Faculty of Electronics and Computer Technology and Engineering

# DEVELOPMENT OF LOW-COST IMAGE CLASSIFICATION SYSTEM USING ESP32-CAM

MUHAMMAD ADNI BIN KAMARUDDIN

Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

2024

# DEVELOPMENT OF LOW-COST IMAGE CLASSIFICATION SYSTEM USING ESP32-CAM

**MUHAMMAD ADNI BIN KAMARUDDIN**

**A project report submitted**
**in partial fulfilment of the requirements for the degree of**
**Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours**

**Faculty of Electronics and Computer Technology and Engineering**

**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2024**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

**BORANG PENGESAHAN STATUS LAPORAN**
**PROJEK SARJANA MUDA II**

| | | |
|---|---|---|
| Tajuk Projek | : | Development of Low-Cost Image Classification System using ESP32-CAM |
| Sesi Pengajian | : | 2024 |

Saya **MUHAMMAD ADNI BIN KAMARUDDIN** mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hak milik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (/)

| | | |
|---|---|---|
| ☐ | SULIT* | (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia sebagaimana yang termaktub dalam AKTA RAHSIA RASMI 1972) |
| ☐ | TERHAD* | (Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan) |
| ✓ | TIDAK TERHAD | |

Disahkan oleh:

M.adni
.....................................
(TANDATANGAN PENULIS)
Alamat Tetap:
Lot 2606, Kampung Hutan Nangka, 22000, Besut, Terengganu

[illegible]
.....................................
(COP DAN TANDATANGAN PENYELIA)
Ts. Dr. Mohd Syafiq bin Mispan
Ketua Program Ijazah Sarjana Muda Teknologi
Kejuruteraan Elektronik (Elektronik Industri)
Fakulti Teknologi dan Kejuruteraan Elektronik dan Komputer (FTKEK)
Universiti Teknikal Malaysia Melaka (UTeM)
Jalan Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

Tarikh: 14th February, 2024

Tarikh: 14th February, 2024

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

## DECLARATION

I declare that this project report entitled “Development of Low-Cost Image Classification System using ESP32-CAM” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

| | | |
|---|---|---|
| Signature | : | M.adni |
| Student Name | : | Muhammad Adni Bin Kamaruddin |
| Date | : | 14th February, 2024 |

**APPROVAL**

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours.

| | | |
|---|---|---|
| Signature | : | [illegible] |
| Supervisor Name | : | Ts. Dr. Mohd Syafiq Bin Mispan |
| Date | : | 14th February, 2024 |

| | | |
|---|---|---|
| Signature | : | |
| Co-Supervisor Name (if any) | : | NOT APPLICABLE |
| Date | : | |

## DEDICATION

To my beloved father and mother.

# ABSTRACT

Advances in Artificial Intelligent (AI) and the availability of large training datasets have made image classification popular in various domains. The market for AI in enterprise applications is growing rapidly, indicating the potential for further development in image classification field. Nevertheless, the high cost in developing an image classification system could hinders its widespread adoption and becomes more challenging to support Industrial Revolution (IR) 4.0. Therefore, in this project, we proposed a low-cost image classification system using ESP32-CAM which capable of performing real-time image classification based on the trained model. The method involves training a deep learning model (i.e., micro neural network) using a dataset of non-defective/defective integrated circuits (ICs) on Edge Impulse platform. Subsequently, deploying the generated code of the successful trained model on ESP32-CAM. The model is optimized to fit the limited resources (i.e., memory and processing power) of the ESP32-CAM. By using the built-in camera in ESP32-CAM, it can performs real-time image classification of non-defective/defective ICs. The proposed system achieves 86.1% prediction accuracy by using 1571 image data set of defective and non-defective ICs.

## *ABSTRAK*

Kemajuan dalam Artificial Intelligent (AI) dan tersedianya set data latihan yang besar telah menjadikan klasifikasi imej popular dalam pelbagai bidang. Pasaran untuk AI dalam aplikasi perusahaan berkembang pesat, menunjukkan potensi untuk pembangunan selanjutnya dalam bidang klasifikasi imej. Namun begitu, kos yang tinggi dalam membangunkan sistem klasifikasi imej boleh menghalang penggunaannya yang meluas dan menjadi lebih mencabar untuk menyokong Revolusi Perindustrian (IR) 4.0. Oleh itu, dalam projek ini, kami mencadangkan sistem klasifikasi imej kos rendah menggunakan ESP32-CAM yang mampu melaksanakan pengelasan imej masa nyata berdasarkan model terlatih. Kaedah ini melibatkan latihan model pembelajaran mendalam (iaitu, rangkaian saraf mikro) menggunakan set data litar bersepadu (IC) yang tidak rosak/rosak pada platform Edge Impulse. Selepas itu, menggunakan kod model terlatih dijana yang berjaya pada ESP32-CAM. Model ini dioptimumkan agar sesuai dengan pengunaan sumber yang terhad (iaitu, memori dan kuasa pemprosesan) ESP32-CAM. Dengan menggunakan kamera terbina dalam dalam ESP32-CAM, ia boleh melaksanakan pengelasan imej masa nyata bagi IC yang tidak rosak/rosak. Sistem yang dicadangkan mencapai 86.1% ketepatan ramalan dengan menggunakan 1571 set data imej bagi IC yang rosak dan tidak rosak.

# ACKNOWLEDGEMENT

I want to begin by extending my heartfelt gratitude to my supervisor, Ts. Dr. Mohd Syafiq Bin Mispan, from the Faculty of Electrical and Electronic Engineering Technology at Universiti Teknikal Malaysia Melaka. His invaluable guidance, support, and encouragement were instrumental in the successful completion of this thesis.

I attribute the successful completion of this thesis to the blessings and guidance of Allah. I express gratitude to God for providing opportunities, overcoming challenges, and making available the resources that facilitated the completion of this thesis. Finally, I extend recognition to everyone associated with this endeavor, including family members and friends, whose significant roles provided motivation during challenging times when hope seemed scarce.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| AI | - | Artificial Intelligience |
| ANN | - | Artificial Neural Network |
| CNN | - | Convolutional Neural Network |
| FMCG | - | Fast Moving Consumer Good |
| IC | - | Intergrated Circuit |
| IoT | - | Internet of Things |
| IR | - | Industry Revolution |
| MIMO | - | Multiple-input-multiple-output |
| MLP | - | Multilayer Perceptron |
| MSE | - | Mean squared error |
| PCA | - | Principal Component Analysis |
| PCB | - | Printed Circuit Board |
| PSM | - | Projek Sarjana Muda |
| ROI | - | Region of Interest |
| SIFT | - | Scale-Invariant Feature Transform |
| SRCNN | - | Super-resolution Convolutional Neural Network |
| UART | - | Universal Asynchronous Receiver / Transmitter |
| USB | - | Universal Serial Bas |
| PCB | - | Printed Circuit Board |

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

This chapter describes about the project background, problem statement, objectives, and scope of work in developing a low-cost image classification system.

## 1.1 Background

Image classification is the process of identifying specific objects in digital images or video sequences. While humans can easily recognize objects in images despite variations in size, scale, perspective, and obstruction, it remains a challenging task for computer vision systems. However, recent advances in Artificial Intelligence (AI) which perform tasks that resemble human cognitive function, along with the availability of large training datasets, have made image classification a popular application in various domains, including enterprise applications.

Figure 1.1 illustrates the global market for AI in enterprise applications, which valued at USD 95,602.77 million in 2021, and it is projected to exceed USD 1,847,495.6 million by 2030. The significant growth in this market is indicative of the potential for further development in image classification and other computer vision tasks. Image classification technology has many practical applications in fields such as healthcare, agriculture, and manufacturing. Unfortunately, the high cost of developing and implementing an image classification system can make it prohibitive for smaller organizations or those with limited budgets. By creating a low-cost image classification solution, smaller organizations can access the benefits of this technology without incurring significant expenses. This can help manufacturers improve their operations, increase efficiency, and reduce costs in areas such as production and quality control.

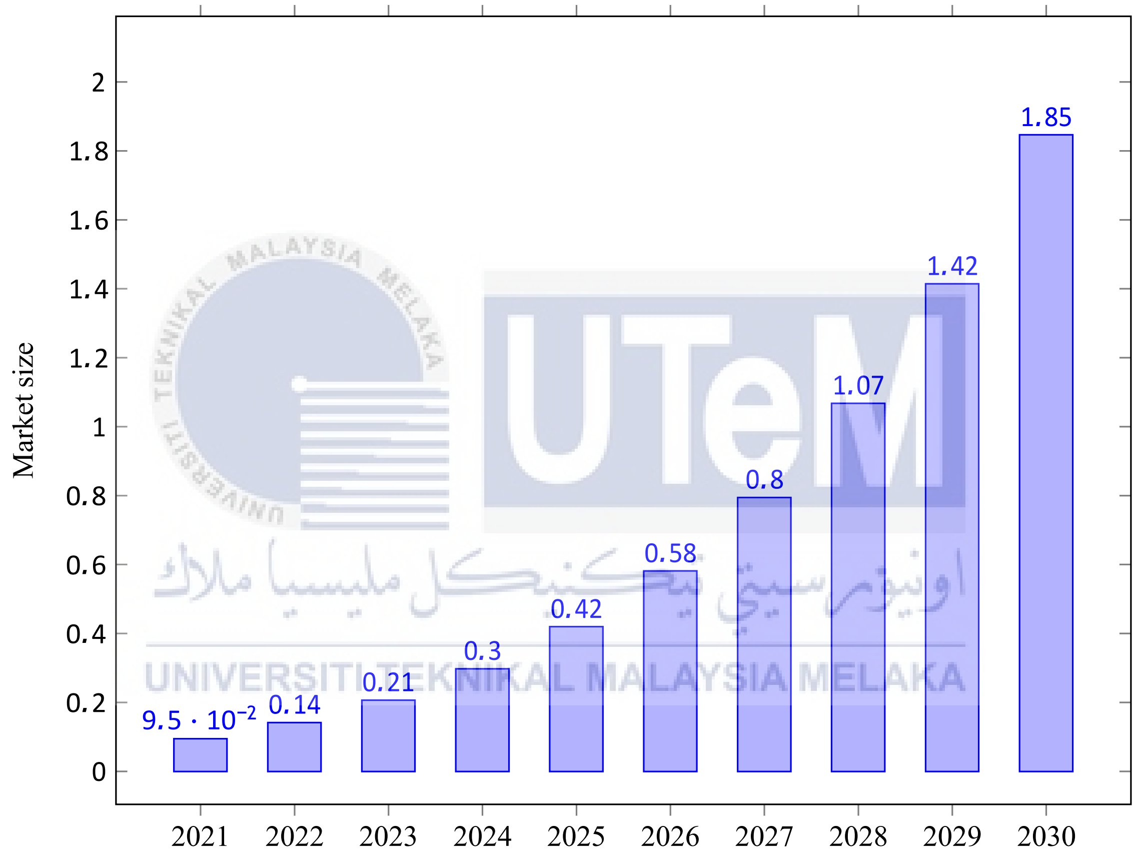


**Figure 1.1:** Market value of AI worlwide from 2021 to 2030.

Recent developments in neural networks have made it possible for them to run on platforms with lower computational power, such as microcontrollers. These networks can analyze data and be trained to make accurate predictions for previously unseen data. Therefore, the aim of developing a low-cost image classification system is to make this technology more widely available and enable smaller organizations to benefit from its capabilities without incurring significant expenses [1].

## 1.2 Problem Statement

With the rise of Industrial Revolution (IR) 4.0, there has been a widespread growth of AI in various applications, including image classification systems. While AI has shown significant improvements in the accuracy and speed of image classification, there are still several challenges that need to be addressed to fully support the IR 4.0. Developing image classification systems that can accurately recognize and classify objects in digital images or videos despite variations in size, scale, perspective, and obstruction is a major challenge that needs to be addressed while keeping costs in mind. While recent advances in AI and large training datasets have made image classification a popular application in various domains due to the benefits it offers such as improved efficiency, accuracy, and increased safety, challenges such as bias, variability in contexts, and scalability still need to be addressed for better performance.

In the past, several techniques have been proposed to classify an image, but yet users are still unable to classify each object accurately [2, 3, 4, 5, 6]. Additionally, image classification systems that are currently available in the market tend to be costly, making them unaffordable for many [7] [8]. Therefore, in this project, we propose to develop a low-cost image classification system that can classify each image with higher accuracy. Our aim is to address the challenges in image classification system by developing a system that is cost-effective and accurate, which can be used in various domains to improve efficiency, reduce costs, and increase safety.

## 1.3 Project Objectives

The main aim of this project is to develop a low-cost image classification system. Specifically, the objectives are as follows:

1. To train the suitable machine learning model for image classification using edge impulse.
2. To build the trained model of an image classification system on ESP32-CAM.
3. To evaluate the prediction accuracy of the image classification system.

## 1.4 Scope of Work

The project scope of developing low-cost image classification system is as follows:

1. A lightweight machine learning technique is used in microcontroller, e.g., micro neural network architecture.
2. The image classification system is based on RGB images (i.e., input to machine learning algorithm).
3. Edge impulse is used to train the image classification model and generate the corresponding Arduino codes.

## 1.5 Report Outline

Chapter 2 provides an overview of previously proposed techniques on low-cost image classification system. The advantages and disadvantages of these techniques are also discussed in this chapter.

Chapter 3 describes the methodology and design steps in developing a low-cost image classification system. Each design step corresponds to each objective is represented by the flow chart and it is discussed thoroughly in this chapter. Timeline for Projek Sarjana Muda (PSM) I and future plan for PSM II are also discussed.

Chapter 4 discusses the performance evaluation of the proposed low-cost image classification system. Moreover, the analysis to improve the prediction accuracy of the proposed system is also discussed in this chapter.

Chapter 5 concludes the project findings in this report. Suggestions for future work directions and potential commercialization are also provided.

# CHAPTER 2

# LITERATURE REVIEW

This chapter provides a broad overview of the project related to the topic in this report. First, the chapter introduces the relevant background knowledge of this project; image processing, image classification, and neural network architecture. Subsequently in this chapter, the previously proposed techniques for image classification system are critically discussed.

## 2.1 Image Processing and Classification

Image processing and image classification are closely related, each serving distinct purposes but often interconnected in practice. Image processing involves manipulating and analyzing digital images to enhance image quality, extract useful information, and interpret information required from the image. Techniques such as noise reduction, image enhancement, feature extraction, and segmentation are used. These techniques aim to pre-process and transform images to make it suitable for subsequent analysis including image classification. Meanwhile, image classification is a specific task within computer vision that involves categorizing images into predefined classes or categories based on their visual content. It focuses on training machine learning models to recognize and assign labels to images automatically. The combination between image processing and image classification techniques produce a lot of application such as object counting and object identification.

Image classification is a fundamental task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a specific label. Typically, image classification refers to images in which only one object appears and is analyzed. In image classification, neural network architectures are designed to process image data by extracting hierarchical features.

### 2.1.1 Architecture of Neural Network

The architecture of neural network consists of an input, output, and hidden layer. The architecture of neural network determines how information flows through the network, how it is processed, and how predictions or how image classification are made. Neural networks function by passing data through the layers of an artificial neuron. Figure 2.1 illustrates the architecture of neural network. Neural network consists of three basic components which are input layer, hidden layer, and output layer. These three layers process and transform input data into meaningful predictions or outputs.

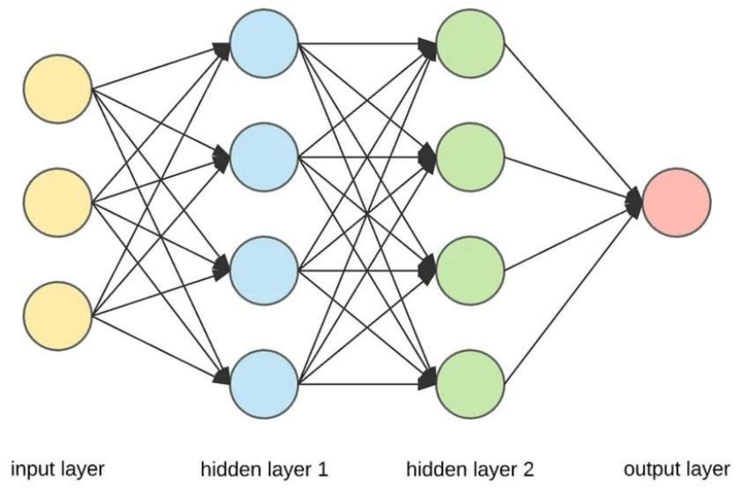


**Figure 2.1:** Neural network architecture.

The input layer is the entry point of the neural network. Each neuron in the input layer corresponds to a specific feature or attribute of the input data. In image classification, each input neuron may represent a pixel value or a particular visual feature. The input layer simply passes the input data to the neurons in the next layer for further processing process.

Meanwhile, hidden layer performs computation and learn representations of the input data. The number of hidden layers and the number of neurons in each layer vary depends on the complexity of the problem and the dataset size. These hidden layers enable the neural network to capture and figure out complex patterns and relationships present in the input data through a series of mathematical transformations. The output layer is the final layer of the neural network that produces the predictions. Each neuron in the output layer corresponds to a specific class or variable.

### 2.1.2 Micro Neural Network

Micro neural network is a specialized type of neural network architecture designed to operate efficiently on low resources devices. The micro neural network consists of multiple micro layers, which serve as the building blocks of the network. Figure 2.2 illustrates the micro neural network architecture. Each micro layer receives the output from the previous layer as its input and performs computations on that input. The exact operations and design choices within each micro layer are determined by the specific requirements and constraints of the task. The first micro layer takes the initial input and processes it using operations tailored for the task at hand. These operations could include linear transformations, non-linear activation functions, pooling, or lightweight convolutional operations.

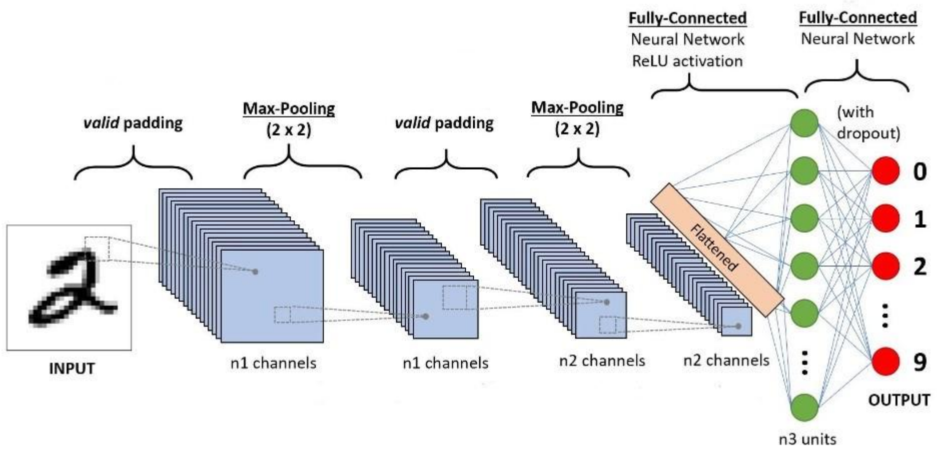


**Figure 2.2:** Micro neural network architecture.

The output of micro layer 1 then becomes the input for micro layer 2. Similar to the previous layer, micro layer 2 performs specific computations to further transform the data. It may introduce additional non-linearities, apply feature extraction techniques, or other relevant operations based on the specific requirements of the task. The output of micro layer 2 serves as the input for micro layer 3, which continues the process of applying transformations to the data. Micro layer 3 can incorporate more complex operations depending on the specific task's requirements.

The number of micro layers in the network can vary depending on the desired depth and complexity of the model. Factors such as the complexity of the task, available computational resources, and desired model performance influence the choice of the number of layers. Finally, the output layer receives the transformed data from the last micro layer and produces the final result or prediction. The structure of the output layer is task-dependent, with different activation functions used for different tasks. For example, softmax activation may be employed for classification tasks to generate class probabilities, while a linear activation function may be used for regression tasks to predict continuous values.

Thus, these devices such as microcontrollers, IoT devices, and mobile devices often have low computational power and memory. Micro neural networks address these constraints by employing various techniques such as network pruning, quantization, and knowledge distillation. These specific neural network aim to reduce the size and complexity of the network while preserving its performance as highly accuracy as possible. By optimizing for resource efficiency, micro neural networks enable the deployment of machine learning models directly on these devices, allowing for real-time and localized inference without depending on external servers. This is particularly beneficial in applications where low latency, privacy, or off-line functionality is crucial. This micro neural network usually used in object identification.

## 2.2 Known Techniques of Image Classification System

Several techniques have been proposed in the past for image classification system. In a recent study, Sriratana *et al.*, [9] proposed application of web-cam for inspection of rice grain quality by using image processing technique. The system start with converting image into rgb to reduce the resolution used. The system introduces concept of a bounding box, which is a rectangular boundary used to enclose specific areas or the entire image. This method assists in estimating various parameters such as the center of the area, major and minor axis lengths, and image coordinates. The process involves identifying pixel groups using the flood fill method to enhance the distinction between objects and the background. A total of 24 samples and 18 experimental designs are employed. The study concludes that inspecting rice grains arranged in a non-overlapping manner yields lower errors, averaging around 0.47%. Figure 2.3 illustrates the sample image that has been converted to RGB. However, this method provide limited effectiveness for certain rice grain characteristics such as surface texture.

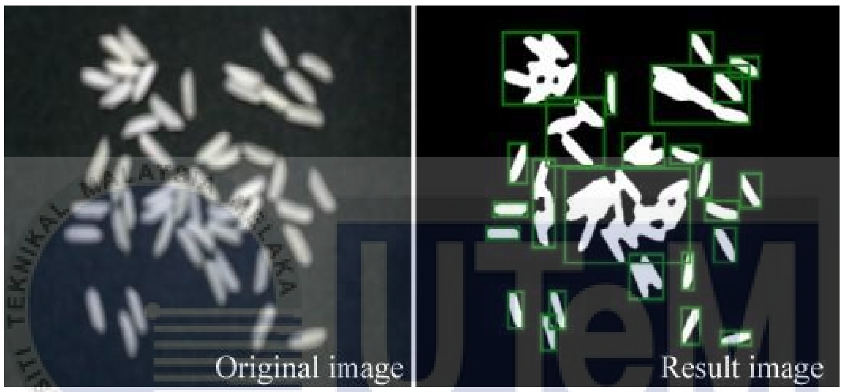


**Figure 2.3:** Sample image processing of rice grains.

Meanwhile, Yang *et al.*, [10] proposed a license plate image super-resolution based on two convolutional neural network architectures: GoogLeNet and multi-scale super-resolution convolutional neural network (SRCNN). GoogLeNet utilizes the inception architecture to extract diverse features by convolving images with filters of different sizes. SRCNN focuses on super-resolution and uses three different filter sizes simultaneously for license plate images. The models are evaluated using mean squared error (MSE) as a loss function, comparing reconstructed images with ground truth images. Figure 2.4 illustrates the license plate image reconstruction results. The system successfullay recognize the license plate with improving a 91.7%. However, the high computational requirements can pose challenges for real-life applications, due to high cost.

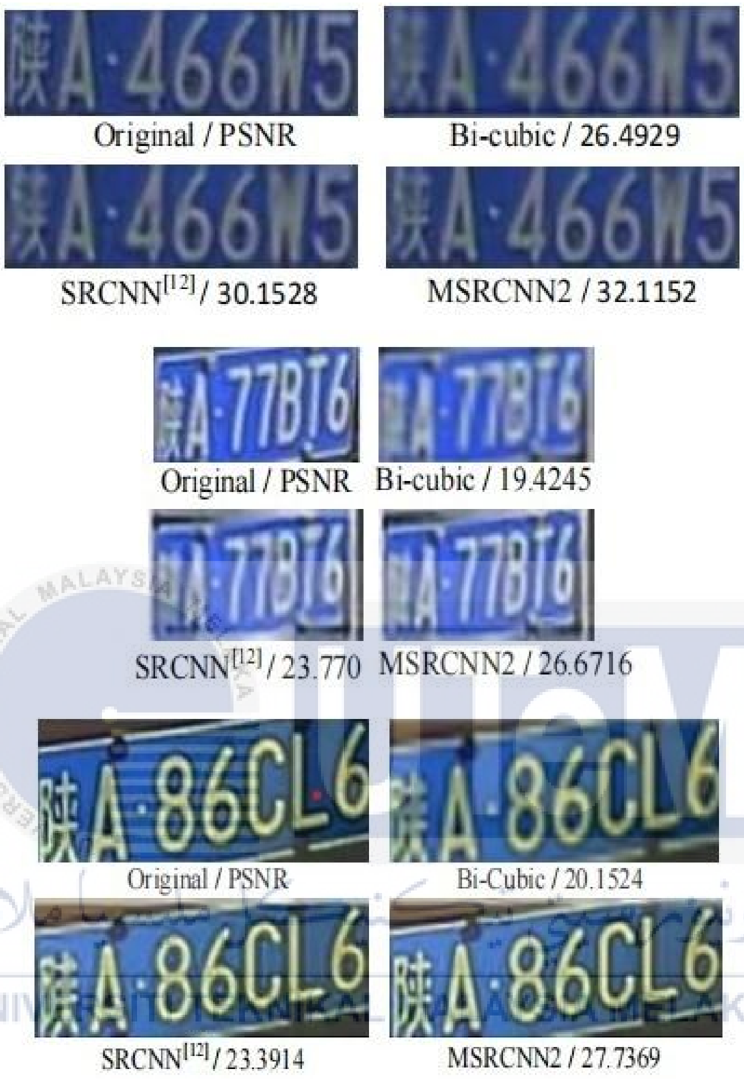


**Figure 2.4:** License plate image reconstruction results.

A similar study has been proposed by Ke Han *et al.*, [11] using artificial neural network (ANN) for processing fingerprint image noise. This method use ANN to detect and classify the fingerprint. Using Gaussian low-past filtering to remove image noise to increase the method accuracy. However, fingerprint images can vary significantly in quality such as image resolution, noise, smudging, or partial capture. Figure 2.5 illustrates fingerprint image after converting to Gaussian low-past filter. ANN can be sensitive to such variations, and their performance might degrade when applied to low-quality or distorted fingerprint images.

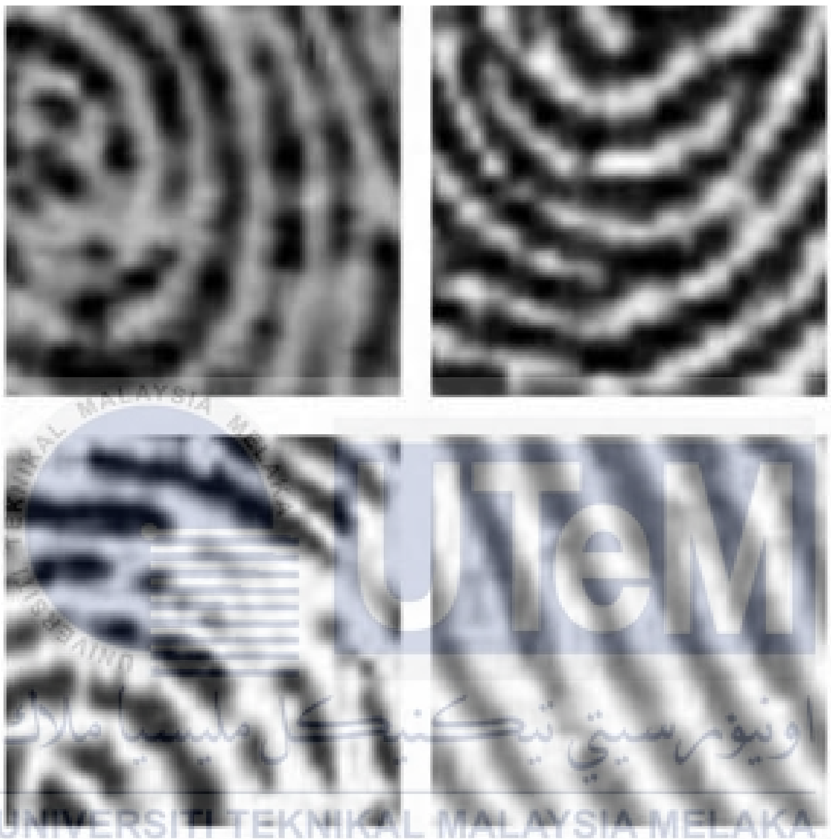
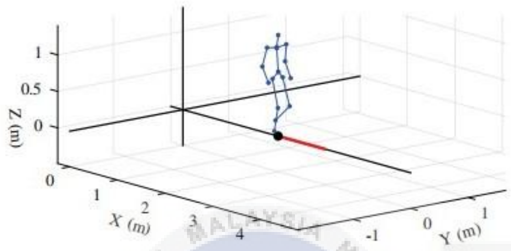
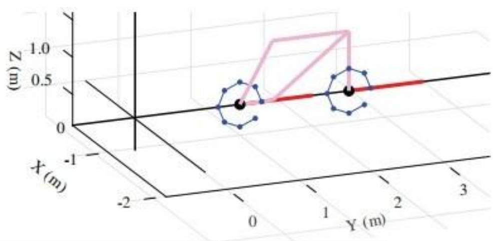


**Figure 2.5:** Fingerprint images processed by Gaussian low-pass filter method.

Elsewhere, Dubey *et al.*, [12] proposed uncertainty analysis of deep neural network for classification of vulnerable road users using micro-doppler. Micro-doppler refers to the analysis of doppler radar signals to extract information about small-scale movements which is particularly useful in identifying and tracking vulnerable road users such as pedestrians and cyclists. The researchers created a simulation environment in Matlab for a MIMO radar system as a preliminary step. This environment allowed them to adjust various parameters that affect the performance of target classification. One important parameter is the velocity resolution, which depends on the number of chirps and their duration. Figure 4.9 illustrates the reflection points. By modifying these parameters, the researchers could control the velocity resolution for better identification of micro-Doppler signatures. The method able to analysis with 99.81% of accuracy.



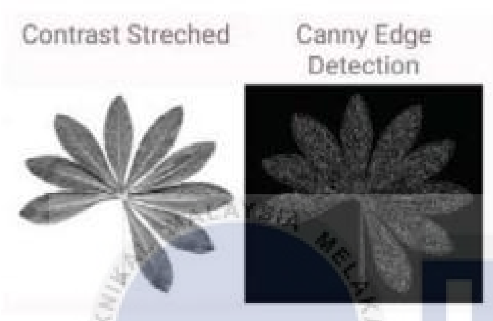
**(a)** Reflection points for a pedestrian.



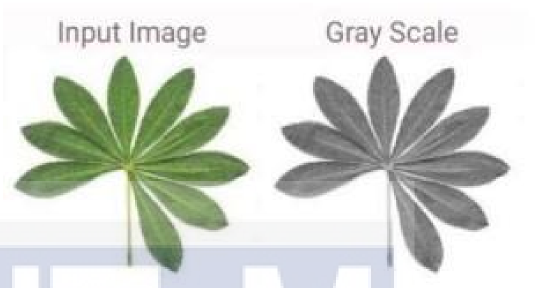
**(b)** Reflection points from rotating wheels.

**Figure 2.6:** Simulation model.

In a study, Geetha *et al.*, [13] proposed an advanced plant leaf classification method described in the given information combines image enhancement and canny edge detection. Figure 2.7 depicts the image conversion for leaf classification using canny edge detection. The approach includes initial image preprocessing and boundary enhancement using a 3x3 spatial mask. Geometric features are then extracted and processed through principal component analysis (PCA) to optimize the feature extraction process. For more accurate edge detection, the technique utilizes multilevel wavelet decomposition and a multilayer perceptron (MLP) specifically designed for grayscale image edge detection. Additionally, leaf vein pixels are extracted using histogram-based vein estimation and the Sobel operation. An enhanced version of the Canny edge detection algorithm is employed to detect real edges while minimizing false noisy edges, particularly in Asphalt concrete applications. In terms of plant recognition, the method combines thresholding and ANN classification. This involves utilizing histogram-based vein region detection and segmentation through an ANN classifier. Furthermore, for plant disease identification, Canny edge detection combined with color histograms is employed. Leaf features are then extracted and classified for various plant species.



**(a)** Gray scale conversion.



**(b)** Canny edge detection.

**Figure 2.7:** Image conversion for leaf classification.

Prathusha *et al.*, [14] proposed an enhanced image edge detection methods for crab species identification by using canny edge detection. However, canny edge detection method typically detects edges in a single direction. This means that it may struggle to detect edges that are not aligned with the dominant orientations of the gradient. Consequently, edges that are not in the preferred direction may be missed or only partially detected. This system still manage to achieve reasonable accuracy. Figure 2.8 illustrates image when converting to canny edge detection

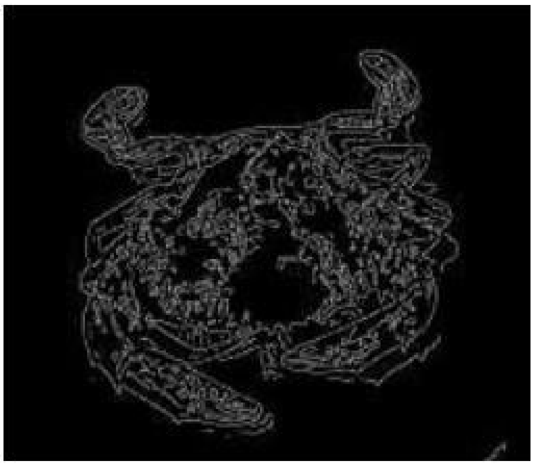


**Figure 2.8:** Canny edge detection applied to a crab image.

In another study, Chua *et al.*, [15] proposed damage identification of selected car parts using image classification and deep learning technique. From these results, the performance of the model is enough to be used on prediction. Prediction accuracy for rear bumper is 60%, car wheel is 90%, and front bumper is 80%. The model yields 100% for precision, recall, and F1-score metrics in all classes. The 60% accuracy for rear bumper is mainly due to the images of rear bumper and front bumper looking similar. Figure 2.9 illustrates example of damaged car parts.

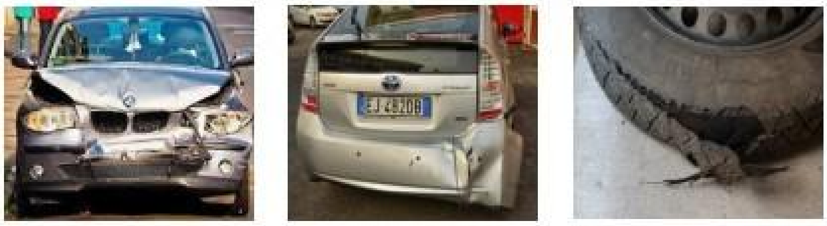
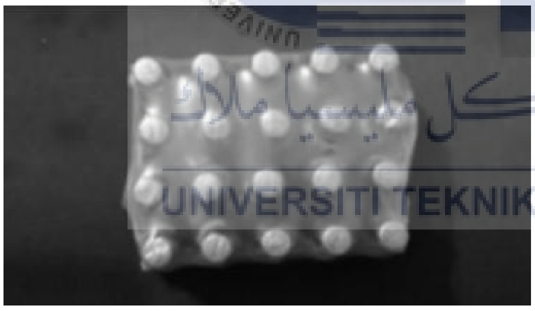


**Figure 2.9:** Left to right: damaged front bumper, rear bumper, and car wheel.

Meanwhile, an image processing application to detect faulty bottle packaging is proposed in [16]. In the pre-processing phase, the images captured from a top view are first converted from the original RGB format to gray scale. This conversion reduces the dimensionality of the images and simplifies subsequent processing steps. The gray scale images are then subjected to further pre-processing such as smoothing by using a Gaussian filter. This filter helps to reduce noise and enhance the overall quality of the images. Additionally, the images undergo circle detection using the Hough transform algorithm. The Hough transform is a technique used to detect shapes such as circles in an image by converting the shape detection problem into a parameter space. By applying the Hough transform specifically for circle detection, the algorithm can identify circular patterns within the pre-processed gray scale images. Figure 2.10 depicts the result of the image conversion using Hough transform. However, variations in lighting can significantly impact the accuracy and reliability of the image processing and circle detection steps.


**(a)** Image converted to gray scale form.

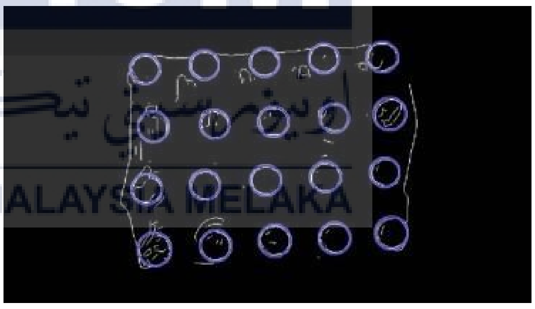

**(b)** Hough transform image.

**Figure 2.10:** Image conversion using Hough transform.

In a study, an image processing for product label quality control on fast moving consumer good (FMCG) products is proposed in [17] by using NI smart camera 1744. The image captured by the camera is processed via Lab View platform to classify each image.

Figure 2.11 illustrates the setup of the proposed method. The weakness of this method is low speed detection of each packet resulted an increase in the production time. The classification accuracy for this method is 95.45%.

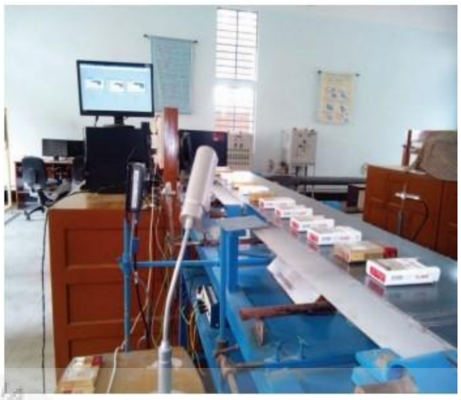


**Figure 2.11:** Experimental setup for product label quality control using image processing and classification techniques.

In another study, real-time image processing for edge inspection and defect detection in stainless steel production line is proposed by Spinola *et al.*, [18]. The proposed algorithm focuses on detecting the edge of a coil by analyzing a narrow region surrounding it. The process involves several steps to accurately determine the position of the coil edge in the image. Figure 2.12 illustrates the experimental setup for the proposed method. Firstly, a region of interest (ROI) is selected which is centred around the coil edge. To enhance the reliability of the subsequent edge detection algorithm, the ROI is binarized to effectively separating the roll pixels from the steel pixels. To strike a balance between computational efficiency and desired accuracy, the ROI is divided into multiple windows along the edge. However this method very sensitive to vibration, hence make it unsuitable for factory usage.

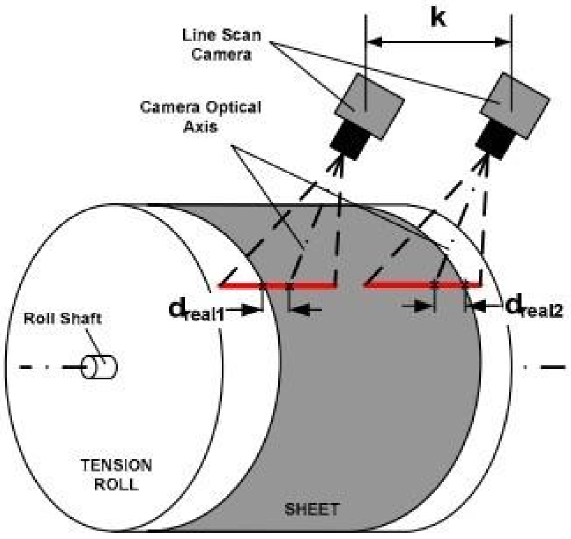


**Figure 2.12:** Experimental setup for edge inspection and defect detection in stainless steel production line.

Elsewhere, Zhu *et al.*, [19] proposed an automatic remote sensing image registration based on SIFT descriptor and image classification. This study discussed that by collecting all possible object samples or spectra within a specific research or application area from images or spectral instruments, a sample database can be created for classification purposes. However, it is important to consider the relationship between samples and image pixel values, taking into account remote sensors and atmospheric conditions. Due to limitations in experimental conditions, this study opted for manual sample collection.

All the above studies show the limitation of the proposed techniques in image classification system. Therefore, in our project, the aim is to develop a low-cost image classification system using ESP32-CAM. All the studies which have been thoroughly discussed above are summarized in Table 2.1.

**Table 2.1:** Summary of the previously proposed techniques.

| Authors | Proposed Technique | Advantages | Disadvantages |
|---|---|---|---|
| Sriratana *et al.*, [9] | Application of webcam for inspection of rice grain quality by using image processing technique | • Capture accurate photo<br>• High accuracy output | • Computational complexity<br>• Struggle to handle extreme scaling factors |
| Yang *et al.*, [10] | License plate recognition using image super-resolution based on convolutional neural network | • Capture complex patterns<br>• Generate high-resolution outputs | • Computational complexity<br>• Struggle to handle extreme scaling factors |
| Ke Han *et al.*, [11] | ANN for processing fingerprint image noise | • Capture complex patterns<br>• High image restoration | • Computationally intensive<br>• Longer processing times |
| Dubey *et al.*, [12] | Uncertainty analysis of deep neural network for classification of vulnerable road users using micro-doppler | • Straightforward and computationally efficient method for estimating uncertainty<br>• Generate high-resolution output | • High computational overhead<br>• Longer processing times |
| Geetha *et al.*, [13] | Plant leaf disease classification and detection system using machine learning | • Achieve high levels of accuracy<br>• Efficient for real-time operation | • Inadequate or biased datasets can lead to inaccurate predictions<br>• Need for continuous training and updates |
| Prathusha *et al.*, [14] | Enhanced image edge detection methods for crab species identification | • Accurate boundary detection<br>• High computational efficiency | • Inadequate or biased datasets can lead to inaccurate predictions<br>• Need for continuous training and updates |
| Chua *et al.*, [15] | Damage identification of selected car parts using image classification and deep learning | • Very efficient<br>• High computational efficiency | • Limited domain coverage<br>• Need for continuous training and updates |

*Continued from previous page . . .*

| Authors | Proposed Technique | Advantages | Disadvantages |
|---|---|---|---|
| Mane *et al.*, [20] | Identification and classification of industrial elements using artificial intelligence and image processing techniques | • High efficiency<br>• Capture complex patterns | • Computationally intensive<br>• Longer processing times |
| Sanver *et al.*, [16] | An image processing application to detect faulty bottle packaging | • Straightforward and computationally efficient<br>• Generate high-resolution output | • Required high processing times<br>• High computational overhead |
| Sarkar *et al.*, [17] | Image processing based product label quality control on FMCG products | • High processing times<br>• Efficient for real-time operation | • Limited data predictions<br>• Need for continuous training and updates |
| Spinola *et al.*, [18] | Real-time image processing for edge inspection and defect detection in stainless steel production lines | • Accurate boundary detection<br>• High computational efficiency | • Sensitive to vibration<br>• Limited resources |
| Zhu *et al.*, [19] | Automatic remote sensing image registration based on SIFT descriptor and image classification | • High efficiency<br>• Capture complex patterns | • Computationally intensive<br>• Longer processing times |

## 2.3 Summary

In this chapter, the relevant topics to the project such as image classificaitona and micro neural network architecture are briefly discussed. Further, the previously proposed techniques for image classification system are discussed. Based on the observations, the limitations of the existing techniques are longer processing time and computationally intensive. Due to this limitation, hence the low-cost image classification system is required to enable the widespread adoption of AI application.

# CHAPTER 3

# METHODOLOGY

This chapter describes the methodology and design steps to develop a low-cost image classification system.

## 3.1 Description of Methodology

The project aims to classify images into their respective groups. In this study, the classification of defective and non-defective intergrated circuits (ICs) is used as a case study. Scratches on IC are considered as defective IC. Scratches are typically the result of mechanical contact between the ICs and another object. This contact can occur when a robotic handling system misaligns or improperly handles the ICs. During the processing of ICs, various machines and robots are involved in the fabrication and handling processes. These robots are responsible for tasks such as picking up and transferring ICs to different stages of processing. However, if misalignment occur, or if the handling is not executed precisely, the component may come into contact with a surface or object that causes scratches [21]. Figure 3.1 illustrates the top-level block diagram of the low-cost image classification system using ESP32-CAM. To design the proposed system as depicted in Figure 3.1, this project is divided into three main parts which are training the dataset with suitable machine learning algorithm in Edge Impulse, deploying the code generated by Edge Impulse in microcontroller, and prediction accuracy evaluation. The above three main parts are elaborated in the next sections.

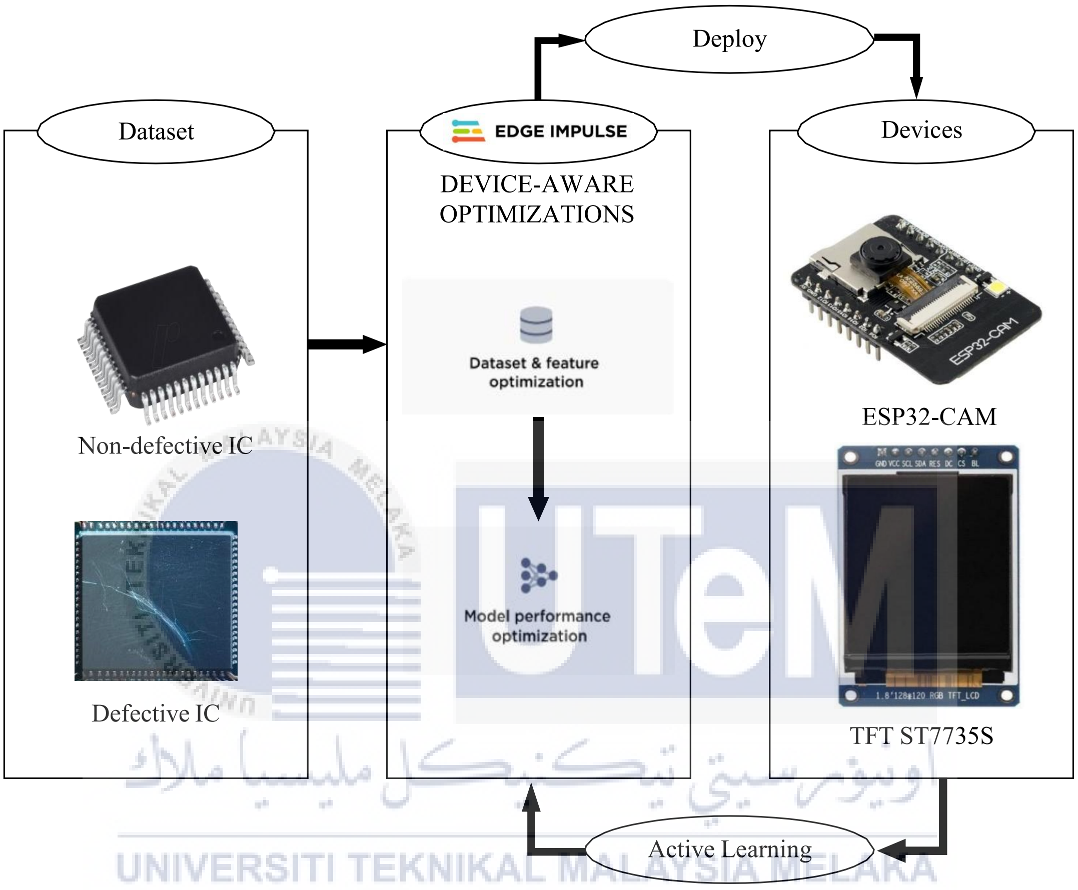


**Figure 3.1:** Top level block diagram of low-cost image classification system.

### 3.1.1 Dataset Training with Edge Impulse

The first step in designing the low-cost image classification system is to train the dataset by using Edge Impulse. Figure 3.2 depicts the design steps to train the image classification using Edge Impulse. The dataset for defect ICs need to be generated, which can be realized by using Microsoft Bing. Subsequently, an analysis of this dataset is conducted to obtain valuable information about its composition and overall quality. Any potential imbalances or biases within the dataset were identified and appropriately handled. In our study, the Edge Impulse application is used to build and train the image classification system. The generated dataset is uploaded in Edge Impulse and micro neural network is selected considering the limitation of the target hardware (i.e., ESP32-CAM). Using the training tools provided by Edge Impulse, the model is optimized to improve both accuracy and efficiency.

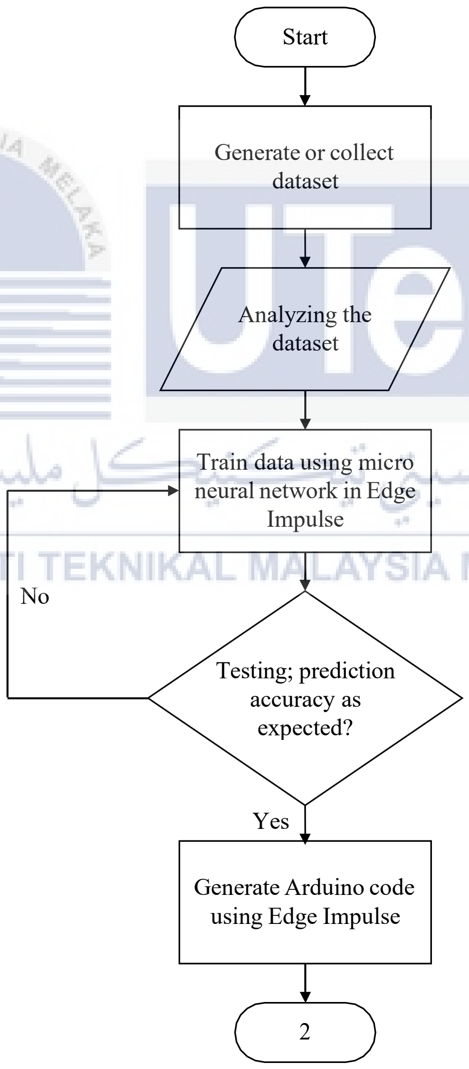

Start
Generate or collect dataset
Analyzing the dataset
Train data using micro neural network in Edge Impulse
No
Testing; prediction accuracy as expected?
Yes
Generate Arduino code using Edge Impulse
2

**Figure 3.2:** Design steps to train the image classification using Edge Impluse.

Once the image classification model is successfully trained, subsequently the model's prediction capability is tested using a separate set of images. Performance metrics such as accuracy, precision are calculated to assess the model's effectiveness. If the desired criteria is not met, further iterations are carried out to further enhance and refine the model. Adjustments to the architecture, hyperparameters, or dataset size are made to improve the accuracy and generalization. In this study, the prediction accuracy is targeted at 90%. After completing the adjustments for the architecture, the Arduino code is generated using Edge Impulse, specifically designed for the ESP32-CAM. Subsequently, the generated code is deployed onto ESP32-CAM, allowing for real-time image classification.

### 3.1.2 Implementation of Image Classification System

Next step in this project is to implement an image classification system using ESP32-CAM, TFT ST7735S, and relay. The design steps to build this system on ESP32-CAM are illustrated in Figure 3.3. First, the connection of ESP32-CAM, TFT ST7735S, and relay are investigated and the circuit diagram is drawn using Fritzing. Subsequently, all these hardware are assembled based on the Fritzing design. To ensure the connection and functionality of the built-in camera and TFT display are working correctly, a program to capture image and display the image on TFT are created as in Appendix 1 and B, respectively. Once the functionality has been validated, the generated code from Edge Impulse is deployed in ESP32-CAM. The real-time image classification is performed to test and validate the functionality of the deployed model.

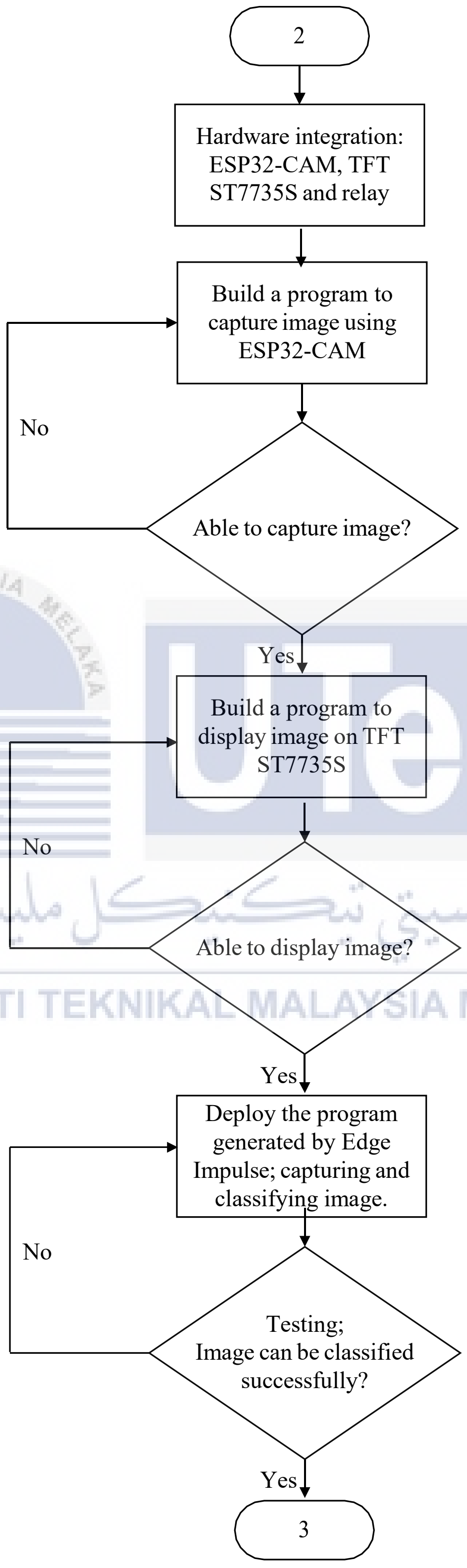
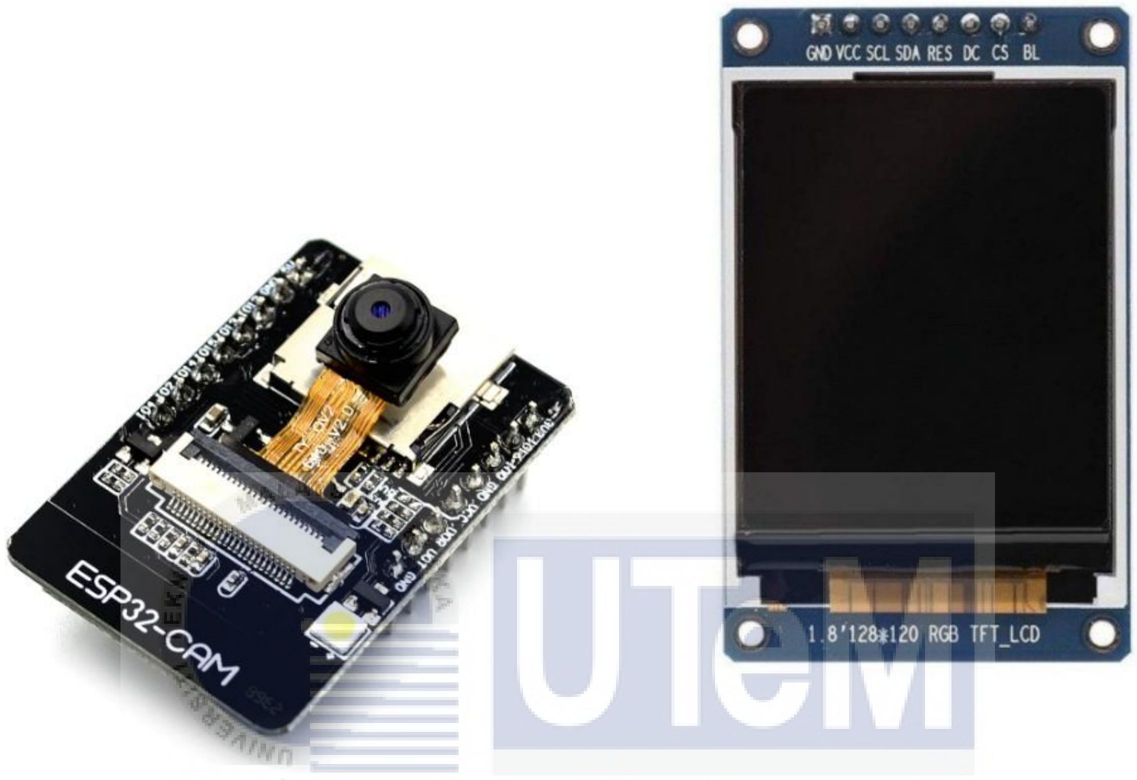

2
Hardware integration: ESP32-CAM, TFT ST7735S and relay
Build a program to capture image using ESP32-CAM
No
Able to capture image?
Yes
Build a program to display image on TFT ST7735S
No
Able to display image?
Yes
Deploy the program generated by Edge Impulse; capturing and classifying image.
No
Testing; Image can be classified successfully?
Yes
3

**Figure 3.3:** Design steps to build the image classification system on ESP32-CAM.

Initially, this project involves assembling the ESP32-CAM with the TFT ST7735S using a relay. Once the hardware setup is complete, the next step is to develop a program for the ESP32-CAM. The ESP32-CAM shown in Figure 3.4a. This program enables the system to capture images and perform classification on them. After implementing the image capture and classification functionality, the subsequent step is to create a program that can display the classification results on the TFT ST7735S screen. Once this program is successfully built, the individual programs for image capture, classification, and display can be combined to form a comprehensive image classification system.

Figure 3.4 depicts the ESP32-CAM and TFT ST7735S display. The TFT ST7735S display was chosen for its compact size, low power consumption, and compatibility with the ESP32-CAM. As shown in Figure 3.4b, the TFT ST7735S has a small form factor, which makes it suitable for integration into the image classification system. Additionally, its low power requirements ensure that it doesn't put excessive strain on the overall power supply of the system. Furthermore, the TFT ST7735S display is compatible with the communication interface of the ESP32-CAM. This compatibility allows for smooth data transfer between the ESP32-CAM and the display, enabling real-time display of the classification results.



**(a)** ESP32-CAM. **(b)** TFT ST7735S.

**Figure 3.4:** ESP32-CAM and TFT display.

### 3.1.3 Prediction Accuracy Evaluation

Approximately 30 iterations of capturing the non-defective and defective IC images (i.e., physical ICs) in real-time is conducted to test the performance of the proposed system. The captured images are analysed by the program and the prediction is made using the trained model that has been deployed in ESP32-CAM. To determine the classification accuracy, the prediction made by the program is compared against the correct labelling of the images.

The accuracy of the deployed model in ESP32-CAM is then compared against the accuracy achieved during the model's training phase in Edge Impulse. If the deployed model demonstrates higher or approximately similar accuracy than the trained model, it indicates positive results. A higher accuracy suggests that the program is function effectively and capable of accurately classifying between non-defective and defective ICs.

## 3.2 Gantt Chart

Figure 3.5 depicts the timeline for PSM I. PSM I focuses on literature review, defining the objective, problem statement, scope of project, methodology, and preliminary analysis. In PSM II, the focus is to develop a fully functional system that can perform a real-time image classification to identify defective and non-defective ICs. Figure 3.6 depicts the time line in PSM II to achieve the above target.

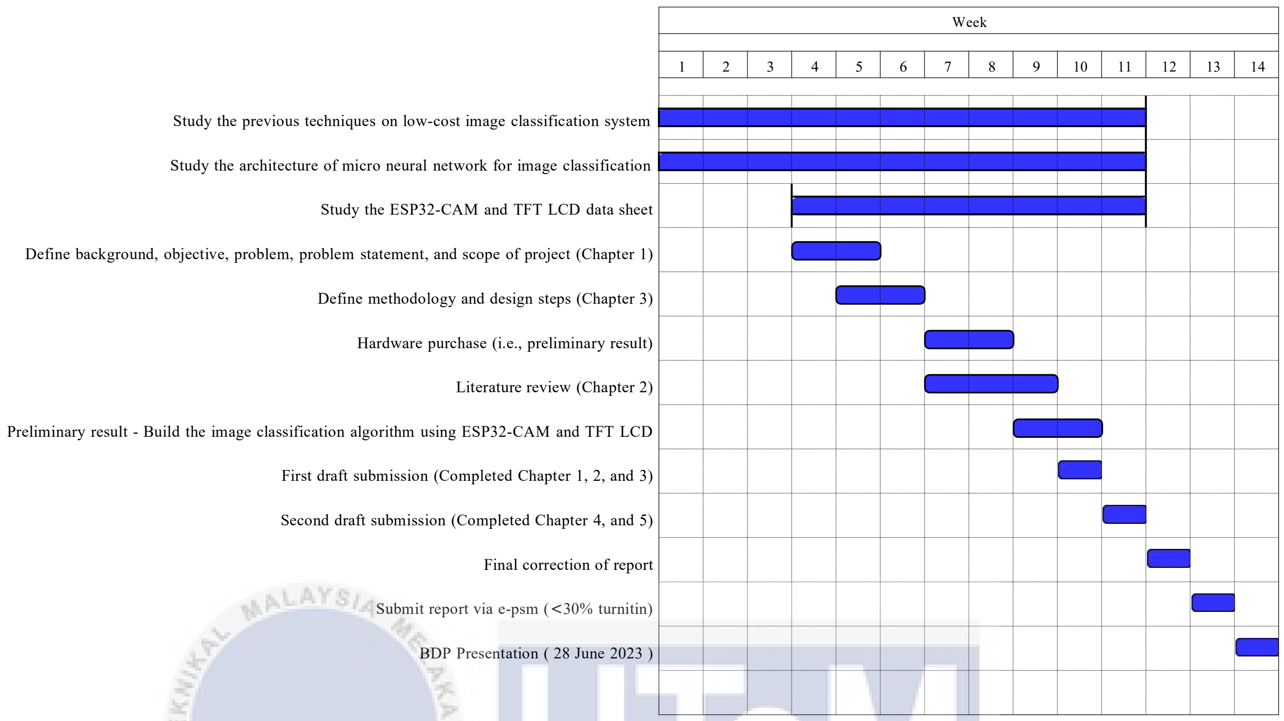


**Figure 3.5:** Timeline for PSM I.

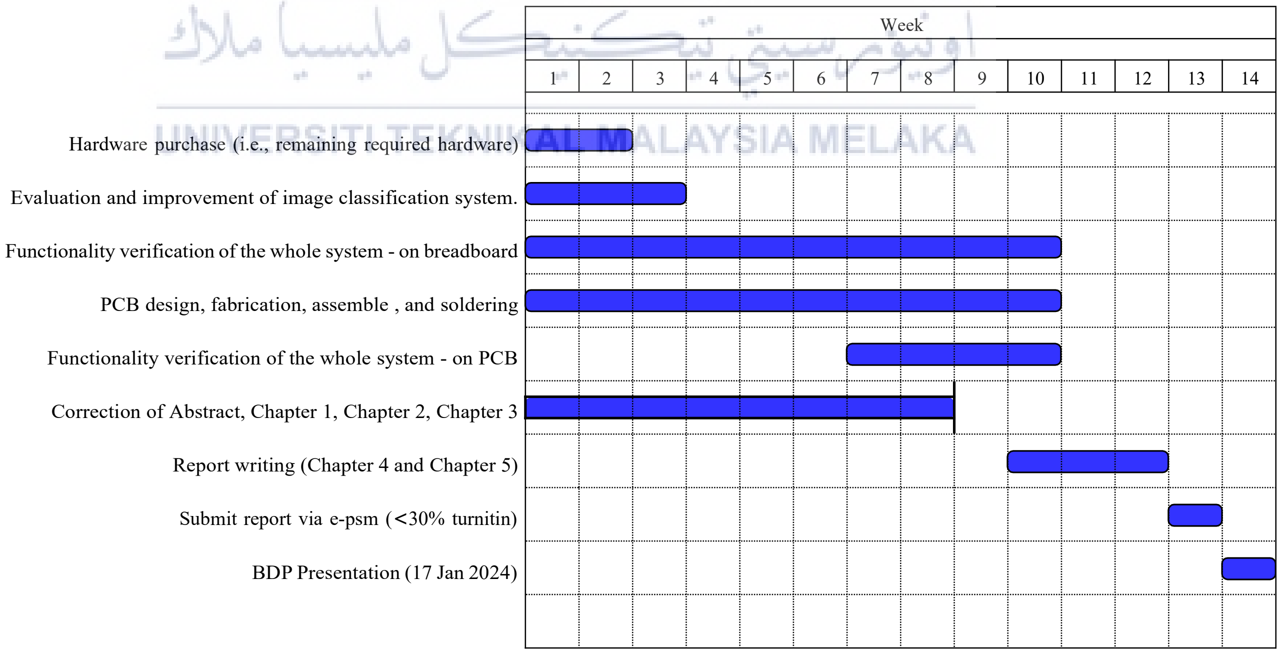


**Figure 3.6:** Timeline for PSM II.

## 3.3 Summary

In this chapter, the method and design steps to build the low-cost image classification system have been described. Edge Impulse is used to enabling the development and deployment of machine learning model on ESP32-CAM. Dataset for defective and non-defective ICs need to be generated. Scratches on IC surface are considered as defect product in our study. Micro-neural network is considered as the algorithm to build the image classification model as it is lightweight and suitable for ESP32-CAM.

# CHAPTER 4

# RESULTS AND DISCUSSION

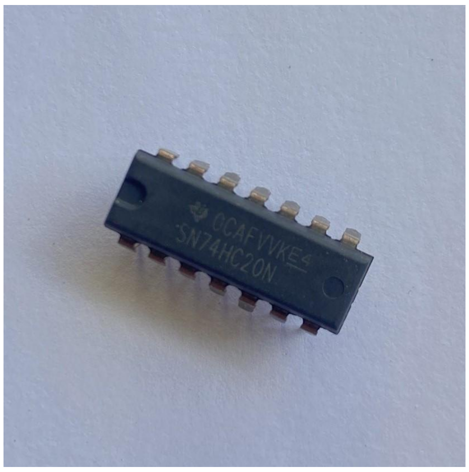
This chapter presents the results and analysis on the development of low-cost image classification system using ESP32-CAM.

## 4.1 Edge Impulse Modelling

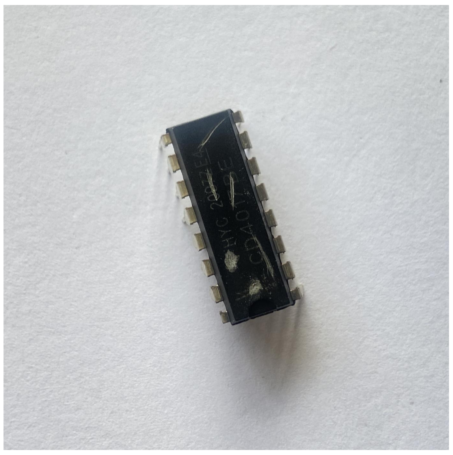
### 4.1.1 Datasets Compilation for Defect and Non-Defect ICs

The data set involved deliberately scratching 8 different physical ICs using blade and 3 of non-defect ICs to create data set totaling of 1,571 images. The data set consists of 1031 images of defect ICs and 540 images of non-defect ICs. Initially, each of the data set was captured using the ESP32-CAM built-in camera. However, the poor quality of ESP32-CAM built-in camera resulting in lower accuracy of the classification system. To solve the issue, the data set were captured using high pixel smartphone's camera to ensure better image quality.

Figure 4.1 illustrates a representative sample image, showcasing both defect and non-defect ICs. The defect ICs that were purposely scratch were used as reference, to simulate the exterior defect that happened during the production process. The generated machine learning model in Edge Impulse is used to differentiate between the defect and non-defect ICs based on the irregularities on the physical structure of the ICs.



**(a)** Non-defect IC.

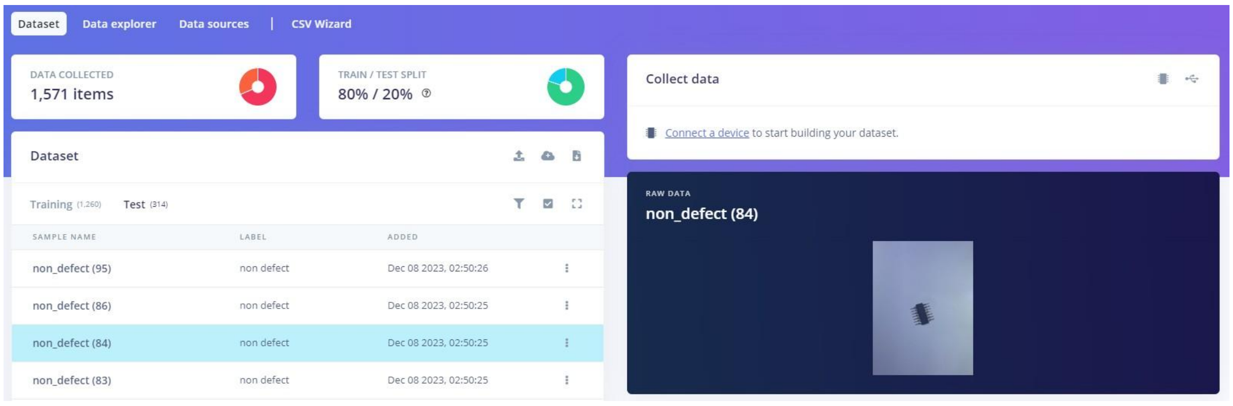


**(b)** Defect IC.

**Figure 4.1:** Example of defect and non-defect ICs.

### 4.1.2 Edge Impulse Image Classification

1571 images of the ICs were uploaded into Edge Impulse in order to create a micro neural network to differentiate the defect and non-defect ICs. The uploaded images were labelled per data, consist two different classes of defect and non-defect ICs. The data set was automatically divided into 80% for training and 20% for testing by the Edge Impulse. A 1260 images comprising 818 data set of defect ICs and 442 data set of non-defect ICs were used to train the micro neural network. Data set for train the micro neural network representating 80% of the overall data set. Afterwards, 20% of the remaining data set were used to test the system accuracy. The testing data set comprises 206 samples of defective ICs and 108 samples of non-defective ICs, resulting in a total of 314 datasets used for verified the system accuracy. Figure 4.2 illustrates the configuration employed in data collection and provides an example of labeled data based on distinct classes.


Dataset | Data explorer | Data sources | CSV Wizard

DATA COLLECTED
1,571 items

TRAIN / TEST SPLIT
80% / 20%

Collect data

Connect a device to start building your dataset.

Dataset

Training (1,260) Test (314)

| SAMPLE NAME | LABEL | ADDED |
|---|---|---|
| non_defect (95) | non defect | Dec 08 2023, 02:50:26 |
| non_defect (86) | non defect | Dec 08 2023, 02:50:25 |
| non_defect (84) | non defect | Dec 08 2023, 02:50:25 |
| non_defect (83) | non defect | Dec 08 2023, 02:50:25 |

RAW DATA
non_defect (84)

**Figure 4.2:** Data acquisition.

Figure 4.3 illustrates the learning block within Edge Impulse, consist of essential stages of the machine learning process. These stages include data collection, data preprocessing, feature extraction, model training, and model evaluation. Each learning block concentrates on a specific facet of the workflow, providing the flexibility to tailor and refine the machine learning models according to the system needs.

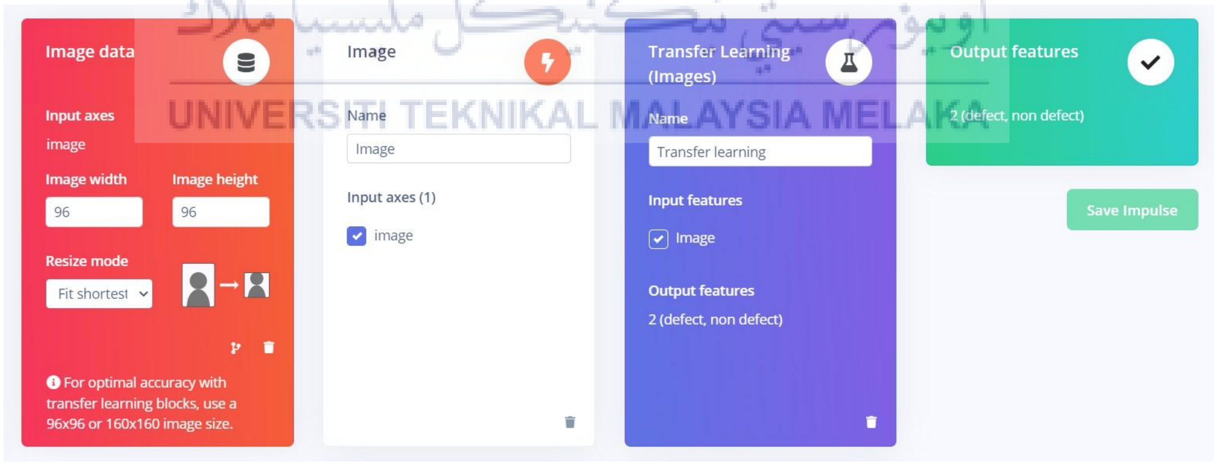


Image data

Input axes
image

Image width: 96
Image height: 96

Resize mode: Fit shortest

For optimal accuracy with transfer learning blocks, use a 96x96 or 160x160 image size.

Image

Name: Image

Input axes (1)
✓ image

Transfer Learning (Images)

Name: Transfer learning

Input features
✓ Image

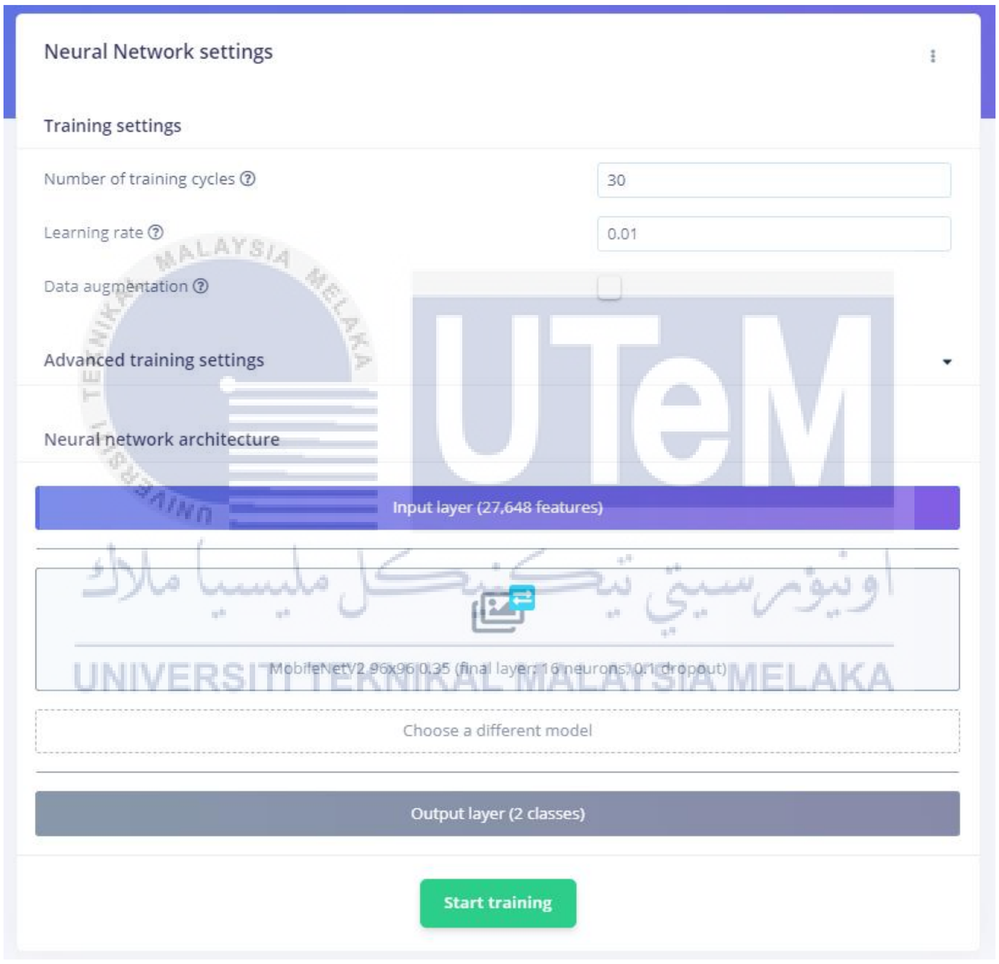
Output features
2 (defect, non defect)

Output features
2 (defect, non defect)

Save Impulse

**Figure 4.3:** Learning block.

To optimize accuracy within transfer learning blocks, images of defect and non-defect ICs underwent resizing to a resolution of 96x96 pixels. Following this, the images were converted to the RGB format, which proves beneficial for tasks involving color analysis and the detailed capture of color information—especially pertinent in the realm of image classification. Additionally, the transfer learning process facilitated the extraction of crucial features from RGB images, allowing for classification based on predefined defect and non-defect classes.


Neural Network settings

Training settings

Number of training cycles: 30

Learning rate: 0.01

Data augmentation: ☐

Advanced training settings

Neural network architecture

Input layer (27,648 features)

MobileNetV2 96x96 0.35 (final layer: 16 neurons, 0.1 dropout)

Choose a different model

Output layer (2 classes)

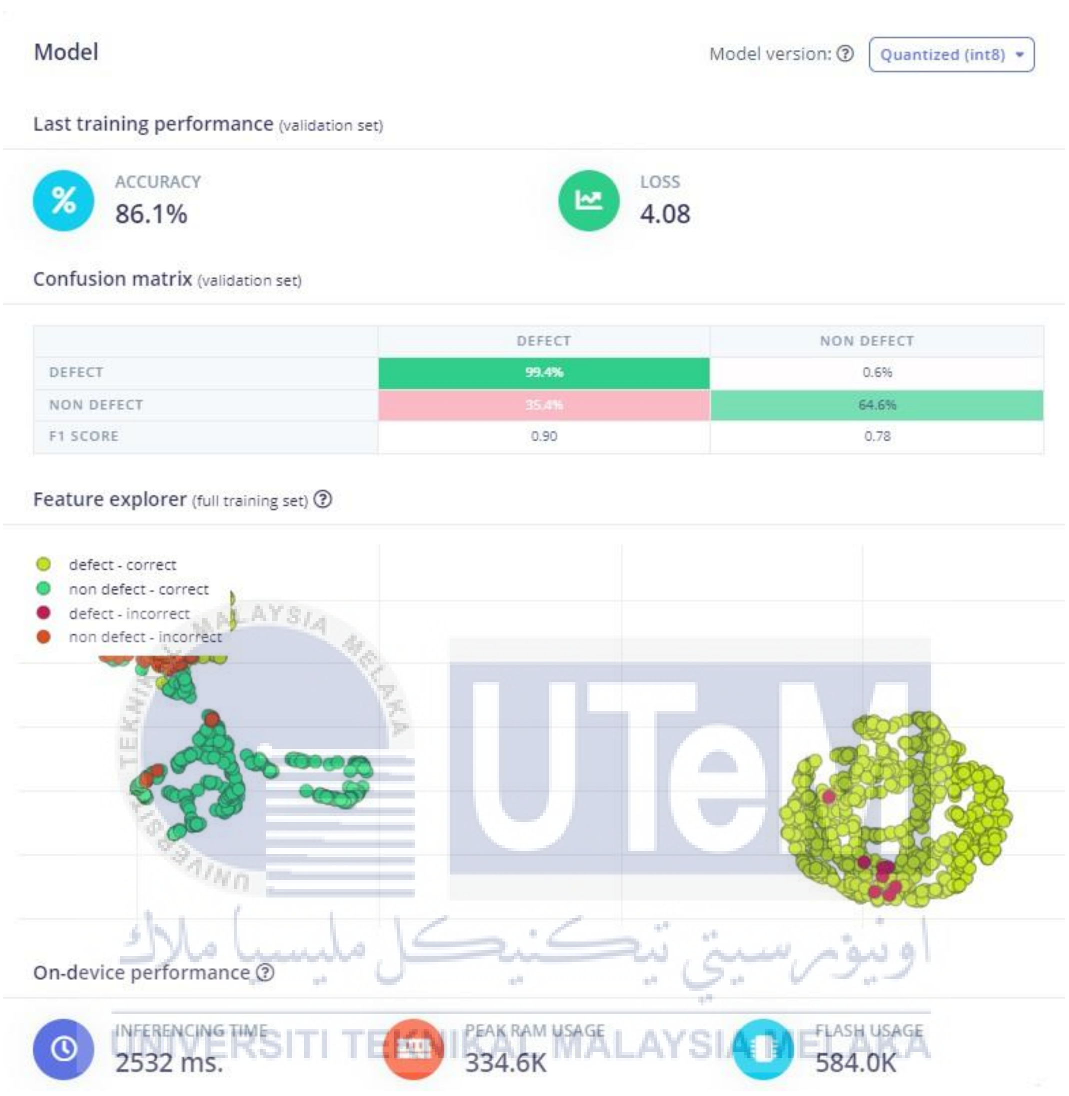
Start training

**Figure 4.4:** Neural network settings.

Figure 4.4 depicts the micro neural network settings in which consists of the configurable options and parameters that establish the architecture and functionality of the neural network model employed in the machine learning tasks. Deploying a micro neural network on the ESP32-CAM for image classification is a judicious choice rooted in several key considerations. The micro neural network, tailored for efficiency and responsiveness, aligns seamlessly with the ESP32-CAM's limitations, optimizing both processing power and memory usage. This decision reflects an overarching commitment to performance, real-time processing capabilities, and energy efficiency, making it an ideal choice for image recognition tasks on the ESP32-CAM.

At the core of this approach is the deliberate decision to limit training cycles to 30. Recognizing the importance of model convergence and computational efficiency, this choice mitigates overfitting risks while accounting for practical constraints like computational resources and time limitations. It represents a strategic compromise, ensuring a time-efficient training process without sacrificing the model's capacity to capture meaningful patterns in the data.

Moreover, the deliberate use of a 0.01 learning rate in neural network training acts as a further measure against overfitting. This lower learning rate facilitates cautious weight adjustments, preventing the model from becoming excessively specialized to the training data. While demanding more epochs for convergence, this measured approach underlines a commitment to learning robust features, contributing to a well-generalized model effective across diverse inputs and minimizing the potential risks of overfitting. The overall settings give ESP32-Cam the potential to enhance the capabilities for image classification.


Model

Model version: ⓘ Quantized (int8)

Last training performance (validation set)

ACCURACY 86.1%

LOSS 4.08

Confusion matrix (validation set)

| | DEFECT | NON DEFECT |
|---|---|---|
| DEFECT | 99.4% | 0.6% |
| NON DEFECT | 35.4% | 64.6% |
| F1 SCORE | 0.90 | 0.78 |

Feature explorer (full training set) ⓘ

● defect - correct
● non defect - correct
● defect - incorrect
● non defect - incorrect

On-device performance ⓘ

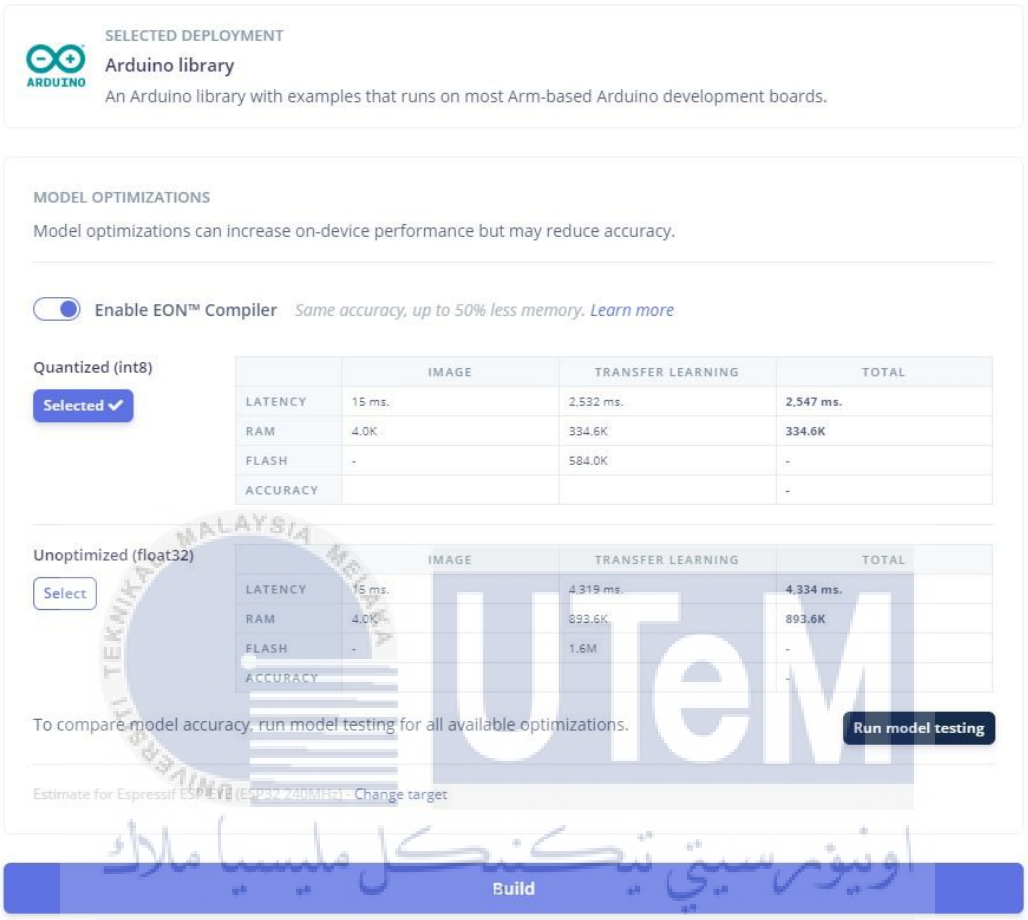
INFERENCING TIME 2532 ms.

PEAK RAM USAGE 334.6K

FLASH USAGE 584.0K

**Figure 4.5:** Accuracy performance of the trained model.

The training results, illustrated in Figure 4.5, encompasses the information and outcomes obtained throughout the machine learning model's training phase. The machine learning model that had done the training then tested with 314 set of data, which splitted earlier on the process. This provides valuable insights into the model's performance and advancements as it assimilates knowledge from the supplied datasets. The trained model achieved 86.1% prediction accuracy with an inference time of 2532 ms. These outcomes signify a satisfactory level of accuracy, particularly for a cost-effective image classification system.


SELECTED DEPLOYMENT

Arduino library

An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

Enable EON™ Compiler *Same accuracy, up to 50% less memory. Learn more*

Quantized (int8) — Selected ✓

| | IMAGE | TRANSFER LEARNING | TOTAL |
|---|---|---|---|
| LATENCY | 15 ms. | 2,532 ms. | **2,547 ms.** |
| RAM | 4.0K | 334.6K | **334.6K** |
| FLASH | - | 584.0K | - |
| ACCURACY | | | - |

Unoptimized (float32) — Select

| | IMAGE | TRANSFER LEARNING | TOTAL |
|---|---|---|---|
| LATENCY | 15 ms. | 4,319 ms. | **4,334 ms.** |
| RAM | 4.0K | 893.6K | **893.6K** |
| FLASH | - | 1.6M | - |
| ACCURACY | | | - |

To compare model accuracy, run model testing for all available optimizations.

Estimate for Espressif ESP-EYE (ESP32 240MHz) - Change target

**Figure 4.6:** Deployment of the trained model.

Subsequently, the machine learning model were deployed into the ESP32-CAM. The setting configuration used are depicted in Figure 4.6, which the machine learning will deployed into an arduino library. The arduino library were chosen due to it's compatibility with the ESP32-CAM. The Quantized(int8) compiler were chosen as the model optimization. This compiler will provide better accuracy and increase on-device performance.

## 4.2 Hardware and Software Intergration

### 4.2.1 Fritzing Design

Figure 4.7 illustrates the entire system, which is powered by a power module capable of providing both 5V and 3.3V outputs. The ESP32-CAM receives power from the 5V output, while the TFT display is powered by the 3.3V output. Table 4.1 illustrates the connection between ESP32-CAM and TFT ST7735S. The script displayed a 120x120 image on the TFT. The push button connected to GPIO 4 is shared with the flash led.

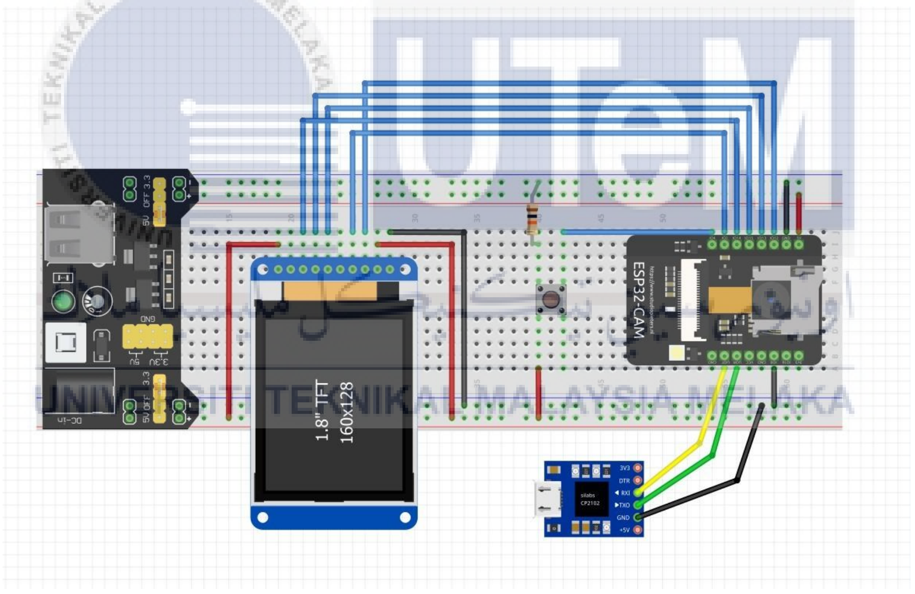


**Figure 4.7:** Circuit diagram.

| TFT pins | ESP32-CAM |
|---|---|
| SCK (SCL) | GPIO 14 |
| MOSI (SDA) | GPIO 13 |
| RESET (RST) | GPIO 12 |
| DC | GPIO 2 |
| CS | GPIO 15 |
| BL (back light) | 3.3V |

**Table 4.1:** TFT pins to ESP32-CAM pins.

Table 4.2 illustrates the connection between USB-TTL pins to the ESP32-CAM. The USB-TTL is used to provide connection between universal serial bus (USB) and serial universal asynchronous receiver/transmitter (UART) interfaces. One important step when uploading the code is to disconnect the power supply to the ESP32-CAM to ensure a stable and safe environment for the upload. In addition, GPIO 0 needs to be connected to ground in order to activate flash mode. This mode prepares the device to receive and process the code being uploaded. By connecting GPIO 0 to GND, an electrical connection is established that triggers the device to enter the flash mode. Once GPIO 0 is securely connected to GND, the power supply to the device can be reconnected. This allows the device to power up while remaining in the flash mode, ready to receive the incoming code. The above steps ensure that the device is in the appropriate mode for code uploading, facilitating a smooth and successful upload process.

| USB-TTL pin | ESP32-CAM |
|---|---|
| Tx | GPIO 3 (UOR) |
| Rx | GPIO 1 (UOT |
| GND | GND |

**Table 4.2:** USB-TTL pins to ESP32-CAM pins.

### 4.2.2 Circuit Assembly

The circuit diagram as discussed in Section 4.2.1 is assembled using the real components. Figure 4.8 depicts the hardware assembly setup that can be used to perform a real-time image classification.

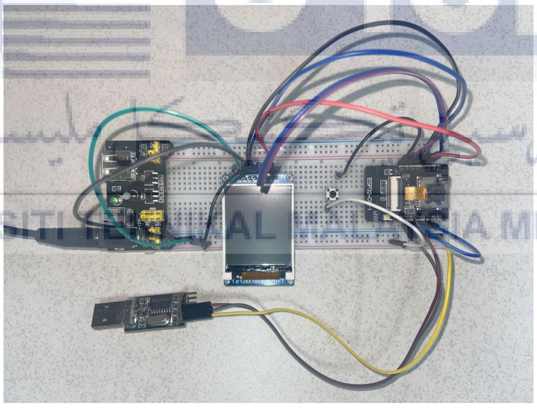
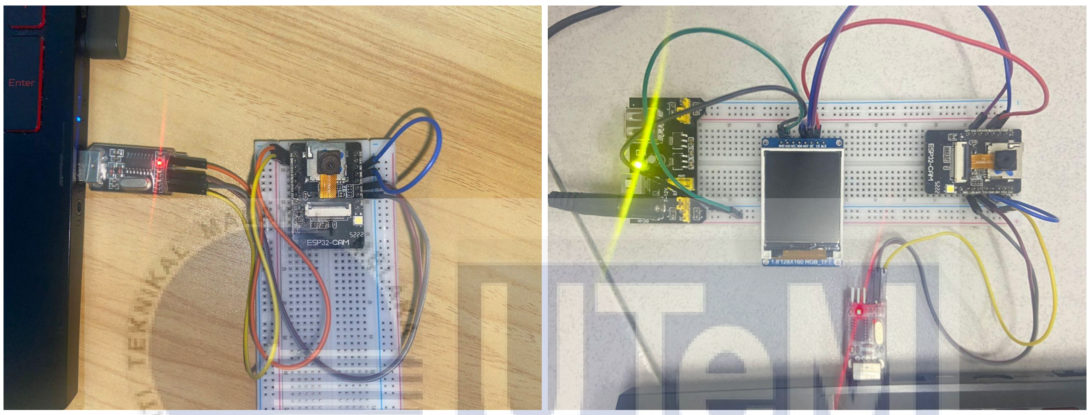


**Figure 4.8:** Hardware setup of ESP32-CAM, TFT ST7735S and relay.

Figure 4.9a illustrates the setup for testing camera on the ESP32-CAM. By using coding in Appendix 1, the testing involves observing the images captured by the camera and evaluating its performance, quality, and characteristics. The testing process is crucial to understand the connectivity and ensure that the built-in camera on ESP32-CAM function as expected. Figure 4.9b illustrates the setup for testing the TFT ST7735S by using coding in Appendix B. Inspecting the TFT ST7735S display serves the purpose of evaluating its functionality, verifying proper operation, and assessing the visual output it provides. This process involves examining the display's performance, image quality, and characteristics to ensure it meets the requirements of the project.

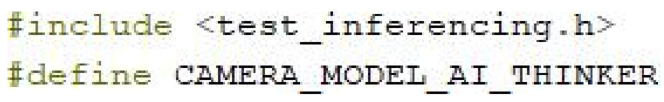


**(a)** Hardware setup to test the built-in camera on ESP32-CAM.

**(b)** Hardware setup to test the TFT ST7735S.

**Figure 4.9:** Hardware setup to test the functionality of built-in camera and TFT component.

The library "test_inferencing.h" shown in the Figure 4.10 were downloaded from the Edge Impulse. Content of the library shown in Appendix D, were the parameters used to classify the ICs. Afterwards, the library that precisely tailored to classify the ICs were intergrated into coding in Appendix C. The coding consists of capture, classify and display of the images.



```
#include <test_inferencing.h>
#define CAMERA_MODEL_AI_THINKER
```

**Figure 4.10:** Library of the micro neural network from the Edge Impulse.

The connection each of the hardware were then made into printed circuit board (PCB) to provide better design of the finishing product. Figure 4.11 shown the completed design of the PCB setup. The PCB design allows for a compact and organized layout of the electronic components, make it much smaller and reducing the overall size of the classification system hardware design. In case of a malfunction or need for repair, the PCB will make it more easier to troubleshoot and figure the cause of the issues.

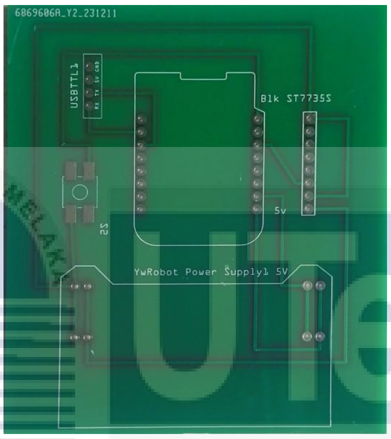


**Figure 4.11:** PCB design of the hardware setup.

The ESP32-CAM, TFT ST7735S and push button then being soldered to the PCB, produced a completed hardware setup which shown in Figure 4.12. Initially, the system capture image of the ICs using the built-in camera of the ESP32-CAM. The images then were classified based on the library that generated from the Edge Impulse. Afterwards, the images were displayed on the TFT ST7735S. The TFT display shown the captured images, percentages of each classes and display the result from the classification. Each of the process were precisely tailored accordance to the development of the low-cost image classification system.

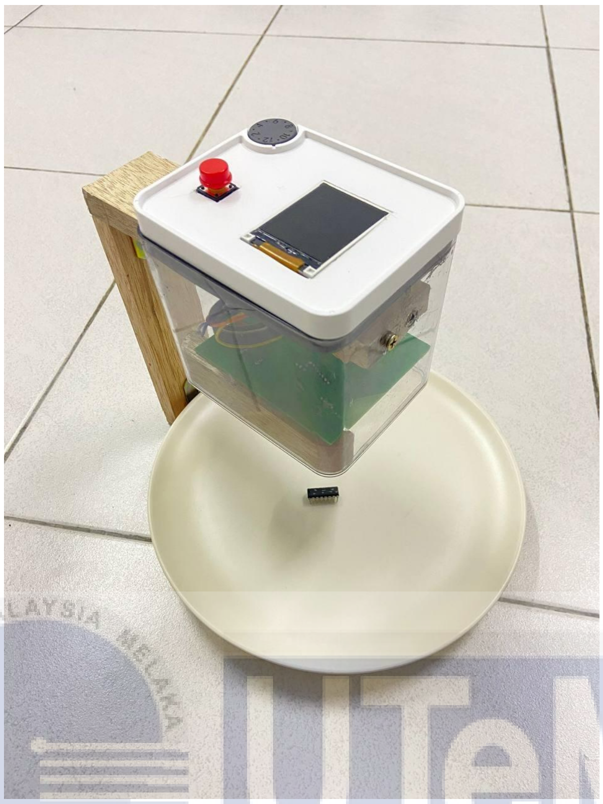


**Figure 4.12:** A prototype of the low-cost image classification system using ESP32-CAM.

### 4.2.3 Real-Time System Accuracy

Embarking the real-time image classification system, accuracy of the micro neural network is a crucial factor which contribute the biggest factor in the system. Table 4.5 depicts the result achieved in classifying the defect and non-defect ICs. The system accuracy was tested using 6 physical ICs, comprising 3 defect and 3 non-defect ICs. The used of 6 physical ICs was due to the limited availability of such components.

A 30 samples of defect ICs were tested resulting into 26 correct classification. The accuracy for the defect ICs were 86.67%, which higher accuracy compared to training performance on the Edge Impulse. However, the accuracy for the samples of non-defect ICs were lower compared to the accuracy of defects ICs. The non-defect ICs only achieved 76.67%. Despite the lower accuracy, it's still gave a satisfactory level due to low quality of the built-in camera of ESP32-CAM. The system only manage to correctly classified 23 out of 30 sample of the non-defect ICs. The overall performance of the system achieved 81.67% of accuracy, which meet the desired performance.

| Samples | Number of data set | Correct Classification | Percentage(%) |
|---|---|---|---|
| Defect | 30 | 26 | 86.67 |
| Non-defective | 30 | 23 | 76.67 |

**Table 4.3:** Real-time performance of the trained model.

The surrounding area emerged as a key factor influencing the accurate classification of the ICs. During system testing, it was observed that brightness played a crucial role. The captured images exhibited varying brightness and contrast, leading to poor image quality. The ESP32-CAM struggled to classify images, particularly when dealing with blurry images.

In essence, the orchestrated sequence of image capture, classification and display images serves as a testament to the system's comprehensive functionality. This process underscores the system's adeptness in effectively classifying ICs based on their defect status, positioning it as a formidable tool in the realm of image analysis and classification.

## 4.3 Advantages and Disadvantages of the Proposed Solution

The ESP32-CAM stands out as a cost-effective and compact solution for image capture and real-time processing. The device can perform computations in real-time, enabling quick and responsive processing of captured images. This real-time processing capability is particularly beneficial for capturing and classifying image, where immediate feedback or analysis are crucial to ensures smooth operation in production lines.

Despite the cost-effective and small design, it’s important to note that the ESP32-CAM have limitations in processing power and memory, which could impact the complexity and size of the image processing tasks it can handle. However, when appropriately optimized and tailored to the project’s requirements, the ESP32-CAM’s cost-effectiveness, small form factor, and real-time processing capabilities make it a compelling choice for a wide range of capturing and classifying the images.

## 4.4 Summary

This chapter discussed the data set collection, training on edge impulse and real-time assessment to identify the capability capturing image. The data set consists of defect and non-defect ICs was used to train on the Edge Impulse and ensuring the functionality of capturing image using ESP32-CAM. All the established configuration also has been provided, ensuring the system works at its optimal performance. The learning results show that the defect and non-defect ICs can be classified successfully with 86.1% prediction accuracy. The real-time image classification system able to successfully classified the defect ICs and non-defect ICs with accuracy of 86.67% and 76.67%, respectively. Moreover, the hardware setup to enable a real-time image classification is also discussed.

# CHAPTER 5

# CONCLUSION AND RECOMMENDATIONS

## 5.1 Conclusion

In this project, a low-cost image classification system using the ESP32-CAM has been developed. The project has been divided into three main objectives, the accomplishment for each objective is described and concluded in this chapter.

The first objective is achieved by training and building the image classification model that able to classify defective and non-defective ICs using suitable machine learning in Edge Impulse. The model were trained using 818 sample images of defective ICs and 442 samples images of non-defective ICs. The image classification model that has been build using Edge Impulse was successfully classified the defect and non-defective ICs with accuracy of 86.1%.

The second objective of the project is achieved by intergrating the trained model in Edge Impulse on ESP32-CAM. The ESP32-CAM and TFT display were intregrated and the functionality was tested using coding in Appendix 1 and B. The coding which fully integrated ESP32-CAM, TFT display and files of micro neural network from Edge Impulse shown in Appendix C.

The third objective is achieved by evaluating the live time prediction accuracy of the image classification system. The entire system which classify a defect and non-defective ICs has been tested using 30 samples of defect and non-defective ICs. Accuracy of 86.67%

has been achieved, which the real-time image classification is slightly lower than the trained model.

## 5.2 Future Works

In the future, our purpose system can be improved as per below:

1. Consider ulitizing more advanced or specialized hardware components that offer improved performance, such as replacing ESP32-CAM camera with higher-resolution cameras.

2. Expanding the system's capability to classify images into more specific classes refers to increasing the number and specificity of the categories or classes that the image classification system can identify. When the system is initially trained, it may be limited to recognizing a certain set of classes. For example, it might be trained to distinguish between scratch surface on IC. However, by diversifying the classification classes, the system can be enhanced to identify a broader range of objects or concepts. For instance, instead of just distinguishing between scratch surface, the system can be trained to recognize different deffects which occurs onto the IC, such as scratch surface due to latching process and etc.

3. Integrate the image classification system with IoT. For example, the system could trigger specific actions or responses based on the classification results, such as controlling other smart devices or sending notifications to a mobile application.

## 5.3 Project Commercialization

The proposed system is suitable to be used in IC manufacturing companies to detect the defective ICs. However, there are remaining things to do in order to make a low-cost image classification system using ESP32-CAM more practical and suitable to be commercialized. One of the ways is to make the proposed system compatible with the machineries in the IC manufacturing line. The IC manufacturing line typically consists of various stages, including wafer fabrication, wafer testing, assembly, packaging, and final testing. Each stage involves specialized equipment and processes to transform silicon wafers into functional ICs. For example, during the final testing stage, the proposed system can capture and analyze the image of the IC products to check for any visual defects or abnormalities, such as cracks, scratches, or misalignment. It can helps to identify faulty IC for rejection, preventing the shipment of defective products. However, the proposed system is not limited to be used in the final testing stage only. It can be used in any critical stage in IC manufacturing for the defect to be detected at the early stage. This improves product quality, reduces warranty claims, and ultimately increases the company's revenues.

# REFERENCES

[1] B. Thormundsson, “Artificial intelligence (AI) market size worldwide in 2021 with a forecast until 2030,” 2021.

[2] Y.-l. Zhang, Z.-j. Guo, and Y.-p. Li, “Research on ship classification based on image processing and fuzzy neural network theory,” in *International Symposium on System and Software Reliability*, 2021, pp. 152–154.

[3] E. Kakkava, N. Borhani, Y. Pu, C. Moser, and D. Psaltis, “Image classification and reconstruction through multimode fibers by deep neural networks,” in *Conference on Lasers and Electro-Optics Pacific Rim*, 2018, pp. 1–2.

[4] J. Xiao and B. Li, “Design of motion recognition system for semiconductor device classification and recognition,” in *International Conference on Intelligent Transportation, Big Data & Smart City*, 2020, pp. 745–748.

[5] Y. Nan and W. Xi, “Classification of press plate image based on attention mechanism,” in *International Conference on Safety Produce Informatization*, 2019, pp. 129–132.

[6] J. Zhuang, G. Mao, Y. Wang, X. Chen, Y. Wang, and Z. Wei, “Classification of wafer backside images via fasterrcnn-based neural network,” in *China Semiconductor Technology International Conference*, 2022, pp. 1–4.

[7] Unicsoft, “How to calculate image recognition app development price,” 2022.

[8] J. McAssey, “What is the cost of artificial intelligence solutions? part 1 of 3. Visual APIs,” 2017.

[9] W. Sriratana, N. Narknam, R. Apichitanon, and N. Tammarugwattana, “Application of webcam for inspection of rice grain quality by using image processing technique,” in *International Conference on Control, Automation and Systems*, 2020, pp. 1134–1139.

[10] Y. Yang, P. Bi, and Y. Liu, “License plate image super-resolution based on convolutional neural network,” in *International Conference on Image, Vision and Computing*, 2018, pp. 723–727.

[11] K. Han, “Artificial neural network for processing fingerprint image noise,” in *International Summer Virtual Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2022, pp. 9–14.

[12] A. Dubey, J. Fuchs, T. Reissland, R. Weigel, and F. Lurz, “Uncertainty analysis of deep neural network for classification of vulnerable road users using micro-doppler,” in *Topical Conference on Wireless Sensors and Sensor Networks*, 2020, pp. 23–26.

[13] G. Geetha, S. Samundeswari, G. Saranya, K. Meenakshi, and M. Nithya, “Plant leaf disease classification and detection system using machine learning,” in *Journal of Physics: Conference Series*, vol. 1712, no. 1, 2020, p. 012012.

[14] P. Prathusha, S. Jyothi, and D. Mamatha, “Enhanced image edge detection methods for crab species identification,” in *International Conference on Soft-computing and Network Security*, 2018, pp. 1–7.

*[15]* A. C. Chua, C. R. B. Mercado, J. P. R. Pin, A. K. T. Tan, J. B. L. Tinhay, E. P. Dadios, and R. K. C. Billones, “Damage identification of selected car parts using image classification and deep learning,” in *International Conference on Humanoid, Nanotechnology,*

*Information Technology, Communication and Control, Environment, and Management*, 2021, pp. 1–5.

[16] U. Sanver, E. Yavuz, and C. Eyupoglu, “An image processing application to detect faulty bottle packaging,” in *Conference of Russian Young Researchers in Electrical and Electronic Engineering*, 2017, pp. 986–989.

[17] A. Sarkar, S. Chakraborty, and B. Roy, “Image processing based product label quality control on fmcg products,” in *International Conference on Energy, Power and Environment: Towards Sustainable Growth*, 2015, pp. 1–5.

[18] C. G. Spinola, J. Canero, G. Moreno-Aranda, J. M. Bonelo, and M. Martin-Vazquez, “Real-time image processing for edge inspection and defect detection in stainless steel production lines,” in *International Conference on Imaging Systems and Techniques*, 2011, pp. 170–175.

[19] Z. Zhu, J. Luo, and Z. Shen, “Automatic remote sensing image registration based on sift descriptor and image classification,” in *International Conference on Geoinformatics*, 2010, pp. 1–5.

[20] T. Mane, G. Raut, A. Pethe, I. Patil, K. Mundada, and A. Iyer, “Identification and classification of industrial elements using artificial intelligence and image processing techniques,” in *International Conference on Emerging Smart Computing and Informatics*, 2021, pp. 165–169.

[21] T. N. Michael McIntyre, “Method and system for recognizing scratch patterns on semiconductor wafers,” 1999.

# APPENDICES

## Appendix A: Coding for Camera Test

```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

const char* ssid = "**********";
const char* password = "**********";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  // ... (camera configuration parameters)

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t* s = esp_camera_sensor_get();
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);
    s->set_brightness(s, 1);
    s->set_saturation(s, -2);
  }

  if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  delay(10000);
}
```

## Appendix B: Coding for Graphic Test TFT ST7735S

```
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <Adafruit_ST7789.h>
#include <SPI.h>

// Pin Definitions
#if defined(ARDUINO_FEATHER_ESP32)
  #define TFT_CS         15
  #define TFT_RST        12
  #define TFT_DC         2
#elif defined(ESP8266)
  #define TFT_CS         4
  #define TFT_RST        16
  #define TFT_DC         5
#else
  #define TFT_CS        10
  #define TFT_RST        9
  #define TFT_DC         8
#endif

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

float p = 3.1415926;

// Function Declarations
void setup();
void loop();
void testlines(uint16_t color);
void testdrawtext(char *text, uint16_t color);
void testfastlines(uint16_t color1, uint16_t color2);
void testdrawrects(uint16_t color);
void testfillrects(uint16_t color1, uint16_t color2);
void testfillcircles(uint8_t radius, uint16_t color);
void testdrawcircles(uint8_t radius, uint16_t color);
void testtriangles();
void testroundrects();
void tftPrintTest();
void mediabuttons();

void setup() {
  Serial.begin(9600);
  tft.initR(INITR_BLACKTAB);

  displayInitializationTests();
  basicShapeTests();
  textTests ();

  Serial.println("Initialization complete");
}

void loop() {
  tft.invertDisplay(true);
  delay(500);
  tft.invertDisplay(false);
  delay(500);
}

void displayInitializationTests() {
  uint16_t time = millis();
  tft.fillScreen(ST77XX_BLACK);
  time = millis() - time;
  delay(500);
}

void basicShapeTests() {
  tft.fillScreen(ST77XX_BLACK);
  testlines(ST77XX_YELLOW);
  delay(500);

  tft.fillScreen(ST77XX_BLACK);
  testfastlines(ST77XX_RED, ST77XX_BLUE);
  delay(500);

  tft.fillScreen(ST77XX_BLACK);
  testdrawrects(ST77XX_GREEN);
  delay(500);

  tft.fillScreen(ST77XX_BLACK);
  testfillrects(ST77XX_YELLOW, ST77XX_MAGENTA);
  delay(500);

  tft.fillScreen(ST77XX_BLACK);
  testfillcircles(10, ST77XX_BLUE);
  testdrawcircles(10, ST77XX_WHITE);
  delay(500);

  tft.fillScreen(ST77XX_BLACK);
  testroundrects();
  delay(500);

  tft.fillScreen(ST77XX_BLACK);
  testtriangles();
  delay(500);
}

void textTests() {
  tft.fillScreen(ST77XX_BLACK);
  tftPrintTest();
  delay(4000);

  tft.setCursor(0, 0);
  tft.fillScreen(ST77XX_BLACK);
  tft.setTextColor(ST77XX_WHITE);
  tft.setTextSize(0);
  tft.println("Hello World!");
  tft.setTextSize(1);
  tft.setTextColor(ST77XX_GREEN);
  tft.print(p, 6);
  tft.println(" Want pi?");
  tft.println(" ");
  tft.print(8675309, HEX); // print 8,675,309 out in HEX!
  tft.println(" Print HEX!");
  tft.println(" ");
  tft.setTextColor(ST77XX_WHITE);
  tft.println("Sketch has been");
  tft.println("running for: ");
  tft.setTextColor(ST77XX_MAGENTA);
  tft.print(millis() / 1000);
  tft.setTextColor(ST77XX_WHITE);
  tft.print(" seconds.");
}

void testlines(uint16_t color) {
  // Drawing lines to test display
}

void testdrawtext(char *text, uint16_t color) {
  // Drawing text to test display
}
```

## Appendix C: Coding for Real-Time Image Classification System

```
#include <test_inferencing.h>
#define CAMERA_MODEL_AI_THINKER

#include "img_converters.h"
#include "image_util.h"
#include "esp_camera.h"
#include "camera_pins.h"

#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library for ST7735

#define TFT_SCLK 14 // SCL
#define TFT_MOSI 13 // SDA
#define TFT_RST  12 // RES (RESET)
#define TFT_DC    2 // Data Command control pin
#define TFT_CS   15 // Chip select control pin
                    // BL (back light) and VCC -> 3V3

#define BTN       4 // button (shared with flash led)

dl_matrix3du_t *resized_matrix = NULL;
ei_impulse_result_t result = {0};

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK, TFT_RST);

// setup
void setup() {
  Serial.begin(115200);

  // button
  pinMode(4, INPUT);

```

```
// TFT display init
tft.initR(INITR_GREENTAB);
tft.setRotation(0);
tft.fillScreen(ST77XX_BLACK);

// cam config
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
config.frame_size = FRAMESIZE_240X240;
config.jpeg_quality = 10;
config.fb_count = 1;

// camera init
esp_err_t err = esp_camera_init(&config);
```

```
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, 0); // lower the saturation
  }

  Serial.println("Camera Ready!...(standby, press button to start)");
  tft_drawtext(4, 4, "Standby", 1, ST77XX_BLUE);
}

// main loop
void loop() {

  // wait until the button is pressed
  while (!digitalRead(BTN));
  delay(100);

  // capture a image and classify it
  String result = classify();

  // display result
  Serial.printf("Result: %s\n", result);
  tft_drawtext(4, 120 - 16, result, 2, ST77XX_GREEN);
}

// classify labels
String classify() {
```

```

// run image capture once to force clear buffer
// otherwise the captured image below would only show up next time you pressed the  button!
capture_quick();

// capture image from camera
if (!capture()) return "Error";
tft_drawtext(4, 4, "Classifying...", 1, ST77XX_CYAN);

Serial.println("Getting image...");
signal_t signal;
signal.total_length = EI_CLASSIFIER_INPUT_WIDTH * EI_CLASSIFIER_INPUT_WIDTH;
signal.get_data = &raw_feature_get_data;

Serial.println("Run classifier...");
// Feed signal to the classifier
EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false /* debug */);
// --- Free memory ---
dl_matrix3du_free(resized_matrix);

// --- Returned error variable "res" while data object.array in "result" ---
ei_printf("run_classifier returned: %d\n", res);
if (res != 0) return "Error";

// --- print the predictions ---
 ei_printf("Predictions (DSP: %d ms., Classification: %d ms., Anomaly: %d ms.): \n",
           result.timing.dsp, result.timing.classification, result.timing.anomaly);
int index;
float score = 0.0;
for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
  // record the most possible label
  if (result.classification[ix].value > score) {
    score = result.classification[ix].value;
```

```
      index = ix;
    }
    ei_printf("    %s: \t%f\r\n", result.classification[ix].label, result.classification[ix].value);
    tft_drawtext(4, 12 + 8 * ix, String(result.classification[ix].label) + " " + String(result.classi
  }

#if EI_CLASSIFIER_HAS_ANOMALY == 1
  ei_printf("    anomaly score: %f\r\n", result.anomaly);
#endif

  // --- return the most possible label ---
  return String(result.classification[index].label);
}

// quick capture (to clear buffer)
void capture_quick() {
  camera_fb_t *fb = NULL;
  fb = esp_camera_fb_get();
  if (!fb) return;
  esp_camera_fb_return(fb);
}

// capture image from cam
bool capture() {

  Serial.println("Capture image...");
  esp_err_t res = ESP_OK;
  camera_fb_t *fb = NULL;
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    return false;
  }

  // --- Convert frame to RGB888  ---
  Serial.println("Converting to RGB888...");
  // Allocate rgb888_matrix buffer
  dl_matrix3du_t *rgb888_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
  fmt2rgb888(fb->buf, fb->len, fb->format, rgb888_matrix->item);

  // --- Resize the RGB888 frame to 96x96 in this example ---
  Serial.println("Resizing the frame buffer...");
  resized_matrix = dl_matrix3du_alloc(1, EI_CLASSIFIER_INPUT_WIDTH, EI_CLASSIFIER_INPUT_HEIGHT, 3);
  image_resize_linear(resized_matrix->item, rgb888_matrix->item, EI_CLASSIFIER_INPUT_WIDTH,  EI_CLASSI

  // --- Convert frame to RGB565 and display on the TFT ---
  Serial.println("Converting to RGB565 and display on TFT...");
  uint8_t *rgb565 = (uint8_t *) malloc(240 * 240 * 3);
  jpg2rgb565(fb->buf, fb->len, rgb565, JPG_SCALE_2X); // scale to half size
  tft.drawRGBBitmap(0, 0, (uint16_t*)rgb565, 120, 120);

  // --- Free memory ---
  rgb565 = NULL;
  dl_matrix3du_free(rgb888_matrix);
  esp_camera_fb_return(fb);

  return  true;
}

int raw_feature_get_data(size_t offset, size_t out_len, float *signal_ptr) {

  size_t pixel_ix = offset * 3;
  size_t bytes_left = out_len;
  size_t out_ptr_ix = 0;

  // read  byte  for  byte
  while (bytes_left != 0) {
    // grab the values and convert to r/g/b
    uint8_t r, g, b;
    r = resized_matrix->item[pixel_ix];
    g = resized_matrix->item[pixel_ix + 1];
    b = resized_matrix->item[pixel_ix + 2];

    // then convert to out_ptr format
    float pixel_f = (r << 16) + (g << 8) + b;
    signal_ptr[out_ptr_ix] = pixel_f;

    // and go to the next pixel
    out_ptr_ix++;
    pixel_ix += 3;
    bytes_left--;
  }

  return 0;
}

// draw test on TFT
void tft_drawtext(int16_t x, int16_t y, String text, uint8_t font_size, uint16_t color) {
  tft.setCursor(x, y);
  tft.setTextSize(font_size); // font size 1 = 6x8, 2 = 12x16, 3 = 18x24
  tft.setTextColor(color);
  tft.setTextWrap(true);
  tft.print(strcpy(new char[text.length() + 1], text.c_str()));
}
```

## Appendix D: Coding for Test Inferencing

```

#ifndef _EI_CLASSIFIER_MODEL_VARIABLES_H_
#define _EI_CLASSIFIER_MODEL_VARIABLES_H_

#include <stdint.h>
#include "model_metadata.h"

#include "tflite-model/tflite_learn_5_compiled.h"
#include "edge-impulse-sdk/classifier/ei_model_types.h"
#include "edge-impulse-sdk/classifier/inferencing_engines/engines.h"

const char* ei_classifier_inferencing_categories[] = { "defect", "non defect" };

uint8_t ei_dsp_config_3_axes[] = { 0 };
const uint32_t ei_dsp_config_3_axes_size = 1;
ei_dsp_config_image_t ei_dsp_config_3 = {
    3, // uint32_t blockId
    1, // int implementationVersion
    1, // int length of axes
    "RGB" // select channels
};

const size_t ei_dsp_blocks_size = 1;
ei_model_dsp_t ei_dsp_blocks[ei_dsp_blocks_size] = {
    { // DSP block 3
        27648,
        &extract_image_features,
        (void*)&ei_dsp_config_3,
        ei_dsp_config_3_axes,
        ei_dsp_config_3_axes_size
    }
};
```

```
const ei_config_tflite_eon_graph_t ei_config_tflite_graph_5 = {
    .implementation_version = 1,
    .model_init = &tflite_learn_5_init,
    .model_invoke = &tflite_learn_5_invoke,
    .model_reset = &tflite_learn_5_reset,
    .model_input = &tflite_learn_5_input,
    .model_output = &tflite_learn_5_output,
};

const ei_learning_block_config_tflite_graph_t ei_learning_block_config_5  = {
    .implementation_version = 1,
    .block_id = 5,
    .object_detection = 0,
    .object_detection_last_layer = EI_CLASSIFIER_LAST_LAYER_UNKNOWN,
    .output_data_tensor  = 0,
    .output_labels_tensor = 1,
    .output_score_tensor  = 2,
    .quantized = 1,
    .compiled = 1,
    .graph_config = (void*)&ei_config_tflite_graph_5
};

const size_t ei_learning_blocks_size = 1;
const ei_learning_block_t ei_learning_blocks[ei_learning_blocks_size] = {
    {
        &run_nn_inference,
        (void*)&ei_learning_block_config_5,
        EI_CLASSIFIER_IMAGE_SCALING_NONE,
    },
};

const ei_model_performance_calibration_t ei_calibration = {
    1, /* integer version number */
    false, /* has configured performance calibration */
    (int32_t)(EI_CLASSIFIER_RAW_SAMPLE_COUNT / ((EI_CLASSIFIER_FREQUENCY > 0) ? EI_CLASSIFIER_FREQUEN
    0.8f, /* Default threshold */
    (int32_t)(EI_CLASSIFIER_RAW_SAMPLE_COUNT / ((EI_CLASSIFIER_FREQUENCY > 0) ? EI_CLASSIFIER_FREQUEN
    0   /* Don't use flags */
};

const ei_impulse_t impulse_321138_3 = {
    .project_id = 321138,
    .project_owner = "Muhammad adni",
    .project_name = "test",
    .deploy_version = 3,

    .nn_input_frame_size = 27648,
    .raw_sample_count = 9216,
    .raw_samples_per_frame = 1,
    .dsp_input_frame_size = 9216 * 1,
    .input_width = 96,
    .input_height = 96,
    .input_frames = 1,
    .interval_ms = 1,
    .frequency = 0,
    .dsp_blocks_size = ei_dsp_blocks_size,
    .dsp_blocks = ei_dsp_blocks,

    .object_detection = 0,
    .object_detection_count = 0,
    .object_detection_threshold = 0,
    .object_detection_last_layer = EI_CLASSIFIER_LAST_LAYER_UNKNOWN,
    .fomo_output_size = 0,

    .tflite_output_features_count = 2,
    .learning_blocks_size = ei_learning_blocks_size,
    .learning_blocks = ei_learning_blocks,

    .inferencing_engine = EI_CLASSIFIER_TFLITE,

    .sensor = EI_CLASSIFIER_SENSOR_CAMERA,
    .fusion_string = "image",
    .slice_size = (9216/4),
    .slices_per_model_window = 4,

    .has_anomaly = 0,
    .label_count = 2,
    .calibration = ei_calibration,
    .categories = ei_classifier_inferencing_categories
};

const ei_impulse_t ei_default_impulse = impulse_321138_3;

#endif // _EI_CLASSIFIER_MODEL_METADATA_H_
```