**A DISSERTION ON**

MACHINE VISION GUIDED ASSEMBLY STATION

**THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF**

**MASTER OF TECHNOLOGY IN**

**ROBOTICS AND AUTOMATION**

SUBMITTED BY

**ZEPHIN S GEORGE**



**Under the Guidance of Dr. Arunkumar Bongale**

**Dr. Manoj Shukla**

**SYMBIOSIS INSTITUTE OF TECHNOLOGY**

**(A CONSTITUTENT OF SYMBIOSIS INTERNATIONAL UNIVERSITY) Pune – 412115**

**YEAR 2022-24**

i

## CERTIFICATE

The thesis entitled Mobile Power on Station submitted to the Symbiosis Institute of Technology, Pune for the award of M. Tech in Robotics and Automation is based on my original work carried out under the guidance of Dr. Arunkumar Bhongale and Dr. Manoj Shukla .The dissertation has not been submitted elsewhere for award of any degree.

The material borrowed from other source and incorporated in the dissertation has been duly acknowledged and/or referenced.

I understand that I myself could be held responsible and accountable for plagiarism, if any, detected later on.

Date: 19/04/2023

## ZEPHIN S GEORGE Signature of the candidate

**Research supervisor HOD**

Dr. Satish Kumar Dr. Arunkumar Bongale

**CONTENTS**

|  |  |
| --- | --- |
| **LIST OF FIGURES** | i |
| **LIST OF TABLES** | iv |
| **LIST IF ABBREVIATION** | v |
| **ABSTRACT** | vi |
| **CHAPTER 1: INTRODUCTION** | 1 |
| **1.1 Introduction to Automation** | 1 |
| **1.2 Necessity of Automation in mobile industry** | 1 |
| **1.3 Motivation of study** | 2 |
| **1.4 Automation system for testing mobile power on condition** | 3 |
| **1.5 Research objectives** | 4 |
| **1.6 Summary** | 5 |
| **CHAPTER 2: Literature Review** | 6 |
| **2.1 PLC based automation** | 6 |
| **2.2 PC and PLC based control system** | 7 |
| **2.3 Computer vision in automation** | 7 |
| **2.4 Deep learning in computer vision** | 8 |
| **2.5 Neutral network and deep learning** | 10 |
| **2.6 Hierarchical feature learning** | 12 |
| **2.7 Depth of deep learning** | 15 |
| **2.8 Object detection algorithms** | 17 |
| **2.8.1 Histogram of Oriented Gradients** | 18 |
| **2.8.2 Region based convolution neural networks** | 19 |

|  |  |
| --- | --- |
| **2.8.3 Faster R-CNN** | 21 |
| **2.8.4 Single shot detector** | 22 |
| **2.8.5 YOLO** | 24 |
| **2.8.6 RetinaNet** | 25 |
| **2.9 Summary** | 27 |
| **CHAPTER 3: Methodology** | 28 |
| **3.1 Data Acquisition** | 30 |
| **3.1.1 Advantages of the GigE interface** | 32 |
| **3.1.2 Communication with code reader** | 33 |
| **3.1.3 Connection with code reader** | 34 |
| **3.1.4 Image configuration** | 34 |
| **3.1.5 Algorithm selection** | 35 |
| **3.1.6 Input and output control** | 35 |
| **3.1.7 Data processing** | 38 |
| **3.1.8 Communication Protocol** | 39 |
| **3.1.9 Configuration Management** | 40 |
| **3.1.10 Assigning IMEI to mobile puck** | 41 |
| **3.2 Power up device** | 43 |
| **3.2.1 Hardware capabilities** | 43 |
| **3.2.2 Pneumatic operations** | 46 |
| **3.2.3 PLC integration with pneumatic system** | 47 |
| **3.2.4 Pneumatic calculations of the system** | 48 |
| **3.3 Video processing** | 50 |
| **3.3.1 High level architecture of single shot object detector** | 51 |
| **3.3.2 YOLOv5 Architecture** | 54 |
| **3.3.3 CSP Darknet 53** | 58 |
| **3.3.4 Neck of YOLOv5** | 61 |
| **3.3.5 Head of network** | 63 |

|  |  |
| --- | --- |
| **3.3.6 Activation Function** | 64 |
| **3.3.7 Loss Function** | 65 |
| **3.3.8 Other Improvements** | 67 |
| **3.3.9 Hardware Integration** | 69 |
| **3.3.10 Communication Protocol** | 71 |
| **3.3.11 Camera configuration** | 78 |
| **3.3.12 Summary** | 81 |
| **CHAPTER 4: Results and Discussions** | 82 |
| **4.1 Obtained result of trained model** | 82 |
| **4.2 Validation of model** | 84 |
| **4.3 Analysis of PR curve** | 85 |
| **4.4 Summary** | 88 |
| **CHAPTER 5: Conclusion** | 89 |
| **5.1 Process status extraction** | 89 |
| **5.2 Automating process of Power on mobile device** | 89 |
| **5.3 Device power on confirmation** | 90 |
| **5.4 Future directions** | 91 |
| **CHAPTER 6 : Internship Report** | 92 |
| **6.1 Mechatronics system** | 92 |
| **6.2 PLC Programming Introduction** | 94 |
| **6.3 PLC Logic Implementation** | 101 |
| **6.4 Communication Protocols** | 104 |
| **6.4.1 Fieldbus** | 104 |
| **6.4.2 TCP/IP** | 105 |
| **6.4.3 Ethernet IP** | 106 |
| **6.4.4 USB Protocol** | 107 |
| **6.4.5 Socket Communication** | 108 |
| **6.5 Python programming language** | 110 |

|  |  |
| --- | --- |
| **6.6 Multiprocessing in programming** | 112 |
| **6.7 Multithreading in programming** | 117 |
| **6.8 Pymodbus library** | 119 |
| **6.9 Pyserial library** | 120 |
| **6.10 Deepclean system** | 122 |
| **6.11 Device profile extraction** | 131 |
| **6.12 Sponge height measurement proof of concept** | 137 |
| **6.13 Image Augmentation** | 154 |
| **6.14 Fanuc M-10id/12 Robot** | 163 |
| **BIBLIOGRAPHY** | 170 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Figure 1.1 | 3D Model of setup |
| Figure 2.1 | Venn diagram of deep learning |
| Figure 2.2 | Perceptron network architecture |
| Figure 2.3 | XOR dataset |
| Figure 2.4 | Multilayer feedforward network architecture |
| Figure 2.6 | Deep learning and traditional learning performance |
| Figure 2.7 | HOG object detection algorithm |
| Figure 2.8 | R CNN object detection algorithm |
| Figure 2.9 | Faster R CNN objection detection algorithm |
| Figure 2.10 | SSD Object detection algorithm |
| Figure 2.11 | YOLOv5 architecture |
| Figure 2.12 | RetinaNET object detection algorithm |
| Figure 3.1 | Methodology diagram |
| Figure 3.2 | Smart code reader |
| Figure 3.3 | TCP/IP Protocol |
| Figure 3.4 | TCP data transmission |
| Figure 3.5 | Data acquisition system |
| Figure 3.6 | Beckhoff C6015 |
| Figure 3.7 | Proximity sensor |

|  |  |
| --- | --- |
| Figure 3.8 | Power up device system |
| Figure 3.9 | One stage detector |
| Figure 3.10 | YOLOv5 performance |
| Figure 3.11 | YOLO Architecture |
| Figure 3.12 | Applying CSPNet to RESNet |
| Figure 3.13 | Bottle NeckCSP module architecture |
| Figure 3.14 | Structure of SPPE block |
| Figure 3.15 | SiLU and ReLU |
| Figure 3.16 | Eliminating grid sensitivity |
| Figure 3.17 | Nvidia xavier |
| Figure 3.18 | ADS Protocol |
| Figure 3.19 | ADS Architecture |
| Figure 3.20 | Camera |
| Figure 3.21 | Video processing system |
| Figure 4.1 | Confusion Matrix |
| Figure 4.2 | Logo detection on mobile device |
| Figure 4.3 | P curve |
| Figure 4.4 | PR curve |
| Figure 4.5 | Model training performance |
| Figure 6.1 | Mechatronics system |
| Figure 6.2.1 | PLC architecture |
| Figure 6.2.2 | Internal architecture of PLC |
| Figure 6.3.1 | Ladder Diagram of problem statement |

|  |  |
| --- | --- |
| Figure 6.4.1 | Fieldbus architecture |
| Figure 6.4.2 | TCP/IP Model |
| Figure 6.4.3 | Ethernet/IP |
| Figure 6.4.4 | USB Architecture |
| Figure 6.4.5 | Socket Communication |
| Figure 6.10.1 | Deepclean |
| Fig 6.10.2 | Innovance Motor and Drive architecture |
| Fig 6.12.1 | Baumer Camera Explorer |
| Figure 6.12.2 | Green colour sponge |
| Figure 6.12.3 | Blue colour sponge |
| Figure 6.12.4 | Yellow colour sponge |
| Figure 6.12.5 | HSV output |
| Figure 6.12.6 | Extracting HSV values |
| Figure 6.12.7 | Obtained white pixels |
| Figure 6.12.8 | Obtained black pixels |
| Figure 6.12.9 | Detection of contour |
| Figure 6.12.10 | Detection of height |
| Figure 6.11.11 | Images of barcode label |
| Figure 6.11.12 | Python program of Image augmentation |
| Figure 6.12.1 | Fanuc M-10iD/12 |
| Figure 6.12.2 | Teach pendant and controller |
| Figure 6.12.3 | Coordinate system -world/tool/user |

**(Annexure 4.5)**

**LIST OF TABLES**

|  |  |
| --- | --- |
| Table 3.1 | Scanner details |
| Table 3.2 | Hardware details in power up device |
| Table 3.3 | YOLOv5 model performance |
| Table 3.4 | Hardware used in video processing |
| Table 6.3.1 | Runtime Test Case |
| Table 6.3.2 | Input and output |

**(Annexure 4.6)**

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
|  | Definition |
| PLC | Programmable logic controller |
| SCADA | Supervisory control and data acquisition |
| DCS | Distributed Control system |
| RLC | Relay logic and contactor |
| ADS | Automation device specification |
| YOLO | You only look once |
| IMEI | International mobile equipment identity |
| CSI | Camera serial interface |
| QR | Quick response |
| GigE | Gigabyte Ethernet |
| POE | Power Over Ethernet |
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| SMPS | Switch mode power supply |
| DC | Direct Current |
| CSP | Cross stage partial network |
| PAN | Path Aggrigation Network |
| TP | True Positive |

|  |  |
| --- | --- |
| FN | False Negative |
| CNN | Convolution Neural Network |
| mAP | Mean Average Precision |

# Abstract

The manual process of testing mobile devices can be time-consuming and prone to errors. To address this issue, an automated system for logo detection on mobile screens and turning on mobile devices is proposed. The system utilizes an object detection algorithm for logo detection and on-screen display detection, controlled by a Programmable Logic Controller (PLC) to press the power button of the device. The system also involves a mobile puck, a mobile carrying unit placed on a conveyor, which simplifies the loading process of the device. The proposed system streamlines the process of functional testing by checking the process status of the device from a database, and passing it through the system if the process status is registered as cosmetic grading done. This system bypasses devices that are not ready for testing, reducing the time required for the overall testing process. The YOLO algorithm continuously captures images of the mobile phone under a camera to detect the logo on the device. By automating the task of turning on the device, the proposed system increases testing efficiency and reduces human error. Overall, this system provides an effective solution to the challenges of manual mobile device testing.

**Chapter 1**

# Introduction

## Introduction to Automation

In recent years, there has been a rapid increase in the use of industrial automation systems in various manufacturing industries. Industrial automation refers to the use of advanced technologies, such as robotics, artificial intelligence, and sensors, to automate industrial processes and improve efficiency, quality, and safety. The implementation of industrial automation systems has significantly impacted the manufacturing industry, leading to increased productivity, reduced costs, and improved quality. However, the implementation of such systems also presents challenges, such as the need for specialized skills, high initial costs, and potential job displacement. This thesis aims to explore the design and implementation of industrial automation systems, their benefits and challenges, and their impact on the manufacturing industry. The use of automation in manufacturing has been around for decades, with the first recorded use of machines in the textile industry in the 18th century. Since then, the implementation of automation systems has evolved and become more advanced, leading to increased efficiency and productivity. Today, industrial automation systems are used in various sectors, including automotive, electronics, food and beverage, and pharmaceuticals.

## Necessity of Automation in Mobile Industry

Automation has become a necessity in the mobile manufacturing and testing industry due to several reasons. Mobile phones have become an integral part of our daily lives, and the demand for them has only increased with time. With the increasing demand, mobile manufacturers need to keep up with the pace to meet the customer's expectations. Automation plays a crucial role in this regard, making it necessary for the industry to adopt automation technologies. Firstly, automation ensures consistency in the production process, making it easier for manufacturers to maintain high-quality standards. In mobile manufacturing, where even the slightest mistake can result in a defective product, automation can significantly reduce the likelihood of errors. Automated systems can perform repetitive tasks with accuracy and speed, reducing the need for human intervention. This, in turn, ensures that each device manufactured meets the same standards of quality, minimizing the possibility of defects or malfunctioning. Secondly,

automation improves efficiency and reduces production time. Mobile manufacturers need to produce millions of devices each year to meet the growing demand. Automation allows manufacturers to produce more devices in less time, increasing their productivity and profitability. Automated systems can perform tasks faster than humans, reducing the production time and allowing manufacturers to meet their deadlines. Thirdly, automation plays a significant role in testing mobile devices. Mobile testing involves several stages, such as functional testing, performance testing, and compatibility testing, among others. Automated testing systems can perform these tasks quickly and accurately, reducing the time required for testing. This, in turn, reduces the time required to bring the product to the market, ensuring that it meets the consumer's demands and preferences. Lastly, automation in the mobile manufacturing and testing industry reduces labor costs. Automated systems can perform tasks that were previously done by humans, reducing the need for manual labor. This not only reduces labor costs but also minimizes the possibility of errors that can occur due to human error.

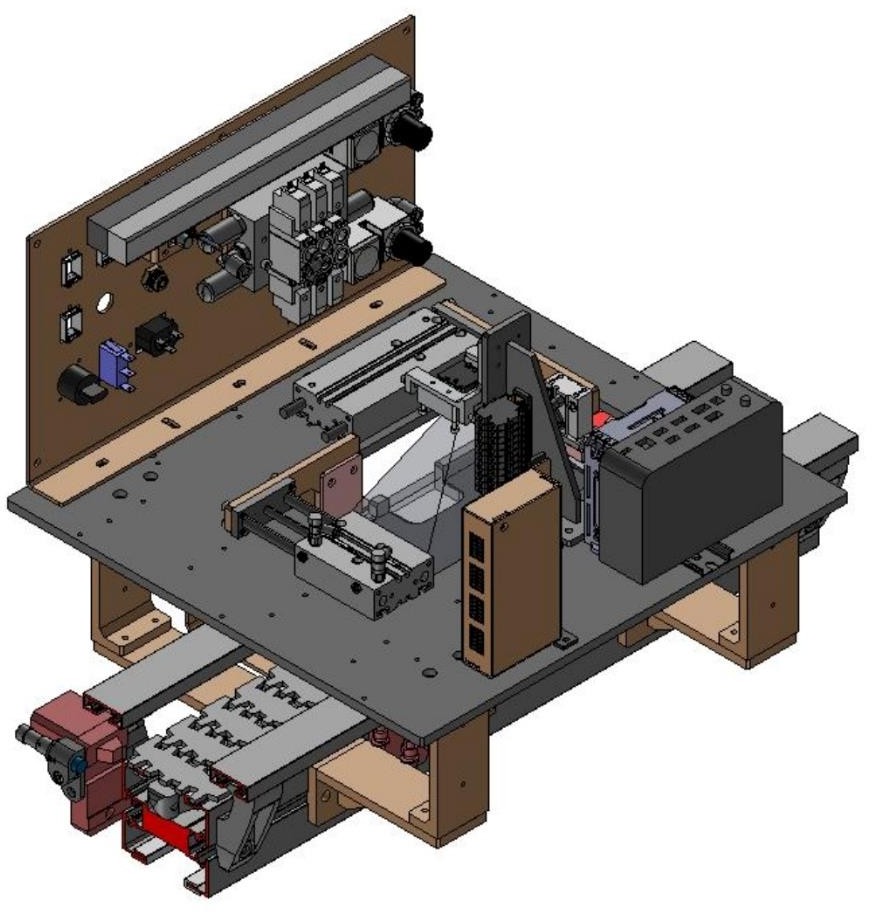
## Motivation of study

Mobile devices have become an essential part of our daily lives, and their usage has only increased with time. With the increasing demand for mobile devices, it has become essential for manufacturers to ensure that the devices are of high quality and meet the customer's expectations. However, the production and testing process of mobile devices can be time-consuming and prone to errors, leading to delays and defects. Automation systems have been developed to address these issues, improving efficiency and reducing the likelihood of errors. However, while automation systems are widely used in the mobile manufacturing industry, they are not as common in the mobile testing process. This gap in automation can lead to delays in the testing process and increase the possibility of defects. Therefore, there is a need to develop an automation system for mobile testing that can address these issues. The automation system can perform tasks such as functional testing, performance testing, and compatibility testing, among others, with speed and accuracy. By automating the testing process, it will be possible to reduce the testing time and minimize the possibility of errors, ensuring that high-quality devices are delivered to customers.

The motivation for this study is to develop an automation system that can address the gaps in the mobile testing process. The study will focus on developing an automation system that can perform mobile turning on and testing tasks and ensure that the devices meet the required quality standards. By developing such a system, it will be possible to improve the efficiency of the testing process, reduce the likelihood of defects, and ensure that the devices meet the customer's expectations.

## Automation system for testing mobile power on condition

Mobile phone testing is one of the processes which is done manually in any market or in the production line of mobile units. The basic process to turn on a mobile device is to press the power button manually until the logo is detected on the device screen and wait for the device screen to turn on its display. If the logo on the mobile screen appears and the mobile display screen turns on then the mobile is considered in working condition. Currently, this task is often done manually by human intervention. Turning the device on is the most crucial step before performing functional testing of any device. In this work, an automation system for Logo detection on mobile screens and turning on a mobile device is discussed. The system includes object detection algorithm for logo detection and on screen display detection with mechanical plungers for pressing the power button of a device operated by a programmable logic controller (PLC) which is referred from earlier work[4]. The task of the person using this system is to simply load the mobile phone in a mobile puck which is mobile carrying unit placed on a conveyor. The process status of device is checked from the database, if the process status of device is registered as cosmetic grading done in database then mobile device is passed in the system for turning it on else it is bypassed by the system. Continues captures of mobile phone under the camera are taken and are feed into the You only look once (YOLO) algorithm for detecting the logo on the device.



## Figure 1.1: 3-D Model of setup

Processor and Controller shares data with shared memory block where the register value is updated, as the logo is detected on device the value associated with respective memory register is updated and plunger is released is pushed in reverse direction and power button of device is released. Similarly, when the background of mobile device is changed while turning it on those images are feed into object detection algorithm. After prediction of on screen by algorithm another register value associated with shared memory is updated and device is released from station according to the controller program. Finally, the process status of device is changed from cosmetic grading completed to power on completed.

## Research Objectives

* + 1. To check process status of mobile device
    2. To power on mobile device using Industrial PC and Pneumatic integrated system
    3. To monitor mobile turning on process using object detection algorithm

## Summary

This chapter covered the automation overview, automation has become a necessity in the mobile manufacturing and testing industry due to its benefits, including improved efficiency, consistency, and accuracy. Automation technologies have the potential to revolutionize the mobile industry, making it possible for manufacturers to meet the growing demand for mobile devices while maintaining high-quality standards. With the increasing competition in the mobile market, manufacturers who fail to adopt automation technologies risk falling behind their competitors. Therefore, it is imperative for mobile manufacturers to embrace automation technologies to remain competitive and meet the changing needs of consumers. The development of an automation system for mobile testing is essential in today's mobile industry. By automating the testing process, it will be possible to reduce the time required for testing, minimize the possibility of defects, and ensure that high-quality devices are delivered to customers. This study aims to develop an automation system that can address the gaps in the mobile testing process, contributing to the improvement of the mobile industry's overall efficiency and quality.

**Chapter 2**

# Literature review

## PLC based automation

In 2017, Mallikarjun Hudedmani highlighted in their paper that the programmable logic controller (PLC) is the primary controlling element in the industry or a process. When run efficiently, with safety factors taken into account, desired goals can be achieved [1]. By identifying and investigating the capabilities of PLCs for any process, the study distinguishes between current automation systems and earlier technology. In the past, relay logic and contactor logic (RLC), which involve human error and intervention, were used. However, the development and use of microprocessors, controllers, and specialized tools like PLCs, Supervisory control and data acquisition (SCADA), Distributed control systems (DCS) have led to an increase in productivity, accuracy, precision, and efficiency. These systems have boosted process control flexibility while reducing human intervention. The term "automation" refers to the incorporation of mechanisms and control sequences in the correct order repeatedly while allowing for allowable variations in the process output. This has made the benefits of automation clear. The growth in automation systems has enabled industries to achieve greater levels of productivity, precision, and consistency, while minimizing human error and intervention. Despite the numerous advantages of automation, there are still many areas where human intervention is required. However, by using the appropriate combination of human skills and automation technologies, it is possible to achieve better performance in terms of quality, efficiency, and safety. The integration of automation systems into industries has transformed the way in which they operate, making them more competitive and adaptive to changing demands. This study aims to further investigate the role of automation in industrial processes, with a specific focus on the use of PLCs in the mobile phone manufacturing industry. The objective of this research is to develop an automated system that can effectively test mobile phones for functionality and condition. This research will explore the capabilities of PLCs in this industry and evaluate their effectiveness in achieving the desired goals. Ultimately, this study aims to contribute to the ongoing development of automation systems and their integration into various industries, including mobile phone manufacturing, to further improve their performance and efficiency.

## PC and PLC based control system

According to Ogawa and Henmi's 2006 paper, manufacturers are constantly seeking out efficient production methods in order to meet consumer demand. To accomplish this, plant operations are utilizing highly adaptable control systems such as PC and PLC-based control systems. However, these control systems have typically been used for small to medium-sized operations with fewer than a few hundred I/O points. In order to apply these systems to larger processes and achieve process automation, researchers have collaborated to develop new large- scale application systems. Through these initiatives, researchers have gained knowledge on how to apply PC and PLC-based control systems in practical and effective ways for large-scale processes with adequate production performance and flexible operations. As a result, these systems have become increasingly important for industries such as manufacturing, transportation, and healthcare, where efficient and reliable process automation is critical to success. The ability to utilize these control systems effectively has significant implications for businesses in terms of cost savings, increased efficiency, and improved quality. Therefore, research in this area remains important in order to stay up-to-date with the latest advancements in process automation and control systems technology.

## Computer vision in automation

In recent years, there has been a growing interest in the application of machine vision technology in industrial production lines. According to Baygin and Karakose (2017), machine vision systems can be used to perform a range of processes, including product counting, defect identification, and dimension measuring, without the need for specialized control. These systems employ cameras that allow for quick, seamless, and accurate measurements, thereby increasing production capacity and enabling the delivery of finished goods to the customer. As technology continues to advance, machine vision systems are becoming more accessible to all production facilities, thanks to the development of measuring technology that can operate at high rates. In line with the move towards intelligent production, the manufacturing sector has also adopted machine vision technology. As Yang and Chen (2020) note, machine vision technology seeks to replace human eyes with machines for various tasks and assessments. This technology has the potential to minimize error rates, increase productivity, standardize work,

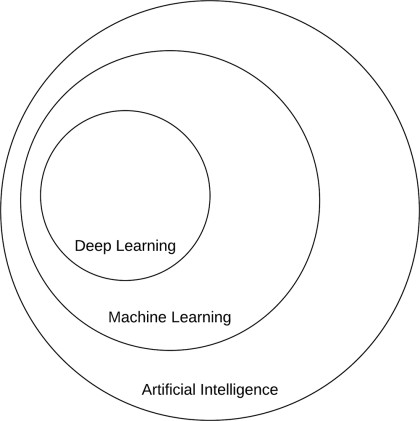
and save labor. It is therefore no surprise that machine vision has become an integral part of the sophisticated technology employed in intelligent production. As machine vision technology continues to evolve, there is a need for more research into its application in the manufacturing industry. For instance, there is a need to investigate the impact of machine vision on production efficiency, quality, and safety, as well as the costs associated with its implementation. Such research would be beneficial to manufacturers looking to integrate machine vision systems into their production lines. In addition, there is a need to examine the ethical and social implications of using machine vision technology in the workplace. For example, the widespread use of machine vision systems may lead to job losses or changes in the nature of work. It is therefore important to investigate ways to mitigate these negative effects and ensure that the benefits of machine vision technology are distributed equitably. Overall, machine vision technology has the potential to revolutionize the manufacturing industry by increasing productivity, improving quality, and reducing costs. However, there is a need for more research into its application in the workplace, including its impact on efficiency, safety, and the workforce. With further study, machine vision systems can be successfully integrated into industrial production lines, leading to greater efficiency and competitiveness.

In automation PLC is the most common controller. To build a multi-functional automatic inspection system and serve as a model for the construction of related intelligent production lines, combining the machine vision inspection system with industrial cameras as the core and the industrial automation control system with PLC as the core. The current research focuses on logos and mobile device screen detection. Detection of logos on a mobile device screen and sharing that data with PLC with help of communication protocol to control the PLC outputs is a challenge undertaken in the study, improving the concept presented in [4].

## Deep learning in computer vision

Deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but nonlinear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. The key aspect of deep learning is that these layers are not designed by human engineers: they are learned from data using a general-purpose learning procedure. Deep

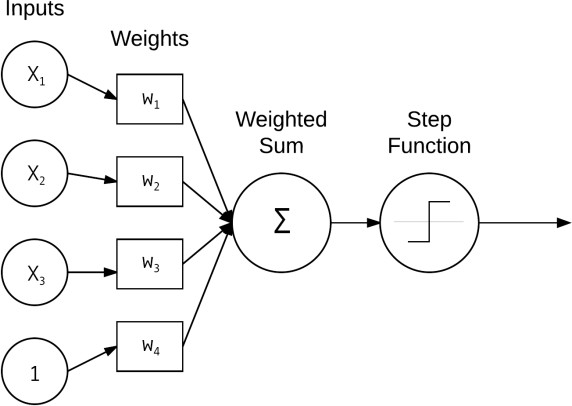
learning is a subfield of machine learning, which is, in turn, a subfield of artificial intelligence (AI). For a graphical depiction of this relationship as presented in Figure 2.1. The central goal of AI is to provide a set of algorithms and techniques that can be used to solve problems that humans perform intuitively and near automatically, but are otherwise very challenging for computers. A great example of such a class of AI problems is interpreting and understanding the contents of an image – this task is something that a human can do with little-to-no effort, but it has proven to be extremely difficult for machines to accomplish. While AI embodies a large, diverse set of work related to automatic machine reasoning (inference, planning, heuristics, etc.), the machine learning subfield tends to be specifically interested in pattern recognition and learning from data. Artificial Neural Networks (ANNs) are a class of machine learning algorithms that learn from data and specialize in pattern recognition, inspired by the structure and function of the brain. As we’ll find out, deep learning belongs to the family of ANN algorithms, and in most cases, the two terms can be used interchangeably. In fact, you may be surprised to learn that the deep learning field has been around for over 60 years, going by different names and incarnations based on research trends, available hardware and datasets, and popular options of prominent researchers at the time.



## Figure 2.1 Veen diagram of deep learning

* 1. **Neutral network and deep learning**

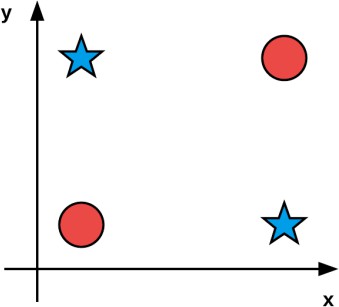
The history of neural networks and deep learning is a long, somewhat confusing one. It may surprise you to know that “deep learning” has existed since the 1940s undergoing various name changes, including cybernetics, connectionism, and the most familiar, Artificial Neural Networks (ANNs). While inspired by the human brain and how its neurons interact with each other, ANNs are not meant to be realistic models of the brain. Instead, they are an inspiration, allowing us to draw parallels between a very basic model of the brain and how we can mimic some of this behavior through artificial neural networks. We’ll discuss ANNs and the relation to the brain in Chapter 10. The first neural network model came from McCulloch and Pitts in 1943. This network was a binary classifier, capable of recognizing two different categories based on some input. The problem was that the weights used to determine the class label for a given input needed to be manually tuned by a human – this type of model clearly does not scale well if a human operator is required to intervene. Then, in the 1950s the seminal Perceptron algorithm was published by Rosenblatt this model could automatically learn the weights required to classify an input (no human intervention required) An example of the Perceptron architecture represented in Figure 2.2. In fact, this automatic training procedure formed the basis of Stochastic Gradient Descent (SGD) which is still used to train very deep neural networks today. During this time period, Perceptron-based techniques were all the rage in the neural network community. However, a 1969 publication by Minsky and Papert effectively stagnated neural network research for nearly a decade. Their work demonstrated that a Perceptron with a linear activation function (regardless of depth) was merely a linear classifier, unable to solve nonlinear problems. The canonical example of a nonlinear problem is the XOR dataset in Figure 2.3.



## Figure 2.2 Perceptron network architecture

Luckily, the backpropagation algorithm and the research by Werbos (1974) Rumelhart (1986) and LeCun (1998) were able to resuscitate neural networks from what could have been an early demise. Their research in the backpropagation algorithm enabled multi-layer feedforward neural networks to be trained (Figure 2.4). Combined with nonlinear activation functions, researchers could now learn nonlinear functions and solve the XOR problem, opening the gates to an entirely new area of research in neural networks. Further research demonstrated that neural networks are universal approximators capable of approximating any continuous function (but placing no guarantee on whether or not the network can actually learn the parameters required to represent a function). The backpropagation algorithm is the cornerstone of modern day neural networks allowing us to efficiently train neural networks and “teach” them to learn from their mistakes. But even so, at this time, due to (1) slow computers (compared to modern day machines) and (2) lack of large, labeled training sets, researchers were unable to (reliably) train neural networks that had more than two hidden layers – it was simply computationally infeasible. Today, the latest incarnation of neural networks as we know it is called deep learning. What sets deep learning apart from its previous incarnations is that we have faster, specialized hardware with more available training data. We can now train networks with many

more hidden layers that are capable of hierarchical learning where simple concepts are learned in the lower layers and more abstract patterns in the higher layers of the network



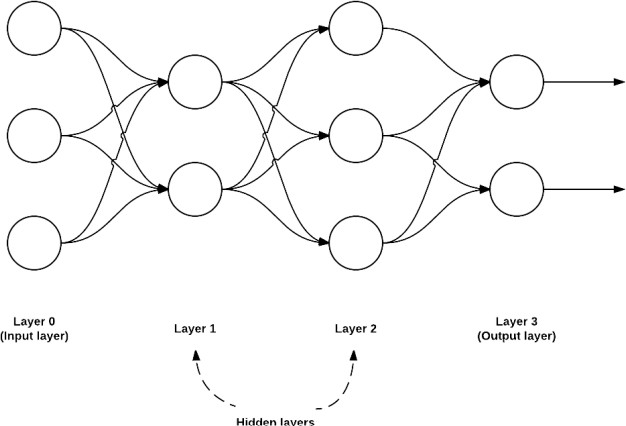
## Figure 2.3 XOR dataset

In many applications, CNNs are now considered the most powerful image classifier and are currently responsible for pushing the state-of-the-art forward in computer vision subfields that lever age machine learning. For a more thorough review of the history of neural networks and deep learning, please refer to Goodfellow et al. as well as this excellent blog post by Jason Brownlee at Machine Learning Mastery.

## Hierarchical feature learning

Machine learning algorithms (generally) fall into three camps – supervised, unsupervised, and semi-supervised learning. We’ll discuss supervised and unsupervised learning in this chapter while saving semi-supervised learning for a future discussion. In the supervised case, a machine learning algorithm is given both a set of inputs and target outputs. The algorithm then tries to learn patterns that can be used to automatically map input data points to their correct target output. Supervised learning is similar to having a teacher watching you take a test. Given your previous knowledge, you do your best to mark the correct answer on your exam; however, if you are incorrect, your teacher guides you toward a better, more educated guess the next time. In an unsupervised case, machine learning algorithms try to automatically discover discriminating features without any hints as to what the inputs are. In this scenario, our student

tries to group similar questions and answers together, even though the student does not know what the correct answer is and the teacher is not there to provide them with the true answer.



## Figure 2.4 Multi layer feedforward network architecture

In the context of machine learning applied to image classification, the goal of a machine learning algorithm is to take these sets of images and identify patterns that can be used to discriminate various image classes/objects from one another. In the past, we used hand-engineered features to quantify the contents of an image – we rarely used raw pixel intensities as inputs to our machine learning models, as is now common with deep learning. For each image in our dataset, we performed feature extraction, or the process of taking an input image, quantifying it according to some algorithm (called a feature extractor or image descriptor), and returning a vector (i.e., a list of numbers) that aimed to quantify the contents of an image. Figure 2.5 below depicts the process of quantifying an image containing prescription pill medication via a series of blackbox color, texture, and shape image descriptors.

Other methods such as Histogram of Oriented Gradients (HOG) proved to be very good at detecting objects in images when the viewpoint angle of our image did not vary dramatically from what our classifier was trained on. An example of using the HOG + Linear SVM detector method can be seen in Figure 2.6 where we detect the presence of stop signs in images. For a

while, research in object detection in images was guided by HOG and its variants, including computationally expensive methods such as the Deformable Parts Model and Exemplar SVMs.

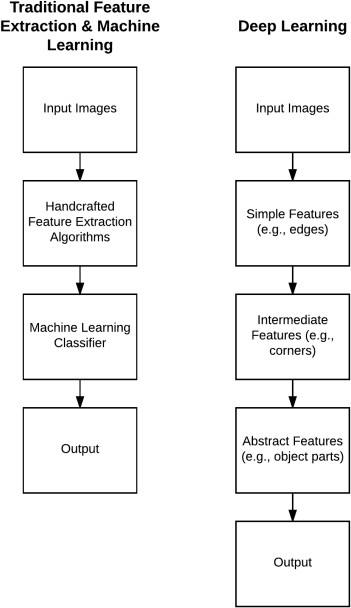
In each of these situations, an algorithm was hand-defined to quantify and encode a particular aspect of an image (i.e., shape, texture, color, etc.). Given an input image of pixels, we would apply our hand-defined algorithm to the pixels, and in return receive a feature vector quantifying the image contents – the image pixels themselves did not serve a purpose other than being inputs to our feature extraction process. The feature vectors that resulted from feature extraction were what we were truly interested in as they served as inputs to our machine learning models. Deep learning, and specifically Convolutional Neural Networks, take a different approach. Instead of hand-defining a set of rules and algorithms to extract features from an image, these features are instead automatically learned from the training process. Again, let’s return to the goal of machine learning: computers should be able to learn from experience (i.e., examples) of the problem they are trying to solve. Using deep learning, we try to understand the problem in terms of a hierarchy of concepts. Each concept builds on top of the others. Concepts in the lower level layers of the network encode some basic representation of the problem, whereas higher level layers use these basic layers to form more abstract concepts. This hierarchical learning allows us to completely remove the hand-designed feature extraction process and treat CNNs as end-to-end learners. Given an image, we supply the pixel intensity values as inputs to the CNN. A series of hidden layers are used to extract features from our input image. These hidden layers build upon each other in a hierarchal fashion. At first, only edge-like regions are detected in the lower level layers of the network. These edge regions are used to define corners (where edges intersect) and contours (outlines of objects). Combining corners and contours can lead to abstract “object parts” in the next layer. Again, keep in mind that the types of concepts these filters are learning to detect are automatically learned – there is no intervention by us in the learning process. Finally, output layer is used to classify the image and obtain the output class label – the output layer is either directly or indirectly influenced by every other node in the network. We can view this process as hierarchical learning: each layer in the network uses the output of previous layers as “building blocks” to construct increasingly more abstract concepts. These layers are learned automatically – there is no hand-crafted feature engineering taking place in our network. Figure

2.5 compares classic image classification algorithms using hand-crafted features to representation learning via deep learning and Convolutional Neural Networks. One of the

primary benefits of deep learning and Convolutional Neural Networks is that it allows us to skip the feature extraction step and instead focus on process of training our network to learn these filters. However, as we’ll find out later in this book, training a network to obtain reasonable accuracy on a given image dataset isn’t always an easy task.

## Depth of deep learning

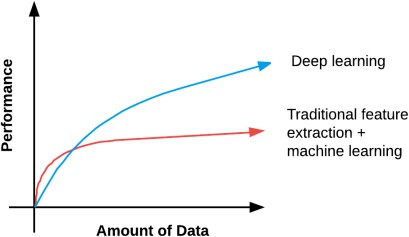
To quote Jeff Dean from his 2016 talk, Deep Learning for Building Intelligent Computer Systems “When you hear the term deep learning, just think of a large, deep neural net. Deep refers to the number of layers typically and so this kind of the popular term that’s been adopted in the press.” This is an excellent quote as it allows us to conceptualize deep learning as large neural networks where layers build on top of each other, gradually increasing in depth. The problem is we still don’t have a concrete answer to the question, “How many layers does a neural network need to be considered deep?. The short answer is there is no consensus amongst experts on the depth of a network to be considered deep.



## Figure 2.5 Classic image classification

And now we need to look at the question of network type. By definition, a Convolutional Neural Network (CNN) is a type of deep learning algorithm. But suppose we had a CNN with only one convolutional layer – is a network that is shallow, but yet still belongs to a family of algorithms inside the deep learning camp considered to be “deep”? My personal opinion is that any network with greater than two hidden layers can be considered “deep”. My reasoning is based on previous research in ANNs that were heavily handicapped by: 1. Our lack of large, labeled datasets available for training 2. Our computers being too slow to train large neural networks 3. Inadequate activation functions Because of these problems, we could not easily train networks with more than two hidden layers during the 1980s and 1990s (and prior, of course). In fact, Geoff Hinton supports this sentiment in his 2016 talk, Deep Learning where he discussed why the previous incarnations of deep learning (ANNs) did not take off during the 1990s phase: 1. Our labeled datasets were thousands of times too small. 2. Our computers were millions of times too slow. 3. We initialized the network weights in a stupid way. 4. We used the wrong type of nonlinearity activation function. All of these reasons point to the fact that training networks with a depth larger than two hidden layers were a futile, if not a computational, impossibility. In the current incarnation we can see that the tides have changed. We now have: 1. Faster computers 2. Highly optimized hardware (i.e., GPUs) 3. Large, labeled datasets in the order of millions of images 4. A better understanding of weight initialization functions and what does/does not work

5. Superior activation functions and an understanding regarding why previous nonlinearity functions stagnated research Paraphrasing Andrew Ng from his 2013 talk, Deep Learning, Self- Taught Learning and Unsupervised Feature Learning are now able to construct deeper neural networks and train them with more data. As the depth of the network increases, so does the classification accuracy. This behavior is different from traditional machine learning algorithms (i.e., logistic regression, SVMs, decision trees, etc.) where we reach a plateau in performance even as available training data increases. A plot inspired by Andrew Ng’s 2015 talk, What data scientists should know about deep learning can be seen in Figure 2.6, providing an example of this behavior.



## Figure 2.6 Deep learning and Traditional learning Performance

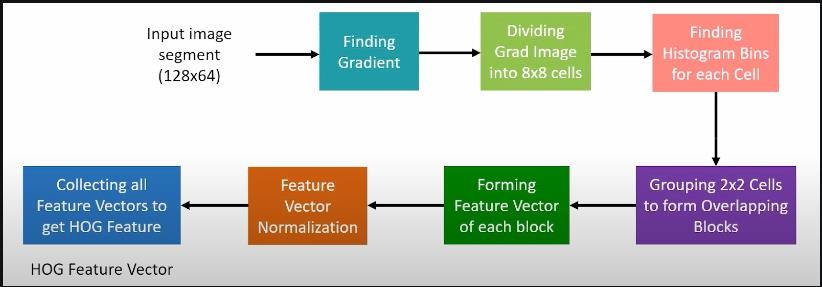
As the amount of training data increases, our neural network algorithms obtain higher classification accuracy, whereas previous methods plateau at a certain point. Because of the relationship between higher accuracy and more data, we tend to associate deep learning with large datasets as well. When working on your own deep learning applications suggestusing the following rule of thumb to determine if your given neural network is deep: 1. Are you using a specialized network architecture such as Convolutional Neural Networks, Recurrent Neural Networks, or Long Short-Term Memory (LSTM) networks? If so, yes, you are performing deep learning. 2. Does your network have a depth > 2? If yes, you are doing deep learning. 3. Does your network have a depth > 10? If so, you are performing very deep learning . All that said, try not to get caught up in the buzzwords surrounding deep learning and what is/is not deep learning. At the very core, deep learning has gone through a number of different incarnations over the past 60 years based on various schools of thought – but each of these schools of thought centralize around artificial neural networks inspired by the structure and function of the brain. Regardless of network depth, width, or specialized network architecture, you’re still performing machine learning using artificial neural networks.

## Object detection algorithms

Since the popularization of deep learning in the early 2010s, there’s been a continuous progression and improvement in the quality of algorithms used to solve object detection. We’re going to explore the most popular algorithms while understanding their working theory, benefits, and their flaws in certain scenarios

## Histogram of Oriented Gradients (HOG)

The is one of the oldest methods of object detection. It was first introduced in 1986. Despite some developments in the upcoming decade, the approach did not gain a lot of popularity until 2005 when it started being used in many tasks related to computer vision. HOG uses a feature extractor to identify objects in an image. The feature descriptor used in HOG is a representation of a part of an image where we extract only the most necessary information while disregarding anything else. The function of the feature descriptor is to convert the overall size of the image into the form of an array or feature vector. In HOG, we use the gradient orientation procedure to localize the most critical parts of an image.



## Figure 2.7 HOG Object detection algorithm

Before we understand the overall architecture of HOG, here’s how it works. For a particular pixel in an image, the histogram of the gradient is calculated by considering the vertical and horizontal values to obtain the feature vectors. With the help of the gradient magnitude and the gradient angles, we can get a clear value for the current pixel by exploring the other entities in their horizontal and vertical surrounding. As shown in figure 2.7 representation, we’ll consider an image segment of a particular size. The first step is to find the gradient by dividing the entire computation of the image into gradient representations of 8×8 cells. With the help of the 64 gradient vectors that are achieved, we can split each cell into angular bins and compute the histogram for the particular area. This process reduces the size of 64 vectors to a smaller size of 9 values. Once we obtain the size of 9 point histogram values (bins) for each cell, we can choose to create overlaps for the blocks of cells. The final steps are to form the feature blocks, normalize the obtained feature vectors, and collect all the features vectors to get an overall HOG feature.

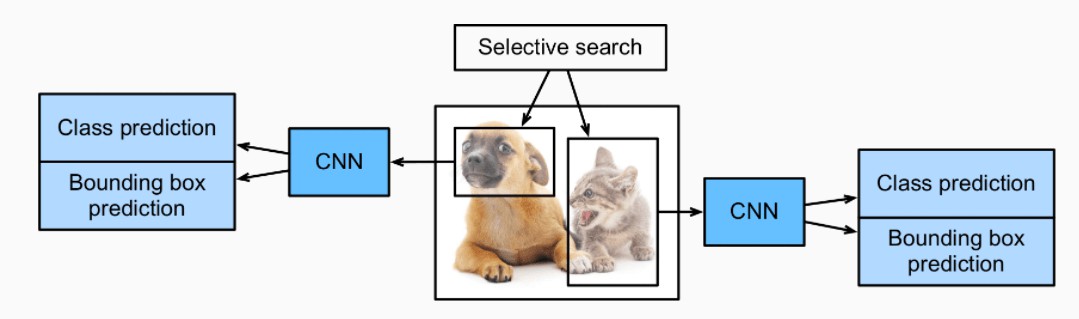
Achievements of HOG

1. Creation of a feature descriptor useful for performing object detection.
2. Ability to be combined with support vector machines (SVMs) to achieve high-accuracy object detection.
3. Creation of a sliding window effect for the computation of each position.

While the Histogram of Oriented Gradients (HOG) was quite revolutionary in the beginning stages of object detection, there were a lot of issues in this method. It’s quite time-consuming for complex pixel computation in images, and ineffective in certain object detection scenarios with tighter spaces. When to use HOG? HOG should often be used as the first method of object detection to test other algorithms and their respective performance. Regardless, HOG finds significant use in most object detection and facial landmark recognition with decent accuracy. Example use cases – One of the popular use cases of HOG is in pedestrian detection due to its smooth edges. Other general applications include object detection of specific objects.

## Region based convolutional neural networks (R-CNN)

The region based neural networks are an improvement in the object detection procedure from the previous methods of HOG and SIFT. In the R-CNN models, we try to extract the most essential features (usually around 2000 features) by making use of selective features. The process of selecting the most significant extractions can be computed with the help of a selective search algorithm that can achieve these more important regional proposals.



## Figure 2.8 R-CNN – Object Detection Algorithm ([Girshick](https://d2l.ai/chapter_references/zreferences.html" \l "id85) *[et al.](https://d2l.ai/chapter_references/zreferences.html" \l "id85)*[, 2014](https://d2l.ai/chapter_references/zreferences.html" \l "id85))

The working procedure of the selective search algorithm shown in figure 2.8 to select the most important regional proposals is to ensure that you generate multiple sub-segmentations on a particular image and select the candidate entries for your task. The greedy algorithm can then be made use of to combine the effective entries accordingly for a recurring process to combine the smaller segments into suitable larger segments. Once the selective search algorithm is successfully completed, our next tasks are to extract the features and make the appropriate predictions. We can then make the final candidate proposals, and the convolutional neural networks can be used for creating an n-dimensional (either 2048 or 4096) feature vector as output. With the help of a pre-trained convolutional neural network, we can achieve the task of feature extraction with ease. The final step of the R-CNN is to make the appropriate predictions for the image and label the respective bounding box accordingly. In order to obtain the best results for each task, the predictions are made by the computation of a classification model for each task, while a regression model is used to correct the bounding box classification for the proposed regions.

Issues with R-CNN

1. Despite producing effective results for feature extraction with the pre-trained CNN models, the overall procedure of extraction of all the region proposals, and ultimately the best regions with the current algorithms, is extremely slow.
2. Another major drawback of the R-CNN model is not only the slow rate of training but also the high prediction time. The solution requires the use of large computational resources, increasing the overall feasibility of the process. Hence, the overall architecture can be considered quite expensive.
3. Sometimes, bad candidate selections can occur at the initial step due to the lack of improvements that can be made in this particular step. A lot of problems in the trained model could be caused by this.

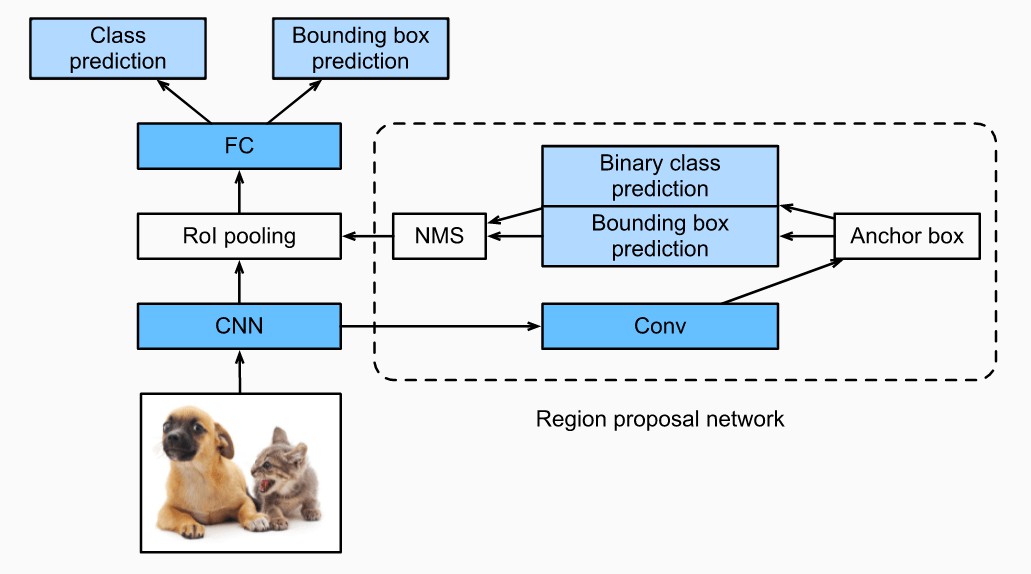
When To Use R-CNN? – R-CNN similar to the HOG object detection method must be used as a first baseline for testing the performance of the object detection models. The time taken for predictions of images and objects can take a bit longer than anticipated, so usually the more modern versions of R-CNN are preferred.

Example use cases – There are several applications of R-CNN for solving different types of tasks related to object detection. For example, tracking objects from a drone-mounted camera, locating text in an image, and enabling object detection in Google Lens

## Faster R-CNN

While the R-CNN model was able to perform the computation of object detection and achieve desirable results, there were some major lacklustre elements, especially the speed of the model. So, faster methods for tackling some of these issues had to be introduced to overcome the problems that existed in R-CNN. Firstly, the Fast R-CNN was introduced to combat some of the pre-existing issues of R-CNN.

In the fast R-CNN method, the entire image is passed through the pre-trained Convolutional Neural Network instead of considering all the sub-segments. The region of interest (RoI) pooling is a special method that takes two inputs of the pre-trained model and selective search algorithm to provide a fully connected layer with an output. In this section, we will learn more about the Faster R-CNN network, which is an improvement on the fast R-CNN model.



## Figure 2.9 Faster R-CNN – Object Detection Algorithm ([Girshick](https://d2l.ai/chapter_references/zreferences.html" \l "id85) *[et al.](https://d2l.ai/chapter_references/zreferences.html" \l "id85)*[, 2014](https://d2l.ai/chapter_references/zreferences.html" \l "id85))

The Faster R-CNN model is one of the best versions of the R-CNN family and improves the speed of performance tremendously from its predecessors. While the R-CNN and Fast R-CNN model make use of a selective search algorithm to compute the region proposals, the Faster R- CNN method replaces this existing method with a superior region proposal network. The region proposal network (RPN) computes images from a wide range and different scales to produce effective outputs. The architecture of Faster R CNN is shown in Figure 2.8 and according to it,the regional proposal network reduces the margin computation time, usually 10 ms per image. This network consists of the convolutional layer from which we can obtain the essential feature maps of each pixel. For each feature map, we have multiple anchor boxes which have varying scales, different sizes, and aspect ratios. For each anchor box, we make a prediction of the binary class and generate a bounding box for the same. The following information is then passed through the non-maximum suppression to remove any unnecessary data since many overlaps are produced while creating the feature maps. The output from the non-maximum suppression is passed through the region of interest, and the rest of the process and computation is similar to the working of Fast R-CNN. One of the main limitations of the Faster R-CNN method is the amount of time delay in the proposition of different objects. Sometimes, the speed depends on the type of system being used. When To Use Faster R-CNN? The time for prediction is faster compared to other CNN methods. While R-CNN usually takes around 40-50 seconds for the prediction of objects in an image, the Fast R-CNN takes around 2 seconds, but the Faster R-CNN returns the optimal result in just about 0.2 seconds.

Example use cases – The examples of use cases for Faster R-CNN are similar to the ones described in the R-CNN methodology. However, with Faster R-CNN, we can perform these tasks optimally and achieve results more effectively.

## Single Shot Detector (SSD)

The single shot detector for multi-box predictions is one of the fastest ways to achieve the real- time computation of object detection tasks. While the Faster R-CNN methodologies can achieve high accuracies of prediction, the overall process is quite time-consuming and it requires the real-time task to run at about 7 frames per second, which is far from desirable. The single-shot detector (SSD) solves this issue by improving the frames per second to almost five times more

than the Faster R-CNN model. It removes the use of the region proposal network and instead makes use of multi-scale features and default boxes.



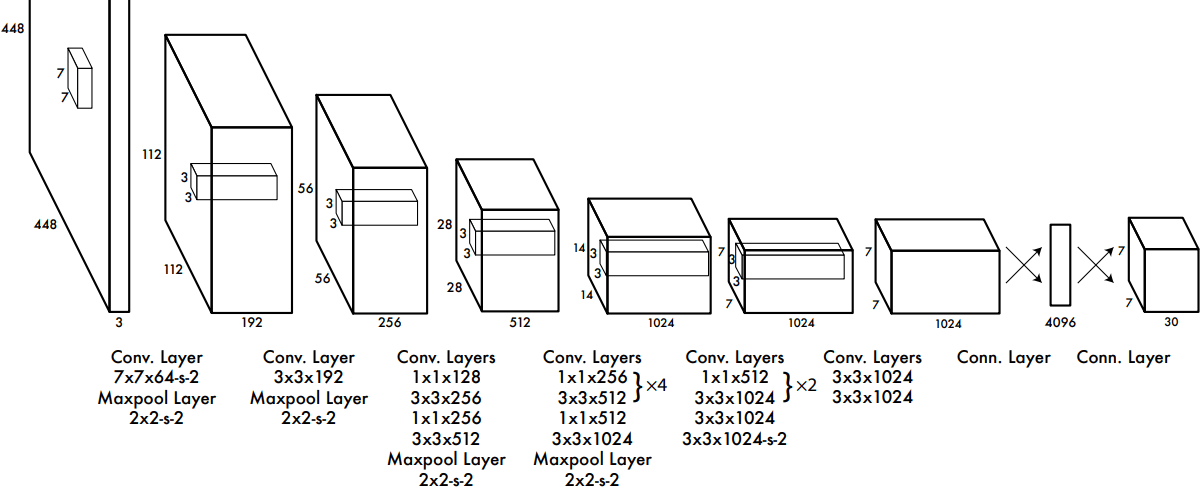
## Figure 2.10 SSD – Object Detection Algorithm (Wei Liu 29dec 2016)

The single-shot multibox detector architecture can be broken down into mainly three components. The first stage of the single-shot detector is the feature extraction step, where all the crucial feature maps are selected. This architectural region consists of only fully convolutional layers and no other layers. After extracting all the essential feature maps, the next step is the process of detecting heads. This step also consists of fully convolutional neural networks as per architecture shown in figure 2.9. However, in the second stage of detection heads, the task is not to find the semantic meaning for the images. Instead, the primary goal is to create the most appropriate bounding maps for all the feature maps. Once we have computed the two essential stages, the final stage is to pass it through the non-maximum suppression layers for reducing the error rate caused by repeated bounding boxes. The SSD, while boosting the performance significantly, suffers from decreasing the resolution of the images to a lower quality. The SSD architecture will typically perform worse than the Faster R-CNN for small-scale objects. When To Use SSD? – The single-shot detector is often the preferred method. The main reason for using the single-shot detector is because we mainly prefer faster predictions on an image for detecting larger objects, where accuracy is not an extremely important concern. However, for more accurate predictions for smaller and precise objects, other methods must be considered.

Example use cases – The Single-shot detector can be trained and experimented on a multitude of datasets, such as PASCAL VOC, COCO, and ILSVRC datasets. They can perform well on larger object detections like the detection of humans, tables, chairs, and other similar entities.

## YOLO (You Only Look Once)

You only look once (YOLO) is one of the most popular model architectures and algorithms for object detection. Usually, the first concept found on a Google search for algorithms on object detection is the YOLO architecture. There are several versions of YOLO, which we will discuss in the upcoming sections. The YOLO model uses one of the best neural network archetypes to produce high accuracy and overall speed of processing. This speed and accuracy is the main reason for its popularity.



## Figure 2.11 YOLO architecture (Real-time object detection,9 May 2016)

The YOLO architecture utilizes three primary terminologies to achieve its goal of object detection. Understanding these three techniques is quite significant to know why exactly this model performs so quickly and accurately in comparison to other object detection algorithms. The first concept in the YOLO model is residual blocks. In the first architectural design, they have used 7×7 residual blocks to create grids in the particular image referring to figure 2.10 each of these grids acts as central points and a particular prediction for each of these grids is made accordingly. In the second technique, each of the central points for a particular prediction is considered for the creation of the bounding boxes. While the classification tasks work well for each grid, it’s more complex to segregate the bounding boxes for each of the predictions that are made. The third and final technique is the use of the intersection of union (IOU) to calculate the best bounding boxes for the particular object detection task. Advantages of YOLO are contents The computation and processing speed of YOLO is quite high, especially in real-time compared

to most of the other training methods and object detection algorithms. Apart from the fast computing speed, the YOLO algorithm also manages to provide an overall high accuracy with the reduction of background errors seen in other methods. The architecture of YOLO allows the model to learn and develop an understanding of numerous objects more efficiently.

The limitations of YOLO are such as Failure to detect smaller objects in an image or video because of the lower recall rate and it cant’t detect two objects that are extremely close to each other due to the limitations of bounding boxes.

The YOLO architecture is one of the most influential and successful object detection algorithms. With the introduction of the YOLO architecture in 2016, their consecutive versions YOLO v2 and YOLO v3 arrived in 2017 and 2018. While there was no new release in 2019, 2020 saw three quick releases: YOLO v4, YOLO v5, and PP-YOLO. Each of the newer versions of YOLO slightly improved on their previous ones. The tiny YOLO was also released to ensure that object detection could be supported on embedded devices. When To Use YOLO? – While all the previously discussed methods perform quite well on images and sometimes video analysis for object detection, the YOLO architecture is one of the most preferred methods for performing object detection in real-time. It achieves high accuracy on most real-time processing tasks with a decent speed and frames per second depending on the device that you’re running the program on.

Example use cases – Some popular use cases of the YOLO architecture apart from object detection on numerous objects include vehicle detection, animal detection, and person detection.

## RetinaNet

The RetinaNet model introduced in 2017 became one of the best models with single-shot object detection capabilities that could surpass other popular object detection algorithms during this time. When the RetinaNet Architecture was released, the object detection capabilities exceeded that of the Yolo v2 and the SSD models. While maintaining the same speed as these models, it was also able to compete with the R-CNN family in terms of accuracy. Due to these reasons, the RetinaNet model finds a high usage in detecting objects through satellite imagery.



## Figure 2.12 RetinaNet – Object Detection Algorithm (Priya Goyal, 7 Feb 2017)

The RetinaNet architecture represented in figure 2.11 is built in such a way that the previous issues of single-shot detectors are somewhat balanced out to produce more effective and efficient results. In this model architecture, the cross-entropy loss in the previous models is replaced with the focal loss. The focal loss handles the class imbalance problem that exist in architectures like YOLO and SSD. The RetinaNet model is a combination of three main entities. RetinaNet is built using three factors, namely the ResNet model (specifically ResNet-101), the feature pyramid network (FPN), and the focal loss. The feature pyramid network is one of the best methods for overcoming a majority of the shortcomings of the previous architecture. It helps in combining the semantic rich features of lower resolution images with that of the semantically weak features of the higher resolution images. In the final output, we can create both the classification and regression models similar to the other object detection methods discussed previously. The classification network is used for appropriate multi-class predictions, while the regression network is built to predict the appropriate bounding boxes for the classified entities. When to use RetinaNet? – RetinaNet is currently one of the best methods for object detection in a number of different tasks. It can be used as a replacement for a single-shot detector for a multitude of tasks to achieve quick and accurate results for images.

Example use cases – There’s a wide array of applications that can be performed with the RetinaNet object detection algorithm. A high-level application of RetinaNet is used for object detection in aerial and satellite imagery.

## Summary

This chapter addressed the complicated question of “What is deep learning?”. As we found out, deep learning has been around since the 1940s, going by different names and incarnations based on various schools of thought and popular research trends at a given time. At the very core, deep learning belongs to the family of Artificial Neural Networks (ANNs), a set of algorithms that learn patterns inspired by the structure and function of the brain. There is no consensus amongst experts on exactly what makes a neural network “deep”; however, we know that: 1. Deep learning algorithms learn in a hierarchical fashion and therefore stack multiple layers on top of each other to learn increasingly more abstract concepts. 2. A network should have > 2 layers to be considered “deep” (this is my anecdotal opinion based on decades of neural network research). 3. A network with > 10 layers is considered very deep (although this number will change as architectures such as ResNet have been successfully trained with over 100 layers). Object detection is still one of the most essential deep learning and computer vision applications to date. We’ve seen a lot of improvements and advancements in the methodologies of object detection. It started with algorithms like the Histogram of Oriented Gradients, introduced way back in 1986 to perform simple object detections on images with decent accuracy. Now, we have modern architectures such as Faster R-CNN, Mask R-CNN, YOLO, and RetinaNet. The restrictions for object detection are not limited to images, as they can be effectively performed on videos and real-time footage with high accuracy. In the future, a lot more successful algorithms and libraries for object detection still await us.

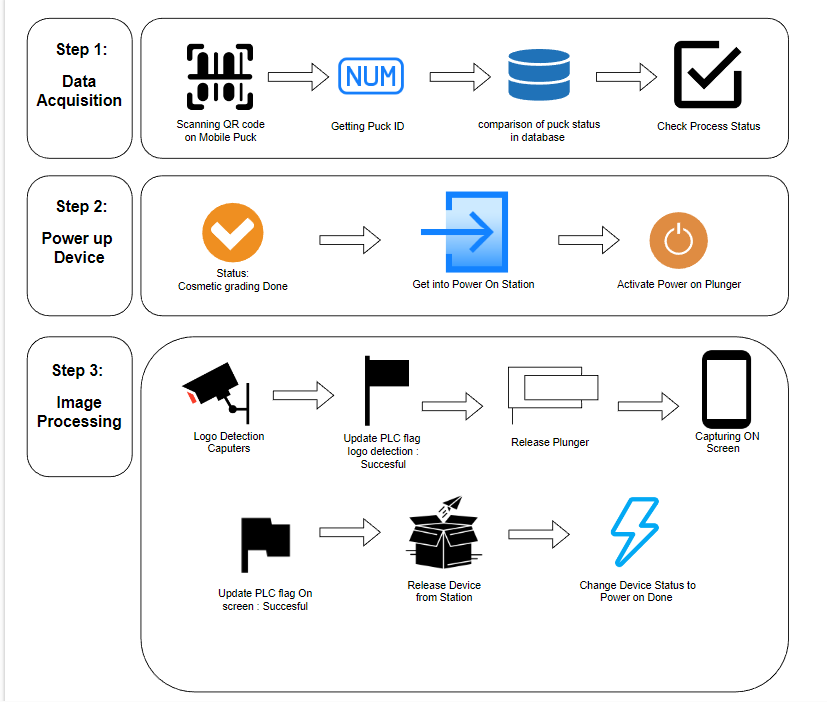
**Chapter 3**

# Methodology

The overview diagram of mobile logo detection and screen-on detection is illustrated in Figure

* 1. The first part is Data acquisition which is used to check the process status of the device. The QR code on the puck is scanned and the IMEI (International Mobile Equipment Identity) number of the device is extracted. Thereafter, the previous process status of that IMEI number device is compared to the current process status from the database. The second part is powering up the device, devices having process status as cosmetic grading completed are passed into the power on station. The pneumatic plunger is activated, and it is pushed forward for pressing the power on button of the device through PLC (Programmable Logic Controller). The third part is video processing, where logo detection on the mobile screenof the device is captured by the CSI (Camera Serial Interface) camera present in the station.

As the logo is detected on a mobile screen, the PLC flag assigned to a particular shared memory block is updated and the device power-on button is released by pushing the pneumatic plunger in the reverse direction. Thereafter, the CSI camera captures the mobile screen on the state as the background of the mobile screen changes after turning it on, and another PLC flag assigned for this state is updated via shared memory. The device is released from the station and the new process status is then updated in the database. Finally, the process status of the device is updated from cosmetic grading completion to mobile power on successful.



## Figure 3.1: Methodology Diagram

* 1. **Data Acquisition**

A QR (Quick Response) code reader is a device or application that scans and decodes QR code. QR codes are two dimensional barcodes that are designed to be read by smartphones and other mobile devices. They were first developed in Japan in the 1990s and have since become widely used in marketing, advertising, and other industries. A QR code reader can be used to scan QR codes on printed materials such as posters, billboards, and product packaging. When a QR code is scanned, the reader uses its camera to capture an image of the code, and then decodes the information contained within the code. The decoded information may be a URL, text, or other data. QR codes have become increasingly popular They can be used to provide additional information about a product, offer discounts or promotions, or direct users to a website or social media page. One of the most common applications of QR code readers is in mobile payment systems. Many payment systems, such as Alipay and WeChat Pay in China, use QR codes to facilitate transactions. Users can simply scan a QR code with their smartphone to initiate a payment, rather than having to enter their payment information manually. QR code readers can also be used for inventory management and tracking. Companies can use QR codes to label their products and track their movement throughout the supply chain. QR code readers can be used to quickly and easily scan these codes, making it easier to keep track of inventory and shipments. Another application of QR code readers is in ticketing and access control systems. QR codes can be used to store information about a ticket or pass, such as the event name, date, and seat number. A QR to scan the code and verify that ticket is valid. Some QR code readers also have additional features, such as the ability to create and share QR codes. Users can create their own QR codes containing information such as contact details or a website URL, and then share these codes with others. QR code readers are available as standalone devices, or as applications that can be downloaded onto a smartphone or tablet. Many smartphones now come with a built-in QR code reader, making it easy for users to scan codes without having to download a separate app. QR code readers have become an important tool in a wide range of industries. They offer a quick and convenient way to store and share information and have the potential to revolutionize the way we get information for goods and services, manage inventory, and access events and attractions.

According to methodology diagram figure 3.1 very first step is scanning which is done with Hikvision Smart Code Reader. It is an intelligent industrial code reader designed to decode a wide range of 1D and 2D codes on various products in manufacturing, logistics, and retail industries. It uses advanced algorithms and machine vision technology to identify codes with high accuracy and speed, improving efficiency and reducing errors in production and distribution processes. The Smart Code Reader features a compact and robust design, making it easy to integrate into existing production lines and automation systems. It supports a range of communication protocols, including Ethernet, RS-232, and RS-485, allowing it to be connected to different devices and networks. One of the key features of the Smart Code Reader is its ability to recognize multiple codes at the same time, even when they are printed closely together or overlapping. This saves time and reduces the need for manual intervention, increasing throughput and productivity. The Smart Code Reader can also read codes from various angles and distances, making it suitable for use in different environments and applications. It can be used to read codes on products moving on a conveyor belt, as well as those in static positions, such as labels or tags.



## Figure 3.2 Smart code reader

Additionally, the Smart Code Reader which is shown in figure 3.2 comes with Hikvision's user- friendly software, which allows for easy setup and configuration, as well as real-time monitoring and data analysis. The software also provides users with the ability to customize the reader's

settings and parameters, such as code type, lighting, and exposure time, to optimize performance for specific applications.

It also has GigE(Gigabit Ethernet) interface Gigabit Ethernet (GigE) has become a popular interface for digital cameras in the professional imaging sector due to its high data transfer rate, which can support high-resolution images and fast frame rates. It is a general purpose interface utilized in industrial inspection, scientific research, medical imaging, and security surveillance.Compared to traditional analog camera systems, GigE offers several advantages, including improved image quality, faster data transfer rates, and greater flexibility in terms of camera placement and connectivity. With GigE, cameras are connected directly to a computer or network switch using Ethernet cables without using specialized hardware or software.

Furthermore, GigE supports power over ethernet (PoE), which powers camera directly from the network cable, simplifying installation and reducing the need for additional power cables. GigE is a versatile and cost-effective interface that has the potential to replace existing analog camera systems in many applications.

## Advantages of the GigE interface

High data rates: GigE (Gigabit Ethernet) offers data rates of up to 1 Gbps (gigabits per second), which is much faster than many other interfaces commonly used for image processing, such as USB or FireWire. Usability of existing Ethernet infrastructure: Because GigE uses the same cabling and networking equipment as standard Ethernet, it is often possible to use existing infrastructure without needing to invest in new hardware.

Cable length up to 100 m: GigE transmission of data is vary easy upto 100 m .

Easy integration into all image processing applications (libraries): Because GigE is so widely used, many image processing libraries and software packages include built in support for it, which makes it relatively easy to integrate into existing systems. High degree of standardization through Gigabit Ethernet and GigE Vision standards: GigE is standardized through both the Gigabit Ethernet and GigE Vision standards, which helps ensure compatibility and interoperability between different devices and systems.

PoE functionality: Some GigE cameras also offer Power over Ethernet (PoE) functionality, which allows them to be powered via the same Ethernet cable that is used to transmit data. This simplifys installation and reduce the need for additional power cables or outlets.

* + 1. **Communication with code reader**

When it comes to code reader settings, the various settings involved in establishing proper communication with the code reader include:

1. Device Connection: This setting allows to make connection between the code reader and the device it is connected to. This include setting such as the baud rate, data bits, parity, and stop bits.
2. Image settings: This setting allows to configure the image settings for the code reader. This include settings such as the image resolution, contrast, and brightness.
3. Algorithm Settings: This setting allows to configure the algorithm used by the code reader to decode barcodes. This include settings such as barcode type, decoding speed, and error correction level.
4. Input and Output Settings: This setting allows to configure the input and output settings for the code reader. This

include settings such as trigger mode, input and output signals, and timing settings.

1. Data Processing: This setting allows to configure the data processing settings for the code reader. This include settings such as data formatting, data filtering, and data output.
2. Communication: This setting allows to configure the communication settings for the code reader. This include settings such as the communication protocol, data transfer rate, and data transfer method.
3. Configuration Management: This setting allows to manage the configuration settings for the code reader. This include settings such as saving and loading configuration files, and restoring default settings.

These code reader settings are essential to ensure proper communication and functionality of the device. Properly configuring these settings allows for accurate and efficient barcode scanning, and greatly enhance the performance and functionality of the code reader.

* + 1. **Connection with code reader**

In the device connection settings, a list of cameras connected to the same network is typically displayed. To establish communication with a specific camera in the list, the user needs to select that camera's Internet Protocol (IP) address, along with the correct MAC address, subnet mask, and gateway. The IP address is a unique identifier assigned to each device on the network. It consists of a series of numbers separated by dots, such as 192.168.0.1. The user needs to ensure that they select the correct IP address for the camera they want to communicate with. The MAC address is a unique identifier which is assigned to the network interface card of the camera. It consists of a series of alphanumeric characters separated by colons, such as 00:11:22:33:44:55. The user needs to ensure that they enter the correct MAC address for the camera they want to communicate with. The subnet mask is used define size of the network. It is typically a set of four numbers separated by dots, such as 255.255.255.0. The user needs to ensure that they enter correct subnet mask for network to which camera is connected to. The gateway is IP address of the device on the network that connects the local network to the Internet. It is typically the IP address of the router. User needs to ensure that they enter the correct gateway address for the network the camera is connected to. By entering correct IP address, MAC address, subnet mask, gateway, the user establishes communication with the desired camera and begin configuring its settings.

* + 1. **Image configuration**

The image settings in code reader software are an important part of configuring the camera for proper capture. These settings include exposure time, gain, gamma, acquisition frame rate, and light settings. Exposure time refers to the duration of time that the camera sensor is exposed to light when capturing an image. It is typically measured in fractions of a second, such as 1/1000th of a second.

Gain refers to the amplification of the signal from the camera sensor. It is used to adjust the brightness of the image without changing the exposure time. Increasing the gain can result in a

brighter image, but also introduce noise into the image. Gamma refers to the correction of brightness values image. It is used to adjust the contrast of the image, and to compensate for differences in the lighting conditions. Acquisition frame rate refers to number of images captured by the camera per second. It is measured in frames per second (fps). Higher frame rates results in smoother video playback, but also require more processing power and storage space. Light settings refer to the configuration of the lighting conditions for proper capture. This include adjusting the brightness and color temperature of the lighting and positioning the lights to minimize shadows and glare. By properly configuring these image settings, the user can optimize the camera for capturing clear and accurate images of barcodes and other objects. This can greatly improve the performance of the code reader software and increase the accuracy of barcode decoding.

* + 1. **Algorithm selection**

Further algorithm settings in code reader software permit user to specify the type of barcode or QR code that the camera should be able to decode after capturing an image. Different types of barcodes have different characteristics, such as the size and shape of the bars or the amount of space between them. The algorithm settings is used to optimize the decoding process for a particular type of barcode. The user can typically select the type of barcode or QR code to be decoded from a dropdown menu or other selection tool in the software interface. Common barcode types include Code 39, Code 128, UPC-A, and EAN-13, among others. QR codes are also a common type of code that can be decoded by code reader software. In addition to selecting the type of barcode, the user is also be able to configure other algorithm settings, such as the minimum and maximum size of the code, the number of characters or digits in the code, and the orientation or direction of the code. By properly configuring the algorithm settings, the code reader software is optimized for the specific type of barcode or QR code being used. This improves the accuracy and speed of decoding process and help to ensure that correct data is captured and processed.

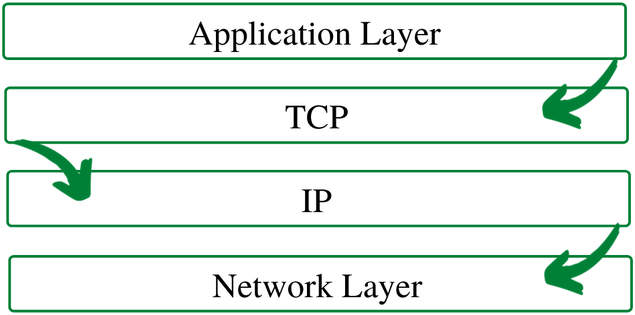
* + 1. **Input and output control**

The input/output control settings in code reader software are used to configure the trigger mode and source for capturing images with the camera. Triggering is used to start the image capture process when a barcode or QR code is detected. The trigger mode can be set to either continuous

or single shot mode. In continuous mode, the camera will continuously capture images until the trigger is released. In single shot mode, the camera will capture a single image when the trigger is activated. The trigger source specifies the source of the trigger signal. In the case of TCP start, the trigger signal is received from a TCP server. The software is configured to listen for a specific string of text, and when that text is received, the trigger is activated and the camera will capture an image. Other trigger sources may include software triggers, which are initiated through the software interface, or physical triggers such as buttons or sensors. The trigger delay can also be configured to introduce a delay between the trigger signal and the actual image capture, which is useful in certain applications. In addition to trigger settings, the input/output control settings also include options for configuring other input and output signals. For example, the software may be configured to output a signal when a barcode is successfully decoded, or to input signals from external sensors or devices. By properly configuring the input/output control settings, the user ensures that the camera is triggered and captures images at the appropriate times, based on the specific requirements of the application. The code reader software typically includes the ability to stop the trigger signal based on certain conditions, such as a timeout or a maximum number of codes read. This helps to prevent unnecessary captures and conserve system resources. The device support different methods of stopping the trigger signal, including via TCP, UDP, I/O, serial port, or USB. In this application, the trigger is being stopped via TCP, which requires the TCP stop trigger to be enabled and configured with the specified TCP trigger port and text.

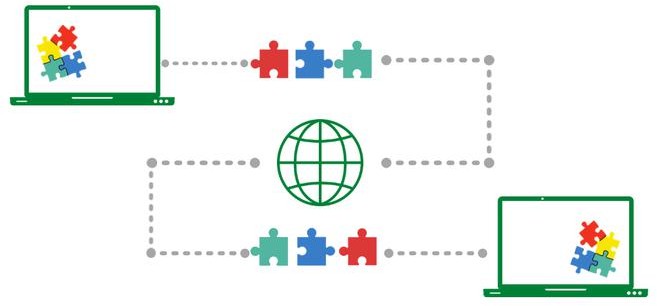
TCP (Transmission Control Protocol) is a key protocol in the Internet protocol suite. Architecture of TCP is shown in Figure 3.3 which states that it operates at the transport layer of the OSI model and works in conjunction with IP (Internet Protocol) to provide reliable and efficient communication over the internet. TCP is a connection oriented protocol, which means it establishes a connection between two devices for transmitting data. This connection ensures that data is transmitted reliably and without errors, even in cases where packets may be lost or delayed during transmission. TCP achieves this reliability through a number of mechanisms, including packet sequencing and acknowledgment, flow control, and congestion control. These mechanisms ensure data is transmitted in the correct order, lost packets are retransmitted, and that the network is not overloaded with too much traffic. In addition to providing reliable communication, TCP also offers other features such as support for multiple simultaneous

connections, error detection and correction, and support for different types of data (such as streaming media or file transfers). Overall, TCP plays a critical role in ensuring that data is transmitted reliably and efficiently over the internet, making it an essential part of modern networking infrastructure.



## Fig 3.3 TCP/IP Protocol

Transmission control protocol model breaks down the data into small bundles to make sure that each and every message reaches it target location intact. This is followed by reassembly of bundles on opposite end into original message, this method is shown in Figure 3.4 . Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go. After a particular message is broken down into bundles, these bundles may travel along multiple routes if one route is jammed but the destination remains the same.



**Figure 3.4 TCP data transmission**

The code reading timeout duration specifies maximum amount of time that software waits for a code to be detected before stopping the trigger signal. If a code is not detected within the specified time frame, the trigger signal will be stopped and the image capture process will be terminated.

Similarly, the maximum code amount to be read specifies the maximum number of codes that software attempts to decode before stopping the trigger signal. If the maximum number of codes is attained, the trigger signal will be stopped and the image capture process will be terminated. By configuring the trigger stop settings, the user can ensure that the code reader software operates efficiently and only captures images when necessary. This can help to improve system performance and reduce processing time, especially in high-volume applications.

* + 1. **Data processing**

In this application, the data processing settings have been configured to limit the output length to a maximum of 256 characters. This means that if a decoded code has a longer output string, it will be truncated to fit within the 256-character limit. The minimum code length has been set to 1, which means that the software will attempt to decode codes that are at least one character in length. Similarly, the maximum code length has been set to 256, which means that the software will not attempt to decode codes that are longer than 256 characters. Numerical filters may also be configured to limit the range of values that can be read, based on the specific needs of the application.

Finally, the code offset number may be set to specify the position of the code within the captured image, which can be useful when working with images that contain multiple codes or other objects that may interfere with the decoding process. By configuring these data processing settings, the user can ensure that the code reader software operates in a way that is tailored to their specific needs, and that the output data is processed in a way that is accurate and reliable.

* + 1. **Communication Protocol**

Communication protocols are essential in the transfer of data from one device to another. They are responsible for establishing and maintaining a reliable and efficient communication link between devices. In the context of barcode scanners, the communication protocol determines how the device outputs the barcode data. The communication protocols available for configuration vary depending on the running mode of the device, such as in Test mode and normal mode. In Test mode, the device is configured for testing purposes, while in normal mode, the device is configured for regular operation.

In this application, the TCP Client protocol has been selected as the communication protocol. The TCP Client protocol is a reliable, connection-oriented protocol that will provide error detection and recovery mechanisms. It is commonly used for network applications requiring higher reliability, such as data transfer and remote access. The TCP Client protocol requires the specification of certain parameters, including the TCP Destination address and TCP port. The TCP Destination address specifies the server IP address to which the barcode scanner will send the data. The TCP port is a unique identifier that specifies the destination application on the server to which the data will be sent. By using the TCP Client protocol, the barcode scanner can send data to a remote server in a reliable and efficient manner. This allows for the data to be processed and analyzed in real-time, enabling businesses to make informed decisions based on the data obtained from the barcode scanner. In conclusion, the communication protocol is an essential aspect of barcode scanner operation. By selecting the appropriate protocol and configuring the necessary parameters, businesses can ensure that their barcode scanner is operating efficiently and effectively. The TCP Client protocol is a reliable and efficient protocol that can be used to transfer data from the barcode scanner to a remote server. In the TCP Client protocol, the TCP Destination address specifies the server IP address that will receive the data, while the TCP port specifies the port number of the server that will receive the data. The IP

address of server is a unique identifier that specifies the location of the server on the network. It is important to ensure that the correct IP address is entered to ensure that the data is sent to the intended server. The port number is a unique identifier that specifies the destination application on the server to which the data will be sent. Different applications use different port numbers, and it is important to ensure that the correct port number is entered to ensure the data is sent to the correct application on the server. By configuring the TCP Destination address and TCP port, the barcode scanner can establish a connection to the server and send the scanned data to the appropriate application for processing and analysis. This enables businesses to collect and analyze data in real-time, improving operational efficiency and decision-making.

* + 1. **Configuration Management**

Configuration management is an essential aspect of managing a barcode scanner device. It involves configuring and managing the user settings for the particular code reader. A user set is a group of parameter values that includes all the settings needed to control the device. In this context, a user set refers to a configuration of the barcode scanner that has been customized to meet specific requirements. When configuring the barcode scanner, it is important to set the device parameters as desired. These parameters include settings such as the scan mode, data format, communication protocol, and other settings that affect the operation of the device. Once the desired parameters have been set, they can be saved into a user set. Saving the parameters into a user set is useful because it enables the user to load the user set to restore the device to the saved group of parameter values again if required. This can be useful in situations where the device needs to be reconfigured, such as when it is being used in a different application or when the device has been reset to its default settings. For example, imagine a warehouse that uses barcode scanners to track inventory. The barcode scanners are configured with specific settings such as the scan mode, data format, and communication protocol that are optimized for the warehouse environment. These settings are saved into a user set and can be loaded onto other barcode scanners in the warehouse to ensure consistent operation across all devices. In addition to saving and loading user sets, configuration management also involves managing user profiles. User profiles are used to store individual user preferences such as display brightness, sound volume, and other settings that are specific to the user. User profiles enable multiple users to share a barcode scanner while maintaining their individual preferences. In conclusion,

configuration management is a critical aspect of managing a barcode scanner device. It involves configuring and managing user settings, saving and loading user sets, and managing user profiles. By effectively managing the configuration of the barcode scanner, businesses can optimize the operation of the device and improve efficiency in their operations.

Finally, in the configuration management setting user need to configure and manage the user settings for the particular code reader. A user set is a group of parameters values with all settings needed to control the device. If you have configured the device parameters as desired it can be saved into user set. It can load the user set to restore device to the saved group of parameter values again if required.

* + 1. **Assigning IMEI to mobile puck**

Hikvision Smart Code Reader is a versatile and reliable solution for automated code reading in manufacturing, logistics, and retail industries. Its high accuracy and speed, combined with its flexible connectivity options and user-friendly software, make it an ideal choice for businesses looking to improve efficiency and reduce errors in their production and distribution processes. In addition to their use in logistics and marketing, QR codes are also used in mobile device management. One example of this is the mobile puck system, which uses QR codes to track the status of mobile devices. Each mobile device has a unique IMEI number assigned by the manufacturer, while each mobile puck has a unique ID number. When a mobile device is stored in a puck, the puck ID is registered in a database and linked to the IMEI number of the device. This makes it easy to track the status of each mobile device throughout the supply chain. QR code readers are used to scan the QR codes on each puck, providing workers with quick access to information about the mobile devices stored inside. This information can include the model and serial number of the device, as well as any repair or maintenance history. By using QR codes to track mobile devices, companies can reduce the risk of loss or theft, as well as ensure that devices are properly maintained and repaired. This can help to improve productivity and reduce costs, while also providing customers with a better experience.

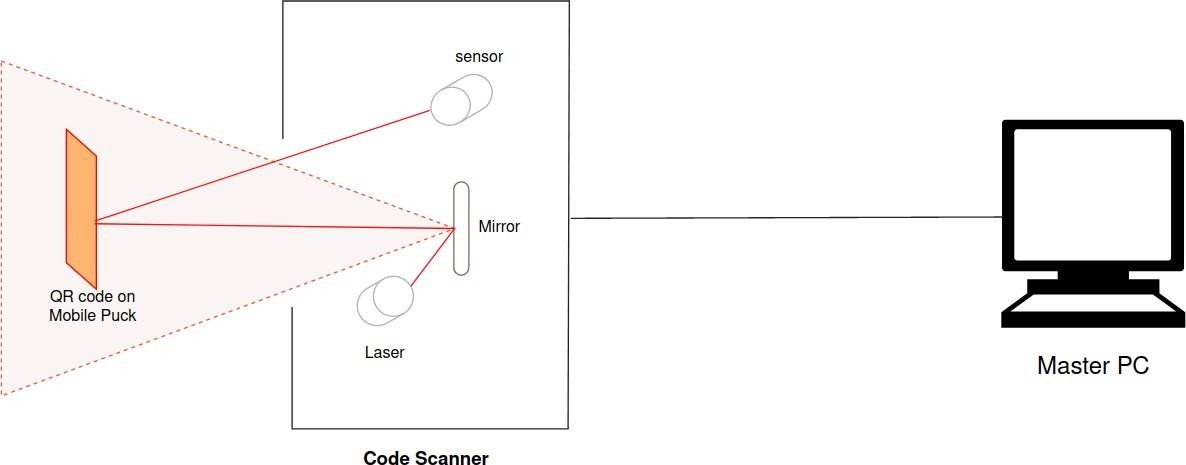


Figure 3.5 Data Acquisition system

Once the puck carrying the mobile device enters the power on station, the QR code is scanned by the code reader. The code reader retrieves the process status of the mobile device from the database stored in the master PC as shown in Figure 3.5. The information retrieved includes the stage of the production process, any defects detected during the process, and the time taken to complete each stage of the process. Each puck has a QR code which is attached to its body which is assigned to the IMEI number of devices that itholds during the cycle. Scanning the QR code (Figure 2) by code reader retrieves the process status of the mobile device from a database stored in the master PC before entering into power on the station.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Resolution | Focal Length | Reading Speed | Frame rate | Software Used |
| MV-ID2004M-  06S-RBN | 704 × 540 | 6mm | 41code/sec | 60 fps | IDVMS |

Table 3.1: Scanner Details

## Power up device

Cosmetic grading is a process of evaluating the external appearance and condition of mobile devices. In the context of mobile devices, cosmetic grading is primarily used to assess the level of wear and tear, scratches, dents, and other signs of usage. The cosmetic grading system typically involves assigning a letter or number grade to a device based on its appearance, with higher grades representing a better condition. A common grading system for mobile devices includes four grades: A, B, C, and D, with A being the highest grade and D being the lowest. Devices that are graded A typically have minimal or no signs of usage, while those graded D may have significant wear and tear or cosmetic damage. Cosmetic grading is an important process for companies involved in buying or selling used mobile devices, as it provides a standardized way to communicate the condition of the device to potential buyers or sellers.

Only those mobile device whose process status is marked as cosmetic grading completed in the database are passed into mobile on station for powering up the device.

## Hardware capabilities

The system includes Beckhoff C6015-0010 which is a powerful industrial PC that offers a high degree of performance and flexibility for automation applications. It is designed to be a cost- effective solution for applications that require reliable, real-time control in a compact form factor. The C6015-0010 as seen in Figure 3.6 is powered by an Intel Atom processor and features a fanless design for improved reliability in harsh environments. It is equipped with multiple interfaces, including Ethernet, USB, and display ports, for easy integration into various automation systems. The PC can be mounted in a variety of ways, including VESA mounting or DIN rail mounting, making it a versatile solution for a range of applications. With its robust design, high-performance capabilities, and flexibility, the Beckhoff C6015-0010 is a reliable choice for industrial automation and control. The Beckhoff C6015-0010 is an Industrial PC (IPC) that can be used to perform many of the same functions as a traditional PLC, making it a versatile and powerful option for industrial automation and control applications.



## Figure 3.6 Beckhoff C6015

This controller is powered by SMPS. The switch mode power supply (SMPS) is a type of power supply that uses a switching regulator to convert electrical power efficiently. SMPS operates by rapidly switching the input voltage on and off, thereby regulating the output voltage. In contrast to the linear power supply, the SMPS generates less heat and provides greater efficiency, making it an ideal choice for powering high-performance electronics. In this case, the PC is powered by a 24 volts DC supply provided by the SMPS. This voltage level is typically used in industrial automation and control systems as it offers a good balance of power and safety. 24 volts DC is a relatively low voltage, which makes it safer to work with compared to higher voltages, reducing the risk of electrical shock. Additionally, the 24 volts DC supply provided by the SMPS is stable and reliable, ensuring that the PC receives a consistent supply of power for optimal performance. The use of the SMPS as a power supply for the PC in an industrial environment ensures that the PC is reliable and efficient, meeting the demands of industrial automation and control systems. The system includes a proximity sensor to ensure the presence of mobile device in the puck while entering into power on station , a proximity sensor is a type of sensor that detects the presence or absence of objects within a certain range without any physical contact. It works by emitting a beam of electromagnetic radiation (such as infrared or ultrasonic waves) and measuring the time it takes for the signal to bounce back after hitting an object. This data is then converted into an electrical signal that is used to trigger a response, such as turning on a light or sounding an alarm. Proximity sensors are commonly used in industrial automation, robotics, and consumer electronics applications. This system has Omron E2K-X which is a series of cylindrical inductive proximity sensors designed for detecting metallic objects. These sensors are available in various sizes with sensing distances ranging from 1.5mm to 15mm. The E2K-X

series as shown in Figure 3.7 is known for its high durability and resistance to shock, vibration, and water. The sensor housing is made of nickel-plated brass or stainless steel, depending on the model, which makes it suitable for use in harsh environments. The E2K-X proximity sensors use an AC-operated coil to generate an electromagnetic field, which is disturbed by the presence of a metallic object. This disturbance is detected by the sensor, which triggers a switch output. The sensors are available with either NPN or PNP output configurations and are compatible with a wide range of input voltages. The E2K-X series also features a built-in LED indicator, which provides visual confirmation of the sensor status. Additionally, the sensors are equipped with a reverse polarity protection circuit and a surge suppressor, which protects them against electrical surges and overvoltage. Omron E2K-X proximity sensors are widely used in automation and robotics applications for detecting metallic objects with high accuracy and reliability.



## Figure 3.7 Proximity sensor

* + 1. **Pneumatic operations**

The system also includes a Pneumatic plunger, A pneumatic plunger is a device that uses compressed air to create a forceful movement of a piston or plunger. This movement is used to push or pull objects or materials, depending on the specific design of the plunger. The compressed air is typically supplied through a hose or tube connected to an air compressor or other source of compressed air. When the air is released into the plunger, it causes the piston or plunger to move rapidly and with great force. Pneumatic plungers are often used in industrial and manufacturing settings to power machines and equipment, but they can also be used in other applications where a strong, quick movement is needed. Pneumatic plungers are commonly used in manufacturing and industrial applications where precise and repetitive movements are required, such as in assembly lines or material handling systems. They are preferred over hydraulic or electric systems in certain situations because they are generally simpler, faster, and more cost-effective to operate. Specifically, the system includes SMC MGPM which is a type of compact guide cylinder that is commonly used in automation and industrial applications. It is designed to provide smooth and precise linear motion for equipment such as conveyor belts, packaging machines, and assembly lines. The MGPM cylinder features an integrated guide rod, which helps to minimize deflection and ensure accurate movement. It also has a compact design that allows for easy installation in tight spaces. The cylinder is made of lightweight aluminum alloy, which makes it durable and resistant to corrosion. The MGPM cylinder is operated by compressed air, which enters through a port on one end of the cylinder and pushes a piston back and forth inside the cylinder. The piston is connected to a rod that extends from the other end of the cylinder, providing linear motion to connected equipment. SMC MGPM cylinders are available in a variety of sizes and stroke lengths to meet the needs of different applications.

It also includes a pneumatic stopper, which is a device used to restrict the movement of the object typically on a conveyor system. It is activated only when PLC sends a command to the respective pneumatic bank solenoid. Thereafter compressed air is passed with help of pneumatic tubes and finally, it stops the mobile device carrying the puck on the conveyor to make the further process of turning on mobile phone.

## PLC integration with pneumatic system

The Process to turn on device is done via PLC logic programmed in ladder logic diagram. In a PLC-based control system, the process of turning on a device such as a motor is typically done through the use of ladder logic diagrams. Ladder logic diagrams are graphical representations of the logic functions used to control industrial processes. They are named after the ladder-like appearance of the diagram, which consists of two vertical rails representing the power supply and the return, and a series of horizontal rungs representing the control circuits. To turn on a device via PLC logic programmed in ladder logic diagram, the following steps are typically taken: The PLC receives input signals from sensors and switches that detect the state of the process or machine. These input signals are wired to the input terminals of the PLC. Based on the inputs received, the PLC logic program evaluates the current state of the process and determines whether to turn on the device or not. This is done by using ladder logic diagrams that consist of a series of logic functions such as AND gates, OR gates, and timers. If the PLC determines that the device needs to be turned on, it sends a signal to the output terminal that is wired to the device. This signal is usually a voltage or current that energizes a relay or a solid- state switch. The device is turned on and begins to perform its function. This could be anything from driving a motor to opening a valve or turning on a light. The PLC continues to monitor the inputs and adjust the output signals as needed to maintain the desired state of the process or machine. The process of turning on a device via PLC logic programmed in ladder logic diagram involves a combination of input signals, logic functions, and output signals that work together to control industrial processes and machines. By using ladder logic diagrams and a PLC, engineers can create flexible and efficient control systems that can be customized to meet the specific needs of their applications.The integration of system is represented in Figure 3.8, when a Programmable Logic Controller (PLC) sends a command to activate the pneumatic cylinder to press the power on button of mobile device, it typically sends an electrical signal to a solenoid valve that controls the flow of compressed air to the cylinder. When the solenoid valve receives the electrical signal from the PLC, it opens and allows compressed air to enter the cylinder, pushing the piston or plunger forward. The speed and force of the plunger's movement can be controlled by adjusting the flow of compressed air into the cylinder. Once the PLC command is completed, the solenoid valve closes, cutting off the flow of compressed air and allowing any

remaining air in the cylinder to escape through an exhaust port. The plunger will then return to its original position, either by the force of a spring or by the pressure of the air escaping through the exhaust port.

## Pneumatic calculations of the system

F = 𝑚𝑔 × µ

Where,

𝑚 = mass of an object

𝑔 = acceleration due to gravity

µ= Co-efficient of friction between rubber and plastic Consider, m = 300 gm

F = 0.3 × 9.81 × 0.5 F = 1.47 N

Considering, Factor of safety (F.O.S) = 2 F = 2 × 1.47

F ≈ 3 N

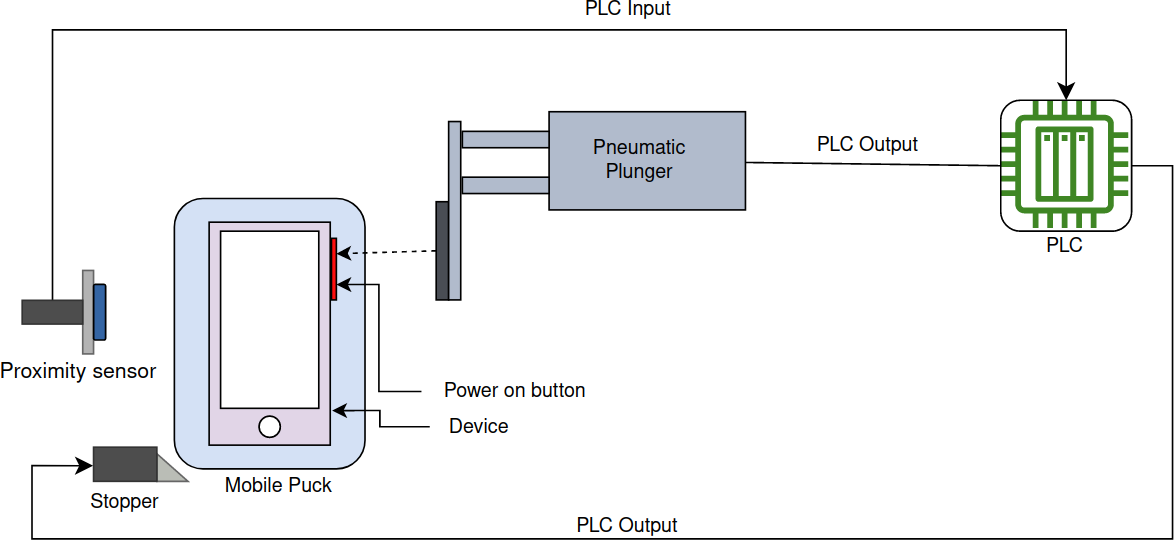
Now consider,

𝐴 = 𝜋𝑟2

Where, dia = 12mm P = F/A

F = 4 × 105 × (3.14) × (0.006)2

= 4.5 N



**Figure 3.8: Power up device system**

|  |  |
| --- | --- |
| Hardware | Specification |
| PLC: Beckhoff C6015-0010 | Memory: 4×512 MB, SSD: 40 GB, Power Supply: 24 VDC |
| Proximity sensor: Omron E2K-X | Sensing distance: 15 mm ± 10%, Power supply: 12 to 24 VD |
| Plunger: SMC MGPM | 16 mm bore, 50 mm stroke |

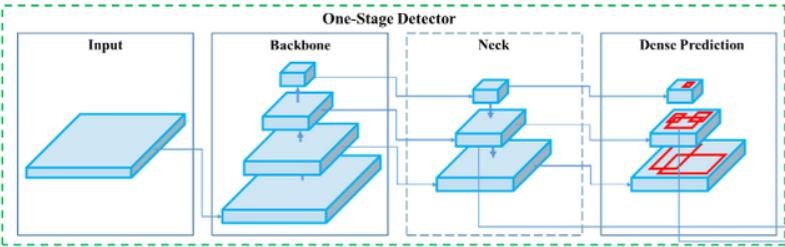
**Table 3.2: Hardware Details**

* 1. **Video Processing:**

YOLOv5 is the fifth iteration of the YOLO (You Only Look Once) object detection algorithm. It was released in June 2020 by Ultralytics team. YOLOv5 builds upon the strengths of the previous versions of YOLO while addressing some of their weaknesses. One of the main improvements of YOLOv5 is its speed. YOLOv5 is faster than its predecessors, with the ability to process up to 140 frames per second on a Tesla V100 GPU. It achieves this speed through a combination of architecture improvements, streamlined implementation, and model compression. In terms of accuracy, YOLOv5 outperforms YOLOv4, with a 10-20% improvement in mAP (mean average precision). YOLOv5 also supports a wider range of object sizes and aspect ratios than previous versions, which can improve accuracy in certain scenarios. YOLOv5 is available in four different sizes: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, with increasing model size and accuracy. Additionally, the Ultralytics team has released pre-trained models for a various of tasks. One of which is object detection and other are instance segmentation, and person detection. YOLOv5 is a influential and adaptable object detection algorithm that can be used in a wide range of applications, including healthcare, security surveillance, and self-driving cars. Object detection is a one of the crucial aspect in computer vision, and there are two main types of object detection models: single stage detection and two stage object detection. This section focuses on single-stage object detectors, and more specifically, the architecture of these models. Single-stage object detectors, like YOLO, are known for their speed and real time object detection capabilities. They consist of three main components: the backbone, neck, and head. The main function of backbone is to extract features from the input image. It typically consists of a convolutional neural network (CNN) that is pre-trained on a large dataset, such as ImageNet. The backbone produces a feature map that encodes high level semantic data about the image. The neck is responsible for aggregating features from different scales and levels of abstraction. It typically consists of a series of convolutional layers that merge the feature maps which are produced from the backbone into a single feature map. The purpose of the neck is to improve the model's apptitude to detect objects of different sizes and aspect ratios. At the end head is responsible for making dense predictions about the objects in the image. It takes the aggregated feature map from the neck and applies a series of convolutional layers to produce a set of bounding boxes and class probabilities for each object in the image.

## High level architecture for single stage object detector

Specific key advantages of single-stage object detectors is their efficiency. Since they process the entire image in a single forward pass, they are much faster than two-stage object detectors, which typically require multiple passes over the image. This makes single-stage object detectors ideal for real time applications, such as autonomous vehicles, video surveillance, and robotics. Another advantage of single stage object detectors is their simplicity. Figure 3.9 illustrates the architecture of these models is relatively straightforward, which makes them easy to train and deploy. Additionally, single-stage object detectors require less memory and computational resources than two-stage object detectors, which makes them more accessible for researchers and developers. However, single stage object detectors also have some limitations. One of the main challenges is their ability to handle small objects, which can be difficult to detect in cluttered or complex scenes. Additionally, single-stage object detectors can struggle with object occlusion, where objects are partially or fully hidden by other objects in the scene. Single-stage object detectors like YOLO have become increasingly popular due to their speed, simplicity, and real time object detection capabilities. The architecture of these models, which contain backbone, neck, and head, allows it to efficiently process the entire image and make dense predictions about the objects in the scene. While single-stage object detectors have some limitations, they remain a valuable tool for a wide range of applications in computer vision.



## Figure 3.9 One stage detector (Alexey,2020)

The model neck is a critical component of single-stage object detection models like YOLO which extracts feature pyramids that enable the model to detect objects of different sizes and scales. In object detection, it is important to detect objects at different scales and sizes. For example, a person in the foreground of an image may be much larger than a person in the background. Additionally, objects of different classes may have different aspect ratios and sizes. Therefore, it is important for the model to be able to extract features that can distinguish objects of different scales and sizes. The feature pyramid is a set of multi-scale feature maps that are obtained from the backbone network. These feature maps contain information about the image at different levels of abstraction. The lower tier of the pyramid contain high resolution feature maps that capture intricate details, while the higher levels of the pyramid contain lower resolution feature maps that capture more global information about the image.

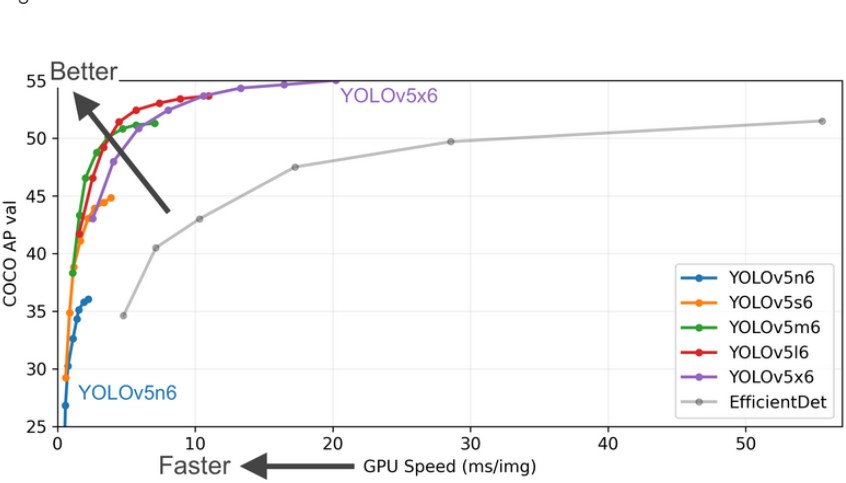
The model neck typically contain of a series of convolutional layers that merge the feature maps from the backbone into a single feature pyramid. The neck may also include additional layers, such as pooling or upsampling layers, to modify` the spatial resolution of the feature maps. The feature pyramid is then used by the model head to make dense predictions about the objects in the image. The head takes the feature pyramid as an input thereafter applies a series of convolutional layers to produce a set of bounding boxes and generates class probabilities for each object in the image. By using a feature pyramid, the model is able to detect objects of different sizes and scales. The high level feature maps in the pyramid are able to capture the global context of the image, which can help the model detect larger objects. The low level feature maps in the pyramid are able to capture fine grained details, which can help the model detect smaller objects. The model neck is an important component of single stage object detection models like YOLO. It is responsible for extracting a feature pyramid that enables the model which recognizes objects of different sizes and scales. By using a feature pyramid, the model can capture both global and fine grained information about the image, which can improve its ability to detect objects in complex scenes. The model head is the final component of a single- stage object detection model like YOLO. Its primary role is to take the feature pyramid generated by the model neck and produce dense predictions about the objects in the image. One of the key tasks performed by the model head is to utilize anchor boxes to the feature maps. Anchor boxes are pre-defined boxes of different sizes and aspect ratios which is used to detect objects at different scales and sizes. The head applies the anchor boxes to the feature maps at

different scales, and then predicts the probability of an object being present within each anchor box. Once the anchor boxes have been applied, the head predicts the class probabilities for each object in the image. This involves assigning a probability to each anchor box for each possible class. For example, if the model is trained to detect cats, dogs, and cars, then the head will predict a probability for each anchor box that indicates the likelihood of the object being a cat, dog, or car. The final stage of the model head is to estimate the bounding boxes for each object in the image which involves determining the coordinates of a bounding box that tightly encloses each object in the image. The bounding box is defined by four values: the x and y coordinates of the box's center, its width, and its height.

To predict the bounding boxes, the model head applies regression to each anchor box. Regression is a technique that involves predicting a set of values that can be used to transform an anchor box into a bounding box. The head predicts four values for each anchor box that indicate the x and y coordinates of the box's center, its width, and its height. Finally the model head combines the predictions for the anchor boxes, class probabilities, and bounding boxes to produce the final output for the object detection task. This output includes a list of all objects detected in the image, along with their class labels, objectness scores, and bounding boxes. The model head is a critical component of single-stage object detection models like YOLO. Its ability to predict object class probabilities and bounding boxes accurately is essential for the success of the model. Moreover, its ability to perform these predictions in real-time is what sets YOLO apart from other object detection algorithms. The model head performs the final stage operations in single-stage object detection models. It applies anchor boxes to the feature maps, predicts the class probabilities for each object, and predicts the bounding boxes for each object. By combining these predictions, the model head produces the final output for the object detection task.

## YOLOv5 Architecture

YOLOv5 has five different sizes: n, s, m, l, and x. Each size corresponds to a different number of parameters in the model, and therefore a different trade-off between speed and accuracy. The "n" size, also known as the "nano" size, is the smallest and fastest variant of YOLOv5. It has the fewest number of parameters and is designed to run on resource-constrained devices such as smartphones or embedded systems. The "s" size is slightly larger than the "n" size, with more parameters and improved accuracy. It is designed for real-time applications where speed is important, but some sacrifice in accuracy is acceptable. The "m" size is the default size for YOLOv5 and strikes a good balance between speed and accuracy. It has a moderate number of parameters and is suitable for a wide range of applications. The "l" size is larger than the "m" size, with more parameters and better accuracy. It is designed for applications where accuracy is more important than speed, such as medical imaging or autonomous driving. Finally, the “x” size is the largest and most accurate variant of YOLOv5. It has the most parameters and is suitable for applications where the highest levels of accuracy are required, such as satellite imaging or robotics. The ability to choose from different sizes of the YOLOv5 model allows developers to select the best trade-off between speed and accuracy for their specific use case. This flexibility makes YOLOv5 a versatile and widely applicable object detection algorithm.



## Figure 3.10 YOLOv5 performance

The main operations used in all five variants of YOLOv5 are similar, there are some differences between them beyond just the number of layers and parameters. These differences are primarily related to the input image size and the number of anchor boxes used. Each of the five variants of YOLOv5 has a different default input image size, ranging from 320x320 pixels for the smallest "n" variant to 640x640 pixels for the largest "x" variant. The input image size can be adjusted as needed, but the default size is optimized for each variant's specific trade off between speed and accuracy. In addition, each variant uses a different number of anchor boxes, which are predefined bounding boxes used to localize objects in the image. The smallest "n" variant uses only 3 anchor boxes per cell, while the largest "x" variant uses 9 anchor boxes per cell. The use of more anchor boxes can improve detection accuracy but also increases computational complexity as presented in Figure 3.10. Other differences between the variants may include the use of different activation functions or other hyperparameters, although these changes are usually minor. While the main operations used in all five variants of YOLOv5 are similar, there are some important differences between them beyond just the number of layers and parameters. Developers should carefully consider these differences when selecting the appropriate variant of YOLOv5 for their specific use case.

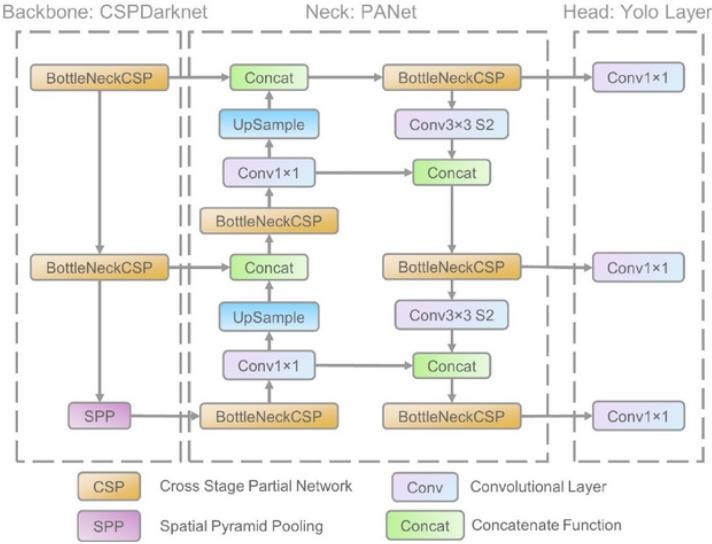


## Table 3.3 Yolo model performance

While it is true that all variants of YOLOv5 share the same three main components CSPDarknet53 as a backbone, SPP and PANet in the model neck, and the head used in YOLOv4 there may be some minor differences in the implementation of these components between the different variants.

For example, while all variants use CSP-Darknet53 as a backbone, the specific configuration and number of layers in the backbone may vary between variants. Similarly, while all variants use SPP and PANet in the model neck, there may be some differences in the specific configuration or hyperparameters used . As per Figure 4.11 the main design principles of YOLOv5 are consistent across all variants, with a focus on using a lightweight backbone network and efficient feature pyramids to achieve high accuracy and real-time performance. The head used in YOLOv4, which includes anchor-based bounding box regression and class prediction, is also a key component of all YOLOv5 variants. While there may be some minor differences in the implementation of the backbone, neck, and head components between different variants of

YOLOv5, the overall design principles and approach to object detection are consistent across all variants.



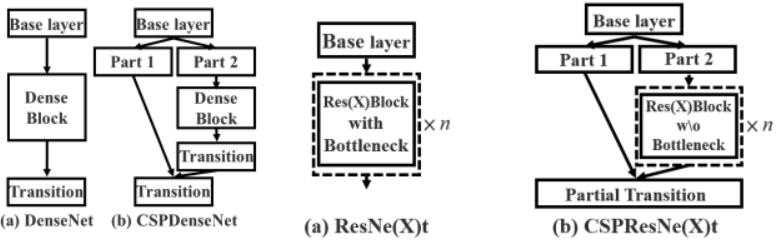
## Figure 3.11 YOLO Architecture (Kangjie ,2021)

* + 1. **CSP Darknet53**

A YOLOv5 uses CSP Darknet53 which is the backbone.It is essentially a modified version of the convolutional network Darknet53, which is used as the backbone for YOLOv3. The modification comes in form of the Cross Stage Partial (CSP) network strategy, which was introduced by the YOLOv5 authors to improve the performance of the network. The CSP network strategy works by splitting each convolutional layer in Darknet53 into two smaller layers. The first layer performs a small number of initial convolutional operations on the input data, while the second layer performs the remaining convolutional operations on the output of the first layer. This split allows the network to perform more efficient and effective feature extraction by lowering the amount of computation required. In addition to the CSP network strategy, the authors of YOLOv5 also made some other modifications to Darknet53 to further improve its performance. For example, they added a number of residual connections to the network to help mitigate the effects of vanishing gradients during training. They also increased the number of filters in the early layers of the network to improve its ability to detect small objects. The resulting CSP- Darknet53 network is highly optimized for object detection tasks, with a good balance between accuracy and computational efficiency. It has been shown to outperform the original Darknet53 network on a range of object detection benchmarks, including COCO and VOC. Overall, the Cross Stage Partial network strategy used in CSP-Darknet53 is an important innovation in the development of object detection models like YOLOv5. By splitting convolutional layers into smaller sub-layers, the CSP strategy enables more efficient and effective feature extraction, leading to improved accuracy and faster inference times. One of the benefits of using residual and dense blocks in deep neural networks like YOLO is that they help to reduce the vanishing gradient problem, which can occur when gradients become too small and fail to propagate through many layers. However, the use of these types of blocks can also lead to the problem of redundant gradients, which slows down training and makes it harder for the network to learn.

The Cross Stage Partial (CSP) network strategy used in YOLOv5 helps to address this problem by minimizing the gradient flow. Specifically CSP strategy presented in Figure 3.12 splits each convolutional layer in the network into two smaller layers, with the first layer performing a small number of initial convolutional operations on the input data, and the second layer performing the remaining convolutional operations on the output of the first layer. In addition to the CSP strategy, YOLOv5 also uses other techniques to help with gradient flow and training stability,

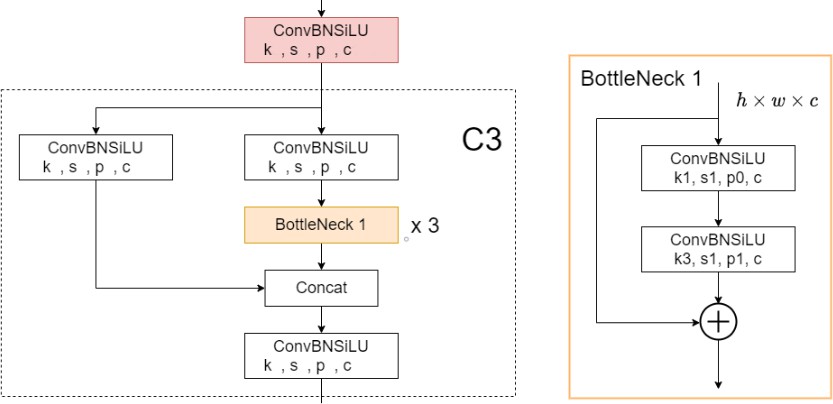
such as skip connections, batch normalization, and weight decay. These techniques work together to ensure that the network is able to learn effectively and efficiently, without getting stuck in local minima or suffering from other common problems that can arise during training. The CSP network strategy is an important innovation in the object detection models like YOLOv5. By truncating the gradient flow and reducing the problem of redundant gradients, it helps to improve the efficiency and stability of the network during training, leading to better performance on object detection tasks.



## Figure 3.12 Applying CSPNet to RESNet

The Cross Stage Partial Network (CSPNet) strategy used in YOLOv5 involves partitioning the feature map of the base layer into two parts and then merging them through a cross-stage hierarchy. The CSPNet architecture is designed to address the issue of redundant gradients that can arise in deep neural networks that use dense or residual connections. In the case of YOLOv5, the CSPNet strategy is applied to the Darknet53 backbone network. In CSPNet, the feature map of the base layer is partitioned into two parts, which are processed by two separate convolutional layers. The output of these two layers is then concatenated and passed through another set of convolutional layers to get final output. This process is repeated multiple times in a cross-stage hierarchy, with the outputs of each stage being concatenated and passed through the next stage of the network. The use of the cross-stage hierarchy in CSPNet helps to reduce the computational complexity of the network while maintaining a high level of accuracy. It also helps to preserve the spatial resolution of the feature maps, which is important for accurate object detection. In YOLOv5, the CSPNet strategy is used to enhance the efficiency and accuracy of the network,

while also reducing the amount of redundant gradient information that is propagated during training. By partitioning the feature map of the base layer and merging it through a cross-stage hierarchy, YOLOv5 is able to achieve state of the art performance on object detection tasks while maintaining real time speed.



**Figure 3.13 Bottle NeckCSP module architecture**

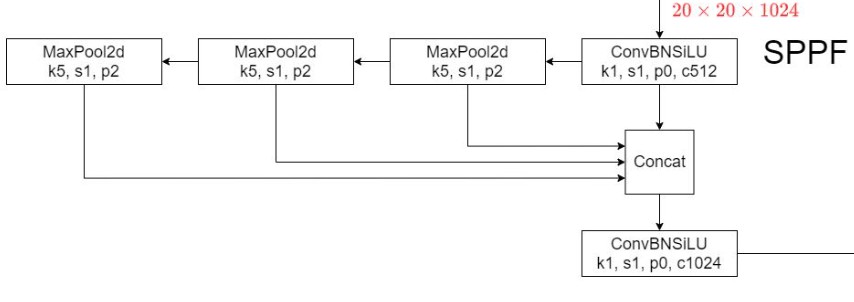
Applying this strategy to YOLOv5 gives significant advantages, as it allows for reduction in number of parameters and computational workload. This, in turn results in faster inference speed which is crucial factor for real time object detection model.

## Neck of YOLOv5

YOLOv5 introduced two major changes to the model neck compared to previous versions of YOLO. The first change is the use of a variant of Spatial Pyramid Pooling (SPP). SPP is a technique used to get features from an input image at multiple scales. In YOLOv5, SPP module is used to extract multi-scale features from the backbone feature maps. This helps to improve the detection performance of the network by capturing objects at different scales and sizes. The second change is the modification of the Path Aggregation Network (PANet) by integrating the BottleNeckCSP in its architecture. The PANet is a network designed to aggregate features across different feature maps to improve object detection performance. In YOLOv5, the PANet module is modified by incorporating the BottleNeckCSP, which is a variant of the CSPNet used in the backbone. The incorporation of the BottleNeckCSP in the PANet architecture helps to further reduce the number of parameters in the network while maintaining or improving its detection performance. This is because the BottleNeckCSP module combines the benefits of the CSPNet and the bottleneck structure, which allows for efficient feature extraction and parameter reduction. The changes made to the model neck in YOLOv5 help to improve the detection performance of the network by extracting multi-scale features and aggregating them efficiently across different feature maps. This leads to more accurate and efficient object detection, which is important for real-time applications such as self-driving cars and security surveillance. SPPF (Spatial Pyramid Pooling Fusion) is an improved version of the SPP block used in YOLOv5. It was introduced to increase the speed of the network by reducing the number of operations required in SPP block. SPPF works by using a 1x1 convolutional layer to reduce the number of input channels before applying the SPP block, and then uses another 1x1 convolutional layer to increase the number of output channels after SPP block.

The SPPF block takes an input tensor and applies three pooling operations (max pooling, average pooling, and maxout pooling) with different kernel sizes to the input tensor. The output of the pooling operations is concatenated to form a single tensor with a fixed size. This tensor is then passed through a series of convolutional layers and batch normalization layers to produce a fixed-length output. By using SPPF, YOLOv5 can effectively increase the receptive field of the network while reducing the computational cost. This allows YOLOv5 to achieve higher accuracy and faster inference times compared to previous versions of YOLO. In addition to SPPF,

YOLOv5 also incorporates the Path Aggregation Network (PANet) with a modified architecture that includes the BottleNeckCSP block which is shown in Figure 3.13. This modification was introduced to further increase the accuracy and speed of the network. BottleNeckCSP is a modification of the original CSP block used in the backbone network. It consists of a bottleneck layer with 1x1 and 3x3 convolutions followed by a CSP block. This modification helps to reduce the number of parameters in PANet and increase the speed of the network.



## Figure 3.14 Structure of SPPE block

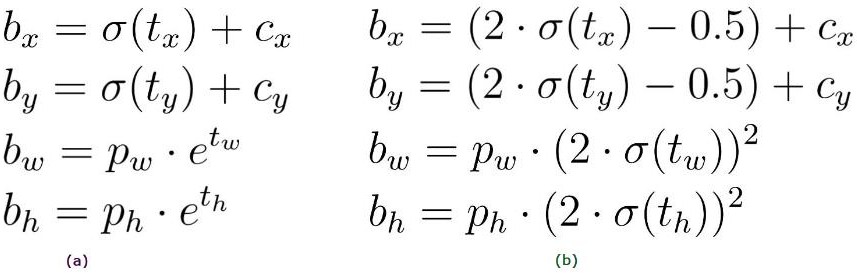
* + 1. **Head of network**

The head of YOLOv5, like its predecessors YOLOv3 and YOLOv4, is accountable for predicting the bounding boxes and their associated class probabilities. The head consists of three convolutional layers: a 1x1 layer, a 3x3 layer and a final 1x1 layer. These layers produce a feature map that is then used to generate the final output. In previous versions of YOLO, the location of the bounding boxes was represented as the center point x,y and the width and height w,h of the bounding box. However, in YOLOv5, the representation has been changed to the top left corner x1,y1 and bottom right corner x2,y2 of the bounding box. This change was made to simplify the training process and to improve the accuracy of the predictions.

The new representation of the bounding box coordinates can be computed using the following equations:

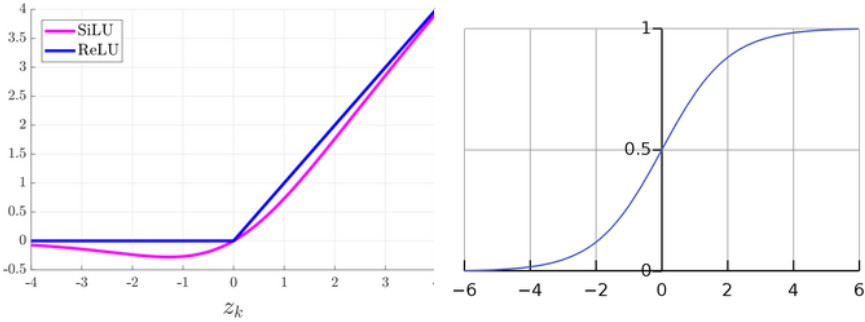
x1 = x – w/2 y1 = y – h/2 x2 = x + w/2 y2 = y + h/2

where x,y are the predicted coordinates of the center of the bounding box and w,h are the predicted width and height of the box. The new representation has several advantages over the old one. For example, it is easier to understand and work with in practice, and it allows the model to learn more easily to predict boxes of varying sizes and aspect ratios. In addition to the change in the representation of the bounding box coordinates, YOLOv5 also introduced several other improvements to the head. For example, the activation function used in the final layer was changed from sigmoid to softmax. This allows the model to more easily handle cases where there are multiple objects in the same region of the image, as it assigns probabilities to each class independently. Another improvement made in YOLOv5 was the addition of a focal loss function, which helps to address the problem of class imbalance in the training data. This loss function gives more weight to hard examples (i.e., those that are misclassified with high confidence) and less weight to easy examples (i.e., those that are correctly classified with high confidence).



## Activation Function

SiLU (Sigmoid Linear Unit) is an activation function that has been gaining popularity recently. It is a smooth non-monotonic activation function that is defined as SiLU(x) = x \* sigmoid(x), where sigmoid(x) = 1 / (1 + exp(-x)). SiLU is a continuous and differentiable function, which makes it useful for backpropagation during the training phase. The use of SiLU in YOLOv5 helps improve the model performance in terms of accuracy and speed. SiLU has been shown to be a faster and more accurate alternative to other activation functions such as ReLU and Leaky ReLU. In addition, SiLU has been shown to be more robust to adversarial attacks. Sigmoid function, on the other hand, is commonly used for binary classification problems. In YOLOv5, it is used to predict the probability of an object being present in a given bounding box. The sigmoid function maps any value to a range between 0 and 1, which is ideal for binary classification tasks. The sigmoid function is defined as sigmoid(x) = 1 / (1 + exp(-x)). The combination of SiLU and Sigmoid activation functions in the YOLOv5 architecture provides a good balance between accuracy and speed. SiLU helps to improve speed of the network while maintaining accuracy, while Sigmoid helps accurately predict the probability of an object being present in each bounding box.



## Figure 3.15 a) SiLU b) ReLU

SiLU is an activation function that was introduced in a paper by Google researchers in 2017. The SiLU function has some advantages over other activation functions, such as the widely used ReLU (Rectified Linear Unit), including a smoother gradient and a better performance on deep networks. In YOLOv5, the SiLU function is used in the hidden layers of the network, where it helps to improve the performance of the model by providing a smoother gradient and reducing the impact of the vanishing gradient problem. The Sigmoid function, on the other hand, is used in the output layer of the network to generate the final predictions, including the objectness scores and the bounding box coordinates. The choice of activation functions is an important consideration when designing a deep learning model, as different functions have a significant impact on the performance of the network. In YOLOv5, the use of SiLU and Sigmoid functions has been shown in figure (3.15) to improve the performance of the model, particularly in the context of object detection tasks.

## Loss Function

The YOLOv5 model yeilds three outputs the classes of the detected objects, their bounding boxes, and the objectness scores. To train the model, loss function is defined as a combination of three different losses classes loss, objectness loss, and location loss. The classes loss and objectness loss are computed using the binary cross-entropy (BCE) loss function, while the location loss is computed using the Complete Intersection over Union (CIoU) loss function. The BCE loss function is commonly used for binary classification problems where each class is either positive or negative. In the case of YOLOv5, the classes loss is computed by comparing

the predicted classes of the detected objects with the true classes of the ground truth objects. The BCE loss function measures the difference between the predicted and true classes and it is defined as:

BCE loss = -1/N × sum(y × log(p) + (1-y) × log(1-p))

where y is the true class either 0 or 1, p is the predicted probability of the class, and N is the total number of objects in the batch. Objectness loss, on the other hand, is used to predict the presence of an object in the image. The objectness score is a measure of how likely it is that an object is present in the image. The objectness loss is also computed using the BCE loss function, but in this case, the true class is always 1 (indicating the presence of an object), and the predicted class is the objectness score. The location loss is computed using CIoU loss function, which is a modified version of the Intersection over Union (IoU) loss function. The IoU is a measure of how well the predicted bounding box coincide with the ground truth bounding box. The CIoU loss function takes into account both the size and location of the bounding boxes and is defined as:

CIoU loss = 1 - IoU + α \* v

where IoU is the intersection over union, alpha is a balancing parameter, and v is a penalty term that accounts for the difference in the aspect ratios of the predicted and ground truth bounding boxes. The CIoU loss function encourages the model to predict more accurate bounding boxes that are closer to the ground truth.

The final loss is a combination of the three losses and is defined as:

Total loss = classes loss + objectness loss + λ\* location loss

where lambda is a balancing parameter that handles the relative importance of the location loss and the other losses. The total loss is used to update the model parameters during the training process using stochastic gradient descent (SGD) or other optimization algorithms.

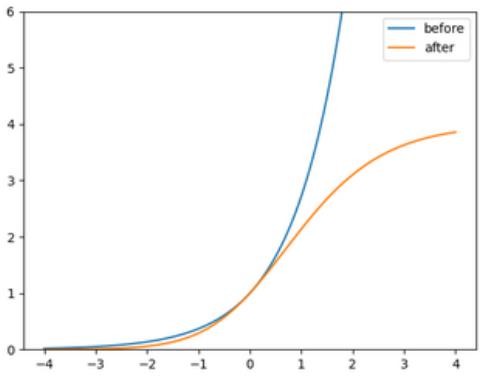
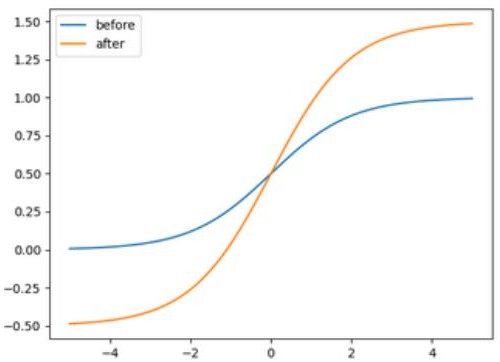
## Other Improvements Focus layer

The first improvement worth mentioning in YOLOv5 is the addition of Focus Layer, which replaces the first three convolutional layers of the network. The purpose of this layer is to reduce the number of parameters, FLOPS which is also known as floating point operations per second, and CUDA memory usage to improve the speed of the forward and backward passes with only minor effects on the mean Average Precision (mAP). The Focus Layer takes an input image and applies a convolutional operation with a small kernel size of 3x3, followed by a down-sampling operation to reduce the resolution by a factor of 2. This operation is repeated once more, resulting in an output feature map with a smaller spatial resolution and a larger number of channels compared to the input image. The use of the smaller kernel size and the down sampling operation decreases the number of parameters and computational requirements, while still preserving the most important features in the input image. The Focus Layer has several advantages over the previous implementation of the first three convolutional layers. First, it reduces the number of parameters in the network, which minimizes the risk of overfitting and improves the speed of training. Second, it suppresses the number of FLOPS required for the forward and backward passes, making the network faster and more efficient. Finally, it reduces the CUDA memory usage, which is important for running the network on GPUs with limited memory. The addition of the Focus Layer is a significant improvement to YOLOv5, as it reduces the computational requirements and improves the speed of the network without sacrificing accuracy.

## Eliminating Grid Sensitivity

In the previous versions of YOLO, the predicted bounding boxes were limited to the cells of the output grid, which resulted in difficulties detecting objects at the image corners. This limitation is referred to as the "grid sensitivity" problem. The issue arises from the fact that the coordinates of the bounding boxes are predicted relative to the top left corner of the cell, and hence if the center of an object is located near the boundary of two cells, it may be detected by only one of the cells, causing the network to miss some objects. To solve this problem, YOLOv5 introduced a modification in the equations used for the prediction of bounding box coordinates. Instead of

predicting the offset of the center point relative to the top left corner of the cell in the range of (0-1), the offset is now predicted relative to the whole image in the range of (-0.5, 1.5). This change expands the range of possible offsets, allowing the network to predict objects located near the image corners. Additionally, scaling ratios of the height and width in the previous equations were unbounded, which can cause issues while training instabilities. In YOLOv5, the height and width scaling ratios are now bounded, which further improves the stability of the training process. These modifications in the equations used to predict the bounding box coordinates help to eliminate the grid sensitivity problem and improve the detection performance of the YOLOv5 network. The result presented in Figure 3.16 shows that YOLOv5 performs better than the previous versions of YOLO in terms of detection accuracy and speed, making it a state of the art object detection network.



## Figure 3.16 Eliminating grid sensitivity

**Running Environment**

YOLOv5 is implemented in PyTorch, a popular open source machine learning framework that provides a Python environment for building and training deep learning models. PyTorch offers a number of advantages over Darknet, the C-based framework used in previous versions of YOLO. One advantage of PyTorch is its dynamic computational graph, which allows for more flexible and efficient computation during training and inference. This is in contrast to the static computational graph used in Darknet, which requires recompiling the entire network architecture every time a change is made. Another advantage of PyTorch is its ease of use and large community support. PyTorch provides a wide range of pre-built modules, including loss functions, activation functions, and optimization algorithms, making it easy to build and train

complex models like YOLOv5. Furthermore, PyTorch's integration with other popular libraries like NumPy and SciPy make it easy to preprocess and manipulate data for use with YOLOv5, as well as visualize and analyze model outputs. Finally, PyTorch offers the ability to run models on GPUs, which can significantly speed up training and inference. With PyTorch, YOLOv5 can be easily deployed on a wide range of hardware, including local workstations and cloud-based platforms. The switch to PyTorch has provided YOLOv5 with a more flexible, efficient, and user-friendly environment for training and inference, and has helped to further improve the performance of the model.

## Hardware Integration

The system uses this algorithm to detect the logo on the mobile device and on-screen condition which is deployed in the Nvidia Xavier hardware. Nvidia Xavier is a high-performance system- on-chip (SoC) that is designed for use in artificial intelligence and machine learning applications. It includes a powerful CPU, GPU, and dedicated hardware for accelerating deep learning algorithms. This makes it an ideal platform for deploying advanced computer vision algorithms like YOLOv5 for detecting logos on mobile devices.When deployed on Nvidia Xavier hardware, YOLOv5 takes advantage of the system's high-performance processing capabilities to achieve even faster and more accurate object detection. The Xavier's GPU as seen in Figure 3.17 is particularly well-suited for accelerating deep learning algorithms like YOLOv5, allowing it to process large amounts of data in parallel.



## Figure 3.17 Nvidia xavier

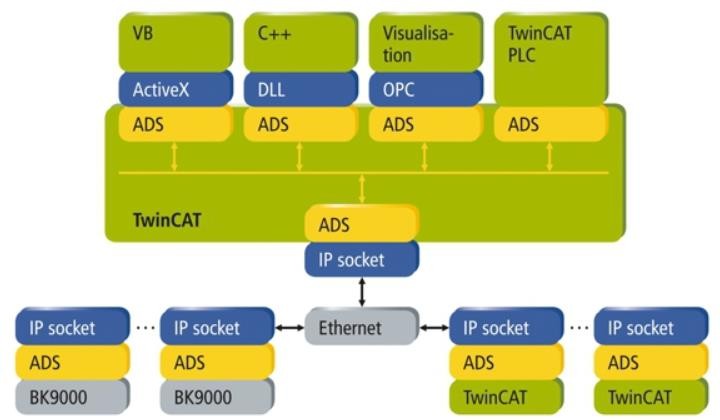
To detect logos on a mobile device screen, the YOLOv5 algorithm is trained on a dataset of images that includes examples of the logo in different sizes, orientations, and lighting conditions. The training data is carefully curated and annotated to ensure that the algorithm can learn to recognize the logo accurately. The algorithm takes an image of the mobile device screen as input and then uses its CNN architecture to detect the logo which is present on a mobile screen. The combination of Nvidia Xavier hardware and the YOLOv5 algorithm provides a powerful platform for object detection and computer vision applications. With its high-performance processing capabilities and advanced deep learning algorithms.

In an object detection system, the output of the detection model refers to the information about the objects that are being detected in an image or video. This information typically includes the location, size, and class of each detected object. When the object detection model runs on a device like an Nvidia Xavier, the output is generated by the model and stored in memory. This output needs to be shared with other devices or systems that may be processing the data, such as an Industrial PC. One way to share the output of the object detection model is through shared memory. Shared memory is a form of inter-process communication (IPC) that allows multiple processes or threads to access the same region of memory. This means that the object detection model can write its output directly to a shared memory location, and the Industrial PC can read the output from that same location. Using shared memory has several advantages over other forms of IPC, such as message passing or sockets. First, it is typically faster and more efficient since there is no need to copy data between processes or serialize and deserialize data for transmission. Second, shared memory can be used for large data structures, making it ideal for passing the output of complex models like object detection. To use shared memory for object detection output, the Industrial PC needs to set up a shared memory region and allocate memory for the output data. The object detection model is then configured to write its output to that shared memory location, using the appropriate data structures and memory offsets. Once the output is written to shared memory, the Industrial PC can read the output and process it as needed. This might involve displaying the detected objects on a screen, logging the output for analysis, or triggering some other action based on the detected objects. This is a powerful and efficient way to integrate computer vision capabilities into industrial applications. It allows multiple devices and systems to work together seamlessly, with minimal overhead and maximum performance.

## Communication Protocol

The data is shared within IPC and Xavier with help of the Automation device specification (ADS) protocol. It is a transport layer within the TwinCAT system, developed by Beckhoff Automation for data exchange between different software modules, such as operation between the NC and the PLC. This protocol enables communication with other tools from any point within TwinCAT, making it a powerful communication tool for automation systems. One of the main advantages of the ADS protocol is that it provides a high level of flexibility and versatility. The protocol allows for easy communication between different devices, regardless of their location within a networked system. This means that data can be accessed and exchanged from any point within the system, which simplifies data management and improves overall system efficiency. Another key benefit of the ADS protocol is its ability to provide reliable and secure communication between devices. The protocol is based on TCP/IP, which is a widely used and well-established network protocol. As a result, the ADS protocol is compatible with a wide range of devices and systems is able to operate reliably in different network environments. The ADS protocol is designed to be lightweight and efficient, which allows it to handle large amounts of data quickly and accurately. This is particularly important for automation systems that require real-time data exchange, as it ensures that data is transmitted quickly and accurately, and that system processes are not interrupted or delayed. The ADS protocol operates by establishing a connection between two devices, which are then able to exchange data using the protocol. When a connection is established, the devices negotiate the transfer mode, which determines how the data will be transferred between the devices. The transfer mode can be either cyclic or acyclic, depending on the requirements of the system. Cyclic data exchange refers to the exchange of data at fixed intervals, which is useful for real-time systems that require continuous data exchange. Acyclic data exchange, on the other hand, refers to the exchange of data on demand, which is useful for systems that require occasional data exchange.

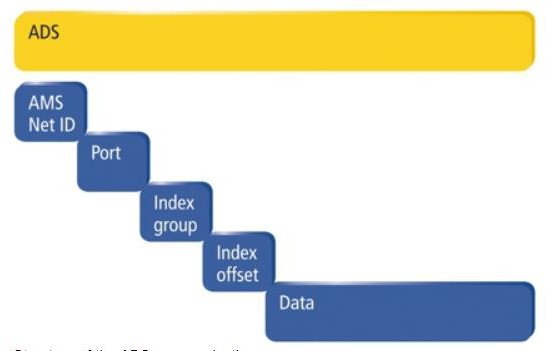
The ADS protocol provides several different data transfer modes, including read and write, notification, and asynchronous communication. These transfer modes enable the system to exchange data in various ways depending on the requirements of the application. The key feature of the ADS protocol is its ability to provide seamless integration with other software tools and systems. The protocol is able to communicate with a wide range of third-party software tools and systems, which makes it an ideal choice for automation systems that require integration with other systems and devices. The ADS protocol is a powerful communication tool for automation systems, enabling reliable, secure, and efficient data exchange between different devices and software modules. The structure of ADS presented in Figure 3.18 signifies its versatility, reliability, and real-time capabilities make it an ideal choice for automation systems that require high levels of data exchange and integration with other systems and devices.



## Figure 3.18 ADS Protocol

The ADS known as Automation Device Specification functions are a set of tools within the Beckhoff system that provide a method for accessing the Bus Coupler information directly from a PC. The Bus Coupler is a hardware device that is used in Beckhoff's I/O systems to connect the input output modules to the fieldbus. One of the main advantages of using the ADS functions is that they provide a simple and efficient way to access the Bus Coupler information without the need for additional software or hardware. This information can be accessed from any point within the system, which provides greater flexibility and control. The ADS functions are

typically used to configure and monitor the Bus Coupler. For example, users can use the functions to read or write the Bus Coupler configuration parameters, such as the IP address or device name. Additionally, the functions can be used to monitor the state of the Bus , such as the connection status or the number of connected I/O modules. Another advantage of using the ADS functions is that they provide a standardized interface for accessing the Bus Coupler information. This means that users can develop custom software tools or interfaces that are compatible with the ADS functions, which can be used to extend the functionality of the Beckhoff system. The ADS functions are also highly configurable and provide a wide range of options for accessing the Bus Coupler information. For example, users can choose to access the information asynchronously or synchronously, depending on their specific requirements. Additionally, users can specify the data format, such as whether the data should be transmitted in binary or ASCII.



## Figure 3.19 ADS Architecture

The ADS functions are a powerful tool within the Beckhoff system that provides a simple and efficient way to access the Bus Coupler information directly from a PC. They provide a standardized interface that can be used to develop custom software tools or interfaces, and are highly configurable, providing a wide range of options. Understanding structure of ADS as in Figure 3.19 it consists of AMSNetID is a unique identifier used to reference the device that needs to be addressed within a networked system. AMSNetID is derived from Media Access Control (MAC) address of the first Ethernet port (X001) of the device. AMSNetID typically uses

bytes 3 to 6 of the MAC address, followed by ".1.1". For example, if the MAC address of a device is 00-01-05-01-02-03, the corresponding AMSNetID would be 5.1.2.3.1.1. The AMSNetID is important in a networked system as it enables devices to communicate with each other using the ADS protocol. By using the AMSNetID, a device can send messages to another device on the network, without needing to know its IP address or location. This simplifies the communication process and allows devices to be added or removed from the network without affecting the system's operation. It is important to note that the AMSNetID is not the same as IP address of the device. While IP address is used for communication over the Internet Protocol (IP), the AMSNetID is specific to the Beckhoff system and is used for communication over the ADS protocol. The AMSNetID provides a unique identifier which allows devices to communicate between a networked system, simplifying the communication process and enabling the efficient exchange of data. In a networked system that uses the ADS protocol, devices communicate with each other using the AMSNetID and a specific port number. The port number is used to distinguish sub-elements or services within the connected device. For example, port 851 is used for local process data in the PLC runtime 1. This means that if a device wants to access the local process data on a PLC that is running runtime 1, it would use the AMSNetID of the PLC and port number 851 to establish a connection and exchange data. It's important that different devices and services may use different port numbers. For instance, a different port number might be used for accessing global process data, accessing the PLC configuration, or accessing diagnostics information.Therefore, understanding the port numbers and their associated services is crucial for configuring and troubleshooting a networked system that uses the ADS protocol. The port number distinguishes sub-elements or services within a device and is used in conjunction with the AMSNetID to establish connections and exchange data between devices within a networked system. In this case, port 851 is used for accessing local process data in the PLC runtime 1. In a networked system that uses the ADS protocol, the index group is used to distinguish different types of data within a specific port number. The index group is a 16-bit number that is used in conjunction with the port number and the AMSNetID to specify the location of the data to be accessed. For example that user want to access a specific variable within a PLC that is connected to the network.

Firstly need to know the AMSNetID of the PLC, the port number that is associated with the service we want to access, and the index group that corresponds to the data we are interested in. Assuming that having this information, we can establish a connection to the PLC and request the data associated with the specific index group. The ADS protocol will then provide us with the data that we requested. Different services and devices may use different index groups to differentiate data types. For example, a PLC might use one index group for process data and another index group for configuration data. By using different index groups, it is possible to provide access to different types of data within the same port number. The index group is used in conjunction with the AMSNetID and port number to specify the location of the data that is to be accessed. By using different index groups, it is possible to differentiate between different types of data within the same port number.

The index offset is used to specify the starting point for reading or writing data within a specific index group and port. It is typically expressed as an integer value that represents the byte offset from the beginning of the data blockThe index offset is only applicable for certain data types, such as arrays or structures, where the data is stored in a non-contiguous manner. For simple data types like integers or floats, the index offset is typically not used since the data is stored contiguously in memory.

In this system, ADS communication with Xavier and Industrial PC is done in python language using pyads library. The pyads library is Python wrapper for the TwinCATs ADS library that enables developers to communicate with TwinCAT devices using the Python programming language. The main goal of pyads is to provide a more pythonic way to communicate with TwinCAT devices, making it easier and more intuitive for developers who are familiar with Python to work with TwinCAT systems. With this library, developers performs a variety of tasks, such as reading and writing data to TwinCAT variables, invoking TwinCAT functions, and handling events. One of the key advantages of pyads is its ease of use. The library comes with a straightforward API that abstracts away the complexities of the TwinCAT ADS protocol, making it easy for developers to get started with minimal effort. In addition, the library is well- documented and has a growing community of users for development. pyads is an excellent tool for developers who need to communicate with TwinCAT devices using Python. Its ease of use, versatility, and community support make it a top choice for anyone who needs to work with

TwinCAT systems. The basic python code for establishing communication between TwinCAT devices is shown below.

import **pyads**

**>>>** *# create some constants for connection*

**>>>** CLIENT\_NETID = "192.168.1.10.1.1"

**>>>** CLIENT\_IP = "192.168.1.10"

**>>>** TARGET\_IP = "192.168.1.11"

**>>>** TARGET\_USERNAME = "Administrator"

**>>>** TARGET\_PASSWORD = "1"

**>>>** ROUTE\_NAME = "route-to-my-plc"

**>>>** *# add a new route to the target plc*

**>>>** pyads.add\_route\_to\_plc(

**>>>** CLIENT\_NETID, CLIENT\_IP, TARGET\_IP, TARGET\_USERNAME, TARGET\_PASSWORD,

**>>>** route\_name=ROUTE\_NAME

**>>>** )

**>>>** *# connect to plc and open connection*

**>>>** *# route is added automatically to client on Linux, on Windows use the TwinCAT router*

**>>>** plc = pyads.Connection('127.0.0.1.1.1', pyads.PORT\_SPS1)

**>>>** plc.open()

**>>>** *# check the connection state*

**>>>** plc.read\_state() (0, 5)

**>>>** *# read int value by name*

**>>>** i = plc.read\_by\_name("GVL.int\_val")

**>>>** *# write int value by name*

**>>>** plc.write\_by\_name("GVL.real\_val", 42.0)

**>>>** *# create a symbol that automatically updates to the plc value*

**>>>** real\_val = plc.get\_symbol("GVL.real\_val", auto\_update=**True**)

**>>>** print(real\_val.value) 42.0

**>>>** real\_val.value = 5.0

**>>>** print(plc.read\_by\_name("GVL.real\_val")) 5.0

**>>>** *# close connection*

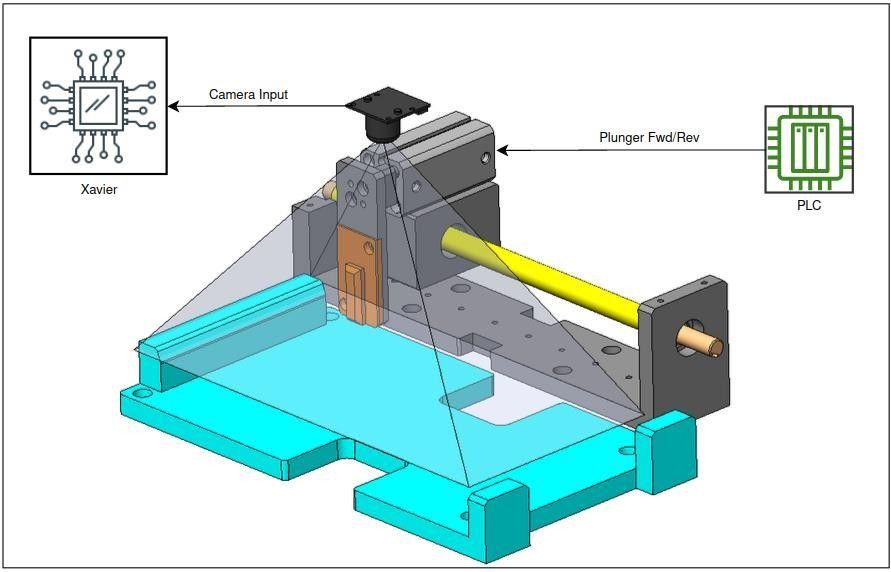
**>>>** plc.close()

## Camera configuration

The camera used in the system has high-resolution with 8 mega pixels and a wide 200-degree field of view. It is equipped with an IMX 219 sensor, which is known for its high-quality imaging capabilities. The camera as shown in Figure 3.20 can capture images with a resolution of 3280\*2464, which is quite impressive. Additionally, it comes with a 1/4 inch lens, a 2.0 aperture, and a 0.87 focal length, which allows it to capture clear and detailed images with excellent depth of field. One of the notable features of this camera is its 3.3V power output, which makes it easy to integrate with other components of the system. It is also worth noting that this camera has high versatility and is used in a wide range of applications including surveillance, robotics, and machine vision. The camera used in the system is a high quality imaging device that can capture clear and detailed images with excellent depth of field. Its wide field of view, high resolution, and 3.3V power output make it an ideal choice for a variety of applications which is integrated with xavier. In this application, a camera is integrated with the GStreamer framework to build a live streaming system for an ongoing process. The camera captures the video data of the process and sends it to the GStreamer pipeline for processing and streaming. The GStreamer framework provides a set of plugins and components that can be used to interface with the camera, capture data from it, process the data, and stream it over a network. The GStreamer pipeline is constructed using the GStreamer domain-specific language and includes various plugins and elements that perform specific tasks such as video encoding, network streaming, and image processing. The pipeline can be customized to suit the specific requirements of the application, such as adjusting the video quality, frame rate, or streaming protocol. Once the pipeline is set up, the video data is processed and streamed in real-time to a remote display or recording device. This enables the users to monitor the ongoing process remotely and in real-time, without the need for physical presence at the location of the process. The integration of the camera with the GStreamer framework enables the building of a live streaming system for ongoing processes, providing users with a powerful tool for remote monitoring and analysis. This system uses YOLOv5 algorithm ito detect the logo and ON screen of the mobile device.



## Figure 3.20 Camera



**Figure 3.21 Video processing system**

This is a system where a mobile device is mounted on a puck, which is then streamed by a CSI camera. The YOLOv5 algorithm running on an Nvidia Xavier is used to detect a logo on the mobile device's screen. If the logo is detected, the bit corresponding to the detection is stored in the Xavier's memory and shared with a PLC memory block using the shared memory concept. For the communication between the PLC and Python code of the algorithm, the Automation Device Specification (ADS) protocol is used. The PLC is configured with the Xavier using the pyADS Python library, and a flag assigned to a particular memory is updated once the logo is detected. The PLC signal then actuates a plunger to turn on the mobile device by releasing the power button. After logo detection, the CSI camera captures the ON Screen of the mobile device, and the captured image is stored in another memory bit of the Xavier this integration is shown in Figure 3.21. This value is again shared with the PLC memory block and updated in the PLC register. Finally, a pneumatic stopper holding the mobile puck under the station is released by receiving a PLC signal, and the

process status of the mobile device is changed from cosmetic grade completion to mobile power on completion.

This system uses advanced technologies such as CSI cameras, YOLOv5 algorithms, Nvidia Xavier, and ADS protocol to automatically detect logos on mobile devices, power them on, and capture their ON Screen images. This provides a powerful tool for streamlining processes and reducing manual labor. Mobile device in the puck is streamed by CSI camera mounted on the setup. As logo is detected by the YOLOv5 algorithm which is deployed in Nvidia Xavier, that particular bit is stored in its memory. This value of memory bit is shared with PLC memory block using shared memory concept. This requires proper communication between PLC and python code which is achieved by ADS (Automation Device Specification) protocol. For establishing communication between PLC and Python code of algorithm PLC is initially configured with Xavier using pyADS python library. Thereafter, PLC flag assigned to particular memory which is shared between two systems (PLC and Xaiver) is updated and plunger which is actuated by PLC signal for turning on device is pushed in reverse direction. Therefore, device power button is released after logo detection on the screen. Consecutively, mobile ON Screen is captured by CSI camera and it is stored in another bit of memory in Xavier. Thereafter value of that particular memory bit is again shared with PLC memory block and is updated in PLC register. Lastly, Pneumatic stopper which holds mobile puck under the station is released by receiving PLC signal and process status of mobile device is changed from cosmetic grade completion to mobile power on completion.

|  |  |
| --- | --- |
| Hardware Details | Specification |
| Nvidia Xavier | CPU: 8-core Nvidia Carmel, Storage: 64 GB |
| CSI Camera : waveshare | Megapixel: 8, Field of view: 200, Sensor: IMX2 |

Table 3.4: Hardware used in video processing

## Summary

The use of advanced technologies in the manufacturing industry has been a game-changer in terms of increasing efficiency and productivity. This chapter explored the use of YOLOv5 for object detection and industrial PC for controlling output in mobile testing, which has proven to be a highly effective solution. YOLOv5, a deep learning model, has gained popularity in recent years due to its remarkable accuracy in identifying objects in images. By leveraging this technology, manufacturers can automate the process of identifying and checking the power on condition of mobile devices. The integration of industrial PC provides a robust solution for controlling the output, ensuring that the system runs smoothly and efficiently. Moreover, chapter discussed the integration of Xavier, a powerful computing platform, which has enhanced the overall performance of the system. Xavier's capability to process a vast amount of data in real-time has been a game-changer for the manufacturing industry. The integration of these advanced technologies has provided a streamlined and efficient solution for mobile testing.

By automating the power on condition check, manufacturers can significantly reduce the need for manual intervention, minimizing the risk of errors and improving overall product quality. This system's accuracy and precision enable manufacturers to produce high-quality mobile devices that meet customer demands.Overall, the integration of YOLOv5 and industrial PC with Xavier integration has revolutionized mobile testing in the manufacturing industry. Manufacturers or mobile testing vendors can now automate the process of power on condition check, streamline the testing process, and significantly improve the quality of their products. As technology continues to advance, we can expect to see further advancements in mobile testing, ultimately leading to improved customer satisfaction and increased profits for manufacturers.

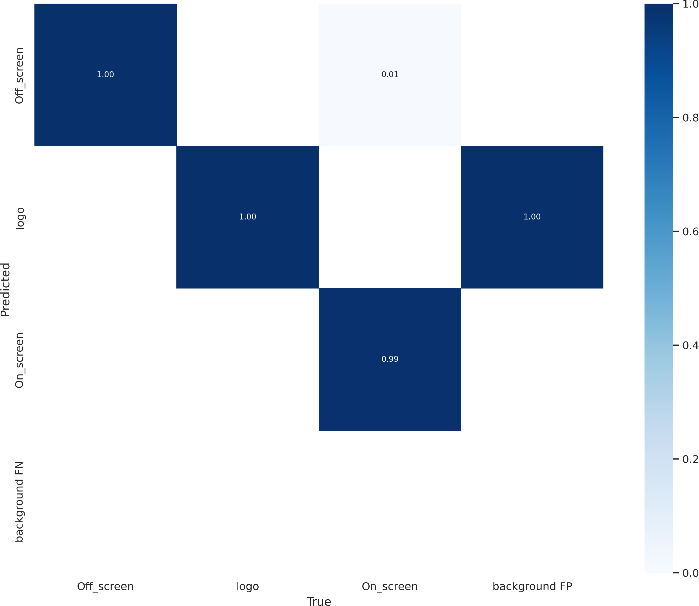
**Chapter 4**

# Results and Discussions

## Obtained result of trained model

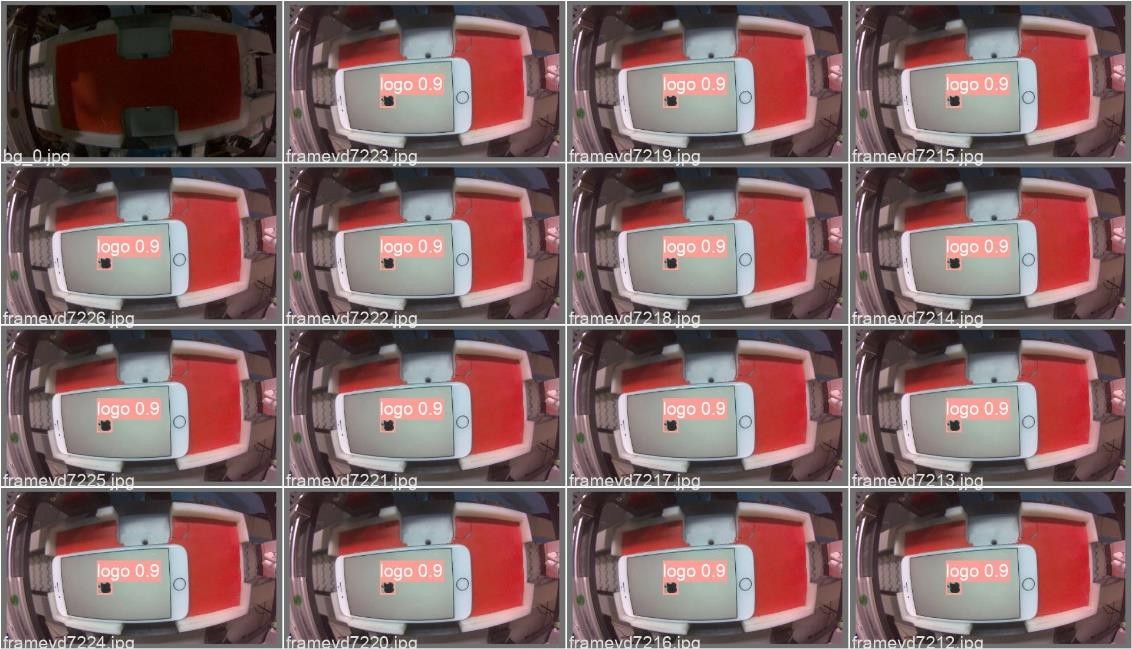
The experimentation conducted to obtain images of the mobile device and evaluate the YOLOv5 algorithm's performance in detecting the logo on the device's screen. The confusion matrix obtained from the model's predictions is also analyzed to determine its accuracy. For experimentation purposes, live streaming of the mobile device was captured under the camera in the mobile puck. Images of each frame were obtained to collect a dataset for the YOLOv5 algorithm to train and validate. The dataset is divided into training and testing sets, with a 70-30 split ratio. After training and validating the model, the confusion matrix of the model's predictions was obtained and is shown in Figure 4.1. The confusion matrix represents the predicted and actual classes of the model. As per obtained confusion matrix, the model correctly predicts off-screen with 100% probability, which is a true positive (TP). Similarly, for the logo class, the predictions of the model are 100% correct, without any false negative (FN) and false positive (FP) value. For the on-screen class, the model shows a probability of 99% for true positive values and 1% for false positive value.

The confusion matrix's accuracy can be determined by calculating different evaluation metrics such as precision, recall, and F1 score. In this case, the YOLOv5 algorithm achieved high precision, recall, and F1 score, indicating that the model's predictions are highly accurate. The model correctly detects the logo on the mobile device's screen, without any false negative or false positive predictions. The deployed model's predictions on the actual system are shown in Figure 4.2. The model detects the logo on the mobile device's screen and triggers the plunger to turn on the device. The ON Screen of the mobile device is then captured, and the image is stored in the Xavier's memory for further processing. For experimentation purposes, images of the mobile device were obtained by capturing images of each frame during live streaming of the device under the camera in the mobile puck. Obtained confusion matrix of the model is represented in Figure 4.1



## Figure 4.1: Confusion Matrix

The following Figure 4.2 is the result obtained by the system where mobile is being processed on the conveyor line. The model has successfully detected the logo on the mobile screen the captures which are taken by the camera which is mounted in the system integrated to xavier and is streamed on monitor connected remotely with a batch size of 16.



## Figure 4.2: Logo detection on Mobile device

* 1. **Validation of model**

A p curve of a model is a graphical representation of the distribution of p values resulting from hypothesis tests applied to a set of statistical models. Specifically, it shows the distribution of p- values for the null hypothesis that model has no effect. A pression curve is used to evaluate the validity of a model and to check whether the results of the hypothesis tests are consistent with the assumptions of the model. If the p-curve shows a distribution of p-values that is uniform or skewed to the right (i.e., many p-values are very small), this suggests that the model is valid and that the results are consistent with the nule hypotheses. On other hand, if the p-curve shows a distribution of p-values that is skewed to the left (i.e., many p-values are very large), this suggests that the model may not be valid and that outcomes are not consistent with the null hypothesis. In this case, further investigation may be needed to determine the source of the problem and to correct it. The P curve is a graphical representation of the relationship between precision and confidence level of a classifier's predictions for each class. The precision of a classifier is a ratio of true positives among all positive predictions. The confidence level of a classifier is a measure of how sure the classifier is about its predictions. The obtained P curve shows that the classifier is well trained and accurate when it makes predictions with a confidence level of around 80%. When the classifier makes predictions in this range, the precision is 100%, which means that all the positive predictions are true positives.

This indicates that the classifier is highly confident and accurate in making predictions for this range of confidence values. However, when the confidence level is 0%, the precision drops to near about 30%, which suggests that the classifier is not as accurate or confident when making predictions with very low confidence values. This could be due to various reasons, lack of sufficient training data or the classifier's inability to accurately capture the patterns or features that are present in the data. The P curve provides valuable insights into the classifier's performance, and it is essential to carefully analyze and understand the performance metrics to draw meaningful conclusions about the classifier's effectiveness.



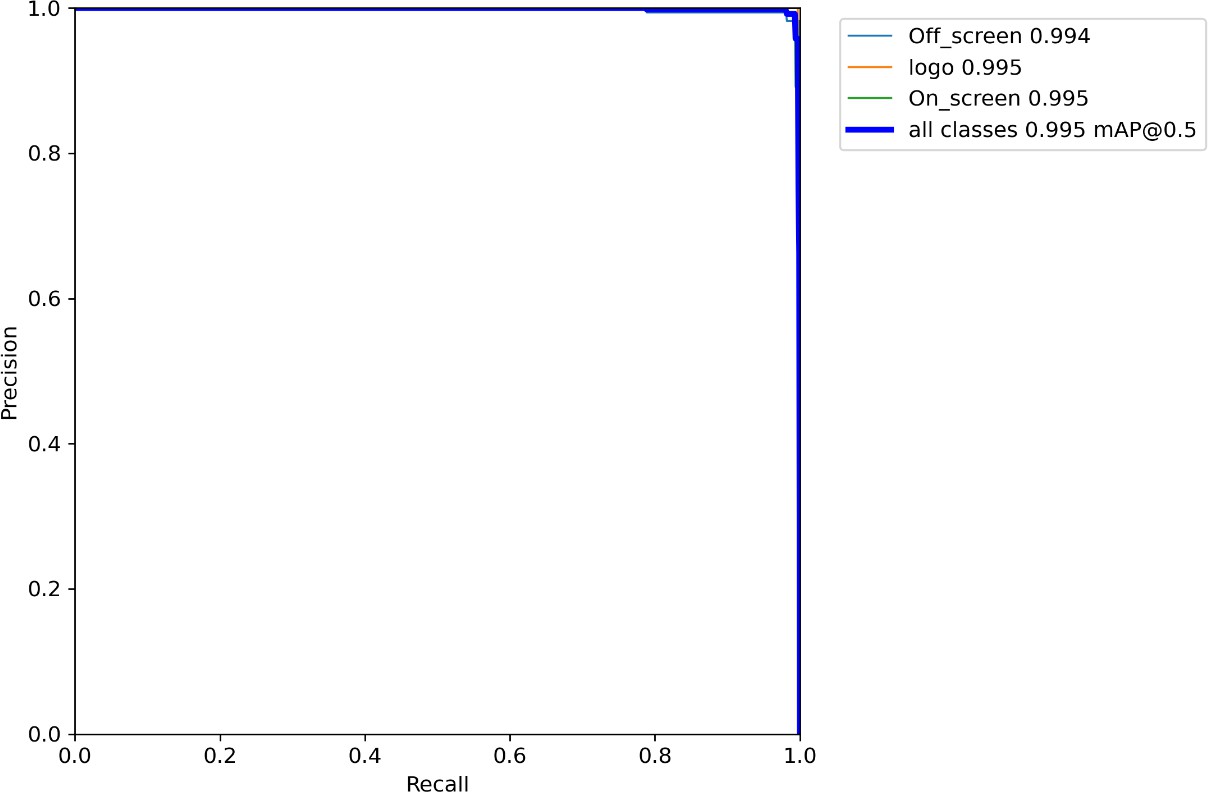
## Figure 4.3 P curve

* 1. **Analysis of PR curve**

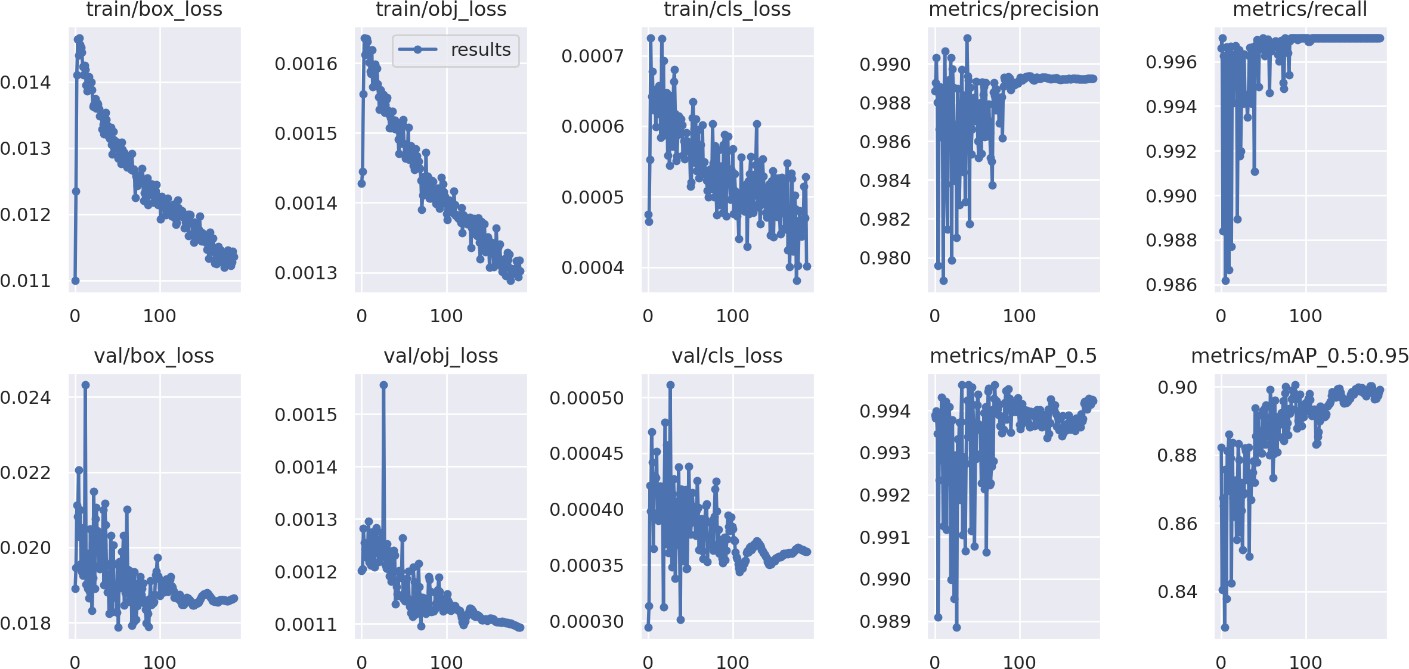
Similarly, PR (Precision-Recall) curve presented in Figure 4.4 is a graphical representation of the precision and recall values of a classifier as a function of decision threshold. It is often used in binary classification problems where the classifier's output is either positive or negative.

The PR curve shows how the precision and recall values change as the decision threshold for classification is varied. The curve is plotted by varying the decision threshold from high to low and calculating the corresponding precision and recall values for each threshold. The resulting curve represents a trade-off between precision and recall, and it can help to choose the optimal threshold value for the classifier. A good PR curve shows high precision and high recall values for a range of decision thresholds. A perfect classifier will have a PR curve that goes straight up to a precision of 1 at a recall of 1. A poor classifier, on the other hand, will have a PR curve that closely resembles a diagonal line, indicating that the precision and recall values are essentially random.

PR curve provides a useful visualization of a classifier's performance and can be used to compare the performance of different classifiers. The resulted PR curve shows a perfect classifier as it goes straight up to a precision value of 1 at recall of 1.



## Figure 4.4 PR curve



**Figure 4.5 Model training performance**

In machine learning, training a model involves iteratively adjusting the model parameters to minimize a chosen loss function which is shown in Figure 4.5. The loss function measures difference between predicted output and the actual output. The aim of training a model is to minimize this loss function and improve model accuracy. During training, it is common to monitor the loss function and other evaluation metrics, such as precision, recall, and mean average precision. The obtained graphs of these metrics during training can provide valuable insights into the performance of the model. One common trend observed in these graphs is that as the number of training epochs increases, the loss function decreases gradually. This is because the model is continuously updated to better approximate the true relationship between the input and output data. As the model trains the predicted outputs become closer to the actual outputs, resulting in a decrease in the loss function. The decrease in loss function indicates that the model is learning from the training data and improving its performance. However minimizing loss function on the training data may not necessarily result in good performance on unseen data. This is because the model may overfit to the training data, i.e., memorizing the training data instead of learning to generalize to new data. To avoid overfitting, it is important to monitor the evaluation metrics during training process. These metrics provide a more meaningful evaluation of model performance than loss function alone.

As the model improves during training, it becomes better at identifying all actual positive samples, resulting in an increase in recall over time. Mean average precision (mAP) which is a commonly used evaluation metric for object detection models. It measures the precision and recall at different confidence thresholds and computes the area under the precision-recall curve. Also it becomes better at detecting objects with high precision and recall at different confidence thresholds, resulting in an increase in mAP over time. The obtained graphs are which are related to loss describes that as the model is being trained on higher epochs the loss related to each metric is decreasing gradually. Similarly on other hand metrics precision, recall and mean average precision is increasing gradually as the number of epochs is increasing.

## Summary

The thesis result summary discusses the obtained results of a model that was trained to detect logos and power on conditions in a developed system. The performance of the model was evaluated using P curve and PR curve, which showed that the model was well trained for detecting the logo and power on condition in the developed system. The results demonstrated that the model had a high level of accuracy, even at low confidence levels. The precision of the model was found to be very high, which indicates that it could accurately identify the target objects in the system. These findings are significant because they demonstrate the effectiveness of the model for detecting logos and power on conditions in the mobile testing industry. The chapter concludes that the model has the potential to be a powerful tool for automating the mobile testing process, resulting in increased efficiency and accuracy. The model's effectiveness for detecting logos and power on conditions could provide significant benefits to the manufacturing industry, streamlining the mobile testing process and improving overall product quality. Overall, the thesis result summary suggests that the model has significant potential for application in the manufacturing industry. The model's high level of accuracy and precision makes it an effective tool for detecting logos and power on conditions in the developed system, providing a powerful solution for automating the mobile testing process. The obtained results of the model indicate that it has the potential to revolutionize the mobile testing industry, resulting in increased efficiency, productivity, and accuracy.

**Chapter 5**

# Conclusion

## Process status extraction

In this thesis experiment, the first objective was to design a system that could acquire the process status of a mobile device by scanning a QR code and verifying it with the database. The findings of the experiment revealed that the developed system was capable of effectively obtaining the process status of the device using the smart code reader integrated into the system. The utilization of the QR code and database check mechanism has considerably enhanced the accuracy and efficiency of the process for obtaining the process status of the mobile device. Overall, the successful implementation of this system has significant implications for the mobile device testing industry. It provides a practical solution that can enhance the overall quality control process for mobile devices. By making the process of obtaining the process status of a mobile device more accurate and efficient, the system can increase productivity, reduce errors, and minimize delays in the testing process. The experiment's findings demonstrate the potential of this system to optimize the mobile device testing process and can serve as a foundation for further research in this area.

## Automating process of Powering on mobile device

After knowing process status of mobile device the second objective of this thesis was to automate and integrate the process of powering on a mobile device in the puck. The experiment aimed to achieve this by integrating an industrial PC and pneumatic system to automate the process of powering on the mobile device. Additionally, the experiment also aimed to establish communication between the industrial PC and Xavier, to enable the sharing of data through shared memory. The results of the experiment showed that the integration of an industrial PC and pneumatic system was successful in automating the process of powering on the mobile device in the puck. The integration of these technologies enabled the system to accurately detect the power on condition of the mobile device and effectively power it on. The use of shared memory also facilitated communication between the industrial PC and Xavier, which allowed for the sharing of data. The successful implementation of this system has significant implications for the mobile device testing industry. It provides a practical solution that can enhance the overall

quality control process for mobile devices. By automating the process of powering on the mobile device, the system reduces the need for manual intervention, minimizes the risk of errors, and improves overall product quality. Furthermore, the use of shared memory allows for efficient and real-time communication between the industrial PC and Xavier, which can improve the accuracy and efficiency of the system. In conclusion, the experiment successfully achieved the second objective of the thesis, which was to automate and integrate the process of powering on a mobile device in the puck. The integration of an industrial PC and pneumatic system, as well as the use of shared memory, proved to be effective in achieving this objective. The system has significant potential for application in the manufacturing industry, and the findings of this experiment can serve as a foundation for further research in this area.

## Device power on confirmation

In conclusion, the experimentation conducted to obtain images of the mobile device and evaluate the YOLOv5 algorithm's performance in detecting the logo on the device's screen was successful. The confusion matrix obtained from the model's predictions shows high accuracy, and the deployed model's predictions on the actual system are reliable. This system provides an automated solution for detecting logos on mobile devices and streamlining processes, reducing manual labor, and increasing efficiency.

A logo and screen-on detection system while powering on mobile has been successfully developed. The results show that precision achieved by system is around 99%. There is a scope for development in machine vision. At present this system is not optimized for the detection of logos of all mobile manufacturing brands. The obtained graphs of loss and evaluation metrics provide valuable insights into performance of model during training. Decreasing loss indicates that the model is able to learn by training data and is improving its performance. Increasing precision, recall, and mAP indicates that the model is becoming better at predicting true positives, detecting all actual positives, and detecting objects with high precision and recall at different confidence thresholds, respectively. However, it is important to avoid overfitting and ensure that the model can generalize to unseen data by monitoring the evaluation metrics during training.

The high precision of around 99% achieved by the system in logo detection is commendable. However, there is room for improvement, especially in the detection of logos of all mobile

manufacturing brands. This could be achieved through the use of more diverse and representative train data, which could improve the model ability to generaleize to different types of logos. Additionally, techniques such as transfer learning could be used to leverage pre trained models on large datasets and fine tune them on the specific task of logo detection. The graphs of loss and evaluation metrics provide valuable insights into the performance of the model during training. The decrease in loss indicates that model is learning and improving its performance over time. However, it is important to monitor the evaluation metrics during training to avoid overfitting, where the model become too specialized to the training data and shows poor performance on unseen data. One way to prevent overfitting is through the use of regularization techniques such as dropout and weight decay which help to prevent the model from memorizing the training data. In the developed automation system for logo detection on mobile screens and screen-on detection while powering on mobile is a promising solution for mobile device production and testing. The system's high precision, reliability, scalability, and adaptability make it an excellent choice for streamlining processes, reducing manual labor, and improving efficiency. With further improvements in machine vision, the system has the potential to become even more accurate and versatile in the detection of logos of all mobile manufacturing brands.

## Future directions

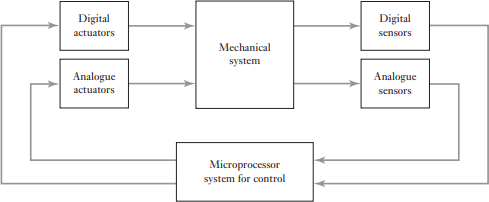
The automation system for logo detection on mobile screens and screen-on detection while powering on mobile has several advantages. The system eliminates the need for human intervention, reducing the risk of human error and improving efficiency. The system's accuracy and reliability ensure that only functioning mobile devices are processed, reducing the likelihood of defective units reaching the market. The system is also scalable and adaptable to different mobile devices and manufacturing brands, making it a versatile solution for mobile device production and testing.

**Chapter 6**

**Internship Report**

* 1. **Mechatronics system**

A mechatronic system isn't just a combination of electrical and mechanical systems or a control system. It's a complete integration of all these components, designed concurrently. This approach to engineering design is becoming more prevalent in the design of cars, robots, washing machines, cameras, and many other machines. To develop cheaper, more reliable, and more flexible systems, integration across traditional boundaries like mechanical engineering, electrical engineering, electronics, and control engineering must occur at the earliest stages of the design process. Mechatronics requires a concurrent approach to these disciplines, rather than a sequential approach where one component is developed after the other. Therefore, mechatronics is an integrating approach to engineering and a design philosophy. It brings together sensor and measurement systems, drive and actuation systems, microprocessor systems, and the analysis of system behavior and control systems.



## Figure 6.1: Mechatronics system

The design process for any system involves several stages:

1. Identification of the Need:

The process begins with identifying the need, which could come from a customer or client. Market research can be conducted to understand the needs of potential customers.

1. Analysis of the Problem:

The first step in developing a design is to understand the nature of the problem by analyzing it. Failing to define the problem accurately can result in wasted time on designs that won't meet the need.

1. Preparation of a Specification:

After analyzing the problem, a specification of the requirements can be prepared. This should include the problem statement, any constraints placed on the solution, and the criteria used to evaluate the design. All functions required of the design and desirable features should be specified, such as mass, dimensions, range of motion, accuracy, input/output requirements, interfaces, power requirements, operating environment, standards, and codes of practice.

1. Generation of Possible Solutions:

Possible solutions are developed in sufficient detail to indicate the means of obtaining each required function, including approximate sizes, shapes, materials, and costs. Prior solutions for similar problems should also be considered.

1. Selection of a Suitable Solution:

The various solutions are evaluated, and the most suitable one is selected. This often involves modeling the system and simulating its response to inputs.

1. Production of a Detailed Design:

The details of the selected design are worked out, which may require the production of prototypes or mock-ups to optimize the design.

1. Production of Working Drawings:

The selected design is translated into working drawings, circuit diagrams, etc., so that the item can be manufactured.

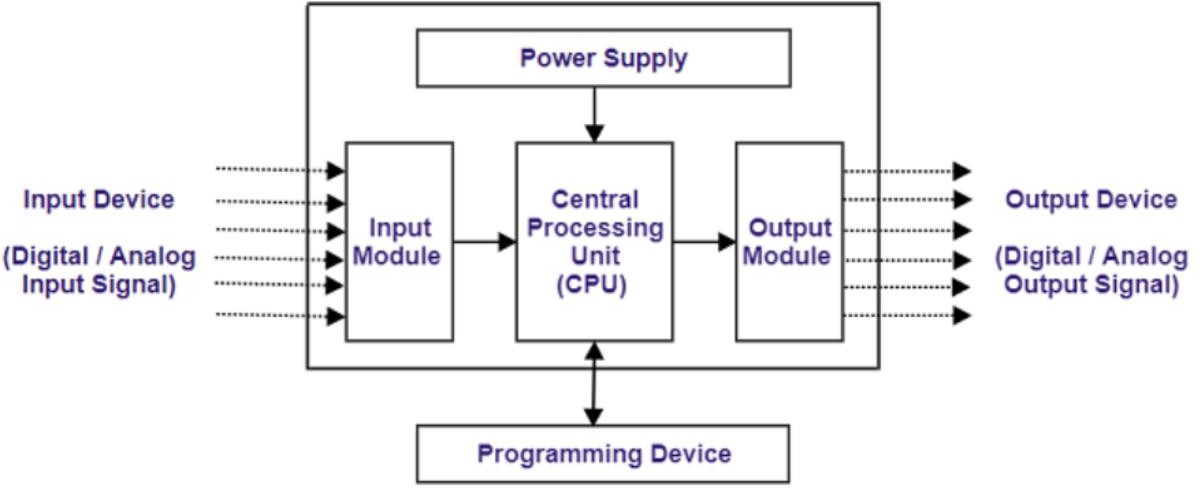
## PLC Programming Introduction

A Programmable Logic Controller (PLC) is an electronic device that automates controls, interlocks operations, and sequences tasks in various industries. It is commonly used in industries with a need for process safety and interlocks, such as process industries. The device operates digitally, using programmable memory to store instructions for logic sequencing, timing, counting, and control of digital or analog input-output modules. PLCs have become an essential component of modern automation systems, with applications in t automotive, food and beverage, packaging, and pharmaceutical industries, among others. They are built to withstand harsh industrial conditions and can be easily programmed to perform different tasks. With the ability to communicate with other devices and systems, PLCs are instrumental in controlling complex manufacturing processes, reducing downtime, and enhancing overall efficiency.

Basic components of PLC

1. Input module
2. CPU
3. Memory
4. Programming device
5. Output module

Input modules are an integral part of Programmable Logic Controllers (PLCs). They receive real-world signals, providing the controller with a dual-time status of variables. These variables can be either analog or digital. Analog inputs can be processed by devices such as limit switches, transducers, RTDs, flow and pressure sensors, and strain gauges. Once the analog inputs are processed, they are converted into binary and transmitted over the input or output bus to the CPU.



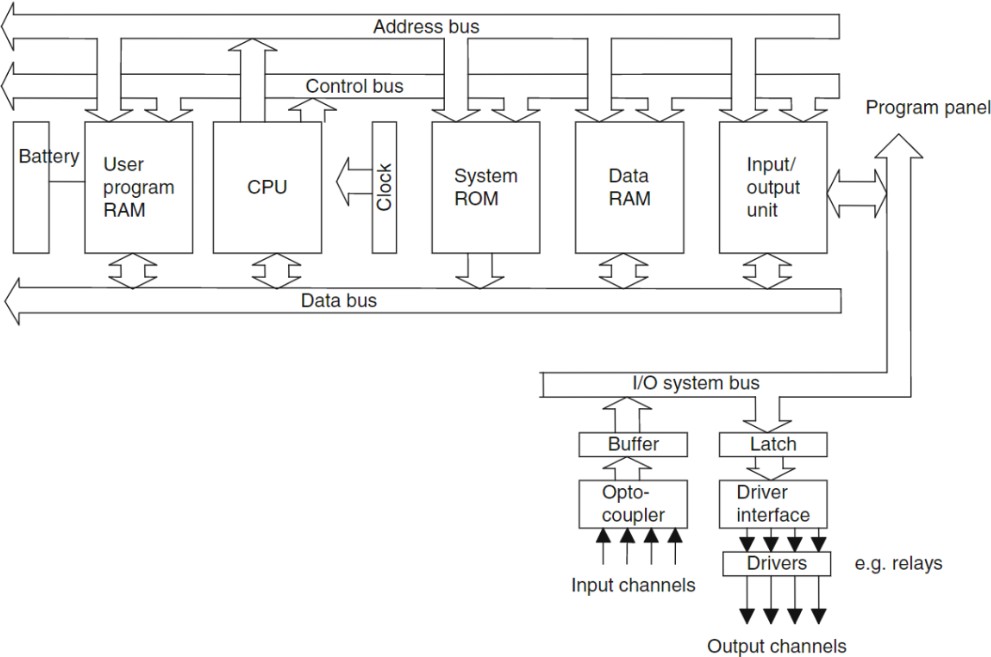
## Figure 6.2.1 PLC architecture

Each input module has a certain number of channels per module, and each channel is equipped with an indicator light to show if the particular input is on or off. The CPU is the brain of the PLC, and it organizes all the controller activity. It electronically scans the control plane logic stored in the memory, along with the status of the inputs. Then, it executes a specified command to appropriate output. In addition to the logic processing, the CPU performs other functions such as timing, counting, launching, comparing, retentive storage, and communication. The word size of the PLC varies from 4 bits to 16 bits. The program to be executed is stored in the memory. The basic memory unit is world memory, and the size varies from 256 words to 192 K words. The complexity of the control plan determines the amount of memory required. In most PLCs, memories are expandable in fixed increments. Memory can be volatile or non-volatile. Volatile memory is lost when power is removed. The three basic types of memories are read-only memory, rewrite memory, and random-access memory. Programming devices are used to communicate between the user and PLC. A programming language is used to convey the PLC by means of instructions on how to execute and carry out the control plan. PLCs are programmed with ladder logic, which is a graphical method of programming. Output modules convert the desired real output as per the instructions stored in the CPU memory signals are used to control the machine or process output device can be motor solenoid lights digital outputs can price panel meters or display input output systems modular in nature this the most important function of io is ability to isolate the real world signals from the low level signals this is accomplished by optical isolator

Input modules, CPUs, memory, and programming devices are crucial components of PLCs. Input modules receive real-world signals, providing the controller with a dual-time status of variables. The CPU organizes all the controller activity and performs various functions such as timing, counting, and communication. Memory stores the program to be executed, and programming devices are used to communicate between the user and the PLC. Understanding these components and their functions is essential for the effective use of PLCs in process industries.

PLC Internal Architecture:

The Central Processing Unit (CPU) is responsible for controlling and processing all operations within the PLC. It is supplied with a clock that typically operates at a frequency of one to eight megahertz, which determines the operating speed of the PLC and provides the timing and synchronization for all elements within the device. The information within the PLC is carried by means of digital signals. The processor is a microprocessor that executes a program to perform operations specified in a ladder diagram or a set of Boolean equations.



## Figure 6.2.2 Internal architecture of PLC

The Arithmetic and Logical Unit (ALU) performs data manipulation and arithmetic and logical operations on input variable data and determines the proper state of output variables. The arithmetic operations include addition and subtraction, while the logical operations include AND, OR, and exclusive OR. The Control Unit (CU) is used to control the timing and operation of the processor. The processor functions under a permanent supervisory operating system that directs the overall operation, from data input and output to the execution of the user program. The controller can perform only one operation at a time, so it scans each input sequentially, evaluates the ladder logic, provides each output, and then repeats the whole process.

Memory unit

PLC programming involves creating a sequence of instructions to be executed by the CPU. These programs are stored in the memory units during the entering, editing, and debugging process. The program is stored in the temporary storage called RAM, which is volatile memory that requires a constant power supply. For network programming of PLCs, the final program is downloaded into a special reprogrammable ROM in the PLC. The memory may be either volatile or non-volatile. Non-volatile memory, such as ROM or flash memory, retains data even when the power supply is disconnected. Once the program is downloaded into the PLC's memory, it can be executed repeatedly by the CPU. The PLC scans each input sequentially, evaluates the ladder logic, provides each output, and then repeats the whole process. The program can be modified, and the modified version can be stored in the PLC's memory for execution. PLC programming devices are used to communicate between the user and the PLC. Programming languages are used to convey instructions to the PLC on how to execute and carry out the control plan. PLCs are programmed with ladder logic, which is a graphical method of programming. The ladder logic language is based on Boolean logic, which is used to create logical statements that determine the state of the outputs based on the inputs.

Buses Introduction:

A bus is a set of parallel lines that facilitates communication between different devices in a system. In a programmable logic controller (PLC), the bus system plays a critical role in

transmitting information and data to and from the CPU, memory, and input/output units. This report discusses the different types of buses in a PLC system, their functions, and how they transmit data.

Types of Buses:

A PLC system consists of four primary buses: data bus, address bus, control bus, and system bus. The data bus contains 8, 16, or 32 parallel signal lines that transmit data between different devices in the system. The data is transmitted in binary form, as zero or one digital signals, or electrical signals that flow inside the bus. The address bus is a unidirectional bus that carries memory and input/output addresses between devices. The control bus is responsible for transmitting control signals between the CPU and other devices, such as timing signals and interrupt signals. The system bus is a combination of the data bus, address bus, and control bus.

Function of Buses:

The data bus facilitates communication between different devices by carrying data to and from the CPU, memory, and input/output units. An 8-bit microprocessor typically handles 8- bit numbers, which can be transmitted through the data bus. The double-ended arrows on the bus line indicate that the data bus is bidirectional, which means that the CPU can read data from memory or input/output units on this line or send data out to memory or input/output units. The address bus is responsible for carrying memory and input/output addresses between different devices in the system. The control bus carries control signals between the CPU and other devices, such as timing signals and interrupt signals. The system bus is responsible for carrying data, address, and control signals between different devices in the system.

Data Transmission:

In a PLC system, many devices may have their outputs connected to the data bus. However, only one device will have its output enabled at any given time. This is to prevent multiple devices from transmitting data simultaneously, which can cause data collisions and communication errors. The data is transmitted in binary form as digital signals or electrical signals that flow inside the bus. An address bus is a set of parallel signal lines that carry memory location addresses, enabling the CPU to access stored data. These memory locations are assigned unique addresses to facilitate easy location and retrieval by the CPU for reading

or writing data. The number of signal lines in an address bus can range from 16 to 32. The control bus, on the other hand, consists of 4 to 10 parallel signal lines that transmit signals related to internal control actions of the CPU. These signals are used by the CPU to manage data transfers, interrupt operations, and perform other internal control functions. The input- output unit provides an interface between the computer system and the outside world, allowing for connections to be made through input-output channels to input-output devices. These devices can include peripherals such as printers, scanners, keyboards, and displays. Programs can be entered into the computer system from a program panel through the input- output unit. The address bus carries memory location addresses, the control bus transmits signals for internal control actions, and the input-output unit provides an interface between the computer system and the outside world.

Ladder diagram

A ladder diagram is a graphical programming method used for representing an event-driven sequential process. It is a special schematic representation of hardware elements and their connections used to combine hardware and describe the sequence of events. The ladder diagram consists of two parallel lines indicating the AC supply lines, and several horizontal lines connecting these parallel lines. These horizontal lines define specific operations and are called rungs. The entire structure looks like a ladder, hence its name. In ladder diagrams, logic functions are represented by symbols such as contacts, coils, timers, and counters. The contacts can be either normally open or normally closed, and they represent the status of an input signal. The coils are output devices that are energized or de-energized by the logic function.The sequence of operation in a ladder diagram is determined by the order of the rungs. The logic functions in each rung are evaluated from left to right, and from top to bottom. The output of each rung is connected to the input of the next rung, and the sequence of rungs determines the overall operation of the process. Ladder diagrams are commonly used in industrial automation and control systems, and they provide an intuitive and easy-to- understand method for programming sequential processes. They are also used in electrical engineering for designing and troubleshooting electrical circuits.

Ladder Programming

Ladder logic programming is a method of representing and programming control systems by creating a circuit diagram, known as a ladder diagram. The ladder diagram is a systematic way of representing the system hardware and controller, consisting of two vertical lines representing the power lines, with circuit devices connected horizontally across the AC power. The ladder diagram is composed of rungs, which consist of input conditions and signal command outputs. Each rung of the ladder diagram defines one operation in the control process, and the diagram must be read from left to right and top to bottom. In run mode, the programmable logic controller (PLC) goes through the entire ladder program until it reaches the end and then starts over again, which is called a cycle. Each rung in the ladder diagram starts with at least one input and one output. Electrical devices are shown in their normal condition, either open or closed contacts. A particular device can appear in more than one rung in the ladder diagram, and some identification number is used to identify the device in each situation. All inputs and outputs are identified by their addresses, and the notation used depends on the PLC manufacturer. Ladder logic programming is widely used in industrial control systems and provides a visual and intuitive way of programming complex control processes.

Counters:

A counter is a device that records the number of times an input event occurs. There are two basic types of counters: count up and countdown. Every programmable logic controller (PLC) has counter instructions. A countdown counter counts down from the present value to zero, while an up counter counts from zero up to the present value. The up counter is commonly used to count the true or false status of an input instruction and trigger an event after a required number of counts is reached. Counters are used in various applications, such as in manufacturing processes, where they count the number of items produced, or in traffic management systems, where they count the number of vehicles passing through a specific location. They are also used in building automation systems to monitor the usage of specific resources, such as electricity, water, or gas. counters are a simple and effective way to monitor and control events, and they play a critical role in many different industries and applications

## PLC Logic Implementation

Problem statement

To maintain a capacity of maximum 120 mobile phones in a storage compartment with two doors, one for loading and one for unloading, a programmable logic controller (PLC) can be used to control the operation. There is a sensor installed in the storage compartment to detect the number of phones, the PLC logic is programmed as follows:

1. When the number of phones in the storage compartment is less than 120, the loading door green light is turned ON.
2. When the number of phones in the storage compartment reaches 120, the loading door green light is turned OFF, and the unloading door red light is turned ON, indicating that the compartment has reached its maximum capacity and is full.
3. When the number of phones in the storage compartment is reduced below 120 due to unloading, the unloading door red light is turned OFF, and the loading door green light is turned ON again.

This logic is implemented using ladder diagrams or other programming languages supported by the PLC. By using this logic, the capacity of the storage compartment is effectively maintained, and the system provides real-time feedback to the user through the status of the loading and unloading doors.

Implemented solution.

Considering the availability of 2 separate doors for loading and unloading 2 separately proximity switches are used to detect the loading and unloading mobile devi One proximity switch is mounted at the loading door and another is mounted shut the unloading door both switches generate 2 different outputs which is then fed to PC to operate the lights according to the ladder logic program in its memory counters is used to count the number of mobile phones loading and unloading comparators is also used to compare the count value with the maximum capacity of mobile phones which is stored in compartment.

List of inputs and outputs

I:1/0 = Proximity switch to detect the loading of every mobile phone (Input) I:1/1 = Proximity switch to detect the unloading of every mobile phone (Input) O:1/1 = Red light to indicate the availability in storage compartment (Output) O:1/0 = Green light to indicate the storage compartment maximm capacity (Output) C5:0 = Counter to calculate the number of devices loading (Counter)

C5:0 = Counter to count the number of devices unloading (Counter) LES = Comparator to compare the counter value

Program description

RUG000 - whenever the device loads the storage rack proximity switch 1 generates a pulse which increments counter by 1

RUG001 - Whenever a phone is unloaded from the storage rack proximity switch 2 generates a pulse which decrements counter by 1

RUNG002 - Whenever the number of devices in the storage compartment is less than 120 green light goes on

RUNG003 – Whenever the number of devices in the class is 120 which is maximum capacity of the storage rack , red light goes ON

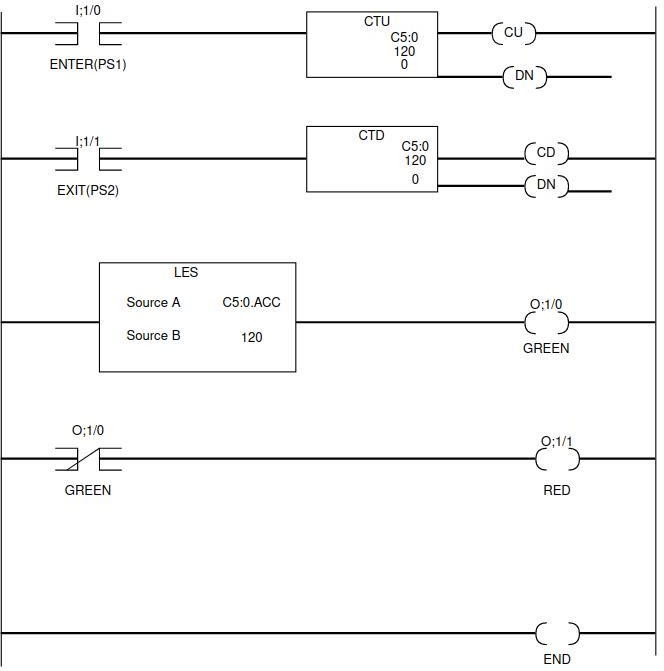
RUNG4 – It terminates the program and the scan cycle is repeated again.

|  |  |
| --- | --- |
| Input | C5:0. ACC |
| PS1 goes low to high | Increment by 1 |
| PS2 goes low to high | Increment by 1 |

**Table 6.3.1 Runtime Test Case**

|  |  |
| --- | --- |
| Comparators | Output |
| Source A(C5:0 .ACC) < Source B(120) | Green = ON , RED = OFF |
| Source A(C5: 0.ACC) = Source B (120) | Green = OFF, Red = ON |
| Source A(C5: 0.ACC) ≥ Source B (120) | C5 :0 , DN bit goes true |

**Table 6.3.2 Input and Output**



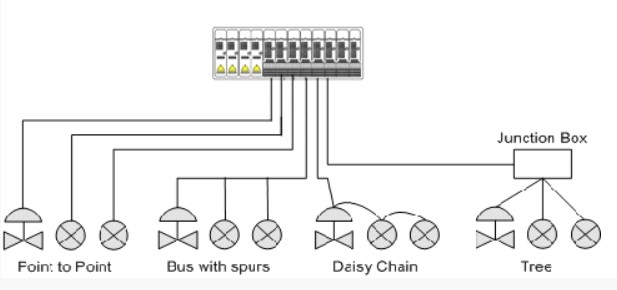
**Figure 6.3.1 Ladder Diagram of problem statement**

* 1. **Communication Protocols**
     1. **Fieldbus**

The Fieldbus Foundation is a non-profit organization that was established in 1994 with the aim of developing an open protocol for use in industrial automation systems. The Foundation is dedicated to reducing end users' reliance on proprietary protocols, providing standards for high reliability, and testing manufacturers' products to verify conformance to its standards. The Foundation's primary focus has been on the development of the FOUNDATION Fieldbus protocols, which are designed for use in distributed control systems (DCS). These protocols comply with the IEC 61158 standard for fieldbus communications and have been widely adopted by OEMs offering compatible transmitters and actuators.

The FOUNDATION Fieldbus H1 protocol is used for communications between a controller and field devices on a network or between field devices. It operates at a speed of 31.25 kbps and supports up to 32 network nodes. The most common arrangement for Fieldbus H1 is a trunkline bus with branches to individual devices. This protocol uses digital communications exclusively, and every communications network requires a fieldbus linking device, a power supply, and terminating resistors. Fieldbus H1 is designed to schedule control activity in, and communications activity between devices. It regularly polls devices for process data and supports twisted-pair wiring that meets certain specifications. The trunk cable requires a power source in the range of 9-32 VDC, with 24 VDC ± 2 VDC recommended for most uses. Individual 2-wire end devices typically draw all their power from the trunk line to which each branch is connected. Trunk cables can be either 3-wire or 4-wire, with DC power wires entirely segregated from signal wires in the latter arrangement.

In addition to the FOUNDATION Fieldbus H1 protocol, the Fieldbus Foundation has also developed the FOUNDATION Fieldbus High Speed Ethernet (HSE) protocol, which is designed for use in high-speed applications. This protocol supports data rates of up to 100 Mbps and uses standard Ethernet hardware, making it easy to integrate into existing Ethernet networks. The Fieldbus Foundation's work has helped to establish open standards for industrial automation systems, reducing end users' reliance on proprietary protocols and promoting interoperability between different manufacturers' products. The adoption of the FOUNDATION Fieldbus protocols has helped to increase the reliability and efficiency of industrial automation systems, improving overall productivity and reducing costs

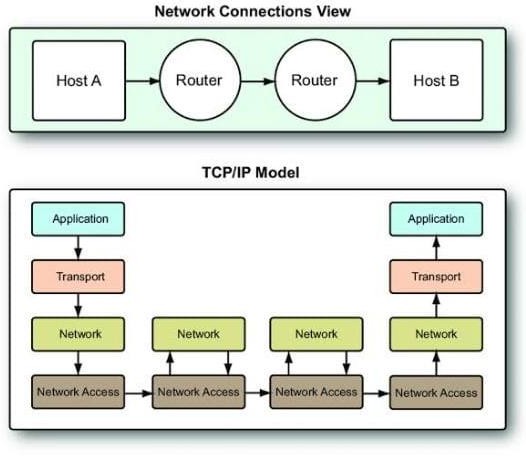


## Figure 6.4.1 Fieldbus architecture

* + 1. **TCP/IP**

TCP/IP is a set of communication protocols that allow different devices to communicate with each other over a network. These protocols are commonly used on the Internet and Ethernet networks. The two main protocols that make up TCP/IP are TCP (Transmission Control Protocol) and IP (Internet Protocol). TCP is responsible for breaking data into small packets, ensuring that each packet is transmitted and received in the correct order, and reassembling the packets at the destination. TCP also provides error checking and flow control, which help ensure that data is transmitted reliably and efficiently. IP, on the other hand, is responsible for addressing and routing packets across the network. IP uses a 32-bit address, called an IP address, to identify each device on the network. An IP address is made up of four octets, separated by periods. Each octet can have a value from 0 to 255. This addressing scheme allows packets to be routed from one device to another across the network. However, with the increasing number of devices connected to the Internet, the available IPv4 address space is running out. To address this issue, IPv6 was introduced. IPv6 uses a 128-bit address, providing a much larger address space than IPv4. This means that there are more than enough addresses to accommodate the growing number of devices on the network. In some control applications, TCP/IP is used as an adaptation, where other protocols use TCP/IP as an envelope for data transmission. For example, Modbus TCP/IP uses TCP/IP as an envelope for

data transmitted per the Modbus protocol. This adaptation allows data to be sent from one node to another on a network using the same addressing scheme as IP.



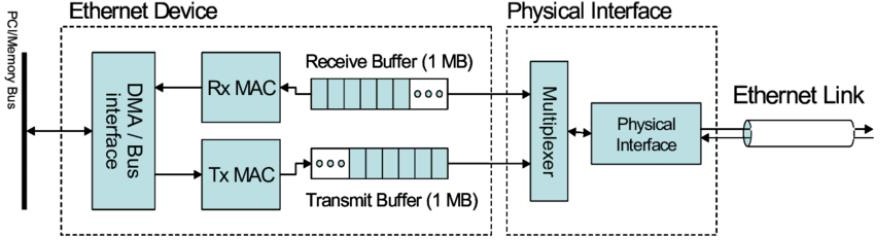
## Figure 6.4.2 TCP/IP Model

* + 1. **Ethernet IP**

EtherNet/IP is an open protocol that is managed by the Open DeviceNet Vendors Association (ODVA). It is a protocol that uses Ethernet and the Common Industrial Protocol (CIP) to enable communication between nodes in an industrial control system. Ethernet is a widely used communication protocol that allows devices to communicate with each other over a network. However, Ethernet is non-deterministic, which means that it does not guarantee a predictable transmission time for data packets EtherNet/IP overcomes this limitation by providing deterministic communication for input and output updates, which makes it suitable for plant control applications. This ensures that the data packets are transmitted and received within a predictable time frame, which is essential in industrial control systems. EtherNet/IP is also compatible with drives and motor control centers, allowing for seamless integration with various devices in a plant. One of the advantages of EtherNet/IP is that it supports both processor-to-remote rack and processor-to-processor communications. This allows for efficient communication between devices and enables a high level of automation in industrial processes. Moreover, the use of EtherNet/IP can eliminate the need for wiring and conduits, reducing installation and maintenance costs.

The use of Ethernet-based protocols like EtherNet/IP is gaining popularity in motion control applications that require fast response times. Ethernet’s high communication speeds make it an ideal choice for such applications. As John Rinaldi of Real Time Automation suggests,

EtherNet/IP is expected to become an even more important protocol in the 2020s due to its high data transfer speed and deterministic communications. In fact, he suggests that EtherNet/IP could potentially serve as a master protocol in large networks, which highlights its potential as a powerful tool in the industrial automation industry.

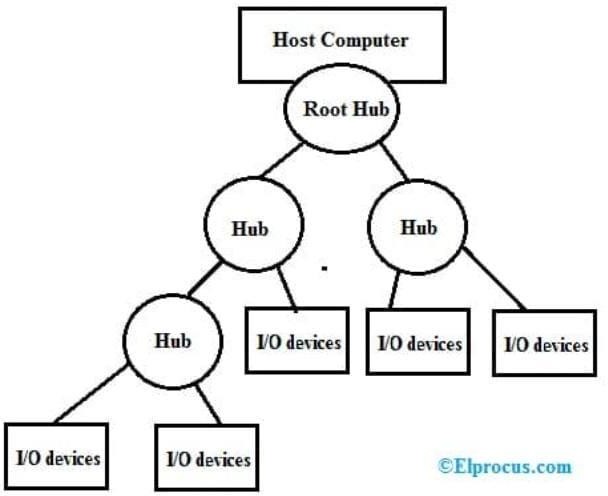


## Figure 6.4.3 Ethernet/IP

* + 1. **USB Protocol**

The USB protocol is a way for different devices like printers, cameras, keyboards, and others to communicate with a computer or smartphone. It allows you to easily connect and disconnect devices without having to restart your computer. The USB has different types of connectors, but the most commonly used are Type A and Type B. Older connectors have been replaced with new ones like Mini-USB, Micro-USB, and USB-C. The USB architecture looks like a tree, where each device connects to a hub, which acts as the connection point between the devices and the computer. The root hub connects the whole structure to the computer, and it checks for changes in the system, like when you add or remove a device. The USB protocol uses a polling principle, where the computer checks if a device is ready to send data, so the device doesn't have to keep updating the computer.

When you connect a new device, the computer assigns it an address and reads its data to determine its capabilities. The computer then uses a driver to talk to the device and assigns it a permanent address. The USB is designed to be easy to use, so you don't have to worry about configuring devices or restarting your computer.



## Figure 6.4.4 USB Architecture

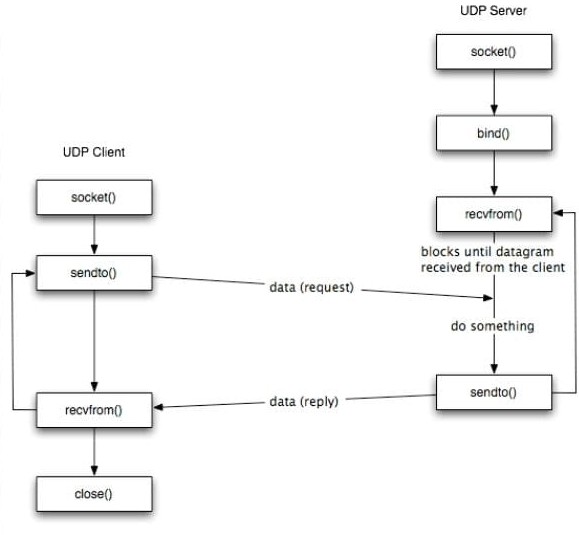
* + 1. **Socket Communication**

A socket is a communication channel that enables two programs to exchange data over a network. This two-way communication between a client and a server uses a well-established protocol, which defines the rules and behavior that both the client and server must follow to establish a connection. To create a socket connection, a server program first creates a socket at a specific port and waits for a client to request a connection. A port is a unique address on the host computer, and once the connection is established, the server creates input and output streams to the socket for sending and receiving messages to and from the client. The client can also close the connection, but it is usually done by the client.

From the client side, the protocol involves creating a socket and attempting to connect to the server using the server's URL and port number. Once a connection has been established, the client creates input and output streams to the socket to exchange messages with the server. The client can then close the connection when the service is completed. A socket connection can be likened to a two-way telephone conversation, where the client initiates the connection by dialing a number and communicating with a service agent.

The socket connection has two channels: the client's output stream is connected to the server's input stream, and vice versa. To code a client/server application in Java, the server first

creates a ServerSocket at a port number and waits for a client request using the accept() method, which blocks until a connection is established. Once a connection is established, the server can begin communicating with the client. After delivering the requested service, the server can close the connection using the close() method. On the other hand, the client initiates the two-way connection by requesting the service, and there are only three steps involved: opening a socket connection to the server, conversing with the server, and closing the connection



## Figure 6.4.5 Socket Communication

* 1. **Python programming language**

Python is a versatile language that can be used for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, and machine learning. Its flexibility comes from its modular design, which allows developers to easily import and use pre-built libraries and modules to extend the language's functionality. Python's built-in support for object-oriented programming, combined with its dynamic typing, makes it an excellent choice for developing complex applications. One of the key advantages of Python is its ease of use and readability. The language's syntax is designed to be intuitive and easy to understand, which makes it a popular choice for beginners and experienced developers alike. Python's built-in data structures, such as lists, dictionaries, and tuples, make it easy to work with complex data sets and manipulate data on the fly. Python's dynamic typing also makes it very flexible, allowing developers to easily switch between different data types without having to worry about type declarations. This feature can save a lot of time and effort, especially when dealing with large or complex data sets. Python's open-source nature also makes it a popular choice for developers, as it allows them to easily share their code and collaborate with other developers. The Python community is large and active, with many third-party libraries and modules available for free, which can save developers a lot of time and effort in building their applications. In addition to its ease of use and flexibility, Python is also known for its speed and efficiency. Although it is an interpreted language, Python's built-in libraries and modules are optimized for speed, which makes it a popular choice for developing high-performance applications. Python is a powerful and versatile language that can be used for a wide range of applications. Its ease of use, flexibility, and speed make it a popular choice for developers of all skill levels. Whether you are building a web application, analyzing large data sets, or developing artificial intelligence algorithms, Python has the tools and features you need to get the job done quickly and efficiently.

Python is a versatile programming language that has gained popularity due to its simplicity, flexibility, and ease of use. Although Python programs are typically slower than Java programs, they are much quicker to develop due to Python's high-level data types, dynamic typing, and powerful built-in libraries. In this essay, we will examine the key features of Python that make it an attractive choice for developers. Python is a high-level programming language, which means that it is designed to be easy to read and write. This is accomplished through a syntax that emphasizes readability, with minimal clutter and simple, English-like keywords. Python also supports object-oriented programming, which allows developers to

create reusable code in the form of classes and objects. This makes it easier to create complex applications and manage large codebases. One of the key features of Python is its built-in high-level data types, which allow developers to manipulate complex data structures with ease. For example, Python includes built-in support for lists, dictionaries, sets, and tuples, which are all commonly used in data manipulation and analysis. Python also includes a number of powerful libraries, such as NumPy, Pandas, and Matplotlib, which make it easy to work with large data sets and visualize data. Python's dynamic typing is another key feature that makes it an attractive choice for developers. Unlike statically typed languages like Java, Python does not require developers to explicitly declare the data types of variables or function arguments. This makes it quicker and easier to write code, since developers do not need to spend time declaring and managing data types. However, dynamic typing can also make Python programs slower than equivalent Java programs, since the interpreter must inspect objects at runtime to determine their type. Python's simplicity and flexibility make it an excellent choice for a wide range of applications. Python is often used for web development, scientific computing, data analysis, and artificial intelligence. In addition, Python is an excellent choice for rapid prototyping and testing, since it is quick and easy to write and modify code. Despite its many advantages, Python is not the right choice for every application. For example, Python programs may not be as fast as equivalent programs written in C or C++, which makes Python less suitable for applications that require high performance or real-time processing. In addition, Python's dynamic typing can make it more difficult to debug certain types of errors, such as type mismatches or syntax errors.

In conclusion, Python is a powerful, flexible, and easy-to-use programming language that has gained popularity due to its high-level data types, dynamic typing, and powerful built-in libraries. Python is an excellent choice for a wide range of applications, including web development, scientific computing, data analysis, and artificial intelligence. While Python may not be the right choice for every application, its simplicity and flexibility make it a popular choice among developers.

## Multiprocessing in Programming

Multiprocessing is a package that supports spawning processes using an API similar to the Threading module. The multiprocessing package offers both local and remote concurrency, effectively side-stepping the global interpreter lock by using subprocesses instead of threads. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine. It runs on both Unix and Windows. The multiprocessing module also introduces APIs that do not have analogs in the thrading module. A prime example of this is the pool object which offers a convenient means of parallelizing the execution of a function across multiple input values, distributing the input data across processes (data parallelism). The following example demonstrates the common practice of defining such functions in a module so that child processes can successfully import that module. This basic example of data parallelism using pool. The multiprocessing module provides a Process class, which is similar to the threading.Thread class in the threading module. The Process class allows us to create a new process by creating a Process object, just as we create a new thread by creating a Thread object. The Process class follows the same API as the Thread class, with some additional methods and properties specific to multiprocessing. To create a new process, we create a new instance of the Process class and pass a target function and its arguments to the constructor, just like we do when creating a new thread with the Thread class. After creating the Process object, we call its start() method to start the new process. The start() method starts a new interpreter process, and in this new process, the target function is called with the arguments passed to the Process constructor. The multiprocessing module also provides some additional features, such as inter-process communication and synchronization mechanisms, to allow processes to share data and coordinate their execution.

Depending on the platform, multiprocessing supports three ways to start a process. These *start* methods are

spawn

The parent process starts a fresh Python interpreter process. The child process will only inherit those resources necessary to run the process object’s run() method. In particular, unnecessary file descriptors and handles from the parent process will not be inherited. Starting a process using this method is rather slow compared to using fork or forkserver.

112

Available on Unix and Windows. The default on Windows and macOS.

fork

The parent process uses os.fork() to fork the Python interpreter. The child process, when it begins, is effectively identical to the parent process. All resources of the parent are inherited by the child process. Note that safely forking a multithreaded process is problematic. Available on Unix only. The default on Unix.

fork server

When the program starts and selects the forkserver start method, a server process is started. From then on, whenever a new process is needed, the parent process connects to the server and requests that it fork a new process. The fork server process is single threaded so it is safe for it to use os.fork(). No unnecessary resources are inherited.

When using multiprocessing in Python, it is possible to create shared objects using shared memory which can be inherited by child processes. One way to create a shared object is to use the multiprocessing.Value() method. This method returns a ctypes object that is allocated from shared memory. The first argument of the method, typecode\_or\_type, determines the type of the returned object. It can either be a ctypes type or a one-character typecode used by the array module. The remaining arguments, \*args, are passed on to the constructor for the type. By default, the returned object is actually a synchronized wrapper for the object, which can be accessed via the value attribute of a Value. This means that access to the object is automatically protected by a lock, making it “process-safe”. If the lock argument is True (the default), a new recursive lock object is created to synchronize access to the value. If the lock argument is a Lock or RLock object, it will be used to synchronize access to the value. If lock is False, then access to the returned object will not be automatically protected by a lock, so it will not necessarily be “process-safe”. It is important to note that operations like += which involve a read and write are not atomic. Therefore, if you want to atomically increment a shared value, it is insufficient to just do x.value += 1. Instead, you must use a lock object to ensure atomicity. For example, you can acquire the lock, increment the value, and then release the lock:

```

import multiprocessing

def increment(counter, lock): with lock:

counter.value += 1

if name == ' main ':

counter = multiprocessing.Value('i', 0) lock = multiprocessing.Lock()

processes = []

for i in range(10):

p = multiprocessing.Process(target=increment, args=(counter, lock)) processes.append(p)

p.start()

for p in processes: p.join()

print(counter.value)

```

In the above example, a Value object of type 'i' (integer) is created with an initial value of 0. A Lock object is also created to synchronize access to the shared value. The increment()

function takes the shared counter and the lock as arguments, acquires the lock, increments the counter, and then releases the lock. This ensures that the increment operation is atomic and that the shared value is updated correctly. Finally, 10 processes are created to call the increment() function, and their results are printed after they finish running. The

`multiprocessing` module in Python provides ways to create and manage processes, and also includes tools to create shared memory that can be inherited by child processes. One way to create shared memory is to use the `Value()` function, which returns a ctypes object allocated from shared memory. By default, the return value is actually a synchronized wrapper for the object. The object itself can be accessed via the `value` attribute of a `Value`. The

`typecode\_or\_type` argument determines the type of the returned object: it is either a ctypes type or a one character typecode of the kind used by the array module. The `\*args` argument is passed on to the constructor for the type. If the `lock` argument is `True` (the default), then a new recursive lock object is created to synchronize access to the value. If `lock` is a `Lock` or `RLock` object then that will be used to synchronize access to the value. If `lock` is `False` then access to the returned object will not be automatically protected by a lock, so it will not necessarily be “process-safe”.

Another way to create shared memory is to use the `RawArray()` function, which returns a ctypes array allocated from shared memory. The `typecode\_or\_type` argument determines the type of the elements of the returned array: it is either a ctypes type or a one character typecode of the kind used by the array module. If `size\_or\_initializer` is an integer then it determines the length of the array, and the array will be initially zeroed. Otherwise

`size\_or\_initializer` is a sequence which is used to initialize the array and whose length determines the length of the array. Note that setting and getting an element is potentially non- atomic – use `Array()` instead to make sure that access is automatically synchronized using a lock. Finally, the `RawValue()` function returns a ctypes object allocated from shared memory. The `typecode\_or\_type` argument determines the type of the returned object: it is either a ctypes type or a one character typecode of the kind used by the array module. `\*args` is passed on to the constructor for the type. Note that setting and getting the value is potentially non-atomic – use `Value()` instead to make sure that access is automatically synchronized using a lock. An array of `ctypes.c\_char` has `value` and `raw` attributes which allow one to use it to store and retrieve strings.

The multiprocessing.sharedctypes module provides functions to create ctypes objects and arrays that are allocated from shared memory, which can be used to share data between different processes.

The functions provided by the module include:

* multiprocessing.sharedctypes.RawArray: Returns a ctypes array allocated from shared memory. The type of the elements in the array is determined by the typecode\_or\_type argument, and the size of the array is specified by the size\_or\_initializer argument. If size\_or\_initializer is an integer, the array is initialized with zeroes. Otherwise, size\_or\_initializer is a sequence used to initialize the array.
* multiprocessing.sharedctypes.RawValue: Returns a ctypes object allocated from shared memory. The type of the object is determined by the typecode\_or\_type argument, and the

\*args argument is passed on to the constructor for the type.

* multiprocessing.sharedctypes.Array: Same as RawArray, but with an additional lock argument. If lock is True (the default), a process-safe synchronization wrapper is returned instead of a raw ctypes array. If lock is a Lock or RLock object, it will be used to synchronize access to the array. If lock is False, access to the array will not be protected by a lock.
* multiprocessing.sharedctypes.Value: Same as RawValue, but with an additional lock argument. If lock is True (the default), a process-safe synchronization wrapper is returned instead of a raw ctypes object. If lock is a Lock or RLock object, it will be used to synchronize access to the object. If lock is False, access to the object will not be protected by a lock.

## Multithreading in programming

To add to multiprocessingthere are some additional details about the Thread class in Python:

-The Thread class is defined in the built-in threading module, which provides high-level thread support in Python.

* When you create a new thread using the Thread class, it spawns a new operating system- level thread in the background. The number of threads that can run concurrently depends on the system resources and the Python interpreter's implementation.
* The Thread class provides various methods and attributes to interact with the thread, such as start(), join(), is\_alive(), name, daemon, etc.
* The start() method is used to start the thread's activity. Once the thread is started, its run() method will be executed in a separate thread of control.
* The join() method blocks the calling thread until the target thread completes its execution. If a timeout argument is passed to join(), the calling thread will block for at most that many seconds.
* The is\_alive() method returns True if the thread is currently alive (i.e., its run() method is executing), and False otherwise.
* The name attribute of a thread is a string that can be used to identify the thread. By default, each thread is assigned a name of the form "Thread-N", where N is a unique integer identifier.
* The daemon attribute of a thread determines whether it is a daemon thread or not. Daemon threads are a type of background threads that will be terminated automatically when all non- daemon threads have exited.
* By default, all threads in Python are non-daemon threads, but you can change this by setting the daemon attribute to True before starting the thread.
* If a thread's run() method raises an unhandled exception, the exception will be printed to the standard error stream by default. You can customize this behavior by registering a custom exception handler using the threading.excepthook() function.

These are some of the benefits of creating a multithreaded application in Python. To elaborate further:

1. Effective utilization of computer system resources: By dividing a program into multiple threads, each thread can perform its task simultaneously, making efficient use of the available CPU resources.
2. More responsive applications: Multithreading enables a program to continue running while waiting for I/O operations or other resource-intensive tasks. This results in more responsive applications as other threads can continue running while a particular thread is blocked.
3. Economical sharing of resources and state: Multithreading allows for the sharing of resources and state between threads, reducing the need for duplication of data or resources. This makes the application more economical in terms of memory usage.
4. Effective use of multiprocessor architectures: Multithreading is especially effective on multiprocessor architectures as different threads can be executed on different processors simultaneously, further increasing the application's performance.
5. Time-saving: By executing multiple threads simultaneously, the time required to execute a program can be significantly reduced, leading to faster execution and improved performance.
6. Reduced memory usage: Threads share the memory space of the parent process, which reduces the amount of memory required for the execution of the program. This can be especially important when dealing with large datasets or resource-intensive tasks.The threading module provided with Python includes a simple-to-implement locking mechanism that allows you to synchronize threads. A new lock is created by calling the *Lock()* method, which returns the new lock. The *acquire(blocking)* method of the new lock object is used to force threads to run synchronously. The optional *blocking* parameter enables you to control whether the thread waits to acquire the lock. If *blocking* is set to 0, the thread returnsimmediately with a 0 value if the lock cannot be acquired and with a 1 if the lock was acquired. If blocking is set to 1, the thread blocks and wait for the lock to be released. The *release()* method of the new lock object is used to release the lock when it is no longer required.

## Pymodbus library

Pymodbus is a Python library that provides a full implementation of the Modbus protocol, which is commonly used in industrial automation systems for communication between devices. Pymodbus supports various communication modes, including TCP, RTU-over-TCP, UDP, serial, and TLS. It can be used either synchronously or asynchronously with asyncio. Pymodbus is a lightweight project that doesn't require any third-party dependencies except pyserial. The library comes with many examples and a server/client simulator that can be controlled via a REST API, making it easy to integrate into test suites. Pymodbus requires Python 3.8 or higher, and its tests are run against multiple versions of Python and operating systems. With its robust features and support for various communication modes, Pymodbus is an excellent choice for implementing Modbus communication in Python-based industrial automation systems.

Pymodbus provides a full read/write protocol for both discrete and register types in Modbus. It also supports most of the extended protocol features such as diagnostic, file, pipe, setting, and information. Pymodbus supports various Modbus communication modes, including TCP, RTU-over-TCP, UDP, TLS, Serial ASCII, Serial RTU, and Serial Binary. It provides both asynchronous and synchronous versions of its core for easy integration into different types of projects. The payload builder/decoder utilities provided by Pymodbus makes it easier to work with the Modbus protocol. Additionally, Pymodbus provides a REPL (Read- Eval-Print Loop) for quick testing of Modbus functionality. One of the significant advantages of Pymodbus is its customizable framer that allows for custom implementations to be created. This enables users to tailor the behavior of Pymodbus to their specific needs, making it a flexible and versatile Modbus library.

The Pymodbus server also provides a number of backend contexts as a datastore, which can be used to store data and respond to modbus requests. Some of the backend contexts provided are:

* ModbusSequentialDataBlock: A datastore that allows the user to specify a sequential block of data to use as a datastore.
* ModbusSparseDataBlock: A datastore that allows the user to specify a sparse block of data to use as a datastore.
* ModbusSlaveContext: A context that allows the user to specify multiple data blocks, each with its own address range, to use as a datastore.
* ModbusServerContext: A context that allows the user to specify multiple slave contexts, each with its own modbus address range, to use as a datastore.

In addition to these backend contexts, the Pymodbus server also provides a number of other features, such as:

* Support for custom framer implementations, allowing for non-standard modbus message formats.
* Support for custom data access functions, allowing the user to specify how data is accessed and modified in the backend context.
* Support for custom exception handling, allowing the user to specify how exceptions are handled by the server.
* Support for custom logging and debugging, allowing the user to specify how logging and debugging information is output by the server.

The pymodbus server can be used to test the connectivity and communication with a large number of devices on a network. It allows for easy scripting and integration into existing solutions, making it a useful tool for system administrators and developers. The server can handle multiple communication modes and is available in both synchronous and asynchronous versions, providing flexibility in its usage. It also offers a range of backend contexts for data storage and provides full control over the server context, including device information and counters.

## Pyserial library

The "serial" module in Python provides a consistent class-based interface for accessing serial ports on different platforms, including Windows, macOS, Linux, BSD, and IronPython. The module automatically selects the appropriate backend based on the operating system.

Some of the features of the "serial" module include:

* Access to port settings through Python properties, including support for different byte sizes, stop bits, parity, and flow control with RTS/CTS and/or Xon/Xoff.
* File-like API with "read" and "write" methods, as well as support for "readline" and other file operations.
* Working with or without receive timeout.
* Compatibility with the standard Python io library.
* Support for RFC 2217 client (experimental), which allows for remote configuration of serial ports, and a server provided in the examples.

The "serial" module also sets up the port for binary transmission, without any automatic stripping of NULL bytes or translation of CR-LF sequences, making it suitable for a wide range of applications.

Python code to open a serial connection import serial.

**>>>** ser = serial.Serial('/dev/ttyUSB0') *# open serial port*

**>>>** print(ser.name) *# check which port was really used*

**>>>** ser.write(b'hello') *# write a string*

**>>>** ser.close() *# close port*

## Deepclean system

Introduction

As smartphones become an essential part of our daily lives, it is imperative that we keep them clean and well maintained. However, manual cleaning of smartphones can be time- consuming, error-prone, and subjective. The use of Deepclean, a cleaning robot for smartphones, can simplify the process and provide a more accurate and objective cleaning experience. This paper will explore the features and benefits of Deepclean, including its automated deep learning-based cleaning system and Ethernet communication capabilities.

Features of Deepclean

Deepclean is an entirely automated cleaning robot designed to remove smudges, dust, and lint from smartphones. Its compact housing makes it easy to use in a variety of settings, and its deep learning-based system ensures a more accurate and efficient cleaning process. Some of the features of Deepclean include:

1. Robust Inspection System

Deepclean has a robust inspection system that enables it to identify areas of a smartphone that require cleaning accurately. This system uses deep learning algorithms to detect and analyze smudges, dust, and lint on a smartphone's surface. This feature ensures that all areas of a smartphone are cleaned and no smudges or dirt are left behind.

1. Accurate and Objective Cleaning System

Deepclean's cleaning system is entirely automated and uses a deep learning-based algorithm to provide an accurate and objective cleaning process. The system analyzes the type of smudges or dirt present on a smartphone and adjusts its cleaning process accordingly. This feature ensures that the smartphone is cleaned thoroughly, and no residue or smudges are left behind.

1. Flexibility in Cleaning Operation

Deepclean provides users with flexibility in their cleaning operation. The system can be adjusted to clean different types of smartphones, including various sizes and shapes. Additionally, users can adjust the cleaning process to meet their specific requirements, providing them with complete control over the cleaning operation.

1. Ethernet Communication

Deepclean communicates through Ethernet, allowing for easy integration into existing networks. This feature makes it easy to control and monitor the cleaning process remotely, providing users with real-time updates on the progress of the cleaning operation.

Benefits of Deepclean

The use of Deepclean provides several benefits, including:

1. Time-Saving

Manual cleaning of smartphones can be time-consuming, especially when cleaning large quantities of devices. Deepclean's automated cleaning system reduces the time needed to clean smartphones significantly. This feature enables users to clean a large number of smartphones in a short amount of time, allowing them to focus on other essential tasks.

1. Error-Free Cleaning

Manual cleaning of smartphones is subjective and can be prone to errors. Deepclean's deep learning-based system ensures an objective cleaning process, reducing the likelihood of errors. This feature ensures that all smartphones are cleaned thoroughly, and no smudges or dirt are left behind.

1. Cost-Effective

The use of Deepclean can be cost-effective, especially for businesses that need to clean a large number of smartphones regularly. The automated cleaning process reduces the need for manual labor, reducing labor costs. Additionally, the objective cleaning process ensures that all smartphones are cleaned thoroughly, reducing the need for re-cleaning.

1. Improved User Experience

A clean smartphone provides a better user experience, especially for those who use their smartphones regularly. The use of Deepclean ensures that smartphones are cleaned thoroughly, providing users with a better overall experience. Additionally, the use of Deepclean can improve the lifespan of smartphones, reducing the need for frequent replacements. The use of Deepclean, a cleaning robot for smartphones, provides a more accurate, efficient, and objective cleaning experience. Its deep learning-based system and

Ethernet communication capabilities make it easy to use and monitor, while its flexibility in cleaning operations provides users with complete control over the cleaning process. The use of Deepclean provides several benefits, including time-saving, error-free cleaning, cost-effect

Deepclean provides two versions for users to choose from: a standalone tabletop version and a fully automated production version with the capability of multiple units working in synchronized mode with robotic leading and unloading. Both versions are fully automatic and can flip, clean, and capture images. The system is fast, accurate, and uses industrial-grade cleaning solutions and fabric for precise and efficient processing.

* + Consistent and accurate industrial-grade cleaning.
  + Easy to maintain and replace worn-out materials.
  + Flexible with advanced machine vision and deep learning algorithms
  + Independent of make model
  + Configurable cleaning parameters
  + Selection of cleaning solutions
* Cleans glossy, shiny, or rough surfaces.
* Easy to operate and can be monitored by unskilled labor.
* Optimized Mode with lower cycle time.
* Cloud-based data storage and reporting for dispute resolution and audits.
* Integration-ready solution for customers' factory network.



## Figure 6.10.1 Deepclean

Deepclean is a sophisticated cleaning robot that employs several sensors to ensure accurate and efficient cleaning of smartphones. Following is the list and explanation of all the sensors present in the system:

1. Mobile Presence Sensor: This sensor detects the presence of a smartphone inside the cleaning compartment. It ensures that the cleaning process only begins when a smartphone is present, avoiding wastage of cleaning solutions.
2. Misplace Sensor: This sensor ensures that the smartphone is placed in the correct position inside the cleaning compartment. It detects any misplacement of the smartphone, preventing damage to the device and ensuring that the cleaning process is carried out accurately.
3. Positioning Sensor: This sensor is used to precisely position the cleaning solution applicator onto the smartphone's surface. It ensures that the cleaning solution is applied evenly and accurately, avoiding wastage of cleaning solutions.
4. Liquid Level Sensor: This sensor monitors the level of cleaning solution in the cleaning compartment. It ensures that the cleaning solution is at the correct level for efficient cleaning, avoiding under or overuse of cleaning solutions.
5. Electrical Panel Temperature Sensor: This sensor monitors the temperature of the electrical panel of the cleaning robot. It ensures that the electrical panel operates within safe temperature limits, avoiding any damage or malfunctioning of the cleaning robot.
6. Inside Environment Temperature Sensor: This sensor monitors the temperature inside the cleaning compartment. It ensures that the temperature is optimal for efficient cleaning and avoids overheating of the cleaning robot.
7. Reed Switches: These sensors are used to detect the position of the cleaning solution applicator. They ensure that the cleaning solution applicator is in the correct position for efficient cleaning and avoid wastage of cleaning solutions.
8. Gripper Sensor for Stepper Motor: This sensor monitors the gripper's position that holds the smartphone during the cleaning process. It ensures that the smartphone is held firmly and prevents any accidental dropping of the device.
9. Pressure Sensor: This sensor monitors the pressure applied by the cleaning solution applicator onto the smartphone's surface. It ensures that the pressure is optimal for efficient cleaning and avoids damaging the device.
10. Rotary Disc Presence Sensor: This sensor detects the presence of the rotary disc that moves the smartphones through the cleaning process. It ensures that the rotary disc is in the correct position for efficient cleaning and avoids any damage or malfunctioning of the cleaning robot.

Deepclean employs a wide range of sensors that ensures that the cleaning process is carried out accurately and efficiently, avoiding any damage or malfunctioning of the cleaning robot or the smartphones being cleaned. These sensors enable Deepclean to provide an objective and thorough cleaning process, ensuring the end-user's satisfaction. In addition to the sensors, Deepclean also employs various motors and motor drives to perform the cleaning process.

Following is the list and explanation of all the motors and motor drives present in the system:

1. Beckhoff Servo Motor: This motor is used to drive the cleaning solution applicator's horizontal movement. It ensures accurate and precise movement of the cleaning solution applicator, enabling the efficient and thorough cleaning of the smartphone.
2. Innovance Servo Drive: This drive is used to control the speed and position of the cleaning solution applicator. It ensures that the cleaning solution applicator moves at the optimal speed and position, providing efficient and precise cleaning.
3. Beckhoff Servo Motor for Device Body Cleaning: This motor is used to drive the cleaning solution applicator's vertical movement. It ensures accurate and precise vertical movement of the cleaning solution applicator, enabling the efficient and thorough cleaning of the smartphone's body.
4. Beckhoff Servo Motor for Vertical Axis: This motor is used to drive the rotary disc's vertical axis movement. It ensures accurate and precise movement of the rotary disc, enabling the efficient and thorough cleaning of the smartphone.
5. Innovance Servo Motor for Rotary: This motor is used to drive the rotary disc's rotational movement. It ensures accurate and precise rotational movement of the rotary disc, enabling the efficient and thorough cleaning of the smartphone.
6. Stepper for Gripper Movement: This motor is used to drive the gripper's movement that holds the smartphone during the cleaning process. It ensures accurate and precise movement of the gripper, enabling the safe and secure holding of the smartphone during the cleaning process.

In conclusion, Deepclean employs a variety of motors and motor drives to drive the cleaning process accurately and precisely. These motors and drives ensure efficient and thorough cleaning of the smartphone, enabling the end-user's satisfaction.

The AM8100 series of servomotors from the AM8000 family is designed for operation on servo I/Os and offers a complete construction kit in the low voltage range. These compact servomotors have a power range of 50 to 750 W and offer high dynamics, making them suitable for a wide range of applications, including pick-and-place applications in industrial robots and mechanical engineering applications that require a compact design and high positioning accuracy.

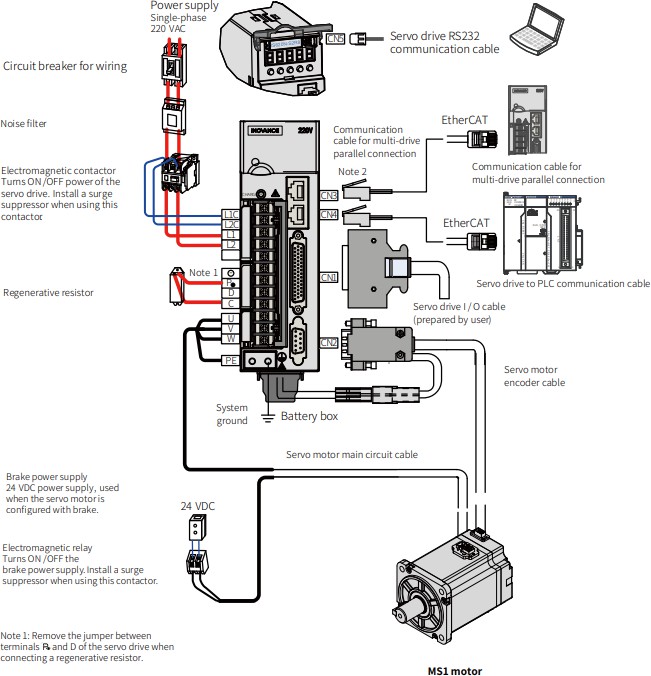
The AM8100 motors come equipped with an absolute encoder that eliminates the need for homing. The position of the drive is saved in the EEPROM, making it ideal for adjustable axes. The encoder data is transmitted digitally via the motor cable, eliminating the need for an encoder cable. The servo terminal is fully integrated into the Beckhoff control system, making it easy to commission the drive axis. The motors also use an electronic identification plate, which simplifies the reading of motor parameters and reduces engineering effort. The TwinCAT automation software enables convenient parameterization of the servomotors. The AM8100 motors can be equipped with a backlash-free permanent magnet holding brake, a sealing ring, or a feather key groove. They are also available with an absolute feedback system (OCT) or a sturdy rotary resolver encoder. Generously dimensioned bearings are used to ensure durability in general mechanical engineering applications. Matching gears and prefabricated connecting cables complete the ultra-compact drive axis.

The following are the specifications of innovance stepper motor high-performance motor suitable for use in industrial automation, robotics, and other motion control applications. Breakdown of some of the features are listed accordindly.

* High current loop bandwidth (> 4 kHz) and high speed loop bandwidth (1.2 kHz): This indicates that the motor is capable of fast, precise movements with high accuracy and repeatability.
* 20-bit resolution serial incremental encoder and 23-bit resolution multi-turn absolute encoder: These encoders provide highly accurate position feedback to the motor control system.
* IP67 degree of protection with oil seal (MS1H1/H4): This indicates that the motor is highly resistant to dust and water ingress, making it suitable for use in harsh environments.
* Low cogging torque (< 1% of rated torque): This means that the motor has a smooth rotation with minimal vibration, providing high accuracy and a high-quality output.
* Wide range of rated torques (from 0.16 Nm up to 48.0 Nm), supply voltages (single-phase 220V, three-phase 220V, three-phase 380V), and motor frames (40, 60, 80, 100, 130, 180): This makes the motor suitable for a wide range of applications with different requirements for torque, voltage, and size.
* Low and medium inertia selection: This indicates that the motor is designed to provide high dynamic response and fast acceleration/deceleration, making it suitable for high-speed applications.
* Network compatibility: The motor can be used with different types of network protocols, including Pulse/Analog, EtherCAT, and CANopen, allowing it to be integrated with various control systems.
* Smart tuning: The motor features automatic tuning capabilities, including inertia auto- tuning, automatic gain tuning, adaptive notch filter, and automatic/manual damping, which make it easier to optimize the motor's performance in different applications.

Overall, the specifications suggest that this servo motor is a high-quality, versatile, and reliable motion control solution suitable for a wide range of industrial applications.

The following is the connection architecture of the innovace motor with its motor driver which is installed in deepclean machine.



## Figure 6.10.2 Innovance Motor and Drive architecture

* 1. **Device profile extraction**

In this system QR code located on the device is scanned by smart code reader and IMEI of that particular device is extracted. After IMEI extraction that particular IMEI number is sent to custom Application program interface where the complete information of the device is stored. This database includes the physical dimensions of the device as well as the data related to cosmetic grading and functional test result. Cosmetic grading of the device is done in another system know as Deepsight as well as the functional test of the device such as device touch screen testing, firmware testing, Bluetooth, wifi testing ,network testing is completed in another system known as Impersonator. The profile of the device includes the physical and geometrical details of the device. This details are stored in the database which is integrated using mule software which is used for data integration with application data interface(API).

API has undergone significant transformations over the years. Unlike earlier APIs, modern APIs adhere to specific standards such as HTTP and REST, making them more developer- friendly, self-described, easily accessible, and broadly understood. They are also designed for consumption by specific audiences, and are documented and versioned in a way that enables users to have clear expectations of their maintenance and lifecycle. One of the most significant benefits of modern APIs is that they can be monitored and managed for both performance and scale, and they have a much stronger discipline for security and governance. This is achieved through the use of small packets of data that are communicated between the server and the user's device, minimizing the amount of data that is exposed to each other. In addition, the modern API is treated more like a product than code, and has its own software development lifecycle, including mocking, designing, testing, building, managing, and retiring. This enables APIs to be well-documented for both consumption and versioning. APIs have transformed the technology space by offering standardization, security, and governance, and by being designed for specific audiences and treated as products rather than code.

MuleSoft offers a comprehensive set of enterprise solutions that enable businesses to integrate and automate their systems and applications. The Anypoint Platform, in particular, provides tools for integration, APIs, and automation, as well as prebuilt assets to speed up application development and delivery. By using MuleSoft's solutions, businesses can automate projects and streamline complex business processes, which can save time and

accelerate development. They can also unlock the value of real-time data and well-integrated systems, which can lead to better decision-making and improved customer experiences. The Anypoint Platform also allows development teams to discover APIs built anywhere using their own CI/CD pipelines, or build APIs from scratch for almost any use case or protocol. This enables centralized data accessibility and creates opportunities to improve processes and deliver intelligent customer experiences faster. Additionally, APIs and reusable assets increase enterprise agility, minimize rework, and speed up time-to-market. MuleSoft's solutions offer a robust set of tools and capabilities that can help businesses improve their operations, increase agility, and deliver better customer experiences. APIs can also help companies to leverage data insights more effectively, enabling them to create new products and services, optimize existing ones, and gain a competitive advantage in the market. APIs make it possible to securely access and share data between systems, devices, and applications in real-time, enabling businesses to make faster and more informed decisions. For example, a retail company can use APIs to integrate data from various sources, such as social media, customer reviews, and sales data, to gain insights into customer behavior and preferences. This information can be used to develop personalized marketing campaigns and promotions, as well as to optimize pricing and inventory management. In addition, APIs can help businesses to streamline their internal operations, enabling different departments and teams to collaborate more effectively. This can help to reduce duplication of effort and increase productivity, while also providing better visibility and control over business processes. APIs are a critical tool for businesses looking to stay competitive in today's digital landscape. By adopting API-driven strategies, companies can accelerate innovation, improve customer experiences, and drive growth and profitability.

ZPL (Zebra Programming Language) is a proprietary language used by Zebra Technologies for their barcode and label printers. It is a command-based language that specifies the layout, content, and formatting of labels to be printed. The language includes commands for font selection, barcodes, graphics, and more.To print a label with a Zebra printer, the data to be printed must first be converted into ZPL language. This can be done using a variety of methods, including manually writing the ZPL code or using a library like the ZPL library in Python. The ZPL library in Python allows developers to easily create ZPL code from within their Python applications. This code can then be sent to the Zebra printer to print the label. The library includes functions for creating barcodes, text, and graphics, as well as formatting options like font selection and label size.Using ZPL language and a Zebra barcode and label

printer allows for efficient and accurate printing of labels with device profile information. The use of Python and the ZPL library simplifies the process and allows for easy integration into existing systems. This ZPL library uses Labelrary which is API where preview of the label can be seen.

ZPL (Zebra Programming Language) is a programming language that is used to create labels and barcodes for printing on Zebra printers. As you mentioned, ZPL is composed of commands and command parameters. Commands are instructions that tell the printer what to do, such as "print a barcode" or "set the label width to 2 inches". Commands in ZPL start with a caret (^) or tilde (~) character. Command names follow the caret or tilde, and are one or two letters long. While command names are case-insensitive, uppercase names are most common. Commands can have a number of parameters, which provide variable information that affects the behavior of the command. For example, a command that prints a barcode might have parameters for the type of barcode to print, the data to encode in the barcode, and the height and width of the barcode. Parameters are separated by comma (,) characters.Whitespace (spaces, tabs, newlines) is mostly ignored in ZPL, except for the ^FD command. The ^FD command is used to print text, and whitespace within the text is significant. In this case, whitespace characters are interpreted as spaces between words.

Overall, the combination of commands and parameters in ZPL allows for great flexibility in creating labels and barcodes that meet specific requirements. After successful extraction of profile of the device, that data which includes height, width , device model, device make, and cosmetic grade is printed on the label using Zebra barcode and label printer.This particular barcode is printed using python code which has a zpl library that converts the data into ZPL language. ZPL language is a special language which is used in Zebra barcode and label printers for printing barcodes and labels.

This system has ubuntu 18 installed in its master pc.

**1. Free Operating System for Personal and Enterprise Computing**

One of the advantages of Ubuntu is that it is a free-to-download and open-source operating system. In other words, unlike Microsoft windows and the macOS from Apple, individuals and organizations can own and maintain working computers without the need for paying software licenses or purchasing exclusive devices. Building and maintaining an information technology infrastructure essentially come with costs. Aside from purchasing the hardware, they need a working OS. Nevertheless, small and large organizations can benefit from the

reduced cost savings that come from using free operating systems, as well as from free productivity applications.

**2. Works in a Range of Devices or Computing Platforms**

The OS can be installed in numerous devices, including Windows computers and Mac computers. It also works on network servers, and IoT devices and robots, as well as in emulated or virtualized computer environments via a virtual machine or containers. Several providers of cloud services have used this OS because it supports OpenStack. Other manufacturers and developers of IoT devices and robotics have tinkered with Ubuntu to experiment on new ideas and deploy new products and solutions.

**3. A Well-Rounded Operating System for Desktop Computing**

Another advantage of Ubuntu is that it can relatively compete against [Windows](https://www.profolus.com/topics/advantages-and-disadvantages-of-windows/) and macOS, particularly in providing users with a complete desktop computing experience. For starters, the Desktop edition comes with office productivity applications from Libreoffice.

The app store also includes a sizeable selection of popular apps such as Firefox, Chromium and Google chrome, opera browser, Slack, Skype, VLC Player, and Spotify, among others. Note that the store also includes apps for software developers such as Sublime Text and PHP Storm, as well as for image and video editors such as GIMP and Shotcut.

## User-Friendly with a High Degree of Customizatio

Any person with a basic knowledge of computers can easily install this OS and setup up his or her entire computer system. In addition, Canonical Ltd. has significantly improved the overall user interface of Ubuntu, thus making it as user-friendly as the more popular Windows and macOS. There is also a high degree of customization and personalization. Note that one of the major advantages of a Linux-based operating system is that users have the freedom to customize their system, particularly the desktop environment. Ubuntu naturally inherits this advantage. There are also so-called “flavors” with their own choice of default applications and settings.

## Minimal Hardware or System Requirements

The default Ubuntu does not require high-end system requirements unlike the latest versions of Windows and macOS. The recommended hardware configurations are at least 700 MHz processor, 512MB of RAM, and 5GB storage space. Note that specific flavors such as Lubuntu and Xubuntu are developed and optimized for lower-end hardware specifications. It is also interesting to note that the OS can run directly from an external storage medium such as a solid-state drive or a USB flash drive, as well as DVDs. A live Ubuntu works almost the same as an installed Ubuntu, and it does not alter the configuration of the computer.

## Continued Support from Developers and the Community

Remember that the OS is open-source. One of the many benefits of open-source software is that there is an active community that can help in fixing bugs or introducing new updates or features faster and more efficiently. The same community can help users with troubleshooting and customization requirements. Of course, Canonical releases the OS every six months and provides long-term support releases every two years. Note that it is also considerably secured because applications run on low privileges and the built-in firewall choice of default applications and settings. As this system includes label printer, the Zebra ZD230t is a high-quality thermal transfer barcode printer that is designed for professional use. It is capable of printing all types of labels up to 4.09 inches in width at a speed of 4 inches per second. One of the major advantages of this printer is its ability to connect to your PC, which makes it easy to print labels as per your needs.

The Zebra ZD230t is an ideal choice for various settings, including seller flex, vendor central, retail shops, shopping malls, supermarkets, hotels, canteens, restaurants, corner grocery stores, eCommerce setups, and many other places. The printer's small profile and high-speed printing make it perfect for businesses with high-volume printing needs.The Zebra ZD230t is designed for hassle-free operation. The printer's clamshell design allows for easy access, which makes it easy to load paper rolls with 5.00 in./127 mm outer diameter and core inner diameter ranging from 0.5 in. (12.7 mm) to 1.5 in. (38.1 mm). The printer also features increased memory, which allows you to store more fonts and graphics. Additionally, a single LED indicator and a single button for feed/pause make it easy to operate and identify the printer's status. In terms of technical specifications, the Zebra ZD230t has a print width of

4.09 in./104 mm, and it can print at a speed of 6 in./152 mm per second. The printer has a

print resolution of 203 dpi/8 dots per mm, which ensures that the labels printed are clear and easy to read. The Zebra ZD230t also supports a variety of 1D and 2D barcode types, including Code 39, Code 128, EAN 8/13, UPC A/E, PDF417, Datamatrix, QR code, and more.Zebra ZD230t is an excellent choice for applications that require a reliable and efficient barcode printer. Its advanced features and technical specifications make it one of the best thermal transfer barcode printers available in the market. Its hassle-free operation, coupled with its fast printing speed, makes it a great choice for businesses with high-volume printing needs. For scanning of the QR code Scanning QR codes using Hikvision smart code reader involves a process where the code scanner communicates with the master PC through a socket server and socket client protocol. The master PC acts as the server while the code reader is the client. When the client (code reader) receives a trigger through a Python program, it captures the image of the QR code and extracts the associated code details based on the respective IMEI number of the device. After the code details are extracted, the process continues by extracting mobile profile details which are returned in dictionary format within the Python code. This means that the data obtained from the QR code is processed and presented in a readable format that can be easily understood by the user.

The process of scanning QR codes using Hikvision smart code reader involves efficient communication between the code scanner and master PC using a socket server and socket client protocol. The extracted data is then processed within a Python program, allowing for easy access to mobile profile details.

Image processing and computer vision

In the context of autonomous vehicles, computer vision is used to provide real-time perception of the surrounding environment to enable the vehicle to safely and accurately navigate through complex and dynamic traffic scenarios. Some specific applications of computer vision in autonomous vehicles include:

1. Object detection and recognition: Computer vision is used to detect and classify objects such as other vehicles, pedestrians, cyclists, road signs, and traffic lights. This allows the autonomous vehicle to make decisions based on the objects in its environment.
2. Lane detection and tracking: Computer vision can be used to detect and track the lanes on the road, which is important for keeping the vehicle in the correct lane and avoiding collisions.
3. Obstacle avoidance: Computer vision can help the autonomous vehicle identify and avoid obstacles on the road, such as parked cars, construction barriers, or pedestrians.
4. Traffic sign recognition: Computer vision can be used to recognize and interpret traffic signs such as stop signs, speed limit signs, and yield signs.
5. Pedestrian detection and tracking: Computer vision can be used to detect and track pedestrians on the road, which is important for ensuring their safety and avoiding accidents.

## Sponge height Measurement proof of concept

The deep clean machine is a specialized device designed to clean mobile devices before they are graded for cosmetic purposes. Once a mobile device is loaded into the machine, a special liquid is applied to the device's screen using a sponge that is attached to the system. The sponge is made of a material that is capable of changing its physical properties based on its wet or dry condition. When the liquid is purged into the sponge using an automated system, it expands to its maximum dimensions. This allows the sponge to hold more of the cleaning liquid and helps to ensure a more thorough cleaning of the mobile device. Conversely, when the sponge is in a dry condition, it deforms to its minimum dimension. This may make it easier to store the sponge when it is not in use or to transport it to different locations.

The deep clean machine is a specialized device that is designed to ensure that mobile devices are thoroughly and efficiently cleaned before they undergo cosmetic grading. The use of the sponge with its changing physical properties helps to ensure a consistent and effective cleaning process. The use of computer vision and image processing in the deep clean machine can help to automate the process of checking whether the sponge is completely wet after the purging of the liquid and has reached its maximum dimensions. This eliminates the need for human intervention and supervision, making the cleaning process more efficient and reliable. Computer vision is a field of artificial intelligence that enables computers to interpret and understand the visual world. It involves the use of algorithms and techniques to analyze images and extract information from them. In the context of the deep clean machine, computer vision can be used to capture images of the sponge after the liquid is purged into it. Image processing, on the other hand, involves the use of algorithms to modify and enhance images. In the case of the deep clean machine, image processing can be used to analyze the

images captured by the computer vision system and extract information about the sponge's condition.

To automate the process of checking the sponge's condition, the deep clean machine can be equipped with a camera and image processing software. After the liquid is purged into the sponge, the camera can capture an image of the sponge. The image processing software can then analyze the image to determine whether the sponge is completely wet and has reached its maximum dimensions. The software can use a range of techniques to analyze the image, such as edge detection, thresholding, and morphological operations. These techniques can help to identify the boundaries of the sponge and determine its overall shape and size. The software can also compare the size and shape of the sponge to a predefined template to determine whether it has reached its maximum dimensions. If the software determines that the sponge is not completely wet or has not reached its maximum dimensions, it can trigger an alert to the operator, indicating that the cleaning process needs to be repeated. This can help to ensure that the mobile device is properly cleaned before it undergoes cosmetic grading, improving the overall quality of the grading process. The use of computer vision and image processing in the deep clean machine is implemented help to automate the process of checking the sponge's condition after the purging of the liquid. This can eliminate the need for human intervention and supervision, making the cleaning process more efficient and reliable. By ensuring that the mobile device is properly cleaned before it undergoes cosmetic grading, the overall quality of the grading process can be improved, leading to higher customer satisfaction and better business outcomes. Implementing the automated process for checking the condition of the sponge in the deep clean machine requires the use of an industrial vision camera. Industrial vision cameras are designed specifically for industrial applications and are built to be robust and withstand harsh conditions, such as environmental factors, vibrations, and pressure. Industrial vision cameras are commonly used in manufacturing and production environments to detect defects in parts and track units throughout the production process. Their high reliability in such applications is due to their robust design and ability to perform under extreme conditions. There are two main types of industrial vision cameras: rolling shutter and global shutter. Rolling shutter cameras are suitable for stationary or slow-moving parts on a conveyor, while global shutter cameras are better suited for parts moving at high speeds. In the case of the deep clean machine, the sponge is stationary within the system, so a rolling shutter camera would be the appropriate choice.

The industrial vision camera can be mounted within the deep clean machine and positioned to capture an image of the sponge after the liquid is purged into it. The camera can be connected to a computer or processing unit that runs image processing software, which can analyze the captured image to determine the sponge's condition. The image processing software can use a range of techniques to analyze the image, such as thresholding, filtering, and feature detection. Thresholding can be used to segment the image and identify the boundaries of the sponge. Filtering can be used to remove noise and enhance the image quality. Feature detection can be used to extract specific features of the sponge, such as its size and shape. Once the image processing software has analyzed the image, it can determine whether the sponge is completely wet and has reached its maximum dimensions. If the sponge is not fully wet or has not reached its maximum dimensions, the software can trigger an alert to the operator, indicating that the cleaning process needs to be repeated. Implementing the automated process for checking the sponge's condition using an industrial vision camera offers several advantages over manual inspection. It eliminates the need for human intervention and supervision, making the cleaning process more efficient and reliable. It also reduces the risk of error and ensures consistent results. The use of an industrial vision camera and image processing software can greatly improve the performance and efficiency of the deep clean machine, ensuring that mobile devices are thoroughly and properly cleaned before they undergo cosmetic grading.

Baumer is a German-based company that specializes in the production of industrial cameras and vision systems for a variety of industrial sectors. Baumer's industrial cameras are designed to be rugged and reliable, making them well-suited for use in harsh industrial environments. They are also highly customizable, allowing for tailored solutions to meet specific application requirements. Baumer's industrial cameras offer a range of features and benefits that make them suitable for the deep clean machine application. For example, their cameras have a high frame rate, which allows for fast and accurate image capture. They also offer high resolution, which ensures that even small details can be captured and analyzed. In addition, Baumer's cameras are equipped with advanced image processing capabilities, such as automatic gain control and digital noise reduction. These features help to improve image quality and reduce noise, making it easier to analyze the captured images.

Baumer's cameras also offer a range of connectivity options, such as Gigabit Ethernet and USB, which make it easy to integrate them into existing systems. They are also compatible with a variety of software and programming languages, which allows for easy customization and integration with existing workflows. Baumer's industrial cameras are a good choice for implementing the automated process for checking the condition of the sponge in the deep clean machine. Their rugged design, advanced image processing capabilities, and flexible connectivity options make them well-suited for use in industrial applications, while their high customization capabilities allow for tailored solutions to meet specific requirements.

Baumer Camera Explorer is an intuitive GUI (Graphical User Interface) application that enables quick and easy evaluation and configuration of Baumer cameras. The software is designed to provide a clear user interface for monitoring and recording image data, as well as for configuring camera settings. One of the key advantages of Baumer Camera Explorer is its ease of use. The software provides a simple and intuitive interface that makes it easy for users to access and configure camera settings. This is particularly useful for applications where quick adjustments to camera settings may be required, such as in the deep clean machine application. Another advantage of Baumer Camera Explorer is its compatibility with a range of operating systems, including Windows, Linux, and ARM-based platforms. This makes it easy to integrate Baumer cameras into existing systems and workflows, regardless of the platform being used. Baumer Camera Explorer also provides a range of features to enhance the usability of Baumer cameras. For example, the software supports live image display, enabling users to monitor image data in real-time. It also provides support for recording and playback of image data, allowing users to review and analyze captured data. Baumer Camera Explorer is a powerful tool for configuring and integrating Baumer cameras into industrial applications. Its intuitive interface, compatibility with multiple operating systems, and advanced features make it a valuable tool for monitoring and recording image data in real- time.

The Baumer Camera Explorer provides a number of highlights that make it a valuable tool for configuring and monitoring Baumer cameras. Some of these highlights include:

1. Clear configuration of connected cameras:

The Baumer Camera Explorer makes it easy to configure connected cameras, with a clear and intuitive interface that guides users through the process of configuring camera settings.

1. Camera features under full control with the “feature-tree”:

The software provides a feature-tree view of camera settings, making it easy to access and adjust individual camera features as needed.

1. Clear and well-structured views for features like auto brightness, auto white-balance, color calculation and correction:

The software provides clear and well-structured views for common camera features such as auto brightness, auto white-balance, and color calculation and correction, making it easy to adjust these settings as needed.

1. Image diagnosis of your camera with the histogram and profile diagram available:

The Baumer Camera Explorer includes image diagnosis tools such as histograms and profile diagrams, which allow users to analyze camera images and adjust settings as needed.

1. Simple recording of videos and image series on hard disk or memory:

The software allows for simple recording of videos and image series directly to hard disk or memory, making it easy to capture and save camera footage.

1. Adaptable user interface to configure the Camera Explorer for your application:

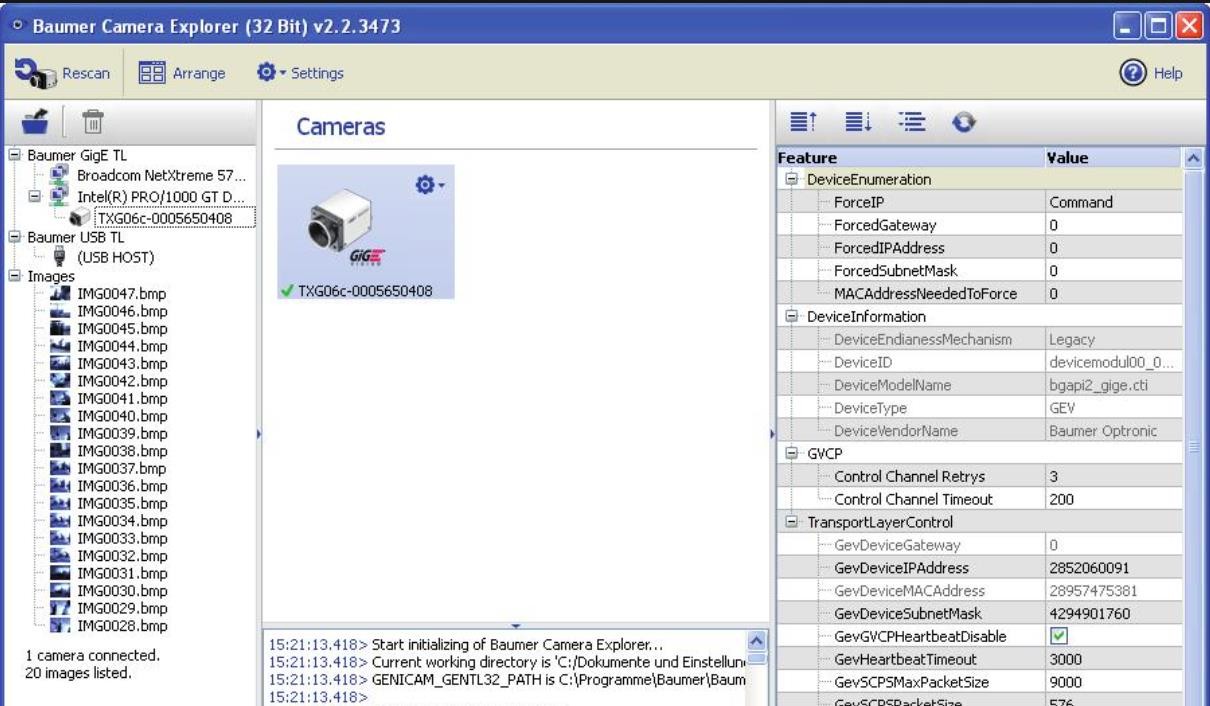
The Baumer Camera Explorer includes an adaptable user interface that can be configured to suit the needs of a specific application.

1. Monitor with multiple cameras and arrange the views on one or more screens:

The software allows for monitoring with multiple cameras, and allows users to arrange views on one or more screens for easy viewing.

1. Get to know and evaluate polarizing cameras and their calculated images:

The Baumer Camera Explorer allows users to evaluate polarizing cameras and their calculated images, making it a valuable tool for a range of applications in which polarization is important. Baumer Camera Explorer provides a range of features and tools that make it a valuable tool for configuring and monitoring Baumer cameras, with a user-friendly interface and flexible, adaptable functionality.



## Figure 6.12.1 Baumer Camera Explorer

Baumer NeoAPI is a software development kit (SDK) provided by Baumer for quick integration of their cameras into various programming languages, including C++, C, and Python. The NeoAPI simplifies the camera integration process by providing a set of libraries that abstract the low-level details of the camera interface, allowing developers to focus on the core functionalities of their application. In the case of integrating Baumer cameras with Python, the NeoAPI provides a Python wrapper library that exposes the camera functionalities as Python functions and classes, allowing developers to use the familiar syntax and idioms of Python for their camera-related code. The Python wrapper library is built on top of the C++ implementation of the NeoAPI, ensuring high performance and low latency. With the NeoAPI, developers can configure the camera settings, acquire and process images

and videos, and control camera parameters such as exposure time, gain, and white balance, all from within their Python code. This makes it easy to integrate Baumer cameras with other Python libraries and frameworks for image processing, machine learning, and computer vision.

In addition to Python, the NeoAPI also supports integration with PLCs (Programmable Logic Controllers) used in industrial automation systems. The NeoAPI provides a C interface for PLCs to interact with the Baumer cameras, allowing for seamless integration of the camera functionalities into the automation process. Baumer NeoAPI makes it easy and efficient to integrate Baumer cameras into various programming languages and platforms, enabling developers to leverage the full potential of Baumer cameras for their application. One of the highlights of NeoApi is its easy-to-understand API. With NeoApi, developers do not need to have extensive GenICam experience to integrate Baumer cameras into their applications. The API has an auto-complete function that eliminates the need to memorize different names and attributes for GenICam features.Another highlight of NeoApi is its integrated documentation. The API comes with integrated documentation that offers direct information support in the integrated development environment (IDE) during programming. This makes it easy for developers to find relevant information about the API while writing their code. NeoApi offers stable and simple integration with image processing libraries and customer systems. This means that developers can easily integrate Baumer cameras into their existing applications without worrying about compatibility issues. After integrating the Baumer camera with Python and PLC using NeoAPI, Python libraries such as OpenCV and imutils is used to perform computer vision applications and image processing techniques on the images captured by the camera. Specifically, use of these libraries is to process the images of the sponge mounted in the system. OpenCV is a popular computer vision library that provides a wide range of image and video processing functions. With OpenCV it is easy to perform operations such as image filtering, object detection, and recognition. For example, it is easy using OpenCV to detect the edges of the sponge in the image, segment the sponge from the background, or apply filters to enhance the contrast of the image.

Imutils is a library that provides a set of convenience functions for image processing tasks. It includes functions for resizing and cropping images, rotating and translating images, and performing basic geometric transformations. For example to rotate the image of the sponge to a specific angle or to resize the image to a desired size. Using these libraries, you can develop a Python program that captures images from the Baumer camera, performs image processing

techniques on them using OpenCV and imutils, and sends the processed images results to the PLC for further processing or analysis. This is particularly useful for applications such as quality control, where images of the sponge can be analyzed to ensure that it meets certain specifications or standards.

The coding environment for this proof of concept is done in Jupyter Notebook, which is an integrated development environment that allows you to write and run code in a web-based environment. Jupyter Notebook is an open-source application that is well suited for all operating systems. In this project, Jupyter Notebook is used to write and run Python code for image processing. To manage the dependencies and packages required for this project, a virtual environment is created using Conda. Conda is a package manager and environment manager that helps you to create and manage virtual environments. With Conda, you can easily create and manage multiple virtual environments on your system. Virtual environments help to keep the dependencies of different projects separate, which helps to avoid conflicts and issues. The use of Jupyter Notebook in this project is particularly useful for getting visualization of the image processing results in the code at every step. With Jupyter Notebook, you can easily display images and graphs in the code, which helps to visualize the effects of the image processing techniques being applied. This makes it easy to test and refine the image processing techniques until the desired results are achieved.

To get started with Jupyter Notebook, you can install it using pip:

```

pip install jupyterlab

```

Once you have installed Jupyter Notebook, you can start it by running the following command:

```

jupyter notebook

```

This will start Jupyter Notebook in your web browser, where you can create a new notebook and start writing code. In the Jupyter Notebook environment, you can easily install and

import the required packages and libraries, such as OpenCV and imutils. For example, to install OpenCV, you can run the following command in a code cell:

!pip install opencv-python

```

To import OpenCV in the code, you can use the following statement:

```python import cv2

```

Similarly, to install and import imutils, you can use the following commands:

```

!pip install imutils

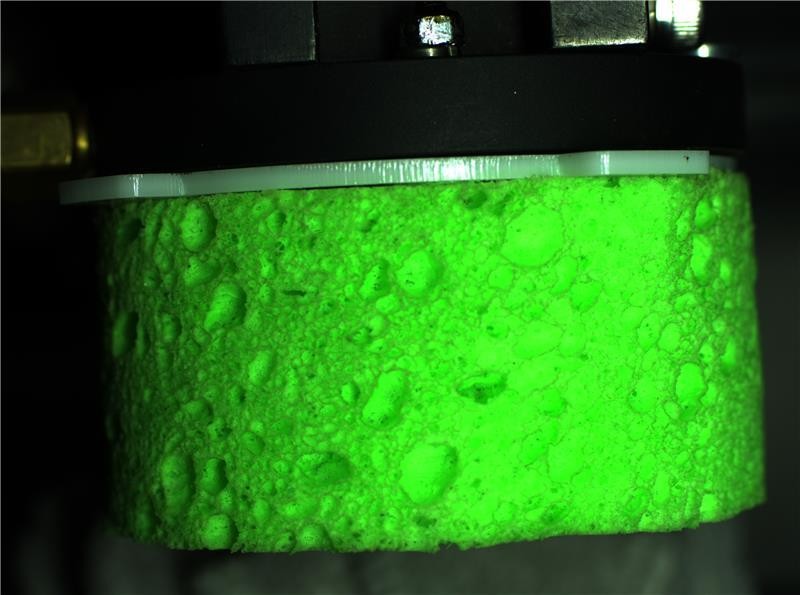
```

```python import imutils

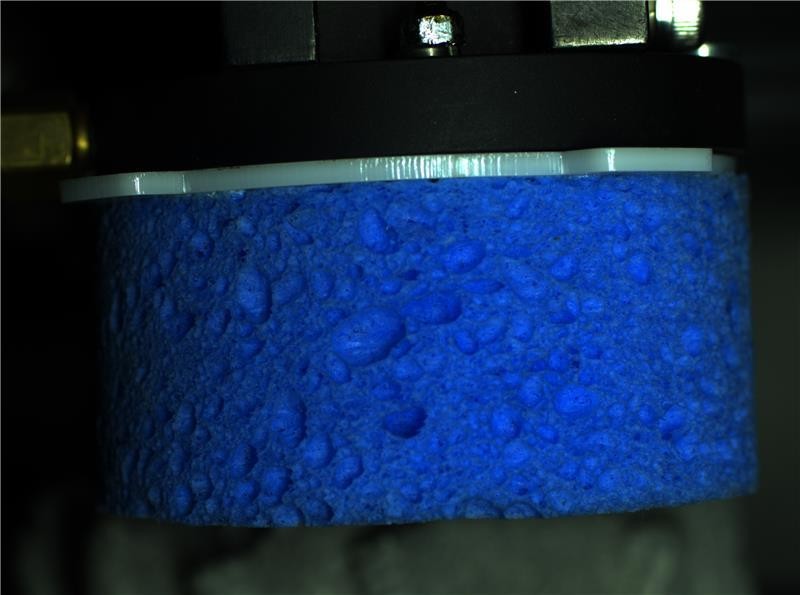
```

Once you have imported the required packages and libraries, you can start writing code to perform the image processing techniques on the images of the sponge. The code can be written in a series of cells, where each cell performs a specific operation or technique. This allows you to easily visualize the intermediate results and refine the techniques until the desired results are achieved. Jupyter Notebook and Conda are powerful tools for developing and testing image processing techniques in Python. The use of Jupyter Notebook allows you to visualize the intermediate results of the image processing techniques, which makes it easy to refine and optimize the techniques until the desired results are achieved. The use of Conda helps to manage the dependencies and packages required for the project, which helps to avoid conflicts and issues.

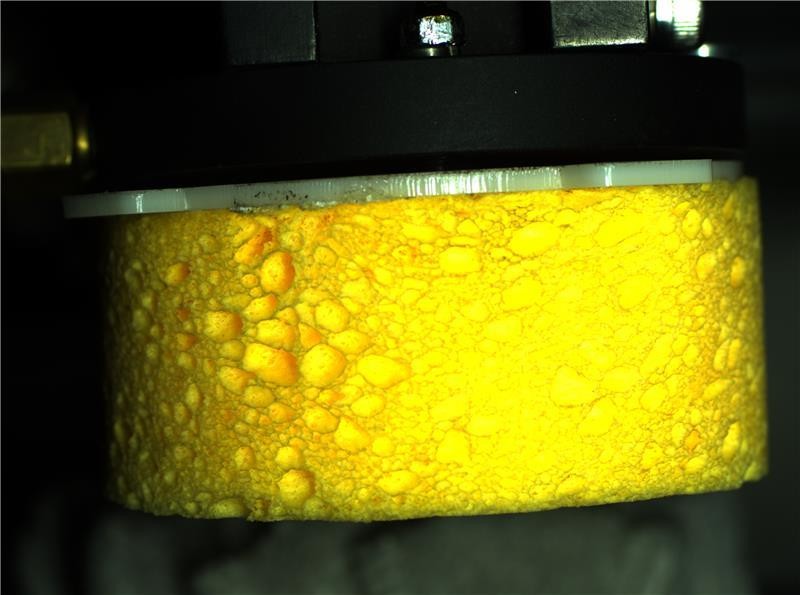
In this system there are three different colors of sponges which are used and for differentiating colors for color recognition color extraction technique is used. Color extraction is a widely used image processing technique that involves isolating specific colors from an image. In the case of the sponge mounted in the system, the images need to be processed for color extraction as the sponge is of three different colors, green, blue, and yellow. By extracting these colors from the images, we can segment the regions of interest and perform further analysis on them.



**Fig 6.12.2 Green colour sponge**



**Figure 6.12.3 Blue colour sponge**



**Fig 6.12.4 Yellow colour sponge**

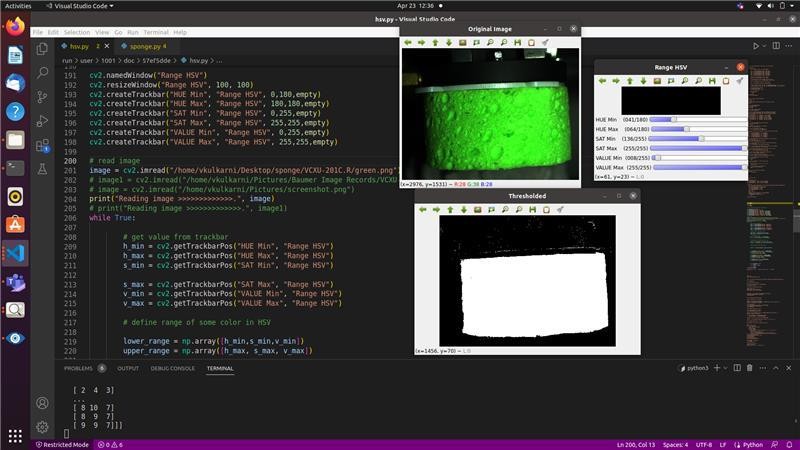
In image processing, extracting the HSV (Hue, Saturation, Value) values of different colors is an important step to perform color-based segmentation. The HSV values describe the color information in terms of hue, saturation, and value, which are more intuitive to understand and work with than the RGB (Red, Green, Blue) values.

Python provides several libraries such as OpenCV and NumPy that make it easy to extract the HSV values of different colors from an image. The `cv2.cvtColor` function in OpenCV is used to convert an image from one color space to another. In this case, we can convert the image to the HSV color space using the `cv2.COLOR\_BGR2HSV` flag.



## Fig 6.12.5 HSV output

After loading the image in the program, we can display the H, S, and V value bars using a graphical user interface (GUI) library like Tkinter or PyQt. The H value bar ranges from 0 to 179, which represents the hue component of the color. The S and V value bars range from 0 to 255, which represent the saturation and value components of the color, respectively. By tuning the HSV values properly, we can extract the exact color range of interest. To do this, we can use the `cv2.inRange` function to create a binary mask of the image where the pixels falling within the specified range are set to 1 and all other pixels to 0. The lower and upper threshold values for the H, S, and V channels are determined by experimenting with different values until the desired color range is extracted. Once we have the binary mask, we can use it to extract the regions of interest from the original image using the `cv2.bitwise\_and` function. This function applies a bitwise AND operation between the binary mask and the original image, resulting in an image that contains only the pixels that fall within the specified color range. Extracting the HSV values of different colors is an important step in image processing that allows us to perform color-based segmentation. Python provides several libraries like OpenCV and NumPy that make it easy to extract the HSV values and create binary masks of the image. By tuning the threshold values, we can extract the exact color range of interest and perform further analysis on the regions of interest.



## Figure 6.12.6 Extracting HSV values

Count NonZero function in OpenCV is used to count the number of non-zero pixels in a binary image. When applied to a binary image, this function counts the number of white pixels present in the image. In a binary image, white pixels are represented by the value 1, while black pixels are represented by the value 0. So, by using the Count NonZero function, we can count the number of white pixels in the image.

The following code uses the Count NonZero function in OpenCV to count the number of white pixels in a binary image:

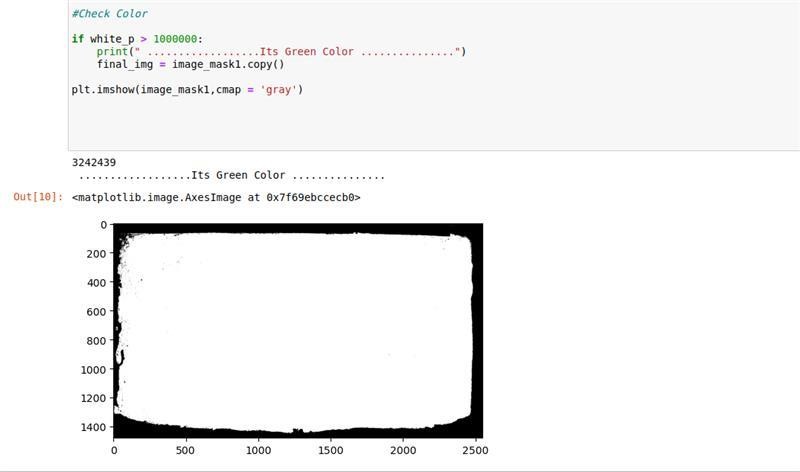
Import numpy as np

Lower\_bound = np.array([51,160,14]) Upper\_bound = np.array([65,255,255])

Image\_mask = cv2.inRange(hsv\_img, Lower\_bound, Upper\_bound) White\_pixels = cv2.countNonzero(image\_mask)

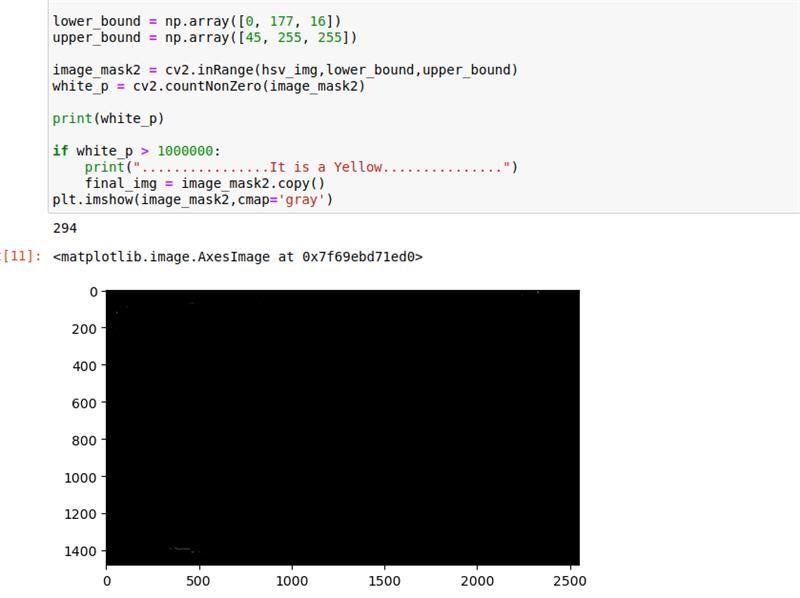
print (White\_pixels)

After getting count of white pixels in the binary image if the white pixels value is greater than some threshold value of that particular color obtained by hsv values then the sponge of that particular color is confirmed.



## Figure 6.12.7 Obtained white pixels.

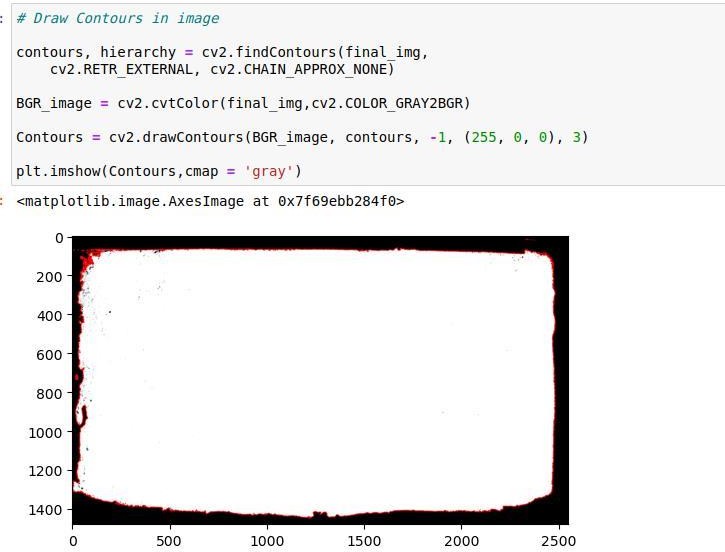
If the extracted color from the image is passed into another color hsv value range then the output is completely black pixels.



## Figure 6.12.8 Obtained black pixels

In image processing, the contour is an important feature that represents the boundary of an object in an image. Finding and drawing contours on binary images is a common task in computer vision, used for tasks such as object detection, tracking, and recognition. The next step after obtaining a binary image is to draw the contour. This is typically done using two functions in OpenCV: `findContours` and `drawContours`. The `findContours` function is used to identify the contours in the binary image, and the `drawContours` function is used to draw the contours onto the original image. The `findContours` function takes three arguments: the input image, the contour retrieval mode, and the contour approximation method. The contour retrieval mode determines the relationship between the contours and the hierarchy of the image, while the contour approximation method specifies the level of accuracy in the approximation of the contour.

Once the contours have been found using `findContours`, the `drawContours` function can be used to draw the contours onto the original image. This function takes three arguments: the input image, the list of contours, and the index of the contour to be drawn. The contour is then drawn onto the image using the specified color and thickness.



## Figure 6.12.9 Detection of contour

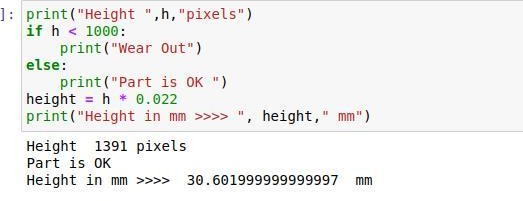
After detecting the number of contours in an image, the next step is to process each contour to extract useful information. One common approach is to calculate the area of each contour and analyze it to identify the object of interest. In Python, the OpenCV library provides a function called `cv2.contourArea()` that calculates the area of a contour in terms of pixel values.

The Python code typically iterates over all the contours in the image and calculates the area of each contour using the `cv2.contourArea()` function. If the area of a contour is very large, it is likely to represent the object of interest. In some cases, there may be multiple large contours, but the contour that represents the object of interest is likely to be the one that is closest in size and shape to the object. To further refine the selection of the object of interest, the Python code may choose the contour that is mostly of the sponge size. This is because the sponge is likely to have a distinct size and shape compared to other objects in the image. Once the contour representing the object of interest has been selected, the Python code can use the `cv2.boundingRect()` function to calculate a bounding rectangle around the contour. The bounding rectangle is a rectangle that completely encloses the contour and is defined by the x, y coordinates of the top-left corner and the width and height of the rectangle.

The process of selecting the object of interest from a set of contours involves analyzing the area and shape of each contour and using heuristics based on the specific context to identify the object of interest. Once the object of interest has been identified, a bounding rectangle can be calculated to provide additional information about the object's location and size.Once a bounding rectangle has been calculated around the contour representing the object of interest, the Python code can extract the height of the bounding rectangle in terms of pixels using the

`height` attribute of the rectangle object. The height of the bounding rectangle represents the maximum extent of the object along the y-axis.

To convert the height of the bounding rectangle from pixels to millimeters, the Python code needs to calibrate the pixels size with the actual part size. This calibration involves measuring a known distance on the object and comparing it to the corresponding distance in pixels in the image. By knowing the size of the object in the real world and its size in pixels, the Python code can determine the conversion factor between pixels and millimeters. Once the conversion factor has been determined, the height of the bounding rectangle can be converted from pixels to millimeters by multiplying the height in pixels by the conversion factor. The resulting value represents the height of the object in millimeters.



## Figure 6.12.10 Detection of height

After calculating the height of the object in millimeters, the Python code can compare this value to a predetermined threshold value to determine whether the object is OK or NOT OK. The threshold value represents the maximum allowable height of the object, and if the height of the object exceeds this value, it is considered NOT OK. If the height of the object is below the threshold value, it is considered OK. Once the status of the object has been determined, the Python code can store this information in a flag or variable. This flag can then be used to communicate with a programmable logic controller (PLC) memory on Modbus protocol using the `pymodbus` library. Modbus is a communication protocol commonly used in industrial automation systems, and the `pymodbus` library provides a Python interface for communicating with Modbus devices. By communicating the status of the object with the PLC, the Python code can generate an alarm or notification for the operator to confirm that the sponge has been completely purged and has reached its maximum dimensions. This can help ensure that the manufacturing process is running smoothly and that any issues or problems are quickly detected and resolved. The process described involves automating the task of detecting the physical dimensions of a sponge used for cleaning mobile devices using computer vision and image processing techniques. By using a camera to capture an image of the sponge and processing the image using Python code and OpenCV library functions, the height of the sponge can be calculated in millimeters, compared to a predetermined threshold value, and the status of the sponge can be communicated with a PLC using Modbus protocol. Automating this task using computer vision and image processing techniques can provide several benefits, including increased accuracy, speed, and consistency compared to manual measurements. Additionally, it can help reduce the need for human intervention and potential errors that may occur during manual measurement processes.

## Image Augmentation

Deep neural networks have revolutionized the field of machine learning, achieving state-of- the-art results in various tasks such as image classification, object detection, speech recognition, and natural language processing. However, one of the major challenges in deep learning is obtaining enough labeled training data to achieve good results and prevent overfitting. In this discussion, we will elaborate on the difficulties of collecting labeled training data and explore some techniques to mitigate these challenges. To train a deep neural network, we need a large dataset of labeled examples. Labeled data means that each data point has a corresponding label or annotation, indicating the correct output for that input. For example, in an image classification task, each image needs to be labeled with the correct class. In object detection, each object in an image needs to be labeled with a bounding box and class label. In semantic segmentation, each pixel in an image needs to be labeled with a class label.

Labeling training data is a laborious task that requires domain expertise and can be very expensive. In some cases, the labels need to be assigned by experts who are trained in the specific domain. For example, in medical imaging, a radiologist would need to examine each image and assign the correct label, which can be time-consuming and expensive. Even for less specialized tasks, labeling can be a tedious and error-prone process, which can lead to incorrect or inconsistent labels. Another challenge is collecting sufficient training data. In some domains, such as healthcare, there are legal and ethical restrictions on collecting and using data. This can make it difficult to obtain enough data to train a deep neural network. In other cases, obtaining the data may be feasible, but it can be very expensive. For example, to obtain satellite images, we need to pay for the satellite operator to take those photos. To collect images for road scene recognition, we need an operator to drive a car and collect the required data. To overcome these challenges, researchers have explored several techniques to reduce the amount of labeled training data required for training deep neural networks. One such technique is data augmentation, where we generate new training data by applying transformations to the existing data. For example, in image classification, we can apply random rotations, translations, and scaling to the images to generate new training examples. Data augmentation can help improve the model's generalization ability and reduce overfitting. Another technique is transfer learning, where we leverage a pre-trained model to extract features from the data and train a new model on top of those features. For example, in image classification, we can use a pre-trained convolutional neural network (CNN) to extract

features from the images and train a new classifier on top of those features. Transfer learning can help reduce the amount of labeled training data required and improve the model's performance.

Semi-supervised learning is another technique that can be used to train deep neural networks with limited labeled data. In semi-supervised learning, we use a small amount of labeled data and a large amount of unlabeled data to train the model. The model learns to leverage the unlabeled data to improve its performance on the labeled data. Semi-supervised learning can be particularly useful when obtaining labeled data is costly or time-consuming. Crowdsourcing is another approach that can be used to collect labeled training data. Crowdsourcing involves outsourcing the labeling task to a large group of people, who may not have domain expertise but can perform the task quickly and at a low cost. Crowdsourcing can be useful for tasks where the labels do not require domain expertise, such as image classification or sentiment analysis. Active learning is another technique that can be used to reduce the amount of labeled data required for training deep neural networks. Active learning involves iteratively selecting the most informative data points to label and adding them to the training set.Albumentations is a powerful image augmentation library that is designed to provide easy and fast data augmentation for machine learning applications. Image augmentation is a technique used in machine learning to increase the diversity and number of training data by randomly transforming the existing images. This can help to improve the generalization of the model and prevent overfitting. One of the key benefits of using Albumentations is its speed. The library is optimized for performance and can process large datasets very quickly, making it suitable for use in production environments. Albumentations provides a wide range of image augmentation techniques, including geometric transformations, color distortions, and image filters.

Some of the most commonly used augmentation techniques include:

1. Random cropping - this involves cropping a random section of the image to create a new image with a different aspect ratio.
2. Rotation - this involves rotating the image by a random angle to create a new image with a different orientation.
3. Flip - this involves flipping the image horizontally or vertically to create a new image with a different perspective.
4. Color jitter - this involves changing the brightness, contrast, and saturation of the image to create a new image with a different color distribution.
5. Gaussian blur - this involves applying a Gaussian blur filter to the image to create a new image with a different level of sharpness.

In addition to these standard augmentation techniques, Albumentations provides many other options for transforming images. For example, it supports different types of distortion, perspective transformations, and affine transformations. Albumentations also includes support for augmenting segmentation masks and keypoint annotations, which can be useful in tasks such as object detection and pose estimation. Albumentations also provides an easy-to- use API for creating image augmentation pipelines. The pipeline can be customized with specific augmentation techniques and then applied to the images in the training dataset. The library also supports parallel processing, allowing multiple transformations to be applied to different images simultaneously, which can greatly speed up the augmentation process. One of the advantages of Albumentations is that it supports both Keras and PyTorch, two of the most popular deep learning frameworks. This makes it easy to integrate the library into your existing machine learning projects. Albumentations also supports the loading and saving of image data in a variety of formats, including popular formats such as JPEG and PNG.

Pixel level transformations in more detail:

1. Blur and sharpen:

Blur is a pixel-level transformation that can be used to reduce noise in an image or create a soft focus effect. Albumentations provides various blur transformations such as GaussianBlur, MedianBlur, MotionBlur, and more. On the other hand, sharpening is the

process of increasing image contrast along the edges to create a more defined look. This can be done using the Sharpen transformation in Albumentations.

1. Histogram equalization and normalization:

Histogram equalization and normalization are pixel-level transformations that alter the distribution of pixel intensities in an image. These transformations can be used to increase image contrast, and make features more prominent. Albumentations provides HistogramEqualization and CLAHE transformations for histogram equalization and normalization, respectively.

1. Noise:

Noise is a common problem in image processing that can be caused by various factors such as camera sensors, low light conditions, and more. Albumentations provides a variety of noise transformations such as GaussianNoise, MultiplicativeNoise, and more, to simulate these conditions and create more realistic training data.

1. Color manipulation:

Color manipulation is a set of pixel-level transformations that can be used to modify the colors of an image. This can include changing the brightness, contrast, saturation, hue, and more. Albumentations provides various color manipulation transformations such as RandomBrightnessContrast, HueSaturationValue, and more. These are just a few of the most commonly used pixel-level transformations provided by Albumentations. There are many more transformations available, each with its own unique use case. By using these transformations, machine learning practitioners can create more diverse and robust training datasets, leading to better model performance. Sharpening an image enhances the edges and fine details present in the image. It works by using a kernel that highlights the edges in the image and then overlaying the resulting image with the original image to create the enhanced image.

On the other hand, blurring an image can be done in multiple ways. Gaussian Blur is one of the most popular methods. It is implemented by convolving the image with a Gaussian filter kernel. The kernel contains coefficients that determine how much each pixel should contribute to the average value of its neighbors. The standard deviation of the kernel controls how much the image should be blurred. The larger the standard deviation, the more blurred the image will be. Another way to blur images is using the Median Blur operation. This

technique replaces each pixel in the image with the median value of the neighboring pixels. It is often used to remove salt and pepper noise from images, as it smoothens out the small isolated points that are created by this type of noise. Overall, blurring and sharpening operations can be very useful in image processing, as they can help to remove noise and enhance important features in an image. Albumentations offers a variety of kernels and methods for performing these operations, making it easy to include them in your image augmentation pipeline.

Blurring images is a common technique in image processing, which involves smoothing out sharp edges and rapid changes in pixel intensity by averaging out the pixel values in a certain area of the image. Albumentations offers different blurring operations, including Blur, AdvancedBlur, GaussianBlur, and MedianBlur. GaussianBlur is the most commonly used blur transformation because it works well with images that have Gaussian noise. On the other hand, MedianBlur is a better option for images with salt-and-pepper noise. AdvancedBlur is the most customizable blur operation, but it is not often necessary to spend time optimizing it, as the standard GaussianBlur transformation is sufficient for most use cases. Gaussian noise is a common type of noise that occurs in images due to various factors, including camera sensor noise, image compression, and transmission. It has a probability density function similar to that of the normal distribution, which makes it a good approximation of many real- life scenarios. Albumentations also offers ISONoise and MultiplicativeNoise, which can be useful in certain situations, but Gaussian noise is the most commonly used type of noise.

Color manipulation

There are different ways of manipulating colors in an image. That is a vary common procedure that a lot of images go through before being fed into a model. If the color itself is not in any way connected to that problem ToGray and ToSepia are simple transformations that convert an image to grayscale or sepia tone, respectively. These are basic color manipulations that can be used to change the look of an image without drastically altering it. RandomBrightnessContrast is a transformation that randomly changes the brightness and contrast of an image. This can be useful when trying to create augmented images for training a model.

HueSaturationValue (HSV) is a color space that separates the hue, saturation, and brightness of an image. By manipulating the values of these channels, you can change the overall color

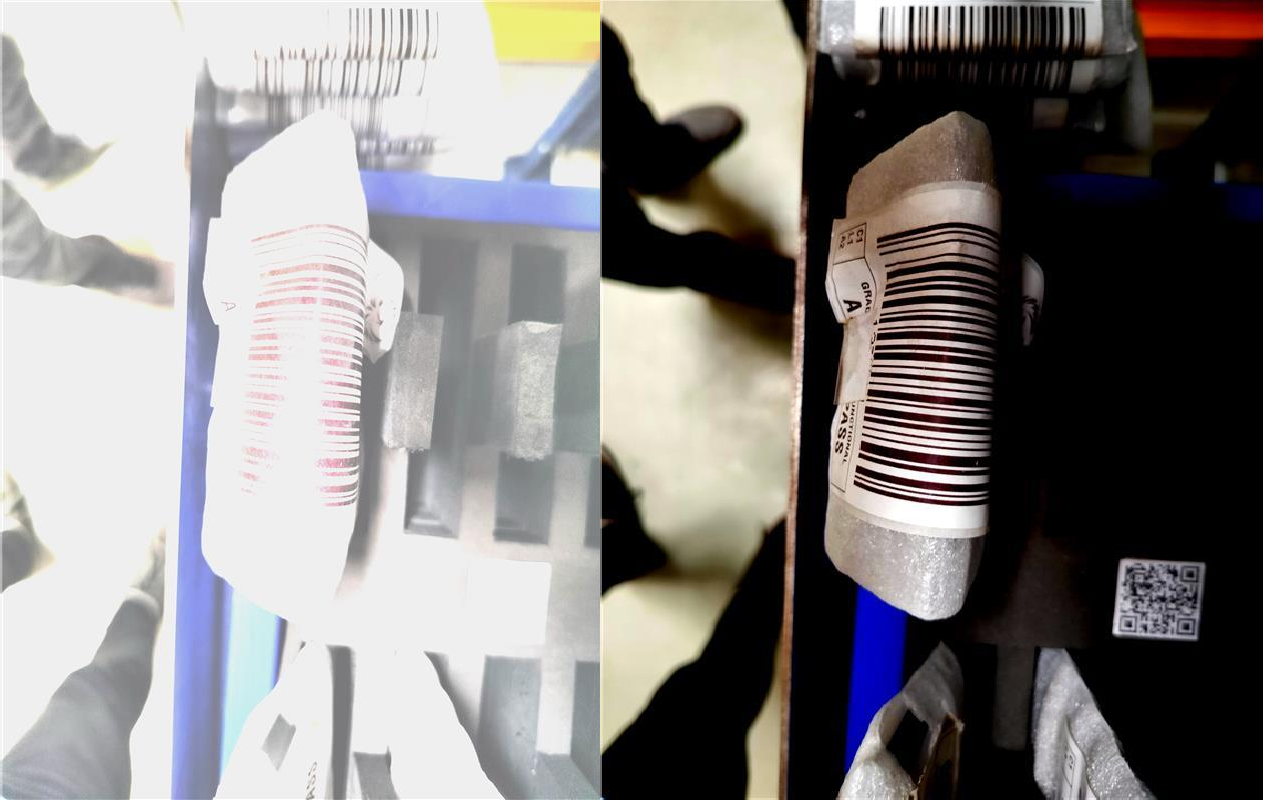
of an image. The HueSaturationValue transformation allows you to adjust these values randomly, which can help create more diverse training data for your model.

ColorJitter is a more advanced transformation that allows you to adjust the brightness, contrast, saturation, and hue of an image. This can be a powerful tool when trying to create diverse training data, as it allows you to make a wide range of changes to an image's color.

FancyPCA is a transformation that performs Principal Component Analysis (PCA) on the pixel values of an image. This can be used to create a more diverse set of training data by randomly changing the color distribution of an image.

Random brightness

When using a random transformation like `RandomBrightnessContrast`, the resulting images will differ slightly each time the code is run due to the randomness of the transformation. These small differences may not be noticeable to the human eye, but they can still affect the performance of machine learning models that are trained on these images. To minimize the impact of these small differences, it's a good practice to use data augmentation techniques such as random transformations during both training and validation. This helps to ensure that the model learns to be robust to small variations in the input data.

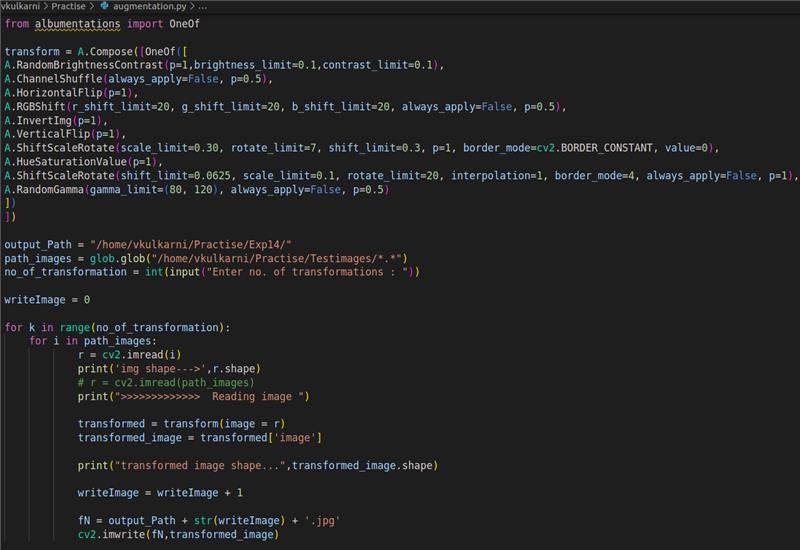


## Fig 6.11.11 Images of barcode label

An augmentation pipeline is a set of image transformation operations that are applied in sequence to augment an input image. These operations can include various types of geometric and color transformations such as rotation, scaling, flipping, cropping, adjusting brightness, contrast, hue, saturation, and so on. The pipeline is usually defined as a list of transformation functions with their respective parameters. Each transformation can be assigned a probability of being applied and a range of magnitudes to control the intensity of the transformation. For example, a rotation operation may have a probability of 0.5 and a magnitude range of -45 to

+45 degrees. The unified interface refers to a standard set of methods that provide a common way of applying image augmentations across different datasets and models. This interface typically includes functions for loading and preprocessing images, applying augmentations, and returning augmented images or batches.In practice, augmentation libraries such as Albumentations, imgaug, and torchvision provide a declarative way of defining augmentation pipelines and a unified interface for applying them. These libraries allow users to define a pipeline of transformations with specific parameters, store and load them to/from different formats, and apply them to images or batches of images in a consistent way.

By using an augmentation pipeline and a unified interface, data scientists and machine learning engineers can easily experiment with different augmentation strategies, compare their effects on model performance, and ultimately improve the generalization and robustness of their models.



## Fig 6.11.12 Python program of Image augmentaion

RandomBrightnessContrast, ChannelShuffle, HorizontalFlip, RGBShift, InvertImg, VerticalFlip, ShiftRotate, HueSaturationValue, ShiftRotateScale, and RandomGamma are some of the commonly used image augmentation techniques in computer vision applications. Discussing each of them in detail:

1. RandomBrightnessContrast: This transformation randomly adjusts the brightness and contrast of the image. This can help simulate the effect of different lighting conditions and improve the model's robustness to variations in brightness.
2. ChannelShuffle: This transformation randomly shuffles the color channels of the image. This can help reduce the model's reliance on color information and make it more invariant to different color distributions.
3. HorizontalFlip: This transformation flips the image horizontally. This can help simulate the effect of objects viewed from different angles and improve the model's ability to recognize objects in different orientations.
4. RGBShift: This transformation randomly shifts the values of the red, green, and blue channels of the image. This can help simulate the effect of changes in lighting conditions and improve the model's robustness to different color distributions.
5. InvertImg: This transformation inverts the colors of the image. This can help simulate the effect of negative images and improve the model's ability to recognize objects with high contrast.
6. VerticalFlip: This transformation flips the image vertically. This can help simulate the effect of objects viewed from different angles and improve the model's ability to recognize objects in different orientations.
7. ShiftRotate: This transformation randomly shifts, rotates, and scales the image. This can help simulate the effect of changes in the camera angle, position, and scale, and improve the model's robustness to different image sizes and orientations.
8. HueSaturationValue: This transformation randomly adjusts the hue, saturation, and value of the image. This can help simulate the effect of changes in the lighting conditions and improve the model's ability to recognize objects in different color distributions.
9. ShiftRotateScale: This transformation randomly shifts, rotates, scales, and shears the image. This can help simulate the effect of changes in the camera angle, position, scale, and perspective, and improve the model's robustness to different image geometries.
10. RandomGamma: This transformation randomly adjusts the gamma value of the image. This can help simulate the effect of different lighting conditions and improve the model's robustness to variations in brightness.

Overall, these image augmentation techniques can significantly improve the performance of computer vision models, especially when dealing with real-world scenarios where the input data can vary widely. By applying such transformations, the models can learn to better recognize and classify objects in different lighting conditions, orientations, sizes, and color distributions.

## FANUC M-10id/12 Robot

The Fanuc M10iD-12 is a compact six-axis industrial robot manufactured by Fanuc, a global leader in factory automation technology. This robot is designed for a range of material handling, assembly, and dispensing applications.

Some of the key specifications of the Fanuc M10iD-12 include:

Payload capacity: 12 kg Maximum reach: 1420 mm Repeatability: +/- 0.02 mm Number of axes: 6

Maximum speed: 6000 degrees per second Power supply: 200-240V, 50/60 Hz Weight: 130 kg

The Fanuc M10iD-12 is a high-performance robot that is ideal for a wide range of industrial applications. It is known for its accuracy, speed, and reliability, which makes it a popular choice among manufacturers worldwide.

Some of the features and benefits of the Fanuc M10iD-12 include:

Compact design: The M10iD-12 has a small footprint, which makes it ideal for use in tight spaces.

High speed and precision: The robot can move at high speeds and offers high precision, which makes it ideal for applications that require fast and accurate movements.

Easy to program: The M10iD-12 is easy to program, thanks to its intuitive interface and user- friendly software.

Versatile: The robot can be used for a variety of applications, including material handling, assembly, dispensing, and more.

Reliable: The M10iD-12 is designed for maximum uptime, with features like durable components and a maintenance-free design.

Overall, the Fanuc M10iD-12 is a versatile and high-performance industrial robot that can help manufacturers improve their efficiency and productivity.



## Figure 6.12.1 Fanuc M-10iD/12 Robot

Teach Pendant and Controller

The teach pendant is a handheld device that is used to program and control the Fanuc M10iD- 12 robot. It allows the operator to move the robot through its various motions and to program it to perform specific tasks.

The teach pendant for the M10iD-12 is designed with a color touchscreen display and an intuitive interface, which makes it easy to use and program the robot. The pendant has a

simple layout, with buttons and controls that allow the operator to start and stop the robot, move it in different directions, and adjust its speed and positioning.

One of the benefits of the teach pendant for the M10iD-12 is that it allows for quick and easy programming. The operator can use the pendant to move the robot through its various motions, and to record those motions for future use.



## Figure 6.12.2 Teach pendant and controller

Robot Programming

FANUC robots can be programmed using a variety of methods, including the teach pendant and programming software.

The teach pendant is a handheld device that allows operators to teach the robot how to perform different tasks. The operator moves the robot arm to the desired position and then records the position in the robot's memory. The robot can then repeat the task on its own.

Offline programming software allows operators to program the robot using a computer. They can create a virtual model of the robot and program it to perform tasks using a graphical user interface. The program can then be transferred to the robot's controller.

Safety, Maintenance and Troubleshooting

Safety is a critical consideration when working with FANUC robots. Operators must be properly trained on how to work with the robot safely, and safety programming must be implemented to ensure that the robot operates within safe parameters.

Physical barriers, such as fencing, can also be used to separate the robot from human workers and prevent accidents.

Regular maintenance is essential to keep FANUC robots running smoothly and efficiently. This can include tasks such as cleaning the robot, lubricating the joints, and replacing worn- out parts.

If problems arise, troubleshooting may be necessary. This can involve diagnosing the issue, repairing or replacing the faulty component, and reprogramming the robot if necessary.

Co-Ordinate System

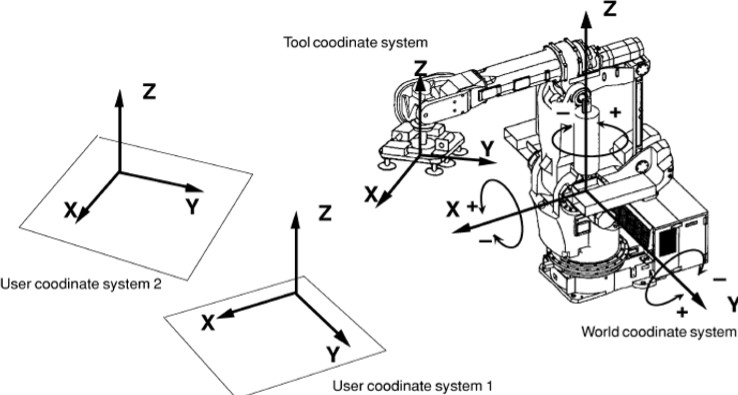
We have two co-ordinate systems, namely Joint Co-ordinate System and Cartesian co- ordinate system.

The joint coordinate system defines the position of the robot's joints, which are the points where the robot arm segments connect. Each joint has a specific angle, and by controlling the angle of each joint, the robot can move to a particular location in space. The joint coordinate system is often used for precise positioning and can be very useful when performing tasks that require a high degree of accuracy.

The Cartesian coordinate system, on the other hand, defines the position of the robot's end effector in three-dimensional space. This is done using three coordinates that define the robot's position in X, Y, and Z axes. By moving the end effector to specific coordinates, the robot can move to a particular location in space. The Cartesian coordinate system is often used for tasks that require a simpler approach to positioning, such as moving objects from one location to another.

FANUC robots can switch between the joint and Cartesian coordinate systems as needed, depending on the task at hand. This allows for maximum flexibility and efficiency in robot programming and operation.

It's also important to note that FANUC robots can be programmed using a variety of coordinate systems, including tool, work, and user frames. These frames can be used to define specific points or objects in space, making it easier to program the robot for specific tasks.



## Figure 6.12.3 Coordinate system -World/tool/user

Robot Jogging

Jogging is a feature in FANUC robots that allows an operator to manually control the robot's movement using a handheld device called a teach pendant. The teach pendant is equipped with various buttons and controls that allow the operator to jog the robot in different directions and at different speeds. There are several different jogging modes available in FANUC robots, including joint jogging and Cartesian jogging. In joint jogging, the operator can jog each joint individually, adjusting the angle of each joint to move the robot in a specific direction. In Cartesian jogging, the operator can jog the robot in X, Y, and Z directions, allowing for movement in a straight line in any direction. Jogging can be useful for a variety of tasks, such as manually moving the robot to a specific location, or for fine- tuning the robot's position during programming. Jogging can also be used to test the robot's movements and ensure that it is working correctly before running a program.

It's important to note that jogging should only be performed by trained operators who understand the potential risks involved in operating a robot. Safety precautions should always

be taken, such as using protective barriers and ensuring that the robot is in a safe mode before jogging. Additionally, FANUC robots have various safety features built in, such as emergency stop buttons and collision detection, to help prevent accidents during jogging and other types of robot operation.

Teach points

Teaching points are specific positions in space that are programmed into a FANUC robot's memory. These points can be used as reference positions for the robot when performing tasks, and can be taught to the robot in a variety of ways.

One way to teach a point is to manually jog the robot to the desired position and then save that position as a teaching point. This can be done using the teach pendant, by pressing a button to save the current position as a point. The robot will then store the coordinates for that point in its memory, along with any other parameters that may be associated with that point, such as speed and orientation.

Teaching points can be very useful when programming a FANUC robot, as they provide a way to quickly and accurately position the robot for specific tasks. For example, if a robot is used to pick up parts from a conveyor belt, teaching points can be used to ensure that the robot always moves to the correct location to pick up each part.

FANUC robots can store thousands of teaching points in their memory, making it possible to program complex tasks that require a large number of reference positions. Additionally, teaching points can be edited or deleted as needed, making it easy to update robot programs or adjust positions as necessary.

Motion, termination types and I/O’s

FANUC robots can perform a variety of different types of motion, each of which is designed for specific applications. Some common types of motion include:

Linear motion: This is motion in a straight line, and is often used for tasks that require the robot to move from one location to another in a straight path.

Joint motion: This is motion in a circular path, and is often used for tasks that require the robot to follow a curved path or to move around an object. It will choose the shortest path.

In FANUC robots, "Fine" and "CNT" are two types of termination points that can be used to control the robot's movements. A fine point is a reference point that is used to fine-tune the robot's position in a specific area. When the robot is programmed to move to a fine point, it will slow down and approach the point more slowly than it would for a regular point. This allows the robot to make small adjustments to its position to ensure that it is precisely aligned with the reference point.

A CNT point, on the other hand, is a count point that is used to control the speed and timing of the robot's movements. When the robot is programmed to move to a CNT point, it will move at a constant speed until it reaches the CNT point, at which point it will pause briefly before continuing on to the next point. This can be useful for tasks that require the robot to pause for a specific amount of time before moving on to the next step.

Both fine and CNT points can be programmed using the FANUC teach pendant or through the use of a programming language such as Karel. By using these termination points in the robot's programming, operators can achieve precise control over the robot's movements, allowing it to perform complex tasks with a high degree of accuracy and repeatability.

FANUC robots have a variety of input and output signals (I/Os) that can be used to control the robot's actions and communicate with external devices. Some common types of I/Os in FANUC robots include:

Digital inputs: These are used to detect signals from external devices, such as sensors or switches. They are typically used to trigger specific robot actions or to detect the presence of objects in the robot's workspace.

Digital outputs: These are used to send signals to external devices, such as valves or relays. They are typically used to control the actions of external devices or to provide feedback signals to other systems.

Analog inputs: These are used to measure analog signals from external devices, such as temperature or pressure sensors. They are typically used to provide the robot with feedback on its environment, or to control specific robot actions based on the measured signal.

Analog outputs: These are used to send analog signals to external devices, such as motors or solenoids. They are typicallyl used to control the speed or position of external devices, or to provide precise control over robot actions.

**(Annexure 4.6)**

**BIBLIOGRAPHY**

|  |  |
| --- | --- |
|  |  |
| **1.** | Mallikarjun G Hudedmani, R.M Umayal, Shiva Kumar Kabberalli, Raghavendra Hittalamani, Programmable Logic Controller in Automation, Advance Journal of Graduate Research 2017 |
| **2.** | Ogawa, M., & Henmi, Y. Recent Developments on PC+PLC based Control Systems for Beer Brewery Process Automation Applications. 2006 SICE-ICASE International Joint Conference. doi:10.1109/sice.2006.315748 |
| **3** | BAYGIN, M., KARAKOSE, M., SARIMADEN, A., & AKIN, E. (2017). Machine  vision based defect detection approach using image processing. 2017 International Artificial Intelligence and Data Processing Symposium  (IDAP). doi:10.1109/idap.2017.8090292 |
| **4** | Yang, L., Chen, L., & Liu, L. (2020). Design of Automatic Inspection System Based on PLC and Machine Vision. Proceedings of the 3rd International Conference on Information Technologies and Electrical Engineering. doi:10.1145/3452940.3453007 |
| **5** | David Vermont. Machine Vision: Automated Visual Inspection and robot vision. Association for computing machinery. 2007 |
| **6** | M Maghnie, F Stegemerten. Cloud-Based Hardware on Loop Testing of building automation controllers. Journal of Physics.2021 |
| **7** | Carter Liam, Security Analysis of a Beckhoff CX- 9020 Programmable Logic Controller. IS. 202 |
| **8** | Krister. Simulation of field devices in the testing of a marine control system. LUT.2021 |
| **9** | Dahlberg, Emil Replacing Setpoint Control with Machine Learning: Model Predictive Using ANN. DIVA. 2020.Ming Xue, Changjun. The Socket Programming and Software Design for communication-Based on Client/Server. IEEE.2009 |

|  |  |
| --- | --- |
| **10** | Qigang Liu, Xiangyang. Research on Web Real-Time Communication Based on WebSocket. International Journal of Communication, Network and Systems Sciences |
| **11** | Ronald Cordova, Balaji Sudhamurty. Design and Implementation of Client-Server Based Application Using Socket Programming.IEEE.2017 |
| **12** | Yanmeing Gou, Song Wou. Deep Learning for Visual Understanding. Science Direct 2016 |
| **13** | Diwan, Anirudh, J.V, Object Detection using YOLO: Challenges, Architectural Successor, dataset and applications. Spinger.2022 |
| **14** | Bochkovskiy A, Wang CY, Liao HY. YOLO V4: Optimal speed and accuracy of object detection. 2004 |
| **15** | Borisyuk F, Gordo A, Sivakumar V (2018) Rosetta: large scale system for text detection and recognition in images. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery data mining |
| **16** | Agarwal S, Terrail JO, Jurie F (2018) Recent advances in object detection in the age of deep convolutional neural networks. |
| **17** | He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770-778 |
| **18** | Akansha Bhahija. Visual object detection and Tracking Using YOLO and SORT. IT, 2019 |
| **19** | S. Moon, J.Lee, D.Nam, H.kim and W. Kim, “A comparative study on multi-object tracking methods for sports events”,19th International Conference on Advances Communication Technology (ICACT)2017 |
| **20** | S.Shinde, A.kothari and V.Gupta, “Yolo-based Human Action Recognition and Localization”, Procedia Computer Science, vol.133,2013 |

|  |  |
| --- | --- |
| **21** | D. Parthasarathy,”A Brief History of CNN in Image Segmentation: From R-CNN to Mask R-CNN”,Medium,2019. Available:https://blog.athelas.com/a-brief-history-of- cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4. |
| **22** | LabelImg,"Tzutalin.github.io,2019.[Online].Available:https://tzutalin.github.io/labelI mg/. |