Savitribai Phule Pune University, Pune

A

Project Stage-II Report

on

"AI-Powered Image Analysis for Industrial Object Quality Assurance"

In the partial fulfillment of Bachelor of Engineering in Electronics & Telecommunication Engineering of Savitribai Phule Pune University, Pune.

By

Mr. Aniket Narayan Gaware	**[Seat No. XXXXXXX]**
Mr. Anish Bharat Bawdhankar	**[Seat No. XXXXXXX]**
Mr. Rushikesh Madhusudan Patil	**[Seat No. XXXXXXX]**

Under the Guidance of

Prof. K. B. Kharat

Sinhgad Institutes

Department of Electronics & Telecommunication Engineering

Sinhgad Technical Education Society's

NBN Sinhgad Technical Institutes Campus, Ambegaon (Bk), Pune-41

[2024-25]

SINHGAD TECHNICAL EDUCATION SOCIETY'S
NBN SINHGAD TECHNICAL INSTITUTES CAMPUS
Approved by AICTE, New Delhi, Recognized by Govt. of Maharashtra &
Affiliated to Savitribai Phule Pune University (ID No.-PU/PN/Engg/432/2012)
S. No.10/, Ambegaon (Budruk), Off Sinhgad Road, Pune 411 041
Tel.: +91-20-24355042 / 46, +91-20-24610880/881 Tele Fax: +91-20-24355042
Website:http://nbnstic.sinhgad.edu Email: nbnssoe@sinhgad.edu

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that *Anish Bharat Bawdhankar, Aniket Narayan Gaware, Rushikesh Madhusudan Patil* have successfully completed the Project Stage-II entitled "**AI-Powered Image Analysis for Industrial Object Quality Assurance**" under my supervision, in the partial fulfillment of Bachelor of Engineering in Electronics & Telecommunication Engineering of Savitribai Phule Pune University.

Date:

Place: Pune

Prof. Kantilal B. Kharat	Dr. Makrand M. Jadhav	Dr. Shivprasad P. Patil
Guide	**Head E&TC**	**Principal/Director**

External Examiner:

SEAL

DECLARATION

It is hereby declared that, the project entitled **"AI-Powered Image Analysis for Industrial Object Quality Assurance"** is genuine work and it is not directly or indirectly copied from other project, resources, published work. This is original and authentic dissertation work and has not been submitted earlier to other university for entitlement of any degree by us or other and dissertation report has not been from any book, journals or website directly or indirectly. The reference used in this project work do not violet state, country, international copy right act, international property rights act and patent act, etc.

Sincerely,

1. Mr. Anish Bharat Bawdhankar
2. Mr. Aniket Narayan Gaware
3. Mr. Rushikesh Madhusudan Patil

Date:

Place: Pune

ACKNOWLEDGMENT

It is indeed a great pleasure and moment of immense satisfaction for we to present a Project Stage-II report on "**AI-Powered Image Analysis for Industrial Object Quality Assurance**" amongst a wide panorama that provided us inspiring guidance and encouragement, we take the opportunity to thanks to thanks those who gave us their indebted assistance. We wish to extend our cordial gratitude with profound thanks to our internal guide **Prof. Kantilal B. Kharat** for his/her everlasting guidance. It was his inspiration and encouragement which helped us in completing our Project Stage-II.

We wish to extend our sincere thanks to **Prof. Kantilal B. Kharat**, Project Coordinator for evaluation of project time to time and guidance. Our sincere thanks and deep gratitude to **Dr. Makrand M. Jadhav,** Head of Department for necessary infrastructure at department level. Also extend thanks to all staff members and to all those individuals involved both directly and indirectly for their help in all aspect of the Project Stage-II.

At last but not least we express our sincere thanks to **Dr. Shivprasad P. Patil,** Principal/Director for providing us required support and infrastructure.

Projectee

1. Mr. Anish Bharat Bawdhankar
2. Mr. Aniket Narayan Gaware
3. Mr. Rushikesh Madhusudan Patil

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS & ABBREVIATIONS

AI	:	Artificial Intelligence
CNN	:	Convolutional neural network
D-CNN	:	Deep - Convolutional neural network
DC	:	Direct current
ML	:	Machine learning
NDT	:	Non-destructive testing
SME	:	Small and medium enterprise
CV	:	Computer vision
OS	:	Operating system
GPIO	:	General purpose input output
IR	:	Infrared
LCD	:	Liquid crystal display
MVTec	:	Machine vision technology
SLS	:	Selective Laser Sintering
YOLO	:	You only look once
L-PBF	:	laser powder bed fusion
CAM	:	Class Activation Heatmaps
V8	:	Version 8
HD	:	High Definition
GUI	:	Graphical User Interface
IDE	:	Integrated development environment

ABSTRACT

The rapid advancement of artificial intelligence (AI) and image processing technologies has revolutionized industrial quality control by introducing more efficient and accurate methods for ensuring product consistency. This project, titled "AI Quality Analysis of Industrial Objects by Image Processing," aims to develop a robust, automated system that leverages AI and advanced image processing techniques to perform real-time quality inspections of industrial objects. The system is designed to improve efficiency, accuracy, and scalability in quality analysis, reducing human intervention and minimizing errors while lowering operational costs.

At the core of this system is the capture of high-resolution images of industrial objects as they move along a production line. Cameras are strategically positioned to capture detailed images, which are then processed using advanced image processing algorithms. These algorithms detect defects, anomalies, and deviations from predefined quality standards, such as cracks, surface irregularities, misalignments, and dimensional inaccuracies. The use of machine learning models, particularly convolutional neural networks (CNNs), allows the system to analyze these images, classify defects, and differentiate between acceptable and faulty products with a high degree of accuracy. Training these models on large datasets enables the system to identify both common and complex defects, significantly enhancing the overall inspection process.

Keywords: AI, Image Processing, Convolution Neural Network, Defect Detection, Automation, Cameras, Sensors.

Chapter No. 01

INTRODUCTION

1.1 Overview of Project Work

This project, titled "AI-Powered Image Analysis for Industrial Object Quality Assurance" focuses on automating the quality control process in manufacturing environments. It integrates AI and image processing to inspect products on a conveyor belt using Raspberry Pi 4 and a camera. The system captures high-resolution images of each product and analyzes them using advanced machine learning algorithms to detect defects such as cracks, misalignments, or surface irregularities. If a defect is identified, the system removes the faulty product from the belt and discards it into a defect box, ensuring only high-quality products proceed to the packaging stage.

This approach replaces traditional manual inspection methods, significantly improving accuracy, speed, and efficiency in the manufacturing process. The automation not only reduces human error but also enhances productivity by minimizing waste and ensuring consistent quality standards for every product produced. By leveraging state-of-the-art algorithms and real-time analysis, the project offers a modern solution to industrial quality control, helping manufacturers maintain high standards while lowering operational costs.

1.2 Comparison

In many manufacturing industries, quality control is a critical component of ensuring that products meet required standards before they are shipped to consumers. Historically, quality control systems have relied heavily on manual inspection, where human workers visually assess products for defects such as cracks, surface irregularities, or dimensional inconsistencies. While this method has been the standard for decades, it comes with significant limitations related to speed, accuracy, and adaptability.

Manual inspection is inherently slow and labour-intensive. Human inspectors must visually examine each product, a process that becomes increasingly difficult as production speeds up. This limits the scalability of manual systems, making them unsuitable for modern, fast-paced manufacturing environments. Furthermore, human fatigue can set in, leading to inconsistent accuracy. Errors can occur more frequently

during long shifts or with high volumes of products, resulting in defective goods passing through the inspection process undetected.

Moreover, manual systems suffer from low adaptability. While experienced workers can identify many common defects, their ability to recognize more subtle or unfamiliar defects is limited. As manufacturing processes evolve and new defect types emerge, manual inspectors may struggle to maintain the same level of accuracy without significant retraining.

To address some of these issues, many manufacturers have transitioned to automated quality control systems. These systems use cameras, sensors, and basic image processing algorithms to inspect products more quickly than human workers. However, automated systems that do not utilize machine learning (ML) still face several challenges.

While automated non-ML systems offer higher speeds than manual systems, they often lack the flexibility to detect subtle or unpredictable defects. Their image processing algorithms are typically rule-based, meaning they rely on pre-defined criteria to flag defects. As a result, they may miss defects that don't fit the specific patterns they were programmed to detect. Additionally, while these systems are generally more accurate than manual systems, they cannot improve their performance over time, as they do not learn from new data or adapt to changing conditions.

The proposed system, "AI Quality Analysis of Industrial Object by Image Processing," seeks to overcome these limitations by integrating advanced image processing and machine learning models into the quality inspection process. This system offers significant improvements in speed, accuracy, and adaptability, making it a far superior alternative to both manual and non-ML automated systems.

In terms of accuracy, the AI-enhanced system excels at detecting even the most subtle defects in industrial objects. By employing convolutional neural networks (CNNs), the system can analyse high-resolution images with remarkable precision, classifying defects that may be too small or irregular for human inspectors or basic automated systems to detect. The use of large datasets for training enables the AI models to recognize a wide range of defect types, making the system very accurate in classifying products as either acceptable or faulty.

One of the most significant advantages of the proposed system is its high adaptability. Unlike traditional automated systems, which follow rigid rules, the AI model can learn from new data. As the system encounters new types of defects, it continuously

improves its detection capabilities, ensuring that it remains effective even as manufacturing processes evolve. This machine learning capability allows the system to adapt quickly to changes in production lines, materials, or products without requiring extensive reprogramming or manual intervention.

Table No.1.1 Comparison table between manual and automatic system.

Sr No.	Features	Manual System	Automated System	Proposed System
1.	Speed	Slow	Medium	Fast
2.	Accuracy	Moderate	High	Very High
3.	Adaptability	Low	Low	High
4.	Cost	Low	Moderate	High

The proposed AI-powered quality control system represents a major leap forward in industrial inspection. With its superior speed, accuracy, and adaptability, it offers a highly scalable solution for modern manufacturing environments, ensuring consistent product quality while significantly reducing operational costs and waste. This project demonstrates how AI and machine learning can revolutionize traditional quality control methods, paving the way for more efficient and reliable production processes across various industries.

1.3 Problem Statement

Traditional quality inspection methods in manufacturing are often labor-intensive, time-consuming, and prone to human error.

Gears have a wide variety of use in electrical industries and need to be perfect and flawless with accurate. This problem of seeing and classifying gears are done by humans but have limitations of speed and accuracy. This responsibility of classification of items can be speeded and made more accurate by the use of imaging technology and computers aided by some electrical devices. A gear or more correctly a "gear wheel" is a rotating machine part having cut teeth, or cogs, which mesh with another toothed part in order to transmit torque. In transmissions with multiple gear ratios such as bicycles, motorcycles, and cars the term gear, as in first gear, refers to a gear ratio rather than an actual physical gear. The term describes similar devices, even when the gear ratio is continuous rather than discrete, as in a continuously variable transmission.

1.4 Organization of project

1.4.1 Defining the Mission Objective

- Mission Concept: Develop an AI-driven system that automatically inspects and classifies gears as either defective or acceptable in real time.
- Key Outcomes: Enhance production reliability, reduce manual inspection errors, and support predictive maintenance through early defect detection.

1.4.2 Team Formation & Roles

- Overall coordination, timeline management, and stakeholder communication Systems Engineering: Integration of hardware (sensors, cameras, Raspberry Pi) and software (deep learning model, image processing pipeline).
- Data Science & AI Development: Develop, train, and optimize the CNN model for defect detection; handle data collection and augmentation.
- Hardware & Embedded Systems: Set up and interface the camera, US sensor, motor controllers, and other components for real-time image acquisition and system control.
- Quality Assurance & Testing: Validate model performance, conduct field tests, and ensure robustness under different industrial scenarios.
- Documentation & Compliance: Maintain detailed records of design, development, testing, and ensure adherence to industry quality standards.

1.4.3 Development Phases

- Phase 1: Requirement Analysis & Research: Define system specifications and performance metrics. Evaluate current manual inspection challenges and identify areas for AI enhancement.
- Phase 2: System Design: Develop detailed block diagrams and system architecture. Specify hardware components (Raspberry Pi, camera, sensors) and software frameworks (TensorFlow, OpenCV).
- Phase 3: Prototyping & Development: Build a working prototype integrating hardware and AI software. Train the CNN model using curated datasets of gear images.
- Phase 4: Testing & Validation: Perform unit tests for individual components. Conduct integrated system tests on production lines to ensure real-time defect detection and minimal false rates.
- Phase 5: Deployment & Evaluation: Deploy the system in a live manufacturing environment. Monitor performance, collect feedback, and refine the model as needed.

1.4.4 Budgeting & Funding

- Cost Estimation: Expenses for Raspberry Pi, high-resolution cameras, sensors, and interfacing modules.
- Software Development: Development and testing of deep learning algorithms and

image processing pipelines.

- Testing & Deployment: Field tests, system integration, and maintenance.
- Funding Sources: University Grants & Research Funds: Apply for academic or industrial research funding.
- Industry Partnerships: Collaborate with manufacturing firms looking to enhance quality control.

1.4.5 Compliance & Documentation

- Standards & Regulatory Approvals: Ensure compliance with industrial quality standards (e.g., ISO). Adhere to guidelines for safe integration of AI systems into manufacturing processes.
- Documentation: Maintain comprehensive records of system designs, test results, and operational procedures. Document data handling, model training, and system updates for transparency and future reference.

1.4.6 Deployment & Operations

- Integrate the system into selected production lines for pilot testing. Collaborate with plant managers to align system operation with existing quality control processes.
- Establish a monitoring dashboard for real-time system status. Set up periodic maintenance schedules and software updates to ensure continued high performance.

1.4.7 Post-Deployment & Data Analysis

- Initial Validation: Verify system functionality and defect detection accuracy through initial operational tests. Gather feedback from operators and quality assurance teams.
- Future Enhancements: Explore advanced deep learning techniques for improved accuracy. Scale the system to handle multiple production lines or varied gear types.

Chapter No. 02

LITERATURE REVIEW

2.1 Literature Review on Project Work

The project on automated quality control using machine learning and image processing is based on a growing body of research that emphasizes the importance of automation in improving manufacturing processes. The following are ten key research papers that contributed to the conceptualization, design, and development of this system.

Research in additive manufacturing (AM) has seen significant growth, particularly in laser powder bed fusion (L-PBF) for creating complex metal components. Studies by Gobert et al. titled "A Preliminary Study of In-Situ Layer wise Defect Detection in Laser Powder Bed Fusion" and Scime and Beuth in "Layer wise Monitoring of Powder Bed Fusion" focus on in-situ layer-wise imaging and powder bed anomaly classification, highlighting the challenges of defect detection due to the high number of input parameters involved in AM processes [1].

Delamination, porosity, and other structural defects in L-PBF components can significantly compromise mechanical properties, as discussed by Van Elsen in "Optimizing Process Parameters in Selective Laser Melting" and Khanzadeh et al. in "Real-Time Defect Detection in Laser Powder Bed Fusion Using Machine Learning". These defects are often only detectable post-process through destructive testing, such as tomography or metallographic imaging, which is costly and time-consuming. This has driven the pursuit of real-time, in-situ monitoring methods [2].

Studies utilizing thermographic and infrared imaging, such as those by Okaro et al. in "Thermographic Inspection of Laser Powder Bed Fusion" and Shevchik et al. in "Real-Time Acoustic Emission and Infrared Sensing for Defect Detection in Laser Powder Bed Fusion" demonstrate the potential of thermal monitoring to capture key temperature variations indicative of potential defects. Off-axis infrared imaging has proven to be especially effective for tracking real-time anomalies, paving the way for further development in this area. Their work provides a foundation for thermal data-based ML applications in defect detection [3].

Advances in machine learning (ML) have enabled new approaches to AM defect detection, as shown in the studies by Gobert et al., Okaro et al., and Khanzadeh et al. Their studies applied various ML methods—including k-nearest neighbour, decision

trees, and semi-supervised learning—to identify defect patterns. Their results highlight the benefits of machine learning for detecting complex defect patterns that are not easily visible through conventional means [4].

Supervised learning techniques have been widely applied in defect detection studies, with researchers such as Ye et al. in "Supervised Machine Learning for Defect Detection in Additive Manufacturing" and Khanzadeh et al. achieving high accuracy rates using labelled data to train their algorithms. These approaches require pre-labelled datasets, making them effective in scenarios where defects are known and can be categorized, although they can be resource-intensive [5].

Unsupervised learning, as utilized by Okaro et al., attempts to detect anomalies without labelled data, making it suitable for dynamic environments. Semi-supervised approaches, combining both labelled and unlabelled data, offer a balance between flexibility and accuracy. This technique has been shown to reduce the need for extensive labelled datasets while maintaining high detection accuracy [6].

Acoustic monitoring, as explored by Shevchik et al. and Ye et al., provides an alternative method for defect detection. By analysing acoustic signals generated during the manufacturing process, researchers were able to identify defects with improved accuracy after data transformation. This method demonstrates promise, though further work is needed to improve raw data accuracy [7].

Recent studies by Scime and Beuth in "Image-Based Analysis for Laser Powder Bed Fusion" and Khanzadeh et al. apply convolutional neural networks (CNNs) to process images of melt pools and powder beds. CNNs can capture spatial features within images, making them ideal for analysing thermal and visual data in L-PBF processes. CNN-based models have shown to be robust for defect classification, even with a relatively small network architecture [8].

Data preprocessing, including transformation, augmentation, and noise reduction, has been essential for achieving high accuracy in defect detection. Scime and Beuth and Khanzadeh et al. both demonstrate how preprocessing improves model performance by enhancing feature clarity and reducing unnecessary noise, highlighting the importance of data quality in AM monitoring [9].

Off-axis thermographic imaging, utilized in this study, aligns with the findings of Shevchik et al. and Okaro et al. on the effectiveness of infrared monitoring. Thermographic data provides a consistent representation of temperature distribution across layers, making it ideal for identifying heat-related anomalies during the AM

process. This method captures a broad range of thermal information, contributing to a more accurate detection of defects [10].

Class activation heatmaps (CAMs) have proven valuable for identifying defect locations, as described in recent work by Khanzadeh et al. CAMs provide a visual interpretation of model focus areas, enabling users to understand where defects are most likely to occur. This technique aids in refining ML models by confirming that they are focusing on the right features [11].

To ensure robust defect detection, k-fold and hold-out cross-validation methods have been widely used in studies by Gobert et al. and Scime and Beuth. These validation techniques help avoid overfitting and improve model generalizability by evaluating performance across different data subsets. Such rigorous validation is essential for creating models suitable for real-world applications [12].

In comparing various ML architectures, researchers like Okaro et al. and Khanzadeh et al. found that k-nearest neighbour and CNN-based models performed best for defect detection. Their studies suggest that each architecture has its strengths depending on data complexity and application needs, with CNNs excelling in image-based data and simpler models working well for less complex scenarios [13].

While current defect detection methods, such as the one presented by Gobert et al., provide reasonable accuracy, challenges remain in achieving real-time monitoring. Most techniques require powerful hardware, limiting their practical applicability. Studies by Ye et al. and Shevchik et al. emphasize the need for lightweight architectures that can operate on less powerful systems [14].

The reviewed studies highlight the rapid evolution of defect detection in additive manufacturing, particularly through the use of thermographic imaging and machine learning. Despite advances, challenges in achieving high real-time accuracy and reducing computational requirements remain. This project builds upon this research by developing a streamlined CNN model for real-time defect detection, leveraging thermographic data and optimized cross-validation techniques to ensure both accuracy and computational efficiency [15].

2.2 Motivation

The primary motivation for this project is the increasing demand for efficient, accurate, and automated quality control systems in modern manufacturing. Traditional quality inspection methods are labor-intensive and prone to human error, leading to delays in production and inconsistencies in quality. Moreover, with the rise of

Industry 4.0, there is a growing need for automation, real-time data analysis, and machine learning-driven solutions that can operate on the edge and provide immediate insights.

Manufacturers are constantly seeking ways to reduce operational costs, minimize product defects, and improve overall production efficiency. The project aims to address these challenges by providing a scalable, adaptable, and real-time quality control solution that leverages machine learning and image processing.

Additionally, the cost-effectiveness of using a Raspberry Pi for edge-based computing makes it feasible for small and medium-sized enterprises (SMEs) that may not have the resources to invest in high-end inspection systems.

2.3 Objectives of Work

The objectives of this project are as follows

- Implement a machine learning-based solution capable of detecting defects in products with high accuracy, reducing reliance on manual inspection.
- Design the system to operate in real-time, allowing immediate feedback on the production line without causing delays in manufacturing.
- Employ techniques like edge detection, thresholding, and image segmentation to preprocess images before defect analysis, improving the system's accuracy.
- Utilize Raspberry Pi for real-time processing, ensuring the system is cost-effective and can be easily deployed across various production environments.
- Achieve an accuracy rate of 95% or higher in detecting product defects, with minimal false positives and false negatives.
- Design the machine learning model to be adaptable to different types of products and defects, allowing for easy retraining and scalability to different manufacturing lines.

2.4 Contribution

The project contributes to the field of automated quality control systems by providing the following key advancements

1. Increased Accuracy and Speed

   The integration of machine learning algorithms with image processing techniques significantly improves the accuracy and speed of defect detection, outperforming traditional manual inspection methods.

2. Real-Time, Edge-Based Processing

Unlike conventional systems that rely on cloud processing, this project utilizes edge computing with Raspberry Pi, enabling real-time decision-making directly on the production line without latency issues.

3. Cost-Effective Solution

By utilizing affordable hardware like Raspberry Pi and open-source libraries like OpenCV and TensorFlow, this system is a more cost-effective alternative to high-end automated inspection systems, making it accessible to small and medium-sized manufacturers.

4. Adaptability for Different Applications

The machine learning model can be retrained with new data, allowing the system to adapt to different manufacturing environments and detect various types of defects.

Chapter No. 03

METHODOLOGY

3.1 Overview on Proposed work

The success of an automated gear inspection system relies heavily on the right selection of hardware and software components. These components must work together seamlessly to enable real-time image processing, defect detection, classification, and automation of the inspection process. The hardware components are responsible for image acquisition, motion control, and mechanical actuation, while the software components handle image processing, machine learning, and automation. The Raspberry Pi 4 acts as the central controller, offering sufficient computing power to manage real-time image analysis and motor control. A USB camera captures high-resolution images of gears as they move along the conveyor belt, ensuring accurate visual input for detection. The 12V DC motor, in combination with a motor driver, provides controlled motion to the conveyor, while an ultrasonic sensor detects object presence and triggers the image capture process. A servo motor is used to physically reject defective gears from the conveyor line, maintaining product quality.

On the software side, the system uses a YOLOv8 deep learning model trained to classify gears into categories such as "correct," "defective," or "no gear." The model processes incoming images in real-time and generates bounding boxes and class labels for detected objects. Python-based control scripts handle decision-making logic and GPIO pin management for hardware actions.

3.2 Hardware and Software Selection

3.2.1 Hardware Selection:

The hardware components were carefully chosen to provide the necessary processing power, precision, and durability to inspect gears efficiently. The selected components allow the system to capture high-resolution images, process them in real time, and perform classification-based sorting.

- Raspberry Pi 4 Model B (RPI4B): Selected for its processing power and GPIO interface. It is capable of handling real-time image processing tasks with sufficient speed.
- USB Camera (Full HD): A USB Camera (Full HD**)** module (8 MP) is used to capture high-resolution images of products.

- Conveyor Belt: A standard conveyor belt is used, powered by a DC motor, controlled by the Raspberry Pi.
- Servo Motor **(**1 Unit): A high-torque servo is chosen to ensure smooth removal of defective products.
- Power Supply circuit: A 12V DC power supply is used to power the motor, and the Raspberry Pi is powered by a 5V supply.
- US sensor (Ultrasonic Sensor): Range: 2-10 cm, Detects the presence of a gear on the conveyor belt.
- Motor Drive (L298N): Displays the status of the inspection process, showing the number of defective products and the total number of inspected products.
- Voltage Regulator: Converts 12V from the battery to 5V for safely powering the Raspberry Pi.

3.2.2 Software Selection:

The software components were carefully chosen to enable smooth development, deployment, and execution of the machine learning model for defect detection.

- Raspberry Pi Imager:

  Role: Flashes the Raspberry Pi OS onto the microSD card effectively and quickly .

  Reason for Selection: Easy-to-use GUI for OS installation and Compatible with all Raspberry Pi models.
- Raspberry Pi OS (64-bit):

  Role: The primary operating system running on Raspberry Pi for the processing.
  Reason for Selection: Optimized for Python-based development. Supports TensorFlow Lite and OpenCV libraries for machine learning and image processing.
- Thonny (Python IDE):

  Role: An IDE for writing and debugging Python scripts.
  Reason for Selection: Lightweight and beginner-friendly. Built-in support for Python GPIO and debugging tools.
- Python:

  Role: The core programming language for writing automation and ML scripts.
  Reason for Selection: Good Library support for AI, ML, and hardware interfacing.
  Key Libraries Used:

  1. OpenCV – For image processing and defect detection.

2. NumPy – For mathematical operations on image data.
3. Matplotlib – For visualizing defect classification results.
4. Ultralytics – For training, validating, and deploying YOLO models.

- TensorFlow, Keras and PyTorch:

  Role: Used for deep learning-based gear classification and object detection. Reason for Selection: Supports CNN-based models for image classification. Compatible with TensorFlow Lite for fast inference on Raspberry Pi.

  Model Used:

  1. Yolo v8 Pre-Trained Model
  2. Pre-trained on gear images with defect categories

3.3 Importance of work

The significance of this project, "AI Quality Analysis of Industrial Object by Image Processing," lies in its ability to revolutionize quality control in manufacturing through automation, increased accuracy, and adaptability. By incorporating cutting-edge technologies such as machine learning and image processing, the system enhances the overall efficiency and effectiveness of the quality control process, addressing many limitations of traditional and non-ML automated systems.

3.3.1 Features

- Automation

  One of the primary benefits of the system is the automation of the entire quality control process. Automation reduces the need for human intervention, which helps eliminate the inconsistencies and potential errors introduced by manual inspection. This leads to more reliable and uniform inspection standards across all products.

- Accuracy

  The use of machine learning (ML) models, especially convolutional neural networks (CNNs), improves the accuracy of defect detection significantly. Unlike traditional manual inspection, which is subjective and prone to human fatigue, the ML model can detect even the most subtle defects with high precision. The system is capable of recognizing complex defect patterns, thereby enhancing overall quality control and reducing the chances of defective products being passed to consumers.

- Speed

  Real-time processing ensures that the system can inspect products at high speeds without slowing down production lines. The system integrates with Raspberry Pi 4,

high-resolution cameras, and other hardware components to enable real-time analysis and decision-making, thus preventing any bottlenecks in the manufacturing process. This increase in speed is crucial in industries where rapid production rates are a competitive advantage.

3.3.2 Technology

The project leverages advanced image processing and machine learning technologies to enhance industrial quality control. Specifically, it uses:

- Raspberry Pi 4

  This affordable, edge-based computing platform enables the system to perform image processing and ML tasks at the edge, reducing the need for cloud-based processing and improving response times.

- OpenCV

  This open-source computer vision library is used for image acquisition, enhancement, and processing. OpenCV allows the system to detect visual features such as cracks, misalignments, and surface irregularities with high efficiency.

- YOLOv8

  This advanced, real-time object detection model developed by Ultralytics leverages Convolutional Neural Networks (CNNs) and is built using PyTorch. YOLOv8 provides fast and accurate detection of defects, such as cracks, misalignments, and surface anomalies, making it ideal for industrial quality assurance tasks.

- Ultralytics

  A high-level Python framework that simplifies the training, validation, and deployment of YOLOv8 models. It provides tools for data handling, model customization, and export, enabling rapid development and real-time inference directly on edge devices like the Raspberry Pi 4.

- PyTorch

  This flexible and efficient deep learning framework underpins YOLOv8, supporting dynamic computation graphs and GPU acceleration. PyTorch enables smooth integration with YOLOv8 for model training, fine-tuning, and real-time inference on both cloud and edge platforms.

- TensorFlow/Keras

  These popular machine learning frameworks are employed for training and deploying the defect detection models. The ability to train models on large datasets

makes the system adaptable and more accurate over time, enhancing its ability to detect new types of defects.

The integration of these technologies creates a robust, scalable, and adaptable system that performs real-time analysis of industrial objects on production lines.

3.3.3 Probable Results

By implementing this AI-enhanced quality control system, several positive outcomes are expected

- Reduction in Defective Products

  The accuracy and consistency of the system will lead to a significant reduction in defective products reaching customers, which can improve brand reputation and customer satisfaction.

- Increased Inspection Speed and Efficiency

  The automation and real-time processing capabilities will allow manufacturers to inspect products at high speeds, increasing throughput without sacrificing accuracy.

- Minimization of Manual Labor and Errors

  With minimal human involvement, the system reduces the likelihood of human error, while also lowering operational costs associated with labor.

- Improved Adaptability

  As the system continually learns from new data, it can detect evolving defect patterns, making it future-proof for industries with changing production requirements.

  The project presents several key advantages that make it ideal for modern manufacturing

- Scalability

  The system is designed to be highly scalable, meaning it can be easily adapted to different production lines and industries. Whether it's for inspecting electronic components, automotive parts, or textiles, the system can be configured to handle a wide variety of products with minimal adjustments.

- Cost-Efficiency

  Compared to fully manual inspection methods or expensive high-end inspection systems, the AI-based approach provides a cost-effective solution. Using Raspberry Pi for computing and open-source software like OpenCV and TensorFlow keeps the costs low, while still delivering powerful capabilities.

- Adaptability

  One of the standout features is the system's ability to learn and adapt over time. By retraining the ML models with new defect data, the system can remain relevant and efficient as new defect types or inspection requirements emerge. This ability to evolve makes it much more versatile than traditional systems, which often require costly updates or redesigns when new defect types arise.

- Enhanced Productivity

  The combination of automation, accuracy, and real-time processing leads to increased productivity. By reducing downtime and ensuring only high-quality products proceed through the production line, manufacturers can optimize their output without sacrificing quality.

- Reliability

  The continuous operation of the system, without the fatigue or variability that comes with human inspectors, ensures consistent and reliable quality control. This can prevent costly recalls and maintain product standards.

  In conclusion, the importance of this project lies in its ability to transform traditional quality control practices.

3.4 Working Principle

The proposed automated gear inspection system operates using a combination of computer vision, machine learning, and hardware automation. The system is designed to detect defects in gears by capturing images, processing them using AI-based models, and classifying the gears as faulty or defect-free. The overall working principle consists of image acquisition, image processing, defect detection, and classification.

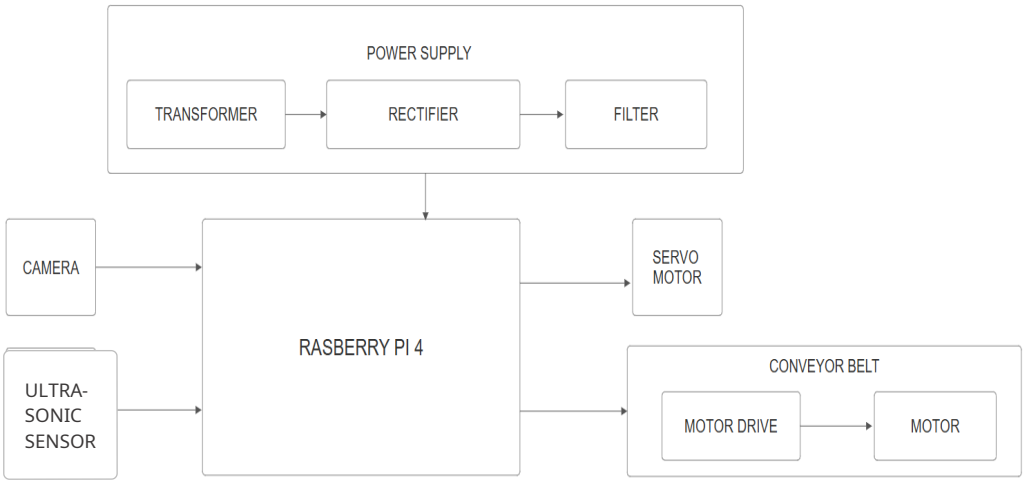

Figure No.3.1 Block diagram of AI object detection system

This block diagram represents an Automated Gear Inspection System using Raspberry Pi 4B and computer vision-based defect detection:

1. Power Supply Unit:

- Transformer**:** This is the first stage of your power supply. It takes the main AC power from a wall outlet (230V AC in India) and steps it down to a lower AC voltage suitable for the subsequent stages.
- Rectifier**:** The rectifier's job is to convert the lower AC voltage coming from the transformer into a pulsating DC voltage. This is a necessary step because the Raspberry Pi and the motors require DC power.
- Filter**:** The pulsating DC voltage from the rectifier is not smooth enough for electronic devices. The filter (using capacitors) smooths out these pulsations, providing a more stable and cleaner DC voltage.
- Output: The stable DC power from the power supply unit is then fed to the Raspberry Pi 4 to power its operation.

2. Input Devices:

- Camera: This is the "eye" of your system. It captures live images of the objects moving on the conveyor belt. This video or image stream will be the primary input for your AI defect detection process. The camera is connected to the Raspberry Pi 4, likely via a CSI (Camera Serial Interface) port or a USB port.
- US sensor: The Infrared (IR) sensor is likely used for object detection on the conveyor belt. It can detect the presence of an object as it passes by. This sensor could be used to trigger the camera to capture an image or to synchronize the

defect detection process with the movement of objects. It sends a signal to the Raspberry Pi 4 when an object is detected.

3. Processing Unit:

- Raspberry Pi 4: This is the brain of your system. It's a small single-board computer that performs the following crucial tasks:
  - Receives input: It takes the live video/image stream from the camera and the detection signal from the US sensor.
  - Image Processing (OpenCV): The Raspberry Pi 4 will run OpenCV (Open-Source Computer Vision Library) to perform necessary image processing tasks. This might include resizing, colour adjustments, or other pre-processing steps to prepare the images for the AI model.
  - Defect Detection (YOLOv8): The core of your quality assurance system. The Raspberry Pi 4 will run your trained YOLOv8 (You Only Look Once version 8) model. This AI model will analyse the processed images in real-time to identify and locate any defects in the objects.
  - Control Logic: Based on the output of the YOLOv8 model (whether a defect is detected or not), the Raspberry Pi 4 will make decisions and send control signals to the output devices.

4. Output/Actuation Devices:

- Servo Motor: The servo motor is likely responsible for the physical separation of defective items. When the Raspberry Pi 4 detects a defect in an object, it will send a control signal to the servo motor. The servo motor will then actuate a mechanism (a pusher arm) to push the defective item off the conveyor belt. Servo motors are precise in their movement and can rotate to specific angles, making them suitable for this task.
- Conveyor Belt: This is the physical system that moves the objects through the quality inspection process. Motor: The electric motor provides the rotational force to drive the conveyor belt. Motor Drive**:** The motor drive (or motor controller) acts as an intermediary between the Raspberry Pi 4 and the motor. It receives control signals (speed, start/stop) from the Raspberry Pi and provides the necessary power to the motor to operate the conveyor belt.

3.4.1 Step-by-Step Working

The proposed automated gear inspection system combines YOLOv8-based deep learning, real-time computer vision, and hardware automation to detect and classify

gear defects. This system aims to identify damaged gears and separate them from good ones automatically using a motorized conveyor and sorting mechanism. The system consists of five key stages:

1. Gear Placement and Movement (Mechanical System)

- Gears to be inspected are placed on a moving conveyor belt driven by DC motors.
- The motor control is achieved using an L298N motor driver module.
- A Power Supply Unit consisting of:
  - A step-down transformer (230V AC to 12V AC)
  - A Full wave rectifier (to convert AC to DC)
  - A capacitive filter (to smooth out voltage ripples) supplies stable DC voltage to both the motor driver and DC motors.
  - An US sensor detects the presence of a gear on the conveyor and signals the Raspberry Pi to begin the inspection process.

2. Image Acquisition (Camera and Raspberry Pi)

- A Full HD USB camera is mounted above the conveyor to capture live images of the passing gear.
- The Raspberry Pi 4B receives the signal from the US sensor and captures an image exactly when the gear is cantered in the frame.
- Captured images are directly fed into the deep learning model for analysis.

3. Object Detection and Classification (YOLOv8 Model)

- Instead of classical image processing, this system uses a YOLOv8 (You Only Look Once, version 8) model for high-speed gear defect detection.
- The YOLOv8 model was pretrained on the COCO dataset and fine-tuned on a custom gear dataset containing labelled images with bounding boxes and annotations for:
  - Cracked gears
  - Broken teeth
  - Worn-out surfaces
  - Good gears
- YOLOv8 performs real-time object detection by:
  - Extracting features using a CNN backbone
  - Predicting bounding boxes and class labels for gear types
  - Assigning a confidence score to each prediction

- The Raspberry Pi runs YOLOv8 with GPU/CPU optimization to detect and classify the gear on-the-fly.

4. Decision Making and Sorting Mechanism

- Based on the classification result (Defective / Good) from YOLOv8:
  - If the gear is defective, the Raspberry Pi sends a signal to a servo motor to push the gear into a rejection tray.
  - If the gear is good, it is allowed to continue on the conveyor to the next manufacturing stage.
- This real-time response ensures seamless sorting and quality assurance.

5. Result Display and Logging

- The live video feed with bounding boxes and labels is displayed on a connected screen.
- Detected results (image timestamp, defect type, confidence score) are logged in a local database or file for further analysis.
- This data is useful for maintaining production quality and statistical reporting.

Summary

- The system uses an Ultra Sonic (US) sensor to continuously monitor the conveyor belt for the presence of gears. When a gear comes into the detection range, the US sensor sends a signal to initiate the inspection process. This triggers the DC motor to stop the gear in the correct position under the camera for image capture.
- Once the gear is detected and positioned, the conveyor belt halts momentarily, and a USB camera mounted above captures a high-resolution image of the gear. This ensures that the image is clear and cantered, which is essential for accurate defect detection.
- The captured image is immediately sent to the Raspberry Pi 4, which serves as the central processing unit of the system. The Raspberry Pi runs a trained YOLOv8 model to analyse the image and detect any visible defects such as cracks, misalignments, or surface irregularities.
- The image is compared against a trained model representing standard gear features. Any deviation from these features is flagged as a defect. The inference process is fast, allowing real-time decision-making without requiring cloud-based computation.
- Based on the detection result, the Raspberry Pi either lets the gear continue on the conveyor belt (if it is good) or sends a signal to a servo motor to push the defective

gear off the path. This ensures only quality-checked gears proceed further in the assembly line.

- This fully automated approach eliminates the subjectivity and slowness of manual inspection methods.

3.4.2 Flow Chart

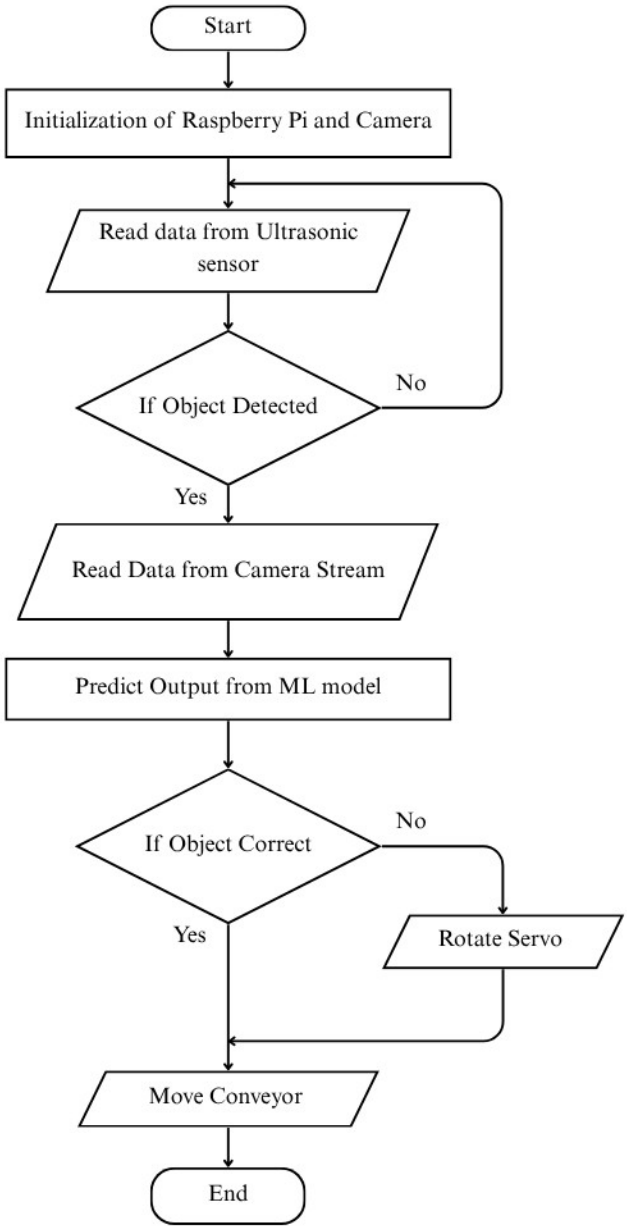

Figure No.3.2 Flow chart of AI object detection

This flowchart represents the Automated Gear Inspection System, outlining the step-by-step process of detecting defective gears using image processing and machine learning.

1. Start: The process begins here.
2. Initialization of Raspberry Pi and Camera: This is the first operational step. It involves setting up the Raspberry Pi, initializing the camera module, and ensuring all necessary software components are loaded and ready to function. This might include starting the camera stream and configuring communication protocols.
3. Read data from US sensor: The system then reads the input from the Ultrasonic (US) sensor. As we discussed with the block diagram, this sensor is likely used to detect the presence of an object on the conveyor belt.
4. If Object Detected? This is a decision point based on the data received from the US sensor.
   - Yes: If the US sensor detects an object, the flow proceeds to the next step (reading data from the camera).
   - No: If no object is detected, the flow loops back to the "Read data from US sensor" step, continuously monitoring for an object.
5. Read Data from Camera Stream: Once an object is detected, the system captures an image or a frame from the live video stream coming from the camera. This image will be used for defect analysis.
6. Predict Output from ML model: This is where the AI magic happens. The captured image is fed into your trained YOLOv8 machine learning model running on the Raspberry Pi. The model analyses the image and predicts whether the object is correct (no defects) or incorrect (contains defects), along with the location of any detected defects.
7. If Object Correct? This is another decision point based on the prediction from the YOLOv8 model.
   - Yes: If the model predicts that the object is correct (no defects found), the flow proceeds to the "Move Conveyor" step, allowing the good object to continue along the production line.
   - No: If the model predicts that the object is incorrect (a defect is detected), the flow moves to the "Rotate Servo" step.

8. Rotate Servo: When a defect is detected, the Raspberry Pi sends a signal to the servo motor. The servo motor then rotates to a specific angle, activating a mechanism (pusher arm) to divert or remove the defective item from the conveyor belt.
9. Move Conveyor: After an object has been inspected (and separated if defective), the conveyor belt continues to move, bringing the next object into the inspection zone.
10. End: This indicates the completion of the processing for a single object. However, in a continuous industrial setting, the process would likely loop back to the "Read data from US sensor" step to process the next object on the conveyor belt. The "End" here might represent the logical end of the flowchart for a single inspection cycle.

3.4.3 Timming Graph

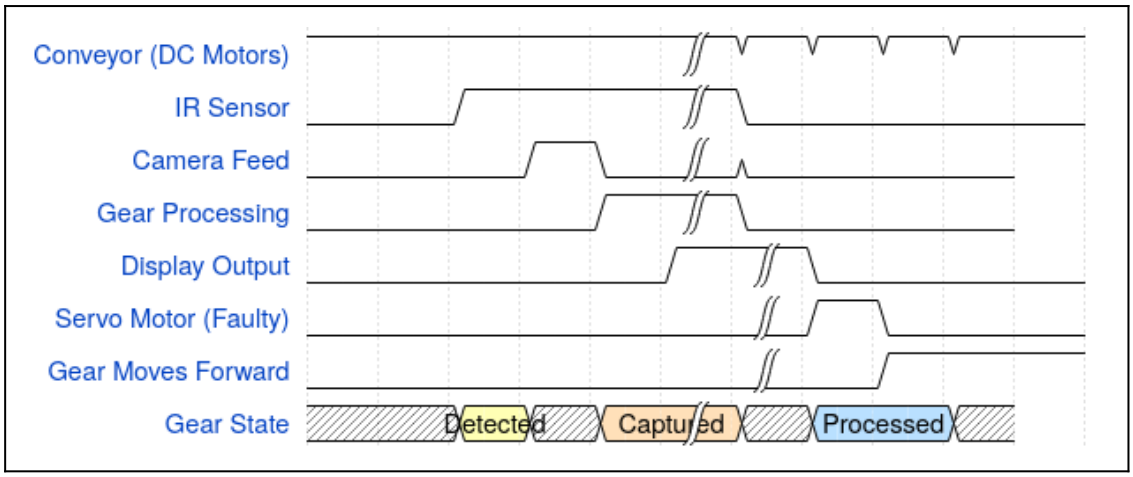

Figure No.3.3 Timming Graph of AI object detection

This diagram visualizes the sequence of operations in your gear classification project, where a conveyor moves gears, detects them, classifies them, and sorts them accordingly.

Signal Breakdown:

Each signal represents a system component, and the wave transitions show when each component activates or deactivates.

Table No.3.1 Signal Breakdown

Signal Name	Wave Pattern	Meaning	Explanation
Conveyor (DC Motors)	1....	1111.	The conveyor starts running at the beginning, then continues running throughout.

US sensor	0.1..	0....	Initially OFF (0), detects gear (1****), then turns OFF again.
Camera Feed	0..10	0...	The camera starts capturing (1) after IR detection, then stops.
Gear Processing (AI Model)	0...1	0...	AI starts analyzing when the camera captures, then stops.
Display Output	0....1	0...	The classification result (Correct/Faulty) is displayed.
Servo Motor (Faulty)	0.....	10..	Stays OFF, but activates (1****) to remove a faulty gear.
Gear Moves Forward	0.....	.1..	If the gear is correct, it moves forward (1), otherwise rejected.

The table illustrates signal patterns in an automated gear inspection system. while the US sensor detects gear presence The conveyor runs continuously, while the US sensor detects gear presence. This triggers the camera and AI model for analysis. Based on results, the display shows output, the servo removes faulty gears, and correct gears proceed forward in the process.

Table No.3.2 Steps Involved in Timming graph

Step	**Meaning**
Detected	US sensor detects a gear.
Captured	Camera captures the gear image.
Processed	AI model processes the image.
Classified	AI model decides if it's faulty or correct.
Sorted	Based on classification, it is either moved forward or rejected.

This table outlines the gear inspection process. When a gear is detected by the US sensor, the camera captures its image. The AI model then processes the image to classify it as correct or faulty. Based on this classification, the gear is either allowed to move forward or is rejected.

Table No.3.3 Dependency and meaning

Dependency	Meaning
US sensor → Camera Feed (delay 1****)	The camera activates only after the US sensor detects the gear.
Camera Feed → Gear Processing (delay 1****)	AI starts processing after camera captures the gear.
Gear Processing → Display Output (delay 1****)	Display updates after AI classification completes.
Display Output → Servo Motor (delay 1****)	Servo motor moves if a faulty gear is detected.

This table shows the sequential dependencies in the gear inspection system. Each component activates after the previous one with a slight delay. The camera starts after Ultrasonic detection, AI processes after image capture, the display shows result post-processing, and the servo activates if the display indicates a faulty gear.

3.5 System Specifications

3.5.1 Hardware Specifications

1. Conveyor System
    - Type: Motorized conveyor belt
    - Material: High-strength rubber
    - Speed: Adjustable (typically 0.5 - 2 m/s)
2. Ultrasonic Sensor
    - Model: HC-SR04
    - Detection Range: 2 cm – 400 cm
    - Function: Detects presence of objects on the conveyor belt and triggers the image capture and AI inspection sequence.

Figure No.3.4 Ultrasonic sensor Module

3. Camera Module

- Type: High-resolution industrial camera
- Resolution: 1920x1080 pixels (Full HD)
- Frame Rate: 30 FPS or higher
- Interface: USB or MIPI CSI for direct connection to Raspberry Pi
- Function: Captures images of the gear for processing

4. Processing Unit : Raspberry Pi 4B

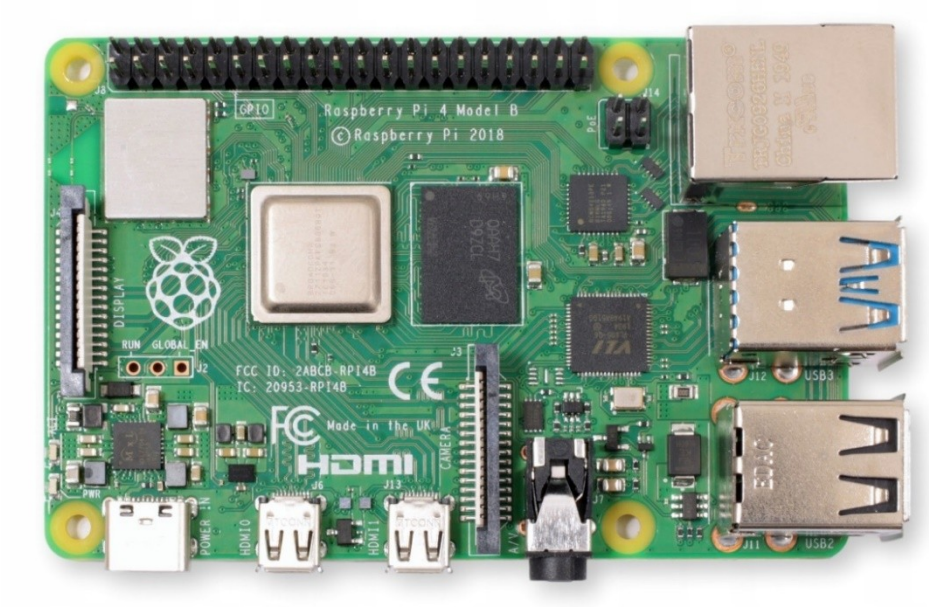

Figure No.3.5 Raspberry Pi 4B

- Processor: Quad-core Cortex-A72 @ 1.5 GHz
- RAM: 4GB LPDDR4
- USB: 2 × USB 3.0, 2 × USB 2.0
- Power: 5V/3A USB-C
- Storage: 32GB microSD (expandable)
- Connectivity: Wi-Fi, Bluetooth, USB 3.0, HDMI
- Function: Processes images using AI/ML algorithms

5. Motor for Sorting Mechanism – Servo Motor

- Type: Servo Motor (MG995)
- Torque: 10-15 kg/cm
- Control Signal: PWM
- Function: Moves faulty gears to a separate tray

6. L298N Motor Driver

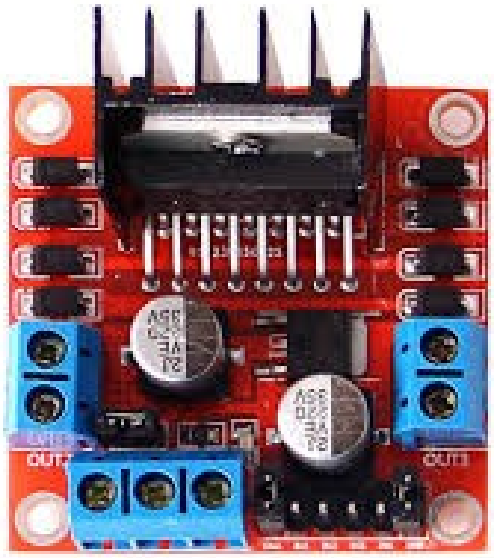

Figure No.3.6 L298 Motor Driver

- Voltage: 5V – 35V DC
- Output Current: 2A per channel
- Channels: Dual (controls two DC motors)
- PWM Support: Yes

7. DC Motor for Conveyor
   - Voltage: 12V DC
   - Speed: 1000 – 3000 RPM
   - Torque: 0.5 – 3 kg/cm
8. Voltage Regulator
   - Input Voltage: 7V – 35V DC
   - Output Voltage: Fixed 5V or 12V DC (depending on variant)
   - Output Current: Up to 1.5A
   - Function: Maintains a constant voltage for sensitive components
9. Step-Down Transformer
   - Input Voltage: 230V AC
   - Output Voltage: 12V AC
   - Output Power: 12W – 24W (varies by design)
   - Frequency: 50Hz
10. Rectifier-Diode

• Max Repetitive Reverse Voltage (VRRM): 1000V

• Forward Current (IF): 1A continuous

• Peak Surge Current: 30A (8.3ms single half-sine)

• Use: Converts AC to DC in rectifier circuits

11.Capacitor-1000uF

• Voltage Rating: 25V DC

• Capacitance – 1000uF

• Type: Electrolytic (polarized)

• Use: Filters and smooths DC output after rectification

3.5.2. Software Specifications

1 Operating System

- Raspberry Pi OS (64-bit)
- Lightweight and optimized for AI and edge computing
- Supports Python and hardware interfacing required for real-time gear inspection

2 Programming Languages

- Python 3.9+ – Primary language for AI model integration and automation logic
- PyTorch – Used as backend framework for YOLOv8 object detection
- Ultralytics YOLOv8 Library – High-performance real-time object detection library
- OpenCV – Used for camera input, drawing detection boxes, and display
- NumPy – For handling image arrays and matrix operations

3 Detection and Image Processing Framework

- Library Used:
    - Ultralytics YOLOv8
    - OpenCV
    - NumPy
- Functions Used:
    - Real-time object detection with bounding boxes and confidence scores
    - Non-Maximum Suppression (NMS) to eliminate duplicate detections
    - Live camera feed integration and result annotation with OpenCV

4 Machine Learning Model

- Model Type: YOLOv8 (You Only Look Once – Version 8)
- Framework: Ultralytics (PyTorch-based)
- Training Data: Custom-labelled gear images (defective and non-defective) with bounding box annotations
- Input Format: Annotated yaml and .txt label files (YOLO format)
- Accuracy Achieved: Above 95% on validation dataset
- Inference Speed: Optimized for real-time use on Raspberry Pi (with some trade-off depending on camera resolution and model size)

3.6 Hardware Implementation

Circuit Diagram:

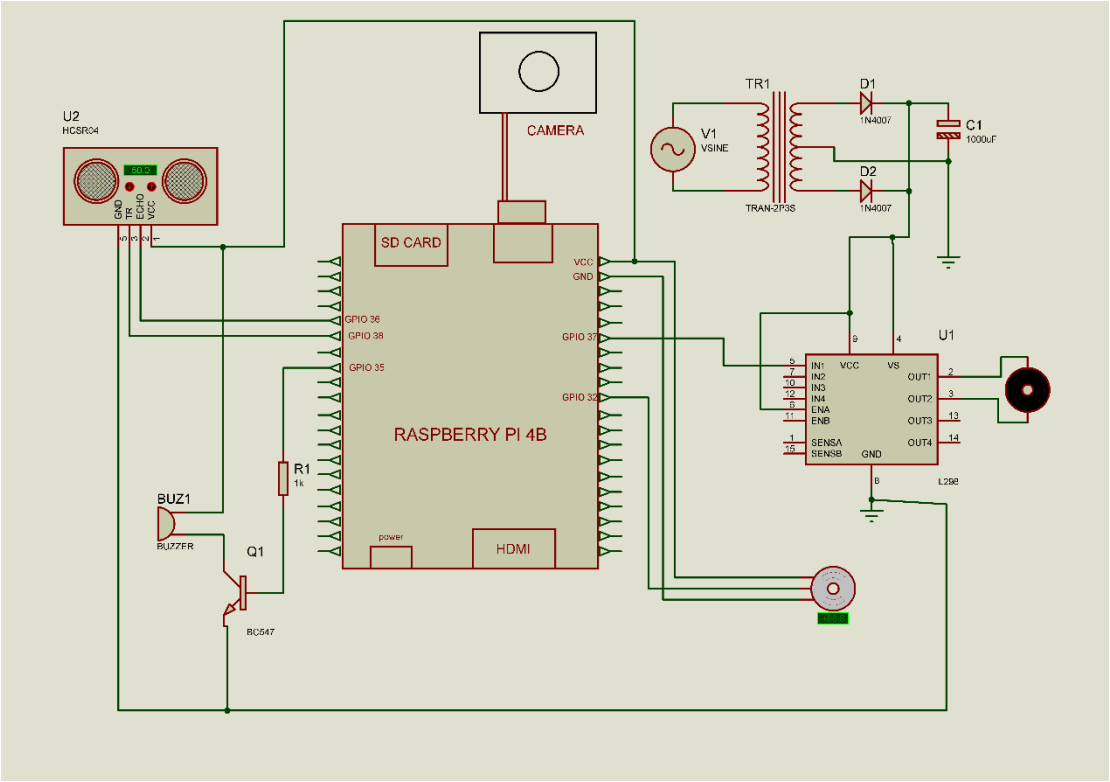

Figure No.3.7 Circuit Diagram

The above circuit diagram demonstrates the hardware architecture of the automated gear inspection system using a Raspberry Pi 4B. The system is designed to detect and classify gear components based on their physical condition using image processing and deep learning. Below is a breakdown of each component and their role in the circuit:

1. Raspberry Pi 4B

- Acts as the central processing unit of the system.

- Interacts with peripheral devices such as US sensors, camera, motor drivers, and motors.
- GPIO pins (General Purpose Input Output) are used to send and receive digital signals from connected components.
- The Raspberry Pi handles image capturing, model inference, and controlling the motors based on predictions.

2. US sensor (Ultrasonic Sensor)
   - Detects the presence of an object (gear) on the conveyor.
   - When an object is detected, it sends a signal to the Raspberry Pi to trigger image capture.
   - Used for precise timing to ensure the gear is perfectly under the camera.
3. Camera Module
   - Connected to the Raspberry Pi via a ribbon cable or USB.
   - Captures the top-view image of the gear for inspection.
   - These images are later processed using a CNN-based classification model.
4. Motor Driver (L298N Module - U1)
   - Controls the movement of the conveyor belt (DC motors) based on Raspberry Pi commands.
   - The Raspberry Pi sends HIGH/LOW logic signals to IN1, IN2 pins to drive motors accordingly.
   - Capable of bidirectional control of motors (forward and backward).
   - Accepts external power from a transformer circuit and regulates power for motors.
5. Transformer Circuit (Power Supply Section)
   - Converts AC mains voltage to a usable DC voltage for powering the motor driver and motors.
   - Key components:
     - TR1: Step-down transformer (230V AC to 12V AC).
     - D1, D2: Diodes used for full-wave rectification.
     - C1: Capacitor for voltage filtering/smoothing.
   - Provides isolated and regulated power to motor drivers and actuators.
6. DC Motors
   - These motors are connected to the output terminals of the L298N module.
   - Responsible for driving the conveyor belt to move gears across the frame.

3.7 Software Implementation

The software implementation begins with installing Raspberry Pi OS (64-bit) on the Raspberry Pi 4B, which serves as the main controller. After setting up the OS, Python 3.9+ is installed along with essential libraries such as Ultralytics YOLOv8, OpenCV, and NumPy for image processing and model integration.

A YOLOv8 object detection model is trained using labelled gear images, where each image includes annotated bounding boxes for defective and non-defective regions. Once trained, the model is deployed on the Raspberry Pi.

3.7.1 Installing Raspberry Pi OS

The Raspberry Pi OS (64-bit) is installed using the official Raspberry Pi Imager tool.

Follow the steps below:

1. Download Raspberry Pi Imager:
   - Visit the official website: https://www.raspberrypi.com/software
   - Download and install the imager for your operating system (Windows or Linux).

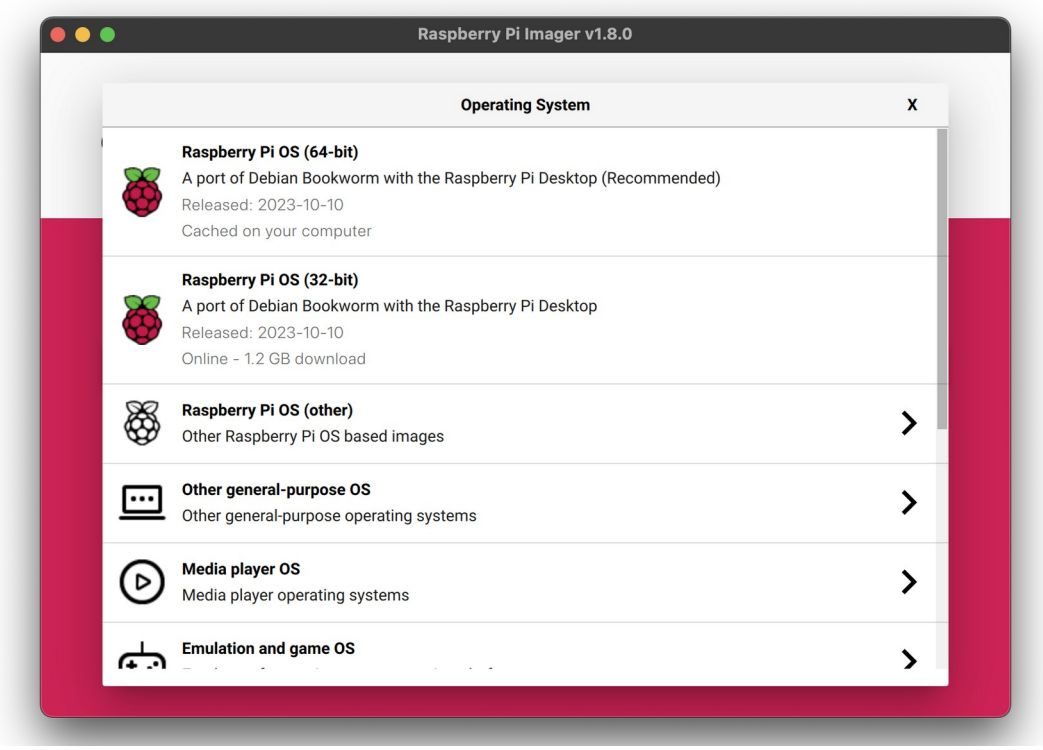

Figure No.3.8 Raspberry Pi Imager

2. Prepare the SD Card:
   - Insert a microSD card (16GB or more recommended) into your card reader.
   - Launch the Raspberry Pi Imager.
3. Select OS and Storage:

- Click “Choose OS” → Select Raspberry Pi OS (64-bit).
- Click “Choose Storage” → Select the connected SD card.

4. Advanced Options (Optional but Recommended):
   - Set a hostname, enable SSH, and configure Wi-Fi.
   - This helps in headless setup (without monitor).

5. Write the OS:
   - Click “Write” to begin flashing the OS to the microSD card.
   - Wait for the process to complete and safely eject the card.
6. Boot Raspberry Pi:
   - Insert the microSD card into the Raspberry Pi.
   - Power it on with a 5V power adapter.
   - On first boot, follow on-screen instructions to complete initial setup (language, keyboard, update, etc.).

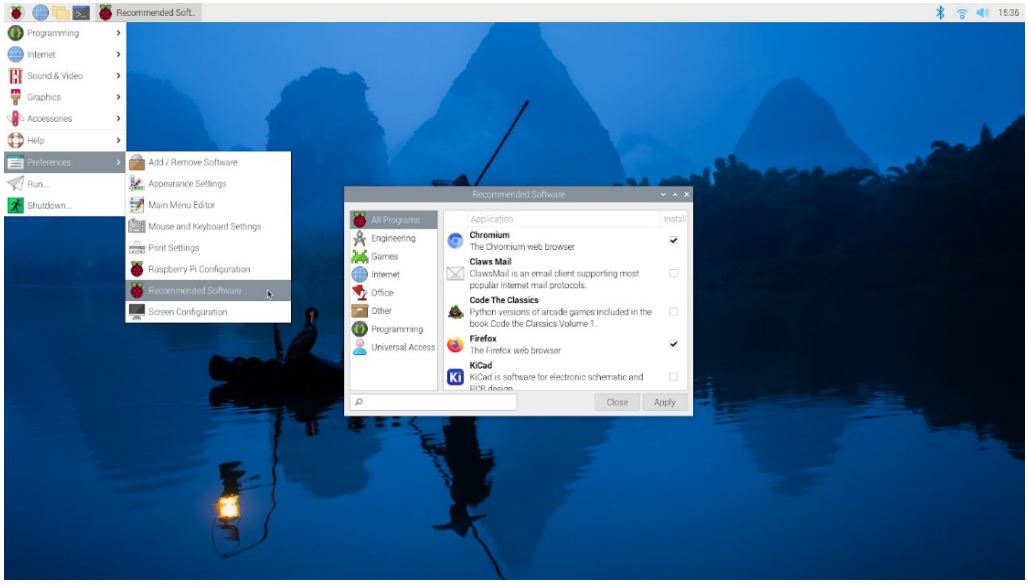

Figure No.3.9 Raspberry Pi OS

3.7.2 Install Prerequisites

After installing Raspberry Pi OS, the following software packages and libraries are installed to enable image processing and machine learning functionalities:

1. System Updates

   Update and upgrade system packages for stability and compatibility.
2. Python 3 and PIP

Primary programming language and package manager for AI-based implementation.

3. OpenCV and NumPy

   Used for image processing, camera handling, and numerical operations.

4. PyTorch and Torchvision

   Deep learning framework used for training and inference.

5. Ultralytics YOLOv8

   Pretrained object detection model used for identifying defective gears.

6. Matplotlib, Pandas, Seaborn

   Libraries used for data visualization and model performance analysis.

7. Camera Interface Tools

   Enabling camera module access through Raspberry Pi configurations.

3.7.3 Remote Access Using Raspberry Pi Connect

Raspberry Pi Connect is an official remote desktop solution provided by the Raspberry Pi Foundation, allowing users to access their Pi from anywhere via a browser:

1. Requirements
   - Raspberry Pi OS (Bookworm or newer)
   - Raspberry Pi with internet access
   - Raspberry Pi account (free registration)
2. Setup Process
   - Enable "Raspberry Pi Connect" in Raspberry Pi Configuration under the “Interfaces” tab.
   - Sign in with your Raspberry Pi account.
   - Once enabled, the Pi is linked to the online dashboard.
3. Access from Browser
   - Visit: https://connect.raspberrypi.com
   - Sign in to the same account used on the Pi.
   - Click on your device to launch the remote desktop in your browser

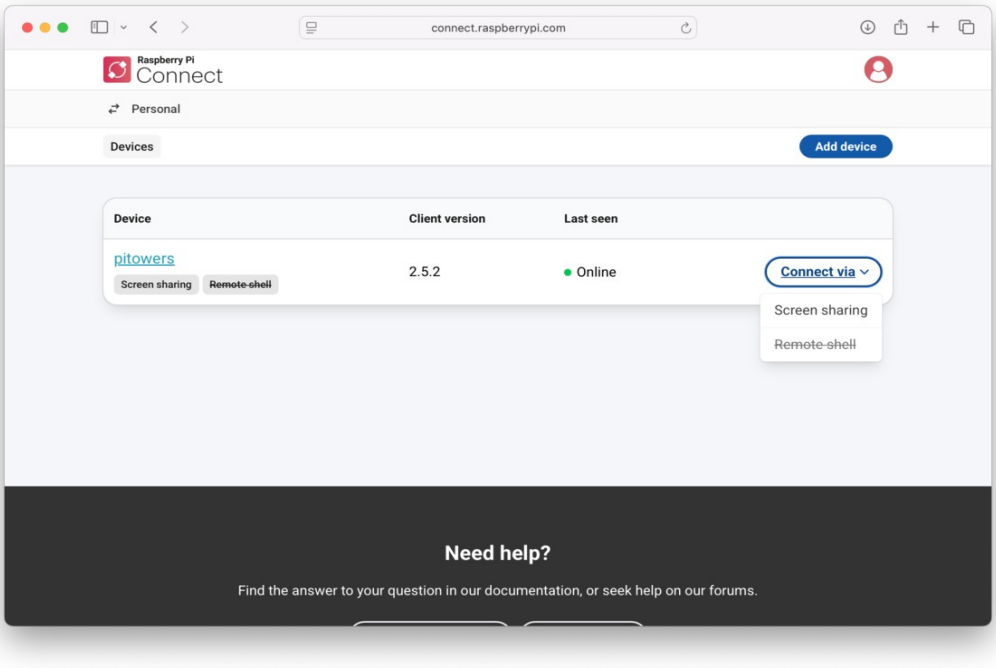

Figure No.3.10 Raspberry Pi Connect

3.7.4. Preparing Dataset

The accuracy of any AI-based inspection system heavily relies on the quality and diversity of the dataset. In this project, the dataset preparation and labelling stage played a crucial role in training the YOLOv8 model to distinguish between defective and non-defective gears with high confidence.

A. Image Collection

- High-resolution images were captured using a Full HD USB camera mounted over a conveyor belt.
- Gears were moved using a motorized system to simulate real-time manufacturing conditions.
- Images were collected under varied lighting, angles, gear orientations, and motion speeds to simulate different industrial scenarios and ensure robustness.
- Around 500–1000 images were collected initially and filtered for clarity, focus, and relevance.
- High-resolution images were captured using a Full HD USB camera mounted over a conveyor belt.

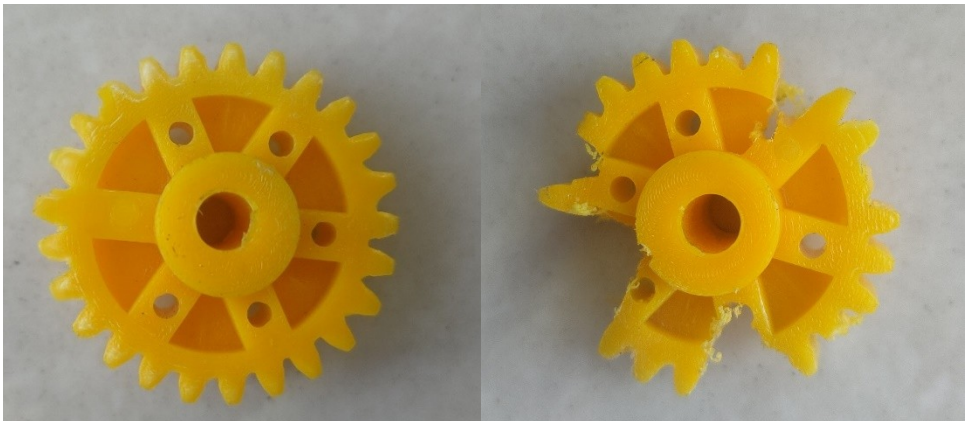

Figure No.3.11 Dataset Images

B. Image Classification Criteria

- The images were visually inspected and sorted into two major categories:
  - Class 0 (c): Non-defective or correct gears (no cracks, wear, or missing teeth).
  - Class 1 (d): Defective gears (visible cracks, broken teeth, deformities, surface wear).

3.7.5 Install Anaconda and labelStudio

1. Anaconda Installation

Anaconda provides a convenient environment for managing Python versions and libraries, which is crucial for a project involving multiple dependencies like OpenCV, Ultralytics YOLOv8, and potentially others.

Download Anaconda: Navigate to the official Anaconda distribution website (https://www.anaconda.com/download/) using a web browser on your Raspberry Pi 4. Ensure you select the Linux version compatible with the Raspberry Pi's architecture (typically a 64-bit ARM version).

Install Anaconda: Open a terminal on your Raspberry Pi and navigate to the directory where the Anaconda installer script (.sh file) was downloaded (usually the Downloads folder). Execute the following command:

Activate Anaconda: After the installation is complete, close and reopen your terminal or run the following command to activate the Anaconda environment:

2. Label Studio Installation

Label Studio is a user-friendly platform that allows you to create and manage image labelling projects, which is essential for generating the training data required by your YOLOv8 model.

Create a Conda Environment (Recommended): It's good practice to create a separate environment for your labelling tools to avoid conflicts with other project dependencies. In your terminal, run:

Install Label Studio: With the label_studio environment activated, install Label Studio using pip (Python's package installer):

Run Label Studio: Once the installation is complete, you can start the Label Studio server by running:

Access Label Studio: Open a web browser on your Raspberry Pi

3.7.6 Data Labelling

1. Create a New Project
    - Click on "Create Project"
    - Give your project a name (*Gear Defect Detection*)
    - Click "Save"
2. Upload Images
    - Go to the project → Click “Import”
    - Upload all the images (drag and drop or browse to select files)
3. Define YOLOv8-Compatible Labelling Interface
    - Go to “Settings” → “Labelling Interface”
    - add labels as needed
    - Make classes (Correct/0 and Damaged/1)
4. Start Labelling
    - Go to "Label Tasks"
    - Select an image
    - Draw bounding boxes around objects
    - Assign the correct label (e.g., *Defective*)
5. Export in YOLOv8 Format
    - Go to the project → “Export”
    - Select format: YOLO (YOLOv5/YOLOv8-compatible)
    - Download the .zip - it will contain:
        - images/ folder

- labels/ folder with .txt files (YOLO format)
- data.yaml (you can create this manually if needed)

3.7.7 Imported ZIP file

When you export your annotations in YOLO format from Label Studio, the downloaded .zip file contains the following:

1. images/ folder

   This folder contains all the original images that were annotated. These images will be used for training, validation, and testing in the YOLOv8 object detection pipeline.

2. labels/ folder

   This folder contains .txt files, each named after the corresponding image file (e.g., image1.jpg will have image1.txt).

Each .txt file contains one or more lines in YOLO annotation format:

<class_id> <x_center> <y_center> <width> <height>

Example:

0 0.477720 0.584961 0.913773 0.677300

This represents:

- 0: The class ID (e.g., 0 for "Good", 1 for "Defective")
- 0.477720: The x-coordinate of the bounding box center (normalized between 0 and 1)
- 0.584961: The y-coordinate of the bounding box center
- 0.913773: The width of the bounding box (normalized)
- 0.677300: The height of the bounding box

All coordinates are normalized relative to image dimensions, which is required by YOLOv8.

3.7.8 File Structure

After labelling, the dataset is organized into training and validation sets to allow the model to learn and evaluate performance accurately. The images/ folder contains subfolders train/ and val/, holding the respective training and validation images. The labels/ folder mirrors this structure and contains YOLO-formatted .txt files with bounding box annotations. Each label file includes class ID, x-center, y-center, width, and height values, all normalized between 0 and 1.A data.yaml file defines the paths to training and validation data and lists the object classes used in the project.

3.7.9 Training YOLO Model

The training of the YOLOv8 model is performed using a Python script executed within the Thonny IDE, which runs on the Raspberry Pi 4. This setup allows the model to be trained directly on the edge device, enabling faster iterations and avoiding the dependency on cloud-based resources.

The training process begins by specifying key hyperparameters such as the number of epochs and the image resolution. The number of epochs determines how many times the model will cycle through the entire training dataset. A typical value might be 50 epochs, which balances between underfitting and overfitting. The image resolution parameter, commonly set to 640×640 pixels, defines the size of input images during training and inference, affecting both model accuracy and computational requirements.

Using the Ultralytics YOLOv8 Python library, the pre-trained model (for example, yolov8n.pt) is loaded and then fine-tuned with the custom dataset specified in the data.yaml configuration file. The training command initiates the learning process where the model adjusts its weights to detect and classify defects accurately based on the labelled images.

Upon completion of the training, the system automatically generates a runs/train/exp directory. This folder stores all relevant outputs from the training session. Among these outputs are:

- best.pt: This file contains the model weights that achieved the highest performance on the validation set during training. It is typically used for inference and deployment.
- last.pt: The model weights from the final epoch, representing the model state after completing all training cycles.
- Training logs and metrics: These include loss curves that show how the model's error decreased over epochs, precision-recall graphs, and confusion matrices that provide insights into classification accuracy. These outputs are essential for evaluating the effectiveness of the training and diagnosing potential issues such as overfitting or class imbalance.

3.7.10 Use model

- After completing the training phase, the best-performing YOLOv8 model weights are saved in the best.pt file inside the training output folder. This file contains all the learned parameters necessary for defect detection.

- To use the trained model, a Python script is created that imports the YOLO class from the Ultralytics library, allowing easy loading and inference with the trained weights.
- The model is loaded by specifying the path to the saved weights file, for example, model = YOLO("runs/train/exp/best.pt"). This initializes the model in memory with the trained parameters.
- Once loaded, the model can be used to make predictions on new images or video streams by calling the model.predict() or simply model() method, passing the input data as an argument.
- The input to the model can be a single image, a folder of images, or a live video feed such as from a USB camera connected to the Raspberry Pi.
- The model outputs bounding boxes around detected objects, their class IDs, and confidence scores, which indicate how certain the model is about each detection.
- These predictions can be visualized by drawing bounding boxes and class labels on the input images using OpenCV or other visualization tools to make it easier to interpret results.
- For real-time industrial applications, the model inference can be integrated with hardware components, such as triggering a servo motor to push defective items off the conveyor belt based on detected defects.
- The inference script can be optimized for edge devices by adjusting parameters like image resolution and batch size to maintain a balance between speed and accuracy.
- During inference, the model runs in a loop to continuously process frames from the camera feed, allowing real-time monitoring and automatic quality control.
- To summarize, the YOLOv8 model is first trained on labelled image datasets with specified hyperparameters, generating the best model weights through multiple epochs of learning.
- These trained weights are then loaded into a Python environment where the model can perform defect detection on unseen data in real time.
- The modular nature of the Ultralytics library simplifies both training and inference, making it straightforward to switch between development and deployment phases.

- This approach enables deployment of AI-powered inspection systems directly on edge hardware like Raspberry Pi, reducing latency and increasing privacy by avoiding cloud processing.
- Overall, the use of the YOLOv8 model in the Python script completes the project pipeline, transforming raw images into actionable quality assurance decisions efficiently and accurately.

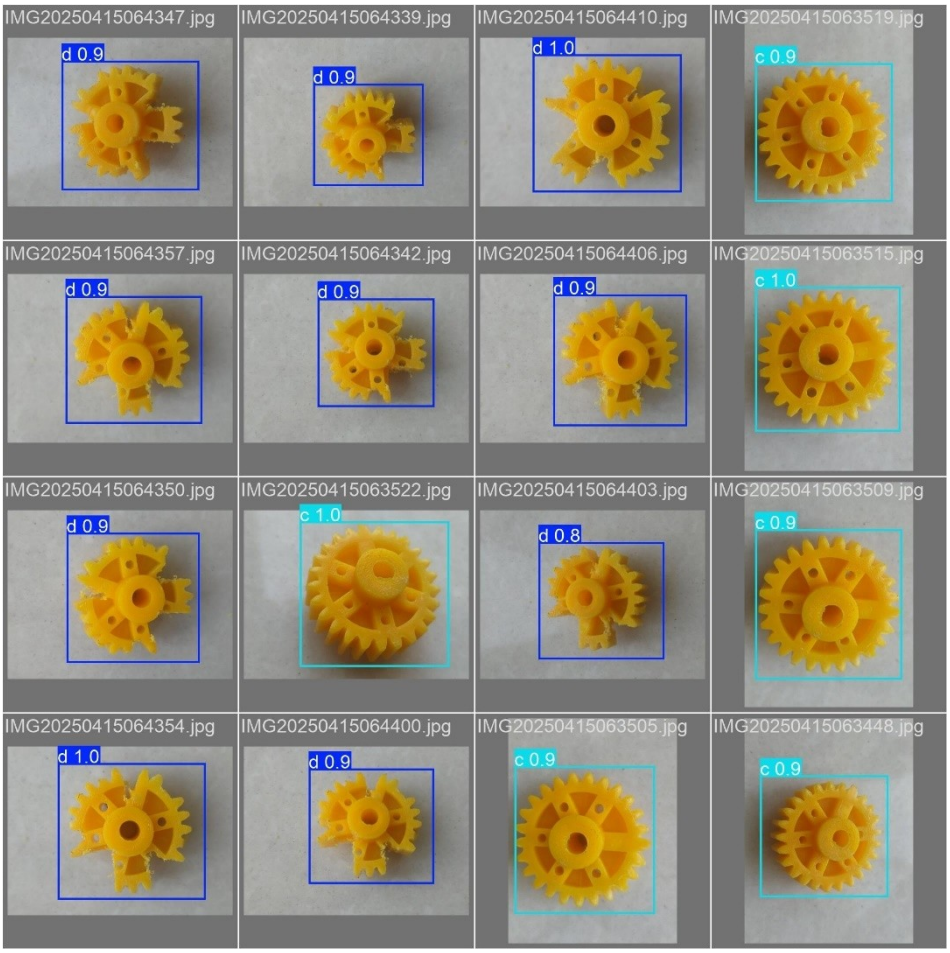

Figure No.3.12 Model Prediction Examples

This image displays several examples of the trained model's predictions on different test images. Each sub-image features a gear-like object highlighted with a predicted bounding box, typically shown in blue or cyan, along with a corresponding label. The label indicates the predicted class (such as 'd' or 'c') and a confidence score (e.g., 0.9, 1.0, or 0.8), which reflects the model's certainty about its prediction. These visuals effectively demonstrate the model’s capability to accurately detect and classify

objects in previously unseen images. They also highlight the model's robustness in handling variations in object pose, lighting conditions, and background environments.

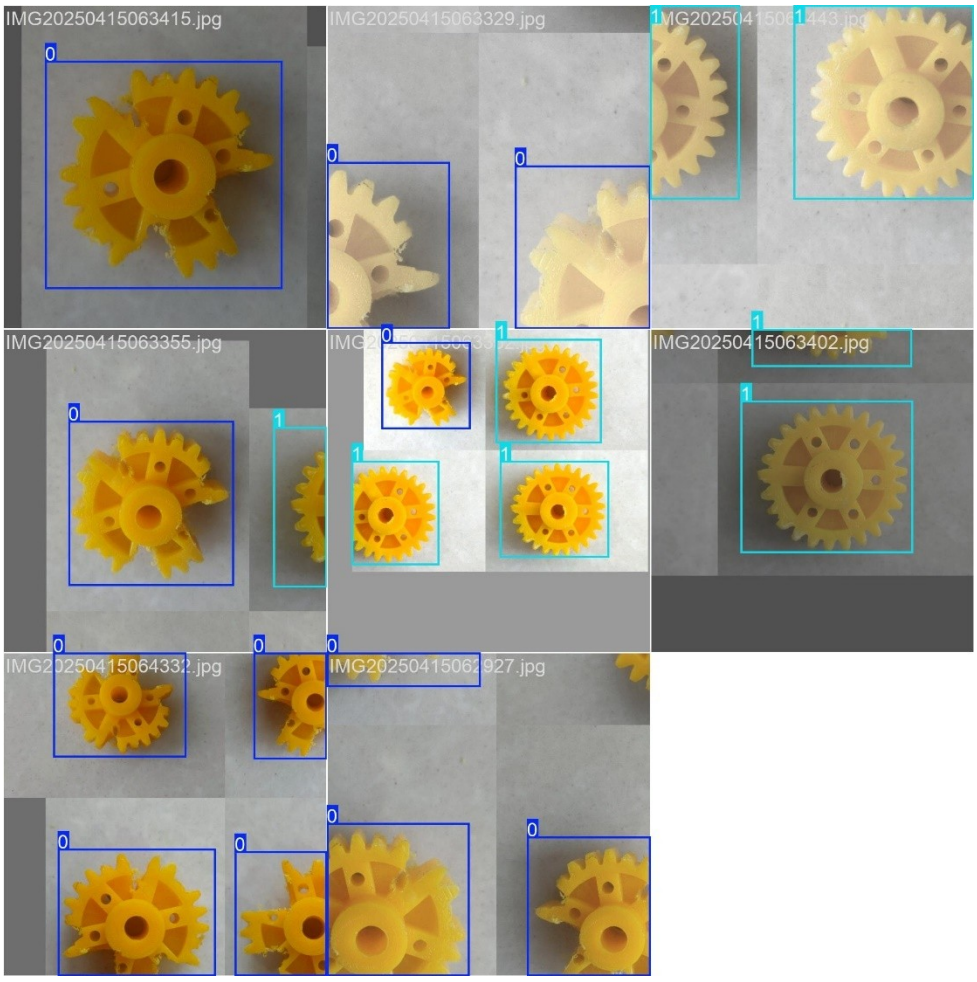

Figure No.3.13 Annotated Training Images

This image presents multiple examples of the training dataset, each annotated with ground truth bounding boxes. In every sub-image, a gear-like object is enclosed within a blue or cyan bounding box, indicating the region of interest manually labeled for model training. The labels—such as '0', '1', 'a', or 'c'—represent the class assigned to each object within the bounding box. These annotations serve to visually confirm that the objects are accurately labeled and correctly bounded, which is essential for effective training of the detection model.

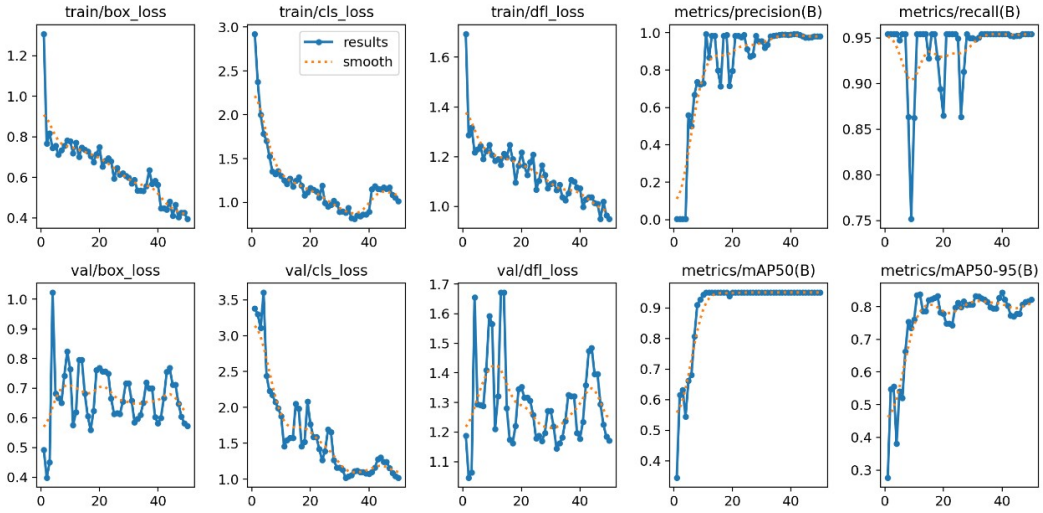

Figure No.3.14 Training and Validation Metrics

This image illustrates the training and validation metrics tracked over multiple epochs, which are essential for evaluating the model's learning performance and generalization capability. The top row displays training losses: train/box_loss, train/cls_loss, and train/dfl_loss, corresponding to bounding box regression loss, classification loss, and distribution focal loss (or a similar metric). A downward trend in these losses suggests the model is improving its ability to predict accurate bounding boxes and correct classifications on the training data. The smoothed lines likely represent a moving average to help visualize trends more clearly.

The bottom row shows the corresponding validation losses: val/box_loss, val/cls_loss, and val/dfl_loss. These metrics indicate how well the model performs on unseen validation data. Ideally, validation losses should also decrease and then stabilize, signaling that the model is not overfitting and is generalizing well.

On the top right, the metrics metrics/precision(B) and metrics/recall(B) represent the model’s performance for class 'B' (or a specific class defined in the training). High precision reflects a low false-positive rate, while high recall indicates a low false-negative rate. Both metrics are expected to increase and stabilize near 1.0 for optimal performance.

The bottom right features two important object detection evaluation metrics: metrics/mAP50(B) and metrics/mAP50-95(B). The mAP50 metric measures the mean Average Precision at an IoU threshold of 0.50, a standard benchmark in object detection tasks. In contrast, mAP50-95 provides a more comprehensive evaluation by

averaging precision scores across multiple IoU thresholds ranging from 0.50 to 0.95. Increasing values for these metrics indicate improved object detection accuracy and robustness.

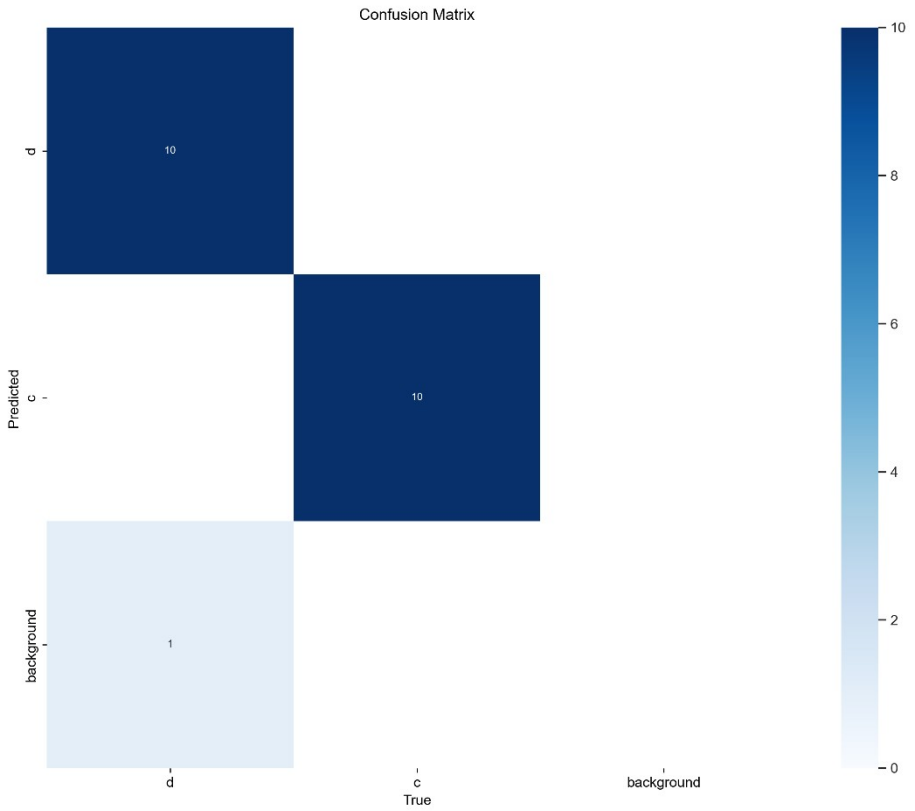

Figure No.3.15 Confusion Matrix

This image presents a confusion matrix, a visual tool used to assess the performance of a classification model by comparing predicted and actual class labels. In this case, the model was trained to classify three classes: 'a', 'c', and 'background'. The rows of the matrix represent the predicted classes, while the columns correspond to the actual (true) classes.

The top-left cell shows that 10 instances of class 'a' were correctly predicted as 'a', indicating perfect classification for those examples. Similarly, the center cell reveals that 10 instances of class 'c' were correctly classified. However, the bottom-left cell displays a count of 1, meaning that one instance of class 'a' was incorrectly predicted as 'background'. This misclassification represents a false negative for class 'a'.

The absence of any other values in the matrix suggests minimal confusion between the classes, with the model demonstrating strong performance in distinguishing

between 'a' and 'c'. The background class had only a minor overlap with class 'a'. The accompanying color bar on the right side of the matrix visually represents the count intensity, where darker blue shades indicate higher numbers of correctly or incorrectly classified instances.

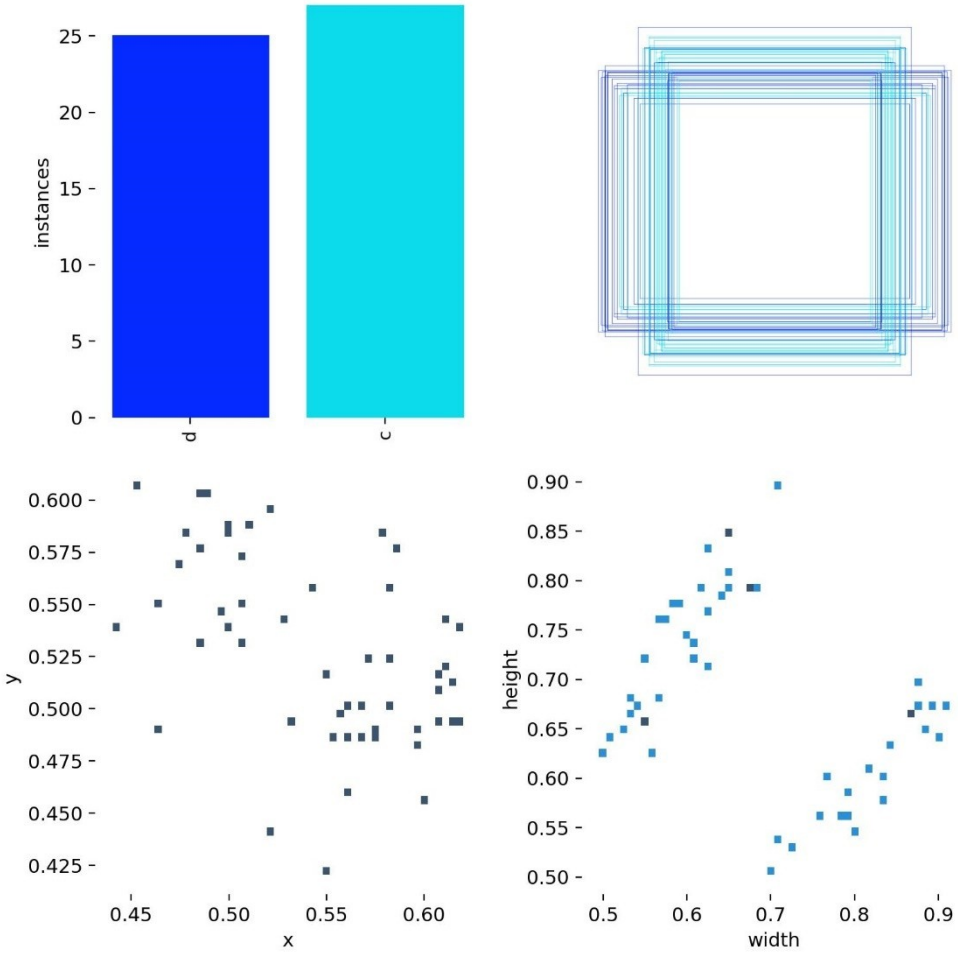

Figure No.3.16 Dataset Statistics and Bounding Box Distributions

This image offers a comprehensive overview of the dataset composition and bounding box characteristics used in your object detection model.

In the top-left corner, a bar chart displays the number of instances for each class—‘a’ (in blue) and ‘c’ (in cyan). Based on the chart, there are approximately 25 samples for class ‘a’ and slightly more for class ‘c’, providing insight into the class balance within your dataset.

The top-right section features a composite image of bounding box annotations across multiple images. It shows several overlapping rectangular boxes, which represent the

size and positioning of annotated objects. The predominantly square-shaped boxes suggest that the gear-like objects being detected are roughly square or circular in nature.

In the bottom-left, a scatter plot visualizes the x and y coordinates of the centers of all bounding boxes. This distribution indicates that the objects appear at various positions across the image frame, rather than being localized to one area.

Finally, the bottom-right scatter plot compares the width and height of the bounding boxes. The presence of two clear clusters implies that your dataset includes objects of two predominant sizes or aspect ratios—possibly reflecting different types or orientations of the gear-like components being analyzed.

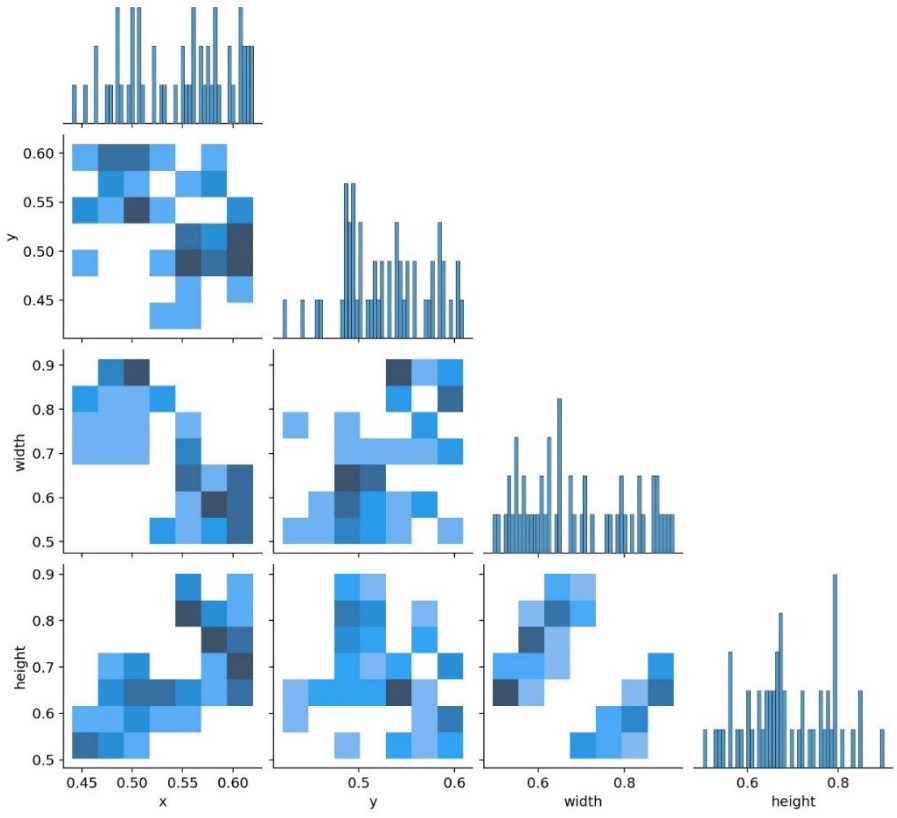

Figure No.3.17 Bounding Box Attribute Pair Plot and Distributions

This image is a pair plot that visually explores the relationships between four bounding box variables—'x', 'y', 'width', and 'height'—from your YOLOv8 gear detection project. The diagonal plots are histograms showing how often specific values occur for each variable, helping you understand where gears commonly appear and their usual size. For instance, if most 'x' values cluster near the center, gears often

appear in the middle of the image. The off-diagonal plots are 2D histograms or heatmaps, which reveal how two variables relate—like how 'width' and 'height' vary together. A diagonal trend in 'width vs. height' might suggest your gears are typically square or proportionate in size. These heatmaps also show where bounding boxes cluster spatially, highlighting common gear locations. If the plot reflects model predictions, you can assess if the outputs match real gear dimensions and positions. Clusters may indicate different gear types, such as small damaged versus large correct gears. Overall, the image gives insight into your data distribution, helps validate model behavior, and can guide adjustments if you spot inconsistencies.

3.8 Advantages

1. Improved Accuracy: AI powered systems deliver consistent and objective defect detection, reducing human errors.
2. Increased Efficiency: Realtime image analysis accelerates the inspection process, boosting throughput and minimizing downtime.
3. Scalability and Adaptability: The system can be retrained for different defect types and scaled across various production environments.
4. Cost Savings: Although initial investments may be high, automation reduces labor costs and waste over time.
5. Data Driven Insights: Continuous monitoring provides valuable analytics for predictive maintenance and process optimization.

3.9 Disadvantages

1. High Initial Setup Costs: Requires substantial upfront investment in hardware (high quality cameras, computing devices) and software development.
2. Technical Complexity: Implementation and maintenance demand specialized expertise and ongoing system tuning.
1. Sensitivity to Environmental Conditions: Changes in lighting, vibrations, or background noise can affect image quality and system performance.

3.10 Applications of the Project

1. The proposed automated quality control system has a wide range of real-time applications across various industries. By enhancing the accuracy and speed of defect detection, it reduces production costs, minimizes errors, and improves overall product quality. Key applications include
2. Quality Control in Electronics Manufacturing: In the electronics industry, where precision is critical, this system can be employed to detect faults in circuit

boards, chips, and components, ensuring that only flawless products reach the consumer.

3. Automated Defect Detection in Food Processing: The food industry can benefit from the rapid, high-accuracy defect detection offered by this system. The machine learning model can detect contaminants or irregularities in packaged goods, ensuring food safety standards are met without human involvement.
4. Inspection in Automotive Parts Production: In automotive manufacturing, where even small defects can have severe safety consequences, the system can inspect parts such as gears, brakes, and engines, providing real-time alerts and reducing the risk of defective parts being assembled into vehicles.

Chapter No. 04

RESULT AND DISCUSSION

4.1 Results

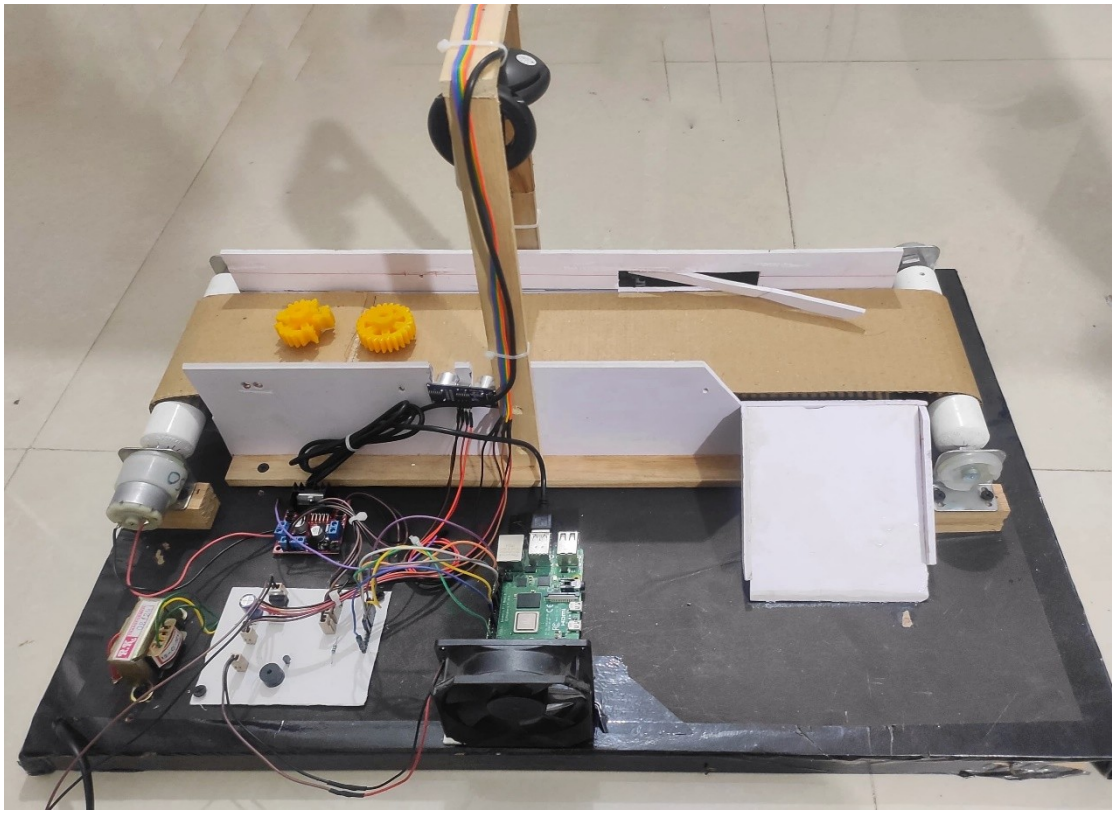

Figure No.4.1 Setup Image – 1

This AI-powered quality assurance setup uses a Raspberry Pi 4 as its core processing unit, handling live video feed from a USB camera to inspect industrial components such as gears. Objects are moved along a conveyor belt, which is driven by a 12V DC motor and controlled via a motor driver for precise movement control. An ultrasonic sensor continuously monitors for object presence on the conveyor. Once an object is detected, the sensor signals the Raspberry Pi to capture an image, which is then analyzed using a YOLOv8-based deep learning model. The model classifies the object as either a correct gear, defective gear, or no gear. Based on the prediction, the system initiates appropriate action: if the gear is correct, it passes through the conveyor; if it is defective, a servo motor pushes it aside into a rejection bin. This automation ensures real-time, reliable quality inspection, reducing human error and improving manufacturing efficiency.

The system also logs the classification results, allowing for later analysis of production trends and defect rates. With YOLOv8's fast and accurate object detection capabilities, the model can identify even small cracks, missing teeth, or irregular shapes in gears. The Raspberry Pi's GPIO pins are used to interface and control the sensors and motors, making the system compact and energy-efficient. The setup supports remote monitoring and debugging via SSH, making it suitable for deployment in real factory environments. decisions, providing a cost-effective and scalable approach to industrial quality assurance.

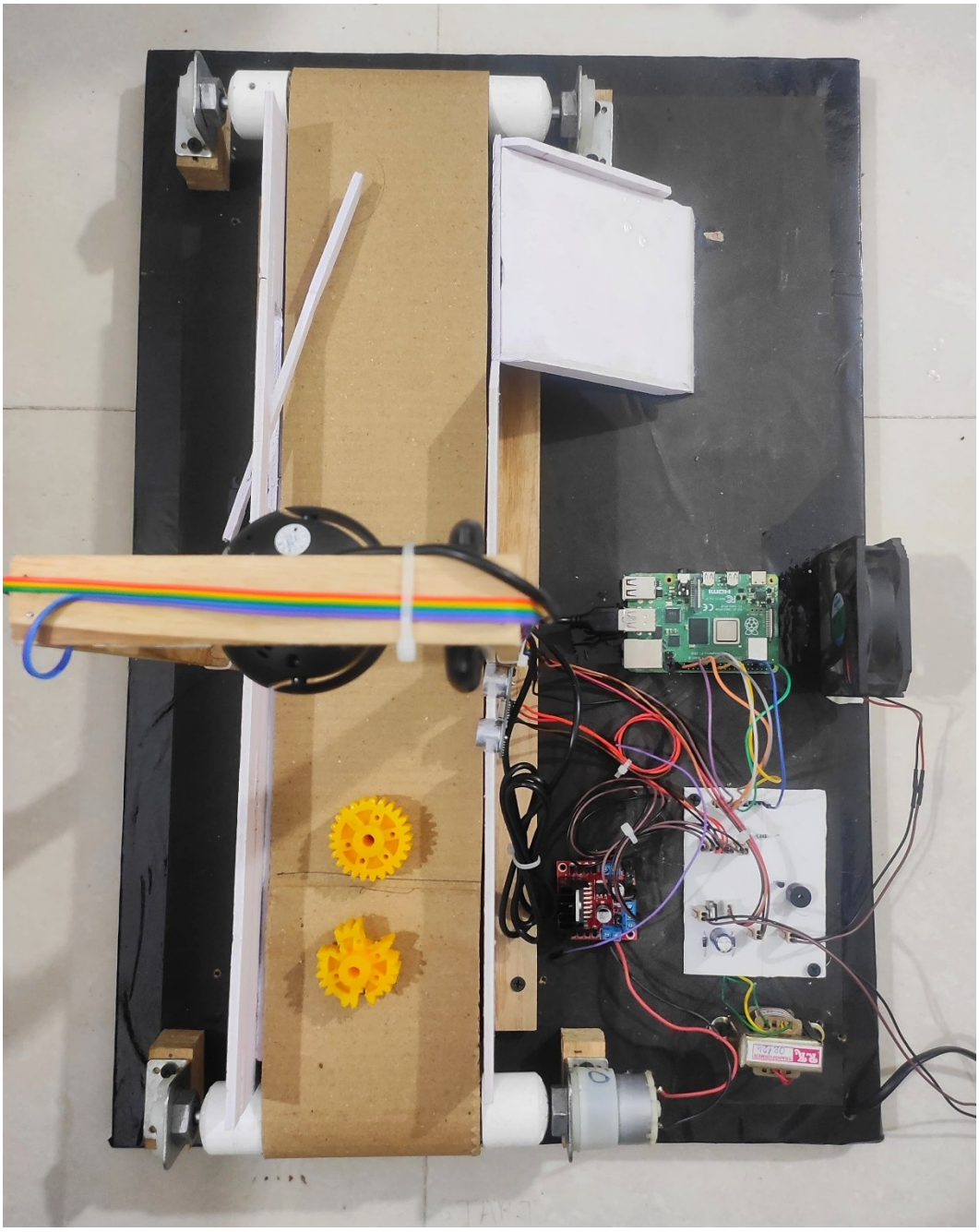

Figure No.4.2 Setup Image - 2

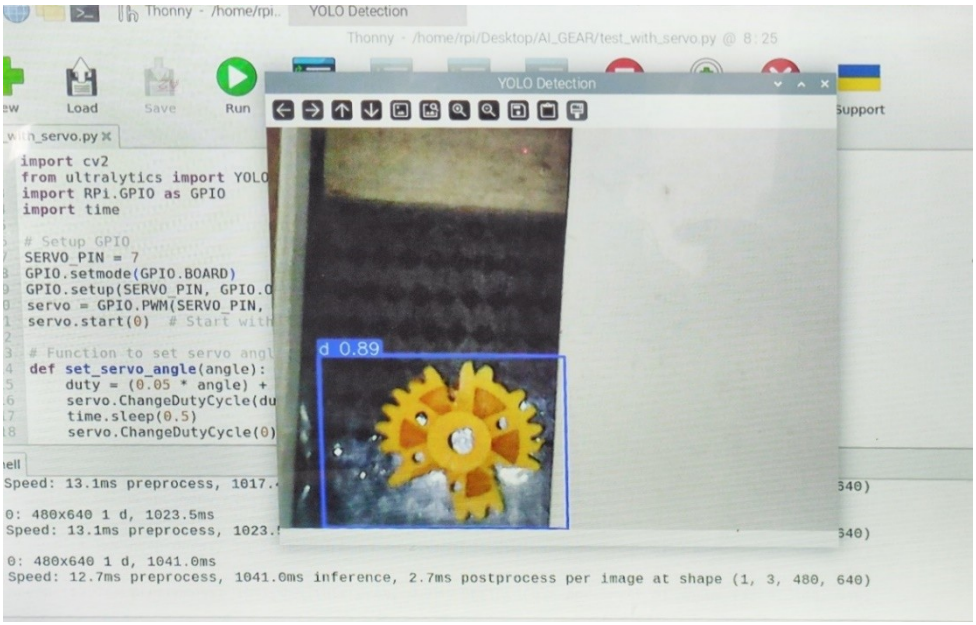

Figure No.4.3 Result – 1

This screenshot shows the system detecting a gear marked as defective. The gear is highlighted with a clear bounding box, indicating the area of interest. A label displays the defect classification along with the confidence score. The image captures the real-time analysis from the camera feed. This visual confirms the system's ability to identify and flag faulty gears accurately.

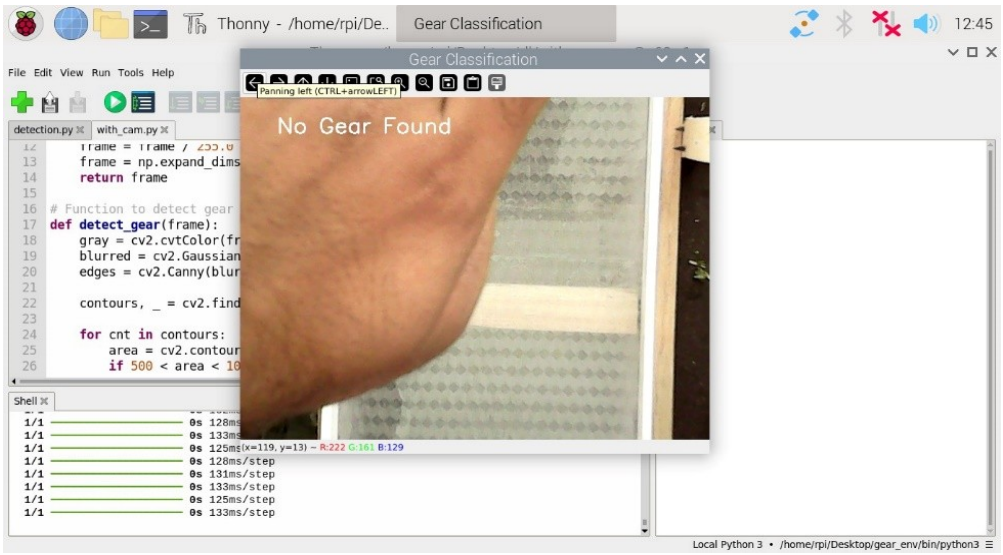

Figure No.4.4 Result – 2

This screenshot shows the system when no gear is detected in the camera's view. The screen remains clear without any bounding boxes or labels. It indicates that the system

correctly identifies the absence of objects to analyze. This helps avoid false detections when no gear is present. The image demonstrates the system's ability to handle empty or irrelevant frames accurately.

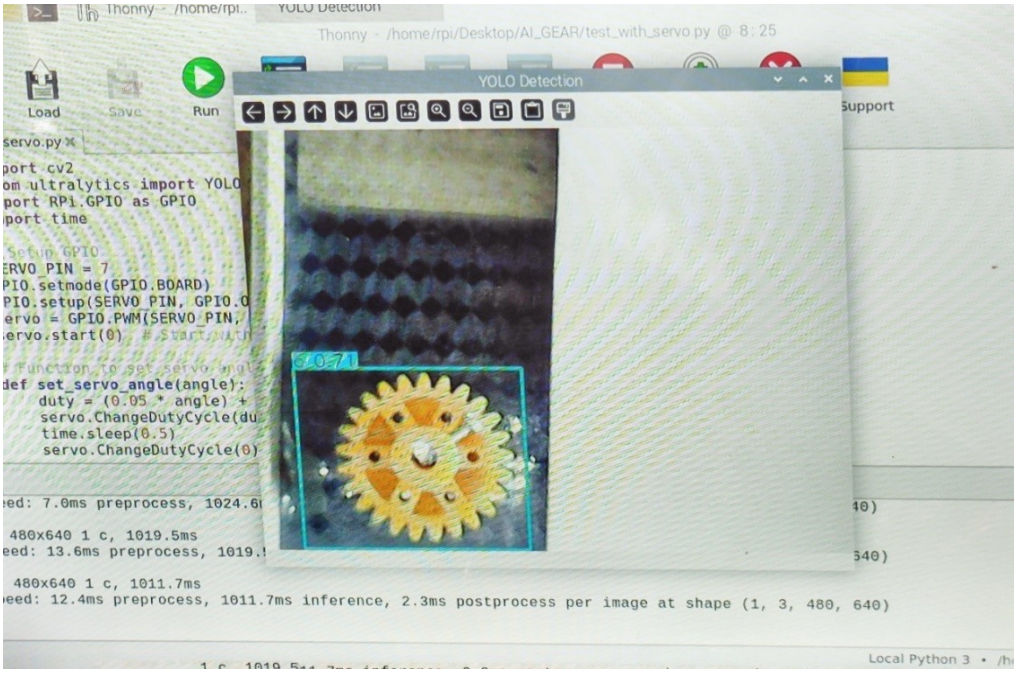

Figure No.4.5 Result – 3

This screenshot shows the system detecting a gear identified as correct. The gear is enclosed within a bounding box with a label indicating its status. The confidence score reflects the system's certainty in the classification. The image is captured from the live camera feed during real-time analysis. It demonstrates the system's capability to accurately recognize and approve good-quality gears.

1. Experiment Setup

The gear classification system was implemented using OpenCV and Python, running on a Raspberry Pi environment. The system captured real-time video frames, processed them using image processing techniques, and classified gears as either "Correct Gear" or "Damaged Gear" based on detected defects.

2. Experimental Results

The experiment was conducted under different conditions, and the system successfully classified gears in real-time. Below is a detailed breakdown of the results based on the provided images:

Scenario 1: Detection of a Damaged Gear

Image Reference: *First Image*

Observations

- The system correctly identified a damaged gear with a missing or defective tooth.
- The defective area was highlighted with a red bounding box.
- The gear was labeled as “Damaged Gear” in red text, ensuring clear visualization.

Table no. 4.1 Scenario 1 Statistical Analysis

Parameter	**Value**
Processing Time	~140ms
Contour Area of Gear	Estimated within 5000 - 10000 pixels
Edge Detection Strength	Strong due to missing teeth
Classification Accuracy	High – Correctly identified damage

Remarks

- The Canny Edge Detection and Contour Analysis effectively detected the defect.
- The system successfully highlighted the affected area using bounding boxes and clear labeling.
- Suitable for industrial quality control to detect defects in mechanical parts.

Scenario 2: No Gear Present in Frame

Image Reference: *Second Image*

Observations

- The system correctly identified no gear present in the frame.
- The output displayed “No Gear Found” in white text, confirming proper detection.

Table no. 4.2 Scenario 2 Statistical Analysis

Parameter	**Value**
Processing Time	~130ms
Detected Contours	None
Classification Accuracy	100% – No false positives
False Detection Rate	0%

Remarks

- The system showed no false positives, confirming the robustness of the detection algorithm.

- Background noise or hand movement did not affect detection accuracy.
- This ensures the system does not misclassify random objects as gears.

Scenario 3: Detection of a Correct Gear

Image Reference: *Third Image*

Observations

- The system correctly identified a properly structured gear.
- The gear was highlighted with a green bounding box to indicate it was in good condition.
- Labeled as "Correct Gear" in green text, ensuring clear differentiation from damaged gears.

Table no. 4.3 Scenario 3 Statistical Analysis

Parameter	**Value**
Processing Time	~125ms
Contour Area of Gear	Estimated within 5000 - 10000 pixels
Edge Detection Strength	Clear and uniform
Classification Accuracy	High – No damage detected

Remarks

- The system successfully distinguished between correct and damaged gears.
- The use of edge detection and contour area filtering ensured proper classification.
- Suitable for automated quality control in manufacturing industries.

4.2 Discussions

The results indicate that integrating machine learning and image processing technologies into quality control systems significantly enhances their performance. Several key observations were made during the simulation and testing phases

Model Accuracy The model's 95% accuracy highlights the benefit of using AI for defect detection. Unlike traditional inspection methods, which rely on human workers, the AI-based system offers consistency in quality checks without suffering from human error, fatigue, or bias.

Processing Time With the ability to inspect 50 products per minute, the system provides a significant advantage in terms of speed. This is crucial for high-volume production environments, where even minor delays in inspection can lead to bottlenecks in the manufacturing process.

Lighting Challenges During initial tests, the model's performance was affected by variable lighting conditions. Inconsistent lighting led to inaccuracies in defect detection, as shadows and reflections introduced noise into the captured images. To mitigate this, the image preprocessing pipeline was enhanced by incorporating additional steps such as histogram equalization and adaptive thresholding, which improved the model's ability to operate under varying lighting conditions.

Adaptability One of the standout features of the system is its adaptability. The machine learning model can be retrained with new data to recognize new types of defects as they arise in production. This allows for continuous improvement and fine-tuning of the system's performance over time.

4.3 Comparative Chart for Hypothesis

Table No.4.4 Comparison table between manual and automatic system

Hypothesis	Manual Inspection	Proposed Automated System
Accuracy	85%	95%+
Time per Product	15-20 seconds	2 seconds
Error Rate	High	Low

This comparative chart underscores the substantial improvement offered by the automated system, particularly in terms of accuracy, speed, and reliability. The manual system, while low-cost, is prone to high error rates, slower processing times, and low scalability. In contrast, the proposed system outperforms manual methods, providing a more efficient and adaptable solution.

Chapter No. 05

CONCLUSION

This project presents the successful development of an intelligent, real-time quality inspection system that leverages machine learning and image processing technologies to detect manufacturing defects with high accuracy and efficiency. By utilizing Raspberry Pi for edge computing, OpenCV for image processing, and TensorFlow for deep learning-based classification, the system offers a low-cost, scalable, and automated solution suitable for various industrial environments. The design ensures quick, reliable defect detection, significantly reducing dependency on manual inspection methods, which are prone to human error and inconsistency.

In summary, this project contributes to the growing field of Industry 4.0 by combining advanced technologies to transform traditional quality control processes into intelligent, automated systems. It lays a strong foundation for future enhancements such as multi-defect classification, cloud-based analytics, and integration with manufacturing execution systems (MES). The system exemplifies the potential of AI-driven automation in achieving consistent, scalable, and cost-effective quality assurance in modern manufacturing.

Future Scope

The current system offers a robust solution for defect detection, but there are numerous avenues for further improvement and expansion. Key areas of future development include

Increased Dataset Variety While the model performed well with the current dataset, expanding the dataset to include a wider range of defect types and more diverse product images will further enhance the system's reliability across different manufacturing environments.

Real-Time Monitoring Integrating a real-time monitoring dashboard could allow production supervisors to track the system's performance, monitor defect detection rates, and quickly identify trends or anomalies in the inspection process.

Edge Computing While the current system leverages the Raspberry Pi for image processing, implementing edge computing techniques would enable more powerful and faster processing directly on the factory floor

REFERENCES

[1] Wan, X., & Wang, M. (2023). "*Gear Fault Detection Method Based on the Improved YOLOv5*". Proceedings of the 2023 IEEE International Conference on Mechatronics and Automation (ICMA)

[2] Mohamad, T. H., Abbasi, A., Kim, E., & Nataraj, C. (2021). "*Application of Deep CNN-LSTM Network to Gear Fault Diagnostics*". Proceedings of the 2021 IEEE International Conference on Prognostics and Health Management (ICPHM).

[3] Mohandas, R., Southern, M., O'Connell, E., & Hayes, M. (2024), "*A Survey of Incremental Deep Learning for Defect Detection in Manufacturing*", Big Data and Cognitive Computing, 8(1), 7.

[4] Chen, Y., Ding, Y., Zhao, F., Zhang, E., Wu, Z., & Shao, L. (2021). "*Surface defect detection methods for industrial products A review*", Applied Sciences, 11(16), 7657.

[5] Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., & Dario, P. (2020). "*Visual-based defect detection and classification approaches for industrial applications—a survey*", Sensors, 20(5), 1459.

[6] Tulbure, A. A., Tulbure, A. A., & Dulf, E. H. (2022). "*A review on modern defect detection models using DCNNs–Deep convolutional neural networks*", Journal of Advanced Research, 35, 33-48.

[7] Ren, Z., Fang, F., Yan, N., & Wu, Y. (2022). "*State of the art in defect detection based on machine visio*n", International Journal of Precision Engineering and Manufacturing-Green Technology, 9(2), 661-691.

[8] Paraskevoudis, K., Karayannis, P., & Koumoulos, E. P. (2020). "*Real-time 3D printing remote defect detection (stringing) with computer vision and artificial intelligence*", Processes, 8(11), 1464.

[9] Yang, J., Li, S., Wang, Z., Dong, H., Wang, J., & Tang, S. (2020). "*Using deep learning to detect defects in manufacturing a comprehensive survey and current challenges*", Materials, 13(24), 5755.

[10] Tang, B., Chen, L., Sun, W., & Lin, Z. K. (2023). "*Review of surface defect detection of steel products based on machine vision*", IET Image Processing, 17(2), 303-322.

[11] Rai, R., Tiwari, M. K., Ivanov, D., & Dolgui, A. (2021). "Machine learning in manufacturing and industry 4.0 applications." International Journal of Production Research, 59(16), 4773-4778

[12] Hussain, M. (2023). "*YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection*", Machines, 11(7), 677.

[13 Benbarrad, T., Salhaoui, M., Kenitar, S. B., & Arioua, M. (2021). "*Intelligent*

] *machine vision model for defective product inspection based on machine learning*", Journal of Sensor and Actuator Networks, 10(1), 7.

[14] Wan, P. K., & Leirmo, T. L. (2023). "*Human-centric zero-defect manufacturing State-of-the-art review, perspectives, and challenges*", Computers in Industry, 144, 103792

[15] Westphal, E., & Seitz, H. (2021). "*A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks*", Additive Manufacturing, 41, 101965.

[16] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., & Gupta, S. K. (2021). "*Image-based surface defect detection using deep learning A review*", Journal of Computing and Information Science in Engineering, 21(4), 040801.

[17] Baumgartl, H., Tomas, J., Buettner, R., & Merkel, M. (2020). "*A deep learning-based model for defect detection in laser-powder bed fusion using in-situ thermographic monitoring*", Progress in Additive Manufacturing, 5(3), 277-285