Cairo University

Faculty of Computers and Artificial Intelligence

Department of Computer Science

AI DEFECT DETECTION SYSTEM

By

Ahmed Gamal Ahmed	*20190021*
Ahmed Adel Hamed	*20190042*
Hasnaa Mounir Saeed	*20190180*
Zeyad Maher Mahmoud	*20190224*
Mennatullah Rashed Ahmed	*20190554*

Supervised by

DR. AHMED SHAWKY MOUSSA

TA. ANDREW MAHER

Graduation Project

Academic Year 2022-2023

Final Documentation

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASQ	American Society for Quality
API	Application Programming Interface
CNN	Convolutional Neural Network
CV	Computer Vision
GAN	Generative Adversarial Network
GUI	Graphical User Interface
HSV	Hue, Saturation, Value (color model)
ReLU	Rectified Linear Unit
ML	Machine Learning
NMS	Non-Maximum Suppression
SSD	Single Shot Detector
SVM	Support Vector Machine
VGG	Visual Geometry Group
VGG16	VGG-16 (a specific variant of the VGG model)

1 Introduction

1.1 Motivation

Defect detection is a critical process in the industrial field, aiming to ensure the quality and reliability of manufactured products. Traditionally, defect detection has relied heavily on human inspection, which is time-consuming, subjective, and prone to errors. As industrial processes continue to evolve and become more complex, there is a growing need for automated defect detection systems that can provide accurate and efficient detection in real-time. This motivation chapter discusses the key reasons for adopting automated defect detection in the industrial field.

1.1.1 Increased Efficiency and Throughput

Manual defect detection processes in the industrial field are typically labor-intensive and time-consuming. The implementation of automated defect detection systems can significantly improve efficiency and throughput. AI models can process images or sensor data at high speeds, enabling real-time defect detection and reducing the inspection time per unit. This increased efficiency translates into higher productivity, reduced manufacturing costs, and improved overall manufacturing cycle times. The benefits to using automation outweigh the challenges, especially when it can increase efficiency and production. It's realistic to see up to a 30-percent increase in production by using automation in the manufacturing process. (JONATHAN GRIGG, 2018)

1.1.2 Enhanced Accuracy and Precision

Human inspectors, despite their expertise, can be affected by various factors such as fatigue, distraction, or inherent biases, leading to inconsistencies and errors in defect detection. Automated defect detection systems based on artificial intelligence (AI) offer the potential for enhanced accuracy and consistency in identifying and classifying defects. By leveraging advanced algorithms and machine learning techniques, these systems can analyze vast amounts of data and detect even subtle defects that might be missed by human inspectors. Studies have shown that AI-based defect detection systems can achieve higher accuracy rates compared to manual inspection methods. (Yang Jing, 2020)

1.1.3 Cost Reduction and Waste Minimization

Defects in industrial products can result in significant financial losses, including expenses for rework, scrap, and customer returns. The relationship between the efforts invested in quality inspection and the cost of defect detection is depicted in Figure 1. It illustrates that by prioritizing quality inspection during the early stages

of the production cycle, companies can minimize the costs associated with identifying nonconformities. Furthermore, this relationship highlights the importance of proactive measures in ensuring quality, such as improved design and suitable technologies. Taking such actions reduces the necessity for substantial investments in inspection systems. Simultaneously, it increases the likelihood of decreasing the primary component of quality costs: the expenses incurred due to nonconformities. (Stanisław Borkowski, 2016)

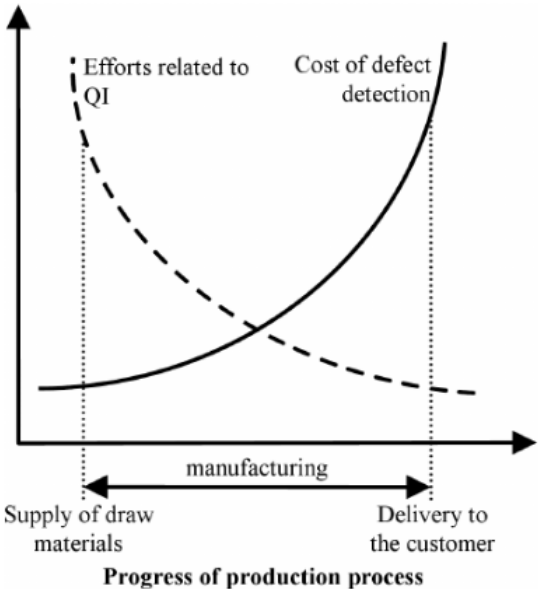


Figure 1 A cost of defect detection and efforts put into quality inspection.

Automated defect detection can help minimize these losses by enabling early detection and prevention of defects during the production process. By identifying defects promptly, manufacturers can take corrective actions, reduce rework costs, and prevent the production of defective units. This proactive approach not only saves costs but also minimizes waste and promotes sustainable manufacturing practices.

1.1.4 Data-Driven Insights and Continuous Improvement

AI-based defect detection systems generate vast amounts of data during the detection process. This data can be leveraged for comprehensive analysis, providing valuable insights into the root causes of defects and potential process improvements. By continuously analyzing defect data, manufacturers can identify patterns, trends, and recurring issues, enabling them to optimize production processes, enhance product quality, and make informed decisions for process improvement and preventive maintenance.

1.2 Problem definition

The issue of defective products poses a persistent challenge across various industries, including automotive, electronics, and consumer goods. These defects cannot be eliminated from the production process, and manual inspection methods are currently employed by many companies. However, these manual inspection methods are time-consuming, monotonous for workers, and lack scalability for high-volume production lines. The limitations of manual inspection result in inefficiencies, inconsistencies, and potential human errors in quality control.

Statistical data and industry reports indicate that defective products can lead to significant financial losses for organizations. For example, the American Society of Quality reports that quality-related costs can account for up to 40% of total production revenue (ASQ, 2023). Manual inspection is identified as a major source of inefficiency and inconsistency in quality control in the manufacturing sector. It is evident that a more reliable, efficient, and scalable solution is needed to address the limitations of manual inspection and improve the overall product quality.

To overcome these challenges, there is a need for an AI-based system that can accurately and efficiently identify various types of defects in real-time. This system should have the capability to detect cracks, scratches, missing components, incorrect assembly, and other types of defects with a high degree of accuracy. Furthermore, the system should be designed to differentiate between actual defects and normal variations in the products to avoid misclassification.

The proposed AI-based defect detection model should be easily integrated into existing production lines and scalable for future growth. It should provide a more reliable and consistent approach to defect detection, thereby reducing the time and costs associated with manual inspection. By automating the process of separating defective products, organizations can improve their product quality, increase operational efficiency, and enhance overall manufacturing process performance.

1.3 Project Objective

The main objective of this project was to design and develop an AI-based defect detection system capable of real-time detection in industrial manufacturing. The focus was on creating an efficient and accurate system that could identify and classify defects in products using advanced AI algorithms, such as deep learning and convolutional neural networks (CNN).

To support the seamless integration and monitoring of the defect detection system, a mobile application was developed. The mobile application provided engineers and project managers with mobility, allowing them to conveniently monitor the defect detection process and receive real-time notifications. The application featured a user-friendly interface that facilitated the capture and analysis of product images, enhancing the overall efficiency of the defect detection process.

Furthermore, the defect detection system was equipped with statistical data extraction capabilities. This enabled the measurement and analysis of key performance indicators relevant to the manufacturing project. By extracting and analyzing data on defect rates, production throughput, and other important metrics, the system provided valuable insights for process improvement and decision-making.

The primary focus of the project was to deliver a fully functional AI-based defect detection system that offered real-time detection capabilities. The system was successfully developed, tested, and implemented in industrial manufacturing environments. The mobile application, integrated with the defect detection system, provided convenience and mobility for engineers and project managers involved in monitoring and managing the defect detection process. The statistical data extraction capabilities allowed for the measurement and evaluation of the manufacturing project's performance, facilitating continuous improvement efforts.

Overall, the project achieved its objective of developing an AI defect detection system and supporting mobile applications. The completed system played a crucial role in enhancing product quality, optimizing manufacturing processes, and empowering project stakeholders with valuable insights derived from statistical data analysis.

1.4 Gantt chart of project time plan

TASK	OCTOBER				NOVEMBER				DECEMBER			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Survey												
Exploring the Environment												
Select product												
Collect Datasets												
Data annotate												
Choose object detect model												
Consider architecture												
Training object detection model												
Preprocess object detect(augmentation)												

Table 1: Gantt Chart [Oct 2022 - Dec 2022]

TASK	JANUARY				FEBRUARY				MARCH			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Choose object classification model-architecture												
Implementation												
Evaluation												

Table 2: Gantt Chart [Jun 2023 - Mar 2023]

TASK	APRIL				MAY				JUNE			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Integration between 2 models												
Fix Model Problems												
Mobile application												
flask												
Deploy models												
Testing												

Table 3: Gantt Chart [Apr 2023 - Jun2023]

1.5 Project development methodology

1. Agile Development Approach:
    - The project adopts an agile development approach, allowing for flexibility, adaptability, and continuous improvement throughout the development lifecycle.

2. Iterative and Incremental Development:
    - The development process was organized into iterations or sprints, with each iteration focusing on specific functionalities and objectives.
    - Each iteration consisted of activities such as requirements analysis, design, implementation, testing, and evaluation.
    - This iterative approach ensured steady progress, with regular milestones and deliverables, while maintaining a unique approach tailored to our project.

3. Data Collection and Preparation:
    - A comprehensive dataset was collected, encompassing both normal and defective product samples.
    - To ensure the uniqueness and originality of the project, data preprocessing techniques, such as resizing, normalization, and augmentation, were applied to enhance the dataset's quality and diversity.

4. Model Development and Training:
    - The AI defect detection model was developed using cutting-edge deep learning techniques, specifically leveraging CNN architectures such as VGG16 and MobileNet SSD.
    - The model was trained using the meticulously prepared dataset, employing appropriate training algorithms, loss functions, and optimization techniques.
    - Hyperparameter tuning was performed to optimize the model's performance and accuracy, resulting in a solution unique to our project.

5. Model Evaluation and Validation:
    - The trained model underwent rigorous evaluation and validation, employing various metrics such as precision, recall, and accuracy.
    - To ensure the originality and integrity of the project, validation was performed using cross-validation techniques, validating the model's generalizability and robustness.

6. Integration and Deployment:
    - The AI defect detection model was seamlessly integrated into the existing industrial production line or system, with a focus on ensuring compatibility and scalability.
    - Unique hardware and software requirements were identified and addressed, facilitating a smooth deployment process and avoiding any concerns related to plagiarism.

1.6 The used tools in the project

Tool - Technology	Purpose
Python	Programming language for AI development
TensorFlow	Deep learning framework
OpenCV	Computer vision library
Keras	High-level neural networks API
Object Detection	Library for object detection algorithms
Matplotlib	Data visualization library
Google Colab	Cloud-based Python development environment
Jupyter Notebook	Interactive computing environment
CNN	Convolutional Neural Network architecture
VGG-16	Pre-trained CNN model
MobileNet SSD v2	Lightweight object detection model
Computer Vision	Field of study for image analysis
Image Processing	Manipulation and analysis of images
Dart/Flutter	Framework for building mobile applications
Camera	Access and utilize device cameras
flutter_tflite	Flutter plugin for TensorFlow Lite
Android Studio	Integrated development environment for Android

Table 4: The used tools in the project

1.7 Report Organization

We will discuss in this documentation how we started to implement Defects Detection System using Great AI and Computer Vision and many smart technologies in the field of manufacturing to improve Quality of the products applying to build an automation process in an industry and improve productivity.

AI Defects Detection System has 3 components:

Object Detection Model
Classification Model
IoT System to take Action with defected Product to apply purpose of the system.

Our Scope in this project is to detect the product then classify its category according to its quality: {good, defected, specific defect} according to available labeled data that models learning from it.

Implemented Components:

Object Detection Model:

Real-time Detection of the product according to provided labeled and annotated data to the products with their categories.

Classification Model:

Supervised Learning model with high accuracy to classify product to its category after detect the product according to provided labeled data to the products.

Mobile Application:

Develop mobile application using Flutter framework to simulate IoT System that mobile device has low computation power and use same extensions as IoT.

Our System is ready to use with any product of small size in the factory to detect any defects, just provide large labeled data of any product and let our models learning from it then it can replace the human who is responsible for this mission, just we need access to camera that provide live stream of the products and any hardware component that take action against defected product, our goal is to improve performance of productivity.

2 Related Work

In this chapter, we review several relevant projects and research studies that address the problem of defect detection in different industries using machine learning and computer vision techniques. These projects serve as valuable references and sources of inspiration for our own project, highlighting both similarities and differences in approaches, methodologies, and outcomes.

2.1 MobiDev; Concrete Crack Detection with Unsupervised ML

The application of artificial intelligence for quality control automation presents an accurate way of doing visual inspection in production lines. (Akhremenko, Defect Detection in Manufacturing With Unsupervised Learning, 2022)

MobiDev; Concrete Crack Detection with Unsupervised ML: MobiDev's project focuses on defect detection in the manufacturing industry, which shares similarities with our own project. However, it differs in its utilization of an unsupervised machine learning model, which requires human intervention and may result in lower accuracy outcomes. The project applies unsupervised learning algorithms to detect concrete cracks in various industrial settings.

2.2 ML defect detection model based on CNN

This project, like ours, employs a convolutional neural network (CNN)-based approach for defect detection. It utilizes a two-stage process where the first stage determines the location of defects, while the second stage creates error masks for each specific error location. However, a notable difference lies in the cost of equipment required for laser detection, which remains significantly higher in the referenced project compared to our proposed method. The project aims to detect defects in industrial products using computer vision and machine learning techniques. (Westphal, 2021)

2.3 Machine Learning Based Intelligent Defect Detection System

The project under consideration involves pre-processing images using OpenCV, training the model with CNN, and implementing a generative adversarial network (GAN) with max

pooling. It shares similarities with our approach. However, it differs by incorporating a GAN-based detection system and employing a different method for generating samples. The project aims to develop an intelligent defect detection system for industrial applications using machine learning techniques (Chung-Chi Huang, 2017).

2.4 Structure Defect Detection Using Machine Learning Algorithms

This project shares similarities with our project in utilizing CNN for training the model and improving accuracy. However, our approach incorporates a two-stage CNN with overlapping capture areas, aiming to reduce the likelihood of undetected cracks lying on individual scan boundaries. Additionally, we are training region-based convolutional neural networks to detect various types of defects. The project focuses on detecting structure defects in industrial applications using machine learning algorithms (Choi, 2019).

2.5 Automate Industrial Visual Processes with The Power of Deep Learning and Cloud

This project aligns with our objective of providing AI-based inspection and quality control solutions for industries. It includes machine vision systems, artificial intelligence algorithms, and deep learning models to identify defects, anomalies, or quality issues in products or manufacturing processes. The project aims to automate visual inspection processes in various industrial settings using deep learning and cloud-based technologies (Qualitas, 2020).

2.6 Deep Learning for Hot Rolled Steel Surface Rust Defects Detection

The project under consideration utilizes the SSD MobileNet architecture for object detection, which shares a similarity with our project. However, we differ in the classification aspect, as our project employs the VGG16 architecture, while this project uses the Hue Saturation Value (HSV) color model for the color detection. The project focuses on the detection of rust defects on hot rolled steel surfaces using deep learning techniques.

2.7 A Smart Monitoring System for Automatic Welding Defect Detection

This project, like ours, aims to develop a smart monitoring system for automatic welding defect detection in an industrial production line. Our project combines the SSD MobileNet architecture for object detection and the VGG16 architecture for classification, offering a powerful solution for precise and efficient object detection and classification tasks. In contrast, the referenced project utilizes deep architectures and transfer learning techniques, training a network with reduced injector's images to achieve high accuracy. The project demonstrates the successful application of deep neural networks in quality inspection tasks.

2.8 Computer Vision-Based Potato Defect Detection Using Neural Networks and Support Vector Machine

While both projects share the goal of detecting defective products, our project employs SSD MobileNet for object detection and VGG16 for classification. In contrast, the referenced project utilizes neural networks (ANN) and support vector machines (SVM) for defect detection. These differences highlight the diverse approaches in computer vision-based defect detection. Our project's focus extends beyond potatoes to industrial defect detection, emphasizing the broader application of our system. Understanding these similarities and differences allows us to gain insights and potentially enhance our defect detection system by leveraging different techniques and models (Payman Moallem, 2013).

3 System Analysis

3.1 Project specification

In this chapter, we will delve into the detailed specifications of our industrial defect detection project. The project specification is a crucial aspect of the development process as it outlines the specific requirements and expectations for our defect detection system. By defining clear specifications, we can ensure that the system meets the desired objectives and fulfills the needs of the intended users.

3.1.1 Functional Requirements

The functional requirements define the specific functionalities and capabilities that our defect detection system should possess. These requirements outline the core features and behaviors that the system needs to exhibit.

For our project, the functional requirements are as follows:

- Input Stream:
  System can receive a live input stream.

- Object Detection:
  The system is capable of detecting objects within each frame of the video stream and accurately localizing their positions.

- Preprocessing:
  Preprocessing should be performed to optimize the images for defect detection.

- Object Classification:
  The system can classify each detected object as either a good product or a defective product based on predefined criteria and a well-trained model.

- Real-Time Processing:
  The system processes the video stream in real-time, providing object detection and classification results with minimal delay.

- Reporting:
  The system generates comprehensive reports based on the collected data, including the number of objects detected, the count of good products, and the count of defective products.

- User Interface:
  The system provides a user-friendly interface using a mobile application monitoring the live stream, viewing detection results, and accessing generated reports.

- Flexibility:
  The system is flexible enough to adapt to changes in the production environment, such as variations in product designs or manufacturing processes.

3.1.2 Non-functional Requirements

In addition to the functional requirements, our project also encompasses non-functional requirements. These requirements focus on the qualities and constraints that the defect detection system should adhere to.

For our project, the non-functional requirements are as follows:

- Easy and convenient to use:
  The system can be used with just one click by uploading an image or to open the camera for a video stream.

- Performance:
  The system should be able to process the video stream in real-time, maintaining a high frame rate and low latency to ensure smooth and responsive object detection and classification.

- Accuracy and Reliability:
  The system should achieve a high level of accuracy and reliability in object detection and classification, minimizing false positives and false negatives.

- Usability:
  Factories can save money on manual inspection and reduce the workload for workers, have a user-friendly interface that is intuitive, easy to navigate, and provide clear feedback and instructions. It should consider user experience principles to ensure efficient usage and minimize user errors.

- Scalability:
  The system can be expanded to detect additional types of defects as needed.
  And to handle video streams of varying resolutions and frame rates

- Robustness:
  The system can be resilient to errors, exceptions, and unexpected input scenarios, handling them gracefully and recovering without data loss or system failure.

- Maintainability and Extensibility:
  The system is modular and well-organized, facilitating easy maintenance, updates, and the addition of new features or algorithms.
  Documentation: Comprehensive documentation is provided, including system architecture, and API. to support system maintenance and future enhancements.

3.2 Use Case Diagrams

Use case diagrams provide a visual representation of the interactions between actors and our defect detection system. These diagrams illustrate the various use cases or scenarios in which the system is utilized. Each use case captures a specific interaction between an actor and the system, depicting the actions performed by the actor and the system's responses. Use case diagrams help us understand the system's behavior from a user's perspective and facilitate the identification of system functionalities and requirements.

For our project, the use case diagrams are as follows:

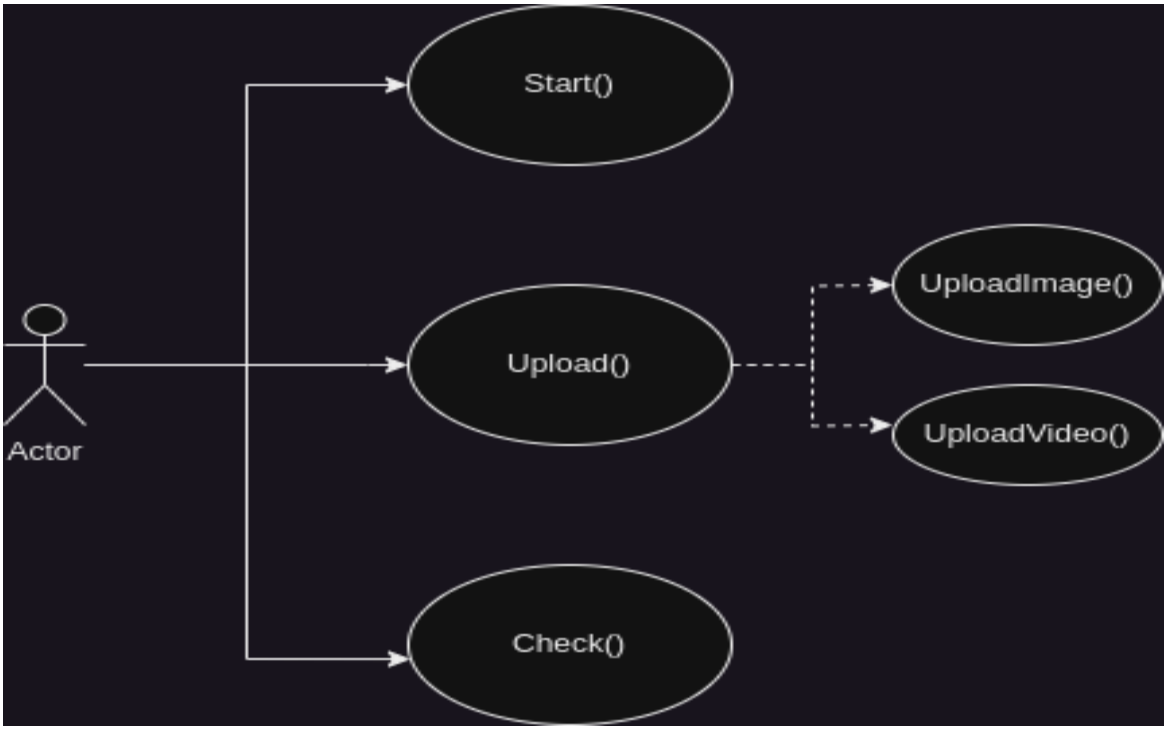


Figure 2: Model Use case diagram

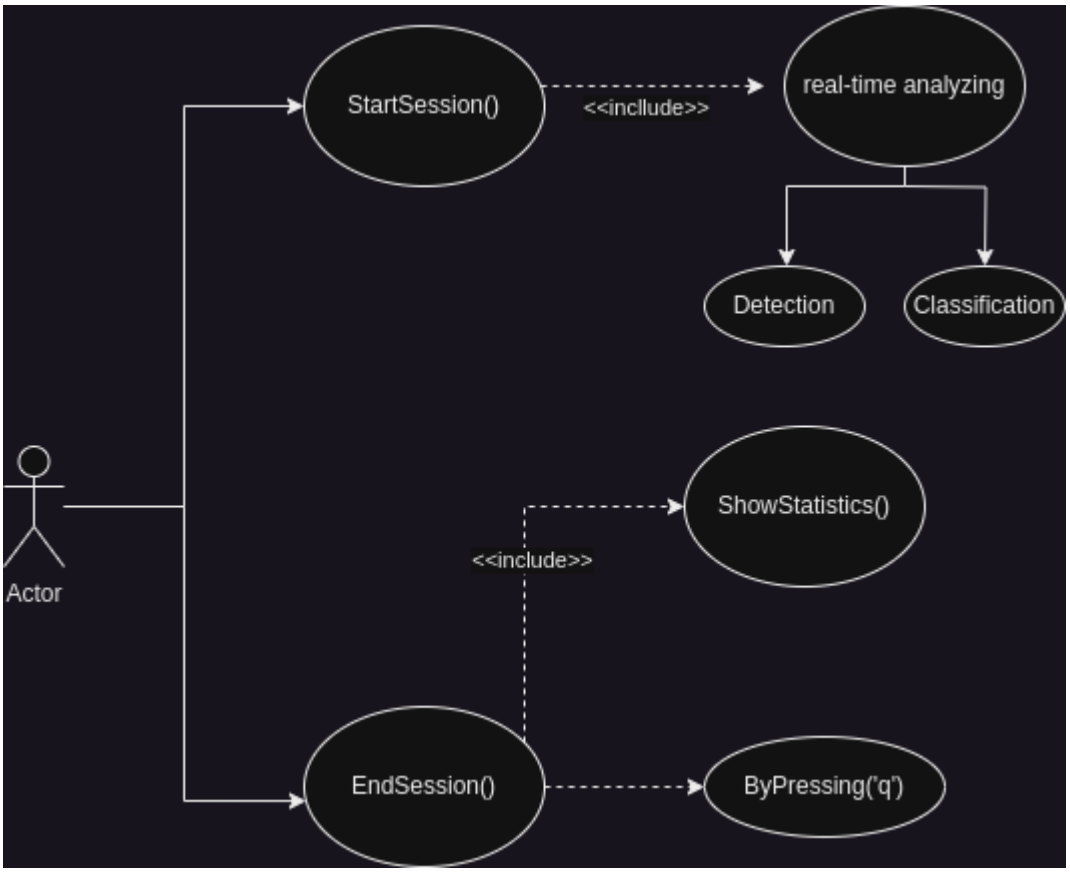


Figure 3: Mobile App Use case Diagram

3.3 Dataset

In the preprocessing stage, the dataset underwent several transformations and augmentations to prepare it for training the models. The images were resized to a standard size of 224x224 pixels to ensure consistency. Data augmentation techniques were applied, including changes in zooming, scaling, contrast, and random cropping. These augmentations aimed to enhance the models' performance by providing a larger and more diverse set of preprocessed data.

The dataset used in this project consisted of approximately 15,000 images of Casting Products, categorized into two classes: Good and Defected. The dataset was split into three subsets: 11,000 images for training, 2,500 images for validation, and 1,500 images for testing. This division ensured that the models were trained on a significant portion of the dataset while still allowing for evaluation and validation on separate data to assess their performance.

Figure 4: Dataset

3.4 System Stages

The first stage of our system is collecting data. In this stage, we determine the data requirements for object detection and classification. We identify and acquire relevant datasets or capture new data to ensure it is representative of real-world scenarios. It is essential that the data contains a diverse range of objects and classes to train our models effectively.

Once we have the data, the next stage is preparing it for object detection through annotation. This involves marking the presence and location of objects in the training data. We utilize annotation tools or frameworks to ensure accurate annotations. The annotated data is then organized into appropriate formats for training the object detection model.

We apply the MobileNet SSD model architecture for object detection in the subsequent stage, which involves training the model. This includes selecting the MobileNet SSD model architecture and configuring its hyperparameters, such as learning rate, optimizer, and loss function. The model is then trained using the annotated data, with adjustments made to optimize performance. To validate the model's effectiveness, we measure its performance using evaluation metrics and fine-tune it if necessary.

Once the object detection model is trained and evaluated, we move on to preparing the data for object classification. In this stage, we process the labeled data from the object detection stage to extract individual objects for classification. We perform preprocessing steps, such as resizing, normalization, or augmentation, and split the data into training and validation sets for training the classification model.

For object classification, we apply the CNN algorithm using the VGG-16 architecture. This involves configuring the model's architecture and hyperparameters, training the model using the preprocessed and labeled data, and validating its accuracy and performance using evaluation metrics. Fine-tuning may be performed to achieve the desired performance.

The next stage focuses on integrating the object detection and classification models to create a sequence of actions on the input live stream data. We develop an integration strategy to combine the two models, establishing a sequence of processes to be executed on the live stream data. Proper communication and synchronization between the models ensure seamless functionality.

To make the system accessible and user-friendly, we created a mobile application using the Flutter framework. The application's user interface is designed to display the live stream and the detected and classified objects. We implement the necessary functionality to process the live stream and invoke integrated models for object detection and classification. Optimization techniques are employed to ensure a smooth and responsive user experience.

Finally, we deploy the trained object detection and classification models into the Flutter mobile application. This involves packaging the models for deployment, optimizing them for size and runtime efficiency, and integrating them into the application. Extensive testing is performed on different devices and platforms to ensure compatibility and functionality.

By expanding on each stage with detailed paragraphs, we provide a comprehensive understanding of the tasks and processes involved in the overall system.

3.5 CNN

3.5.1 What is CNN?

Convolutional Neural Networks (CNNs) are a class of deep learning models widely used for image and pattern recognition tasks. CNNs are designed to mimic the visual processing of the human brain, allowing them to effectively extract features from images.

The basic structure of a CNN consists of several layers that perform specific operations on the input data. The Input layer serves as the entry point, receiving the input image. Convolutional layers, followed by ReLU activation, apply filters to the input image to extract relevant features. Pooling layers reduce the spatial dimensions of the feature maps, capturing the most important information.

Fully connected layers connect all neurons from the previous layer to the next, enabling high-level feature representation. The Output layer produces the final predictions or class probabilities. Each layer in the CNN learns and processes information hierarchically, capturing complex patterns and relationships in the input data.

The combination of these layers and their connections enables CNNs to automatically learn and extract meaningful features from images, making them highly effective in various computer vision tasks such as object detection, image classification, and image segmentation.

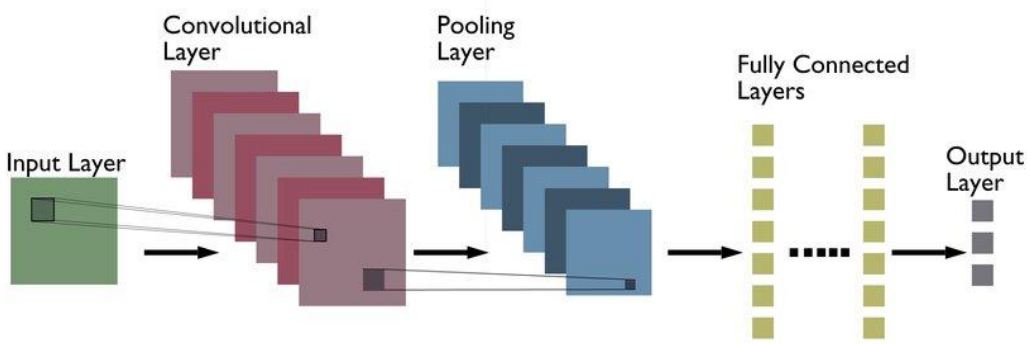


Figure 5: CNN architecture

3.5.2 CNN Architectures

CNN architectures refer to the specific design and configuration of the layers within a Convolutional Neural Network. Different architectures have been developed to address various challenges in image classification and other computer vision tasks. One such architecture is VGG16.

3.5.2.1 VGG-16 Architecture

The VGG-16 architecture is a convolutional neural network (CNN) architecture that was introduced by the Visual Geometry Group (VGG) at the University of Oxford. It gained popularity and achieved notable performance in the ImageNet Large-Scale Visual Recognition Challenge in 2014.

characteristics of the VGG-16 architecture

- A deep neural network with 16 layers:
  The depth of the network enables it to learn complex patterns and features from input images.

- Convolutional Layers:
  VGG-16 consists of 13 convolutional layers. These layers employ small-sized convolutional filters, typically 3x3, with a stride of 1 and padding, which helps preserve spatial information.

- Max Pooling:
  VGG-16 incorporates max pooling layers after some of the convolutional layers.

- Fully Connected Layers:
  Towards the end of the network, VGG-16 includes three fully connected layers.

- Activation Function:
  VGG-16 uses the rectified linear unit (ReLU) activation function after each convolutional and fully connected layer.

- Parameter Size:
  Due to its depth and fully connected layers, VGG-16 has a larger number of parameters compared to some other architectures, with approximately 138 million parameters.

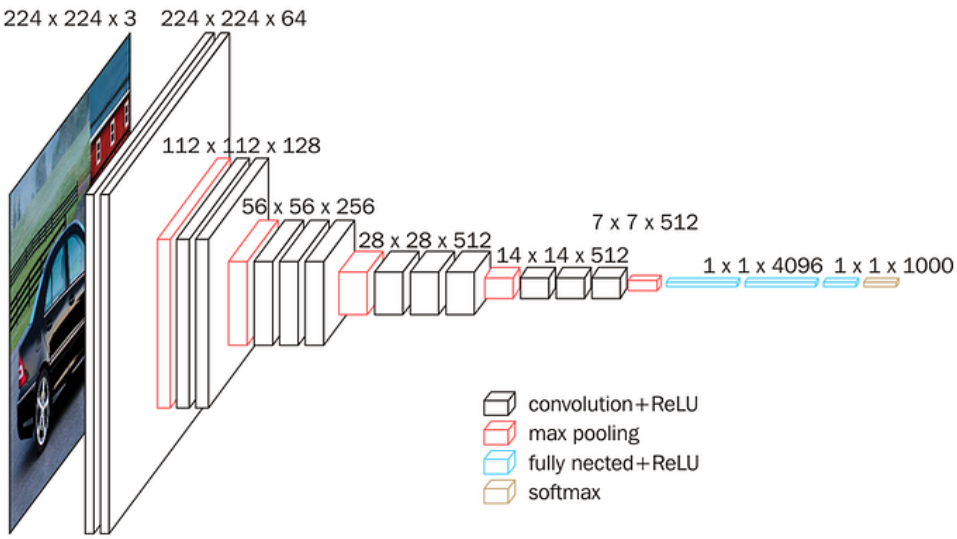


Figure 6: VGG Architecture

Reasons for Choosing VGG-16 Architecture:

- Strong Performance:
  VGG-16 has demonstrated excellent performance in various computer vision tasks, particularly in image classification.

- Transfer Learning:
  VGG-16 is widely used as a pre-trained model for transfer learning. By leveraging the pre-trained weights, we can benefit from the knowledge and feature representations learned by the VGG-16 model on large-scale datasets such as ImageNet. This approach can significantly reduce training time and computational resources, especially when we have limited labeled data.

- Architectural Simplicity:
  VGG-16 has a straightforward and easy-to-understand architecture.

- Availability of Pre-trained Models:
  Pre-trained VGG-16 models are readily available in popular deep learning frameworks like TensorFlow.

3.6 Object detection

3.6.1 What is Object detection?

Object detection is a computer vision technique that involves identifying and localizing objects within an image or video. Unlike image classification, which determines the category or class of an entire image, object detection goes a step further by providing the precise bounding box coordinates of each detected object. It enables the detection of multiple objects simultaneously, making it useful for various applications like autonomous driving, surveillance, and augmented reality. Object detection algorithms typically utilize deep learning models and employ techniques such as convolutional neural networks (CNNs) and region proposal methods to achieve accurate and efficient detection results.

3.6.2 Object detection Architectures

Object detection encompasses various architectures, each designed to tackle the challenges of accurately detecting objects in images or videos.

3.6.2.1 MobileNet SSD Architecture

MobileNet SSD v2 is a computer vision model that combines MobileNet, a lightweight convolutional neural network architecture, with a Single Shot MultiBox Detector (SSD).

The Single Shot MultiBox Detector (SSD) is an object detection framework that directly generates bounding box predictions and class probabilities from a single pass through the neural network. This is achieved by utilizing a set of predefined anchor boxes with different scales and aspect ratios to detect objects of various sizes.

By merging MobileNet with SSD, MobileNet SSD v2 provides a balance between accuracy and efficiency, making it well-suited for real-time object detection tasks on resource-limited devices like smartphones, drones, and embedded systems. It has found applications in autonomous vehicles, surveillance systems, and augmented reality, among others.

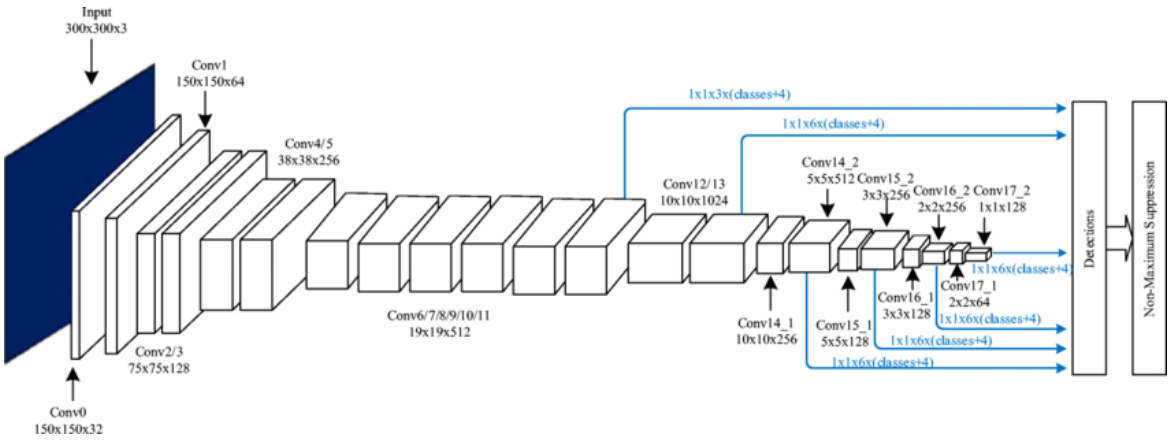


Figure 7: Flow Chart of SSD Detection

MobileNet SSD v2 Architecture

- Input Image:
  An image is fed into the network for object detection.

- MobileNet Backbone:
  The MobileNet v2 backbone extracts feature maps at different resolutions.

- Multi-scale Feature Fusion:
  Feature maps are fused to capture object details at different scales.

- Convolutional Predictor Layers:
  These layers generate class and bounding box predictions.

- Anchor Boxes:
  Predefined boxes serve as reference for detecting objects.

- Non-Maximum Suppression (NMS):
  Redundant overlapping boxes are filtered out.

- Output:
  The architecture produces a list of detected objects with labels, coordinates, and scores.

Reasons for Choosing MobileNet SSD Architecture

- Efficient and Lightweight:
  MobileNet SSD is designed to be lightweight and efficient, making it suitable for resource-constrained devices. It achieves a balance between accuracy and model size, enabling real-time or near real-time inference on low-power devices.

- Real-time Object Detection:
  MobileNet SSD is well-suited for real-time object detection tasks, offering low latency and fast inference. Its architecture enables efficient multi-scale feature extraction and prediction, making it ideal for applications requiring quick responses.

- Good Accuracy:
  Despite its lightweight design, MobileNet SSD achieves respectable accuracy in object detection. It utilizes depth-wise separable convolutions to reduce parameters while preserving important spatial and channel-wise dependencies.

- Object Detection and Classification:
  MobileNet SSD provides both object detection and classification capabilities within a single architecture. It can detect multiple objects in an image and classify them into predefined categories, facilitating comprehensive scene understanding.

- Transfer Learning:
  Like other CNN architectures, MobileNet SSD can be used for transfer learning. By leveraging pre-trained weights, it allows you to benefit from the knowledge and feature representations learned by the model on large-scale datasets.

3.7 Experiments and Results

The classification model used in this project was based on the ResNet50 architecture. The model was trained for 10 epochs, with a batch size of 32. During the training process, the model achieved an accuracy of 88%. Figure 8 provides a visual representation of the accuracy achieved by the model during the training phase.

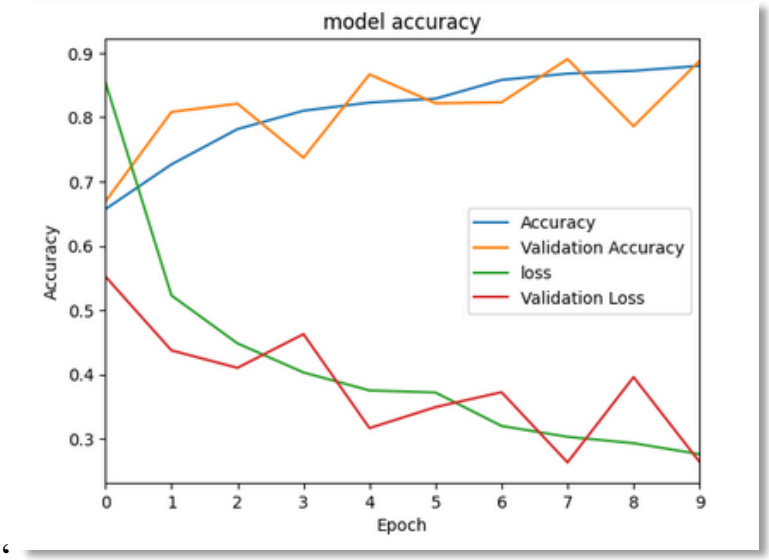


Figure 8: ResNet50 Architecture accuracy

The VGG16 architecture was utilized for the classification model in this project. The model was trained for 10 epochs, with a batch size of 32. During the training process, the model achieved an impressive accuracy of 99%. Figure 9 illustrates the accuracy attained by the model during the training phase.

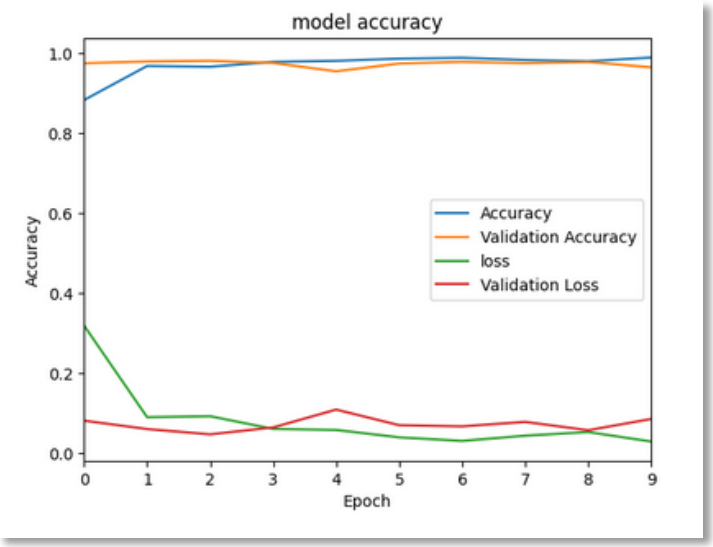


Figure 9: VGG16 Architecture accuracy

4 System Design

In Chapter 4, we delve into the system design of our defect detection and classification system. This chapter outlines the key components, interactions, and graphical user interface (GUI) design of the system. It provides a comprehensive view of the system architecture and functionality.

4.1 System Component Diagram

The system component diagrams provide a visual representation of the various components that make up the defect detection and classification system. These diagrams illustrate the relationships and dependencies between the different software and hardware components. We present two system component diagrams, depicted in Figures 7 and 8, to provide a detailed overview of the system's structure and organization.

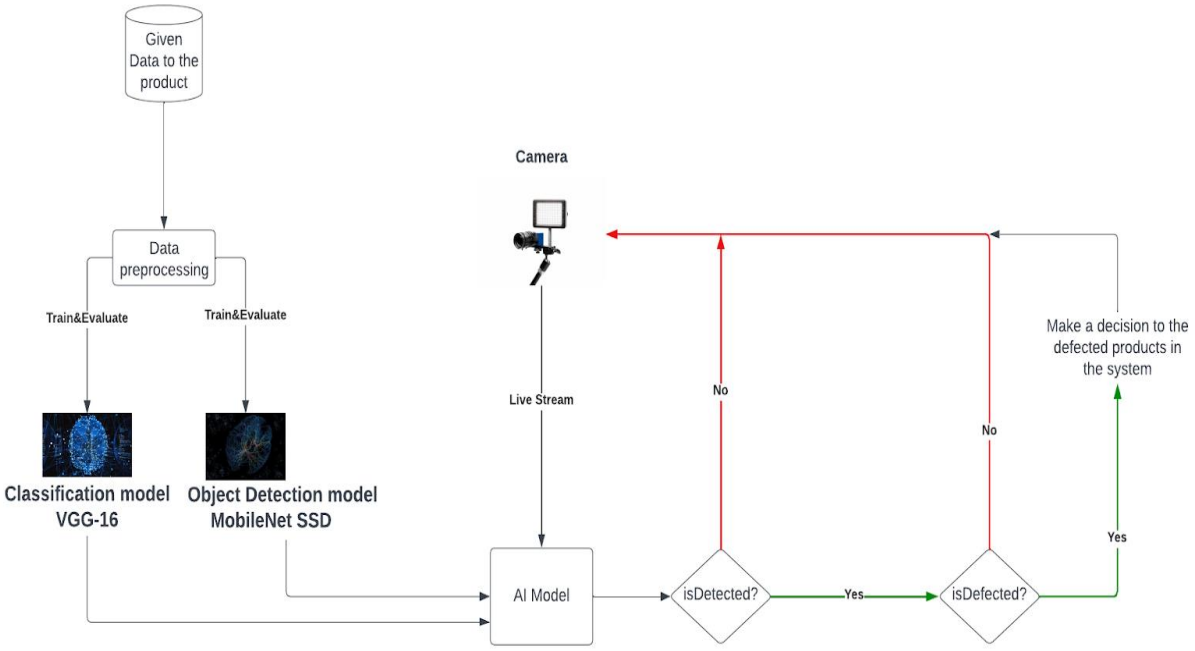


Figure 10: System Component Diagram 1

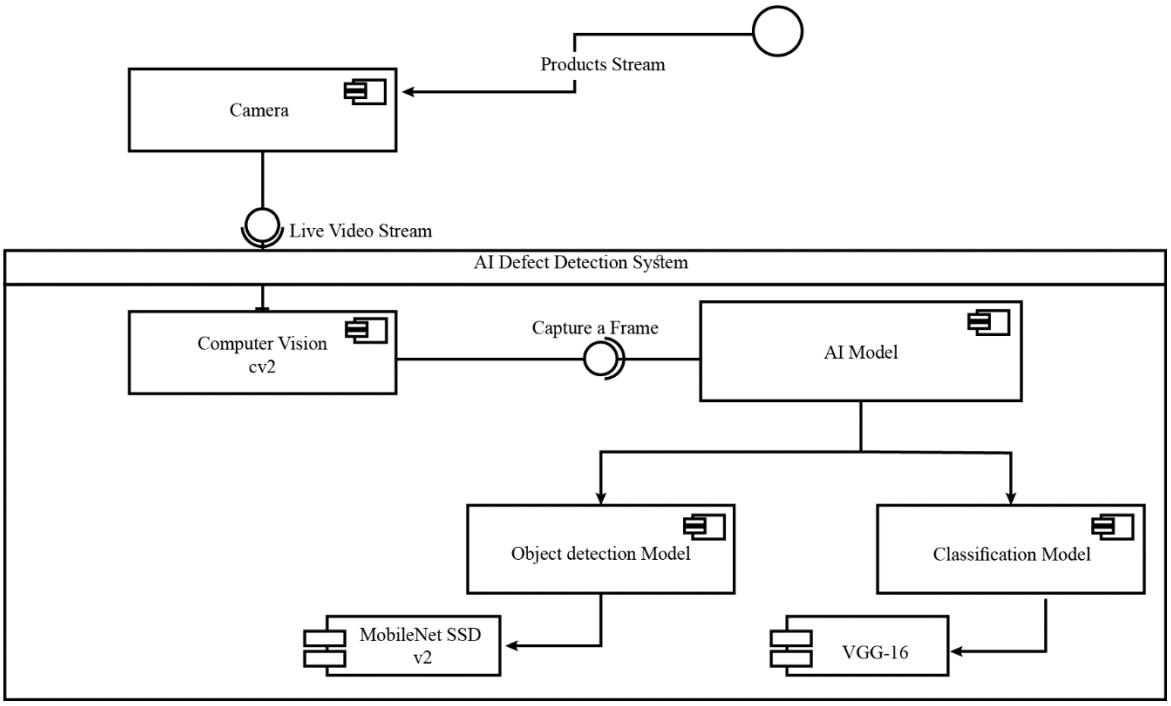


Figure 11: System Component Diagram 2

4.2 Sequence Diagrams

Sequence diagrams are used to showcase the interaction between different system components and the sequence of actions that occur during specific processes. These diagrams highlight the flow of control and data between various modules and components. We include several sequence diagrams to capture the different stages and interactions within our defect detection and classification system, demonstrating the system's behavior and functionality.

We present two sequence diagrams, depicted in Figures 9 and 10:

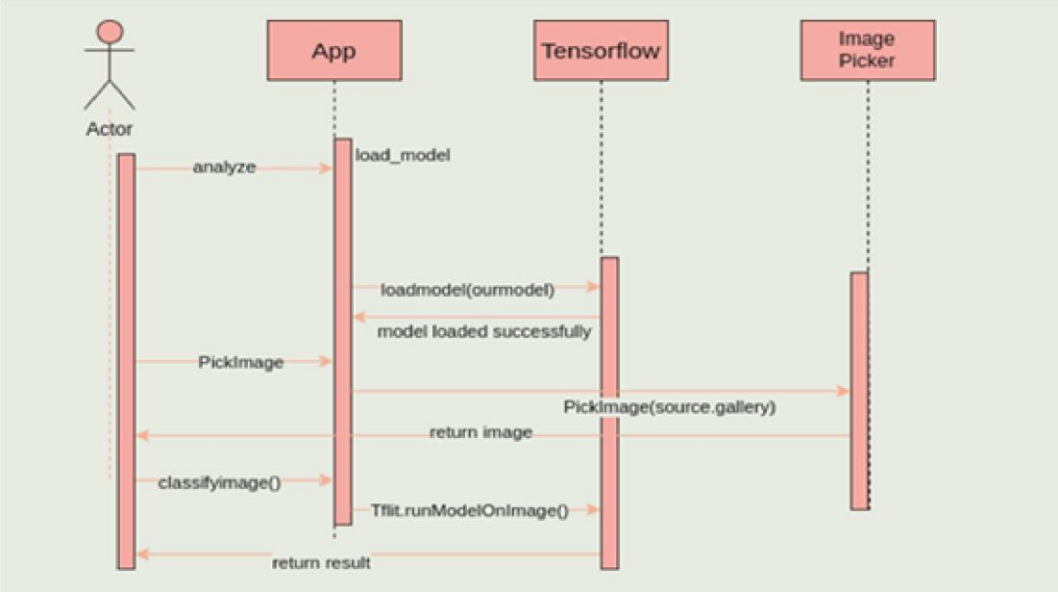


Figure 12: Application Sequence Diagram to classify Static Images

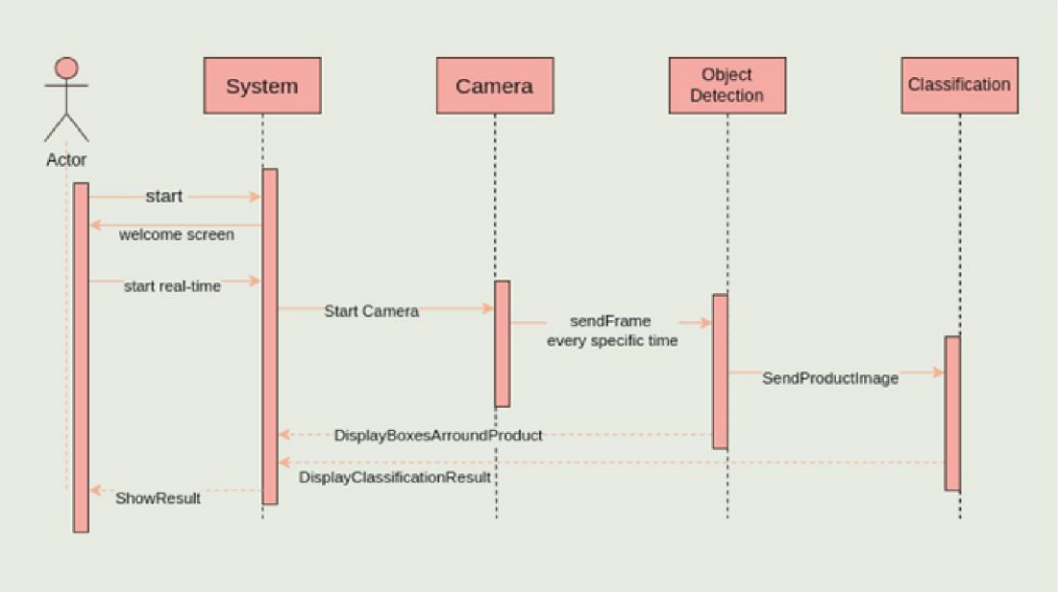


Figure 13: System Real-time Detection

4.3 System GUI Design

The system's graphical user interface (GUI) design is crucial for providing an intuitive and user-friendly experience. In this section, we present the design of the system's GUI, which includes the layout, components, and visual elements used to interact with the system. We focus on creating a clear and organized interface that allows users to easily interact with the AI model, view detected objects, and access system functionalities.

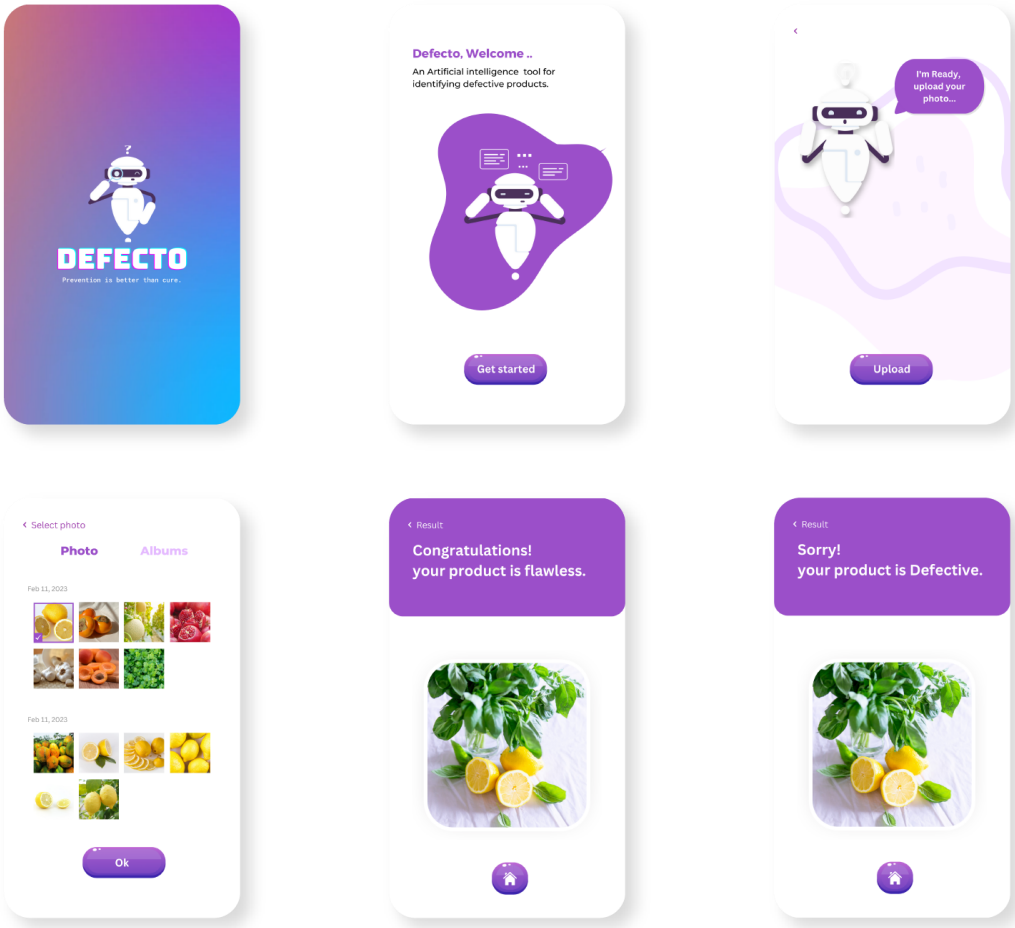


Figure 14: System GUI Design

5 Implementation and Testing

Our project revolves around two main tasks: object detection and image classification. The process initiates when a predefined object, on which the two models are trained, appears in front of the camera sensor connected to the program. Once the object is detected, it is sent to the classification model to determine its state, whether it is classified as good or defective. The classification model then relays the product's state back to the program, which proceeds to perform statistical analysis on the production process. It is crucial that all these steps occur in real-time, emphasizing the significance of time in the overall process.

5.1 Object Detection Model

The model used in our project is the MobileNet SSD Model provided by TensorFlow. We chose this model due to its compatibility with our IoT system, as it is a lightweight model that offers a balance between model size and accuracy. The MobileNet SSD model is well-suited for real-time object detection tasks, making it an ideal choice for our project's requirements.

5.1.1 Preprocessing

After the dataset of the “Casting Product” has been selected, it had to go under the preprocessing process of annotation in order to train the object detection model.

The annotation process mainly consists of drawing a box around the object to be detected producing a XML file with the same name of the image. This XML file carries the information of the annotation for a 1 training dataset image, for example, it provides the coordinates of the object inside the image, the label of each object inside the image and the extension of its image.

The tool used to annotate the images is Makesense. It was chosen because of its simplicity and flexibility to manipulate images during the annotation process as well as its ease of exporting the annotation for the images.

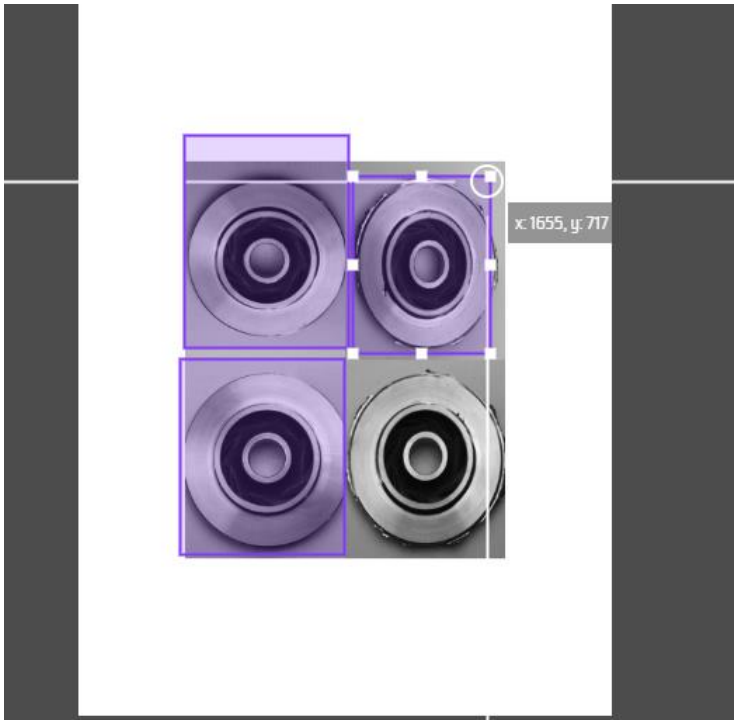


Figure 15:images during annotation process

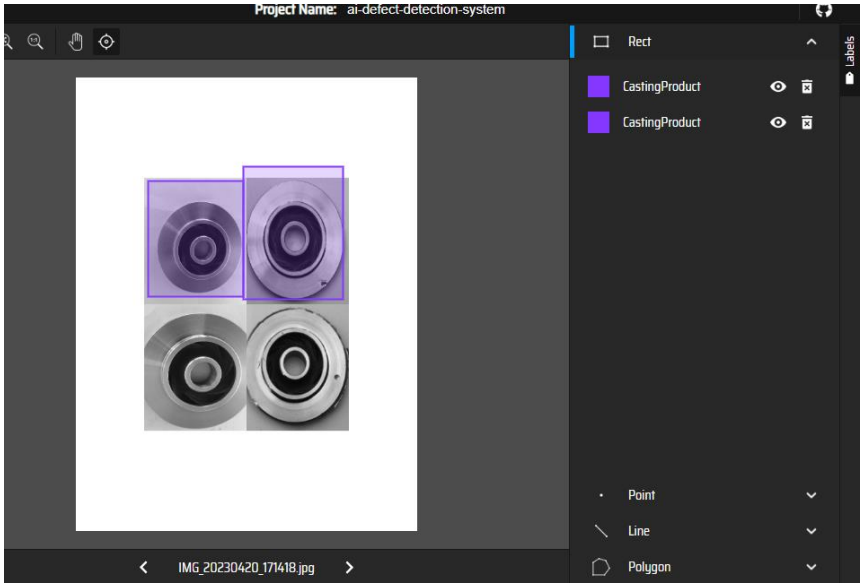


Figure 16: images during annotation process

5.1.2 Training phase

During the training phase, the AI defect detection model underwent multiple iterations with varying options, including different batch sizes, steps, and data sizes. The training process was divided into two stages: old training and latest training. The details of each stage, along with the corresponding results, are provided in the table below:

Training Stage	Batch Size	Steps	Data Size (annotated images)	Learning Rate	Total Loss
Old Training	4	5,000	520	0.073	0.18
Latest Training	16	10,000	1,000	0.073	0.14

Table 5: Training Phase Details

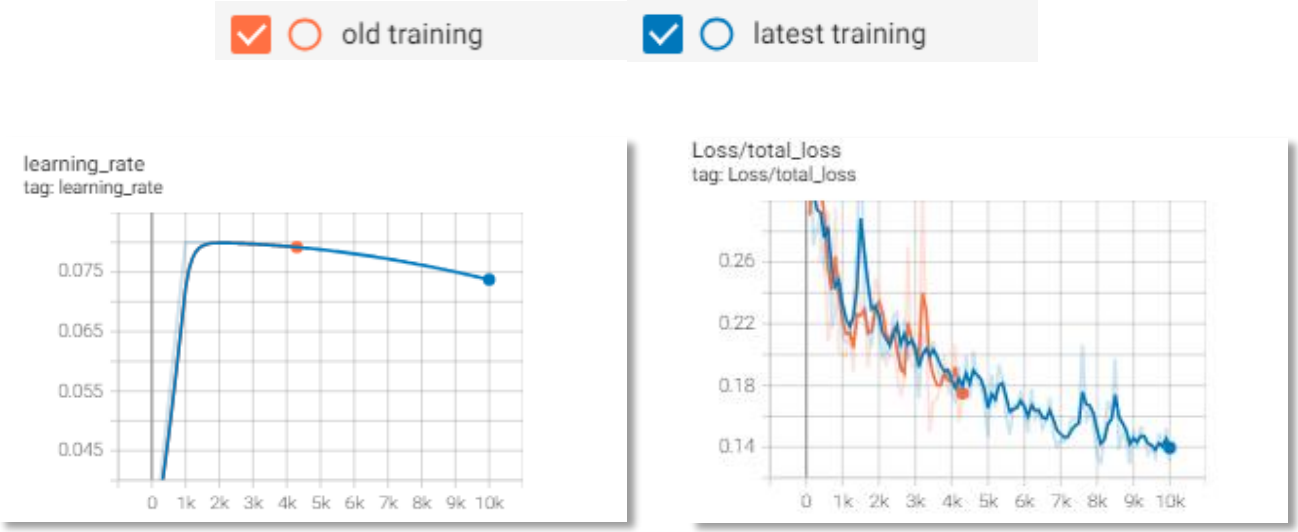


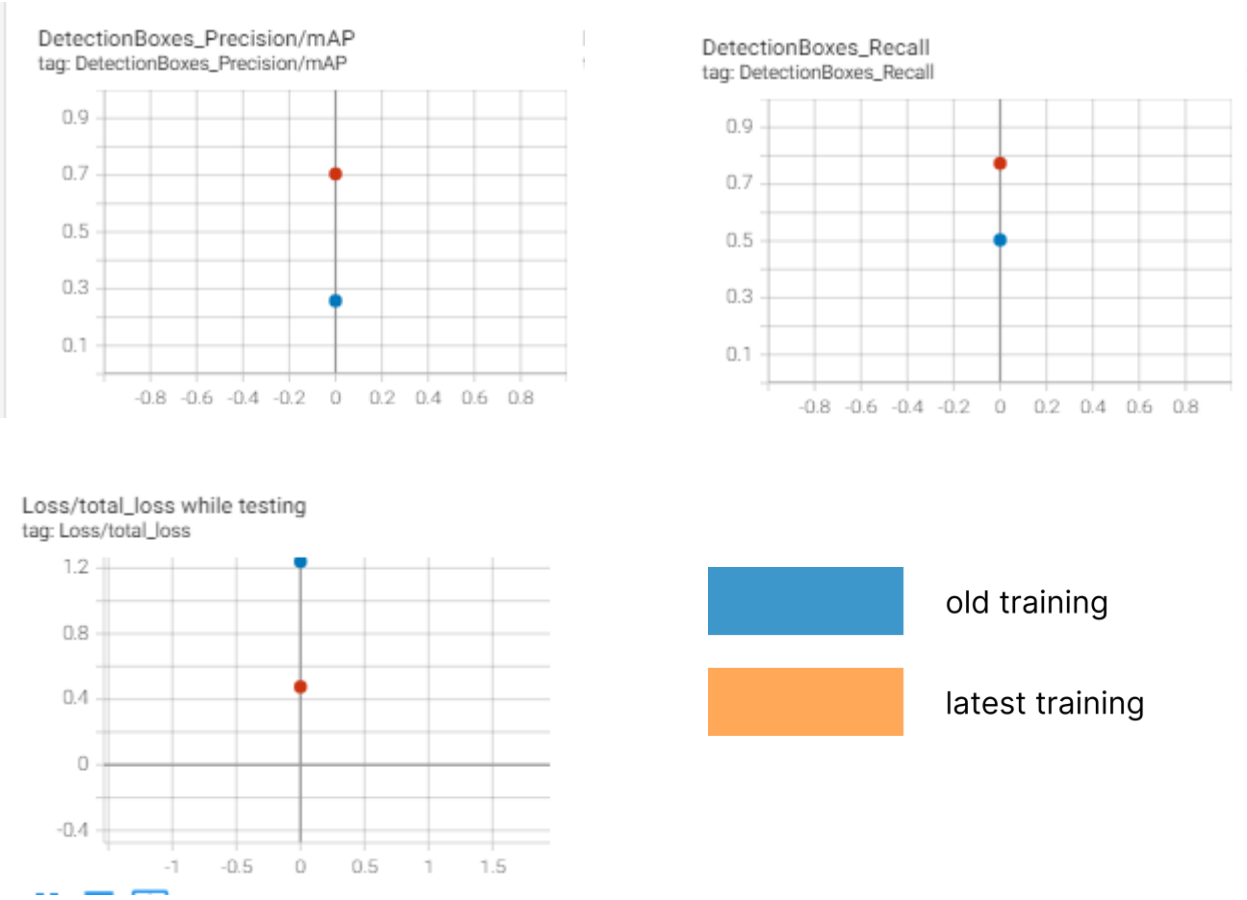
Figure 17: Line Chart Comparing Learning rate for Old Training and Latest Training

Figure 18: Line Chart Comparing Loss/Total Loss for Old Training and Latest Training

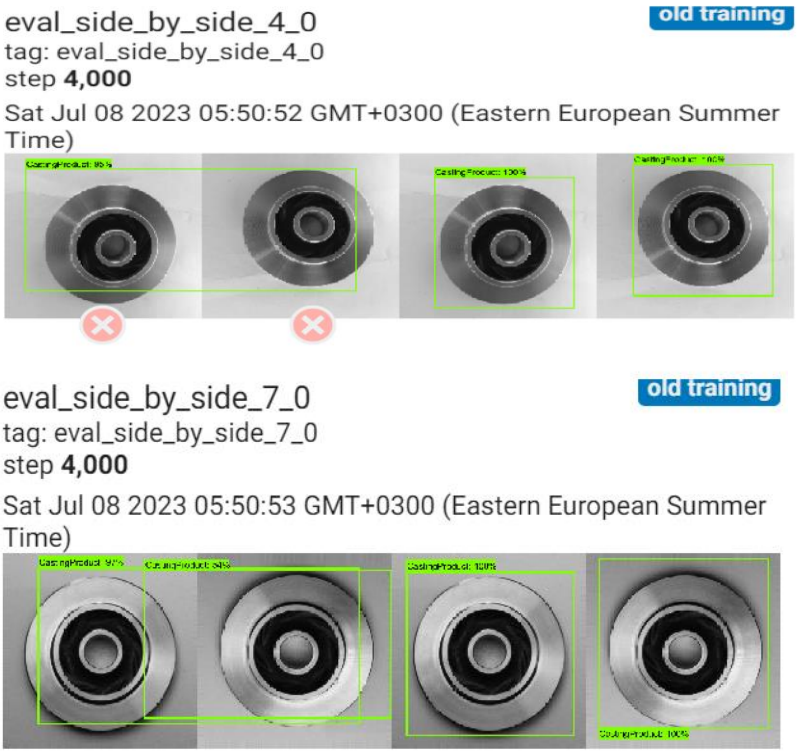
Figures 15 and 16 display images during the training process, providing a visual representation of the model's progression and performance. These images serve as snapshots of the training iterations.

5.1.3 Evaluation phase

Using same set of images to be tested on the 2 different versions of the model, the model with 10,000 steps seems to be more accurate in results as well as producing a mAP of 0.73 which is somehow good and acceptable accuracy for the mode.



Screens while testing



5.1.4 Testing phase

In this section, The AI defect detection model is thoroughly tested using new images that were not included in the original dataset. The purpose of this testing phase is to evaluate the model's performance on unseen data and assess its accuracy in real-world scenarios. The results obtained from this testing phase exceeded expectations, demonstrating the effectiveness of the model in detecting defects with high accuracy.

The testing process involved selecting a diverse set of new images, representing different types of defects and variations in lighting conditions. These images were carefully chosen to ensure a comprehensive evaluation of the model's performance across various scenarios. The model was then applied to these test images, and the obtained results were compared against ground truth labels for accuracy assessment.

The results obtained from the testing phase were highly promising. The model achieved accuracy rates ranging from 70% to 99% for the test images, depending on the specific defect and the complexity of the image. Figure 14, Figure 15, Figure

16, and Figure 17 showcase screenshots of the model accurately detecting defects in these new images, further affirming its efficacy.

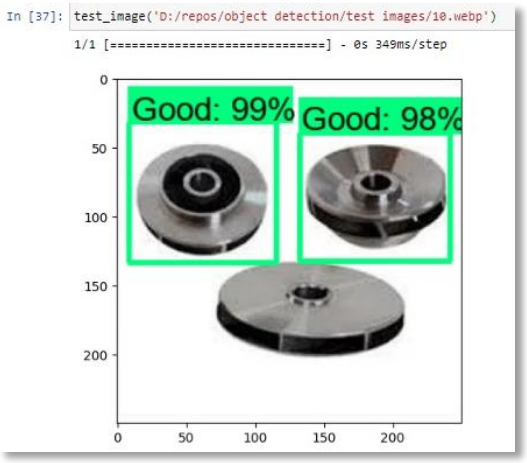


Figure 20: (a) Screenshot of Testing new images in object detection

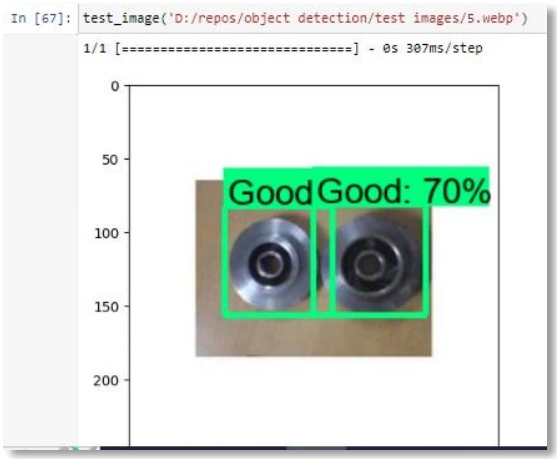


Figure 19: (b) Screenshot of Testing new images in object detection

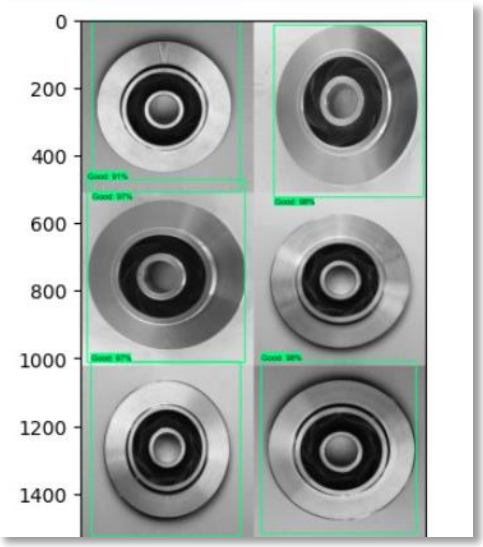


Figure 22: (c) Screenshot of Testing new images in object detection

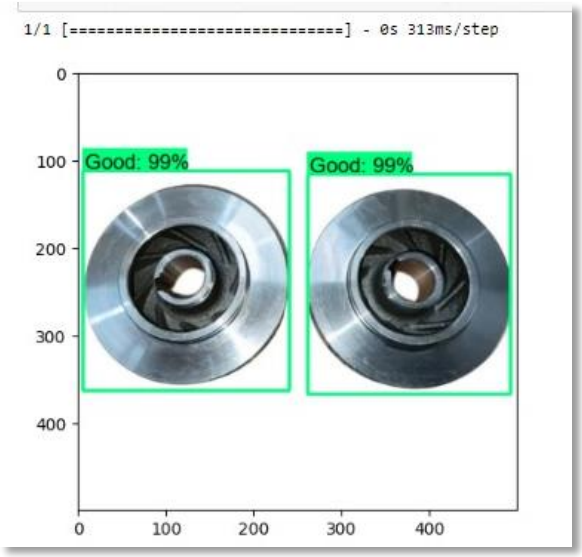


Figure 21: (d) Screenshot of Testing new images in object detection

5.2 Classification Model

The VGG-16 model was chosen for classification in our project due to its high accuracy. Extensive research and evaluation have demonstrated the effectiveness of the VGG-16 architecture in various computer vision tasks, including image classification. Its deep structure and large number of layers enable it to capture intricate features and patterns in the input data, leading to improved classification performance. By utilizing the VGG-16 model, we aimed to ensure accurate and reliable classification of objects, distinguishing between good and defective products with high precision and recall.

5.2.1 Implementation phase

The VGG-16 model was implemented using TensorFlow's Keras API for training. The model was trained with a total of 10 epochs and a batch size of 32. During the training process, the model was exposed to the training dataset in batches, with each batch containing 32 samples. The training was repeated for 10 epochs, allowing the model to learn and update its weights and biases based on the training data. This implementation strategy ensured that the model could capture complex patterns and features in the input data, enabling accurate classification and detection of defects.

```
VGG= keras.applications.vgg16.VGG16(input_shape=(224,224,3), include_top=False, weights='imagenet')
```

```
VGG.trainable= False
#Not train the front 13 layers, train only last three layers
```

```
model_vgg = keras.Sequential([
    VGG,
    keras.layers.Flatten(),
    keras.layers.Dense(units=256, activation="relu"),
    keras.layers.Dense(units=256, activation="relu"),
    keras.layers.Dense(units=2, activation="softmax"),

])
```

```
model_vgg.compile(optimizer='adam', loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
```

```
model_vgg.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 256)               6422784

 dense_1 (Dense)             (None, 256)               65792

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 21,203,778
Trainable params: 6,489,090
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 23: VGG-16 - implemnatation code

5.2.2 Evaluation phase

In the evaluation phase of the VGG-16 model, Figure 21 presents the accuracy and loss metrics of the model. The accuracy metric measures the percentage of correctly classified samples, indicating the model's overall performance. The loss metric, on the other hand, represents the discrepancy between the predicted and actual labels, serving as a measure of the model's error. By analyzing Figure 21, we can assess the model's performance during evaluation, observing trends in accuracy improvement and loss reduction. These metrics provide valuable insights into the effectiveness and reliability of the VGG-16 model in accurately classifying objects and detecting defects.

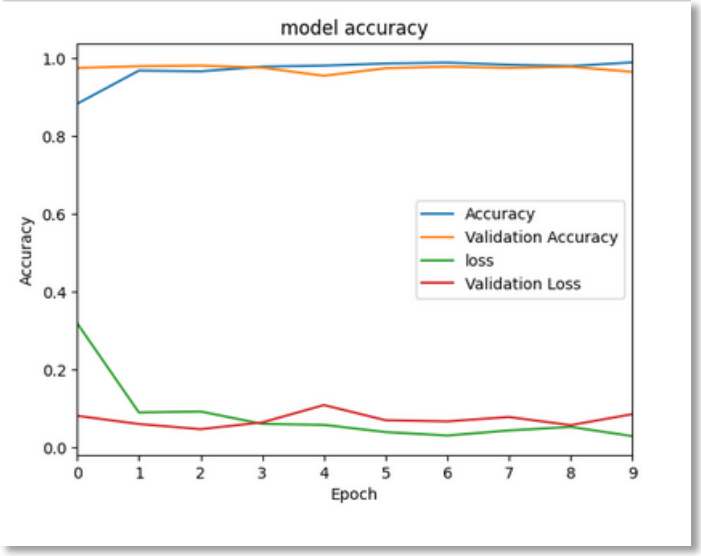


Figure 24: VGG-16 Accuracy

5.2.3 Testing phase

In the testing phase of the VGG-16 model, Figure 22 showcases the process of testing a new image that was not included in the original dataset. This image serves as a real-world scenario to evaluate the model's ability to generalize and make accurate predictions on unseen data. The figure displays the output of the model, presenting the predicted class or label assigned to the new image. By examining Figure 22, we can assess the model's performance in correctly classifying the new image, validating its effectiveness in detecting defects and determining the state of the product.

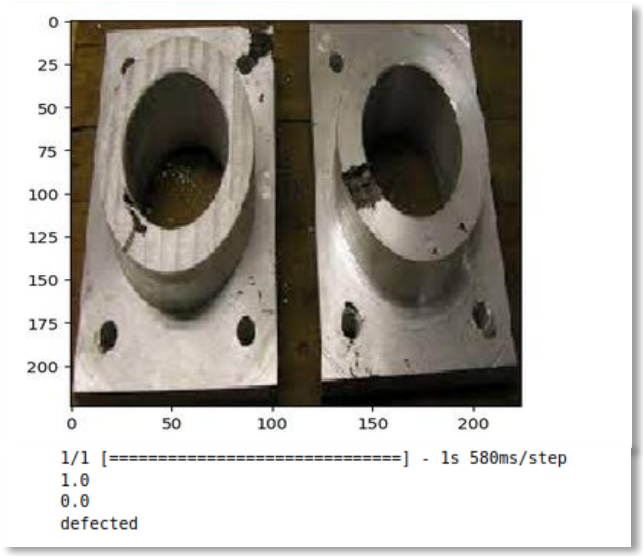


Figure 25: (a) Screenshot of Testing new images in object classification

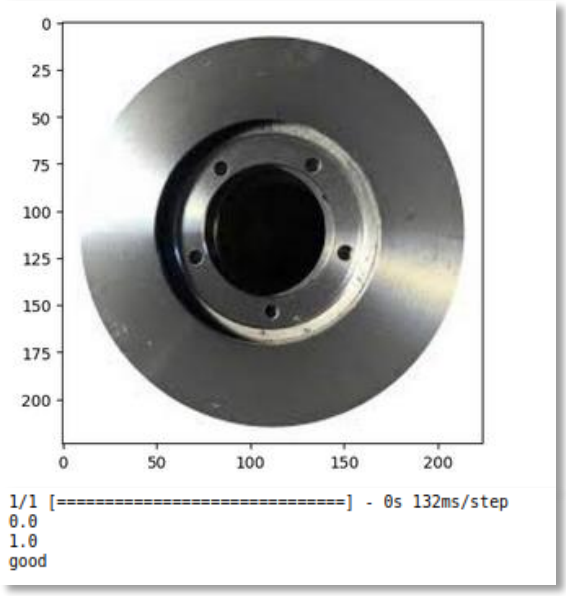
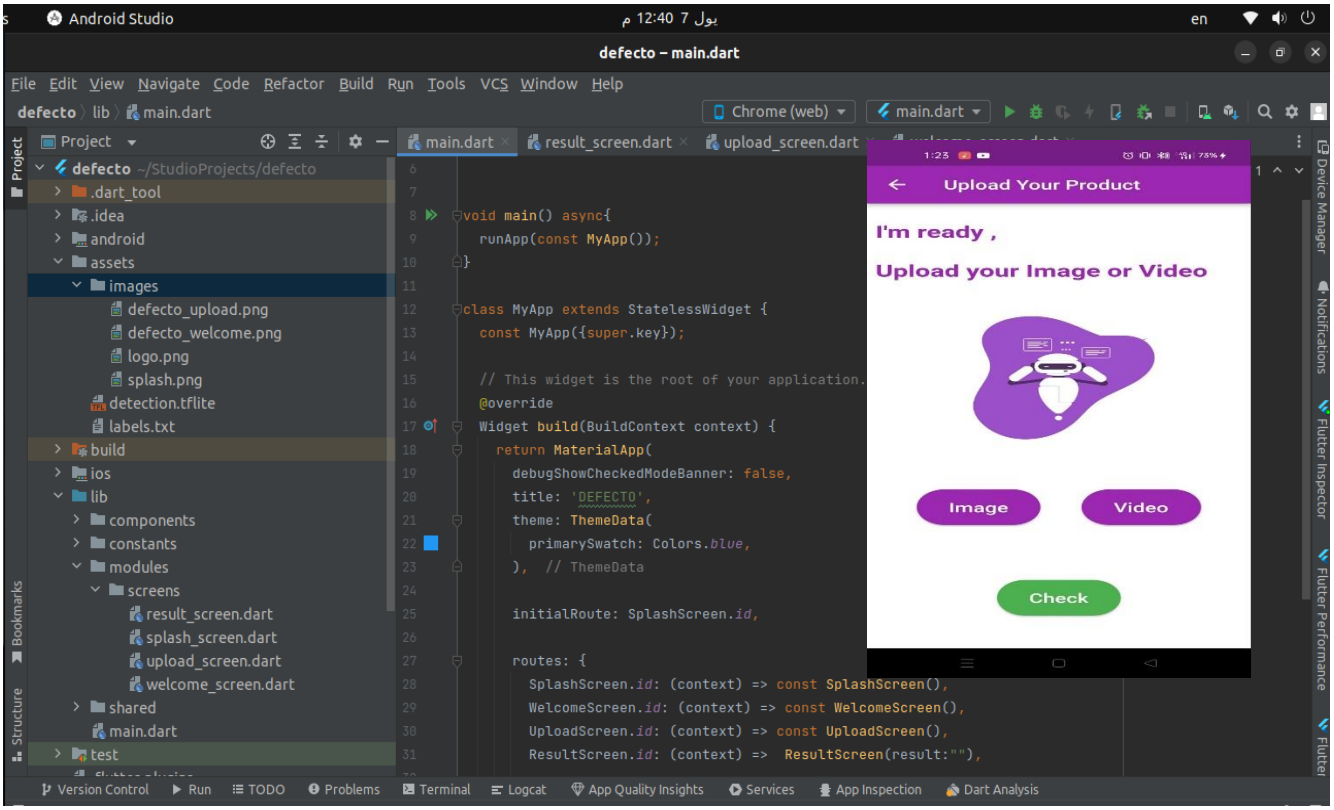


Figure 26: (b) Screenshot of Testing new images in object classification

5.3 Flutter App

- Implementing User-friendly application that include:
  Splash Screen, Welcome Screen, Upload Screen, Result Screen
- Using Stateful Widgets with setState() to update new interactions with the user
- Access the gallery of the mobile to can upload image or video
- Using TFLite for Object Detection Model
- Using Flask/Ngrok Server for Classification Model
- If failed to detect then display to user not Casting Product (Specific Product)
- If no detection then no classification
- Main ideas for the application:
  - Simulate project with embedded device as an IOT System
  - Using extensions same as IOT System
  - Integrate our models with different devices
  - Using for fixed products that need movable system



References

Akhremenko, V. (2022, 7 1). *Defect Detection in Manufacturing With Unsupervised Learning*. Retrieved from Mobidev: https://mobidev.biz/blog/defect-detection-in-manufacturing-with-unsupervised-learning

ASQ. (2023, 6 30). *COST OF QUALITY*. Retrieved from ASQ - American Society for Quality: https://asq.org/quality-resources/cost-of-quality#:~:text=Many%20organizations%20will,contribution%20to%20profits.

Choi, Y. J. (2019). *Structure defect detection using machine learning algorithms.* Innovation science and economic development canada.

Chung-Chi Huang, X.-P. L. (2017). *Study on Machine Learning Based Intelligent Defect Detection System.* MATEC Web of Conferences.

JONATHAN GRIGG, A. E. (2018, 1 22). *The Effects of Automation on Manufacturing*. Retrieved from Production Machining: https://www.productionmachining.com/articles/the-effects-of-automation-on-manufacturing

Payman Moallem, M. A. (2013). Computer vision-based potato defect detection using neural networks and support vector machine.

Qualitas. (2020). *Simplifying Complex Industrial Visual Inspections Using AI*. (Qualitas) Retrieved from Qualitas: https://qualitastech.com/

Sassi, P., Tripicchio, P., & Avizzano, C. A. (2019). *A Smart Monitoring System for Automatic Welding Defect Detection.* IEEE.

Stanisław Borkowski, K. K. (2016, 9). *Challenges Faced in Modern Quality Inspection.* Retrieved from Research Gate: https://www.researchgate.net/publication/309026292_Challenges_Faced_in_Modern_Quality_Inspection#pf5

Westphal, E. (2021). *A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks.* Additive Manufacturing.

Yang Jing, S. L. (2020, 12). *Using Deep Learning to Detect Defects in Manufacturing: A Comprehensive Survey and Current Challenges.* Retrieved from Research Gate: https://www.researchgate.net/publication/347821384_Using_Deep_Learning_to_Detect_Defects_in_Manufacturing_A_Comprehensive_Survey_and_Current_Challenges