

Automatic waste classification using deep learning and  
computer vision techniques

by

MD. Akash

18101534

Umme Sabiha Shama

18301051

Dibash Dey

18301167

Ria Ghosh

20101626

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science And Engineering

Department of Computer Science and Engineering  
School of Data and Sciences  
Brac University  
23 March 2023

© 2023. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**



---

Md. Akash  
18101534



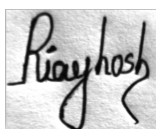
---

Umme Sabiha Shama  
18301051

Dibash Dey

---

Dibash Dey  
18301167



---

Ria Ghosh  
20101626

# Approval

The thesis/project titled “Automatic waste classification using deep learning and computer vision techniques” submitted by

1. Md. Akash (18101534)
2. Umme Sabiha Shama (18301051)
3. Dibash Dey (18301167)
4. Ria Ghosh (20101626)

Of Spring 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on April 3 , 2023.

## Examining Committee:

Supervisor:  
(Member)



---

Dewan Ziaul Karim  
Lecturer  
Department of Computer Science and Engineering  
BRAC University

Thesis Coordinator:  
(Member)

---

Md. Golam Rabiul Alam  
Professor  
Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)

---

Sadia Hamid Kazi  
Chairperson  
Department of Computer Science and Engineering  
Brac University

# Abstract

Waste management refers to a system that starts with classifying different kinds of waste and gradually managing it from its inception to its final disposal. Labeling waste in a proper manner can ensure the best outcome of recycling. The reason we think that our thesis topic will bring about a positive change in the waste management system is because we are emphasizing on making the environment pollution free and reuse the waste as much as we can by classifying and detecting the recyclable stuff from the waste that are considered useless. A custom CNN model has been implemented in our paper to classify things more accurately. Here, we have utilized a large dataset “garbage classification” [19] with a big number of images but to train our model, we have used 8 different classes: battery, biological, cardboard, clothes, green-glass, paper, plastic, trash which have been augmented in order to make all the classes equal in size which has resulted in a total of 16,000 images. Pre-trained CNN models such as VGGNet16, Resnet50, MobileNetV2, InceptionV3, EfficientNetB0 along with custom CNN models have been used and successfully achieved 87.57 percent, 94.34 percent, 96.99 percent, 95.71 percent, 35.92 percent, 97.16 percent train accuracy and 89.38 percent, 94.34 percent, 96.81 percent, 94.47 percent, 36.75 percent and 97.58 percent validation accuracy respectively. Later on, the paper also evaluates the custom CNN model’s performance on an unseen test dataset via confusion matrix. In this study, we have also proposed YOLOv4 and YOLOv4-tiny with Darknet-53 as a method for the detection of waste. Here we have used the same dataset which we have used in the custom CNN model. During the testing phase, every model makes use of three different types of inputs, including videos, webcams and images. The outcome demonstrates that YOLOv4 exceeds YOLOv4-tiny in terms of object detection, despite YOLOv4-tiny’s advantages in aspects of computational speed. The best YOLOv4 results are mAP 85.73 percent, precision 0.78, recall 0.84, F1-score 0.81, and Average IoU 62.05 percent. The best YOLOv4-tiny results are mAP 81.28 percent, precision 0.60, recall 0.87, F1-score 0.71, and Average IoU 45.67 percent.

**Keywords:** Custom CNN, Resnet50, VGG16, MobileNetV2, InceptionV3, EfficientNetB0, Pretrained, Validation, Accuracy, Detection, Classification, YOLOv4, Deep Learning, YOLOv4-tiny

## **Acknowledgement**

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our Advisor Mr. Dewan Ziaul Karim Sir for his kind support and advice in our work. He helped us whenever we needed help.

Finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Dedication</b>	<b>iv</b>
<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Problem Statement . . . . .	4
<b>2 Research</b>	<b>5</b>
2.1 Research Motivation . . . . .	5
2.2 Research Objectives . . . . .	5
<b>3 Related Work</b>	<b>6</b>
3.1 Literature Review . . . . .	6
<b>4 Approach</b>	<b>9</b>
4.1 Custom CNN Model . . . . .	9
4.1.1 Data Acquisition . . . . .	9
4.1.2 Image augmentation . . . . .	9
4.1.3 Proposed Model . . . . .	11
4.2 YOLOv4 and YOLOv4-Tiny Model . . . . .	12
4.2.1 Data Acquisition and Image Augmentation . . . . .	12
4.2.2 Image annotation for YOLOv4 and YOLOv4-tiny . . . . .	12
4.2.3 Final Dataset (YOLOv4 and YOLOv4-tiny) . . . . .	13
4.2.4 Proposed Model . . . . .	13
<b>5 Dataset and Model</b>	<b>14</b>
5.1 Custom CNN Model Architecture . . . . .	14
5.1.1 CNN Model Architecture . . . . .	14
5.1.2 Custom CNN Model Architecture . . . . .	15
5.1.3 ResNet50 . . . . .	17
5.1.4 VGG16 . . . . .	18
5.1.5 MobileNetV2 . . . . .	19

5.1.6	InceptionV3 . . . . .	20
5.1.7	EfficientNetB0 . . . . .	21
5.2	YOLOv4 and YOLOv4-Tiny Model	
	Architecture: . . . . .	23
5.2.1	YOLOv4 Architecture: . . . . .	23
5.2.2	YOLOv4-tiny Architecture: . . . . .	24
<b>6</b>	<b>Result and Analysis</b>	<b>25</b>
6.1	Custom CNN Model . . . . .	25
6.1.1	Pretrained Model . . . . .	30
6.2	YOLOv4 and YOLOv4-tiny Model . . . . .	36
6.2.1	YOLOv4-tiny Model . . . . .	36
6.2.2	YOLOv4 Model . . . . .	38
6.2.3	Comparison between YOLOv4 and YOLOv4-tiny . . . . .	40
<b>7</b>	<b>Conclusion</b>	<b>41</b>

# List of Figures

4.1	Sample data . . . . .	9
4.2	Distribution of different classes in training dataset . . . . .	10
4.3	Distribution of different classes in validation dataset . . . . .	10
4.4	Training and validation Dataset . . . . .	11
4.5	Approach of custom CNN model . . . . .	11
4.6	Sample data . . . . .	12
4.7	Example of LabelImg Images . . . . .	12
4.8	Approach of YOLOv4 and YOLOv4-tiny . . . . .	13
5.1	CNN model architecture . . . . .	15
5.2	Custom CNN model Visualization . . . . .	16
5.3	Custom CNN model Visualization . . . . .	17
5.4	ResNet50 model Summary . . . . .	18
5.5	VGG16 model Summary . . . . .	19
5.6	MobileNetV2 model Summary . . . . .	20
5.7	InceptionV3 model Summary . . . . .	21
5.8	EfficientNetB0 model Summary . . . . .	22
6.1	Last 5 Training and Validation Accuracy . . . . .	25
6.2	Training and Validation Accuracy . . . . .	26
6.3	Training and Validation Loss . . . . .	26
6.4	Confusion Matrix . . . . .	28
6.5	Test Prediction . . . . .	29
6.6	Test Prediction . . . . .	29
6.7	ResNet50 Training and Validation Accuracy . . . . .	30
6.8	ResNet50 Training and Validation Loss . . . . .	30
6.9	VGG16 Training and Validation Accuracy . . . . .	31
6.10	VGG16 Training and Validation Loss . . . . .	31
6.11	InceptionV3 Training and Validation Accuracy . . . . .	32
6.12	InceptionV3 Training and Validation Loss . . . . .	32
6.13	MobileNetV2 Training and Validation Accuracy . . . . .	33
6.14	MobileNetV2 Training and Validation Loss . . . . .	33
6.15	EfficientNetB0 Training and Validation Accuracy . . . . .	34
6.16	EfficientNetB0 Training and Validation Loss . . . . .	34
6.17	Model Training Accuracy . . . . .	35
6.18	Model Validation Accuracy . . . . .	36
6.19	Average loss and mAP . . . . .	36
6.20	Testing image detection . . . . .	37
6.21	Testing image detection . . . . .	37



6.22	Testing image detection . . . . .	37
6.23	Testing image detection . . . . .	38
6.24	Average loss and mAP . . . . .	38
6.25	Testing image detection . . . . .	39
6.26	Webcam image detection . . . . .	39
6.27	Testing image detection . . . . .	39
6.28	Testing image detection . . . . .	40
6.29	mAP results of YOLOv4 and YOLOv4-tiny . . . . .	40

# Chapter 1

## Introduction

Every year, the world generates almost 2.1 billion metric tons of municipal solid garbage, and at least 33 percent of it is not treated safely [18]. The average amount of garbage produced by a person per day around the world is enormous. However, the entire process of recycling has a significant hidden cost, which is created by the categorization and the manufacturing from the recycled waste. While humans in several nations are willing to sort their own garbage at home these days, they may be unclear about how to identify the appropriate category of waste at the time of discarding items. Finding an automatic method of recycling, which not only has positive impacts on the environment but also has beneficial effects on the economy. This has the dual effect of improving both. Garbage piling is one of the most well-known threats to human health because it encourages the spread of illness by carriers like flies, mosquitoes, and other insects. Soil and water may be contaminated owing to harmful substances present in inadequately handled materials, and this is in addition to the destruction of the magnificence of biodiversity, deforestation, and territory occupancy to give enough room for waste. Conversely, pollution may disrupt food webs, which in turn increases the prevalence of illnesses and health problems in both human populations and the planet's natural habitats. The process of dumping solid garbage which comprises paper, plastic, metal, glass, etc., is becoming a concern as the variety of industries in metropolitan areas grows. Burning garbage is another typical waste disposal strategic tool, although it results in air pollution and the release of carcinogenic chemicals from the dump itself.

CNN focuses on the significant advancements made in image classification. They are frequently combined with picture categorization and are utilized frequently for the purpose of evaluating visual images. Garbage can be sorted and separated by hand, but the efficiency of this method is lower. In order to overcome this challenge, we classify waste using an algorithm that utilizes deep learning in order to achieve results that are both superior and more effective. Garbage is identified by means of an algorithm that a computer has been instructed to use for the purpose.

YOLO's ability to perform inferences quickly, which enables it to interpret pictures in real time, is one of the system's primary strengths. It functions very well in applications like video surveillances, autonomous vehicles, and augmented reality, amongst others. The main distinction between YOLOv4 and YOLOv4 tiny is the significantly smaller network size. A decrease in the amount of convolutional layers is made in the CSP's main body. With only two YOLO layers rather than three and fewer anchor boxes, prediction accuracy decreases. For easier implementation

on mobile and embedded devices, we offer YOLOv4-tiny as a lightweight variant of YOLOv4 to simplify the network structure and minimize requirements. The real-time performance of object detection is enhanced by a suggested rapid object detection approach using YOLOv4-tiny. Based on YOLOv4, YOLOv4-tiny simplifies network layout and reduces parameters for mobile as well as embedded device development. YOLOv4-tiny's simpler architecture has an impact on the algorithm's performance in terms of both prediction time and prediction probability, despite being simpler than YOLOv4's architecture. This is one area where YOLOv4-tiny performs better. In comparison to YOLOv4, which has three YOLO heads, this version only has two. Unlike YOLOv4, which was trained using 137 pre-trained convolutional layers, it was only trained using 29. But in terms of accuracy and other areas YOLOv4 performs better than YOLOv4-tiny

## 1.1 Problem Statement

In the past, people used to get rid of their waste by digging a hole in rural areas. Due to the relatively low population density, this method of garbage disposal was expected to be effective. With fewer people around, there was less garbage to sort through. However, as the population has risen, so has the amount of garbage, making disposal a challenge. With population growth and industrialization happening all over the world, it is important to keep cities safe and clean. Putting waste in illegal places affects the environment if we don't dispose of these garbage in time, then it can lead to a lot of serious health problems and pollution [16]. It is completely obvious that recycling is important for a number of reasons, including the economy and the surroundings. By classifying the waste in categories and by detecting we can make sure that both biodegradable and non biodegradable garbage are well managed . In this way we can deduct the amount of waste that is being produced everyday and make the disposal easier. Studies being done right now on how to automatically find waste are hard to compare because there aren't any benchmarks or widely accepted standards for the metrics and data being used. [11] Which is why, Deep learning has quickly become one of the most interesting and cutting-edge subfields within the field of computer science research over the past decade. The public in general now has access to systems that are both more advanced and intelligent. [16] Nowadays, CNN is used quite frequently for the process of image classification. The accuracy with which it can categorize images is significantly improved. Convolutional neural networks (CNN), do not need to rely on the manual extraction of features. This is a significant advantage of it. In the field of image detection, YOLOv4 is now among the most used algorithms. It is a member of the YOLO (You Only Look Once) group of object identification algorithms, and it is well-known for both its accuracy and its quickness. In this paper we propose a waste classification system which uses CNN as a classifier as well as waste detection systems which use YOLOv4 and YOLOv4-tiny as detectors.

# Chapter 2

## Research

### 2.1 Research Motivation

The main motivation of our research work is to keep our nature free from the bad effects of pollution caused by the garbage. Waste management will preserve nature, environmental assets, and living beings. Waste can be classified and detected in order to better manage the associated potential risks to both the environment and human health. In addition to this, it can assist us in fulfilling our responsibilities in accordance with the overall environmental duty. Proper waste management begins with accurate waste classification and detection. The primary goals of waste management are to decrease the amount of waste produced, reduce the effects of pollution and protect groundwater sources. These goals are all interrelated. It is simpler to deal with wastes and dispose of them in an appropriate manner if we categorize them into groups and detect them according to the risks they pose to both the environment and to human health

### 2.2 Research Objectives

In our research, we have used the Convolutional Neural Network (CNN) to classify waste as well as YOLOv4 and YOLOv4-tiny to detect waste from images which has been separated in 8 classes. Our main research objectives are

- Understand image processing and how it works.
- To understand data pre-processing techniques like augmentation and labeling.
- To understand the impact of deep learning in our model.
- To classify waste of different categories more accurately.
- To achieve better optimal speed and accuracy of waste detection.

# Chapter 3

## Related Work

### 3.1 Literature Review

Olugboja Adedeji et al. on their paper [1] they developed a smart method of sorting garbage by using 50 layer residual net pre-train (ResNet-50). After testing the method on the garbage image dataset, it was found to be 87 percent accurate. On the other paper Sai Sushanth G et al. their main purpose is to sort garbage using AI. Initially AlexNet, VGG16, ResNet50, DenseNet 169 models were used. They made use of the pre-trained architectures and six waste classification categories to accomplish that goal. They performed this by employing the six waste classification categories and the pre-trained structures already available to them. The accuracy for DenseNet169 was 94.9 percent and for ResNet50 it was 93.4 percent.[2]

In another paper the authors Dipesh Gyawali et al. proposed an algorithm based on ResNet-18. The accuracy of the original ResNet-18 is 87.8 percent. They evaluated the performance of several popular Deep Learning Network architectures for garbage classification. For this purpose, they employed a convolutional neural network. The hardware in the shape of a waste can be utilized to separate many types of garbage.[3] For another article, the authors Sehrish Munawar Cheema et al. proposed (SWMACM-CA) system. The primary goal of this work is to show that their proposed system is superior to the current superior to previous methods in terms of precision. The trained algorithm has an efficiency of over 90 percent, making it highly powerful.[4]

Stephen L et al. in their paper, they used a dataset of 2527 images. Jpg-formatted garbage photos with 6 categories were utilized for the training. Additionally, they quantized and refined their baseline model, which test results that were 87.2 percent accurate. In S6 Edge+ smartphone, model application was well installed.[5]

Dongwei Guo et al. offered a refined version of the YOLOV4 detector to identify trash. In order to improve edge detection in deep networks, the emphasis component is incorporated into the algorithm and Context-Based Accurate Modeling (CBAM) is introduced to the extracting features network. TrashSet, a dataset of 45,910 photos from 47 types of urban waste collected, has also been generated. Results from tests on the TrashSet indicate that our detector performs admirably, and mAP achieves 97.15 percent. They explore the improved YOLOV4 object detector's usefulness in detecting waste in this work [6]. Also Andhy Panca Saputra et al. [7] proposed YOLOv4 and YOLOv4-tiny are used in combination with Darknet-53 to detect objects using a deep learning algorithm. There are 4 classes of the dataset's

3870 waste images. The results demonstrate that, when it comes to object identification, YOLOv4 outperforms its smaller version, YOLOv4-tiny. even if YOLOv4-tiny performs better in terms of computing speed. Top YOLOv4 model findings include a mAP of 89.59 percent, accuracy of 0.76, recall of 0.90 percent, F1-score of 0.82, and an Average IoU of 64.01 percent; top YOLOv4-tiny model results include a mAP of 81.4 percent, precision of 0.59, recall of 0.83, F1-score of 0.69, and an Average IoU of 48.35 percent. As well as the author Yongchuang Yangin his research proposes a deep learning-based classification solution for restaurant recycling. It is constructed to use Yolov4, SSD, and fast RCNN as an experimental platform. The author of this paper applies several novel techniques. Convolution neural network with many targets Multiple targets can be found using YOLO v4 in a single image. Yolov4 has the maximum detection effect, according to their experimental findings, with a detection accuracy of 77.78 percent and a detection speed of 39.3 FPS.[8]

The writers of this publication developed the Skip-YOLO system for real-world garbage recognition using an eye for feature mapping in multiple neural networks. Combining the multi-scale, high-dimensional feature mappings and then delivering them to the YOLO layer allows for accurate prediction of trash kind and location. In comparison to the YOLOv3, their testing shows a 22.5 percentage point increase in overall detection accuracy and an 18.6 percentage point increase in average recall rate.[9] In contrast to the ideas presented here, those of Saurav Kumar et al. in order to train a custom dataset, the framework Darknet neural network has taken use of the YOLOv3 method. To test the algorithm's performance, YOLOv3-tiny was used to conduct the detection job. According to the results of the experiments, the suggested YOLOv3 approach achieves around 94 percent by YOLOv3, but YOLOv3-tiny only achieves 45.96 percent. After that, YOLOv3's mAP value stabilized at 94.99 percent (best value).[10]

This study by Meena Malik et al. presents an architecture for sorting waste into the categories required by various benchmark methods. The categorization process was carried out using the EfficientNet-B0 architecture. This study suggested fine-tuning with the EfficientNet-B0 model for effective categorization of photos based on geographic location. This model work through transfer learning and creates a region-optimized classification model. It was shown that the accuracy of this model was on par with that of EfficientNet-B3, although using a far fewer amount of parameters.[11]

As per Andhy Panca Saputra et al. [12] In their study, they propose a solution based on a deep learning algorithm, namely YOLOv4 and YOLOv4-tiny, in conjunction with the anonymized database Darknet-53, for the problem of object detection. There are a total of 3870 photos in the collection, and they include a wide variety of waste .The best mAP 89.59 percent , accuracy 0.76, recall 0.90, F1-score 0.82, and Average IoU 64.01 percent are achieved by the YOLOv4 model, whereas the best mAP 81.84 percent, precision 0.59 percent, recall 83 percent, F1-score 69 percent, and Average IoU 48.35 percent are achieved by the YOLOv4-tiny model. This study also demonstrates the superior performance of mosaic-based models with reduced subdivision values. ZICONG JIANG1 et al propose the YOLO v4-tiny model. According to the authors Simulation findings demonstrate that the suggested technique detects objects quicker than YOLOv4-tiny and YOLOv3-tiny, and with almost the same precision as YOLOv4-tiny. It is better ideal for real-time object identification, particularly for embedded device development. [13]

Aghilan M et al. in their article they used the CNN model and The goal of this research was to develop a low-cost, user-friendly waste separation method for urban homes. This technique is useful when dealing with a lot of data or a lot of parameters since it uses less memory and runs quickly. To conclude, the final accuracy for the CNN model was 79 percent. [14]

According to Victoria Ruiz et al. In order to train and evaluate several deep learning algorithms for autonomously sorting waste, TrashNet dataset is utilized by them. In particular, VGG, Inception, and ResNet, which are all types of Convolutional Neural Networks (CNN), were compared. The best classification results came from a model that combined Inception and ResNet and got an accuracy of 88.6 percent.[15]

According to Dip Patel et al. In this new technique, an Android app called Spot-Garbage is introduced to identify and pinpoint trash in a geo-tagged photograph that the user has clicked on. Inception-ResNet, a hybrid network design, produced the greatest results, with an impressive 88.6 percent accuracy. Wang et al. used Faster-RCNN and ResNet algorithms to identify real-time trash in urban photos.[16]



# Chapter 4

## Approach

### 4.1 Custom CNN Model

#### 4.1.1 Data Acquisition

To train our model, we used "garbage classification" [19] dataset, which contains a huge number of images across 8 categories: battery, biological, cardboard, clothes, green-glass, paper, plastic, trash that were collected through Kaggle. These photos contain different types of biodegradable and non-biodegradable waste. From this dataset, we have separated 400 images for our test dataset.



Figure 4.1: Sample data

#### 4.1.2 Image augmentation

In order to expand our dataset, we performed image augmentation for 8 classes. Rotation, scaling, flipping, transpose, grid-distortion were also done with image augmentation parameters. The reformation was done to balance the number of images (2000 per category) in every class to get better accuracy. This augmented dataset is splitted into two categories: training data and validation data which have a ratio of 8:2.

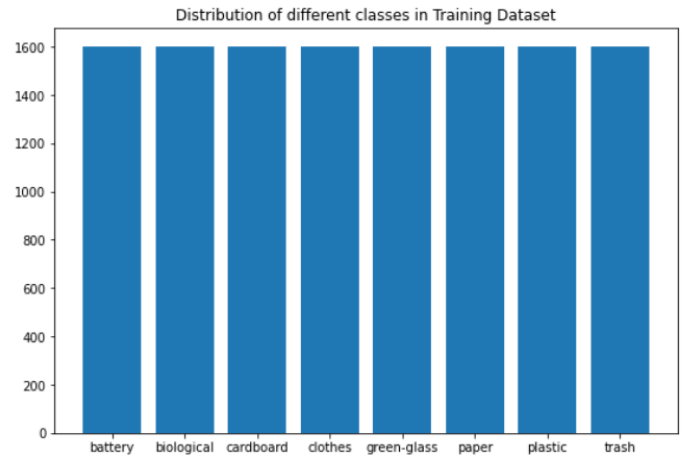


Figure 4.2: Distribution of different classes in training dataset

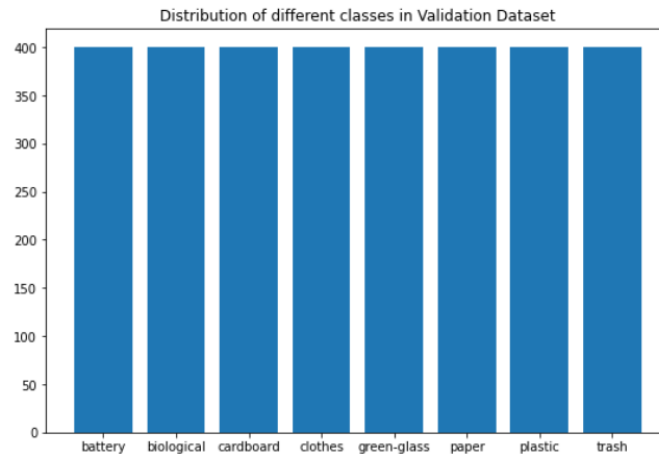


Figure 4.3: Distribution of different classes in validation dataset

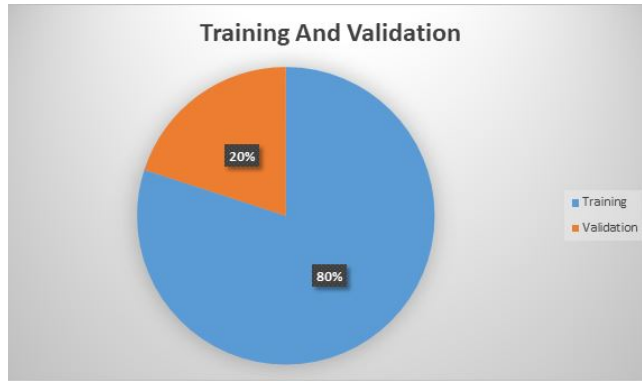


Figure 4.4: Training and validation Dataset

### 4.1.3 Proposed Model

In our proposed system, we have built a custom Convolutional Neural Network (CNN) consisting of several layers that can identify and classify images of 8 classes (battery, biological, cardboard, clothes, green-glass, plastic, paper, trash). Our dataset was collected from "Garbage classification" [19]. A dataset of 16,000 augmented images is provided for our proposed system. After splitting the dataset for training and validation, we had 12,800 images as our training data and 3,200 images as our validation data.

For our custom Convolutional Neural Network (CNN) model, we have built sequential models with many layers.

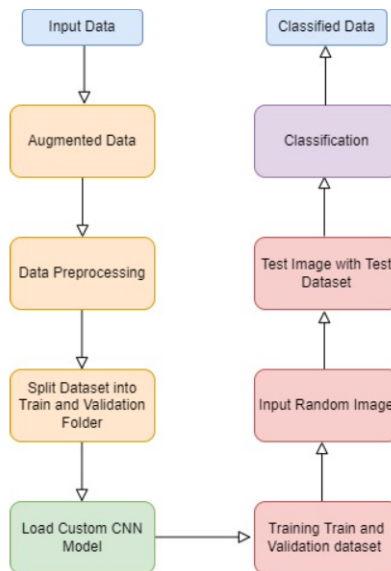


Figure 4.5: Approach of custom CNN model

## 4.2 YOLOv4 and YOLOv4-Tiny Model

### 4.2.1 Data Acquisition and Image Augmentation

To train our YOLOv4 and YOLOv4-Tiny model, we have used same dataset which we have augmented for our Custom CNN model. For which, we have not done any further augmentation for these models.



Figure 4.6: Sample data

### 4.2.2 Image annotation for YOLOv4 and YOLOv4-tiny

Image annotation refers to the process of assigning labels to parts of an image. Labels are used as data inputs towards a computer vision model, providing information about the pictures being used as training data. To label both training and testing images. We have utilized the Labelling tool to create bounding boxes and produce coordinates for every train and test images.

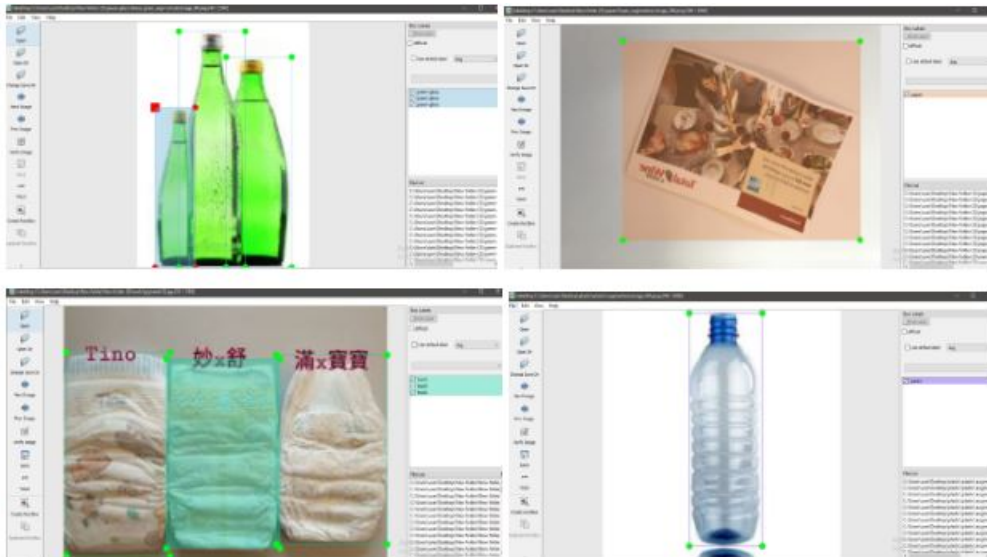


Figure 4.7: Example of Labelling Images

### 4.2.3 Final Dataset (YOLOv4 and YOLOv4-tiny)

For YOLOv4 and YOLOv4-tiny model, we have generated two more files. These new files are named "obj.names" and "obj.data." The file referred to as "obj.names" is where you can find the class names for our model and in obj.data we have added the class number, training path, validation path and as well as a backup path where our training weight will be saved. Because of the nature of our circumstances, we have classified everything into one of these eight categories: battery, biology, cardboard, clothing, green-glass, paper, plastic, and garbage. and we have created yolov4-obj.cfg and yolov4-tiny-custom.cfg . In these .cfg files we have changed the filter size, iteration size based on our class number. we set the weight and height 416. We have also used generate-train.py and generate-test.py to divide our dataset into 80 percent and 20 percent to create our train and test dataset

### 4.2.4 Proposed Model

Our suggested system involves developing YOLOv4 and YOLOv4-tiny Model to identify various object classes in still and moving visual media. For our proposed model, we have supplied a dataset containing 16,000 augmented images with 16,000.txt files. Once the dataset was divided into train and test sets, we were left with 12,800 images for training and 3,200 for testing.

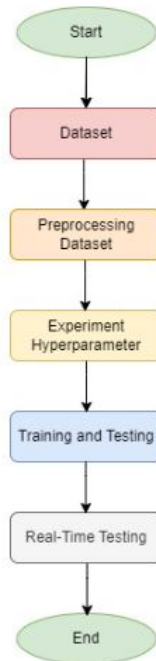


Figure 4.8: Approach of YOLOv4 and YOLOv4-tiny

# Chapter 5

## Dataset and Model

### 5.1 Custom CNN Model Architecture

#### 5.1.1 CNN Model Architecture

Specifically we are working with a custom CNN model. As known, modern deep learning applications want model inference to occur at the edge devices. This is done for a number of reasons, including reducing latency, making it easier for the network to connect to the cloud, and keeping user privacy safe. CNN is among the most applied model families.

In convolution, a process called "feature extraction" is utilized to separate and identify the unique characteristics of the picture to be analyzed. The feature extraction is made up of many layers, many of which are pairs of convolutional layers also known as pooling layers. One that uses the output of the convolutional method to determine the image's classification based on the characteristics that were previously retrieved from the data. This method for extracting CNN features from datasets aims to do it with as few characteristics as possible. It creates new features by combining the characteristics like an existing collection of features into one. In the CNN architecture diagram, many layers can be seen.

The architecture of a CNN includes components such as convolution layers, fully linked layers and pooling layers. Typically, an architecture will include a stack of convolution layers followed by a pooling layer, and then one or more fully linked layers. Forward propagation describes the transformation of incoming data into outgoing data at these levels.

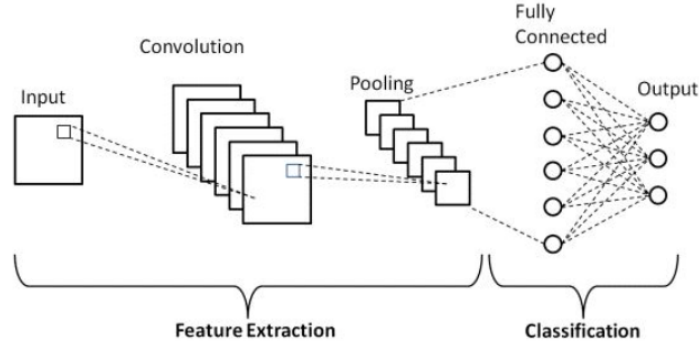


Figure 5.1: CNN model architecture

**Convolutional Layer:** We used multiple layers of 2D convolutional layers for our proposed system. For the first layer, we gave an input size of 150x150 so that it would match the size of the images we were using. For this layer, we set the kernel to 32 and used "relu" activation.

$$W_{out} = [(W - F + 2P)/S + 1] \quad (5.1)$$

This is the Formula for output size in Convolution layer

**Max pooling layer:** The Max - pooling is often used as a transitional layer between the Convolutional Layer and the Fully Connected Layer. Our model utilized max pooling. The primary objective was to reduce the dimensions of the convolution layer to cut down on computational costs.

**Fully Connected Layer:** All of the neurons, weights, and biases, are all part of the Fully Connected layer. Originally, it connected the two layers. It mainly serves as a way to help the classification process work.

### 5.1.2 Custom CNN Model Architecture

**Dataset Preprocessing:** The data must be preprocessed before it can be used in the custom Convolutional Neural Network (CNN) algorithm. This is a necessary stage in the data preparation process. This procedure is used, in its most basic form, to eliminate the variables that do not contribute in any potential to increase the CNN model's performance or accuracy. In contrast, this procedure provides us the ability to conduct any and all necessary modifications on the raw data, which improves CNN model's performance and accuracy. Also, we've customized some features about our training dataset. This change includes the images size, batch size, resizing, rotating, zooming, scaling and flipping them horizontally. All of these have the following values:

$$imgsize = 150 * 150 \tag{5.2}$$

$$batchsize = 256 \tag{5.3}$$

$$rescale = 1/255 \tag{5.4}$$

$$rotation = 30 \tag{5.5}$$

$$zoomrange = .4 \tag{5.6}$$

$$horizontalflip = True \tag{5.7}$$

Our custom CNN model was constructed with ten Conv 2D layers. To make computations run more quickly and efficiently, pooling is utilized to aggregate similar features into larger ones. In response, the amount of trainable parameters that can be retrieved. Then a single layer of flatten layer was applied. Overall, this is good for the system as a whole. We also used a dense layer. Every neuron in this layer receives the information transmitted from the preceding layers. Then in our model we use an initial learning rate of 0.00001, This affects the rate at which the model adjusts to the data. For compiling the model we use Adam optimizer.

### Model Summary:

```

model.summary()
max_pooling2d_1 (MaxPooling (None, 37, 37, 32) 0
2D)
conv2d_3 (Conv2D) (None, 37, 37, 64) 18496
batch_normalization_1 (Batc (None, 37, 37, 64) 256 conv2d_8 (Conv2D) (None, 4, 4, 1024) 2360320
hNormalization)
conv2d_4 (Conv2D) (None, 37, 37, 128) 73856 max_pooling2d_5 (MaxPooling (None, 2, 2, 1024) 0
2D)
batch_normalization_2 (Batc (None, 37, 37, 128) 512 conv2d_9 (Conv2D) (None, 2, 2, 512) 4719104
hNormalization)
max_pooling2d_2 (MaxPooling (None, 18, 18, 128) 0 max_pooling2d_6 (MaxPooling (None, 1, 1, 512) 0
2D)
conv2d_5 (Conv2D) (None, 18, 18, 256) 295168 flatten (Flatten) (None, 512) 0
batch_normalization_3 (Batc (None, 18, 18, 256) 1024 dense (Dense) (None, 64) 32832
hNormalization)
max_pooling2d_3 (MaxPooling (None, 9, 9, 256) 0 batch_normalization_5 (Batc (None, 64) 256
2D)
conv2d_6 (Conv2D) (None, 9, 9, 512) 1180160 dense_1 (Dense) (None, 8) 520
conv2d_7 (Conv2D) (None, 9, 9, 256) 1179904
batch_normalization_4 (Batc (None, 9, 9, 256) 1024
hNormalization)
max_pooling2d_4 (MaxPooling (None, 4, 4, 256) 0
2D)
conv2d_8 (Conv2D) (None, 4, 4, 1024) 2360320
-----
Total params: 9,901,416
Trainable params: 9,899,816
Non-trainable params: 1,600

```

Figure 5.2: Custom CNN model Visualization



## Visualization:

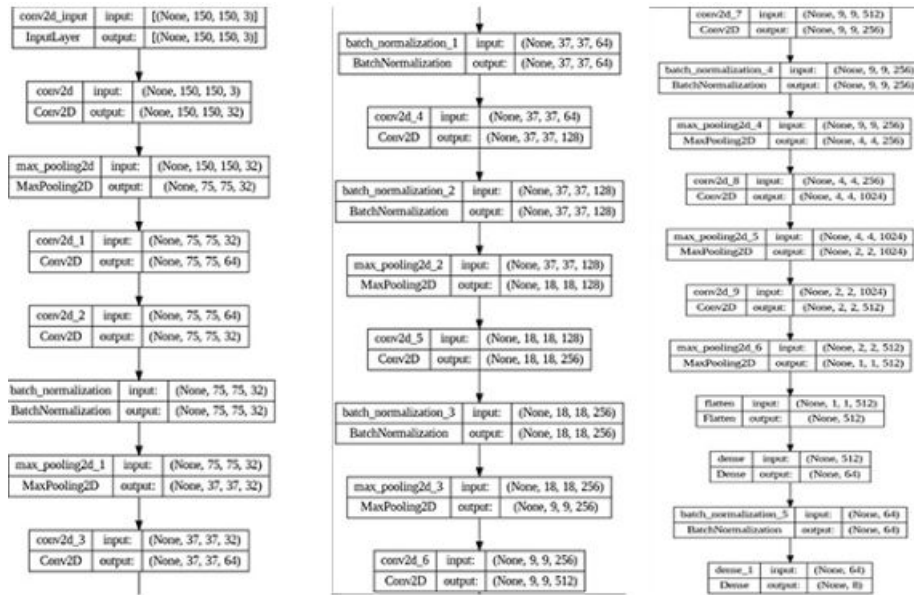


Figure 5.3: Custom CNN model Visualization

Besides the custom CNN model, our dataset also has been trained with five others pre-trained models, which are ResNet50, VGG16, MobileNetV2, InceptionV3, EfficientNetB0

### 5.1.3 ResNet50

Resnet50 is a deep convolutional neural network (CNN) that includes a total of 50 layers, 48 of which are convolutional layers, as well as one MaxPooling layer and one Average Pooling layer. Residual Neural Network, often known as ResNet, is an alternative for Artificial Neural Network (ANN), a form of neural network where a network is built by stacking blocks of leftover data. Using this approach, a network that has already been trained on more than twenty-four thousand pictures captured from a “Garbage Classification” dataset [19]. As the input, a 150x150x3 image is used, and then a MaxPooling layer containing a 3x3 filter is added on top of that. Ultra-deep neural networks are trained by using ResNet50 , which means it may include hundreds or even thousands of layers and yet perform exceptionally well. In a word, Resnet is the most well-known neural network that is used to solve a number of issues related to computer vision.

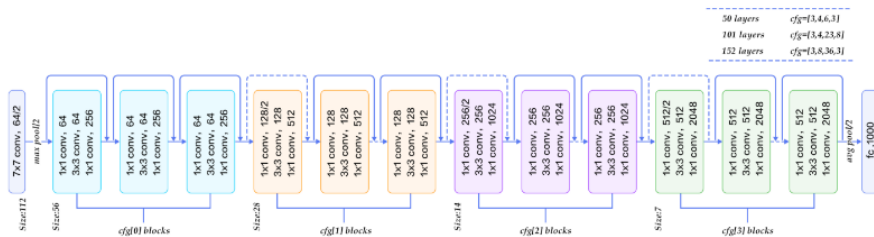


Figure 5.3: ResNet50 architecture

## ResNet50 Model Summary

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50v2 (Functional)	(None, 5, 5, 2048)	23564800
flatten (Flatten)	(None, 51200)	0
dense (Dense)	(None, 512)	26214912
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 8)	4104

-----  
Total params: 49,783,816  
Trainable params: 26,219,016  
Non-trainable params: 23,564,800  
-----

Figure 5.4: ResNet50 model Summary

### 5.1.4 VGG16

VGG16 which is a form of CNN is widely regarded as one of the most effective computer vision models ever created. The model's creators evaluated the networks and increased the depth using an architecture with extremely tiny convolution filters (3x3), that showed a significant improvement above the state-of-the-art at the time. They boosted the number of weight layers from 16 to 19, which raised the number of tunable features to about 138. It is a well-known technique for classifying images and may be easily implemented using transfer learning. The number "16" indicates that there are 16 weighted layers in VGG16. With 5 Max Pooling, 3 Dense layers and 13 convolutional. VGG16 has a total of 21, but just Sixteen weight layers. The convolution and max pool layers in the design all have the same structure.

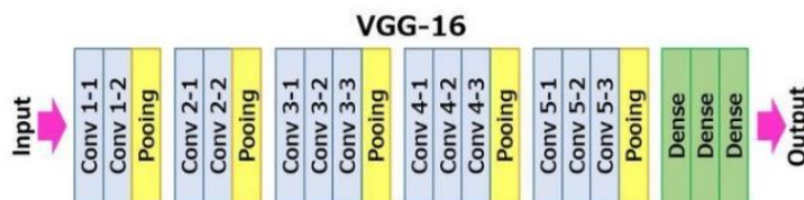


Figure 5.3: VGG16 architecture

## VGG16 Model Summary

```
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590880
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590880
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 512)	4194816
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 8)	4104

=====  
Total params: 18,913,608  
Trainable params: 4,198,928  
Non-trainable params: 14,714,688  
=====

Figure 5.5: VGG16 model Summary

### 5.1.5 MobileNetV2

The convolutional neural network MobileNetV2 is designed to function effectively on mobile platforms. Its foundation is an inverted latent architecture with residual connections between the levels of the bottleneck. In order to filter features, the additional expansion layer uses a non-linear component in the form of compact fully convolutional completely connected layers.. A total of 32 filters make up the first fully convolutional layer in MobileNetV2's architecture, which is followed by 19 remaining bottleneck layers.

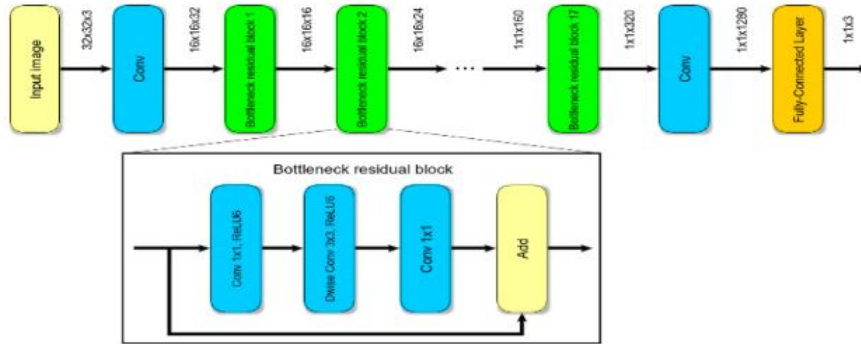


Figure 5.3: MobileNetV2 architecture

### MobileNetV2 Model Summary

```

▶ model.summary()
└─ Model: "sequential_2"
├── Layer (type)                Output Shape                Param #
├── ────────────                ────────────                ───
├── mobilenetv2_1.00_224 (Func  (None, 5, 5, 1280)        2257984
├── ional)
├── average_pooling2d_2 (Averag  (None, 2, 2, 1280)        0
├── ePooling2D)
├── flatten_2 (Flatten)          (None, 5120)               0
├── dense_6 (Dense)              (None, 512)                2621952
├── dropout_4 (Dropout)         (None, 512)                0
├── dense_7 (Dense)              (None, 50)                 25650
├── dropout_5 (Dropout)         (None, 50)                 0
├── dense_8 (Dense)              (None, 8)                  408
├── ────────────                ────────────                ───
├── Total params: 4,905,994
├── Trainable params: 3,060,810
├── Non-trainable params: 1,845,184

```

Figure 5.6: MobileNetV2 model Summary

### 5.1.6 InceptionV3

More research has shown that Inception Networks are more computationally efficient than their predecessors, both in terms of the amount of parameters produced by the network and the cost to the user. It is important to maintain the computational benefits of an Inception Network if at all possible. Label Smoothing, Factorized 7 x 7 convolutions, as well as an auxiliary classifier are just a few examples of how the Inception-v3 design of convolutional neural networks improves upon previous versions.

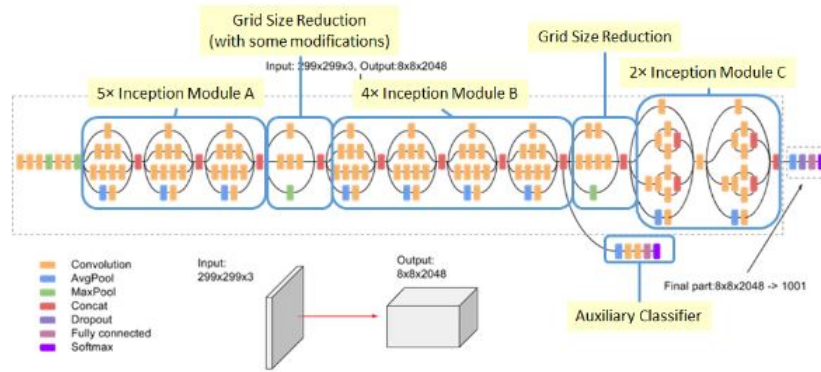


Figure 5.3: InceptionV3 architecture

## InceptionV3 Model Summary

```

model.summary()
C-
-----
batch_normalization_207 (Batch Normalization) [conv2d_101[4]]
batch_normalization_208 (Batch Normalization) [conv2d_102[4]]
conv2d_107 (Conv2D) (None, 1, 1, 242) 191228 [average_pooling2d_38[4]]
batch_normalization_209 (Batch Normalization) [conv2d_108[4]]
activation_209 (Activation) (None, 1, 1, 242) 0 [batch_normalization_209[4]]
activation_208 (Activation) (None, 1, 1, 242) 0 [batch_normalization_208[4]]
activation_407 (Activation) (None, 1, 1, 242) 0 [batch_normalization_207[4]]
activation_408 (Activation) (None, 1, 1, 242) 0 [batch_normalization_208[4]]
batch_normalization_209 (Batch Normalization) (None, 1, 1, 252) 170 [conv2d_107[4]]
activation_409 (Activation) (None, 1, 1, 252) 0 [batch_normalization_209[4]]
conv2d_110 (Conv2D) (None, 1, 1, 768) 0 [activation_407[4]-
activation_209[4]]
conv2d_111 (Conv2D) (None, 1, 1, 768) 0 [activation_408[4]-
activation_209[4]]
activation_209 (Activation) (None, 1, 1, 242) 0 [batch_normalization_209[4]]
conv2d_112 (Conv2D) (None, 1, 1, 2048) 0 [activation_409[4]-
conv2d_110[4]-
conv2d_111[4]-
activation_407[4]-
activation_209[4]]

global_average_pooling2d_3 (GlobalAveragePooling2D) (None, 2048) 0 [conv2d_112[4]]
batch_normalization_209 (Batch Normalization) (None, 2048) 8202 [global_average_pooling2d_3[4]]
dropout_11 (Dropout) (None, 2048) 0 [batch_normalization_209[4]]
dense_11 (Dense) (None, 2048) 268478 [dropout_11[4]]
dropout_12 (Dropout) (None, 2048) 0 [dense_11[4]]
dense_12 (Dense) (None, 1) 8208 [dropout_12[4]]

Total params: 23,527,352
Trainable params: 22,528,472
Non-trainable params: 998,880

```

Figure 5.7: InceptionV3 model Summary

### 5.1.7 EfficientNetB0

According to studies, EfficientNet is among the most effective models since it can get the best results in imagenet and transfer learning for image classification. The various models available in EfficientNet range from B0 to B7. The foundation of the EfficientNet framework is the EfficientNet-B0 version. The whole structure of EfficientNet-B0, with its 237 layers with 11 M trainable parameters, is shown in Figure. This model uses a mobile inverted bottleneck convolution layer and a 3\*3 receptive fields to ensure that features are preserved throughout all layers.

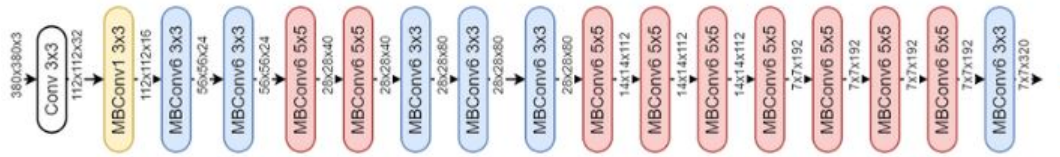


Figure 5.3: EfficientNetB0 architecture

## EfficientNetB0 Model Summary

```

Model: "sequential_2"
-----
Layer (type)                Output Shape              Param #
-----
efficientnetb0 (Functional)  (None, 5, 5, 1280)      4049571

global_average_pooling2d_2  (None, 1280)             0
(GlobalAveragePooling2D)

batch_normalization_4 (Batc (None, 1280)             5120
hNormalization)

dense_2 (Dense)              (None, 8)                10248

batch_normalization_5 (Batc (None, 8)                32
hNormalization)

activation_2 (Activation)    (None, 8)                0
-----
Total params: 4,064,971
Trainable params: 12,824
Non-trainable params: 4,052,147

```

Figure 5.8: EfficientNetB0 model Summary

## 5.2 YOLOv4 and YOLOv4-Tiny Model Architecture:

### 5.2.1 YOLOv4 Architecture:

The "You only look once v4" (YOLOv4) algorithm is a deep learning technique used for object detection. It is well-known for its rapid response time and pinpoint precision while recognizing things in real time. YOLOv4 incorporates a number of cutting-edge methods into its design, which is built on a fully convolutional neural network. CSPDarknet53, a more advanced and complicated variant of Darknet53, serves as the foundation for YOLOv4's core network. By combining convolutional layers with Cross-Stage Partial Networks (CSP), CSPDarknet53 is able to decrease the network's reliance on parameters while simultaneously increasing its performance. Convolutional layers in YOLOv4 diminish the spatial resolution of the feature maps before the network's output layers make their final detections. Objectness scores Predicting bounding boxes, and class probabilities for such a portion of grid cells falls under the purview of each layer of output. Anchor boxes are used in YOLOv4 to make more precise bounding box predictions. The network can learn to optimize the size and shape of anchor boxes, which are predetermined bounding boxes, to better match the elements in the image. It can identify objects of varying sizes and shapes because of its usage of three anchor boxes for each output layer. Here in our YOLOv4 model we have customized

$$BatchSize = 64 \quad (5.8)$$

$$NetworkSize = 416 \times 416 \quad (5.9)$$

$$MaxBatches16000, Steps12800 \text{ and } 14400 \quad (5.10)$$

$$Filters = 39 \quad (5.11)$$

$$Classes = 8 \quad (5.12)$$

Here shows the architecture of YOLOv4 :

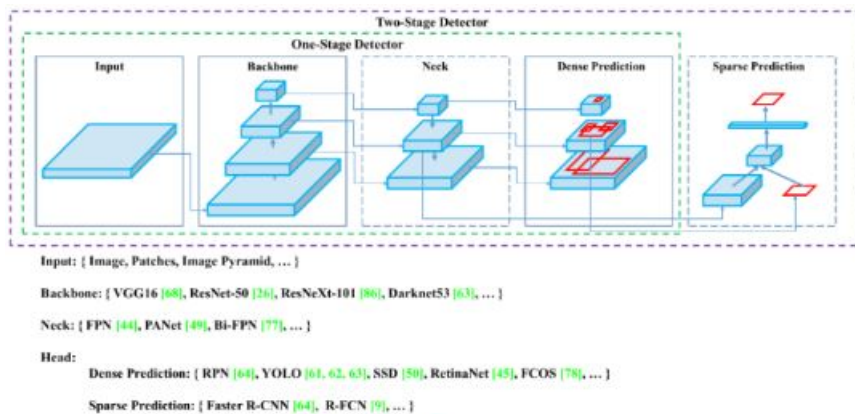


Figure 5.9: YOLOv4 architecture

## 5.2.2 YOLOv4-tiny Architecture:

The YOLOv4-tiny approach was developed from the ground up to provide significantly quicker object detection times than the original YOLOv4 method. Although the YOLOv4 technique uses the CSPDarknet53 network as its backbone, the YOLOv4-tiny approach employs the CSPDarknet53-tiny network. The CSPBlock module, rather than ResBlock module in the residual network, is used in the CSPDarknet53-tiny very small network. Splitting the feature map in half and then combining the halves using cross stage residual edge is what the CSPBlock module does. Hence, the gradient flow is enabled to spread over two distinct network channels, thereby increasing the correlation difference of gradient data. In comparison to the ResBlock module, the CSPBlock module improves the convolution network's ability to learn. YOLOv4-tiny eliminates the Mish activation function from the YOLOv4-tiny network and instead utilizes the LeakyReLU function as the activation function. Instead of spatial pyramid pooling as well as path aggregation network used by YOLOv4 to speed up object identification, the YOLOv4-tiny technique instead employs a feature pyramid network in order to extract feature maps at various sizes. Annotation tools, such as LabelImg, are used to label each image in the collection, resulting in a text file that describes the images. Here in our YOLOv4-tiny model we customized

$$BatchSize = 64 \quad (5.13)$$

$$NetworkSize = 416 \times 416 \quad (5.14)$$

$$MaxBatches 16000, Steps 12800 \text{ and } 14400 \quad (5.15)$$

$$Filters = 39 \quad (5.16)$$

$$Classes = 8 \quad (5.17)$$

Here is the architecture of YOLOv4-tiny which is given below:

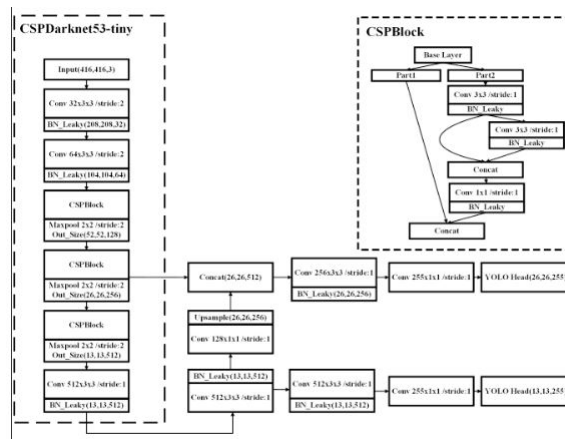


Figure 5.10: YOLOv4-tiny architecture



# Chapter 6

## Result and Analysis

### 6.1 Custom CNN Model

After developing the CNN model for our proposed system, we proceeded to train and validate the model with our processed dataset. We ran 65 training epochs and attained an accuracy of 97.16 percent on the training data as well as 97.58 percent on the validation dataset.

```
Epoch 61/65
Epoch 61/65
100/100 [=====] - 250s 3s/step - loss: 0.1194 - accuracy: 0.9699 - val_loss: 0.1395 - val_accuracy: 0.9603 - lr: 1.0000e-05
100/100 [=====] - 250s 3s/step - loss: 0.1194 - accuracy: 0.9699 - val_loss: 0.1395 - val_accuracy: 0.9603 - lr: 1.0000e-05
Epoch 62/65
Epoch 62/65
100/100 [=====] - 250s 3s/step - loss: 0.1166 - accuracy: 0.9701 - val_loss: 0.0998 - val_accuracy: 0.9759 - lr: 1.0000e-05
100/100 [=====] - 250s 3s/step - loss: 0.1166 - accuracy: 0.9701 - val_loss: 0.0998 - val_accuracy: 0.9759 - lr: 1.0000e-05
Epoch 63/65
Epoch 63/65
100/100 [=====] - 253s 3s/step - loss: 0.1133 - accuracy: 0.9704 - val_loss: 0.1185 - val_accuracy: 0.9674 - lr: 1.0000e-05
100/100 [=====] - 253s 3s/step - loss: 0.1133 - accuracy: 0.9704 - val_loss: 0.1185 - val_accuracy: 0.9674 - lr: 1.0000e-05
Epoch 64/65
Epoch 64/65
100/100 [=====] - 253s 3s/step - loss: 0.1119 - accuracy: 0.9705 - val_loss: 0.1048 - val_accuracy: 0.9718 - lr: 1.0000e-05
100/100 [=====] - 253s 3s/step - loss: 0.1119 - accuracy: 0.9705 - val_loss: 0.1048 - val_accuracy: 0.9718 - lr: 1.0000e-05
Epoch 65/65
Epoch 65/65
100/100 [=====] - 252s 3s/step - loss: 0.1088 - accuracy: 0.9716 - val_loss: 0.0977 - val_accuracy: 0.9758 - lr: 1.0000e-05
100/100 [=====] - 252s 3s/step - loss: 0.1088 - accuracy: 0.9716 - val_loss: 0.0977 - val_accuracy: 0.9758 - lr: 1.0000e-05
```

Figure 6.1: Last 5 Training and Validation Accuracy

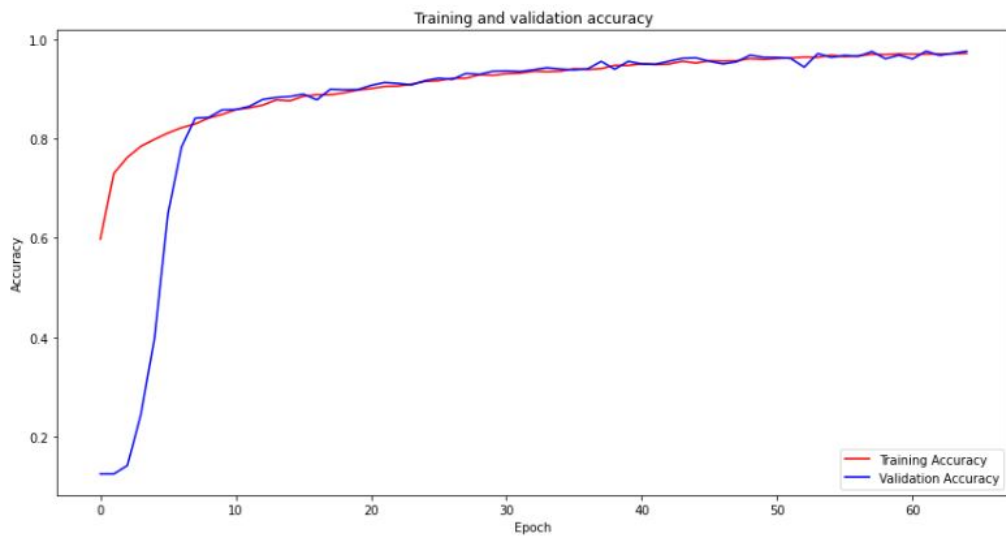


Figure 6.2: Training and Validation Accuracy

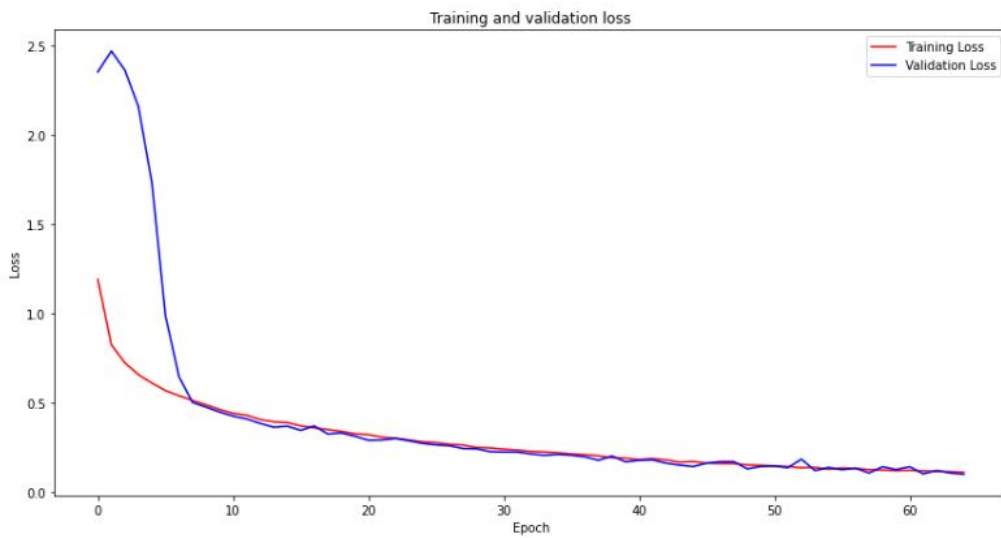


Figure 6.3: Training and Validation Loss

For our test dataset where we have separated 400 images. We calculate our test dataset accuracy, precision, recall, f1 score using a confusion matrix.

**Confusion matrix :**

A  $N \times N$  matrix used to analyze the model is called a confusion matrix. In the matrix, the actual goal values are compared with the values that the machine learning model says they should be. This gives us a complete picture of the way our classification model works and what kinds of mistakes it makes.

**True Positive (TP):**

The real result was good, just like the model thought it would be.

**True Negative (TN):**

Both the calculated and the theoretical values are identical. The model predicted that the number would be negative, which is exactly what happened.

**False Positive (FP):**

When the estimation is false, this is a type 1 error. The model predicted a good result, but the real work turned out to be bad. The Type 1 Error is another term for it.

**False Negative (FN):**

When the estimation is false, this is a Type 2 error. The model said that the result would be negative, but the real result was satisfactory.

**Precision and Recall:** Precision reflects the proportion of accurately predicted favorable outcomes.

$$Precision = TP / (TP + FP) \tag{6.1}$$

The amount of real positive cases that our model accurately identified is known as recall.

$$Recall = TP / (TP + FN) \tag{6.2}$$

**F1-score:** As the F1-score is the harmonic average of both recall and precision, it provides a centralized view of these two measurements. It reaches its highest value when Precision equals recall.

$$F1score = 2 / (1/Recall + 1/Precision) \tag{6.3}$$

**Accuracy:** The proportion of correctly predicted classes is a measure of a classification problem's accuracy. Just divide the entire number of predictions by the overall number of accurate estimations to get the accuracy rate.

$$Accuracy = (TP + TN) / (TP + FN + FP + TN) \tag{6.4}$$

Here, TP = True Positives.

FP = False Positives.

TN = True Negatives.

FN = False Negatives.

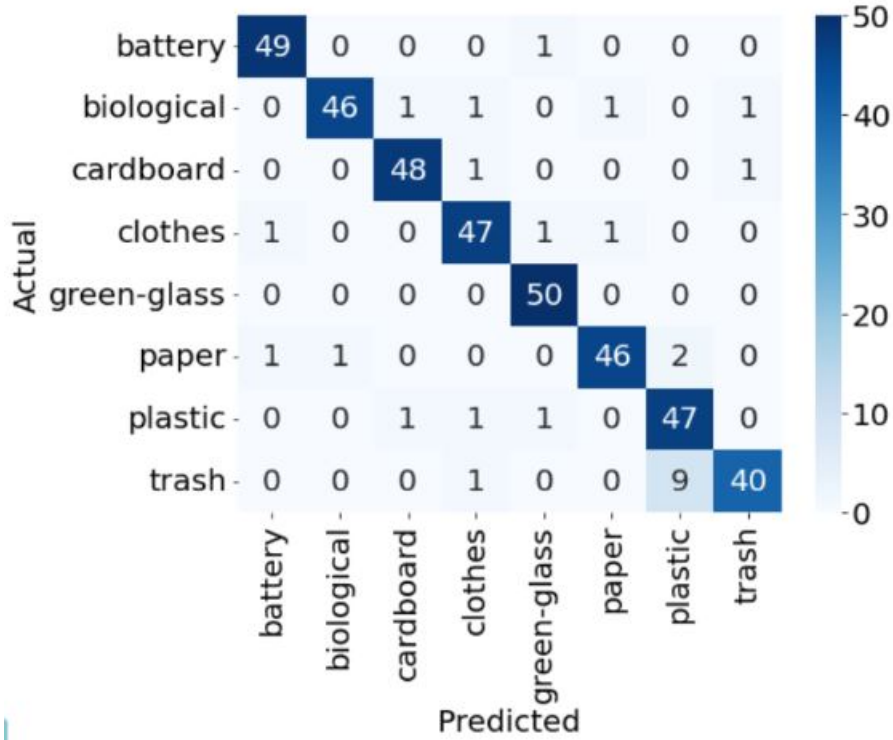


Figure 6.4: Confusion Matrix

In Fig 6.4: the confusion matrix for the Custom CNN Model is provided. The confusion matrix clearly shows one sample was incorrectly sorted into battery and just two misclassified for cardboard. We can also see that three sample was incorrectly stored in clothes and plastic and for four biological and paper. Lastly we can see the trash is giving the lowest performance which gave us 10 misclassified sample.

Table I: Confusion Matrix

	Precision	Recall	f1-score	support
battery	0.96	0.98	0.97	50
biological	0.98	0.92	0.95	50
cardboard	0.96	0.96	0.96	50
clothes	0.92	0.94	0.93	50
green-glass	0.94	1.00	0.97	50
paper	0.96	0.92	0.94	50
plastic	0.81	0.94	0.87	50
trash	0.95	0.80	0.87	50
accuracy			0.93	400
micro avg	0.94	0.93	0.93	400
weighted avg	0.94	0.93	0.93	400

Table I indicates the classification report. From this report we can see the Precision, Recall, f1 score, Accuracy. after calculating true positive, true negative, false positive, false negative it calculated this Precision, Recall, f1 score. And from this three it gave us accuracy.

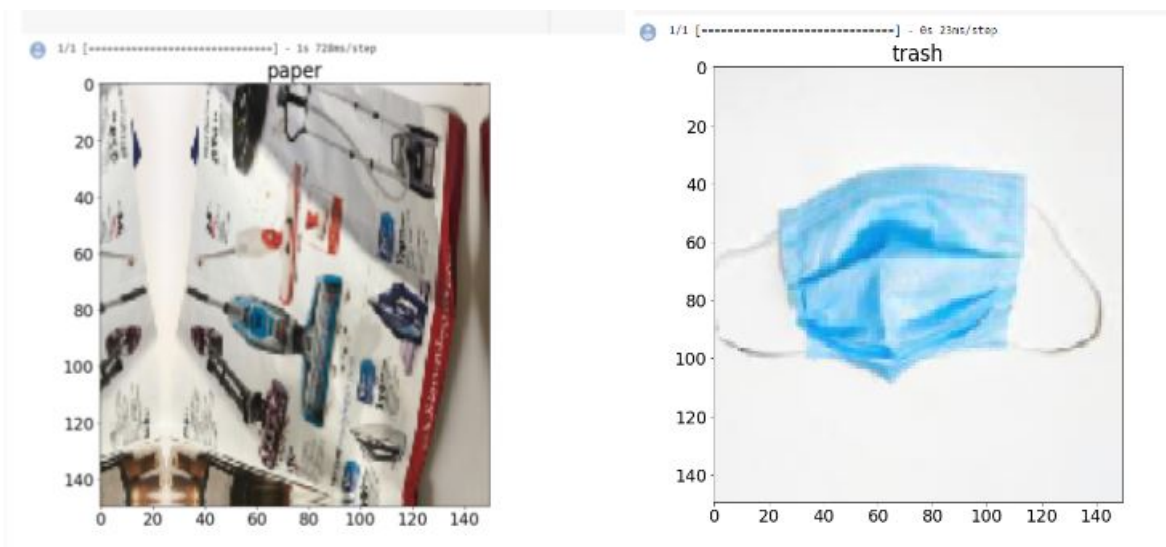


Figure 6.5: Test Prediction

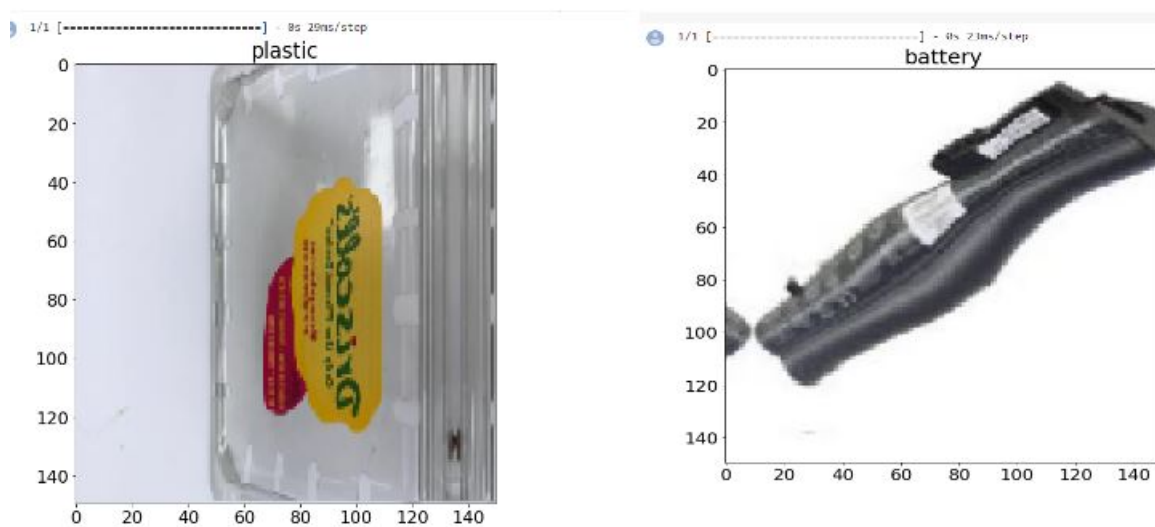


Figure 6.6: Test Prediction

We have also trained our model with the default pre-trained CNN models such as VGG16, ResNet50, MobileNetV2, InceptionV3, EfficientNetB0. Outcomes are given below :

### 6.1.1 Pretrained Model

#### ResNet50

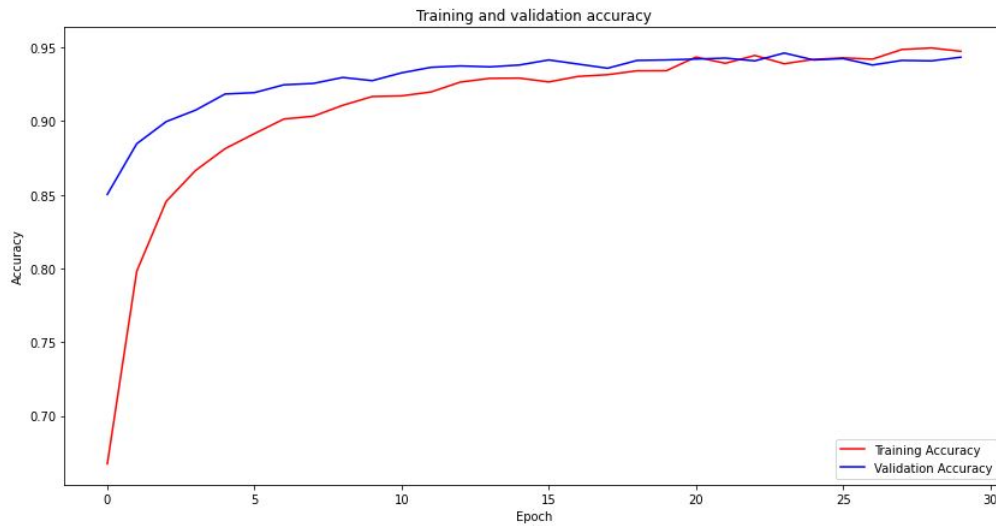


Figure 6.7: ResNet50 Training and Validation Accuracy

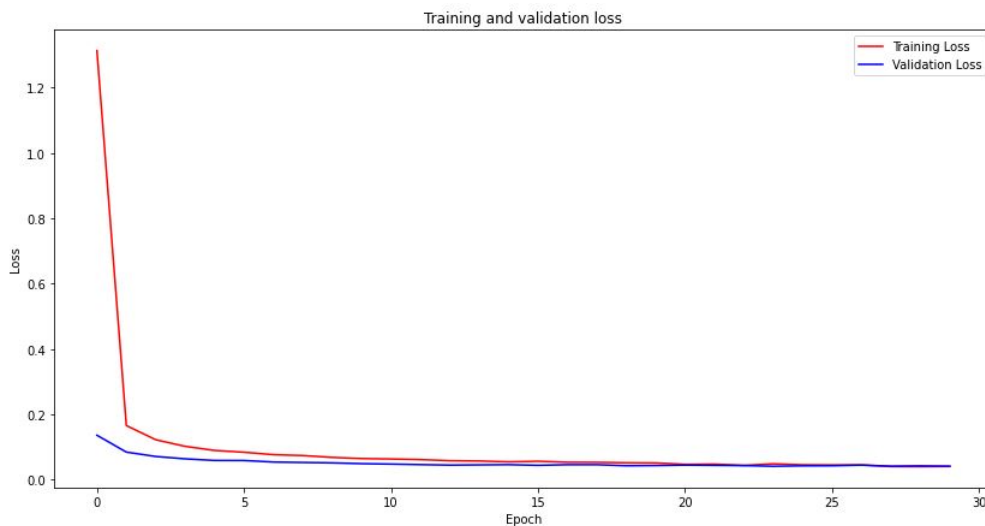


Figure 6.8: ResNet50 Training and Validation Loss

## VGG16

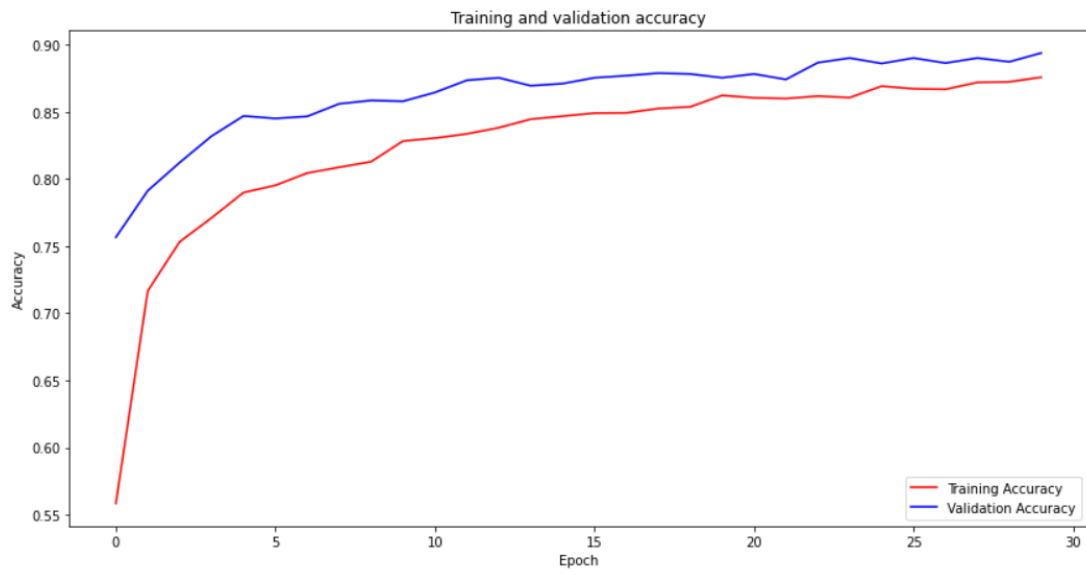


Figure 6.9: VGG16 Training and Validation Accuracy

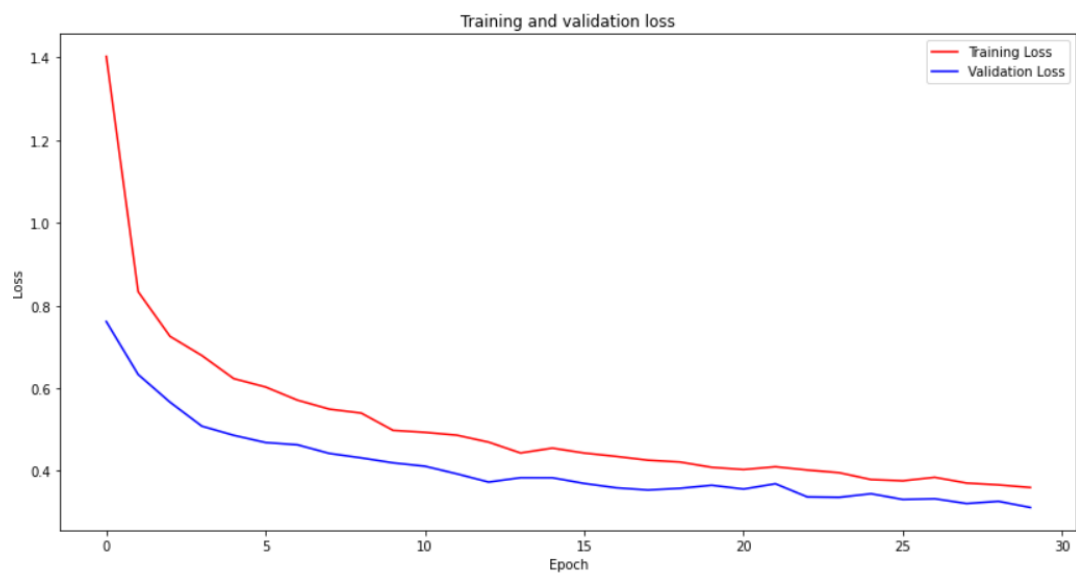


Figure 6.10: VGG16 Training and Validation Loss

## InceptionV3

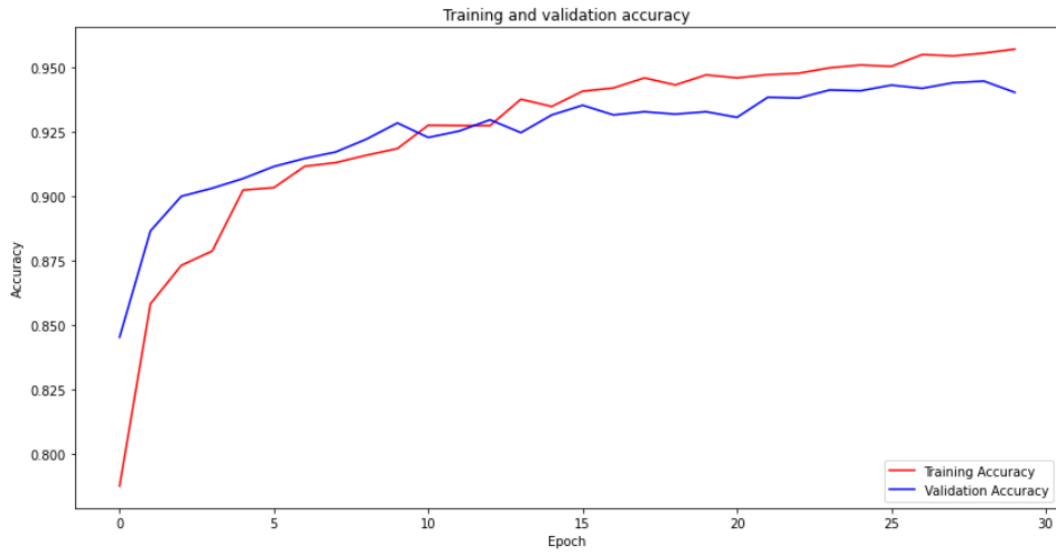


Figure 6.11: InceptionV3 Training and Validation Accuracy

Text(0.5, 1.0, 'Training and validation loss')

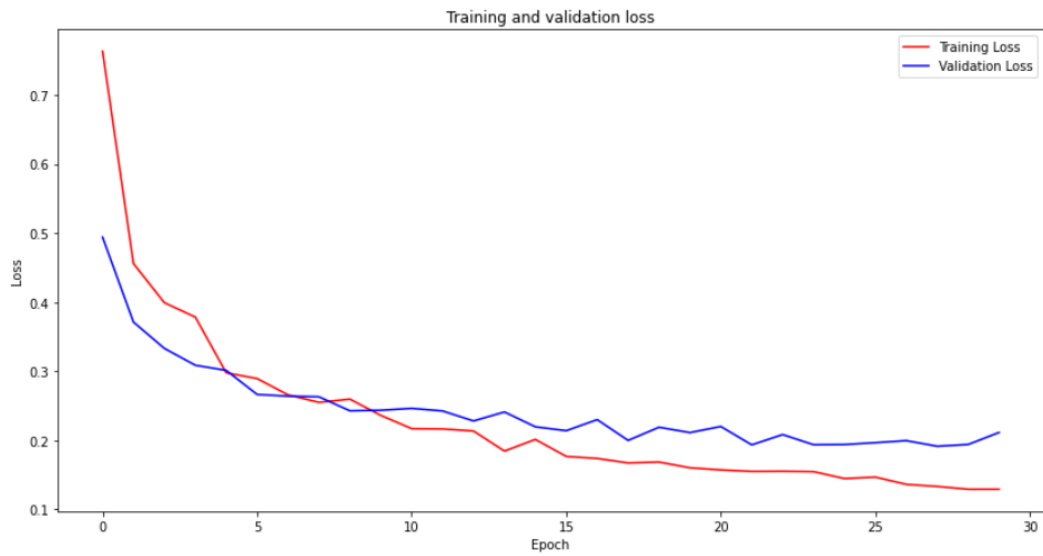


Figure 6.12: InceptionV3 Training and Validation Loss



## MobileNetV2

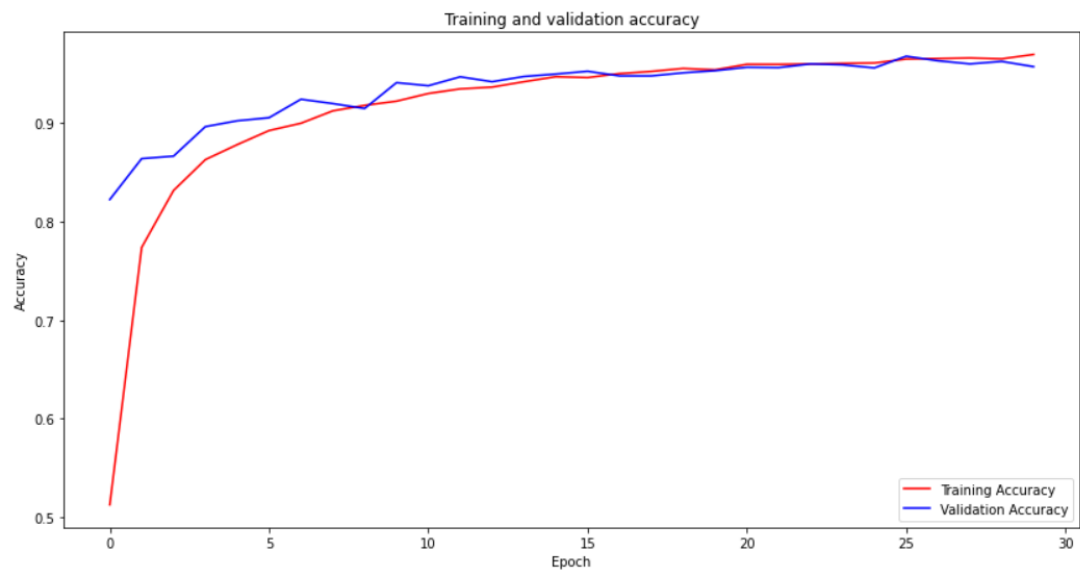


Figure 6.13: MobileNetV2 Training and Validation Accuracy

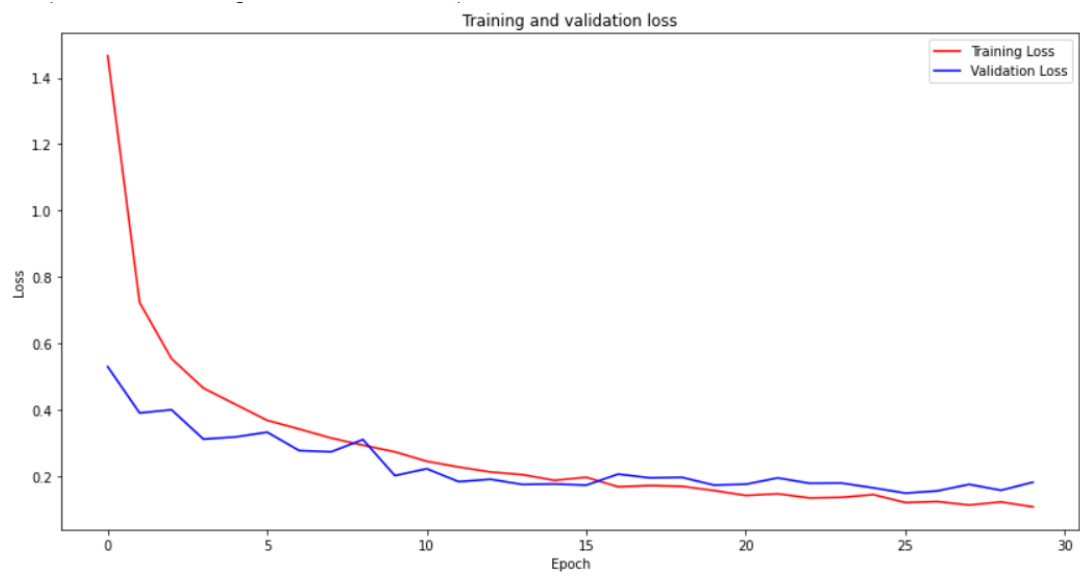


Figure 6.14: MobileNetV2 Training and Validation Loss

## EfficientNetB0



Figure 6.15: EfficientNetB0 Training and Validation Accuracy

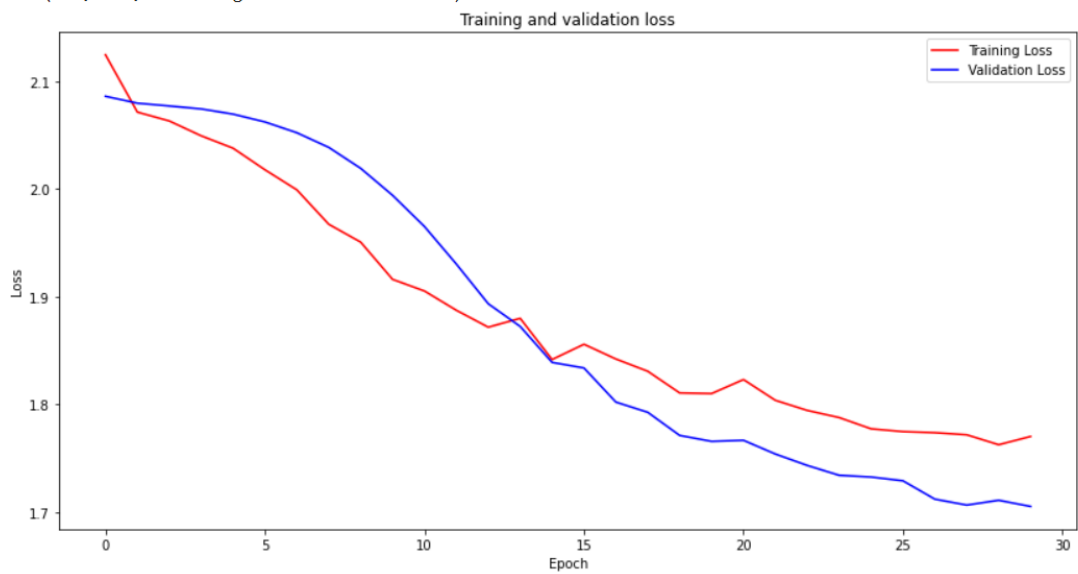


Figure 6.16: EfficientNetB0 Training and Validation Loss

**Table II: Table of Comparison**

Model Name	Train Accuracy	Val Accuracy	Parameters [M]
Custom CNN	97.16	97.58	9.9
ResNet50	94.97	94.34	49.7
VGG16	87.57	89.38	18.9
MobileNetV2	96.99	96.81	4.9
InceptionV3	95.71	94.47	23.9
EfficientNetB0	35.92	36.75	4.1

Table II shows the comparison between custom CNN model along with other pre-trained models.

Here, the results of five pre-trained CNN models and our custom CNN model are shown. Comparing the six models, we can state that our custom CNN model delivers almost the same results to the other five pre-trained models, which is an incredibly satisfying outcome for us.

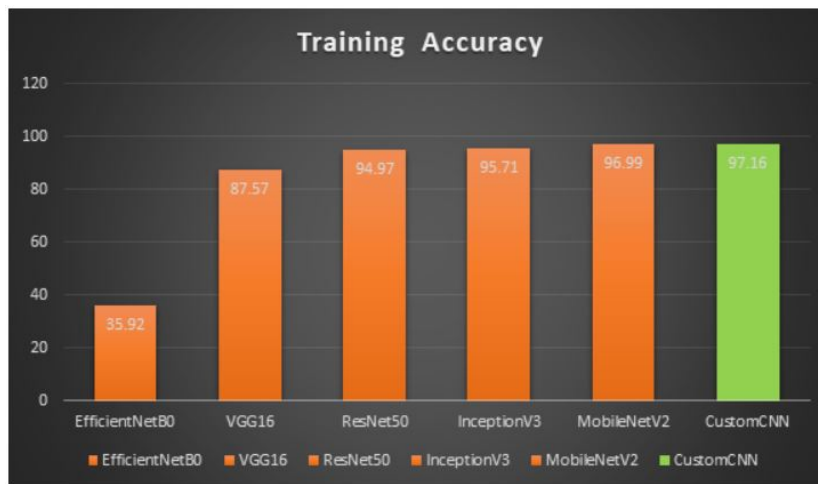


Figure 6.17: Model Training Accuracy

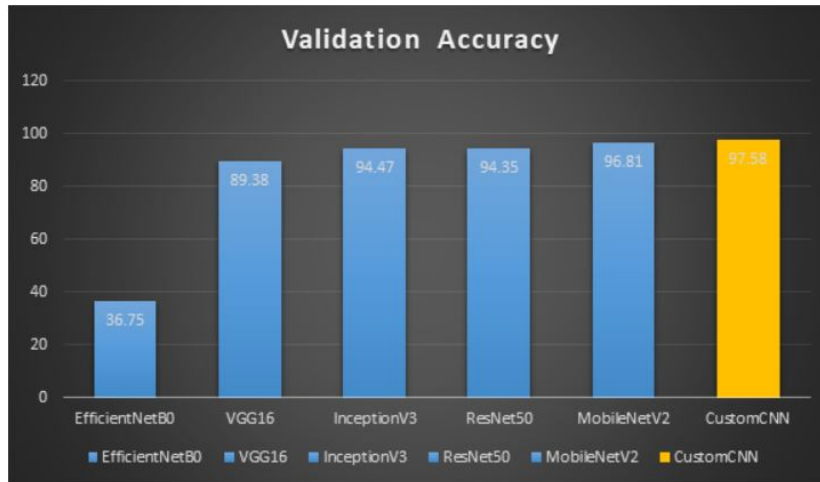


Figure 6.18: Model Validation Accuracy

## 6.2 YOLOv4 and YOLOv4-tiny Model

### 6.2.1 YOLOv4-tiny Model

From our YOLOv4-tiny model we have noticed that the time needed to complete the training process using YOLOv4-tiny is approx 5 hours. The YOLOv4-tiny model produces the mAP of 81.28 percent and an average loss of 0.2816. Other than these we also got the precision 0.60, recall 0.87, F1-score 0.71, and Average IoU 45.67 percent.

The Average Loss and mAP for YOLOv4 is shown in the below figure :

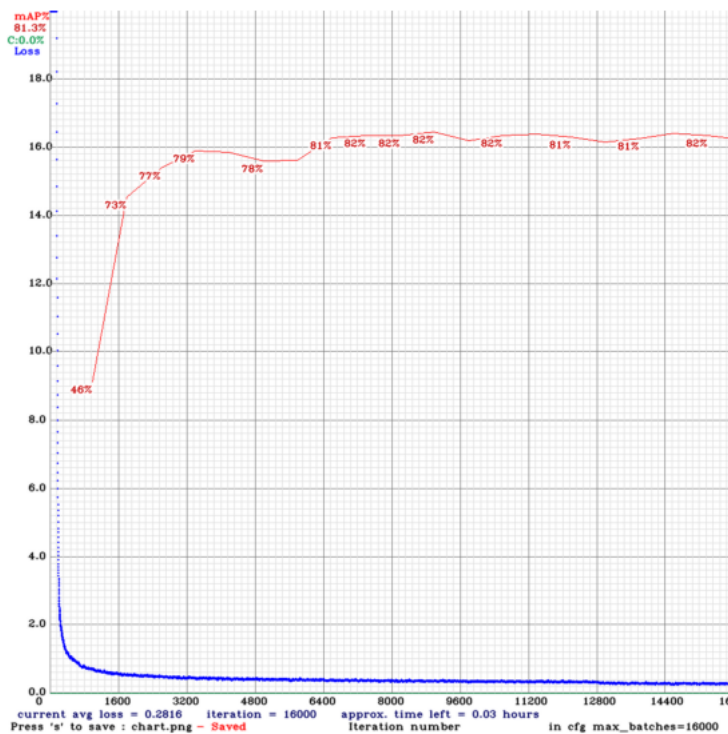


Figure 6.19: Average loss and mAP

The output of test images:



Figure 6.20: Testing image detection



Figure 6.21: Testing image detection

The output of webcam test images:



Figure 6.22: Testing image detection

The output of video images:

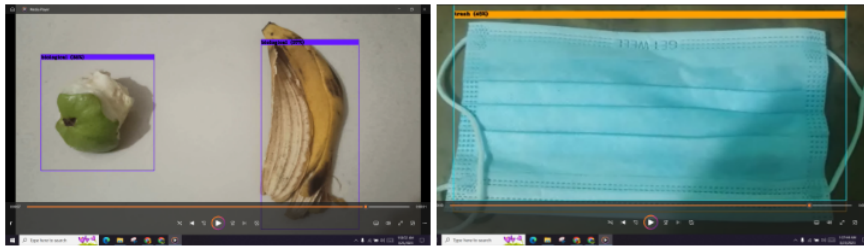


Figure 6.23: Testing image detection

## 6.2.2 YOLOv4 Model

From our YOLOv4 model we have noticed that the time needed to complete the training process using YOLOv4 is approx 24 hours. The YOLOv4 model produces the mAP of 85.73 percent and an average loss of 2.2254. other than these we also got the precision 0.78, recall 0.84, F1-score 0.81, and Average IoU 62.05. percent. The Average Loss and mAP for YOLOv4 is shown in the below figure :

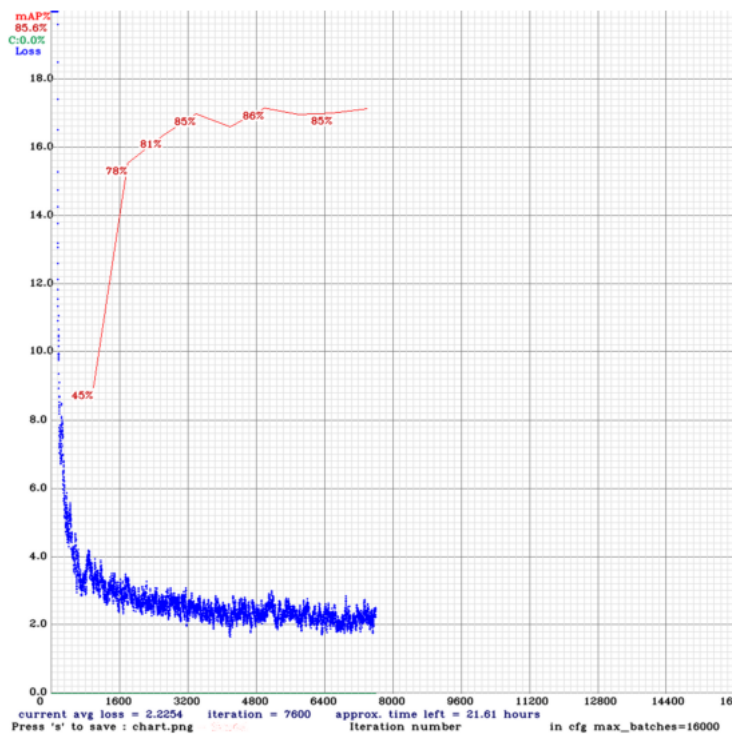


Figure 6.24: Average loss and mAP

The output of test images:



Figure 6.25: Testing image detection



Figure 6.26: Webcam image detection

The output of webcam test images:



Figure 6.27: Testing image detection

The output of video test images:

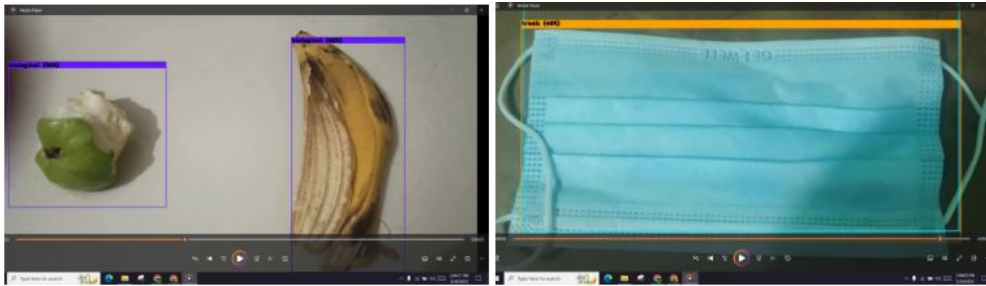


Figure 6.28: Testing image detection

### 6.2.3 Comparison between YOLOv4 and YOLOv4-tiny

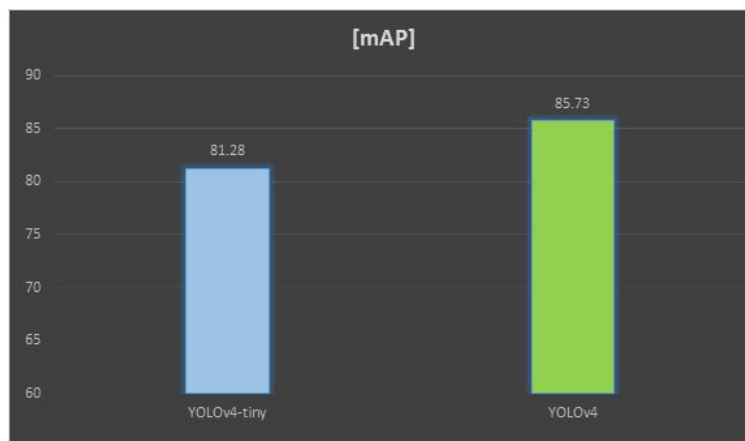


Figure 6.29: mAP results of YOLOv4 and YOLOv4-tiny

To summarize we can say that YOLOv4 is taking more time to compute but it is giving us better results. Though we couldn't complete the full iteration of 16000, it has given us more accurate result than YOLOv4-tiny. To conclude, YOLOv4-tiny provides the result more faster but it is not as accurate as YOLOv4.



# Chapter 7

## Conclusion

The largest cities in developing countries are expanding quickly. In spite of major investment, the city's waste management system remains outdated and dependent on processing performed by hand. The number of people working to bring this industry up to date is minimal, making it among the very few that has yet to benefit from modern science and technology. Our study's ultimate goal is to improve upon existing methods for sorting garbage in order to facilitate more efficient and cost-effective recycling. In this paper, we are proposing a system that will classify waste by using custom CNN model. We have trained our dataset which contains images and evaluated models using Custom CNN, VGG16, ResNet50, MobileNetV2, InceptionV3, EfficientNetB0 and made a comparison between them. We have utilized the YOLOv4-tiny architecture for purposes of detection. This architecture, which is less complicated than the YOLOv4 architecture and affects the performance of the algorithm in terms of predicting probability and predicting time, respectively. YOLOv4-tiny is capable of doing computations more quickly than its predecessor. The ability to detect and categorize things using our custom CNN model and YOLOv4-tiny model can contribute to the creation of an automated system for sorting garbage, which in turn can facilitate the recycling of more garbage. In this way, the findings of this study can contribute to cleaner environments and more pleasant urban settings. To summarise what we've learned, we can conclude that the YOLOv4 algorithm is more time-consuming to calculate, but it produces more accurate results. Although we were unable to finish the entire iteration of 16000, the results that it is producing are more accurate than those produced by YOLOv4-tiny. So, it is safe to claim that YOLOv4-tiny provides us with results more quickly, but they are not as exact as those provided by YOLOv4.

## Reference

- [1] Olugboja Adedeji, Zenghui Wang, 2019 "Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network" *Procedia Manufacturing* Volume 35, 2019, Pages 607-612
- [2] Sai Sushanth G, Jenila Livingston L M, Agnel Livingston L G X, 2021 "Garbage Waste Segregation Using Deep Learning Techniques" *IOP Conf. Series: Materials Science and Engineering* 1012 (2021) 012040
- [3] Gyawali, D., Regmi, A., Shakya, A., Gautam, A., Shrestha, S. (2020). Comparative Analysis of Multiple Deep CNN Models for Waste Classification. *ArXiv: Computer Vision and Pattern Recognition*. <http://export.arxiv.org/pdf/2004.02168>
- [4] Cheema, S. M., Hannan, A., Pires, I. M. (2022). Smart Waste Management and Classification Systems Using Cutting Edge Approach. *Sustainability*, 14(16), 10226. <https://doi.org/10.3390/su141610226>
- [5] Stephen L. Rabano, Melvin K. Cabatuan, Edwin Sybingco, Elmer P. Dadios, Edwin J. Calilung, 2018 "Common Garbage Classification Using MobileNet" <https://ieeexplore.ieee.org/xpl/conhome/8662828/proceeding>
- [6] Dongwei Guo, Lufei Cheng, Meng Zhang and Yingying Sun, 2021 "Garbage detection and classification based on improved YOLOV4" *Journal of Physics: Conference Series*, Volume 2024, 2nd International Conference on Computer Vision and Data Mining (ICCVDM 2021) 20-22 August 2021, Changsha, China
- [7] Andhy Panca Saputra, Kusriani, 2021 "Waste Object Detection and Classification using Deep Learning Algorithm: YOLOv4 and YOLOv4-tiny" *Turkish Journal of Computer and Mathematics Education*
- [8] Yongchuang Yang, 2021 "Waste Classification Based On Yolov4" *JOURNAL OF SIMULATION*, VOL. 9, NO. 6
- [9] Lun Zhaoa, Yunlong Panc, Sen Wangc, Liang Zhanga and Md Shafiqul Islamd, 2021 "Skip-YOLO: Domestic Garbage Detection Using Deep Learning Method in Complex Multi-scenes" *Research Square*
- [10] Saurav Kumar, Drishti Yadav, Himanshu Gupta, Om Prakash Verma, Irshad Ahmad Ansari, Chang Wook Ahn "A Novel YOLOv3 Algorithm-Based Deep Learning Approach for Waste Segregation: Towards Smart Waste Management" in 2020 <https://www.mdpi.com/2079-9292/10/1/14>
- [11] Malik, M. (n.d.). Waste Classification for Sustainable Development Using Image Recognition with Deep Learning Neural Network Models. *MDPI*. <https://www.mdpi.com/2071-1050/14/12/7222>
- [12] Jiang, Z., Zhao, L., Li, S., Jia, Y. (2020). Real-time object detection method

based on improved YOLOv4-tiny. ArXiv (Cornell University).  
<https://doi.org/10.48550/arxiv.2011.04244>

[13] Saputra, K. a. P. (2021). Waste Object Detection and Classification using Deep Learning Algorithm: YOLOv4 and YOLOv4-tiny. *Turkish Journal of Computer and Mathematics Education(TURCOMAT)*,12(14),1666–1677.

<https://turcomat.org/index.php/turkbilmata/article/view/10504>

[14] Aghilan M, Arun Kumar M, Mohammed Aafrid TS, Nirmal Kumar A, S.Muthulakshmi, March- 2020. “Garbage Waste Classification Using Supervised Deep Learning Techniques” *International Journal of Emerging Technology and Innovative Engineering* Volume 6, Issue 03.

[15] Ruiz, V., Sánchez, N., Vélez, J. F., Raducanu, B. (2019). Automatic Image-Based Waste Classification. *From Bioinspired Systems and Biomedical Applications to Machine Learning*, 422–431. [https://doi.org/10.1007/978-3-030-19651-6\\_41](https://doi.org/10.1007/978-3-030-19651-6_41)

[16] Deep Patel, Foram Patel, Samir Patel, Nihar Patel, Dhruvil Shah, Vibha Patel “Garbage Detection using Advanced Object Detection Techniques” in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)

[17] Sylwia Majchrowska, Agnieszka Mikołajczyk, Maria Ferlin. (Feb-2022) “Deep learning-based waste detection in natural and urban environments” *Waste Management* Volume 138, 1 February 2022, Pages 274-284.

[18] Nakib, A. A., Talukder, M., Majumder, C., Biswas, S., Hassan, J. (2021). Deep learning-based waste classification system for efficient waste management (Doctoral dissertation, Brac University).

[19] Garbage Classification (12 classes). (2021, January 24). Kaggle.  
<https://www.kaggle.com/datasets/mostafaabla/garbage-classification>

[20] Rastogi, A. (2022, March 19). ResNet50 - Dev Genius. Medium.  
<https://blog.devgenius.io/resnet50-6b42934db431>

[22] Great Learning. (2022, January 5). Everything you need to know about VGG16 - Great Learning. Medium.  
<https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918>

[23] Praharsha, V. (2022, January 11). YOLOv4 model architecture. OpenGenus IQ: Computing Expertise Legacy. <https://iq.opengenus.org/yolov4-model-architecture/>