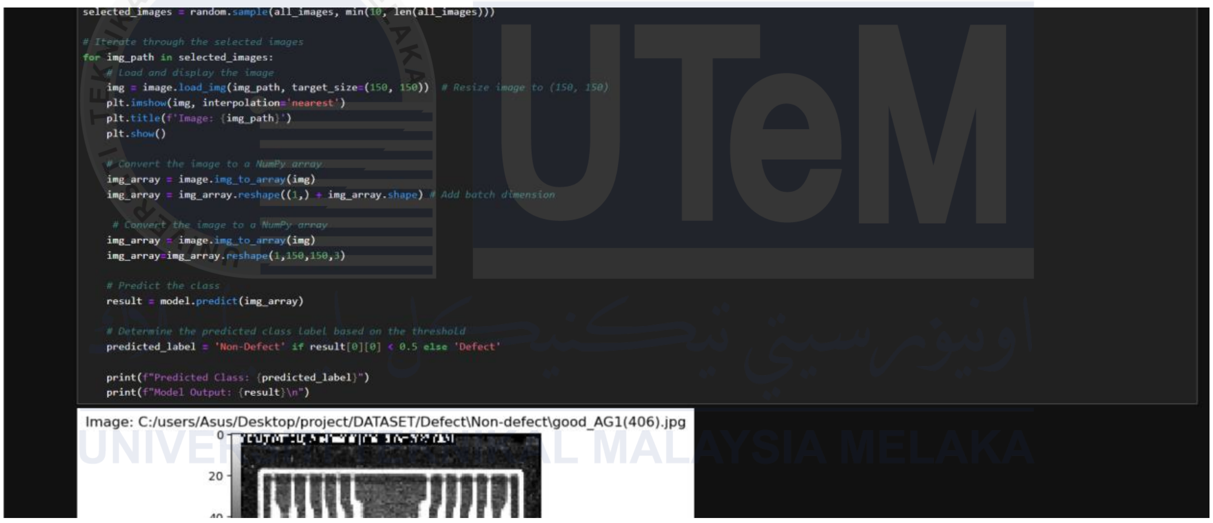
```
[8]: def model():

         #Create model
         model = Sequential()
         model.add(Conv2D(filters = 16, kernel_size = (3, 3), activation = 'relu', padding='same', input_shape = (150,150,3)))
         model.add(MaxPooling2D(pool_size = (2, 2)))
         model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu', padding='same'))
         model.add(MaxPooling2D(pool_size = (2, 2)))
         model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
         model.add(MaxPooling2D(pool_size = (2, 2)))
         model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
         model.add(MaxPooling2D(pool_size = (2, 2)))
         model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
         model.add(MaxPooling2D(pool_size = (2, 2)))
         model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding='same'))
         model.add(MaxPooling2D(pool_size = (2, 2)))
         model.add(Flatten()) #Flatten to feed into a dense layer
         model.add(Dense(100, activation = 'relu')) #100 neurons in the fully connected layer
         model.add(Dense(2, activation = 'softmax')) #2 output neurons for 2 classes with the softmax activation
        #model.add(Dropout(rate=0.2)) #Remove certain dense nodes to prevent overfitting

         return model

[9]: model = model()
     model.summary()
     print(model.summary())

     Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      conv2d (Conv2D)             (None, 150, 150, 16)      448
```

APPENDIX 2: Model Layering



```
[10]: model.compile(optimizer = 'adam',
                    loss = 'categorical_crossentropy',
                    metrics = ['accuracy'])

[11]: r = model.fit(train_data, batch_size=32, steps_per_epoch=8, epochs=100, verbose=1, validation_data=test_data, validation_steps=8)
      print("\nTraining Time (in minutes) =",(time()-time0) / 60)

      Epoch 1/100
      8/8 [==============================] - ETA: 0s - loss: 0.4624 - accuracy: 0.8086WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or gener
      ator can generate at least `steps_per_epoch * epochs` batches (in this case, 8 batches). You may need to use the repeat() function when building your dataset.
      8/8 [==============================] - 21s 2s/step - loss: 0.4624 - accuracy: 0.8086 - val_loss: 0.2826 - val_accuracy: 0.9297
      Epoch 2/100
      8/8 [==============================] - 10s 1s/step - loss: 0.2832 - accuracy: 0.9219
      Epoch 3/100
      8/8 [==============================] - 8s 1000ms/step - loss: 0.2803 - accuracy: 0.9177
      Epoch 4/100
      8/8 [==============================] - 8s 932ms/step - loss: 0.2792 - accuracy: 0.9180
      Epoch 5/100
      8/8 [==============================] - 8s 948ms/step - loss: 0.2849 - accuracy: 0.9177
      Epoch 6/100
      8/8 [==============================] - 7s 817ms/step - loss: 0.2141 - accuracy: 0.9414
      Epoch 7/100
      8/8 [==============================] - 8s 1s/step - loss: 0.2414 - accuracy: 0.9258
      Epoch 8/100
      8/8 [==============================] - 6s 675ms/step - loss: 0.2537 - accuracy: 0.9264
      Epoch 9/100
      8/8 [==============================] - 8s 1s/step - loss: 0.2408 - accuracy: 0.9336
      Epoch 10/100
      8/8 [==============================] - 7s 800ms/step - loss: 0.2597 - accuracy: 0.9221
      Epoch 11/100
      8/8 [==============================] - 8s 984ms/step - loss: 0.2744 - accuracy: 0.9219
```

APPENDIX 3: Model Training



```
selected_images = random.sample(all_images, min(10, len(all_images)))

# Iterate through the selected images
for img_path in selected_images:
    # Load and display the image
    img = image.load_img(img_path, target_size=(150, 150))  # Resize image to (150, 150)
    plt.imshow(img, interpolation='nearest')
    plt.title(f'Image: {img_path}')
    plt.show()

    # Convert the image to a NumPy array
    img_array = image.img_to_array(img)
    img_array = img_array.reshape((1,) + img_array.shape) # Add batch dimension

     # Convert the image to a NumPy array
    img_array = image.img_to_array(img)
    img_array=img_array.reshape(1,150,150,3)

    # Predict the class
    result = model.predict(img_array)

    # Determine the predicted class label based on the threshold
    predicted_label = 'Non-Defect' if result[0][0] < 0.5 else 'Defect'

    print(f"Predicted Class: {predicted_label}")
    print(f"Model Output: {result}\n")
```

Image: C:/users/Asus/Desktop/project/DATASET/Defect\Non-defect\good_AG1(406).jpg
0
20
40

APPENDIX 4: Model Prediction