Visvesvaraya Technological University
Belagavi, Karnataka-590 018.

A Project Report on

"Garbage Segregation System using Image Processing"

Thesis submitted in partial fulfilment for the award of degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by,

Jyothsna P	1JS16CS045
Cauvery A	1JS16CS031
Bhargav Hegde	1JS16CS027
Akshitha Y V	1JS16CS010

Under the guidance of
Dr. Naveen N C
Professor and Head, Dept. of CSE,
JSSATE, Bengaluru.

JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU
Department of Computer Science and Engineering

2019 – 2020

Visvesvaraya Technological University
Belagavi, Karnataka-590 018.

A Project Report on

"Garbage Segregation System using Image Processing"

Thesis submitted in partial fulfilment for the award of degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by,

Jyothsna P	1JS16CS045
Cauvery A	1JS16CS031
Bhargav Hegde	1JS16CS027
Akshitha Y V	1JS16CS010

Under the guidance of
Dr. Naveen N C
Professor and Head, Dept. of CSE,
JSSATE, Bengaluru.

JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU
Department of Computer Science and Engineering

2019 – 2020

JSS MAHAVIDYAPEETHA, MYSURU

JSS Academy of Technical Education

JSS Campus, Uttarahalli Kengeri Main Road, Bengaluru - 560060

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the thesis work entitled **"Garbage Segregation System using Image Processing"** carried out by **JYOTHSNA P (1JS16CS045)**, **CAUVERY A (1JS16CS031)**, **BHARGAV HEGDE (1JS16CS027)**, **and AKSHITHA Y V (1JS16CS010)** in partial fulfilment for **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the academic year 2019 - 2020. It is certified that all corrections and suggestions indicated during internal assessment have been incorporated in the report. The seminar report has been approved as it satisfies the academic requirements in respect of project work prescribed for the award of degree of Bachelor of Engineering.

Dr. Naveen N C	**Dr. Mrityunjaya V Latte**
Professor and Head, Dept. of CSE,	Principal,
JSSATE, Bengaluru.	JSSATE, Bengaluru.

Name of the examiners	**Signature with Date**
1. ……	**……**
2. ……	**……**

JSS MAHAVIDYAPEETHA, MYSURU

JSS Academy of Technical Education

JSS Campus, Uttarahalli, Kengeri Main Road, Bengaluru - 560060

Department of Computer Science and Engineering

DECLARATION

We hereby declare that the entire work embodied in this project report titled **"Garbage Segregation System using Image Processing"** submitted to **Visvesvaraya Technological University, Belagavi** is carried out at the department of **Computer Science and Engineering, JSS Academy of Technical Education, Bengaluru** under the guidance of **Dr. Naveen N C, Professor & Head**. This report has not been submitted for the award of any Diploma or Degree of this or any other University.

Name	USN	Signature
1. Jyothsna P	1JS16CS045	…….
2. Cauvery A	1JS16CS031	…….
3. Bhargav Hegde	1JS16CS027	…….
4. Akshitha Y V	1JS16CS010	…….

ABSTRACT

It is known that the garbage problem in Bengaluru and throughout India has only been growing over the past few years. The landfills continue to be overfilled and there still is an enormous pile of unfiltered garbage at the end of every street simply put into heaps and thrown out without any proper care. The workers tasked with cleaning this up have to go through a process of segregating this discarded waste before they can do anything with it. A lot of their work and a lot of the space and time taken up by simply segregating the waste can be saved if the waste is separated beforehand. This is now a state law. Residents must separate the wet waste from the dry waste before passing it on to the government workers to dispose of them appropriately. But this is not fully followed by a majority of the citizens. It takes a conscious effort to follow this rule, which seems to be a drawback for a lot of people.[1]

To solve this problem, this project aims to automate the task of segregation of the waste. With this project, the attempt is to segregate waste with the use of machine learning algorithms and image processing technology to identify the different types of waste automatically, and send the waste to the appropriate bin with the help of an IoT based robotic model.[2]

ACKNOWLEDGEMENT

We express our humble gratitude to Holiness Jagadguru Sri Sri Sri Shivarathri Deshikendra Mahaswamiji who has showered their blessings on us for framing our career successfully.

The completion of any project involves the efforts of many people. We have been lucky enough to have received a lot of help and support from all quarters during the making of this report, so with gratitude, we take this opportunity to acknowledge all those whose guidance and encouragement helped us emerge successful.

We are thankful for Karnataka State Council for Science and Technology (KSCST) for allowing us the opportunity to showcase the project and funding of the robotic arm. We are grateful for JSS Management for their support in financing this endeavor and help in making a distinct impression for the Department of Computer Science and Engineering, JSS Academy of Technical Education, Bangalore.

We are thankful to the resourceful guidance, timely assistance and graceful gesture of our guide Dr. Naveen N C, Head of Department of Computer Science and Engineering, who has helped us in every aspect of our project and for the facilities and support extended towards us. We express our sincere thanks to our beloved principal, Dr. Mrityunjaya V Latte for having supported us in our academic endeavors.

And last but not the least; we would be very pleased to express our heart full thanks to all the teaching and non-teaching staff of CSE department and our friends who have rendered their help, motivation and support.

JYOTHSNA P
CAUVERY A
BHARGAV HEGDE
AKSHITHA Y V

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CNN	Convolutional Neural Network
COCO	Common Objects in Context
CUDA	Compute Unified Device Architecture
FPS	Frames Per Second
GPU	Graphical Processing Unit
IoT	Internet of Things
mAP	mean Average Precision
OpenCV	Open source Computer Vision
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machines
GPIO	General Purpose Input Output

Chapter 1

INTRODUCTION

1.1 Overview

Garbage is waste material that is discarded by humans, usually due to a perceived lack of utility. It doesn't include bodily waste products, purely liquid or gaseous wastes, or toxic waste products. They are commonly sorted and classified for specific kinds of disposal. Garbage is actually the technical term for putrescent organic matter. Garbage is discarded in ways that causes it to end up in the environment in an eco-friendly way as possible. Garbage segregation means dividing waste into wet and dry. Garbage collected in each area is to be separated before treating or reusing them for any other purposes.

Garbage is now being dumped into landfills without separation, or separated improperly. This decreases efficiency in the process and also contributes more to pollution. The proper disposal of waste largely impacts the environment and public health. This helps reduce hazard emissions, thereby mitigating climate change.

Image processing is a method to perform operations on an image, in order to extract some useful information from it. Processing of images is faster and more cost-effective than using sensors. It is more ecological to process images.

1.2 Motivation

Bengaluru has a major garbage problem and the municipal corporation mandates that the households must segregate the waste into wet and dry waste. Unfortunately, the basic waste segregation is not being followed. There is a requirement of an effective system that can ensure perfect segregation of wastes. The system must be able to perform the task of separating garbage appropriately in an automated fashion.

Classifying waste remains a challenge because of many factors, such as the complexity of the landscape in a study area and image-processing may affect the success of a classification. Segregated waste is often cheaper to dispose. One such system could save a lot of money when done right.

1.3 Objectives

The project aims to develop a technology for garbage segregation using image processing for the following things:

- To identify and categorize different types of waste simultaneously
- To highlight the categorized waste in a captured image
- To ensure that the accuracy is high enough for it to be practical

1.4 Organization

The report consists of 8 chapters. Chapter 2 consists of Literature Survey which includes the abstract of papers studied before starting the project. Chapter 3 consists of Comparative Research for the selection of system requirements. Chapter 4 contains System Requirements which consists of both software and hardware components used. Chapter 5 explains the System Architecture along with the block diagrams. Chapter 6 gives the detailed explanation of the implementation of the project. Chapter 7 contains the results of the project and Chapter 8 gives the conclusion and future enhancements on the project.

Chapter 2

LITERATURE SURVEY

2.1 Automated Waste Segregator [3]

This paper was presented by Amrutha Chandramohan, Joyal Mendonca, Nikhil Ravi Shankar, Nikhil U Baheti, Nitin Kumar Krishnan and Suma M S from RVCE, Bangalore in 2014 Texas Instruments India Educators' Conference.

During 2014, the common method of waste disposal was by unplanned and uncontrolled open dumping at the landfill sites. This method is injurious to human health, plant and animal life. There is a dependency on rag pickers for collecting the recyclable wastes from the dump. This is not the efficient way of segregating. When the waste is segregated into basic streams such as wet, dry and metallic, there was higher potential of recovery, recyclability and reusability. There were large scale industrial waste segregators present. In this, larger items were removed by manual sorting, then they were passed through rotating drums with perforated holes of certain size.

Proposed system, Automated Waste Segregator (AWS), proximity sensors were used to identify the incoming of waste and this starts the entire system. The waste falls on metal detection system which detects the metallic waste. After this, the object falls into capacitive sensing module, which distinguishes between wet and dry waste. After the identification the circular base which holds the containers is rotated to collect corresponding waste.

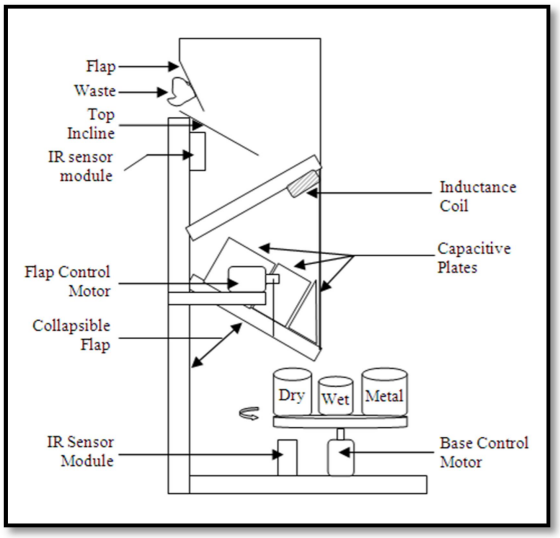

Figure 2.1: Automated Waste Segregator

The differentiation of wet and dry waste is done using dielectric constant. When a dielectric is introduced between the plates of capacitor, the capacitance increases. Wet waste has a higher relative dielectric constant than dry waste coz of the moisture, oil and fat content present in organic waste. A threshold is set, if the value is greater than the threshold then it is inferred as wet waste or else it is dry waste.

2.2 Classification of Trash for Recyclability status [4]

This paper was presented by Mindy Yang, Gary Thung from Stanford University. This paper focuses on a computer vision approach to classify the garbage into recycling categories. The objective of this project was to take images of a single piece of recycling or garbage and classify it to one of the classes. SVM with SIFT features and a CNN were the models used for this project.

Data set was created by hand by the presented students. It contains images of recycled objects like paper, glass, plastic, metal, cardboard, etc. Data augmentation techniques were performed due to the small size of each class. SVM was used for the classification of trash into categories. Features used for SVM were SIFT features. Torch7 Framework was used to construct CNN. The CNN was eleven layered network which is very similar to AlexNet. CNN was trained with train-val-test split of 70-13-17 with 60 epochs and a batch size of 32. Only single objects were supposed to be given as input for the classification.

Test accuracy with the SVM was 63% and that of CNN was only 22%. To improve the CNN results, more data should have been collected and used. The system with 22% accuracy wasn't practical. Identification of multiple objects from a single image or video is required for practical implementation.

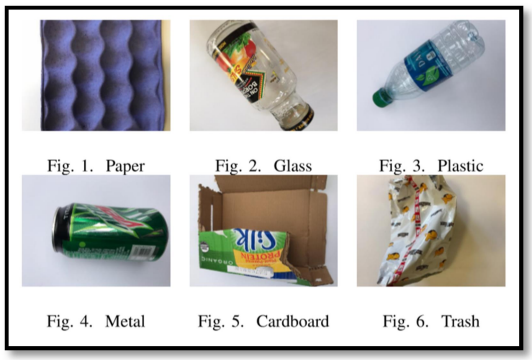

Figure 2.2: Example images from the dataset

2.3 Automatic waste segregator [5]

This paper was presented by Sharanya A, U Harika, N Sriya, Sreeja Kochvila from Amrita University, India. This paper aims to sort the waste to 3 major categories, namely metallic, wet, dry. Dry waste is further separated into paper and plastic. Arduino UNO is used with the sensors are used for the smooth running of the system.

Proposed system consists of 2 discs, rotating and stationary disc. Bins are placed between the discs. All the sensors are placed on the rotating disc at specified spots. IR proximity sensor detects the waste when kept and starts the system. Moisture sensor detects the wet waste. If not wet waste, it is passed to next sensor i.e., Inductive Proximity sensor for metal detection. The last module has a Laser LDR circuit that detects plastic and paper. If the LASER kept passes through the trash it is detected as plastic or else it is paper. Once the detection is done, the waste is pushed to respective bins with the help of Servo Motors.

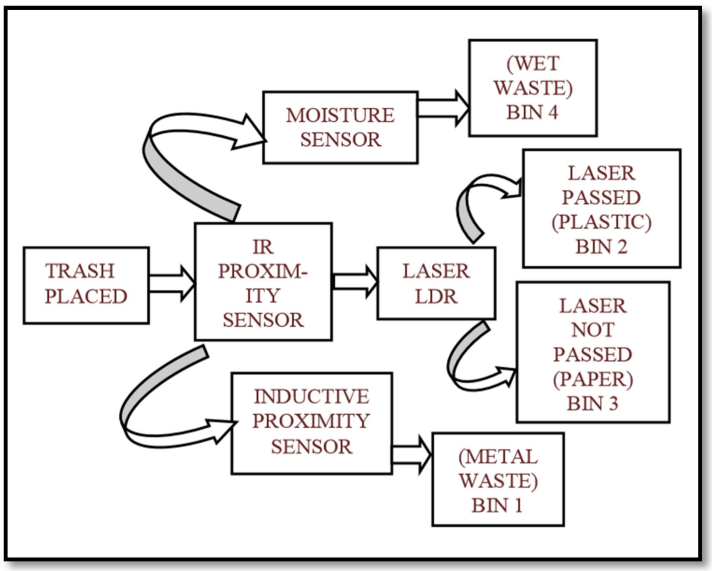

Figure 2.3: Block diagram of Proposed System

Even though this system could be used in large scale, there are limitations. The limitations include the size of trash. It should be minimum of 30mm width. The system can segregate only one type of waste at a time. Further improvements are required for it to be actually used in public places. The system is to be constructed as per that size and more sensors should be used to detect more objects and multiple objects at the same time.

2.4 Waste segregation using Machine learning [6]

This paper was presented by Yesha Desai, Asmita Dalvi, Pruthviraj Jadhav, Abhilash Baphna by University of Mumbai in International Journal for Research in Applied Science & Engineering Technology (IJRASET).

The proposed idea was to create an autonomous system which segregates the waste using CNN algorithm in machine learning. The algorithm detects and classifies the waste according to the dataset provided to CNN. The algorithm classifies the waste as biodegradable and non-biodegradable. The waste material is recognized based on shape and size of the objects. Raspberry Pi is used to capture the image and to push the classified object to respective bin.

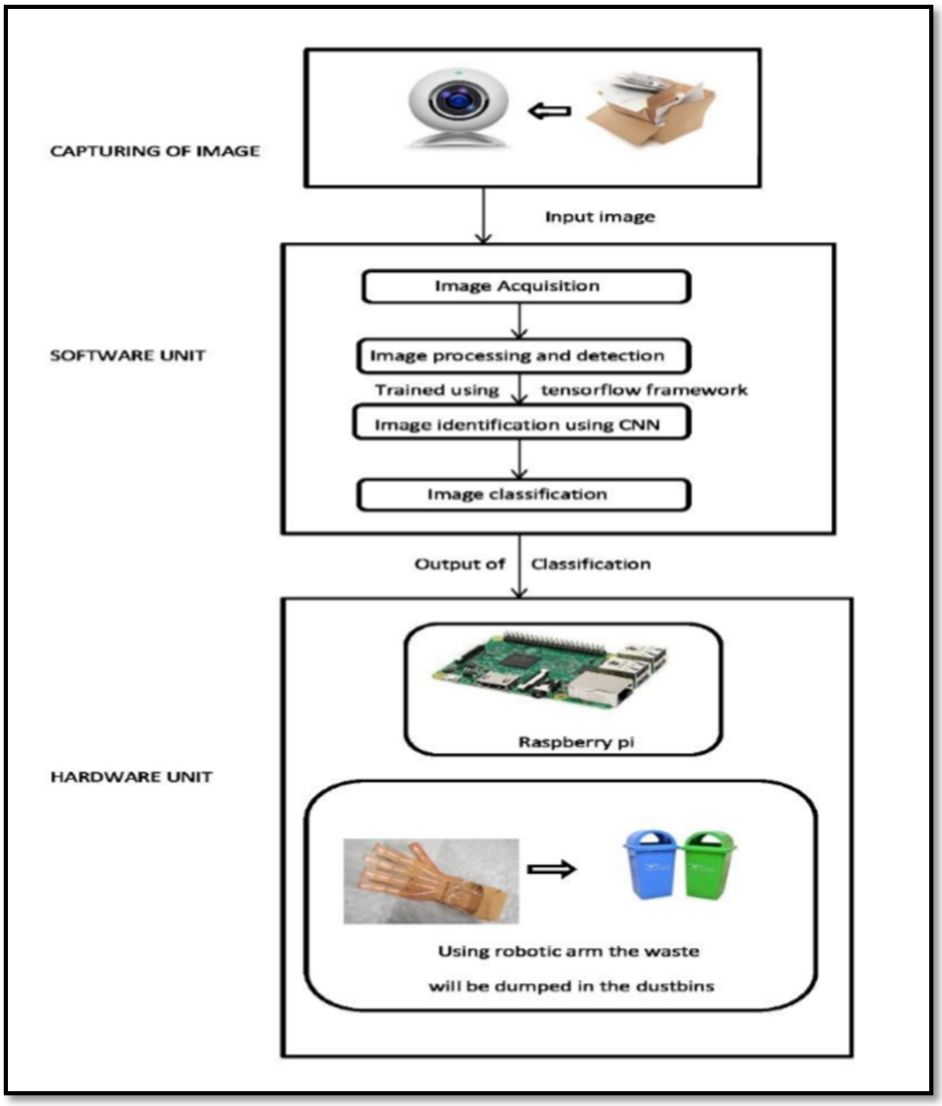

Figure 2.4: Architecture diagram for Waste Segregation

2.5 Smart Bin for Waste Management System [7]

This paper was presented by Sreejith S, Sanjay Kumar A, Ramya R, Roja R from SNS College of Technology in 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS).

The proposed solution empowers waste monitoring personnel by notifying when the fill-level or safe gas emission levels are surpassed. The proposed smart bin consists of a number of sensors. Once the bin gets filled it will automatically move to garbage collecting area, dispose the waste and return back to its place. A rain sensor is used to sense rain and close the bin automatically; a gas sensor is used to detect the gas level and to alert the passersby with the buzzer. All these sensors are connected to Arduino UNO. A DC motor is used to move around. The process is repeated by means of a microcontroller.

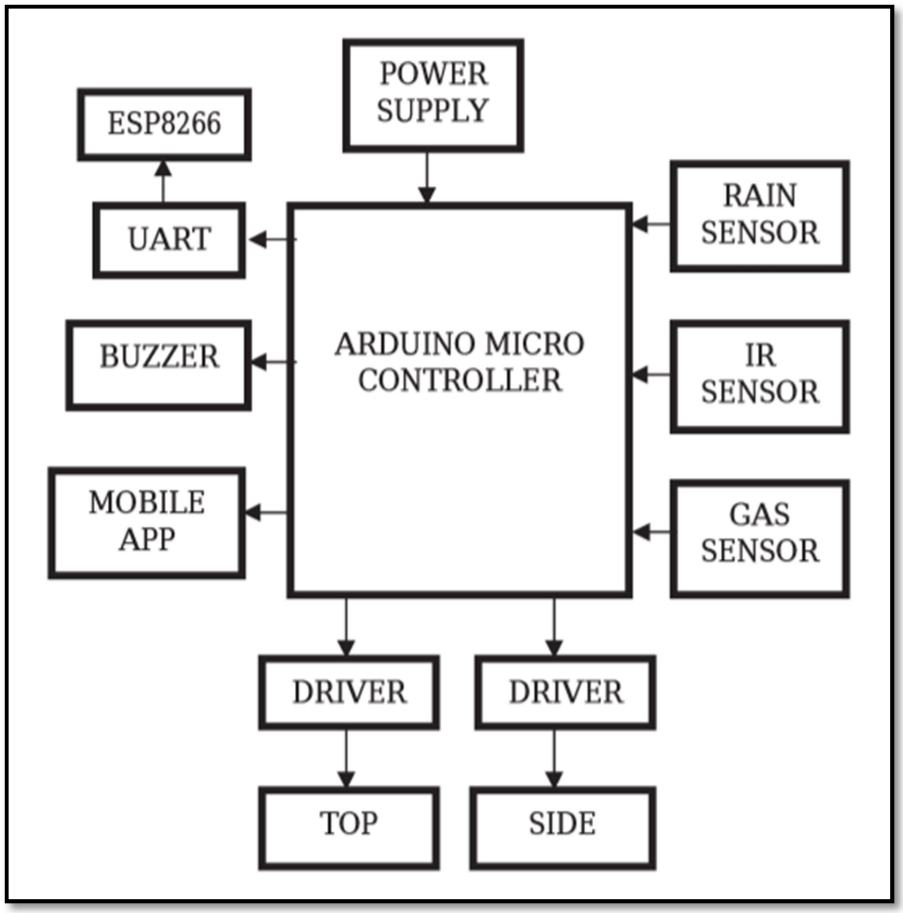

Figure 2.5: Block diagram of the proposed system

2.6 Smart Garbage Segregation & Management System Using Internet of Things (Iot) & Machine Learning (ML) [8]

This paper was presented by Shamin N, P Mohamed Fathimal, Raghvendran R and Kamalesh Prakash from SRM Institute of Science and Technology, Chennai in 2019 1st International Conference on Innovations in Information and Technology (ICIICT).

The paper proposes IoT stationed smart waste segregation and management device which detects the wastes using sensors and the information is directly transferred to cloud database via IoT. Microcontroller is used as a mediator between the sensors and IoT system. Numbers of sensors were used to detect the waste class. Ultra-sonic sensor was used to detect the presence of waste. Moisture sensor is used to detect the moisture content in waste and metal sensor to separate metal items from the rest of the waste. Image processing is used to identify other plastics and degradable items. The dustbin data were uploaded to cloud database.

SURF algorithm is used for feature extraction of the images. KNN algorithm is used to compare the test image with dataset images. The accuracy of this system was 99%. The limitation of the system was only one waste material can be put into the dustbin at a time. The quality of the image decides the classification accuracy. And if the so kept material is not a part of the database it affects the result. Due to these limitations, this system was not fit to be used in large scale.

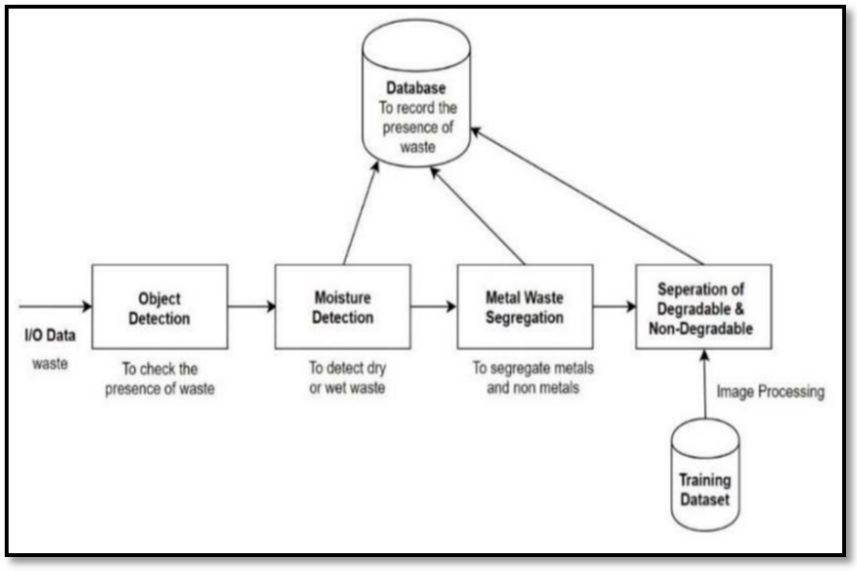

Figure 2.6: Architecture diagram of the proposed system

2.7 Electronically assisted automatic waste segregation [9]

This paper was presented by Nandhini S, Mrinal Sharma, Naveen Balachandran, K Suryanarayana, D S Harish Ram from Amrita School of Engineering in Third International Conference on Trends in Electronics and Informatics (ICOEI 2019).

The proposed system is based on a robotic assembly and machine learning based classification. The robotic arm is used to move the object from one place to the classifier platform. The robot senses the object using the ultrasonic sensors to calculate the distance to the target object. It uses CNN based classifier for identifying the class, a robotic arm is driven by servo motor for handling the waste. Arduino board is used for controlling the assembly.

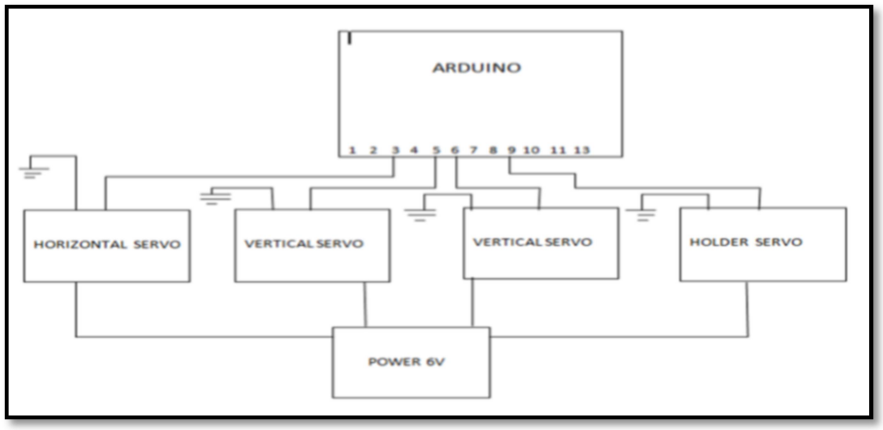

Figure 2.7: Block diagram of the proposed system

2.8 Eco-Friendly IOT Based Waste Segregation and Management [18]

This paper was presented by Santhosh Kumar B.R., Varalakshmi N, Soundarya S. Lokeshwari, Rohit K, Manjunath and Sahana D.N. of K.S. Institute of Technology, Bangalore at the International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2017.

The proposed idea was to use an STM32 controller to create an eco-friendly waste segregator that would segregate between biodegradable, metal and plastic waste. The waste that is being dropped into the smart dustbin is segregated right at the source, thus preventing

any improper waste management in the future. This method is proposed to be used in primary waste generation locations. This model is equipped into the dustbin itself.

The model when compared with the technology previously used waste segregation methods shows the following advantages:

Parameters	Existing Methods	Proposed method
Technology	[1] Microcontroller, [2] Microcontroller, [3] Android app, [4] Sound frequency, [5] Microcontroller, [6] Image Processing,[7] IOT, [8] Artificial intelligence.	STM 32 Industry level microcontroller. IOT is used for continuous monitoring of physical devices.
Segments	[1] Plastic/paper, Biowaste, glass. [2] Dry, wet and metal. [3] No segregation. [4] Plastic bottles, tin cans. [5] Dry, wet, metal. [6] Recyclable and Nonrecyclable. [7] No segregation. [8] Bio and Nonbiodegradable.	a)Biodegradable waste with proper chemical treatment b) Metallic waste segment coated with acrylic to avoid reactance. c) Plastic wastes.
Sensors	[1],[2],[5]- Capacitive, inductance, Moisture, IR proximity. [3],[4],[6],[7],[8]-No sensors used.	Moisture sensor, inductive sensor, capacitive sensor, gas sensor, Bacteria sensor.
Compatibility	The above methods occupy large space and it is complex in nature.	Area occupied is less. It is simple in nature. User friendly.

Table 2.1: Comparison with Proposing Methods

The model segregates any waste dropped in the bin at the panel with the help of sensors and the corresponding valves on the segment are opened and the waste is dumped into their respective segment. Furthermore, alarming levels of microbe activity in the biodegradable segment are controlled with sensors and chemical treatment. The sensors in this segment include gas sensors for methane gas produced and even controls the odour of the gas. The system is equipped with a Wi-Fi module for connectivity with a data service that

will continuously monitor the threshold levels of the waste in the bin. Once a threshold level is reached, it alerts that the garbage needs to be disposed. Similarly, for metal and plastic waste, inductive and capacitive sensors are used. Metal objects dumped are coated with an acrylic coating to prevent reactivity. Overall, the bins are equipped with level sensors to indicate the collected waste levels and sends the status message with the means of the Wi-Fi module. Further appropriate disposal of each segment can be carried out.

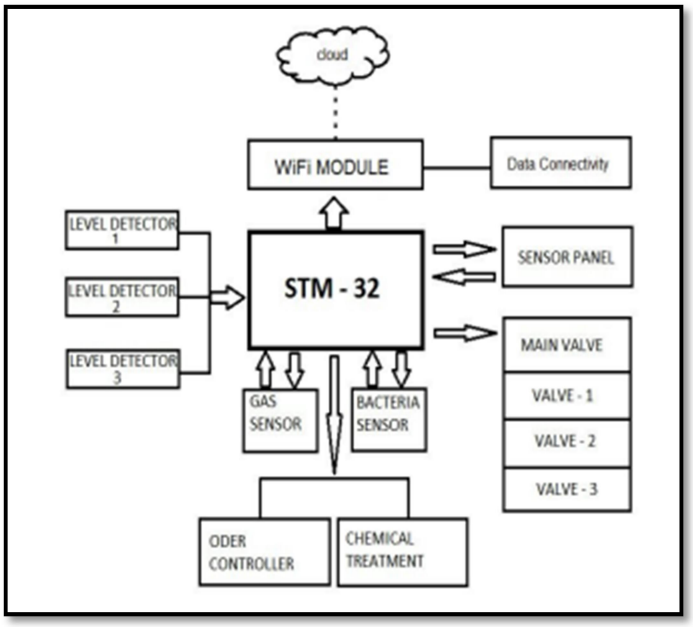

Fig 2.8: System Segregation Model

The system is as shown above, and while it considers many aspects of waste collection and segregation in a timely manner, this model is not practical with different mixes of waste at the same time as it can only detect one type of waste at the segment at a time.

2.9 You Only Look Once: Unified, Real-Time Object Detection [19]

This paper was presented by Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi from University of Washington and published in 2016.

This paper develops a new approach to object detection in the form of an algorithm of the name You Only Look Once (YOLO) for real-time object detection using a 24-layered

convolutional neural network. YOLO frames the object detection as a regression problem to spatially separated bounding boxes and associated class probabilities instead of repurposing classifiers. By this method, the neural network passes the image through it only once and performs the detection by predicting bounding boxes and class probabilities directly from the images. It is a single network pipeline and therefore optimized for fast detection performance.

This algorithm works on the principle of a fully connected convolutional network with 2 YOLO layers for different scaled object detection and accuracy. It breaks the images up into grids and assigns class probabilities to each grid while predicting up to 2 bounding boxes per grid. Since this only happens with a full confidence of an object existing within the specific grid, it ignores whitespace. Once it has made these predictions in each grid, it scales back and looks at the whole image with global context. This way it unifies the objects in the image while simultaneously being real-time with its detection.

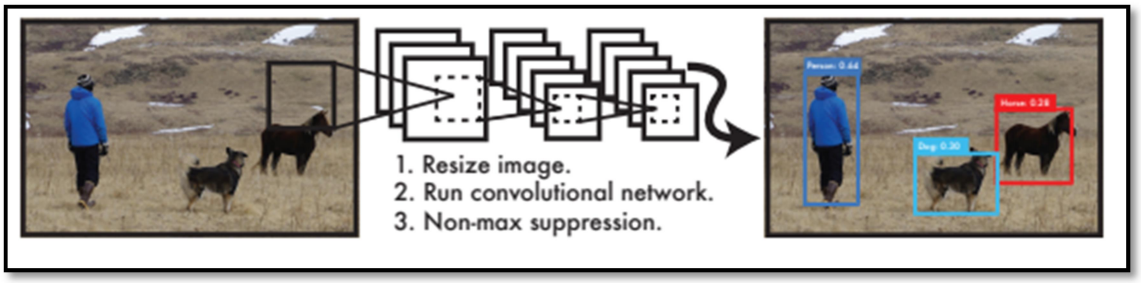

Fig 2.9: The YOLO Detection System

YOLO learns the generalized representations of objects to use during detection, by training on natural images. This algorithm outperforms top detection methods like DPM and R-CNN by a wide margin. This model is less likely to break down when applied to new unidentified objects as it is highly generalizable. However, despite it lagging behind some state of the art-detection systems in accuracy, it makes up for it in speed and is real-time. These trade-offs are insignificant for the use case of garbage segregation.

To further examine the differences between YOLO and state-of-the-art detectors, experiments are run on the VOC-2007 and the results are broken down. Fast-RCNN and YOLO are compared as Fast-RCNN is one of the highest performing detectors on PASCAL and its detections are publicly available. This comparison is done by taking top N predictions

for each category at test time. Each prediction is either correct or it is classified based on the type of error:

- Correct: correct class and IOU >.5
- Localization: correct class, .1<IOU<.5
- Similar: class is similar IOU>.1
- Other: class is wrong IOU >.1
- Background: IOU <.1 for any object

The breakdown of each error type is shown in the figure, across all 20 classes of VOC-2007.

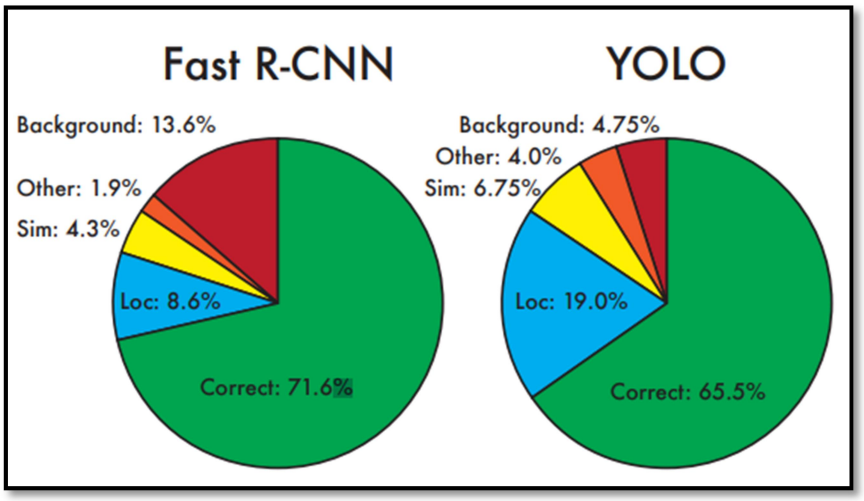

Fig: 2.10: Error Analysis: Fast R-CNN vs. YOLO

Most of the errors committed by YOLO are localization errors in comparison to all other sources, while R-CNN makes fewer of these but far more background errors. 13.6% of its top detections are false positives that don't contain any objects. This is almost completely avoided in YOLO and therefore much more effective for the use case of waste detection in a heap pile. Fast R-CNN is almost 3 times more likely to predict background detections than YOLO.

When similar experiments are run on real-time systems, a different factor is considered, mainly the speed of the detection, which is a far greater factor to consider when dealing with constantly changing objects for detection. Here YOLO performs fairly well,

with only two other competitors in terms of real time detection, both of which are Deformable parts Models (DPM), as shown in the table below.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 2.2: Real-time Systems on PASCAL VOC 2007

Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real time detector. YOLO is 10mAP more accurate than the fast version while still well above real-time in speed. YOLO achieves a 6-times faster speed than the VGG-16 version of Faster R-CNN and 2.5 times higher speed along with more accuracy against The Zeiler-Fergus Faster R-CNN.

2.10 YOLOv3: An Incremental Improvement [20]

This paper was presented by Joseph Redmon and Ali Farhadi from University of Washington in 2018 as an update on the previous version of You Only Look Once (YOLO).

Many design changes made to the original version of YOLO resulted in a much more accurate YOLO network. Here a new trained network proves to be the biggest change. At 320x320 YOLOv3 runs in 22 ms at 28.2 mAP., which is just as accurate as Single Shot Detector (SSD) but 3x faster.

One of the many design changes include the way that bounding boxes are predicted. An objectness score is predicted for each bounding box using logistic regression, and it will maximise at 1 if the bounding box prior overlaps a ground truth object more than any other

bounding box prior. Another design change would be class prediction which now uses binary cross-entropy loss. Independent logistic classifiers replaced softmax for better performance. Another feature to this version would be prediction across 3 different scales and feature extraction across these scales, similar to feature pyramid networks. This resulted in addition of several convolutional layers. The final layer in these predicts a 3-d tensor encoding bounding box, objectness and class predictions. Feature extraction, a new network is used which uses a hybrid approach with YOLOv2, Darknet-19 and residual networks. This new network is comprised of successive 3 x 3 and 1 x 1 convolutional layers with some shortcut connections as well. This makes the network much larger i.e. 53 convolutional layers so it is called Darknet-53.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.11: Darkent-53

This network proves to be much more powerful than Darknet-19 but still more efficient than ResNet-101 or ResNet-152. The results shown are measured on ImageNet in

terms of accuracy, Billions of Operations, Billion floating-point operations per second and FPS for various networks.

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

TABLE 2.3: Comparison of Backbones of Networks

All networks are trained with identical settings and tested at 256 x 256, single crop accuracy. Thus Darknet-53 performs on par with state-of-the-art classifiers but with fewer floating-point operations and more speed. This network structure better utilizes the GPU, making it more efficient to evaluate and thus faster.

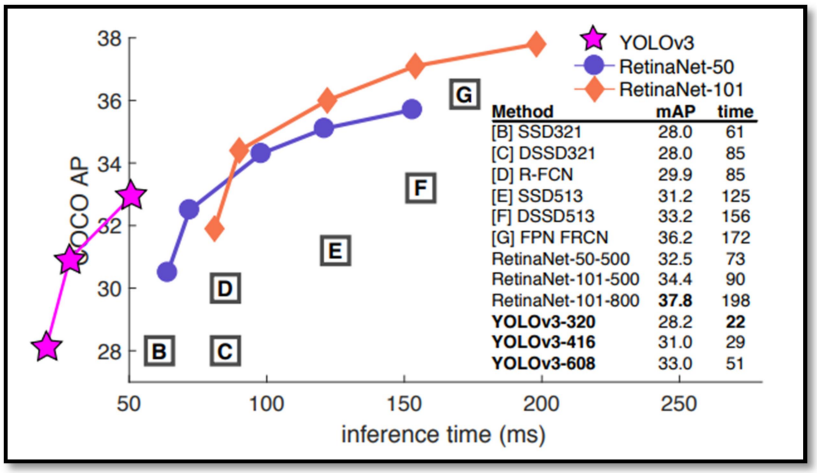

Figure 2.12: YOLOv3 Run Time vs. Other Methods

Thus, with these changes made, and a larger network with more efficient designs, YOLOv3 runs significantly faster than other detection methods with comparable performance. Times are from either an M-40 or Titan X which are similar GPUs.

A smaller version of the more efficient YOLOv3 is also constructed in order to avoid usage of GPUs when testing, and faster detection with lower usage of disk space. This is the version the proposed garbage detection model uses that is loaded into the Raspberry Pi and detects the waste.

Chapter 3

COMPARATIVE RESEARCH

3.1 Object detection using CNN

A CNN is a supervised learning technique which needs both input data and target output data to be supplied. These are classified by using their labels in order to provide a learned model for future data analysis.

Typically a CNN has three main constituents - a Convolutional Layer, a Pooling Layer and a Fully connected Dense Network. The Convolutional layer takes the input image and applies m number of *n x n* filters to receive a feature map. The feature map is next fed into the max pool layer which is essentially used for dimensionality reduction, it picks only the best features from the feature map. Finally, all the features are flattened and sent as input to the fully connected dense neural network which learns the weights using backpropagation and provides the classification output.

The motivation behind the CNN is that it is based on the way the visual cortex functions, where one object in the scene is in focus while the rest is blurred, similarly the CNN takes one section/window of the input image at a time for classification. Each time the CNN will produce a feature map for each section, in the convolutional layer. In the Pooling layer it removes the excess features and takes only the most important features for that section, thereby performing feature extraction. Hence, with the use of CNNs we don't have to perform an additional feature extraction technique.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets. Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

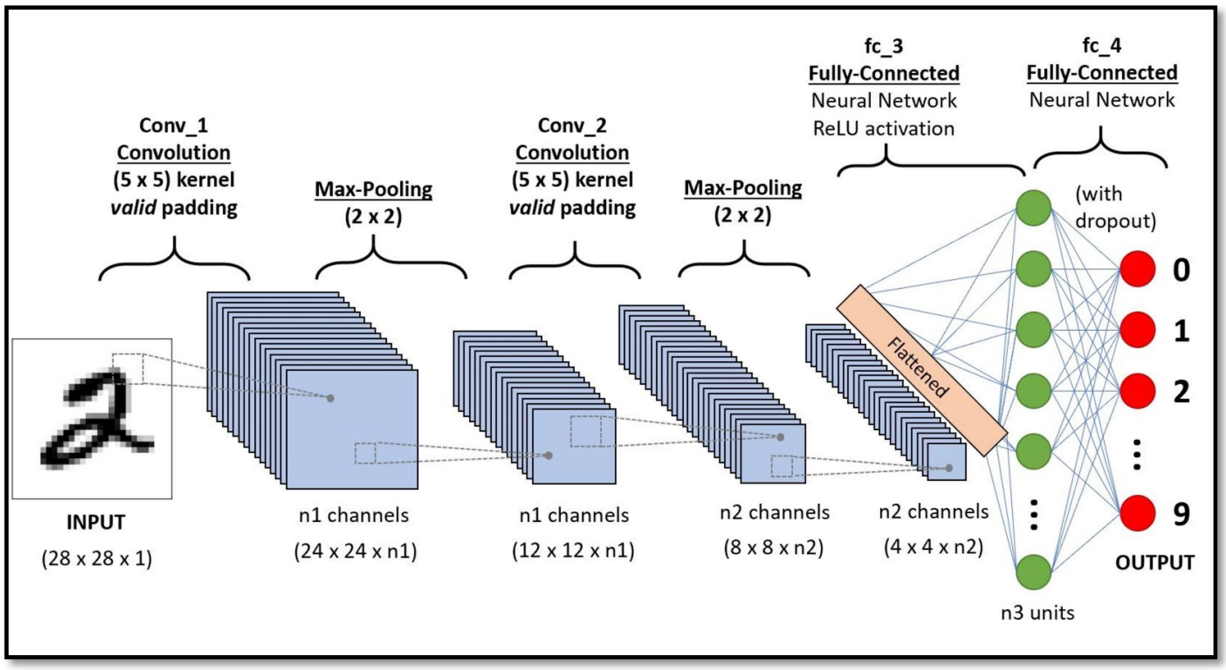

Figure 3.1: Convolutional Neural Network for MNIST dataset

CNNs require lesser pre-processing as compared to other similar classification algorithms. While traditional MLP (Multi-Layer Perceptron) algorithms have significant accuracy for image recognition, they suffer from the curse of dimensionality due to the nodes being fully connected, and hence cannot be scaled to high resolution images. CNNs overcome these challenges posed by MLP by exploiting the spatial correlation of an image. This is done by enforcing a pattern of local connectivity between adjacent neuron layers. Hence, CNNs prove to be superior at Image classification, Video Analysis, Natural Language Processing and wide range of other applications as compared to other techniques.

3.2 Object detection using R-CNN [21]

In this paper "Rich feature hierarchies for accurate object detection and semantic segmentation" by R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, the authors introduced a fundamental concept for all modern object detection networks: combining region proposals with CNNs. They called it as R-CNN.

This approach is a combination of 2 key insights:

1. One can apply high-capacity CNNs to bottom-up region proposals to localize and segment objects
2. When the training data is scarce, supervised pre-training followed by domain-specific fine-tuning, gives significant performance boost.

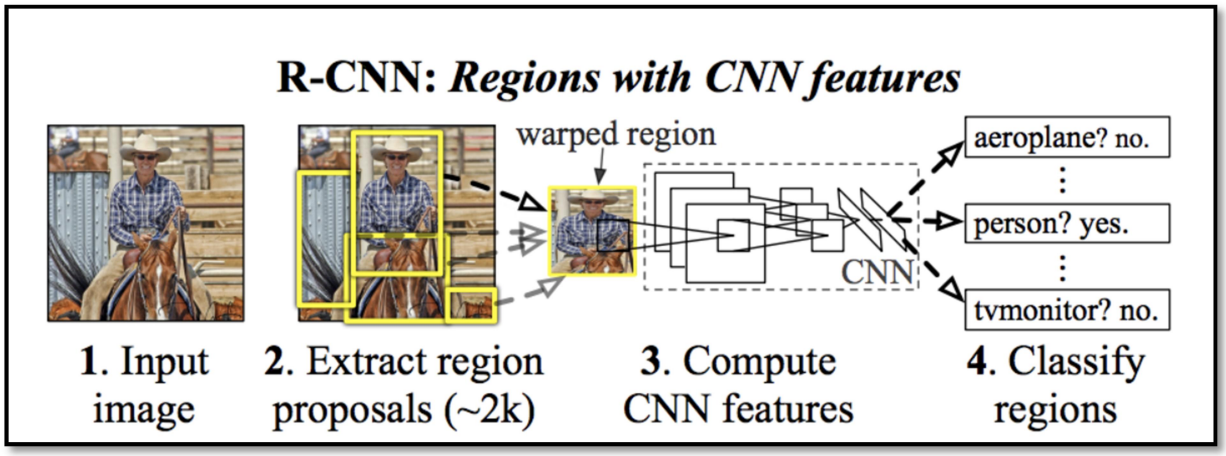

Figure 3.2: Object Detection System overview

This system consists of 3 modules. First module generates category-independent region proposals. Second module is a large Convolutional Neural Network. Third is set of SVMs.

3.2.1 Region proposals

Selective search method is used to generate region proposals. This algorithm is based on computing hierarchical grouping of similar regions based on colour, texture, size and shape compatibility. Selective search starts by over-segmenting the image based on intensity of the pixels. These oversegments are used as an initial seed and then add all bounding boxes

corresponding to segmented parts to the list of regional proposals, group adjacent segments based on similarity. These steps are repeated to form larger segments. This Selective Search method will create nearly 2000 different regions.

3.2.2 CNN

Each region proposal is taken individually and a feature vector representing this image is created using Convolutional Neural Network (CNN).

3.2.3 SVM

Once the feature vector is created, we need to identify the classes each object belongs to. SVM is a classifier used for this purpose. SVM gives the output as confidence levels.

3.3 Object detection using Fast R-CNN [22]

R-CNN was a major breakthrough using region based CNNs. But it had problems:

- Slow: Calculating a feature map for each region proposal makes it slow.
- Hard to train: R-CNN system consists of different parts (CNN, SVM) that have to be trained separately. This makes training very difficult.
- Large memory Requirement: Saving every feature map for each region proposal takes up a lot of memory.

In this paper "Fast R-CNN" by R. B. Girshick, the author tries to solve these problems. The different parts of R-CNN are combined to one architecture.

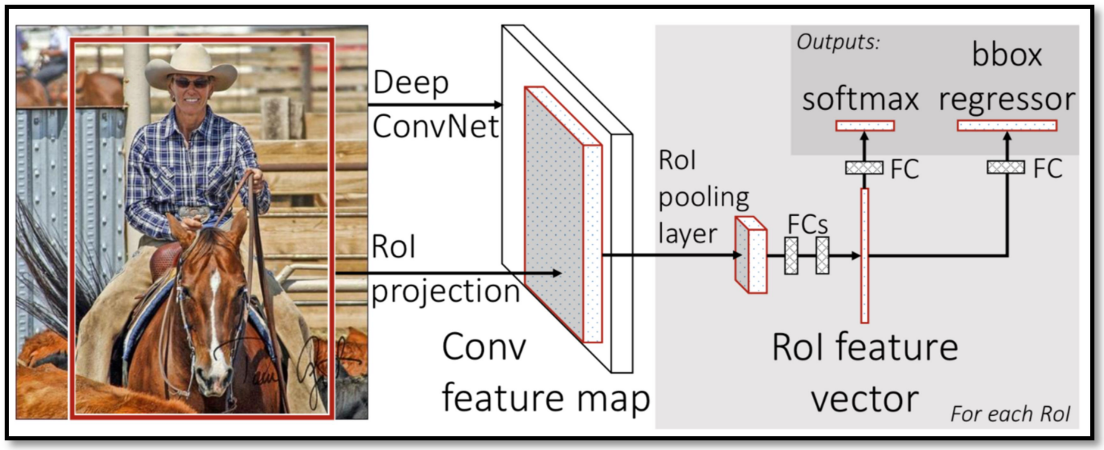

Figure 3.3: Fast R-CNN architecture.

This architecture works like this:

1. Whole image is processed with CNN and gives the feature map as the output.
2. Using this feature map, for each region proposal extract the corresponding part called as Region Proposal Feature Map. Pooling layer is used to resize all the region proposal feature map to a fixed size.
3. This fixed region proposal map is flattened. Map becomes a feature vector which doesn't change its size.

This part has 2 fully connected layers that have 2 outputs. First is Softmax layer which decides the object class, second is Bounding Box Regressor, where output is a bounding box coordinates for each object class.

3.4 Object detection using Faster R-CNN [23]

Fast R-CNN also uses selective search to find region proposals. This Selective search makes it slow and time consuming process affecting the performance of the network.

In this paper, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, tries to solve this problem by using a different approach than Selective Search. Faster R-CNN lets the network learn the region proposals.

Like Fast R-CNN, the feature map is provided by the CNN. To identify the region proposals a separate network is used to predict them. Predicted region proposals use RoI (Region of Interest) pooling layer to reshape the images and then used to classify the image within that proposed region and predict values for bounding boxes.

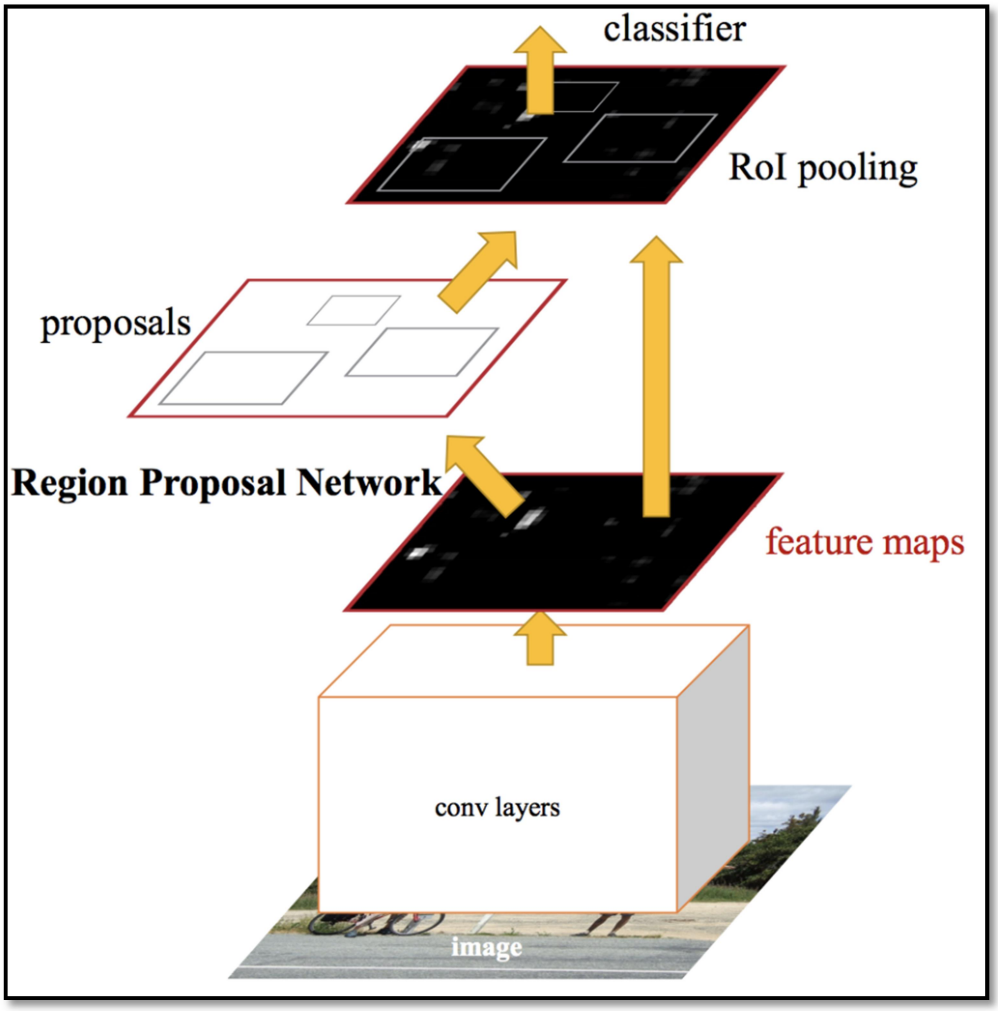

Figure 3.4: Faster R-CNN Architecture

3.5 Object detection using YOLO [24]

You-Only-Look-Once (YOLO) and Faster RCNN both have some similarities. But R-CNN techniques primarily use regions to localize the objects within the images. YOLO framework takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. YOLO is incredible fast and can process 45 Frames Per Second. This makes YOLO as one the best algorithms for object detection.

The overall process of YOLO is very simple. YOLO takes an input image. Then the image is divided into grids. Image classification and localization are applied on each grid.

A single neural network which predicts bounding boxes and class probabilities directly from full images in one evaluation is being explained.

A single convolutional network simultaneously predicts multiple bounding box coordinates and class probabilities. Base network runs at 45 frames per second on GPU and fast version runs at 155 frames per second. While streaming video in real-time we have less than 25 milliseconds of latency.

3.5.1 Model

Input image is divided into S × S grid. B bounding boxes are defined in each grid cell with confidence score. Confidence is nothing but the probability an object exists in each box.

Confidence is defined as:

$$\mathbf{C = Pr\ (Object) * IOU}_{\boldsymbol{pred}}^{\boldsymbol{truth}}$$

Where **IOU** is Intersection Over Union, **truth** is ground truth and **pred** is prediction.

Intersection is overlapping area between predicted box and truth, and union is the total area between both predicted and truth. IOU should be close to1 which indicates that predicted bounding box is close to truth.

Simultaneously, C conditional class probability is predicted. Class specific probability for each grid cell is defined as:

$$\mathbf{Pr\ (Class_i \mid Object) * Pr\ (Object) * IOU}_{\boldsymbol{pred}}^{\boldsymbol{truth}} \mathbf{=\ Pr\ (Class_i) * IOU}_{\boldsymbol{pred}}^{\boldsymbol{truth}}$$

These probabilities are encoded to tensor of a fixed dimension. Tensor dimension is defined as: **S ×S ×(B ∗5+C)**

Based on the considered YOLO network, the model produces 7 × 7 × 30.

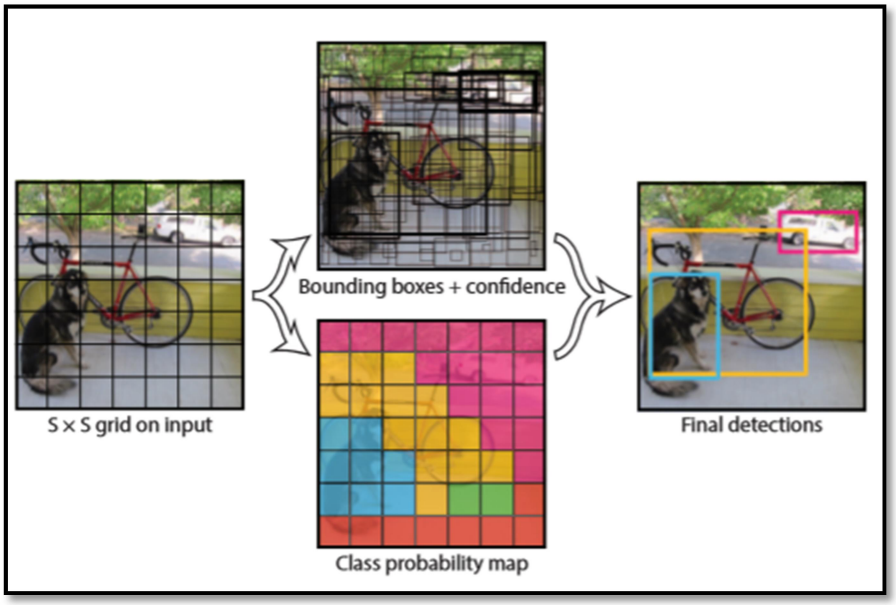

Figure 3.5: The model

3.5.2 Network Design

This network is inspired by GoogLeNet model for image classification. This network has 24 convolutional layers followed by 2 fully connected layers.

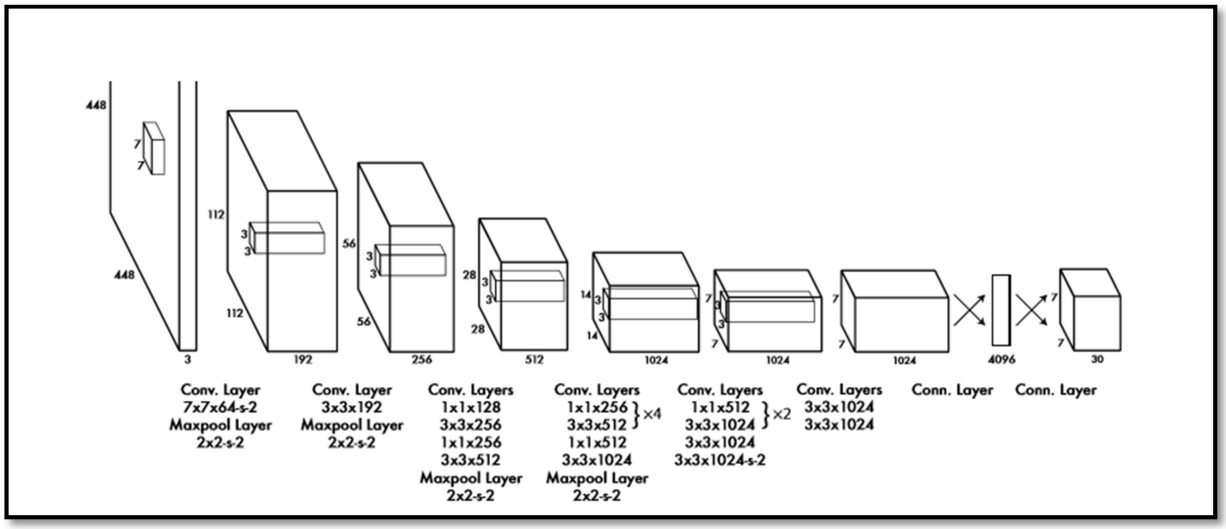

Figure 3.6: The Architecture

Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. Convolutional layers are pre-trained at half resolution for image classification and then double the resolution for detection. Final output of this network is the 7 × 7 × 30 tensor of predictions.

3.5.3 Training [25]

For pre-training we use the first 20 convolutional layers, average-pooling layer and a fully connected layer. Final layer predicts class probabilities and bounding box coordinates. Bounding box height and width are to be normalized by image height and width so that they fall between 0 and 1.

Linear activation function is used for final layer and all other layers uses leaky rectified linear activation which is defined as:

$$\phi(x) = \begin{cases} x, if\ x > 0 \\ 0.1x, if\ otherwise \end{cases}$$

YOLO predicts multiple bounding boxes per cell. One predictor is assigned to predict an object based on the highest current IOU with ground truth.

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}$$

Equation for Loss function

The loss function is used to correct the center and bounding box of each prediction. λ_{coord} and λ_{noobj} variables are used to increase emphasis on boxes with objects and lower the emphasis on boxes with no objects. C refers to confidence and p(c) refers to classification prediction.

YOLO has been improved with different versions such as YOLOv2 or YOLOv3. Condensed versions of these versions are also available which are smaller and faster than their original versions.

Model	Layers	FLOPS (B)	FPS	mAP	Dataset
YOLOv1	26	not reported	45	63.4	VOC
YOLOv1-Tiny	9	not reported	155	52.7	VOC
YOLOv2	32	62.94	40	48.1	COCO
YOLOv2-Tiny	16	5.41	244	23.7	COCO
YOLOv3	106	140.69	20	57.9	COCO
YOLOv3-Tiny	24	5.56	220	33.1	COCO

Table 3.1: Performance of each version of YOLO

3.6 Arduino and Raspberry Pi

Raspberry Pi (RPi) board is a credit card sized computer. It has memory, processor and graphics card. It also has a specially designed Linux OS that is easy to install. RPi doesn't offer internal storage but SD cards can be used in place of it. For network connectivity, SSH and FTP can be used. Arduino board is an interface for devices, sensors and for any hardware project. It allows us to better manage devices and actuators.[26]

RPi is useful when Internet connectivity is necessary. It is completely Linux based and hence makes things easier. It can be programmed with various programming languages.

Arduino is very easy to get started for real-time applications. Not much programming practice and experience is necessary to get started with working with Arduino. With RPi, accessing hardware is not real-time. Sometimes there is a delay. There is no built-in Analog to Digital converter.

RPi hardware is not open source. Arduino is not as powerful as RPi. Only programming languages like C/C++ can be used to program the Arduino. Connecting to the Internet is difficult. If any project requires Internet connectivity, less hardware interaction and is slightly complex on the software side, then RPi is the best option. If we need/want to use other languages other than C/C++, then RPi is a better option. [27]

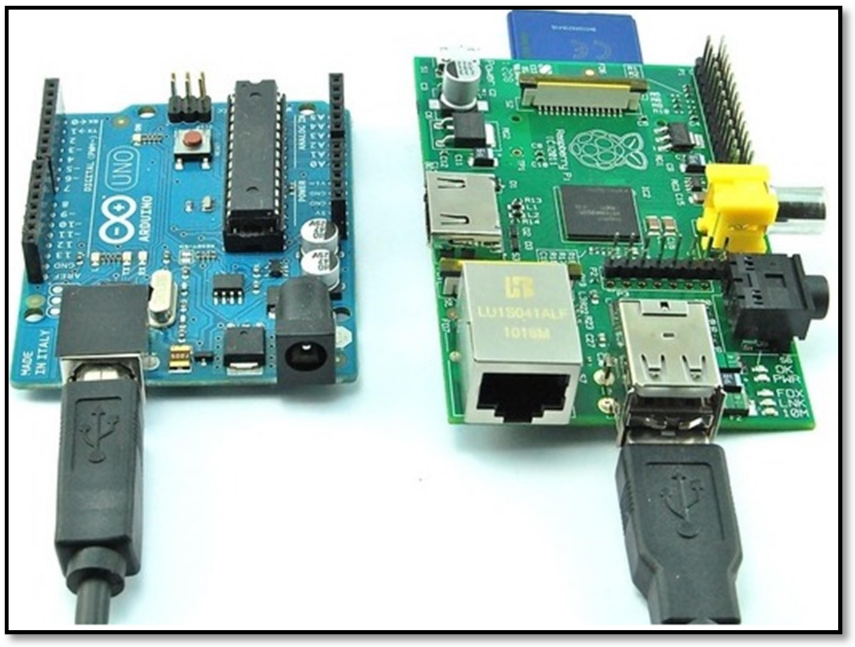

Figure 3.7: Arduino board (left), Raspberry Pi Model 3 (right)

S NO.	ARDUINO	RASPBERRY PI
1.	Control unit of Arduino is from Atmega family.	While control unit of Raspberry Pi is from ARM family.
2.	Arduino is based on a microcontroller.	While Raspberry Pi is based on a microprocessor.
3.	It is designed to control the electrical components connected to the circuit board in a system.	While Raspberry Pi computes data and produces valuable outputs, and controls components in a system based on the outcome of its computation.
4.	Arduino boards have a simple hardware and software structure.	While Raspberry Pi boards have a complex architecture of hardware and software.
5.	CPU architecture: 8 bit.	CPU architecture: 64 bit.
6.	It uses very less RAM, 2 kB.	While Raspberry Pi requires more RAM, 1 GB.
7.	It clocks a processing speed of 16 MHz.	While Raspberry Pi clocks a processing speed of 1.4 GHz.
8.	It is cheaper in cost.	While Raspberry Pi is expensive.
9.	It has a higher I/O current drive strength.	While Raspberry Pi has a lower I/O current drive strength.
10.	It consumes about 200 MW of power.	While it consumes about 700 MW of power.

Table 3.2: Arduino vs Raspberry Pi

3.7 CPU and GPU

CPU (Central Processing Unit) is a device primarily acts as the brain for every embedded system. It consists of an ALU (Arithmetic Logic Unit) used to temporarily store the data and perform calculations and a CU (Control Unit) which performs instruction sequencing and branching. It also interacts to the other units of the computer such as memory, input and output, for executing the instruction from the memory this is the reason an

interface is also a crucial part of the CPU. The I/O interface is sometimes included in the control unit. [28]

It provides address, data and control signal while receives instructions, data, status signal and interrupt which is processed with the help of the system bus. A system bus is a group of various busses such as address, control and data bus. The CPU assigns more hardware unit to fast cache while low to computation, unlike GPU.

The GPU (Graphics Processing Unit) is a processor specifically designed for computing the graphical displays. It is typically incorporated with CPU for sharing RAM with CPU which is good for the most computing task. It is needed for the high-end graphics intensive processing. The discrete GPU unit contains its own RAM known as VRAM for video RAM. The advanced GPU system cooperatively works with the multi-core CPUs. At first, the graphics unit was introduced by the Intel and IBM in the 1980s. These cards were enabled with simple functionalities such as area filling, alteration of simple images, shape drawing and so on. [29]

The modern graphics are capable of performing the research and analysis task, often surpassing CPUs because of its extreme parallel processing. In the GPU the several processing units are stripped together where no cache coherency exist.

S.NO	CPU	GPU
1.	CPU stands for Central Processing Unit.	While GPU stands for Graphics Processing Unit.
2.	CPU consumes or needs more memory than GPU.	While it consumes or requires less memory than CPU.
3.	The speed of CPU is less than GPU's speed.	While GPU is faster than CPU's speed.
4.	CPU contain minute powerful cores.	While it contain more weak cores.
5.	CPU is suitable for serial instruction processing.	While GPU is not suitable for serial instruction processing.
6.	CPU is not suitable for parallel instruction processing.	While GPU is suitable for parallel instruction processing.
7.	CPU emphasis on low latency.	While GPU emphasis on high throughput.

Table 3.3: CPU vs GPU

Chapter 4

SYSTEM REQUIREMENTS

System requirements are the configuration that a system must have for a hardware and software for the proposed idea to run easily and proficiently.

4.1 Hardware Specifications

4.1.1 Raspberry Pi [10]

Raspberry Pi is a single-board computer developed by Raspberry Pi foundation. It was originally developed for teaching purposes. Now it is widely used in research projects because of its low cost and portability. Raspberry Pi 3 Model B was the model used in this project.

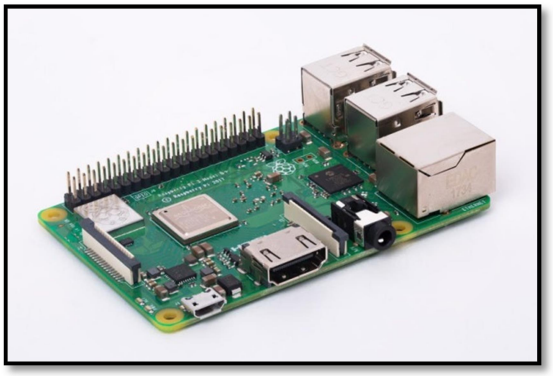

Figure 4.1: Raspberry Pi 3 Model B

Specifications

Processor:	64-bit SoC 2 1.4GHz
Memory:	1GB SDRAM
Connectivity:	2.4GHz and 5GHz wireless LAN
	4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
SD card support:	Micro SD format

Input power: 5V/2.5A DC via micro USB controller

5V DC via GPIO header

4.1.2 Pi Camera [11]

Camera modules which are official products from Raspberry Pi Foundation are called Pi cameras. Camera module is used to take high-definition video and photographs. Raspberry Pi Module v2 is the camera module used in this project. The v2 camera module has a Sony IMX219 8-megapixel sensor. It supports 1080p30, 720p60 and VGA90 video modes and still captures.

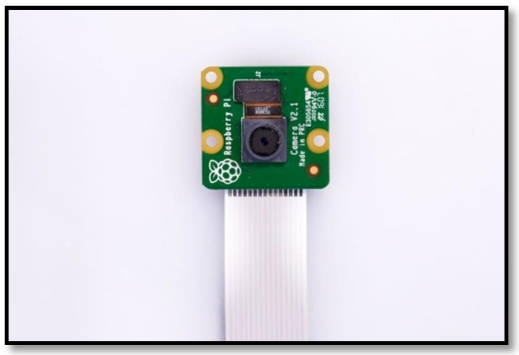

Figure 4.2: Raspberry Pi Camera Module v2

4.1.3 Robotic Arm [12]

The robotic arm used in this project is "Aluminium Alloy 4 DOF Manipulator Steering Gear Bracket Mechanical Paws". It is a 4-DOF manipulator designed from multiple servo sets.

Figure 4.3: Robotic arm

Arm Parameters

Working Voltage:	4.8-6V
Paw weight:	68g
Maximum opening angle of the paw:	52mm
Arm height:	35.5cm
Overall length of paw:	115mm (when paw is closed)
	110mm (when paw is opened)

4.1.4 Servo Motor [12]

Servo motors are DC motors that allow for precise control of the angular position. The servo motors have a revolution cut-off from 90˚ to 180˚. The servo motors used in this project are MG996R. It is a metal gear servo motor with features mentioned below:

Operating Voltage:	+5V
Current:	2.5A(6V)
Stall Torque:	9.4 kg/cm
Operating speed:	0.17s / 60˚

Rotation: 0˚ - 180˚

Weight of motor: 55gm

Figure 4.4: MG996 Servo motor

It has 3 wires:

1. Brown Ground wire connected to ground of system
2. Red Powers the motor
3. Orange PWM signal is given in through this wire to drive the motor

4.2 Software requirements

This section gives a detailed description of the software requirement specification. This allows the developer or analyst to understand the system, function to be carried out the performance level to be obtained and corresponding interfaces to be established.

4.2.1 Google Colab [13]

Figure 4.5: Google Colab Logo

Collaboratory or "Colab" is a product from Google Research. It allows us to write and execute Python in browser with free access to GPUs, zero configurations required and easy sharing. Colab allows anybody to write and execute python code through the browser. Colab notebooks are Jupyter notebooks that are hosted by Colab.

Colab supports Python 2.7 and 3.6. There is a limit to the sessions and size for the colab notebooks. It is easy to work with deep learning libraries like PyTorsch, Keras, TensorFlow and OpenCV using Google colab.

Google colab provides free NVIDIA Tesla K80 GPU of about 12 GB. A GPU provides 1.8TFlops and has a RAM of 12GB. GPU allocation is restricted to 12 hours at a time per user. We can connect your session to Google Drive as an external storage.

4.2.2 Darknet Framework [14]

Figure 4.6: Darknet Framework Logo

Darknet is an open source neural network framework written in C and CUDA. It supports both CPU and GPU computation. Darknet has 2 optional dependencies: OpenCV for wide variety of supported image types, CUDA for GPU computation. This framework features YOLO, a state-of-the-art, real-time object detection system. It has a mAP of 78.6% on VOC 2007 and 44.0% on COCO dataset.

Darknet displays information as it loads the config file and weights then it classifies the image and prints top classes for the image. This framework can be used to run RNN. RNNs are powerful models for representing data that changes over time and Darknet can handle them without using CUDA or OpenCV. The framework also allows its users to venture into game-playing neural networks.

4.2.3 YOLO framework [15]

Figure 4.7: YOLO Framework Logo

YOLO is a state-of-the-art, real-time object detection system. On a Pascal Titan X, it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev. YOLO framework applies a single neural network to the full image. The network divides the image into regions and predicts bounding boxes and probabilities for each region.

YOLO makes predictions with a single network evaluation. It is more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

4.2.4 OpenCV [16]

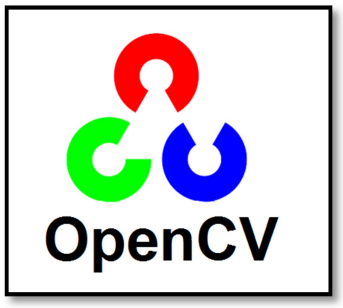

Figure 4.8: OpenCV Logo

OpenCV is an open source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications. OpenCV was originally developed by Intel. OpenCV is written in C++ and its primary interface is in C++. There are bindings in in Pyhton, Java and MATLAB.

OpenCV runs on windows, Linux, macOS, etc. It can also run on mobile Operating systems like Android, iOS, Blackberry 10. OpenCV library has more than 2500 optimized algorithms. It includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, track camera movements, track moving objects, etc.

4.2.5 PIGPIO [17]

pigpio is a Python module for the Raspberry to control the GPIO. The pigpio Python module can run on Windows, MACs, or Linux. It controls one or more Pi's. It is useful in creating and transmitting precisely timed waveforms, reading/writing GPIO and setting their modes. Transmitted waveforms are accurate to a microsecond. This module uses the services of the C pigpio library. pigpio must be running on Pi(s) whose GPIO are to be manipulated.

Chapter 5

SYSTEM ARCHITECTURE

5.1 System structure

The proposed system divides the whole system into modules. First module deals with collection of dataset and pre-processing of the images. Pre-processing of the images include labelling them with the classes and finding the height and width of every object in the image given. Second module deals with creation of model. Creation of module consists of training the model and testing the model for accuracy and precision. This training and testing process is repeated until a practical accuracy is reached. Third module deals with capturing the image and sorting the objects using the model. Image captured by the camera is saved and cropped to the required size. Cropped image is given to the model to detect the objects in the image and to obtain the coordinates of each object in image. These coordinates are given to arm to sort the objects accordingly.

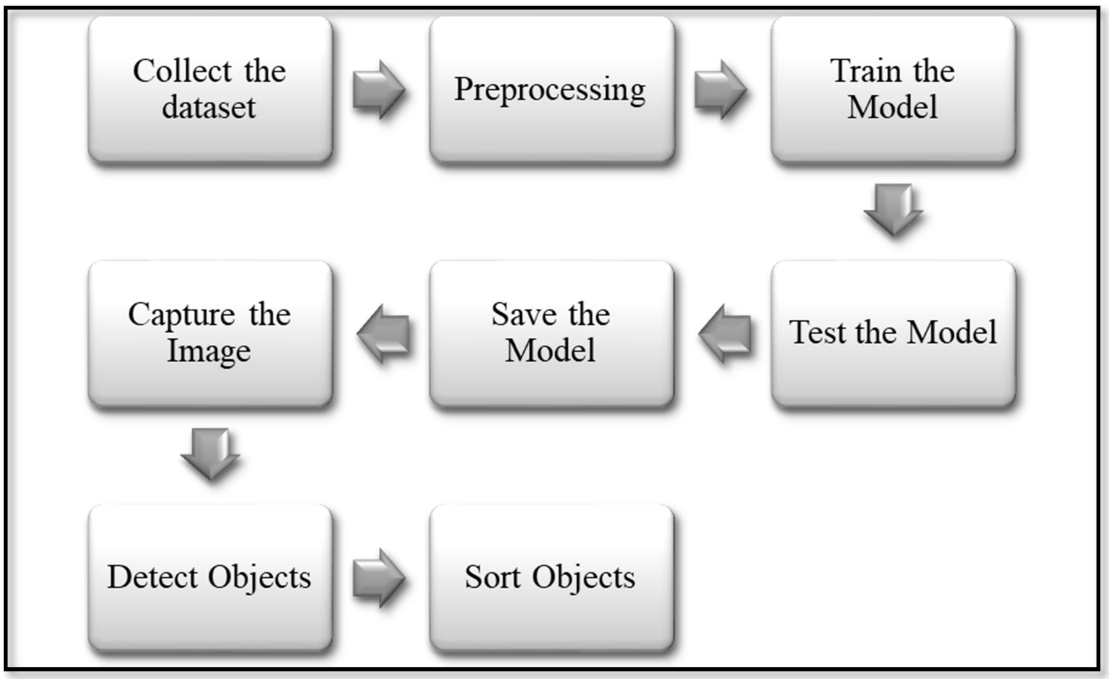

Figure 5.1: Overall Block diagram

5.2 Data flow diagram

A flowchart is a diagram which depicts the algorithm and/or process as a sequence. Different symbols are used to represent different processes in flowchart. The order of the process is shown by arrows connecting the boxes for different kinds of statements.

Data flow diagrams can be of 2 types, Logical DFD and Physical DFD. Figure 4.2 shows physical DFD of the proposed system. It describes the processes in detail and shows all processes regardless of it being manual or automated.

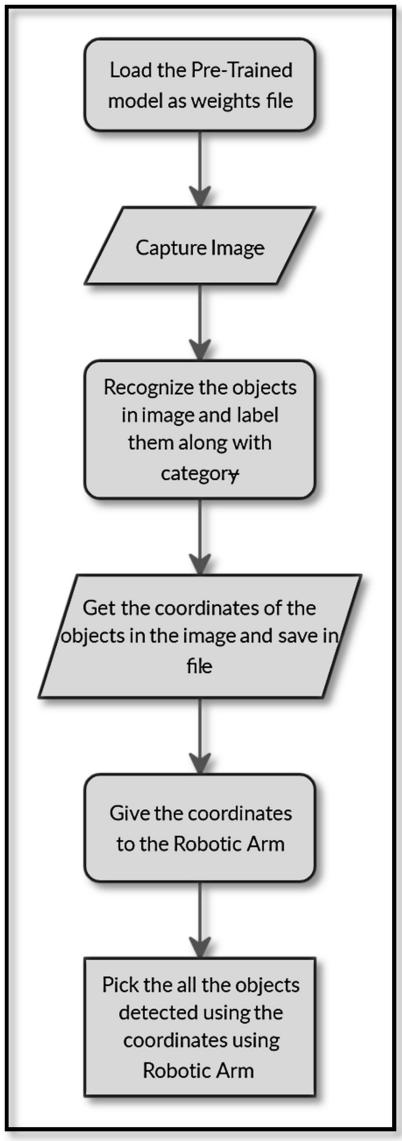

Figure 5.2: Data Flow diagram of proposed system

5.3 Use case diagram

A use case diagram is a simple way of representing the interaction of a user with the system. It shows the relationship between the user and the system. It is also known as behaviour diagram as it depicts the behaviour of the system with the external actors. A use case diagram summarizes relationship between use cases and actors.

Here the user, system and arm are considered as actors that interact with each other for the final output.

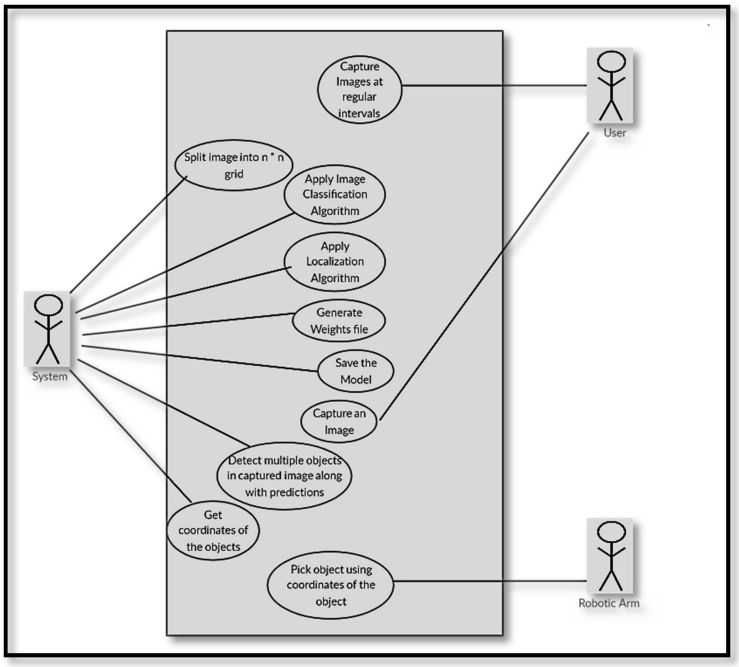

Figure 5.3: Use case diagram

5.4 Sequence diagram

A sequence diagram depicts the active processes that live simultaneously as vertical lines. The horizontal arrows show the messages that are being transferred from one process or object to another. The messages given are in the order that they are exchanges form the top to bottom of the sequence diagram. The sequence diagrams are also called as event diagrams or event scenarios as they represent the various events that occur in the system in order.

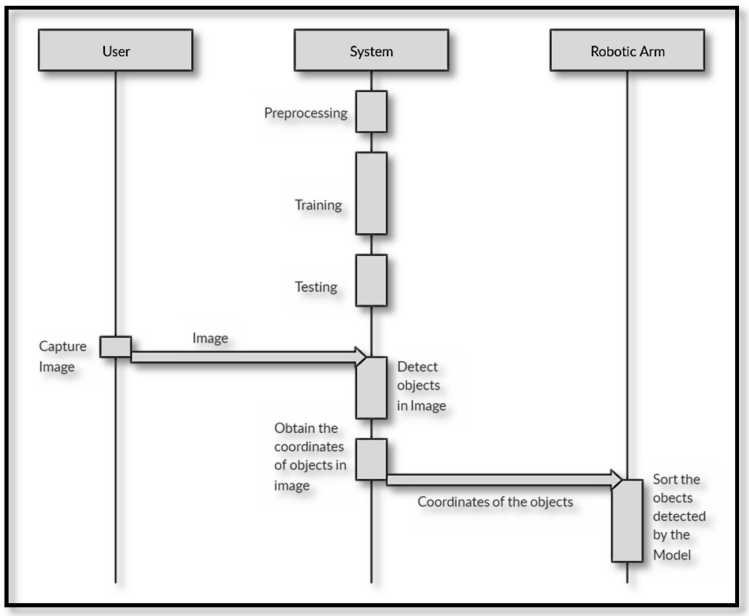

Figure 5.4: Sequence diagram

Chapter 6

METHODOLODY AND IMPLEMENTATION

6.1 Introduction

This chapter covers the methodology and the practical implementation of our proposed model. In order to bring the thesis to life, we have several factors to consider before we can implement the system. The mapping out of these stages is known as the methodology. It can be dissected into many modules or steps, such as:

- Programming Language Selection
- Installation and Configuration of Darknet and YOLO Framework
- Training & Testing of Tiny-YOLO model
- Construction of Robotic Arm
- Real Time Garbage Detection and Segregation

The implementation stage consists of proper planning, detailed study of the existing system and its constraints and designing of any alternative methods and their evaluation.

6.2 Programming Language Selection

For this project, the selection of the programming is crucial, as we require a language that is optimal for both remote computation of YOLO weights and Raspberry Pi configuration. The ideal choice for this was therefore Python.

6.2.1 Python

Python is an object-oriented scripting language which is very easy to learn. Hence it is also commonly called as a beginner's language. It is an interpreted and interactive language. This language was created by Guido van Rossum and its implementation began in 1989 at Centrum Wiskunde & Informatica (CWI), Netherland. Python is basically developed under and open source license which has been approved by OSI. Hence, this language is free for use and distribution (even for commercial use).

The main reasons for using Python in our proposed system are:

1. Easy to learn and use.

2. OpenCV library & Darknet frameworks are crucial features and effortlessly accessible in Python.

3. Python provides interfaces to Raspberry Pi processor and therefore streamlines the configuration of the hardware used in this project.

4. The Python language is portable and scalable.

5. Python provides simple & quick integration of GPUs.

6.3 Darknet and YOLO Framework

As mentioned earlier, Darknet can be used with OpenCV and CUDA. For this project, we have made use of both the features. The framework is basically written in C but it has an excellent integration in Python.

6.3.1 Installation

The process of installation is straightforward and done through an open source git repository created by Joseph Redmon. For our needs however, an updated version of the repository created by AlexyAB has been used as this supports Windows platforms as well as streamlining the training and validation process.

Once the base system framework of Darknet has been compiled, we can make use of the tiny yolo model present.

6.3.2 Dataset Creation

In order to use the tiny-yolov3 algorithm, a dataset of the types of waste needed to be segregated has to be compiled. This dataset consists of at least 1000 different images of one

type or class of waste. We have compiled our dataset by scraping the web as well as making use of Kaggle, a hub of open source datasets for the world of Machine Learning.

Our model attempts to segregate the recyclable waste into 5 classes, namely: Wood, Metal, Paper, Plastic and Glass. The waste in the dataset of over 7000 images is accurately labeled and saved for the model to read. Organic waste is ignored for the large part, and not segregated; hence we have no use for images of such kind.

This dataset, once labeled is split into training and test sets at a ratio of 90-10 and loaded into the darknet framework for training.

6.3.3 Tiny-YOLOv3

From the many algorithms available on the darknet framework, we have chosen Tiny-Yolov3 as it is a succinct neural network of 24 convolutional layers and 2 yolo layers.

The tiny version of YOLO allows us to make garbage detection on a small processing unit such as a Raspberry Pi 3 and therefore ideal for the project.

YOLO allows the detection and segregation to be made quickly and accurately, since the images will be passing through a single neural network. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

The default mAP of the YOLO system is 57.9% on the universal COCO dataset mainly used to predict large objects. However, we are utilizing this system to detect more undefined objects like waste with a considerably smaller dataset of jpeg images.

6.4 Pseudocode

The working model can be differentiated into various modules. The algorithm of the complete sequence is presented below in Figure 6.1. This sequence continues to function as long as the number of objects presented in front of the system is greater than zero.

It follows through on the complete procedure of segregation, beginning with capturing an image of the workspace below, transmitting the image to the YOLO network which

processes it in real time. The network returns the updated number of objects, midpoints of the objects as well as its classes so as to dispose of them appropriately. 'Predictions.jpg' image contains the final output of the software module complete with bounding boxes and their labels. The function `Pick_Up_Recyclable_Waste()` uses the co-ordinates returned to move the robotic arm to the location of the recyclable objects while the `Move_Arm_to_Bin()` function uses the default bin position to throw the recyclable waste. This bin position can be configured to different locations depending on the class predictions given to that particular object. After each object, the arm resets using `Reset_Arm()` function to come to rest. This continues for all obtained co-ordinates.

```
Algorithm Segregation:
While(NoOfObjects > 0)
    Image = Capture_Image();
    NoOfObjects,midpoints,Predictions = Yolo(Image);
    Show_Image(Predictions.jpg);
    For x,y in midpoints:
        Pick_Up_Recylable_Waste(x,y)
        Move_Arm_to_Bin(bin_position)
        Reset_Arm()
End while
```

Figure 6.1: Algorithm for Automatic Waste Segregation

The detection of recyclable objects takes place within the YOLO function where the image is partitioned into 13x13 cells in which it is capable of detecting up to five objects per cell. The image is only processed once through the tiny-YOLO network and each cell is examined carefully where a class prediction and bounding box prediction is done and appended to the final predictions of the entire image. This is demonstrated in the algorithm for YOLO in Figure 6.2.

```
Algorithm Yolo(Image):

#default image size is 260x260
read(TestImage)
NoOfClasses= 5
#image is partitioned to 13x13 parts
NoOfCells = 13
#size of step to take when moving across the image.
step = height(image)/NoOfCells
#stores class-prediction of each class
predictionClass = array[NoOfCells, NoOfCells, NoOfClasses]
#stores 2 bounding box suggestions for each cell
#cell will have 2 bounding boxes, with
#x, y, width, height and confidence predictions.
predictionsBox = array [NoOfCells, NoOfCells, NoOfCells, NoOfCells]
#final_predictions in which the final list of
#predictions will be added
final_predictions = []
for (i,j) in image:
    #each cell will be of size (step, step)
    cell = image(i:i+step,j:j+step)
    #First make a prediction on each cell
    predictionClass = class_predictor(cell)
    #Predictions of bounding box is made based on
    #x, y, width, height and confidence values
    #2 suggestions are made
    predictionsBox[i,j] = boundingBoxPredictor(cell)
    best_box = [0 if predictionsBox[i,j,0,5] > predictionsBox [i,j,1,5] else 1]
    #Get the class which has the highest probability
    predicted = maxValueIndex(predictionClass[i,j])
    #the prediction is an array which has the x, y
    #coordinate of the box the height and the width
    prediction = [predictionsBox[i,j,best_box,0:5], predicted]

    final_predictions.append(prediction)

NumberOfObjects = length(final_predictions)
midpoints = CalculateMidpoints(final_predictions)

return(NumberOfObjects, midpoints, Predictions.jpg)
```

Figure 6.2: Algorithm of Tiny-YOLOv3 Network Used

As explained, the number of cells the image is divided into and number of classes are taken into consideration. For each grid cell in the 13x13 division, up to 5 objects can be detected and this detection is made by using the step size to traverse the image until all cells are analysed. The `predictionClass` stores the values predicted by the `class_predictor()` function for that particular cell. These predictions are then given bounding boxes (multiple) which is stored in `predictionsBox` array. The best box is determined by considering the bounding box with higher confidence. The highest probable class stored in `predictionClass` and then this value is attached with the bounding box

(`prediction` variable) and then appended to `final_predictions` array that stores all final predictions of final objects in the image.

Since these final predictions are stored in the array, the size of it should inform the program about the number of objects it has detected which can be used to keep the loop of segregation continuous. An additional function `CalculateMidpoints()` is used to find the correct co-ordinates where the robotic arm needs to travel to pick up the waste. The predictions, midpoints and number of objects are returned.

Once the co-ordinates of the recyclable objects are obtained, it is easily picked mapped to the workspace and segregated by the robotic arm. This is done repeatedly till no recyclable objects remain and new waste can be dumped.

6.5 Training

For the training of the system on the dataset of images, a default weights file is installed and loaded initially. This allows the model to work with more accurate numbers rather than a random value and provides a faster training experience and greater drops in loss values.

The initial weights used for our project is the tiny-yolov3.conv.15 file which is optimized for tiny yolov3. Moreover, further configurations are required for the network, as the resolution of images, number of classes, filters, maximum iterations, steps, batch sizes and learning rate is communicated to obtain the most accurate result.

These parameters are set and the model is allowed to train. For our project, the parameter values are shown as below:

Resolution of Image: 720p × 1080p

Number of Classes: 5

Filters: 15

Maximum Iterations: 10000

Steps: 8000,9000

Batch Sizes: 64

Learning Rate: 0.00001

For every 100 and 1000 iterations, it maps out the progress and saves the trained weights for safe-keeping. Once the maximum iterations is reached, the final weights file is saved for detection purposes.

Here the main indication of rising accuracy is the decrease in average loss with the escalation of the number of images trained. Object confidence and class prediction confidence also increases with each iteration. The ideal number of iterations to train is 2000 or more per the number of classes. In this project, the model is trained over 10000 iterations. The training is only done on the training set.

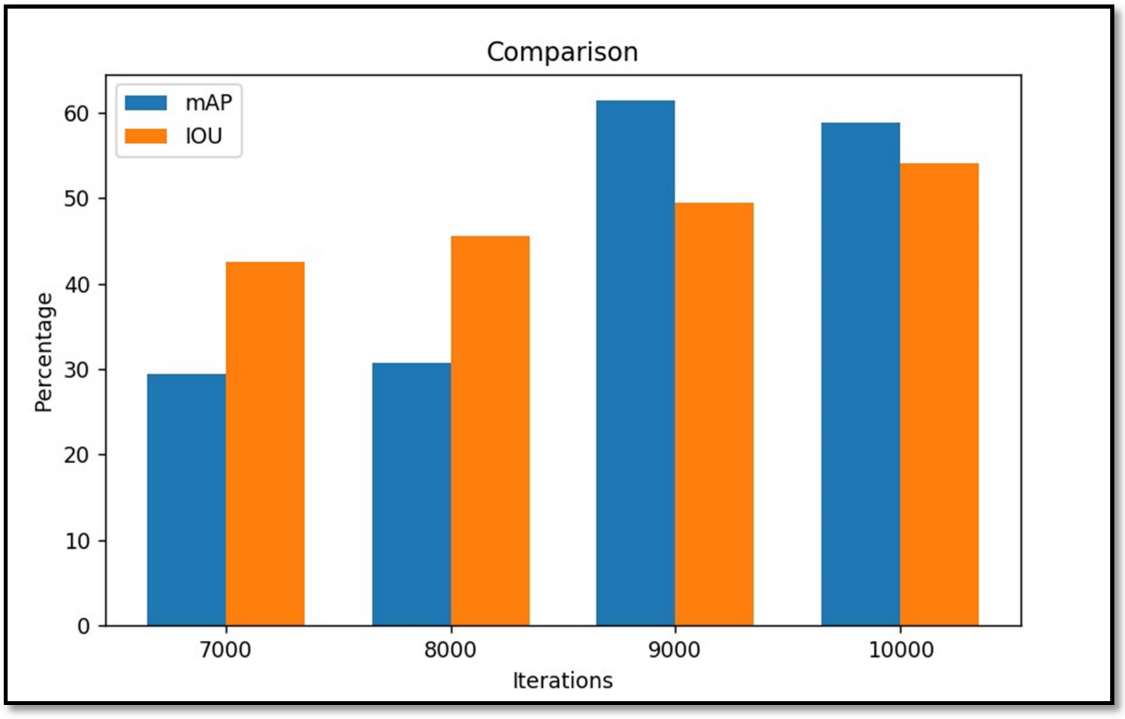

Figure 6.3: Comparison for different iterations

In order to ensure the continued decline in average loss, the learning rate is monitored and deceased exponentially every time the loss stalls or experiences an upturn. The initial learning rate specified is 0.01.

```
 10000: 1.728827, 1.593340 avg loss, 0.000000 rate, 1.077099 seconds, 640000 images
Saving weights to backup//yolov3-tiny-train_10000.weights
Saving weights to backup//yolov3-tiny-train_last.weights
Saving weights to backup//yolov3-tiny-train_final.weights
```

Figure 6.4: Trained model after 10000 (max) iterations

Training is stopped when maximum iterations are reached, or when average loss does not decrease after more than 100 steps, and the weights are saved.

6.6 Testing

The trained model of Tiny-YOLOv3 is validated or tested using the test set that was separated out initially. These images are tested on and the predicted boxes are matched with the labeled values to calculate mAP, Precision, Recall, F1 score, and the Intersection over Union (IOU) percentage.

The results of each of the weights files saved at every 1000 iterations is compared since it is not mandatory that the final weights are best suited for the dataset. Sometimes, an early stopping point can have the ideal weights that is normalized for both training and validation set. The figure below shows the progression of average loss values when the different weights are used on both training and validation sets.

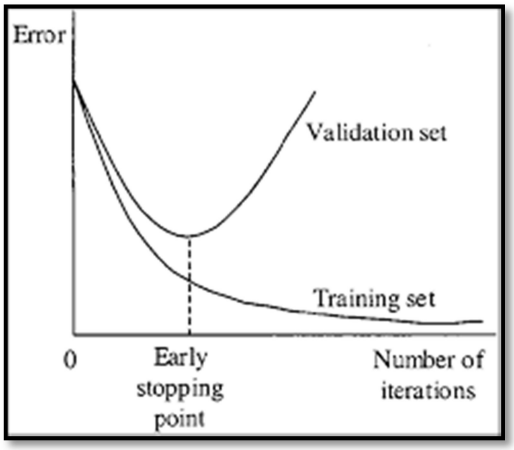

Figure 6.5: Factors for Weights File Selection

In that case, the weights file that gives the lowest loss with the highest mAP or IOU is considered for detection in future stages. For this project, the weights file generated at 9000 iterations gives the highest mAP for the dataset and therefore is considered for detection.

6.7 Configuration of Hardware Components

The construction of the hardware set up is as shown in figure 6.6:

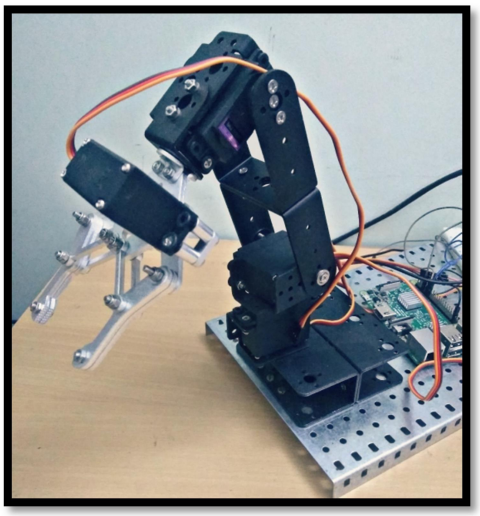

Figure 6.6: Robotic arm after hardware setup

The hardware model is constructed so that the robotic arm may move freely over the mapped workspace where the garbage is placed for segregation, with the camera suspended at an appropriate distance to capture the status of this grid with clarity.

The Raspberry Pi is used to control the 4 servo motors to guide the arm to these points on the grid. The pre-trained software tiny-yolov3 model is loaded into the raspberry pi unit so that it may perform the detection locally. The camera is configured to capture the image of the grid and transfer the knowledge to the processing unit.

A program is necessary to convert the co-ordinates of the waste in an image into co-ordinates on the grid.

The workspace is a 16x24cm rectangle with a white background so as to enable accurate detection of the waste. The arm is able to reach any point on this grid and is programmed to travel to a specified point, pick up the object in question and transfer it to the corresponding bin marked for the waste.

Once the hardware has been set up, it is ready to detect the waste and perform real time waste segregation.

6.8 Detection and Real-time Segregation

For the process of segregating waste, the model goes through the following steps:

- Capture image of waste on the grid & transfer image to Raspberry Pi processor
- Detect the waste type and position in the image using tiny-yolov3
- Obtain & Transfer the Co-ordinates of the objects to the robotic arm
- Segregate the waste into the appropriate bin using robotic arm

The waste detected is segregated into the categories of Wood, Metal, Paper, Plastic and Glass. Multiple types of waste and objects are detected simultaneously, but the segregation is done linearly. Once all the detected waste is segregated, the camera captures another image and the next set of waste goes through the same process.

This continues till there is no more waste to segregate.

Chapter 7

RESULTS AND OBSERVATIONS

The analysis of this project is done on basis of 2 modules. One is the software and other one is hardware. Software analysis is based on the values of mAP or IOU mainly and hardware analysis is based on how a fast and accurately can the arm sort objects.

For every 1000 iterations a weights file is updated as mentioned above. We need to choose best weights file to be used for detection. Weights file with highest mAP or IOU should be considered for the detection purposes.

IOU is an evaluation metric used to measure the accuracy of YOLO model on particular dataset. IOU is the average intersect over union of objects and detection for a certain threshold.

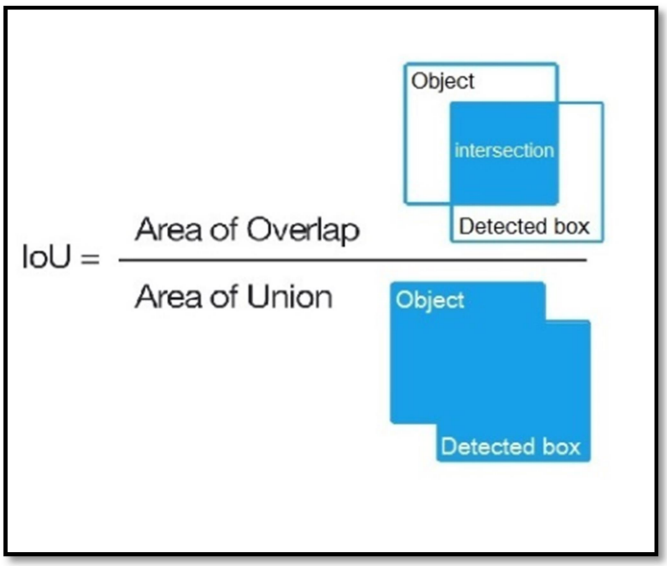

Figure 7.1: IOU measure

Precision is the ratio of true positive and the total number of predicted positives where true positive is when the prediction is positive and the ground truth is also positive.

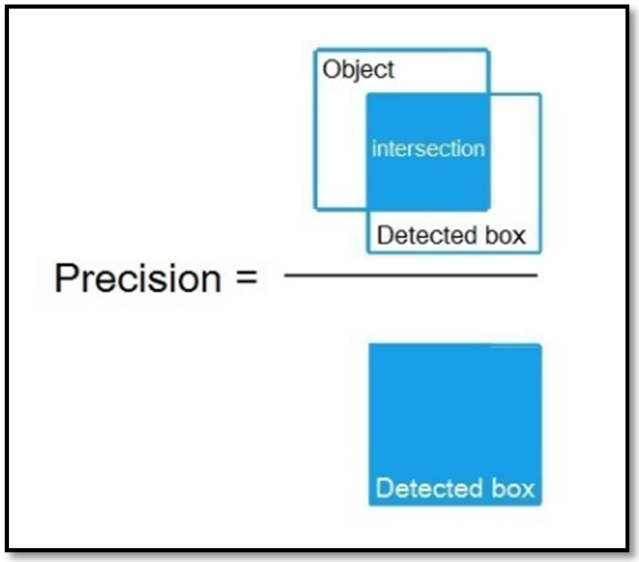

Figure 7.2: Precision measure

Recall is another evaluation metric which can be defined as ratio of true positive and the total of ground truth positives. It gives the basic idea about how good the model detects all objects in the data.

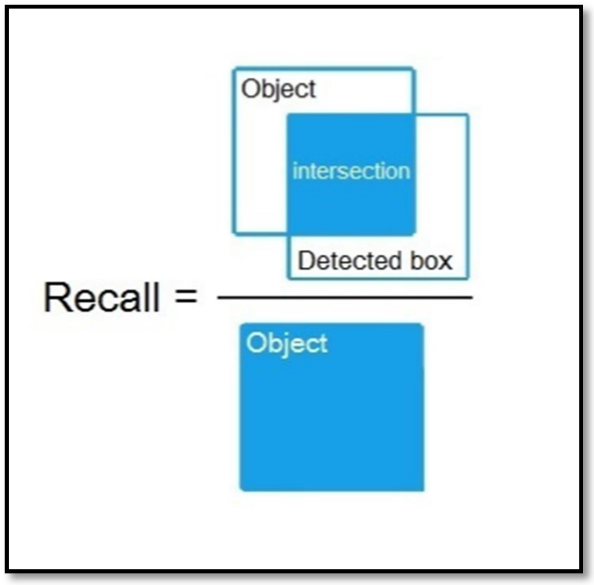

Figure 7.3: Recall measure

mAP is the mean value of 'average precisions for each class'. It can also be defined as the area under precision-recall curve. The highest mAP means that the model detects all the objects in the image as precisely as possible.

After training for 9000 iterations, the evaluation metrics values are as follows:

```
 calculation mAP (mean average precision)...
692
 detections_count = 8277, unique_truth_count = 1458
class_id = 0, name = Wood, ap = 52.13%           (TP = 18, FP = 11)
class_id = 1, name = Metal, ap = 67.52%          (TP = 150, FP = 93)
class_id = 2, name = Paper, ap = 63.03%          (TP = 211, FP = 105)
class_id = 3, name = Plastic , ap = 42.85%       (TP = 195, FP = 134)
class_id = 4, name = Glass, ap = 81.81%          (TP = 177, FP = 66)

 for conf_thresh = 0.25, precision = 0.65, recall = 0.52, F1-score = 0.57
 for conf_thresh = 0.25, TP = 751, FP = 409, FN = 707, average IoU = 49.49 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.614691, or 61.47 %
Total Detection Time: 8 Seconds
```

Figure 7.4: Evaluation metrics value

Command for detection:

```
darknet detector test data/obj.data cfg/yolov3-tiny-test.cfg
backup/yolov3-tiny-train_9000.weights data/DATASET/Metal13.jpg
```

where,

`data/obj.data`	Path to the .data file which consists of paths to other required files.
`cfg/yolov3-tiny-test.cfg`	Path to test configuration file.
`backup/yolov3-tiny-train_9000.weights`	Path to weights file.
`data/DATASET/Metal13.jpg`	Path to the image where objects are to be detected.

After this command is run, the model gives the output which consists of IOU, scale, time taken to detect the object and the confidence of the class of the detected object which is saved in a file called “result.txt”. The model created detects only the recyclable objects as it is trained to be. Along with this an image will be saved as “predictions.jpg” and displayed on the screen with a bounding box around the each object detected.

```
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 5.454
avg_outputs = 325691
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3-tiny-train_9000.weights...
 seen 64, trained: 576 K-images (9 Kilo-batches_64)
Done! Loaded 24 layers from weights-file
data/DATASET/Metal13.jpg: Predicted in 3.288000 milli-seconds.
Metal: 93%
```

Figure 7.5: Output after testing

Figure 7.6: Predictions.jpg for single object

If the image consists of multiple objects, multiple bounding boxes will be drawn over the image indicating multiple objects and the confidence of each class of each object will be displayed on the screen.

```
 GPU isn't used
mini_batch = 1, batch = 1, time_steps = 1, train = 0

 seen 64, trained: 512 K-images (8 Kilo-batches_64)
test.jpg: Predicted in 59206.175000 milli-seconds.
plastic : 35%   (left_x:  194   top_y:  274   width:   64   height:   79)
metal: 28%      (left_x:  198   top_y:  117   width:   61   height:   80)
paper: 31%      (left_x:  366   top_y:  272   width:  163   height:   84)
plastic : 37%   (left_x:  410   top_y:  108   width:  105   height:   83)
```

Figure 7.7: result.txt for multiple objects

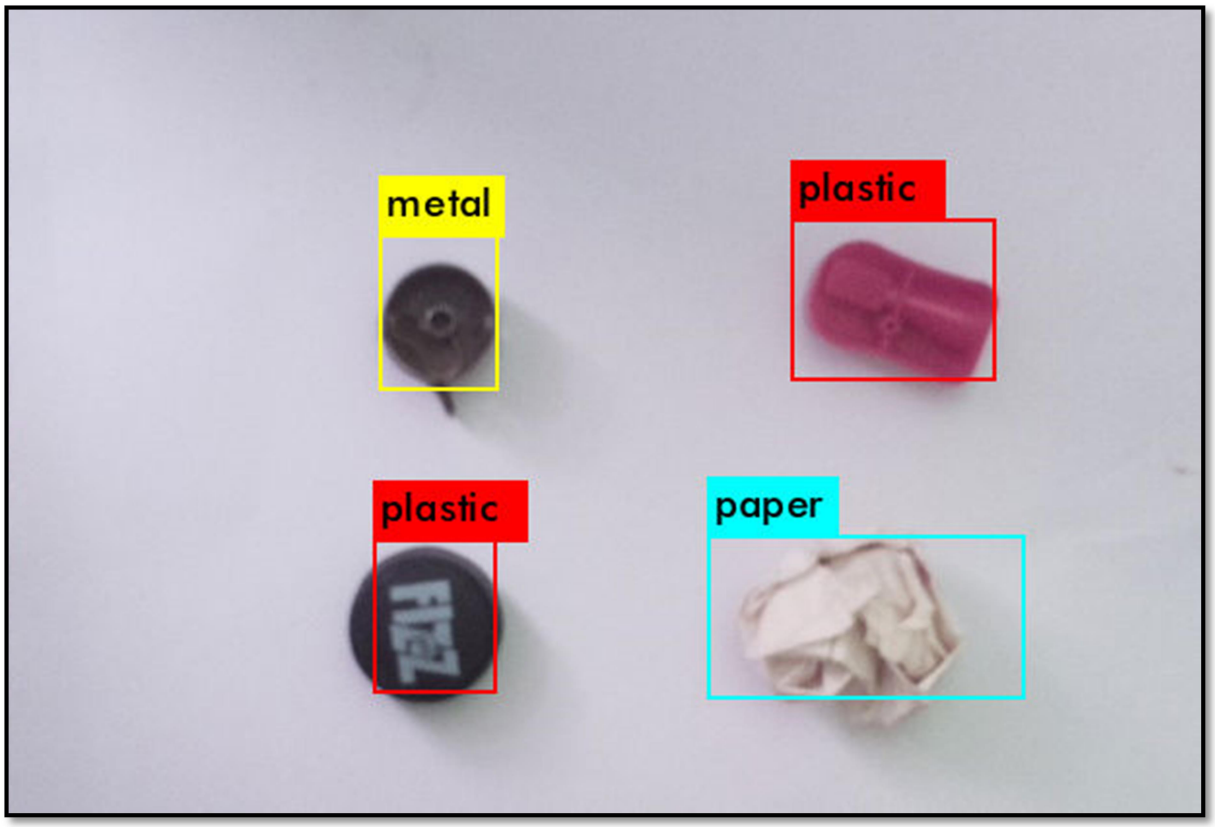

Figure 7.8: Image after multiple object detection

The complete image is divided into 12*8 grids. Each grid has its own function written related to the servo motors. The values left_x, top_y, width and height written in “result.txt” are values corresponding to the bounding box drawn over the objects and these are used to calculate the midpoint of the object. Using the midpoint of the object we check the grid the

object is present in. With this the function corresponding to the grid is called and the object is picked up by the robotic arm and thrown to a bin kept beside it. The inorganic waste objects which are beside these recyclable objects are left behind which has to picked out and put to another bin altogether.

Chapter 8

CONCLUSION AND SCOPE OF FUTURE WORK

8.1 Conclusion

Our proposed system effectively detects the garbage in the sightline of the camera, identifies the different categories of recyclable waste and segregates it using the robotic arm's reach. Using YOLO method to completely scan the workspace ensures a quick and reliable detection of the waste. There may be some latency in transference of knowledge between the camera, processing unit and the servo-motors, but this can be streamlined to be more efficient. Multiple types of waste are detected simultaneously and the co-ordinates transmitted allow the robotic-arm to queue the segregation of these objects.

The pre-trained tiny-yolo model allows the testing of the model on a small embedded processing unit such as the Raspberry Pi, however the much larger version of YOLOv3 with 74 layers instead of 25 would be beneficial if more memory space, RAM and GPU was made available. This model can be implemented realistically in various scales with a couple of real-world tweaks.

8.2 Future Scope

The project has tremendous potential for implementation in many facets and scales, provided the processing power is increased along with GPUs to validate the garbage detected. This can be facilitated with the latest Raspberry Pi model 4 as well as enabling high-speed internet, which would allow the use of cloud graphics processor units that can compute the results at least ten times faster rates.

Further improvement can be made to the accuracy of the model by the addition of a feature that would allow real time dataset creation. If the software could add more items to its catalogue every iteration that it detects an unknown object, it would learn with each new test and automatically improve its accuracy. This would also require high speed internet facilities to be able to retrain the model multiple times while also allowing for data augmentation.

Another improvement that can be done with this system is to implement it with the conveyor belt. Objects can be kept on a conveyor belt and transported to the system which detects and picks out the recyclable items and lets the decomposable objects to move through the belt to the decomposable pit directly. And the recyclable objects which are collected in a different bin can be further divided to separate categories and sent to its respective recycle stations.

With a more accurate model and larger scale hardware, this model may be implemented at both household/residential locations as well as garbage segregation centers for much larger quantities of waste. Automating this process would benefit the streamlining of disposal in urban as well as rural areas.

REFERENCES

[1] https://www.ijraset.com/fileserve.php?FID=14058

[2] https://www.ijraset.com/fileserve.php?FID=17556

[3] "Automated Waste Segregator". Amrutha Chandramohan, Joyal Mendonca, Nikhil Ravi Shankar, Nikhil U Baheti, Nitin Kumar Krishnan, M. S. Suma. 2014 Texas Instruments India Educators' Conference (TIIEC)

[4] "Classification of Trash for Recyclability Status". Gary Thung, Mingxiang Yang.

[5] "Automatic Waste Segregator". A. Sharanya, U. Harika, N. Sriya, Sreeja Kochuvila. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)

[6] "Waste segregation using Machine Learning". Yesha Desai, Asmita Dalvi, Pruthviraj Jadhav, Abhilash Baphna. 2018 International Journal for Research in Applied Science & Engineering Technology.

[7] "Smart Bin for Waste Management System". Sreejith S, Sanjay Kumar A, Ramya R, Roja R. 2019 5th International Conference on Advanced Computing & Communication Systems

[8] "Smart Garbage Segregation & Management System Using Internet of Things (Iot) & Machine Learning (ML)". Shamin N, P Mohamed Fathimal, Raghvendran R and Kamalesh Prakash. 2019 1st International Conference on Innovations in Information and Technology.

[9] "Electronically Assisted Automatic Waste Segregation". Nandhini S, Mrinal Sharma, Naveen Balachandran, K Suryanarayana, D S Harish Ram. 2019 Third International Conference on Trends in Electronics and Informatics

[10] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[11] https://www.raspberrypi.org/products/camera-module-v2/

[12] https://thinkrobotics.in/products/aluminum-4-dof-manipulator-steering-mechanical-

paws?_pos=3&_sid=7e06b0e75&_ss=r

[13] https://colab.research.google.com/notebooks/intro.ipynb

[14] https://pjreddie.com/darknet/

[15] https://pjreddie.com/darknet/yolo/

[16] https://opencv.org/

[17] http://abyz.me.uk/rpi/pigpio/python.html

[18] Eco-friendly, IOT-Based Waste Segregation and Management, B R Santhosh Kumar ; N Varalakshmi ; Soundarya S Lokeshwari ; K Rohit ; Manjunath ; D N Sahana, 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)

[19] "You Only Look Once: Unified, Real-time Object Detection", Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, 2016

[20] "YOLOv3: An Incremental Improvement", Joseph Redmon, Ali Farhadi, 2018

[21] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," CoRR, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524 1, 2

[22] R. B. Girshick, "Fast R-CNN," CoRR, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083 2

[23] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," CoRR, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506. 01497 2

[24] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv preprint, 2017. 3, 4

[25] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015

[26] https://www.geeksforgeeks.org/difference-between-arduino-and-raspberry-pi/

[27] https://www.elprocus.com/difference-between-arduino-and-raspberry-pi/

[28] https://techdifferences.com/difference-between-cpu-and-gpu.html

[29] https://www.geeksforgeeks.org/difference-between-cpu-and-gpu/

[30] https://www.datenreise.de/en/raspberry-pi-3b-and-3b-in-comparison/

APPENDIX