# DEEP LEARNING MODEL FOR DETECTING DISEASES IN TEA LEAVES

## **NAAN MUDHALVAN PROJECT REPORT**

***Submitted by***

|  |  |
| --- | --- |
| **Karthick V** | **111420104043** |
| **Gokul R** | **111420104029** |
| **Abishek revanth**  **Akash S** | **111420104004**  **111420104005** |

***in partial fulfilment for the award of the degree of***

# BACHELOR OF ENGINEERING

## **IN**

**COMPUTER SCIENCE ENGINEERING**

**PRATHYUSHA ENGINEERING COLLEGE THIRUVALLUR – 602 025**

# ANNA UNIVERSITY : CHENNAI 600 025

**MAY 2023**

**ABSTRACT**

Plastic is an unavoidable material in the world. It is the first man-made material that mother nature couldn’t destroy that easily. Humans are fully now dependent on plastics, without this material most of the products in our world will not exist. As the demand for plastics increases day by day, on the other side the disposal of these materials is now itself a question mark. Scientists are been in research the safe disposal of plastics. As a part of that, we people should do some basic work like separating the plastic materials from the garbage in our house. But most the people are not doing it due to many reasons like laziness, disgusting of their waste. Due to this, the garbage men do the segregation process with their bare hands. This will cause many diseases and infections to them. Our nation has now been moving forward with **“Clean India”.** As a part of this, we have developed a technique to separate the plastics with the help of robots and Deep learning technology. Here we have developed a robotic arm that is programmed with a deep neural network and is capable of identifying the plastic and non-plastic materials. It can separate plastics of any type from the garbage and store them separately. This will reduce the manpower required for the separation of the plastics from the garbage. Our project will be a big step toward increasing the robot density in our country and also increases the usage of artificial intelligence in small-scale operations.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ABSTRACT iv**

**LIST OF TABLES viii**

**LIST OF FIGURES viii**

**LIST OF ABBREVIATIONS ix**

1. **INTRODUCTION 1** 
   1. OVERVIEW 1
   2. DEEP LEARNING 2

1.2.1 Convolutional Neural Networks 3

1.2.2 Computer Vision 4

1.2.2.1 Image Classification 5

1.2.2.2 Object Detection 5

1.2.2.3 Object Tracking 6

1.2.2.4 Semantic Segmentation 6

1.2.2.5 Instance Segmentation 7

1.3 ROBOTICS 7

1.3.1 Types of Robots 8

1.3.1.1 Autonomous Mobile Robots 8

1.3.1.2 Automated Guided Vehicles 8

1.3.1.3 Robotic Arms 9

1.3.1.4 Humanoid Robots 9

1.3.1.5 Hybrids 9

1.3.1.6 Cobots 9

v

1.4 EMBEDDED MACHINE LEARNING 10

1.5 OBJECTIVES OF THE PROJECT 11

**2. LITERATURE REVIEW 12**

2.1 EXISTING SYSTEM 15

2.2 DISADVANTAGES OF 15

THE EXISTING SYSTEM

2.3 PROPOSED SYSTEM 16

2.4 ADVANTAGES OF 16

PROPOSED SYSTEM

**3. SYSTEM SPECIFICATIONS 17**

3.1 HARDWARE 17

3.2 SOFTWARE 18

3.2.1 Google Colab 18

3.2.2 Pycharm IDE 19

3.2.3 Python Packages 20

3.2.4 Arduino IDE 22

**4. SYSTEM ANALYSIS 23**

4.1 DATASET PREPARATION 23

4.1.1 Data Collection 23

4.1.2 Data Cleaning 25

4.1.3 Data Annotation and Labelling 26

4.1.4 Data Splitting 27

4.1.5 Pre-Processing 29

4.1.6 Data Augmentation 30

4.2 TRAINING 31

vi

**5. ARDUINO SERIAL COMMUNICATION 34**

5.1 WORKING 34

**6. RESULTS AND DISCUSSION 37**

6.1 TRAINING RESULTS 38

**7. CONCLUSION 41**

**APPENDIX 42**

**REFERENCES 55**

vii

**LIST OF TABLES**

**FIGURE NO. TITLE PAGE NO.**

4.4 Splitted Dataset 28

**LIST OF FIGURES**

**FIGURE NO. TITLE PAGE NO.**

1.1 Convolutional Neural Network Architecture 4

1.2 Computer Vision Techniques 5

1.3 Object Detection 6

4.1 Dataset Images 25

4.2 Annotating the image in Roboflow 26

4.3 Annotated Dataset 27

4.5 Pre-processing Options in Roboflow 29

4.6 Data Augmentation Options in Roboflow 30

4.7 Selecting the Dataset Type 32

5.1 Flowchart 35

5.2 Circuit Diagram 36

5.3 Plastic Removing AI Robot 36

6.1 Tensorboard results 37

6.2 Performance Curves 38

6.3 Labels Correlogram 38

6.4 Anaconda Navigator 39

6.5 Anaconda \_Prompt 39

viii

6.6 Output 40

6.7 Camera Output 40

**LIST OF ABBREVIATIONS**

SWM – Solid Waste Management

AI – Artificial Intelligence

CNN – Convolutional Neural Network

AMR – Autonomous Mobile Robots

AGV – Automated Guided Vehicles

ML – Machine Learning

IOT – Internet of Things

FGPA – Field-Programmable Gate Array

ASIC – Application Specific Integrated Circuit

AP – Asia Pacific

UNEP-AR – United Nations Environment Programme

PET – Polyethylene Teraphthalate

PE – Polyethylene

PVC – Polyvinyl Chloride

PP – Polypropylene

PS – Polystyrene

ABS – Acrylonitrile Butadiene Styrene

PLC – Programmable Logic Controllers

YOLO – You Only Look Once

PWM – Pulse Width Modulation

USB – Universal Serial Bus

GPU -Graphics Processing Unit

VCS – Version Control System

VIM – Vi IMitation

API – Application Programming Interface

ix

RTSP URL – Real time Streaming Protocol Uniform Resource Locater

OSX – Operating System X version

BSD – Berkely Software Distributuion

POSIX – Portable Operating System Interface

YAML – Yet Another Markup Language

cuDNN – Computed Unified Device Architecture Deep Neural Network

IDE – Integrated Development Environment

COM – Communication Port

PC – Personal Computer

x

**CHAPTER 1**

**1. INTRODUCTION**

**1.1 OVERVIEW**

Since the 20th century, we have depended on plastic as an affordable, versatile, and durable material. However, since the majority of plastic materials take centuries to degrade, all of the plastic that has been sent to landfills in the UK still exists – and yet we’re still producing and consuming more of it. That plastic has to go somewhere, and it’s frequently either dumped carelessly on land or in rivers in developing countries, before ending up in the ocean, where it threatens marine life. The fact is, we simply can’t cope with the amount of plastic on our planet – nor the amount that continues to be produced. For this reason, our attitudes and behaviors towards plastic must change to ensure a safe and healthy future for our planet.

According to the United Nations, at least 800 species worldwide are affected by marine debris, and as much as 80 percent of that litter is plastic.’ Marine animals can either get caught in plastic objects (such as the plastic rings that hold drinks cans together), ingest the plastic, and/or be exposed to plastic chemicals. Which can alter their physiology over time. Six decades ago, mass production of plastics began – accelerating so rapidly that it has created 8.3 billion tonnes of plastic – and over 90% of it isn’t recycled.

Source segregation is globally acknowledged as a key aspect that will result in effective and sustainable solid waste management for cities and towns,

1

reducing the quantum of waste and the financial and environmental cost of maintaining large, sprawling legacy waste sites. As per the solid waste management (SWM)rules, everyone who generates waste has the responsibility to segregate wet waste from dry waste before disposal, ensuring the separation of domestic hazardous waste too. This waste is ultimately thrown into municipal waste collection centers from where it is collected by the area municipalities to be further thrown into the landfills and dumps. However, either due to a resource crunch or inefficient infrastructure, not all this waste gets collected and transported to the final dumpsites.

If at this stage the management and disposal are improperly done, it can cause serious impacts on health and problems to the surrounding environment. To solve this problem, we should implement robots for this task. AI robots should be used to see and sort the waste materials so that they can be sorted as degradable and non-degradable waste. With a more advanced neural network, we can segregate different types of plastics which are suitable for recycling. This can be achieved with the help of deep learning technology, which will train the robot to sort the plastics from the waste. This will reduce the manpower and health effects.

**1.2 DEEP LEARNING**

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain – albeit far from matching its ability – allowing it to “learn” from a large amount of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

2

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit fraud detection) as well as emerging technologies (such as self-driving cars).

**1.2.1 Convolutional Neural Network**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets can learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

**Types of CNN Models:**

* LeNet
* AlexNet
* ResNet
* GoogleNet
* MobileNetV1
* ZfNet

3

Convolution

Input

Convolution

Convolution

Dense

Output

Dense

Max Pooling

Dropout

Max Pooling

Max Pooling

3

**Fig 1.1 Convolutional Neural Network Architecture**

**1.2.2 Computer Vision**

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs – and take actions or make recommendations based on that information.

4

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognizes images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.

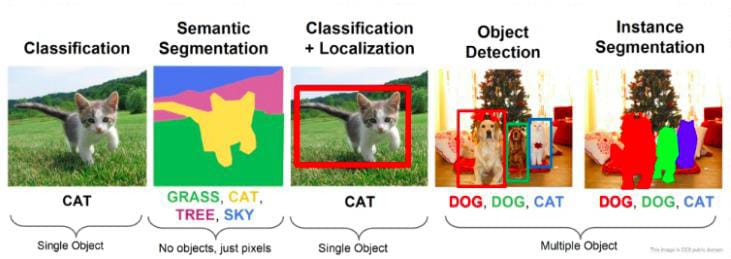


Fig 1.2 Computer Vision Techniques

**1.2.2.1** **Image Classification**

**Image classification** sees an image and can classify it (a dog, an apple, a person’s face). More precisely, it can accurately predict that a given image belongs to a certain class. For example, a social media company might want to use it to automatically identify and segregate objectionable images uploaded by users.

**1.2.2.2 Object Detection**

**Object Detection** can use image classification to identify a certain class of images and then detect and tabulate its appearance in an image or video. Examples include detecting damage to the assembly line or identifying machinery that requires maintenance. In our project, we used this Object Detection method to sort the plastics from other materials.

5

**1.2.2.3 Object Tracking**

**Object tracking** follows or tracks an object once it is detected. This task is often executed with images captured in sequence or real-time video feeds. Autonomous vehicles, for example, need to not only classify and detect objects such as pedestrians, other cars, and road infrastructure, they need to track them in motion to avoid collisions and obey traffic laws.

****

Fig 1.3 Object Tracking

**1.2.2.4 Semantic Segmentation**

**Semantic Segmentation** is a Deep learning algorithm that associates a label or category with every pixel in an image. It is used to recognize a collection of pixels that form distinct categories. For example, an autonomous vehicle needs to identify vehicles, pedestrians, traffic signs, pavement, and other road features. Semantic Segmentation is used in many applications such as automated driving, medical imaging, and industrial inspection.

U-Net is **a** semantic segmentation technique originally proposed for medical imaging segmentation.

6

**1.2.2.5 Instance Segmentation**

Instance Segmentation is identifying each object instance for every known object within an image. Instance segmentation assigns a label to each pixel of the image. It is used for tasks such as counting the number of objects. Instance Segmentation is identifying each object instance for every known object within an image. Instance segmentation assigns a label to each pixel of the image. It is used for tasks such as counting the number of objects. Instance segmentation.

**1.3 ROBOTICS**

The application of deep learning to robotics over the past decade has led to a wave of research into deep artificial neural networks and to very specific problems and questions that are not usually addressed by the computer vision and machine learning communities. Robots have always faced many unique challenges as the robotic platforms move from the lab to the real world.  Minutely, the sheer amount of diversity we encounter in real-world environments is a huge challenge to deal with today’s robotic control algorithms and this necessitates the use of machine learning algorithms that can learn the controls of a given data. However, deep learning algorithms are general non-linear models capable of learning features directly from data making them an excellent choice for such robotic applications. Indeed, robotics and artificial intelligence (AI) are increasing and amplifying human potential, enhancing productivity, and moving from simple thinking to human-like cognitive abilities.

7

**1.3.1 Types of Robots**

As robotics manufacturers continue to deliver innovations across capabilities, price, and form factors, robotics solutions are being implemented in an ever-increasing number of industries and applications. Advancements in processing power and AI capabilities mean that we can now use robots to fulfill critical purposes in a plethora of ways.

While robotics applications vary greatly—giving directions, stocking shelves, welding metal in dangerous environments, and much more—today’s robots can generally be grouped into six categories.

**1.3.1.1 Autonomous Mobile Robots (AMRs)**

AMRs move throughout the world and make decisions in near real-time as they go. Technologies such as sensors and cameras help them ingest information about their surroundings. Onboard processing equipment helps them analyze it and make an informed decision—whether that’s moving to avoid an oncoming worker, picking precisely the right parcel, or selecting an appropriate surface to disinfect. They’re mobile solutions that require limited human input to do their job.

**1.3.1.2 Automated Guided Vehicles (AGV)**

While AMRs traverse environments freely, AGVs rely on tracks or predefined paths and often require operator oversight. These are commonly used to deliver materials and move items in controlled environments such as warehouses and factory floors.

8

**1.3.1.3 Articulated Robots**

Articulated robots (also known as robotic arms) are meant to emulate the functions of a human arm. Typically, these can feature anywhere from two to 10 rotary joints. Each additional joint or axis allows for a greater degree of motion—making this ideal for arc welding, material handling, machine tending, and packaging.

**1.3.1.4 Humanoid Robots**

While many mobile humanoid robots may technically fall under the domain of an AMR, the term is used to identify robots that perform human-centric functions and often take human-like forms. They use many of the same technology components as AMRs to sense, plan, and act as they carry out tasks such as providing directions or offering concierge services.

**1.3.1.5 Hybrids**

The various types of robots are often combined to create hybrid solutions that are capable of more complex tasks. For example, an AMR might be combined with a robotic arm to create a robot for handling packages inside of a warehouse. As more functionality is combined into single solutions, compute capabilities are also consolidated.

**1.3.1.6 Cobots**

Cobots are designed to function alongside or directly with humans. While most other types of robots perform their tasks independently, or in strictly isolated work areas, Cobots can share spaces with workers to help them accomplish more. They’re often used to eliminate manual, dangerous, or strenuous tasks from day-to-day workflows. In some cases, Cobots can operate by responding to and learning from human movements.

9

**1.4 EMBEDDED MACHINE LEARNING**

Machine learning (ML) enables electronic systems to learn autonomously from existing data and to use this acquired knowledge to independently make assessments, predictions, and decisions. These kinds of applications are highly compute-intensive, so they are traditionally executed on PCs and cloud servers. Thanks to new concepts and algorithms, as well as powerful dedicated processors, it is now possible to perform machine learning directly on devices used in the field (= embedded machine learning).

Embedded devices for machine learning applications can fulfill many tasks in the industry. One typical example: is sensor devices that detect acoustic or optical anomalies and discrepancies and, in this way, support quality assurance in production or system condition monitoring. In addition to cameras for monitoring visual parameters and microphones for recording soundwaves, these devices also use sensors for, instance, vibration, contact, voltage, current, speed, pressure, and temperature.

The Internet of Things (IoT) is the main reason for the rapidly increasing number of sensors used and with them the amount of data collected by embedded and IoT systems. Although the transmission technology, too, continues to evolve, and the new 5G standard will provide a powerful data network, complete transmission of sensor data to the cloud is not always practical or feasible.

To avoid these problems, it is often possible to process at least a portion of the sensor signals locally in the embedded device. For simple sensor data, this can be done with standard microcontrollers. Microcontrollers are well suited, for instance, for simple machine learning problems and applications with few channels and a low sampling rate, which don’t require frequent analyses.

10

For more complex analyses, e.g. for image sensors, specialized deep learning accelerators can also be used. For other applications, such as speech recognition, an accelerator IP core on an FPGA can be useful, and ASICs are a suitable option for high-volume applications.

**1.5 OBJECTIVE OF THE PROJECT**

Increasing health effects on the workers in the dump makes us initiate this project. As the first step in it, we have created an innovative method to segregate only plastics from the garbage. We have chosen the plastics first because they are the major pollutants in the dump area and also the ones that are essential to be separated for the recycling process. Artificial intelligence today is emerging as fast as it can in other countries. To increase the AI usage in our country we have to implement them in simple operations like this first, make them accessible to normal people. It will improve the growth of technology among the youths of our country. Robotics is also an everlasting emerging field in the world. The future of robots is in our hands, we have to develop them and implement them in small-scale operations to large-scale operations. As per the report given by the IFR (International Federation of Robotics), India has 4 robots per 10,000 employees. This is just a 39% growth rate of robots in India up to 2018.

So the main objective of our project is to give a suitable technique for the separation of plastics from the garbage. Next, to increase the usage of AI technology in our country by the normal people. Last but not least is to increase the robot density of our country. We hope that we will fulfill these objectives with our project as the first stepping stone.

11

**CHAPTER 2**

**LITERATURE REVIEW**

In the case of the Asia-Pacific (AP) region, the per capita plastic production and waste generation are low. This is because the population of this region is the highest in the world with China and India together contributing to more than one-third of the world’s population. It would be worth mentioning that in 2018 China alone contributed to about 23% of plastic production and 21% of waste generated in the world **(PlasticsEurope 2019; Ritchie and Roster 2019).**

Crazy as it may sound, there is no simple way to wish away this pervasive material. At least there is a common understanding of how plastic is polluting our water bodies, destroying wildlife, entering our food chain, and ultimately damaging our well-being (**[Jambeck et al. 2015](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B27-admsci-10-00023)**; **[Rochman et al. 2013](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B54-admsci-10-00023)**). Most countries around the world are committing resources to collect and recycle this material (**[Giacovelli 2018](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B20-admsci-10-00023)**; [**UNEP-AR 2018**](https://www.mdpi.com/2076-3387/10/2/23/htm#B67-admsci-10-00023); [**UNEP-SP 2018**](https://www.mdpi.com/2076-3387/10/2/23/htm#B68-admsci-10-00023)). But this is far away from the ideal scenario ([**Hook and Reed 2018**](https://www.mdpi.com/2076-3387/10/2/23/htm#B24-admsci-10-00023)).

Producing plastic products is carbon- and energy-intensive. These processes emit a huge amount of greenhouse gases, either directly or indirectly. If we take into account the complete supply chain of the source of various plastics and their respective disposal pathways, the overall carbon footprint increases tremendously. Regardless of post-disposal pathways—landfilling, incinerating, or recycling—we have to deal with resulting carbon emissions. In 2015, global carbon emission due to virgin (fossil fuel-based) plastic production was approximately 1.8 Gt.

12

To put this into perspective, this amount corresponds to roughly 3.8% of the overall global carbon emission in that year due to various human activities. If we take into account all the emissions coming from the post-disposal and recycling processes, the carbon footprint of plastics during their entire life cycle will be much higher ([**Chaffee and Yoros 2007**](https://www.mdpi.com/2076-3387/10/2/23/htm#B8-admsci-10-00023); **[Pilz et al. 2010](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B41-admsci-10-00023)**; [**Zheng and Suh 2019**](https://www.mdpi.com/2076-3387/10/2/23/htm#B73-admsci-10-00023)).

At the segregation stage, we employ a different types of sensors supported by AI. Multi-sensor data fusion has been very successful in training intelligent robots for various tasks ([**Luo et al. 1988**](https://www.mdpi.com/2076-3387/10/2/23/htm#B33-admsci-10-00023); **[Masoumi et al. 2012](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B35-admsci-10-00023)**; [**Mitchell 2007**](https://www.mdpi.com/2076-3387/10/2/23/htm#B37-admsci-10-00023); [**Scott 1995**](https://www.mdpi.com/2076-3387/10/2/23/htm#B61-admsci-10-00023)). Some researchers have tried combining multi-sensor data fusion with neural networks to identify and segregate plastic waste ([**Scott and Waterland 1995**](https://www.mdpi.com/2076-3387/10/2/23/htm#B62-admsci-10-00023)). The data about the shape, color, and texture are retrieved from high-definition optical sensors (cameras). Using this, for example, we can separate certain shaped or colored plastic bottles. We can also employ (near-infrared) laser diodes. The light absorption spectroscopy of plastics, particularly in the wavelength range from 300 to 3000 nanometre [nm] showed a new possibility of optical sensing of plastics. Near-infrared laser diodes are also increasingly used to study the resonant frequencies of different plastics. These laser sensors can differentiate six types of detectable plastics, namely, PET (polyethylene terephthalate), PE (polyethylene), PVC (polyvinyl chloride), and PP (polypropylene), PS (polystyrene), and ABS (acrylonitrile butadiene styrene).

Essentially, these sensors differentiate different grades of plastic based on the resonant frequency of each type of plastic. Those frequencies at which the absorption spectrum of the plastic has a peak, are called resonant frequencies. Different plastic molecules have different resonant frequencies. By scanning the plastic wastes using these laser detectors over a range of frequencies we can find their resonant frequency. Depending on their resonant frequency, we can identify the type of plastic.

13

Recently, researchers have tried using advanced terahertz technologies for plastic identification ([**Hailu and Saeedkia 2017**](https://www.mdpi.com/2076-3387/10/2/23/htm#B22-admsci-10-00023)). These technologies can further help in the accurate identification and segregation of plastic waste. For segregation based on color, manual separation is not a good option for industrial-scale operations. Taking into account both the speed and accuracy required for industrial-scale set-ups, these technologies can yield nearly 99% accuracy for color-based segregation and 95-98% accuracy for plastic type-based segregation ([**Inada et al. 2001**](https://www.mdpi.com/2076-3387/10/2/23/htm#B26-admsci-10-00023); [**Scott and Waterland 1995**](https://www.mdpi.com/2076-3387/10/2/23/htm#B62-admsci-10-00023); [**Zhu et al. 2019**](https://www.mdpi.com/2076-3387/10/2/23/htm#B74-admsci-10-00023)).

Blockchain technology serves as a trust-based platform between plastic waste segregation, recyclers, and recycled feedstock buyers (manufacturers). The blockchain network has distributed, but not copied, digital information ([**Adebiyi-Abiola et al. 2019**](https://www.mdpi.com/2076-3387/10/2/23/htm#B1-admsci-10-00023); [**Crosby et al. 2016**](https://www.mdpi.com/2076-3387/10/2/23/htm#B9-admsci-10-00023); [**Drescher 2017**](https://www.mdpi.com/2076-3387/10/2/23/htm#B13-admsci-10-00023); **[Kouhizadeh and Sarkis 2018](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B29-admsci-10-00023)**; **[Kouhizadeh et al. 2019](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B30-admsci-10-00023)**; [**Mansfield-Devine 2017**](https://www.mdpi.com/2076-3387/10/2/23/htm#B34-admsci-10-00023); [**Romano and Schmid 2017**](https://www.mdpi.com/2076-3387/10/2/23/htm#B55-admsci-10-00023); **[Sekhri 2018](https://www.mdpi.com/2076-3387/10/2/23/htm" \l "B63-admsci-10-00023)**; [**Wang and Qu 2019**](https://www.mdpi.com/2076-3387/10/2/23/htm#B69-admsci-10-00023)). This digital information is validated by various partners during each transaction. A transaction is an exchange of information between segregators, recyclers, and manufacturers. Information transacted contains data regarding supply, demand, specifications (quality), bidding- and offer price.

Finally, through various experiments, we have analyzed that by applying DL techniques through neural networks, the accuracy rate of hand gesture recognition has increased to a greater extent. The output data obtained using DL techniques can be fed to control a robotic arm which can be employed for various real-time applications like performing surgeries, driving a car from home, and doing any other tasks by being at one place. This results in less time consumption from this experimentation. **(D.J. Hemanth, V. Vieira Estrela)**

14

**2.1 EXISTING SYSTEM**

Segregation of waste and its management is a difficult task for a sustainable green environment. A case is assumed here in Chennai, a high-density population in Tamilnadu, India. In Chennai, the household wastes are collected in terms of tonnes per month. And the separation of this waste for recycling is a challenging task. This project focuses on the separation of wastes such as paper, plastic, and metals through the developed system with the capability of identifying materials using various sensors such as proximity, inductive and capacitive sensors. Integration of these sensors into PLC, Arduino can be used to operate a Robotic arm in a systematic procedure. These integrated mechanisms are also providing information through an IoT establishment with sensors. The system is enabled to read, collect and transmit a huge volume of data over the internet and also segregation through robots. The observed experimental results showed the system developed through this procedure is helpful to achieve about 85% accuracy for the categorization of selective different kinds of waste. The results of this project work can be used for routine segregation work and reduce human involvement during repetitive tasks. Data received through sensors have been helpful to operate the Robotic arm in time. Usage of this robotic arm has been presented in this project work.

**2.2 DISADVANTAGES OF THE EXISTING SYSTEM**

* There is a need for several types of sensors like proximity, inductive and capacitive sensors for identifying the materials which are plastic or not by sensing the object at a very close distance.
* It is programmed only based on the assembly language program which suits the Arduino. It doesn’t use any Artificial intelligence technology.
* The accuracy of the above system is below 90%.

15

**2.3 PROPOSED SYSTEM**

We have designed a Deep learning Neural Network model for the identification of various types of plastics in the garbage dump. We have created a total of 2 classes named Plastic and Non-Plastic. These two classes consist of all types of plastics and non-plastic materials. We have used about 2500 images for these two classes. Images like plastic water bottles, plastic plates, plastic pens, and plastic cups in the Plastic class. In Non-plastic class, it contains 5 sub-classes like Cardboard, Paper, Metal, Glass, and Others. These subclasses will contain images like cardboard boxes, different types of papers, metal tins, steel plates, glass cups, and some other images which are doesn’t belong to these subclasses like pencils, erasers, and circuit boards in the Others sub-class. Each image is in the dimension of 416 x 416.

Here we have used the YOLOV5 for training this model. It gives a good result of 95% of accuracy in identifying the plastics without the help of any sensors and with no human involvement. We have used a robotic arm that is controlled by the Arduino UNO R3 which fetches the result of the model through serial communication and a robotic arm to do the tasks according to the values it fetches.

**2.4 ADVANTAGES OF THE PROPOSED SYSTEM**

* No need for different types of sensors for detecting the objects.
* Can detect objects from a considerable long distance.
* Using the Deep Learning technology for detecting the objects.
* No manual calculations are required for detection.
* Gains accuracy of about 95%.

16

**CHAPTER 3**

**SYSTEM SPECIFICATIONS**

**3.1 HARDWARE**

* **Plywood**

The plywood is used for the construction of the robotic arm and also for handling the objects on it.

* **Electronics**
  + **Arduino UNO R3**

Arduino is an ATMega328 microcontroller board. It is a DIP (dual in-line package). In this Arduino, we have 20 input/output pins, out of which 6 can be used as PWM pins, i.e., which can also be used for analog inputs. It contains a crystal oscillator whose frequency can vary according to the manufacturer, but generally, it is a 16 MHz quartz oscillator. Supplies power by external using USB cable and **external 9v battery**.

* + **Servo motor**

This servo motor was used for greater precision of rotation of the object to an angle and distances.

* + - * Torque: 1.8kg-cm (4.8 V)
      * Speed: 0.10 s/60 degree
      * Weight: 9 gm
      * Gear type: Plastic
      * Rotational range: 180 degrees

**17**

* **Camera**

Here we have mentioned the specifications of the camera which we have used in our project.

* Model: ZEB-CRISP-PRO
* Frames per second: 24 fps
* Lens: 5P Lens
* Resolution: HD Quality (1280 x 720)
* **PC Recommended Specifications**
* RAM range: 8GB to 16GB
* SSD size: 256GB to 516GB
* HDD space: 1TB to 2TB
* Graphics Card: Anyone among GTX 1650, GTX 1660, GTX 1660 Ti, RTX 2060, GTX 1070.

**3.2 SOFTWARE**

**3.2.1 Google Colab**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. Colab is free of charge to use.

Here we have used this Google Colab for training our YOLOV5 model. It is the main source in our project because without its dedicated free GPUs we can’t train our model.

18

**3.2.2 Pycharm IDE**

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. PyCharm provides smart code completion, code inspections, on-the-fly error highlighting, and quick fixes, along with automated code refactorings and rich navigation capabilities. PyCharm’s smart code editor provides first-class support for Python, JavaScript, CoffeeScript, TypeScript, CSS, popular template languages, and more.

Refactor your code the intelligent way, with safe Rename and Delete, Extract Method, Introduce Variable, Inline Variable or Method, and other refactorings. Language and framework-specific refactorings help you perform project-wide changes. PyCharm’s huge collection of tools out of the box includes an integrated debugger and test runner; Python profiler; a built-in terminal; integration with major VCS and built-in database tools; remote development capabilities with remote interpreters; an integrated ssh terminal; and integration with Docker and Vagrant.

In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more. PyCharm integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy. We can use PyCharm on Windows, macOS, and Linux with a single license key. Enjoy a fine-tuned workspace with customizable color schemes and key bindings, with VIM emulation available.

19

**3.2.3 Python Packages**

* **Pytorch**

PyTorch is an optimized tensor library for deep learning using CPUs and GPUs. PyTorch has a rich set of packages that are used to perform deep learning concepts. These packages help us in optimization, conversion, loss calculation, etc.

* **Roboflow**

Roboflow makes managing, preprocessing, augmenting, and versioning datasets for computer vision seamless. It is the official Roboflow python package that interfaces with the Roboflow API. Key features of Roboflow: Import and Export image datasets into any supported formats.

* **Numpy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

* **OpenCV**

OpenCV has a function to read video, which is cv2. VideoCapture(). We can access our webcam using pass 0 in the function parameter. If you want to capture CCTV footage then we can pass the RTSP URL in the function parameter, which is useful for video analysis.

* **Serial**

This module encapsulates the access to the serial port. It provides backends for [Python](http://python.org/) running on Windows, OSX, Linux, BSD (possibly any POSIX compliant system), and IronPython. The module named “serial” automatically selects the appropriate backend.

20

* **Time**

The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.

* **Tensorboard**

TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embeddings to a lower-dimensional space, and much more.

* **Yaml**

YAML is a data serialization format designed for human readability and interaction with scripting languages. PyYAML is a YAML parser and emitter for Python.

* **cuDNN**

The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers.

* **Argparse**

The argparse module makes it easy to write user-friendly command-line interfaces. It parses the defined arguments from the sys. argv. The argparse module also automatically generates help and usage messages, and issues errors when users give the program invalid arguments.

21

**3.2.4 Arduino IDE**

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Port** menus. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for a UNO or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1** (for a Duemilanove or earlier USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably **COM1** or **COM2** (for a serial board) or **COM4**, **COM5**, **COM7**, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be **/dev/ttyACMx**, **/dev/ttyUSBx** or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **Sketch** menu. Current Arduino boards will reset automatically and begin the upload.

22

**CHAPTER 4**

**SYSTEM ANALYSIS**

**4.1 DATASET PREPARATION**

In the field of deep learning, the first and foremost step is dataset preparation and it is also the most hectic process. It is a long time-consuming process, where most people do many mistakes and become frustrated with their work. The quality of the dataset will decide the performance of the model. The higher the quality the higher will be the performance. To make the dataset quality there are many data augmentation techniques available. There is a step-by-step procedure for creating a clean and quality dataset. They are,

* Data Collection
* Data Cleaning
* Data Annotation and Labelling
* Data Splitting
* Pre-processing
* Data Augmentation

**4.1.1 Data Collection**

Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained.

23

Following are some of the problems that can arise in data collection:

* Inaccurate data. The collected data could be unrelated to the problem statement.
* Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.
* Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.
* Data bias. Depending on how the data, subjects, and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age, or region, for example. Data bias is difficult to detect and remove.

Several techniques can be applied to address those problems:

* Pre-cleaned, freely available datasets. If the problem statement (for example, image classification, object recognition) aligns with a clean, pre-existing, properly formulated dataset, then take advantage of existing, open-source expertise.
* Web crawling and scraping. Automated tools, bots, and headless browsers can crawl and scrape websites for data.
* Private data. ML engineers can create their data. This is helpful when the amount of data required to train the model is small and the problem statement is too specific to generalize over an open-source dataset.

24

****

Fig 4.1 Dataset Images

For this project, we have collected the datasets from Kaggle where we can get plenty of datasets and codes for our projects, and also take photos with a camera.

**4.1.2 Data Cleaning**

 Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. Data cleaning is a lot of muscle work. There’s a reason data cleaning is the most important step if you want to create a data culture, let alone make airtight predictions. It involves:

* Fixing spelling and syntax errors
* Standardizing data sets
* Correcting mistakes such as empty fields
* Identifying duplicate data points

25

**4.1.3 Data Annotation and Labelling**

**To summarize, data labelling and data annotation are all about labelling or tagging relevant information/metadata in a dataset to let machines understand what they are. The dataset could be in any form i.e., image, an audio file, video footage, or even text.** When we label elements in data, ML models accurately comprehend what they are going to process and keep that information to automatically process newer information that is built on existing knowledge to take timely decisions.

Data annotation is inevitable because AI and machine learning models need to be trained consistently to become more efficient and effective in delivering required outputs. In supervised learning, the process becomes all the more crucial because the more annotated data that is fed to the model, the sooner it trains itself to learn autonomously.

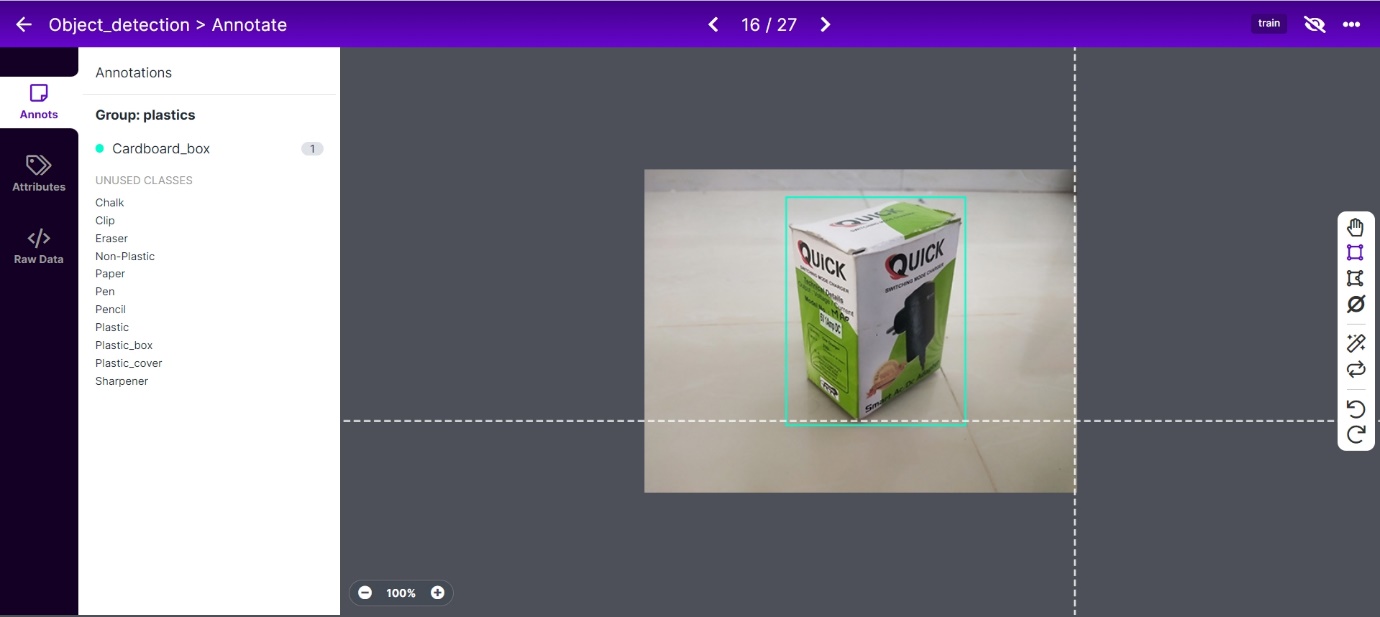


Fig 4.2 Annotating the images in Roboflow

A data labelling tool is an on-prem, or cloud-based solution that annotates high-quality training data for machine learning models.

26

Here we have used the roboflow website for annotating and labelling our dataset. It is a very useful tool for a data scientist which has many features for labelling the datasets. Here we have labelled our dataset into two classes,

* Plastic – 1000 images
* Non-Plastic – 1000 images

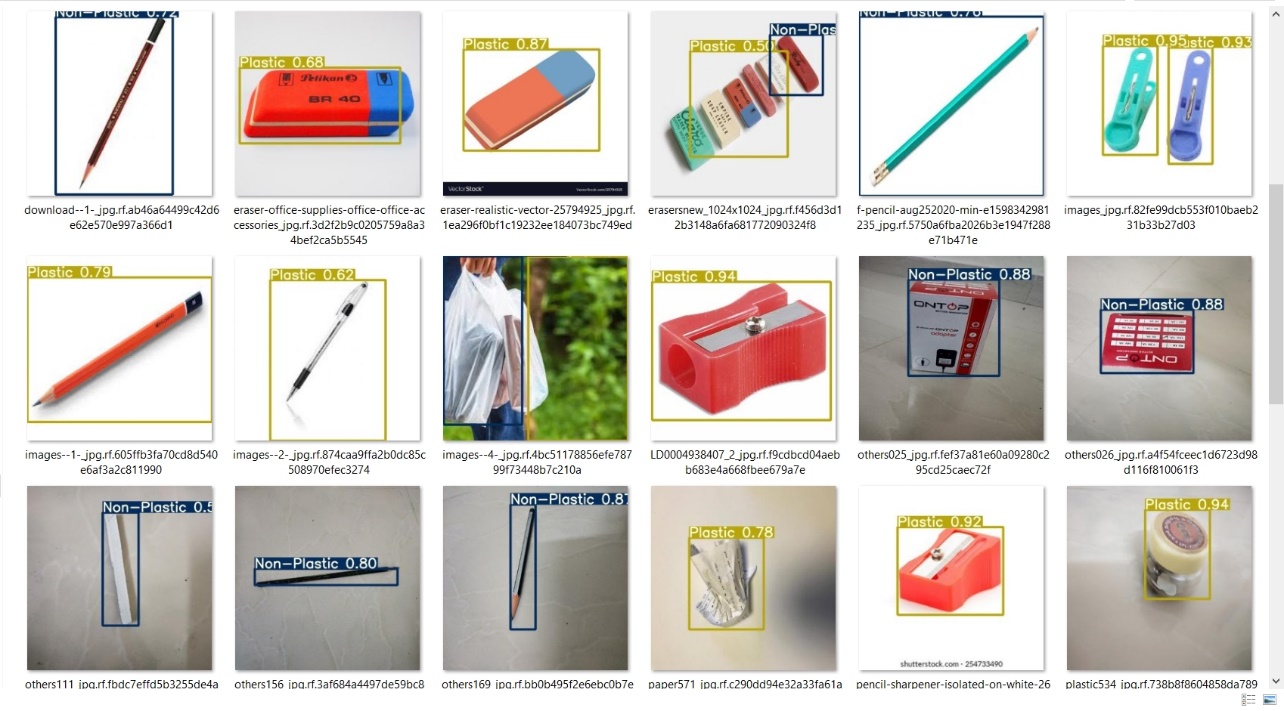


Fig 4.3 Annotated Dataset

**4.1.4 Data Splitting**

Data splitting is **when data is divided into two or more subsets**. Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model. Data splitting is an important aspect of data science, particularly for creating models based on data. Here, the Roboflow website will automatically divide our datasets into Train, Valid, and Test datasets by the ratio of 70:20:10.

27

|  |  |  |
| --- | --- | --- |
| Train | Valid | Test |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Fig 4.4 Splitted Dataset

28

**4.1.5 Pre-Processing**

Preprocessing data is a common first step in the deep learning workflow to prepare raw data in a format that the network can accept. For example, you can resize image input to match the size of an image input layer. You can also preprocess data to enhance desired features or reduce artifacts that can bias the network.

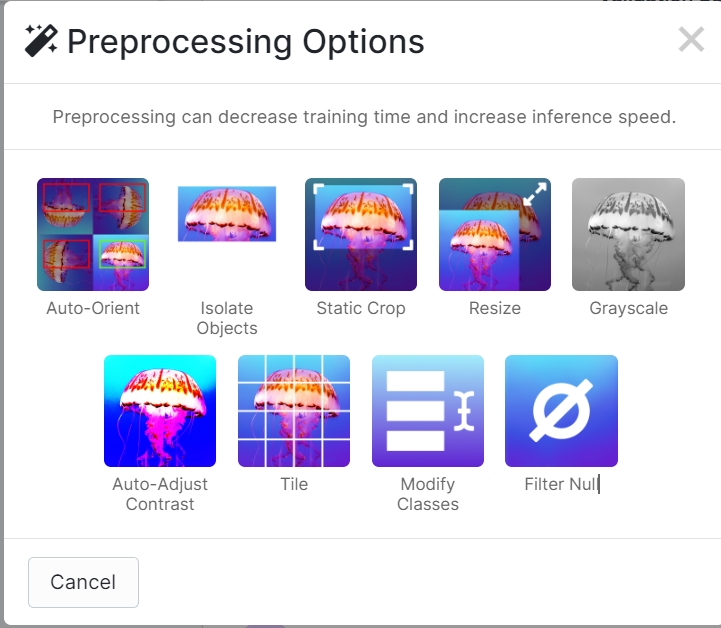


Fig 4.5 Pre-Processing Options in Roboflow

**4.1.6 Data Augmentation**

A [convolutional neural network](https://nanonets.com/blog/human-pose-estimation-2d-guide/) that can robustly classify objects even if its placed in different orientations is said to have the property called**invariance**. More specifically, a CNN can be invariant to **translation, viewpoint, size**, or **illumination**(Or a combination of the above). This essentially is the premise of **data augmentation**. In the real-world scenario, we may have a **dataset**of images taken under a **limited set of conditions**.

29

But, our **target application** may exist in a **variety of conditions**, such as different orientations, locations, scale, brightness, etc. We account for these situations by training our neural network with additional **synthetically modified data**.

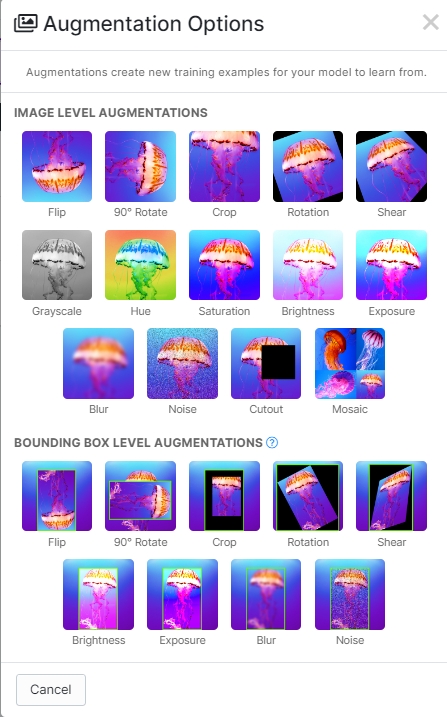
****

Fig 4.6 Data Augmentation Options in Roboflow

30

**4.2 TRAINING**

Here in our project, we’ve used our custom dataset to train it in the YOLOv5 model with the help of Roboflow.ai. They have given a step-by-step tutorial for training our custom dataset in a YOLOv5 model. There are several various other models are available in Roboflow such as,

* YOLOS
* Mask RCNN
* YOLOX
* Vision Transformer and many more…

[Roboflow](https://roboflow.com/) enables teams to deploy custom computer vision models quickly and accurately. Convert data to annotation format, assess dataset health, preprocess, augment, and more. It's free for your first 1000 source images. To train our custom dataset, we have to select a particular dataset type, here we have selected the YOLOv5 PyTorch type. After selecting the type, open the Roboflow model’s webpage and select the YOLOv5 model where it shows a Google Colab notebook. Click the notebook that will take you to the new page opening a google colab notebook called Roboflow-Train-YOLOv5. You have to save a copy of this notebook in your google drive so that if there is a problem you can access your dataset on your google drive. To save a copy go to File>save a copy in Drive. The code for training our dataset is given in the appendix.

**Install Dependencies**

We begin by cloning the YOLO v5 repository and setting up the dependencies required to run YOLO v5. You might need sudo rights to install some of the packages.

31

I recommend you create a new conda or a virtualenv environment to run your YOLO v5 experiments to not mess up the dependencies of any existing project. Once you have activated the new environment, install the dependencies using pip. Make sure that the pip you are using is that of the new environment. You can do so by typing in the terminal. With the dependencies installed, let us now import the required modules like a torch, and Image from IPython. The display.

**Download Correctly Formatted Custom Dataset**

We'll download our dataset from Roboflow. Use the "**YOLOv5 PyTorch**" export format. Note that the Ultralytics implementation calls for a YAML file defining where your training and test data is.

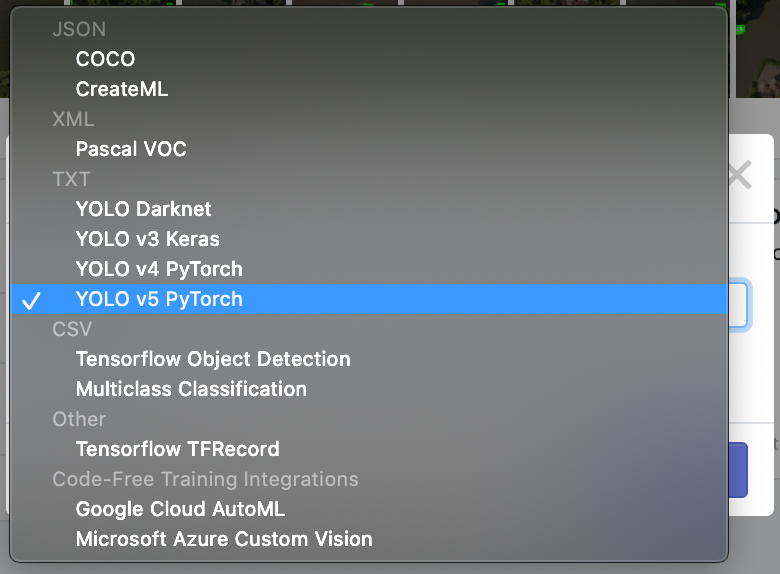


Fig 4.7 Selecting the dataset type

The dataset contains images of plastic and non-plastic materials as we discussed before. It contains almost 2000 images in total. After selecting the dataset format we have to import Roboflow and get a special link to our dataset. With the link, you get a python code which we have entered into the google colab to extract our dataset into google colab.

32

**Define Model Configuration and Architecture**

We will write a YAML script that defines the parameters for our models like the number of classes, anchors, and each layer.

**Train Custom YOLOv5 Detector**

### **Next, we'll fire off training!**

Here, we can pass several arguments:

* **IMGmg:** define input image size.
* **batch:** determine batch size
* **epochs:** define the number of training epochs. (Note: often, 3000+ are common here!)
* **data:** set the path to our YAML file
* **cfg:** specify our model configuration
* **weights:** specify a custom path to weights. (Note: you can download weights from the Ultralytics Google Drive [folder](https://drive.google.com/open?id=1Drs_Aiu7xx6S-ix95f9kNsA6ueKRpN2J))
* **name:** result names
* **nosave:** only save the final checkpoint
* **cache:** cache images for faster training

**Batches:** 16

**img:** 416

**epochs:** 200

Once the training is finished, it is saved as best. pt in the runs folder. We should download it and store it on our computer for deploying in our project.

33

**CHAPTER 5**

**ARDUINO SERIAL COMMUNICATION**

**5.1 WORKING**

After the training of the deep learning model, it is now time to do the Arduino serial communication programming. We can use the Arduino IDE for programming. To establish serial communication between the Arduino and the PC we should import serial in the detect.py file and enter the respected COM port of the Arduino with a baud rate and write if constructor which says that if the detect.py results a plastic write 0 to the Arduino else write 1 for non-plastic.

In the Arduino IDE, include the servo library for the servo motors and begin the serial communication by Serial.begin() and check whether the serial communication is established or not by Serial.available() and declare a variable for the Serial.read() and write a loop like if the variable reads a 0 then the robotic arm should pick the object and place it on the left side.

If it reads a 1 then the robotic arm should pick the object and place it on the right side. Then if no object is detected the robot should stop and become idle. The circuit diagram and the flowchart of the robotic arm are given below.

34

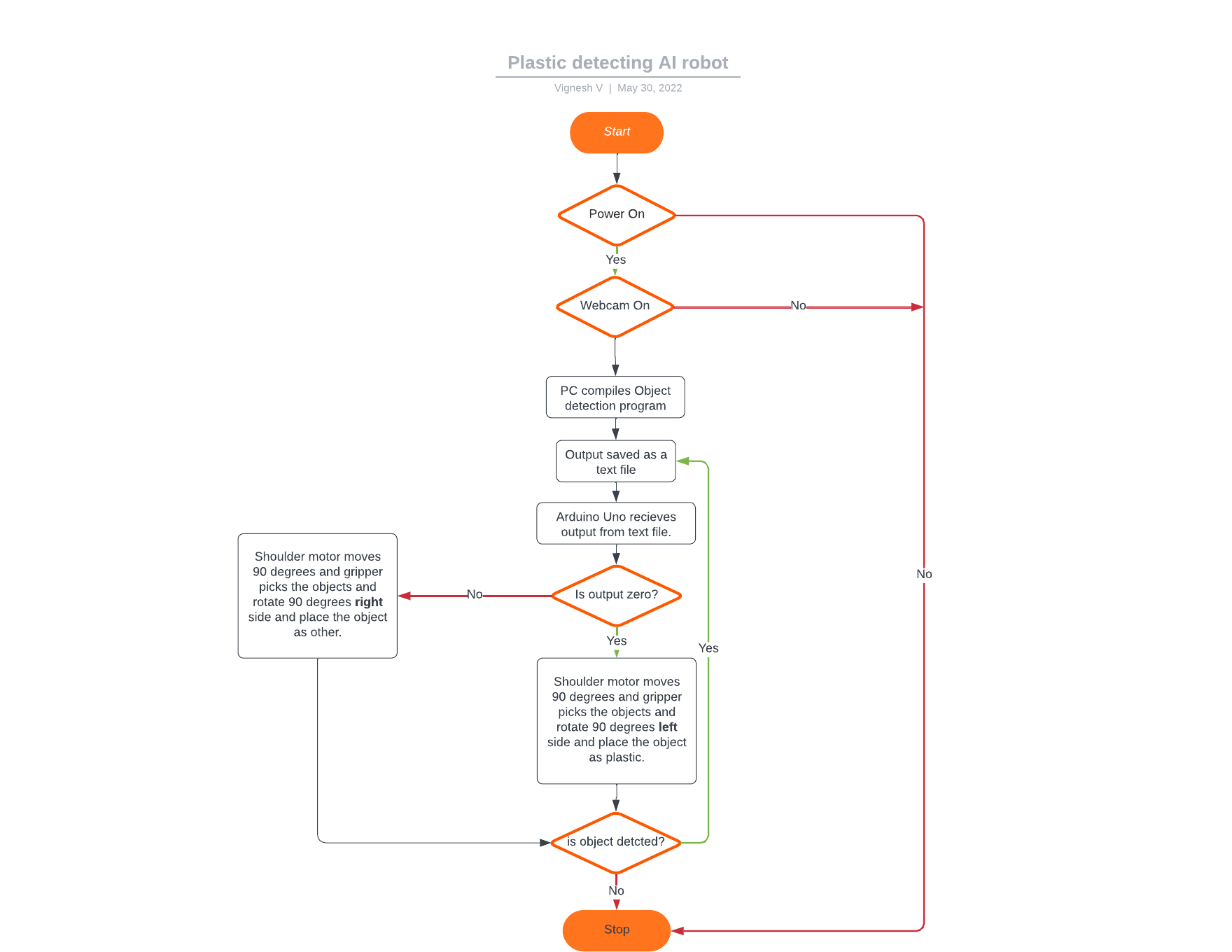


Fig 5.1 Flowchart

35

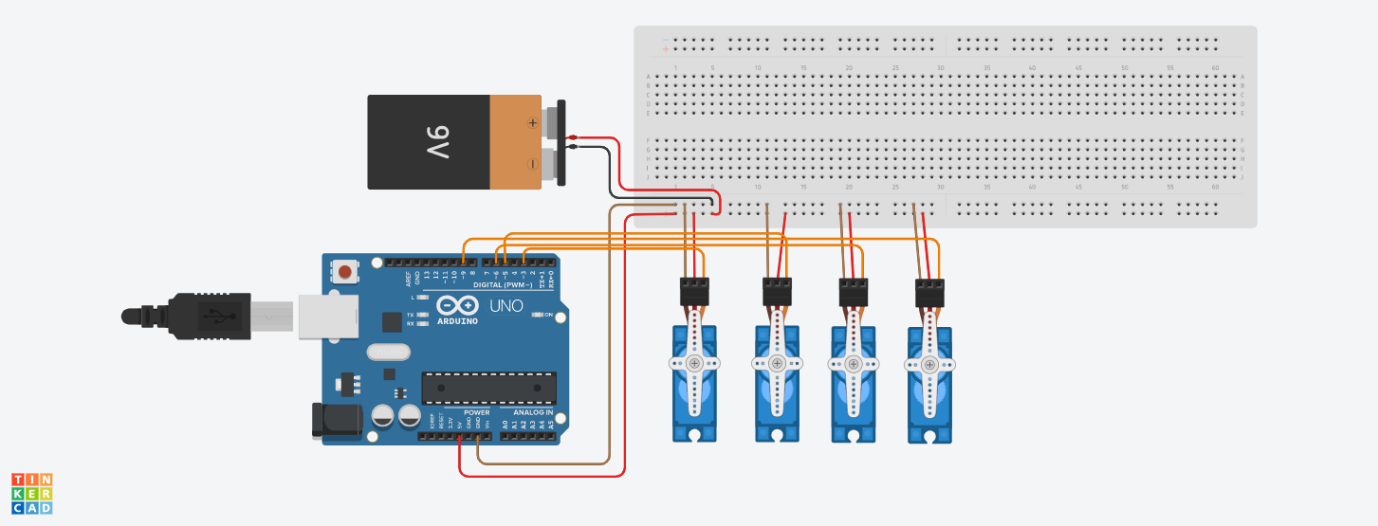


Fig 5.2 Circuit Diagram



Fig 5.3 Plastic Removing Robotic Arm

36

# CHAPTER 6

# RESULTS AND DISCUSSION

# 6.1 TRAINING RESULTS

# Training losses and performance metrics are saved to Tensorboard and also to a logfile defined above with the ****--name**** flag when we train. In our case, we named this yolov5s\_results. (If given no name, it defaults to results.txt.) The results file is plotted as a png after training completes. Start the tensorboard and launch after you have started training and the logs will be saved in the folder “runs”.

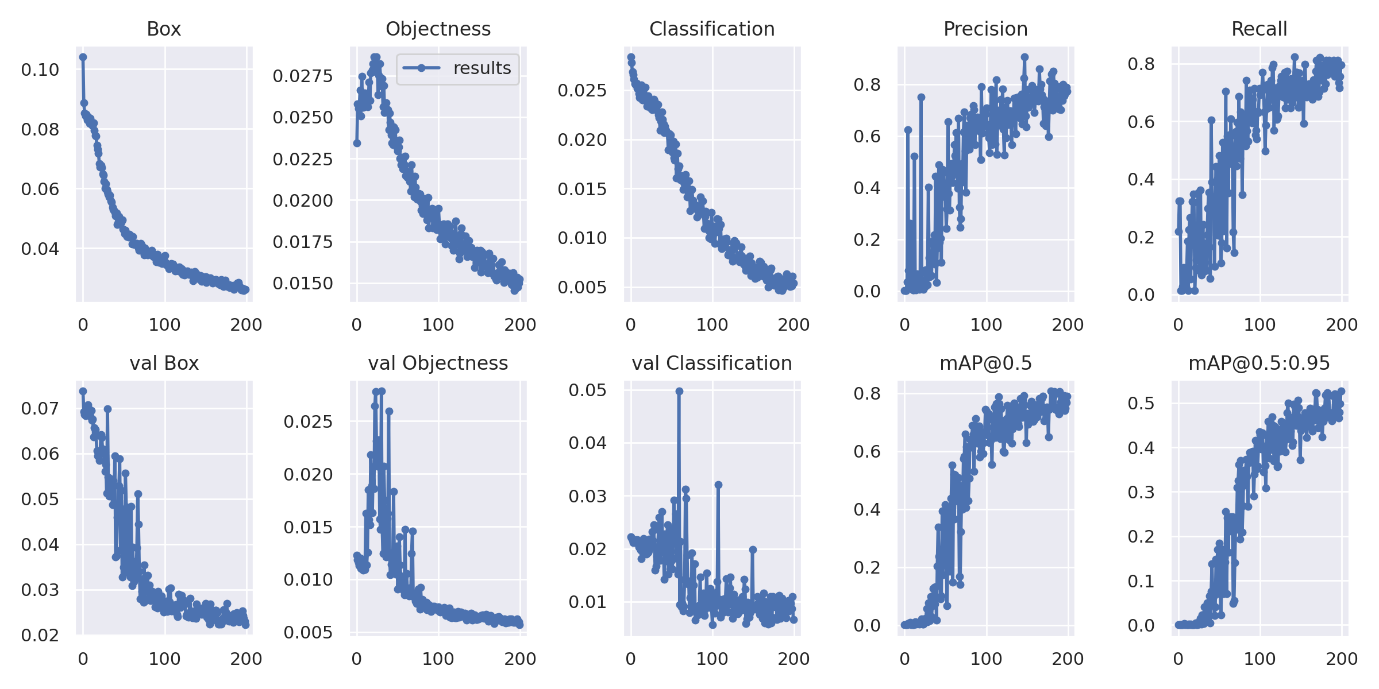
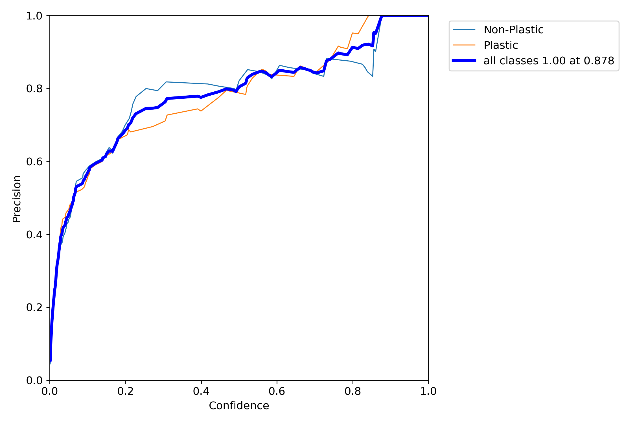
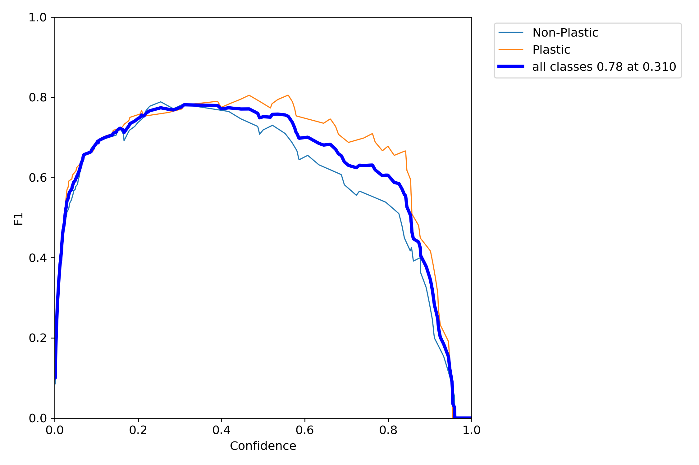


Fig 6.1 Tensorboard results

37



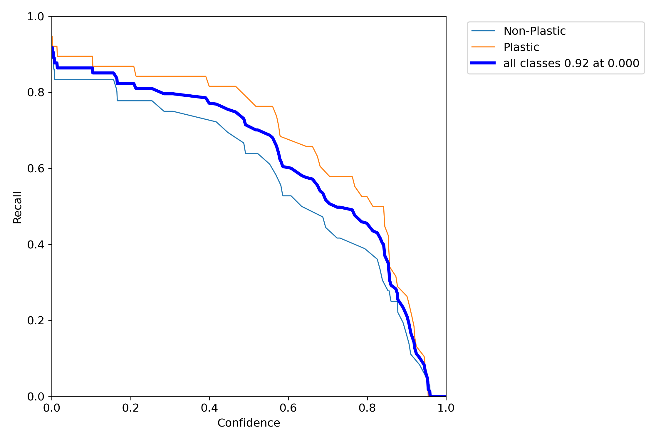
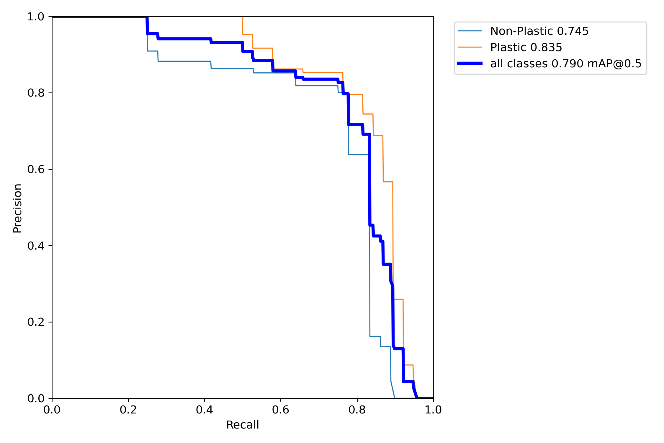


Fig 6.2 Performance Curves

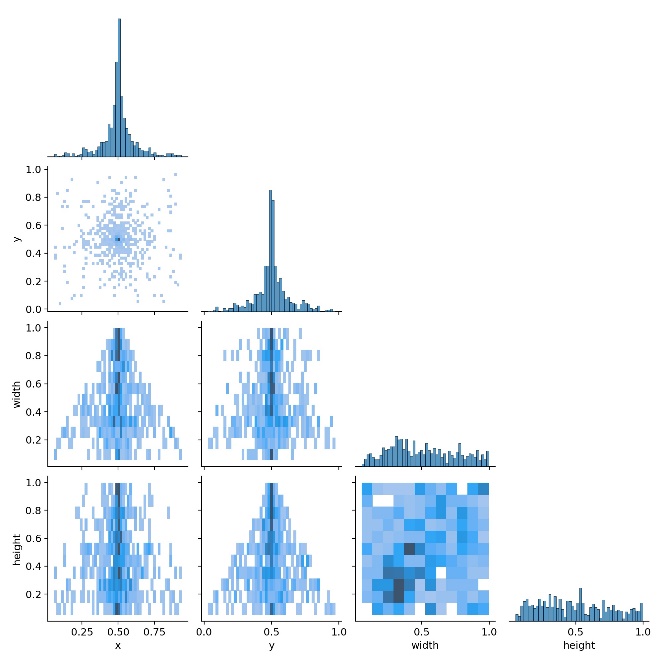
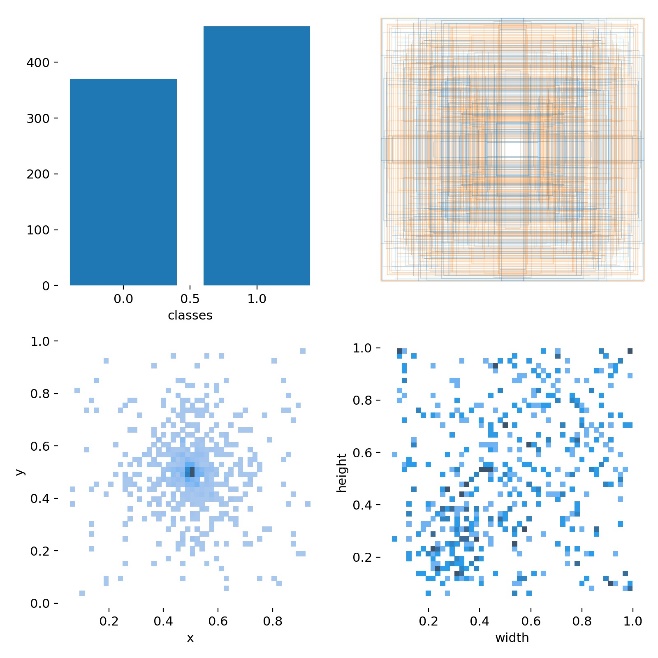


Fig 6.3 Labels Correlogram

38

After training the model, we have to download the model as a zip file and extract it to our disk. In the extracted file you can see a file named detect.py. You have to open any python IDE and open the detect.py file. Here we are using the anaconda terminal to run the detect, py file. In the anaconda, the navigator creates a new environment called yolov5 and activates the environment in the anaconda prompt. There you have to change the directory to your yolov5 folder and run the detect.py with the camera named 0.

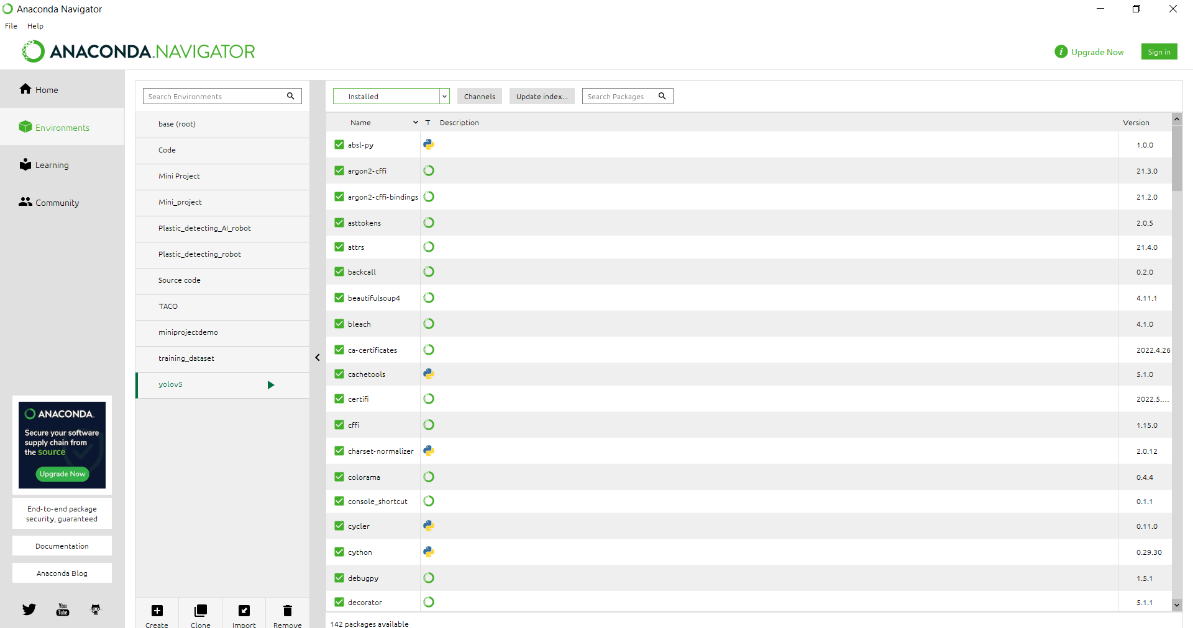


Fig 6.4 Anaconda Navigator

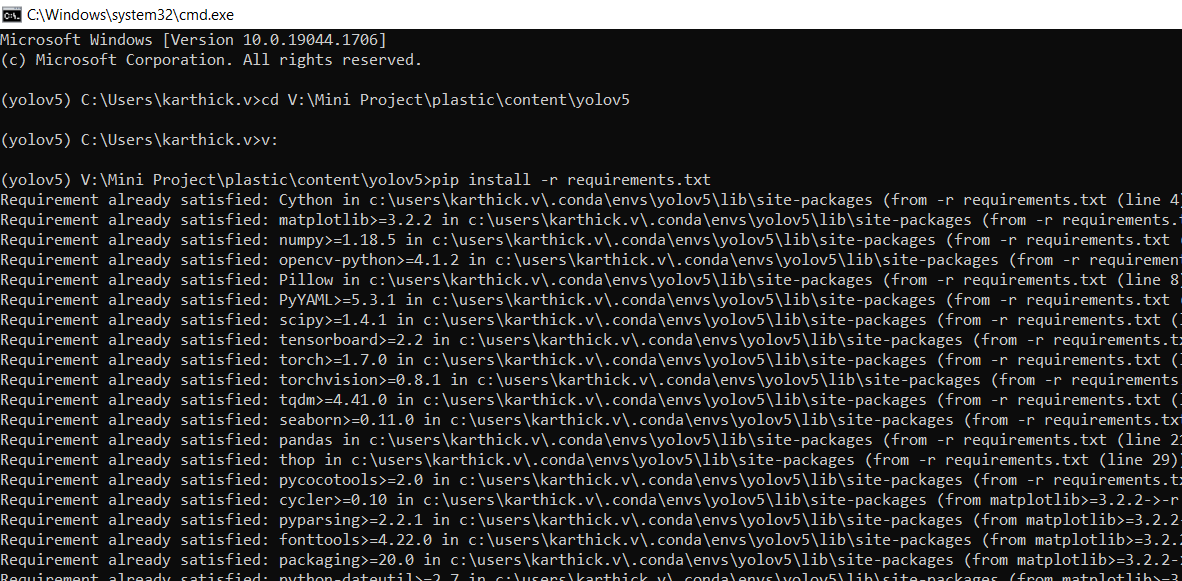


Fig 6.5 Anaconda\_Prompt

39

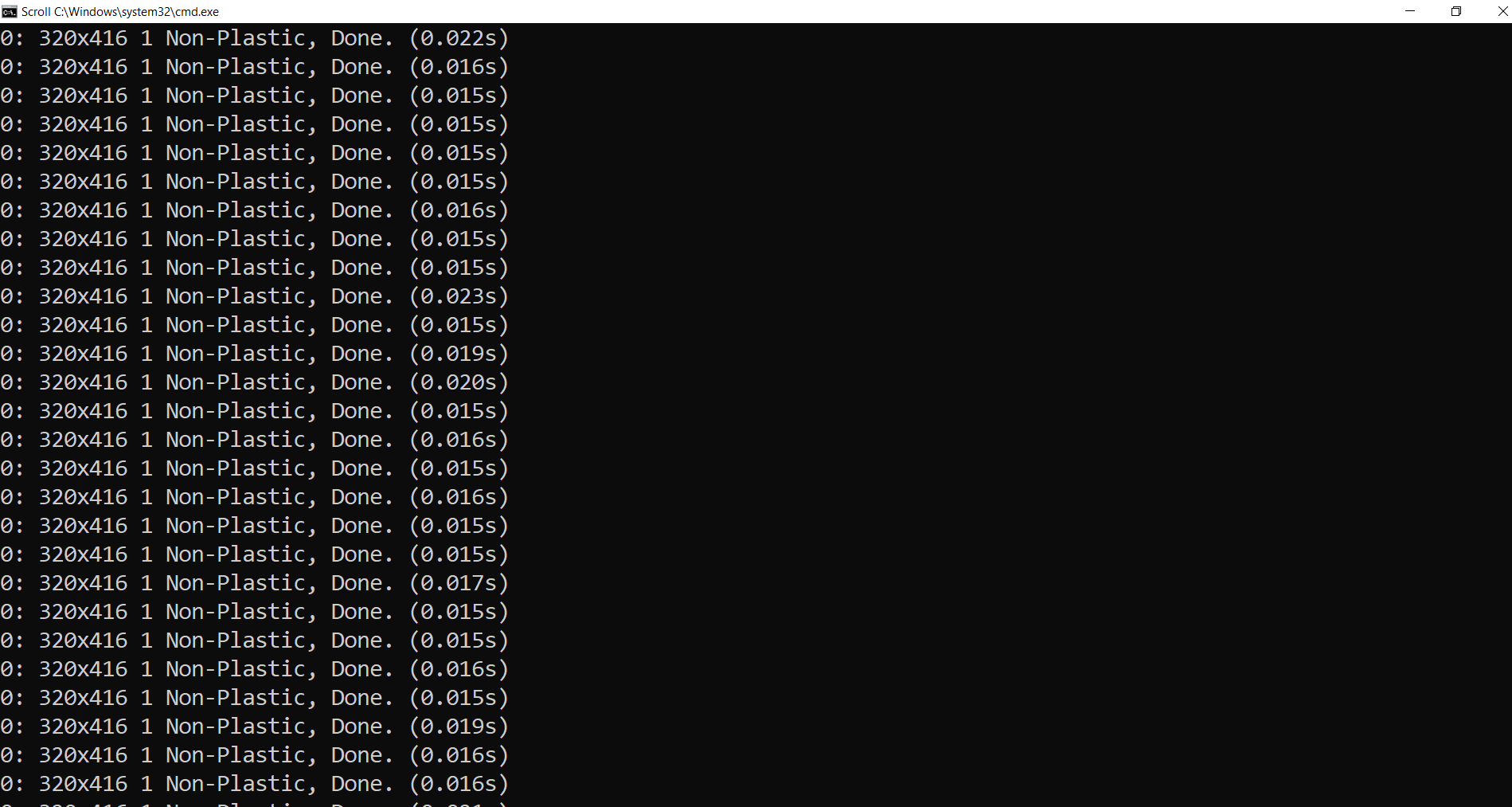


Fig 6.6 Output



Fig 6.7 Camera Output

40

**CHAPTER 7**

**CONCLUSION**

Various factors in the environment result in the transformation of model concepts and change in performance. The factors that drive the model should be understood to make certain decisions. Currently, the interpretability research on plastic waste identification models is a big gap in this field and must be explored. In the plastic type recognition field, deep learning models can be applied to identify plastic. However, these algorithms are still a ``black box'' to generate predictions based on input data.

There is no specific interpretation of what plastic waste detection features the deep learning model as a basis for judgment. The development of deep learning should be trustworthy and explainable. Making the algorithm public and transparent in decision-making gives users reliability and security. Interpretability research on plastic waste detection can resolve prejudices and auditing brought about by artificial intelligence Interpretability makes artificial intelligence open and transparent in legal, moral, and philosophical aspects.

41

**APPENDIX**

**YOLOv5 MODEL TRAINING**

# clone YOLOv5 repository

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

!git reset --hard 886f1c03d839575afecb059accf74296fad395b6

# install dependencies as necessary

!pip install -QR requirements.txt  # install dependencies (ignore errors)

import torch

from IPython.display import Image, clear\_output  # to display images

from utils.google\_utils import gdrive\_download  # to download models/datasets

# clear\_output()

print('Setup complete. Using torch %s %s' % (torch.\_\_version\_\_, torch.cuda.get\_device\_properties(0) if torch.cuda.is\_available() else 'CPU'))

#follow the link below to get your download code from from Roboflow

!pip install -q roboflow

from roboflow import Roboflow

rf = Roboflow(model\_format="yolov5", notebook="roboflow-yolov5")

%cd /content/yolov5

#after following the link above, recieve python code with these fields filled in

from roboflow import Roboflow

rf = Roboflow(api\_key="o11aEOAl80RX5BldhtRD")

project = rf.workspace().project("plastic\_detection-ksbf9")

dataset = project.version(1).download("yolov5")

# this is the YAML file Roboflow wrote and loading it in notebook

%cat {dataset.location}/data.yaml

42

# define number of classes based on YAML

import yaml

with open(dataset.location + "/data.yaml", 'r') as stream:

    num\_classes = str(yaml.safe\_load(stream)['nc'])

#this is the model configuration we will use for our tutorial

%cat /content/yolov5/models/yolov5s.yaml

#customize iPython writefile so we can write variables

from IPython.core.magic import register\_line\_cell\_magic

@register\_line\_cell\_magic

def writetemplate(line, cell):

    with open(line, 'w') as f:

        f.write(cell.format(\*\*globals()))

%%writetemplate /content/yolov5/models/custom\_yolov5s.yaml

# parameters

nc: {num\_classes}  # number of classes

depth\_multiple: 0.33  # model depth multiple

width\_multiple: 0.50  # layer channel multiple

# anchors

anchors:

  - [10,13, 16,30, 33,23]  # P3/8

  - [30,61, 62,45, 59,119]  # P4/16

  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone

backbone:

  # [from, number, module, args]

  [[-1, 1, Focus, [64, 3]],  # 0-P1/2

   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4

   [-1, 3, BottleneckCSP, [128]],

43

[-1, 1, Conv, [256, 3, 2]],  # 3-P3/8

   [-1, 9, BottleneckCSP, [256]],

   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16

   [-1, 9, BottleneckCSP, [512]],

   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32

   [-1, 1, SPP, [1024, [5, 9, 13]]],

   [-1, 3, BottleneckCSP, [1024, False]],  # 9

  ]

# YOLOv5 head

head:

  [[-1, 1, Conv, [512, 1, 1]],

   [-1, 1, nn.Upsample, [None, 2, 'nearest']],

   [[-1, 6], 1, Concat, [1]],  # cat backbone P4

   [-1, 3, BottleneckCSP, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],

   [-1, 1, nn.Upsample, [None, 2, 'nearest']],

   [[-1, 4], 1, Concat, [1]],  # cat backbone P3

   [-1, 3, BottleneckCSP, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],

   [[-1, 14], 1, Concat, [1]],  # cat head P4

   [-1, 3, BottleneckCSP, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],

   [[-1, 10], 1, Concat, [1]],  # cat head P5

   [-1, 3, BottleneckCSP, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)

  ]

# train yolov5s on custom data for 200 epochs

# time its performance

%%time

%cd /content/yolov5/

44

!python train.py --img 416 --batch 16 --epochs 200 --data {dataset.location}/data.yaml --cfg ./models/custom\_yolov5s.yaml --weights '' --name yolov5s\_results  --cache

**PERFORMANCE CURVES**

# Start tensorboard

# Launch after you have started training

# logs save in the folder "runs"

%load\_ext tensorboard

%tensorboard --logdir runs

# we can also output some older school graphs if the tensor board isn't working for whatever reason...

from utils.plots import plot\_results  # plot results.txt as results.png

Image(filename='/content/yolov5/runs/train/yolov5s\_results/results.png', width=1000)  # view results.png

# first, display our ground truth data

print("GROUND TRUTH TRAINING DATA:")

Image(filename='/content/yolov5/runs/train/yolov5s\_results/test\_batch0\_labels.jpg', width=900)

# print out an augmented training example

print("GROUND TRUTH AUGMENTED TRAINING DATA:")

Image(filename='/content/yolov5/runs/train/yolov5s\_results/train\_batch0.jpg', width=900)

# trained weights are saved by default in our weights folder

%ls runs/

%ls runs/train/yolov5s\_results/weights

# when we ran this, we saw .007 second inference time. That is 140 FPS on a TESLA P100!

# use the best weights!

%cd /content/yolov5/

45

!python detect.py --weights runs/train/yolov5s\_results/weights/best.pt --img 416 --conf 0.4 --source ../test/images

#display inference on ALL test images

#this looks much better with longer training above

import glob

from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/\*.jpg'): #assuming JPG

    display(Image(filename=imageName))

    print("\n")

**DETECT.py**

import serial  
import argparse  
import time  
from pathlib import Path  
  
import cv2  
import torch  
import torch.backends.cudnn as cudnn  
from numpy import random  
  
from models.experimental import attempt\_load  
from utils.datasets import LoadStreams, LoadImages  
from utils.general import check\_img\_size, check\_requirements, check\_imshow, non\_max\_suppression, apply\_classifier, \  
 scale\_coords, xyxy2xywh, strip\_optimizer, set\_logging, increment\_path  
from utils.plots import plot\_one\_box  
from utils.torch\_utils import select\_device, load\_classifier, time\_synchronized  
  
#ArduinoSerial = serial.Serial('com6', 9600)

46

def detect(save\_img=False):  
 source, weights, view\_img, save\_txt, imgsz = opt.source, opt.weights, opt.view\_img, opt.save\_txt, opt.img\_size

webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(  
 ('rtsp://', 'rtmp://', 'http://'))  
  
 # Directories  
 save\_dir = Path(increment\_path(Path(opt.project) / opt.name, exist\_ok=opt.exist\_ok)) # increment run  
 (save\_dir / 'labels' if save\_txt else save\_dir).mkdir(parents=True, exist\_ok=True) # make dir  
  
 # Initialize  
 set\_logging()  
 device = select\_device(opt.device)  
 half = device.type != 'cpu' # half precision only supported on CUDA  
  
 # Load model  
 model = attempt\_load(weights, map\_location=device) # load FP32 model  
 stride = int(model.stride.max()) # model stride  
 imgsz = check\_img\_size(imgsz, s=stride) # check img\_size  
 if half:  
 model.half() # to FP16  
  
 # Second-stage classifier  
 classify = False  
 if classify:  
 modelc = load\_classifier(name='resnet101', n=2) # initialize  
 modelc.load\_state\_dict(torch.load('weights/resnet101.pt', map\_location=device)['model']).to(device).eval()

47

# Set Dataloader  
 vid\_path, vid\_writer = None, None  
 if webcam:  
 view\_img = check\_imshow()  
 cudnn. benchmark = True # set True to speed up constant image size inference  
 dataset = LoadStreams(source, img\_size=imgsz, stride=stride)  
 else:  
 save\_img = True  
 dataset = LoadImages(source, img\_size=imgsz, stride=stride)  
  
 # Get names and colors  
 names = model.module.names if hasattr(model, 'module') else model.names  
 colors = [[random.randint(0, 255) for \_ in range(3)] for \_ in names]  
  
 # Run inference  
 if device.type != 'cpu':  
 model(torch.zeros(1, 3, imgsz, imgsz).to(device).type\_as(next(model.parameters()))) # run once  
 t0 = time.time()  
 for path, img, im0s, vid\_cap in dataset:  
 img = torch.from\_numpy(img).to(device)  
 img = img.half() if half else img.float() # uint8 to fp16/32  
 img /= 255.0 # 0 - 255 to 0.0 - 1.0  
 if img.ndimension() == 3:  
 img = img.unsqueeze(0)  
  
 # Inference  
 t1 = time\_synchronized()  
 pred = model(img, augment=opt.augment)[0]  
 # Apply NMS  
 pred = non\_max\_suppression(pred, opt.conf\_thres, opt.iou\_thres, classes=opt.classes, agnostic=opt.agnostic\_nms)  
 t2 = time\_synchronized()

48

# Apply Classifier  
 if classify:  
 pred = apply\_classifier(pred, modelc, img, im0s)  
  
 # Process detections  
 for i, det in enumerate(pred): # detections per image  
 if webcam: # batch\_size >= 1  
 p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(), dataset.count  
 else:  
 p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)  
  
 p = Path(p) # to Path  
 save\_path = str(save\_dir / p.name) # img.jpg  
 txt\_path = str(save\_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'\_{frame}') # img.txt  
 s += '%gx%g ' % img.shape[2:] # print string  
 gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh  
 if len(det):  
 # Rescale boxes from img\_size to im0 size  
 det[:, :4] = scale\_coords(img.shape[2:], det[:, :4], im0.shape).round()  
  
 # Print results  
 for c in det[:, -1].unique():  
 n = (det[:, -1] == c).sum() # detections per class  
 s += f"{n} {names[int(c)]}{'s' \* (n > 1)}, " # add to string  
  
 # Write results  
 for \*xyxy, conf, cls in reversed(det):  
 if save\_txt: # Write to file  
 xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() # normalized xywh  
 line = (cls, \*xywh, conf) if opt.save\_conf else (cls, \*xywh) # label format

49

with open(txt\_path + '.txt', 'a') as f:  
 f.write(('%g ' \* len(line)).rstrip() % line + '\n')  
  
 if save\_img or view\_img: # Add bbox to image  
 label = f'{names[int(cls)]} {conf:.2f}'  
 plot\_one\_box(xyxy, im0, label=label, color=colors[int(cls)], line\_thickness=3)  
  
 # Print time (inference + NMS)  
 print(f'{s}Done. ({t2 - t1:.3f}s)')  
  
 # Stream results  
 if view\_img:  
 cv2.imshow(str(p), im0)  
 cv2.waitKey(1) # 1 millisecond  
  
 # Save results (image with detections)  
 if save\_img:  
 if dataset.mode == 'image':  
 cv2.imwrite(save\_path, im0)  
 else: # 'video'  
 if vid\_path != save\_path: # new video  
 vid\_path = save\_path  
 if isinstance(vid\_writer, cv2.VideoWriter):  
 vid\_writer.release() # release previous video writer  
  
 fourcc = 'mp4v' # output video codec  
 fps = vid\_cap.get(cv2.CAP\_PROP\_FPS)  
 w = int(vid\_cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
 h = int(vid\_cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
 vid\_writer = cv2.VideoWriter(save\_path, cv2.VideoWriter\_fourcc(\*fourcc), fps, (w, h))  
 vid\_writer.write(im0)

50

if save\_txt or save\_img:  
 s = f"\n{len(list(save\_dir.glob('labels/\*.txt')))} labels saved to {save\_dir / 'labels'}" if save\_txt else ''  
 print(f"Results saved to {save\_dir}{s}")  
  
 print(f'Done. ({time.time() - t0:.3f}s)')  
if \_\_name\_\_ == '\_\_main\_\_':  
 parser = argparse.ArgumentParser()  
 parser.add\_argument('--weights', nargs='+', type=str, default='yolov5s.pt', help='model.pt path(s)')  
 parser.add\_argument('--source', type=str, default='data/images', help='source') # file/folder, 0 for webcam  
 parser.add\_argument('--img-size', type=int, default=640, help='inference size (pixels)')  
 parser.add\_argument('--conf-thres', type=float, default=0.25, help='object confidence threshold')  
 parser.add\_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')  
 parser.add\_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')  
 parser.add\_argument('--view-img', action='store\_true', help='display results')  
 parser.add\_argument('--save-txt', action='store\_true', help='save results to \*.txt')  
 parser.add\_argument('--save-conf', action='store\_true', help='save confidences in --save-txt labels')  
 parser.add\_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')  
 parser.add\_argument('--agnostic-nms', action='store\_true', help='class-agnostic NMS')  
 parser.add\_argument('--augment', action='store\_true', help='augmented inference')  
 parser.add\_argument('--update', action='store\_true', help='update all models')  
 parser.add\_argument('--project', default='runs/detect', help='save results to project/name')  
 parser.add\_argument('--name', default='exp', help='save results to

51

project/name')  
 parser.add\_argument('--exist-ok', action='store\_true', help='existing project/name ok, do not increment')  
 opt = parser.parse\_args()  
 print(opt)  
 check\_requirements()  
  
 with torch.no\_grad():  
 if opt.update: # update all models (to fix SourceChangeWarning)  
 for opt.weights in ['yolov5s.pt', 'yolov5m.pt', 'yolov5l.pt', 'yolov5x.pt']:  
 detect()  
 strip\_optimizer(opt.weights)  
 else:  
 detect()

**ARDUINO PROGRAM**

#include<Servo.h>

Servo bm;

Servo cm;

Servo rm;

Servo lm;

void setup() {

Serial.begin(9600);

bm.attach(9);

cm.attach(10);

rm.attach(5);

lm.attach(6);

bm.write(0);

delay(300);

rm.write(90);

delay(300);

lm.write(90);

delay(300);

52

cm.write(0);

delay(300);

}

void loop() {

while (Serial.available()== 0){

Data = Serial.read()

if (Data == 0){

for( int i=90; i<=150; i+=1){

rm.write(i);

delay(15);

}

for( int i=0; i<=150; i+=1){

cm.write(i);

delay(15);

}

for( int i=150; i>=90; i-=1){

rm.write(i);

delay(15);

}

for( int i=0; i<=90; i+=1){

bm.write(i);

delay(15);

}

for( int i=90; i<=150; i+=1){

rm.write(i);

delay(15);

}

53

for( int i=150; i>=0; i-=1){

cm.write(i);

delay(15);

}

}

else{

if (Data == 1){

for( int i=90; i<=150; i+=1){

rm.write(i);

delay(15);

}

for( int i=0; i<=150; i+=1){

cm.write(i);

delay(15);

}

for( int i=150; i>=90; i-=1){

rm.write(i);

delay(15);

}

for( int i=90; i<=150; i+=1){

bm.write(i);

delay(15);

}

for( int i=90; i<=150; i+=1){

rm.write(i);

delay(15);

}

for( int i=150; i>=0; i-=1){

cm.write(i);

delay(15);

}}}}}

54

**REFERENCES**

* Adebiyi-Abiola, Bilikiss, Solomon Assefa, Kareem Sheikh, and Jeannette M. García. 2019. Cleaning up plastic pollution in Africa. *Science* 365: 1249–51. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Cleaning+up+plastic+pollution+in+Africa&author=Adebiyi-Abiola,+Bilikiss&author=Solomon+Assefa&author=Kareem+Sheikh&author=and+Jeannette+M.+Garc%C3%ADa&publication_year=2019&journal=Science&volume=365&pages=1249%E2%80%9351&doi=10.1126/science.aax3539&pmid=31604228)] [**[CrossRef](https://doi.org/10.1126/science.aax3539" \t "_blank)**] [[**PubMed**](https://www.ncbi.nlm.nih.gov/pubmed/31604228)]
* Adrados, Aitziber, Isabel De Marco, Blanca Maria Caballero, A. López, M. Felisa Laresgoiti, and Amelia Torres. 2012. Pyrolysis of plastic packaging waste: A comparison of plastic residuals from material recovery facilities with simulated plastic waste. *Waste Management* 32: 826–32. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Pyrolysis+of+plastic+packaging+waste:+A+comparison+of+plastic+residuals+from+material+recovery+facilities+with+simulated+plastic+waste&author=Adrados,+Aitziber&author=Isabel+De+Marco&author=Blanca+Maria+Caballero&author=A.+L%C3%B3pez&author=M.+Felisa+Laresgoiti&author=and+Amelia+Torres&publication_year=2012&journal=Waste+Management&volume=32&pages=826%E2%80%9332&doi=10.1016/j.wasman.2011.06.016)] [**[CrossRef](https://doi.org/10.1016/j.wasman.2011.06.016" \t "_blank)**]
* Barnes, David K. A., Francois Galgani, Richard C. Thompson, and Morton Barlaz. 2009. Accumulation and fragmentation of plastic debris in global environments. *Philosophical Transactions of the Royal Society B: Biological Sciences* 364: 1985–98. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Accumulation+and+fragmentation+of+plastic+debris+in+global+environments&author=Barnes,+David+K.+A.&author=Francois+Galgani&author=Richard+C.+Thompson&author=and+Morton+Barlaz&publication_year=2009&journal=Philosophical+Transactions+of+the+Royal+Society+B:+Biological+Sciences&volume=364&pages=1985%E2%80%9398&doi=10.1098/rstb.2008.0205)] [**[CrossRef](https://doi.org/10.1098/rstb.2008.0205" \t "_blank)**]
* Bhatt, Aditi, Aakash, and Krishnaswamy Sankaran. 2018. Spaceborne Ocean Monitors: Radar Imaging of Illicit Oil-Spills. In *IEEE-INAE Workshop on Electromagnetics II*. Trivandrum: Indian National Academy of Engineering. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Spaceborne+Ocean+Monitors:+Radar+Imaging+of+Illicit+Oil-Spills&author=Bhatt,+Aditi&author=Aakash&author=and+Krishnaswamy+Sankaran&publication_year=2018)]
* Brooks, Amy L., Shunli Wang, and Jenna R. Jambeck. 2018. The Chinese import ban and its impact on the global plastic waste trade. *Science Advances* 4: 1–7. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=The+Chinese+import+ban+and+its+impact+on+global+plastic+waste+trade&author=Brooks,+Amy+L.&author=Shunli+Wang&author=and+Jenna+R.+Jambeck&publication_year=2018&journal=Science+Advances&volume=4&pages=1%E2%80%937&doi=10.1126/sciadv.aat0131)] [**[CrossRef](https://doi.org/10.1126/sciadv.aat0131" \t "_blank)**]
* Hailu, Daniel M., and Daryoosh Saeedkia. 2017. *Applications of Terahertz Technology for Plastic Industry*. Boca Raton: CRC Press, Chapter 5. pp. 107–17. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Applications+of+Terahertz+Technology+for+Plastic+Industry&author=Hailu,+Daniel+M.&author=and+Daryoosh+Saeedkia&publication_year=2017)]
* Inada, Koji, Rina Matsuda, Chizuru Fujiwara, Mitsutoshi Nomura, Tatsumi Tamon, Ikkan Nishihara, Takeshi Takao, and Toshihiro Fujita. 2001. Identification of plastics by infrared absorption using InGaAsP laser

55

* diode. *Resources, Conservation and Recycling* 33: 131–46. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Identification+of+plastics+by+infrared+absorption+using+InGaAsP+laser+diode&author=Inada,+Koji&author=Rina+Matsuda&author=Chizuru+Fujiwara&author=Mitsutoshi+Nomura&author=Tatsumi+Tamon&author=Ikkan+Nishihara&author=Takeshi+Takao&author=and+Toshihiro+Fujita&publication_year=2001&journal=Resources,+Conservation+and+Recycling&volume=33&pages=131%E2%80%9346&doi=10.1016/S0921-3449(01)00080-5)] [**[CrossRef](https://doi.org/10.1016/S0921-3449(01)00080-5" \t "_blank)**]
* Romano, Diego, and Giovanni Schmid. 2017. Beyond bitcoin: A critical look at Blockchain-based systems. *Cryptography* 1: 15. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Beyond+bitcoin:+A+critical+look+at+Blockchain-based+systems&author=Romano,+Diego&author=and+Giovanni+Schmid&publication_year=2017&journal=Cryptography&volume=1&pages=15&doi=10.3390/cryptography1020015)] [**[CrossRef](https://doi.org/10.3390/cryptography1020015" \t "_blank)**]
* Sankaran, Krishnaswamy. 2019a. Carbon Emission and Plastic Pollution: How Circular Economy, Blockchain, and Artificial Intelligence Support Energy Transition? *Journal of Innovation Management* 7. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Carbon+Emission+and+Plastic+Pollution:+How+Circular+Economy,+Blockchain,+and+Artificial+Intelligence+Support+Energy+Transition?&author=Sankaran,+Krishnaswamy&publication_year=2019a&journal=Journal+of+Innovation+Management&volume=7&doi=10.24840/2183-0606_007.004_0002)] [**[CrossRef](https://doi.org/10.24840/2183-0606_007.004_0002" \t "_blank)**]
* Sankaran, Krishnaswamy. 2019b. Protecting Oceans from Illicit Oil Spills: Environment Control and Remote Sensing using Spaceborne Imaging Radars. *Journal of Electromagnetic Waves and Applications* 33: 2373–403. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Protecting+Oceans+from+Illicit+Oil+Spills:+Environment+Control+and+Remote+Sensing+using+Spaceborne+Imaging+Radars&author=Sankaran,+Krishnaswamy&publication_year=2019b&journal=Journal+of+Electromagnetic+Waves+and+Applications&volume=33&pages=2373%E2%80%93403&doi=10.1080/09205071.2019.1685409)] [**[CrossRef](https://doi.org/10.1080/09205071.2019.1685409" \t "_blank)**]
* Sankaran, Krishnaswamy. 2020. Spaceborne Radar Remote Sensing of Ocean Surfaces: Electromagnetic Modelling and Applications. *Journal of Electromagnetic Waves and Applications* 34: 1–34. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Spaceborne+Radar+Remote+Sensing+of+Ocean+Surfaces:+Electromagnetic+Modelling+and+Applications&author=Sankaran,+Krishnaswamy&publication_year=2020&journal=Journal+of+Electromagnetic+Waves+and+Applications&volume=34&pages=1%E2%80%9334&doi=10.1080/09205071.2019.1696237)] [**[CrossRef](https://doi.org/10.1080/09205071.2019.1696237" \t "_blank)**]
* Sankaran, Krishnaswamy, and Joaquim Fortuny-Guasch. 2004. Radar remote sensing for oil spill classification (optimization for enhanced classification). Paper presented at the 12th IEEE Mediterranean Electrotechnical Conference, Dubrovnik, Croatia, May 12–15; pp. 511–514. [[**Google Scholar**](https://scholar.google.com/scholar_lookup?title=Radar+remote+sensing+for+oil+spill+classification+(optimization+for+enhanced+classification)&conference=Paper+presented+at+the+12th+IEEE+Mediterranean+Electrotechnical+Conference&author=Sankaran,+Krishnaswamy&author=and+Joaquim+Fortuny-Guasch&publication_year=2004&pages=511%E2%80%93514&doi=10.1109/MELCON.2004.1346979)] [**[CrossRef](https://doi.org/10.1109/MELCON.2004.1346979" \t "_blank)**]
* Utilization of hyperspectral imaging for impurities detection in secondary plastics,” 2012 S. Serranti, A. Gargiulo, G. Bonifazi, A. Toldy, S. Patachia, and R. Buican.

56

* Classification of polyolefins from building and construction waste using NIR hyperspectral imaging system,” 2012 S. Serranti, A. Gargiulo, and G. Bonifazi. 29
* Rapid discrimination of plastic packaging materials using MIR spectroscopy coupled with independent components analysis (ICA),” 2014 A. Kassouf, J. Maalouly, D. N. Rutledge, H. Chebib, and V. Ducruet.
* Deep: convolutional neural network applies to face recognition in small and medium databases,” 2018 M. Wang, Z. Wang, and J. Li.
* PET waste classification method and plastic waste database - WaDaBa,” 2018 J. Bobulski and J. Piatkowski.
* Waste classification system using image processing and convolutional neural networks,” 2019 J. Bobulski and M. Kubanek.
* CNN use for plastic garbage classification method,” 2019 J. Bobulski and M. Kubanek.
* T. P. Cabre, M. T. Cairol, D. F. Calafell, M. T. Ribes, and J. P. Roca, "Project-Based Learning Example: Controlling an Educational Robotic Arm With Computer Vision," Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de, vol. 8, pp. 135-142, 2013.
* P. J. Sanz, R. Marin, and J. S. Sánchez, "Including efficient object recognition capabilities in online robots: from a statistical to a Neural network classifier," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 35, pp. 87-96, 2005.
* V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani, "Visual grasp planning for unknown objects using a multi-fingered robotic hand," Mechatronics, IEEE/ASME Transactions on, vol. 18, pp. 1050-1059, 2013.

57