Waste Material Detection

Twisha Kotecha

A project report submitted in conformity with the requirements
for the degree of Master's of Science in Information Technology

Department of Mathematical and Physical Sciences
Faculty of Graduate Studies
Concordia University of Edmonton

Waste Material Detection

Twisha Kotecha

Approved:

Nasim Hajari, Ph.D.

Supervisor Date

Committee Member Name, Ph.D.

Committee Member Date

Dr. Alison Yacyshyn

Dean of Graduate Studies Date

Abstract

We use plastic every day. Plastics can be found anywhere, in households and businesses. Plastic dependency has fueled the global production of plastic resins and fibers, growing at an annual compound rate of 8.4 percent. As a result, strong waste management is critical. A traditional waste management system is not so efficient and handy for regular households or small businesses. People do not recycle their waste correctly. Smartphones and apps can be used to overcome this issue. Such apps can perform real-time material detection and allow better waste management. The goal of this project is to create a material detection app that can perform smart waste management for household and small-scale businesses. The Android app analyses a live camera feed and recognizes materials of detected objects in real-time using a machine learning model which is based on TensorFlow Lite and TensorFlow. Material detection and waste classification are carried out in the TensorFlow framework using a pre-trained material detection model. TensorFlow Lite allows running TensorFlow machine learning (ML) models in Android apps. The TensorFlow Lite system renders prebuilt and customizable execution environments for running models on Android quickly and efficiently, along with the hardware acceleration option. We performed experimental analysis on various pre-trained models to analyze their accuracy. The developed app works for a single object at a time, It works in an indoor environment with good lighting conditions and requires a clear background.

Keywords: Material Detection,Deep Learning,TensorFlow,Machine Learning

Contents

1 Introduction **1**

2 Literature review **3**

3 Background **5**
3.1 Neural Networks . . . 5
3.2 Convolution Neural Networks . . . 5
3.3 Faster R-CNN . . . 6
3.4 OS . . . 7
3.5 TensorFlow . . . 7
3.6 Utils . . . 7

4 Project Design **8**
4.1 Project Architecture . . . 8
4.1.1 TensorFlow Lite for Android . . . 8
4.1.1.1 Machine learning models . . . 8
4.1.1.2 Run models on Android . . . 8
4.1.1.3 Build apps with TensorFlow Lite . . . 8
4.1.1.3.1 Run-time environment options . . . 9
4.1.1.3.2 Development APIs and libraries . . . 9
4.1.2 Application Overview . . . 10
4.2 Models . . . 15
4.2.1 MobileNetV2 . . . 15
4.2.2 InceptionV3 . . . 16
4.2.3 NasNetMobile . . . 16
4.3 Plugins Used . . . 17
4.3.0.1 NumPy . . . 17
4.3.0.2 Keras . . . 17
4.3.0.3 Keras-Applications . . . 18
4.3.0.4 Keras-Preprocessing . . . 18
4.3.0.5 h5py . . . 18
4.3.0.6 Matplotlib . . . 18
4.3.0.7 openCv-python . . . 18
4.3.0.8 pandas . . . 18
4.3.0.9 tensorboard . . . 19
4.3.0.9.1 tensorflow . . . 19
4.3.0.10 tensorflow-estimator . . . 19

5 Experimental Results **20**
5.1 Experiment . . . 20
5.2 Dataset . . . 23

6 Conclusions **25**

List of Tables

1 Training results. 22
2 Optimization results. 23

List of Figures

1 Functional execution flow for TensorFlow Lite models in Android apps. 9
2 Application features 11
3 Cardboard Detection 12
4 Glass Detection 12
5 Metal Detection 12
6 Paper Detection 12
7 Plastic Detection 13
8 Trash Detection 13
9 TensorFlow Lite libraries in app and models in assets folder 14
10 Diagram of MobileNetV2, which includes 17 inverted residual module (IRM) [31] 15
11 Diagram of InceptionV3 and its three core parts, i.e., InceptionV3 module I (IM I), InceptionV3 module II (IM II), and InceptionV3 module III (IM III). [31] 16
12 Nasnet Normal and Reduction Cell Architecture [37] 17
13 InceptionV3 model accuracy 20
14 MobileNetV2 model accuracy 21
15 NASNetMobile model accuracy 22

List of Abbreviations

AI	Artificial Intelligence
BN	batch normalization
CNN	convolution Neural Network
GIS	geographic information system
GLCM	grey level co-occurrence matrix
GPRS	general packet radio system
IRM	inverted residual module
KNN	K-nearest neighbour
LRN	Local Response Normalization
MACS	multiplying capacity
MLP	multilayer perceptron
ML	machine learning
RFID	radio frequency identification
ROI	Region of Interest
RPN	Region Propose Network
SIFT	scale-invariant feature transform
SVM	support vector machine

1 Introduction

Waste Management is quite a known issue currently but not everyone is considering it a serious issue so which is leading to an incorrect garbage disposal.[1]. Illegal dumping has long been a problem in many urban areas around the world. The odors and pollutants produced by abandoned household items, unloading rubbish, and construction remnants degrade the city and endanger the people's health. A few cities have proposed network-based voluntary reporting frameworks and observation camera-based monitoring frameworks to reduce illegal dumping. However, these approaches necessitate manual observation and recognition, which is prone to false alarms and proved to be costly. Garbage is a global problem that impacts any living organism in the ecosystem. According to a study, the Philippines' rubbish is responsible for 74 percent of the plastics that end up in the ocean [2]. Regular households or small businesses may neglect or lack the required infrastructure to properly segregate the daily waste. The waste management organizations will then need to spend more resources to fix this issue. Splitting rubbish into divergent components is the commonly used technique for properly segregating garbage. This is mostly done manually, which can be extremely dangerous to human health if not done properly. To work out this issue, waste classification and detection are necessary which can help an individual to differentiate between recyclable and solid waste.

Effective waste classification through smart frameworks such as smart apps can automate the process of recycling. Using material detection software optimized and the time-saving way can be achieved to manage waste at least at the primary level like daily waste from household and some business premises by easily recognizing materials. The traditional strategy is based on the goodwill of human labor, which is prone to failure when it comes to garbage sorting for recycling [3]. Deep learning techniques are being successfully used in different fields, including medical imaging, autonomous driving, and a variety of industrial applications such as computer vision and robot manipulation, with outstanding results on object recognition challenges. Applying these approaches to garbage sorting can increase the amount of recycled material, making life easier for the general public while also increasing the efficiency of the sector.

Deep Learning is a type of Machine Learning Algorithm that employs many layers of data representation and feature extraction. It is a subset of Artificial Intelligence. Deep learning is used in a variety of applications such as voice and visual recognition, speech to text conversion, face identification and recognition, object detection and recognition, drug detection, weather prediction, and so on. One group of deep learning model is convolution Neural Network (CNN). . This algorithm proved to be very effective in computer vision and object recognition applications[4]. Because it has already adapted or acquired the visual features and can transfer that information through transfer learning, a previously trained CNN will always perform better on new images for classification.

The goal of this study is to create and develop a waste segregation architecture using a deep learning approach. The image will be detected with the use of an image processing approach that recognises wastes based on their shape, colour, dimension, and size [5].This technique will automatically assist the algorithm in learning relevant aspects from trash training images and, as a result, recognising such features in new photographs. Garbage will be categorised into multiple classes using the CNN technique. TensorFlow's Object Detection API and the Faster R-CNN approach were used to create this characterization scheme.By demonstrating with the class, the system will provide information about garbage and what kind of material it contains (cardboard, paper, metal, glass, plastic, and trash). The major goal of this research is to create software that can detect material that is contained in a certain object for each trash item with a percentage, reducing human effort in waste segregation and speeding up the complete process.

Previously, the vast majority of investigations in this study were conducted on images containing only objects, with the implementation which detects objects not materials, however, this paper is focusing on the material detection of a specific object, in order to determine which material is present in that object. Machine learning techniques like SVM and deep learning methods like VGG16 were largely used in previous work for classification [6]. In this project we test 3 different models that are **MobileNetV2 (float32, float16, integer quantized)**, **InceptionV3 (float32, float16, integer quantized)**, and **NASNetMobile (float32, float16, integer quantized)**, to see which is the most accurate.

2 Literature review

Various works have been carried out in the recent years with the goal of decreasing the impact of improper trash disposal. Over the years many techniques based on machine learning and artificial neural networks have been developed to detect waste and recognize materials have been developed.

Mindy Yang et al.[7] compared SVMs with scale-invariant feature transform (SIFT) [8] and an eleven-layer CNN design like AlexNet in order to categorise rubbish. The result demonstrates that the SVM outperforms CNN[30].

Another paper has worked on deep learning CNN and ML algorithm SVM for garbage classification to perform powerful waste sorting [10]. SVM had a rate of accuracy that was nearly as high as CNN. Nonetheless, as data and GPU utilisation increased, the CNN algorithm improved accuracy and reduced the impact of over-fitting. For classification purposes, the SVM model was used in the final hardware execution. It makes use of a Raspberry Pi 3 and a high-definition HD camera. The camera captures a glimpse of the waste and saves it as a PNG file. The taken image is sent to the preset classifier for grouping, and different LED coloration is turned on depending on the category.

Arebey, M. offers detection based on grey level co-occurrence matrix (GLCM), a method for waste classification, which combines sophisticated communication mechanizations with GLCM to strengthen waste collection [11]. The suggested framework integrates a camera and streams out solid waste monitoring using a few communication technologies such as geographic information system (GIS), radio frequency identification (RFID), and general packet radio system (GPRS). The GLCM's highlights are then used as inputs to a waste segregation approach involving a multilayer perceptron (MLP) and a K-nearest neighbour (KNN). The KNN classifier outperformed the MLP, according to the results.

Recycle Net's creators experimented with the topologies of the well-known deep convolutional neural network. The training is being done without pre-trained weights, and Inception-v4 outperformed as compared to used other model such as mobilenetv2, with a test accuracy of 90 percentage. Using ImageNet weights, the designers performed transfer learning and fine-tuning of weight parameters, with DenseNet121 achieving the best result with 95 percent. The prediction time for this last setup is longer.

The authors suggest a computerised framework based on a deep learning method as well as traditional strategies for properly separating waste into four distinct recycling classes (Paper, metal, glass, and plastic) [12]. The results showed that VGG-16 approaches are a useful methodology for this problem, with an accuracy rate of 93 percent in the best case scenario [13].

"Auto Rubbish," a group's creation at the TechCrunch Disrupt Hackathon, is an autonomous garbage bin that separates trash based on recycling and composting

characteristics. Their architecture features a revolving top and employs a Raspberry Pi camera. For object detection, the team used Google's TensorFlow AI engine and built their layer on top of it [14].

In general, computer vision has been viewed as a tool to aid waste sorting in recent years, and deep learning algorithms have produced useful results in controlled scenarios. Oluwasanya Awe et.al. [15] used the Faster R-CNN model to investigate object identification in waste management and found that the results were reasonable. In the same paper, the authors presented an approach for obtaining region proposals and object classification with a mean average precision of 68.3 percent using the Faster Region-based Convolutional Neural Networks (Faster R-CNN) procedure[16]. The garbage was divided into three categories (landfill, recycling, and paper).

3 Background

3.1 Neural Networks

The artificial neuron is at the heart of machine learning (including deep learning). The learning computations are carried out with the help of neurons. A large number of neurons operate together in an Artificial Intelligence model to do difficult numerical/mathematical calculations. The artificial neural network is the result of these neurons forming a network. Artificial neurons are inspired by the neurons that exist biologically in the human sensory or neurological system. An artificial neural network, like the human brain's neural network, is divided into layers. In an artificial neuron, the dendrites are just the neurons' information terminals.The input is processed by the axon through synapses and dendrites of another neuron, and its output is transferred to separate neurons. In the computational model, the weight of the line increases the input signals that move along the line of the input. The mathematical function is used to process the weighted input signal. The activation function f is the name given to this function. The signal that has already been processed is sent to the next layer neurons for further processing. The weight of the link between neurons is recognised to be a learning factor in this model.Throughout the training, the value of this model is updated with the goal of reducing the error to zero. The signals conveyed by the dendrites in the human body are added in the cell body, and if the sum exceeds a threshold amount, signals are initiated by the axon. A similar process is used in a mathematical or numerical model. The activation function f determines the threshold value. The standard decision of the activation function is known as the Sigmoid Function. The summing value is passed to the sigmoid function, which turns it to a reach between 0 and 1 [17].

3.2 Convolution Neural Networks

The primary purpose of convolutional neural networks (CNN) is to group or categorise images, cluster them by similarity, and conduct object recognition with the help of artificial neural networks. The visual information is taken by the convolutional neural network, which interprets the image as a tensor, which is a matrices of numbers with extra dimensions, and does a search[18].The instances that recognise the images as volumes are made up of 3D objects[19] [20]. It's being used in a variety of applications, including facial recognition and object identification. It's one of the most interesting non-trivial assignments [21]. The three separate layer types that are recognised as a part of CNN are the convolutional layer, subsampling layer, and fully connected layer in neural networks[22]. CNN is mostly utilised for image recognition since it has a number of advantages over other methods.

3.3 Faster R-CNN

Only a single item in the image can be grouped or categorised using a rudimentary CNN algorithm. Faster R-CNN is a CNN with Region Propose Network augmentation (RPN)[23]. The Faster R-CNN technique is used because it aids in the recognition of several objects in a single image. Two modules make up a faster R-CNN. The main module is a deep convolutional network, in which RPN will be used to propose regions, and the succeeding module will classify the proposed photos. In RPN, the output for a particular picture is displayed as a rectangle object position with the item's score.Anchors are the name given to the object's proposal. The potential of objects will be predicted in the background with the help of an RPN. To do so, you'll need to build a training dataset comprising labelled and named items in the image. To restructure the projected areas, a Region of Interest (ROI) pooling layer is used. It will then be used to classify the image within the area and forecast the offset values around the bounding boxes. The accuracy of the final model will be determined by how well the essential regions are proposed.If the regions recommended to choose the appropriate region depending on the object, it might very well be classified into multiple classification classes[25].

The loss of Faster RCNN is divided into two parts[16].

1. loss of RPN
2. loss of Fast RCNN

Both the losses include:

- Classification loss
- Regression loss

The equation of the normalized classification loss is:

$$\frac{1}{N_{cls}} \sum_{i} L_{cls}(p_i, p_i^*) \tag{1}$$

Where,

N_{cls} = total number of anchors for classification,

p_i = predicted probability of the anchor I,

$L_{cls}(p_i, p_i^*)$ = classification loss.

The equation of the normalized regression loss is:

$$\lambda \frac{1}{N_{reg}} \sum_{i} p_i^* L_{reg}(q_i, q_i^*) \tag{2}$$

Where,

λ = balancing parameter,

N_{reg} = total number of anchor for regression,

q_i = vector representing the predicted bounding box,

$L_{reg}(q_i, q_i^*)$ = regression loss.

To extract feature maps Faster RCNN uses convolution layers. The equation of the entire loss function of the Faster RCNN network is:

$$L(\{p_i\}, \{q_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(q_i, q_i^*) \tag{3}$$

3.4 OS

OS is one of Python's import libraries. It is used to provide paths with OS-dependent functionality and is also used to manipulate paths.This allows the system to connect to the underlying operating system running python on os. For the path, specify the access path and sys. path uses the path defined for system activity.Tarfile is used to represent large amounts of data by reading and writing. The import file contained therein reduces the size of the code used for writing and reading [4].

3.5 TensorFlow

TensorFlow is used to perform simple math calculations developed by Google. Modern Python libraries can use this library to build deep learning models directly. This is a kind of math library, an AI / ML application like a neural network.There are different types of deep learning models that can be accessed and deployed using the TensorFlow pip command. Before you start training your model, TensorFlow helps you grow your data. In addition, it is used to perfect the efficiency of the algorithm and then downloads the weights of the pre-trained image mesh [6].

3.6 Utils

One of the parts of the Python library is used for collections of functions and classes. The main function of the utility is to support the implementation of convolutional neural networks. This project has the TensorFlow library utilities installed [6].

4 Project Design

This section describes the libraries needed to implement TensorFlow's embedded Android applications. It also describes the architecture, machine learning models, project directories, application overview, datasets, and plugins used when running pre-trained models.

4.1 Project Architecture

4.1.1 TensorFlow Lite for Android

TensorFlow Lite enables to run TensorFlow machine learning (ML) models in Android apps.TensorFlow Lite system come up with pre-built and customizable execution environment for running model on Android quickly and efficiently, including options for hardware acceleration.

4.1.1.1 Machine learning models

TensorFlow Lite uses TensorFlow models that are transformed into tiny, portable, and more systematic machine learning model formats. Either of them pre-build models or customized models by converting them into TensorFlow Lite format can be used.

4.1.1.2 Run models on Android

A TensorFlow Lite model is recurring inside an Android app that takes in data, undertakes the data, and produces a prediction based on the model's logic. A TensorFlow Lite model needs a particular runtime environment for execution and the data which is streamed into the model must be in a specific data format which is known as a tensor. When a model processes the data that is known as running an inference and it produces prediction results as new tensors and sends it to the Android app in order to perform an action, like showing the result to the user or executing additional required logic.

Android app needs following elements to run it at the functional design level:

- TensorFlow Lite **run-time environment** for executing the model
- **Model input handler** to transform data into tensors
- **Model output handler** to receive output result tensors and interpret them as prediction results

4.1.1.3 Build apps with TensorFlow Lite

This section evokes recommended and quite an ordinary path for implementing TensorFlow lite in Android apps. Mainly run-time environment and development libraries are important sections.

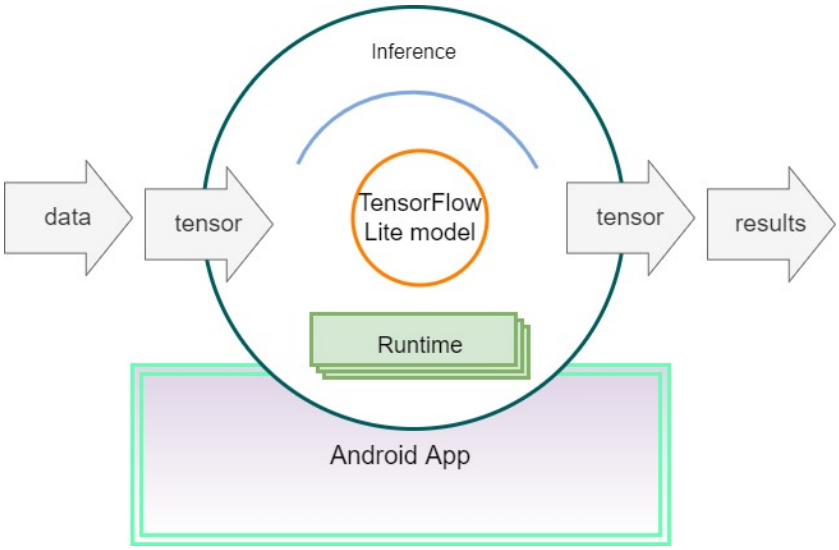

Figure 1: Functional execution flow for TensorFlow Lite models in Android apps.

4.1.1.3.1 Run-time environment options

There are various ways to delegate a run-time environment for executing models in an Android app. These are the preferred options:

- **Standard TensorFlow Lite run-time environment (recommended)**
- Google Play services run-time environment for TensorFlow Lite (Beta)

Generally, a TensorFlow Lite run-time environment is recommended as its a more adaptable environment running the model on Android. The run-time environment offered by Google Play services which is more suitable and space-efficient than the standard environment. Because it's loaded from Google Play store resources and not bundled in a specific app. Some advanced use cases require customization of a model run-time environment. To access these run-time environments in an Android app, it's required to add TensorFlow Lite development libraries to in app development environment.

4.1.1.3.2 Development APIs and libraries

There are two main APIs that can use to integrate TensorFlow Lite machine learning models into the Android app:

- **TensorFlow Lite Task API (recommended))**
- TensorFlow Lite Interpreter API

Interpreter API comes up with classes and methods for running inferences with existing TensorFlow Lite models. The TensorFlow Lite Task API encases the Interpreter API and delivers a high-level programming interface for conducting common machine learning tasks on handling visual, audio, and text data.

1. **Obtain models**
   Running a model in an Android app requires a TensorFlow Lite-format model. Pre-built models or build ones are also can be used with TensorFlow and converting them to the Lite format.

2. **Handle input data**
   Any data which is going to pass into the ML model requires to be in form of a tensor with a specific data structure, which is known as the shape of the tensor. To process data with a model, app code requires a transformation of data from its native formats, like image, text, or audio data, into a tensor in the required shape for the individual model.

   The TensorFlow Lite Task library provides data handling logic for transforming visual, text, and audio data into tensors with the correct shape to be processed by a TensorFlow Lite model.

3. **Run inferences**

   Processing data through a model to generate a prediction result is known as running an inference. Running an inference in an Android app requires a TensorFlow Lite run-time environment, a model and input data.

   The speed of generating inference of a model on a particular device depends on the size of the data processed, the complexity of the model, and available computing resources like memory and CPU or specialized processors are known as accelerators.Machine learning models can run swiftly on these diversified processors such as graphics processing units (GPUs) and tensor processing units (TPUs), using TensorFlow Lite hardware drivers known as delegates.

4. **Handle output results**
   Models produce prediction results as tensors, which are required to be handled by the Android app by taking action or displaying a result to the user. Model output results can be as simple as a number corresponding to a single result (0 = dog, 1 = cat, 2 = bird) for image classification, to much more complex results, such as multiple bounding boxes for several classified objects in an image, with prediction confidence ratings between 0 and 1.

4.1.2 Application Overview

This application is developed in Java for the front-end part which is done using **Android Studio IDE** and to run a machine learning model on an app it uses 3 libraries of TensorFlow Lite as shown in the Figure 9. In the android application project, there is one directory named **assets** which contains models that are used in the android application. Also, that folder contains **labels** file which contains labels used with these models and it mainly has these six categories as follows:

1. cardboard

2. glass
3. metal
4. paper
5. plastic
6. trash

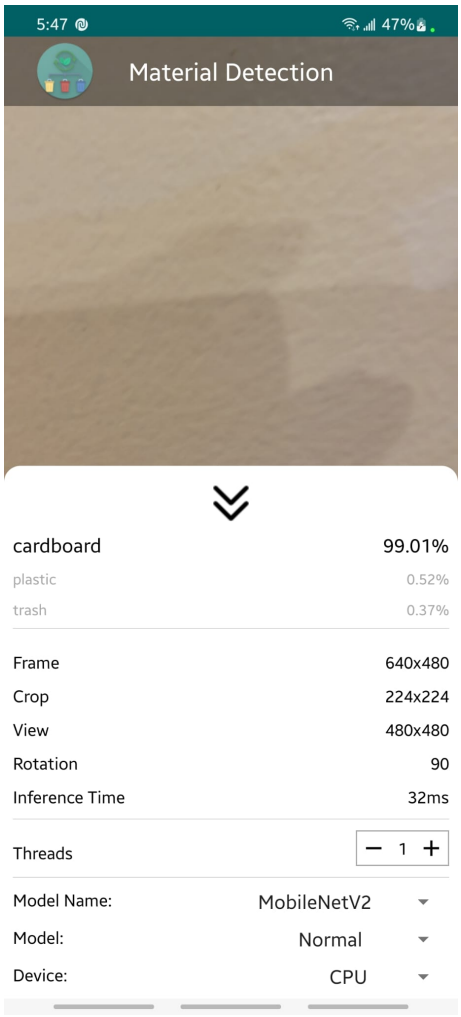

Figure 2: Application features

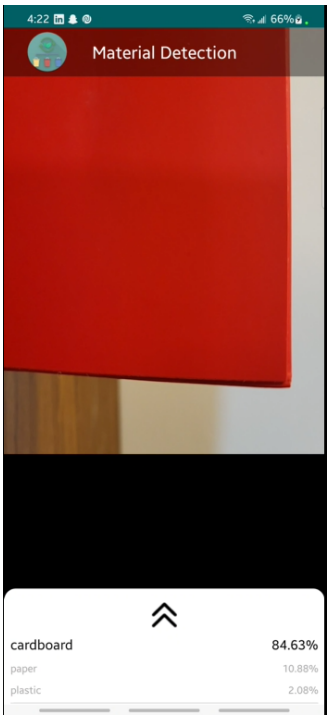

Figure 3: Cardboard Detection

Figure 4: Glass Detection

Figure 5: Metal Detection

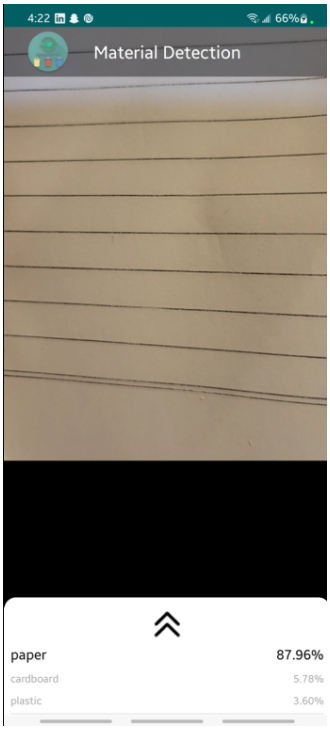

Figure 6: Paper Detection

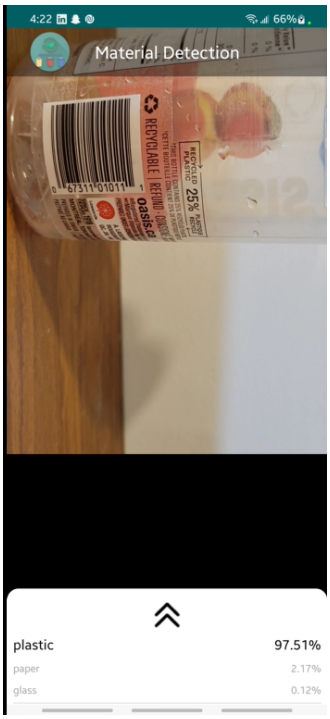

Figure 7: Plastic Detection

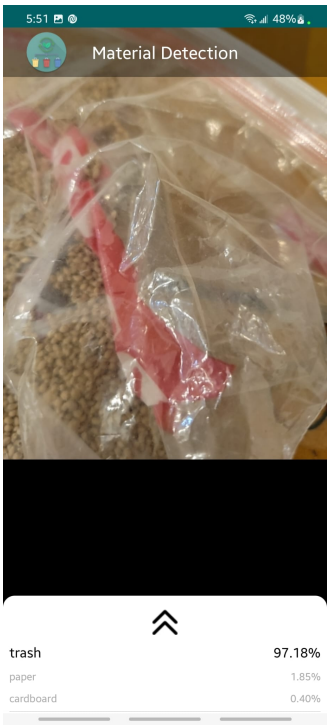

Figure 8: Trash Detection

The application requires camera permission in order to detect the material using the camera of the android device.

This application provides options for switching devices such as **CPU**, **GPU**, **NNAPI**. The inference time depends on the device selection. Even if you choose NNAPI, the inference time is short, and with the CPU, the GPU is on the other side of the NNAPI and the CPU. In other words, GPU inference time is shorter than CPU and longer than NNAPI.

Not only that but also, this application allows you to switch between the three models used in apps such as **MobileNet1V2**,**InceptionV3** and **NASNetMobile**. You can also select a model type such as Normal, Quantized, Float16. In addition, the app will display the inference time each time.

It's common that many systems have some sort of limitations, This application requires clean background to obtain more accurate results also it provides good results with only placing one object to detect material of any object. Additionally, an indoor environment with good lighting provides a better result. Also with different angles results may vary.

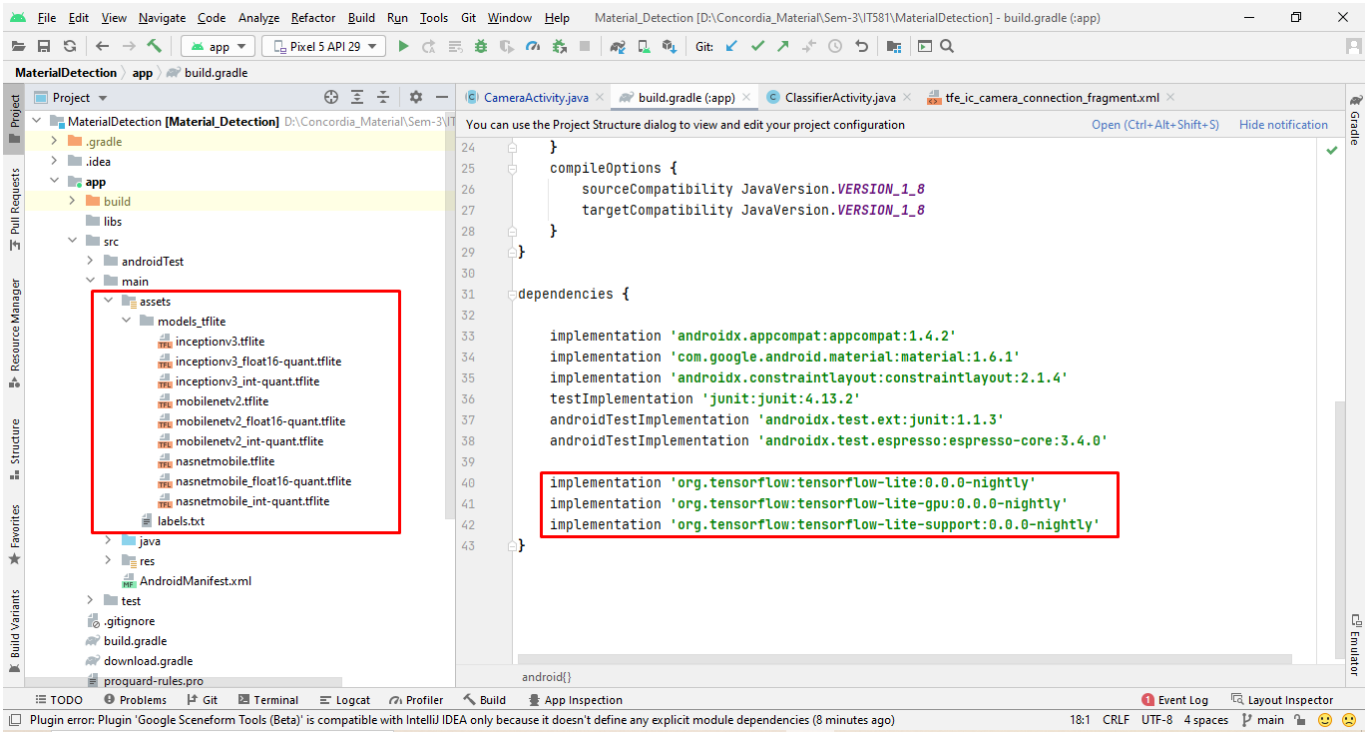

Figure 9: TensorFlow Lite libraries in app and models in assets folder

Purpose of this library: **implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'** and **implementation 'org.tensorflow:tensorflow-lite-support:0.0.0-nightly'**, is The TensorFlow Lite Android Support Library was developed to handle the inputs and outputs of TensorFlow Lite models and make the TensorFlow Lite interpreter more user-friendly.

TensorFlow Lite supports multiple hardware accelerators. This library: **implementation 'org.tensorflow:tensorflow-lite-gpu:0.0.0-nightly'** allows to use the GPU backend with the TensorFlow Lite delegate API on Android and iOS.

GPUs are designed to deliver high throughput for large, parallelizable workloads. Therefore, they are best suited for deep neural networks, each consisting of a large number of operators operating on one or more input tensors. These tensors are usually low latency because they can be easily split into smaller workloads and run in parallel.At best, GPU inference is now running fast enough for previously unavailable real-time applications.

Unlike CPUs, GPUs use 16-bit or 32-bit floating point numbers and do not require quantization for optimal performance.The delegate accepts an 8-bit quantization model, but the computation is done in floating point. See the extended documentation for more information.

Another advantage of GPU inference is power efficiency. The GPU performs calculations in a very efficient and optimized way, so it consumes less power and generates

less heat than doing the same task on the CPU.

Application is using pre-trained models as following:

- MobileNetV2 (float32, float16, integer quantized)
- InceptionV3 (float32, float16, integer quantized)
- NASNetMobile (float32, float16, integer quantized)

Mainly pre-trained model is builded based on a neural network built with keras and optimized with TensorFlow Lite API.

Here the difference between float32 and float16 is, that float16 is less accurate but faster than float32, and flaot32 is more accurate than float16 but consumes more memory. If accuracy is more important than speed,float32 is preferable. and if speed is more important than accuracy,float16 can be used.

4.2 Models

4.2.1 MobileNetV2

Google released the network, MobileNetV2 in 2018, which is mainly designed to target mobile and embedded devices [26]. As shown in the **Figure 10** that the diagram of MobileNetV2 that includes 17 IRMs, the inverted residual module (IRM) is the basic and key unit of MobileNetV2.

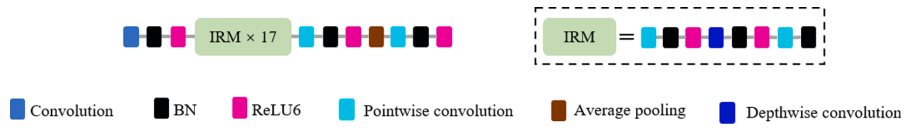

Figure 10: Diagram of MobileNetV2, which includes 17 inverted residual module (IRM) [31] .

Firstly complete IRM performs point-wise convolution to inflate the number of features depicting channels to 6 times the input number and then manipulates a depth-wise convolution and another point-wise convolution. The depth-wise convolution changes the width and height of input feature maps [27]. The second point-wise convolution reduces the number of feature map channels to the input number [28].Furthermore, In order to improve the model convergence speed and avoid gradient explosion, batch normalization (BN) layers are attached to each convolution layer [29]. ReLU6 (rectified linear unit 6) non-linearity activation function hikes the sparsity of the network and scales down the interdependence between parameters, thus being widely used after BN layers [9]. Nevertheless, the ReLU6 function will steer to crucial information loss in the case of low-dimensional input. Hence, the feature maps are directly delivered to the next convolution layer without using a RuLU6 afterward the second point-wise convolution layer and the BN layer.

4.2.2 InceptionV3

Google also proposed InceptionV3 in 2015 which can also be run on android phones [32]. Which has mainly 3 core modules that mean InceptionV3 module 1, InceptionV3 module 2, and InceptionV3 module 3, as shown in **Figure 11**. All these 3 modules have the same principle and that is increasing the depth and width of the network while leading to little computational cost.

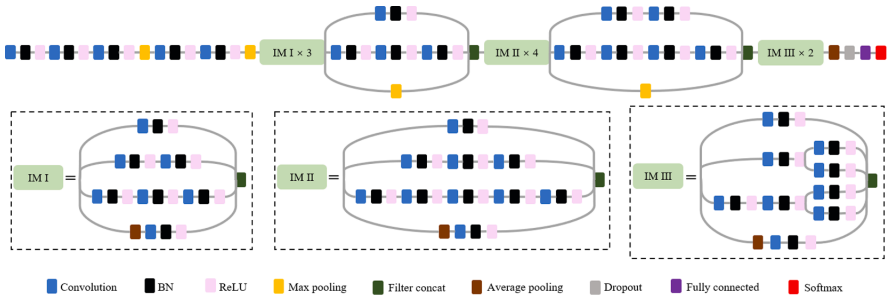

Figure 11: Diagram of InceptionV3 and its three core parts, i.e., InceptionV3 module I (IM I), InceptionV3 module II (IM II), and InceptionV3 module III (IM III). [31] .

In these 3 modules, the n x n convolution kernel is split into two convolution kernels and that is 1 x n and n x 1, which is greatly optimized network parameters and outstandingly refined detection speed [33]. The last layer of these three modules is Filter concat layer which contains the LRN (Local Response Normalization) layer and the Depthconcat layer, Where the Depthconcat layer fuses the features pull out by convolutional layers. At last, InceptionV3 utilizes an average pooling layer and dropout layer before the fully connected layer and appends a softmax layer for further transmission to avoid the vanishing gradient problem [34]. All convolution layer is followed by the BN layer and ReLU non-linearity.

4.2.3 NasNetMobile

NasNetMobile is efficient CNN architecture, which restrains cardinal building blocks (cells) and is upgraded with aid of reinforcement training [35]. Blocks are just having some operations such as a few separable convolutions and bundles and it's replicated many times relying on necessary network capacity. The mobile version (NasnetMobile) contains 12 cells with a capacity of 5.3 million and a multiplying capacity of 564 million (MACS).

NasNetMobile is a pretty straightforward and tiny pre-trained model. The usual arrangement of the convolutionary network is preestablished in this method handoperated [36]. They consist of convolutional cells that are duplicated multiple times, and each cell has the same architecture but different weights.It is necessary to have two types of convolutional cells in order to provide two key roles if we take a map of a feature to directly construct scalable architectures for a picture of any scale. Such as input:

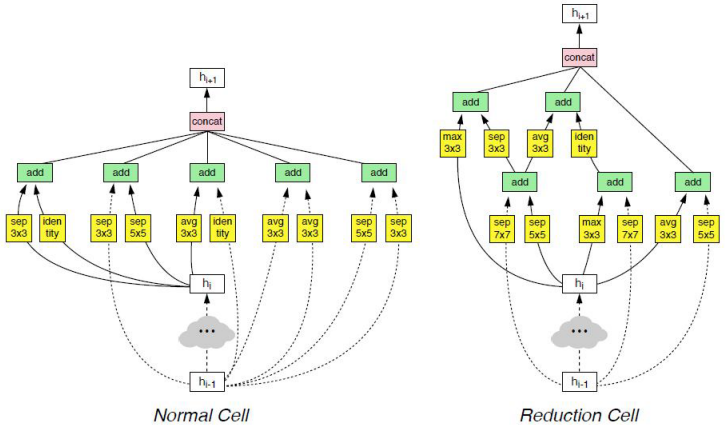

Figure 12: Nasnet Normal and Reduction Cell Architecture [37]

1. A convolution cell that returns a feature map of the same dimension
2. The convolution cell returns a feature map with the height and width of the feature map reduced by half.

Experts have suggested looking for components of the smaller dataset architecture and then moving the block to a larger dataset.Specifically, it first scans CIFAR-10 for the best co-evolution layer or cell, and then uses that cell to further stack a copy of that cell with ImageNet.A new regularization strategy ScheduledDropPath has been proposed, significantly increasing the widespread use of the NASNet model.

4.3 Plugins Used

There are many plugins used for the model in this implementation but some essential plugins are as follows:

4.3.0.1 NumPy

An array of multidimensional matrices that support high-level mathematical calculations.NumPy can use to perform operations on arrays included in mathematics such as algebra, statistics, and trigonometric patterns.The image is converted to matrix format. The matrix format of the image is used for interpretation and analysis using convolutional neural networks [38].During the image pre-processing stage, the image is magnified to 224 x 224 pixels. The image annotations were then changed to the NumPy array style. Finally, the exact specification of the image is included in the dataset.SciPy is also built on top of this. It works with NumPy arrays and provides the better execution needed for a variety of technical and logical applications.

4.3.0.2 Keras

Keras is an open source library for building deep learning applications.Written in Python, Keras provides an integrated interface to various deep learning backends

such as TensorFlow and Theano.Deep learning is a sub-genre of machine learning based on artificial neural networks.

4.3.0.3 Keras-Applications

The Keras application is a deep learning model provided with pre-trained weights.These models can be used for prediction, feature extraction, and fine-tuning.

Weights are automatically downloaded when the model is instantiated. They are stored under **~/ .keras / models /**.

4.3.0.4 Keras-Preprocessing

The Keras Preprocessing Layer API allows developers to create Keras native pipelines for input processing.These input processing pipelines can be used as independent preprocessing code in non-Keras workflows, combined directly with Keras models, and exported as part of KerasSavedModel.

Keras pre-processing layers allow you to create and export truly end-to-end models. A model that accepts raw images or raw structured data as input. A model that handles feature normalization or feature value indexing.

4.3.0.5 h5py

The h5py package is a Pythonic interface to the HDF5 binary data format. You can store large amounts of numerical data and easily manipulate that data from NumPy.

IT can be divide it into multi-terabyte datasets stored on disk, for example, as if it were a real NumPy array.

4.3.0.6 Matplotlib

Plot functions in the Python programming language are supported by Matplotlib. In this project, it will be used to draw a bounding box to indicate the image name and rating area. The bounding box is used to display the object detection name in the evaluation area of the image. Matplotlib provides an object-oriented application programming interface. Numpy is one of Matplotlib's mathematical extensions [4].

4.3.0.7 openCv-python

OpenCV is an open source computer vision library. It provides the machine with the ability to recognize faces and objects. In this tutorial, you will learn the concepts of OpenCV using the Python programming language.

4.3.0.8 pandas

pandas is a Python package that provides fast, flexible and expressive data structures that allow you to work with "relational" or "labeled" data easily and intuitively. It

aims to be a basic high-level building block for practical real-world data analysis in Python.

In addition, it has the broad goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

4.3.0.9 tensorboard

TensorBoard is a tool that provides the measurements and visualizations you need during your machine learning workflow. This allows you to track experimental indicators such as loss and accuracy, visualize model plots, and project embeddings in low-dimensional space.

4.3.0.9.1 tensorflow

TensorFlow Addons is a repository of contributions that adheres to established API patterns but implements new features not available in the TensorFlow core. TensorFlow natively supports a large number of operators, layers, metrics, losses and optimizers.However, in fast-moving areas like ML, there are many interesting new developments that cannot be integrated into the core of TensorFlow.

4.3.0.10 tensorflow-estimator

The Estimator has the following advantages:

- Estimator-based models can run in a local host or distributed multi-server environment without changing the model. In addition, you can run an Estimator-based model on the CPU, GPU, or TPU without having to recode the model.
- The Estimator provides a securely distributed training loop to control the following methods and timings:
    - Load data
    - Handle exceptions
    - Create checkpoint files and recover from failures
    - Save summaries for TensorBoard

While using Estimator to create application, It is necessary to separate the data entry pipeline from your model. This separation simplifies experimentation with different datasets.

5 Experimental Results

This section describes experiments run on pre-trained models like **MobileNetV2**, **InceptionV3**, and **NASNetMobile** to determine which model is the most accurate along with the dataset.

5.1 Experiment

To run the model GoogleColab is used. To compare all there three models initially downloaded the already trained Keras model.

While running this pre-trained InceptionV3 model, Total params were 24,480,040, among them, the Trainable params were 24,445,108 and the Non-trainable params were 34,932 there. The test set size was 376 which were belonging to 6 classes.

And accuracy of this model was 94.95 percent.

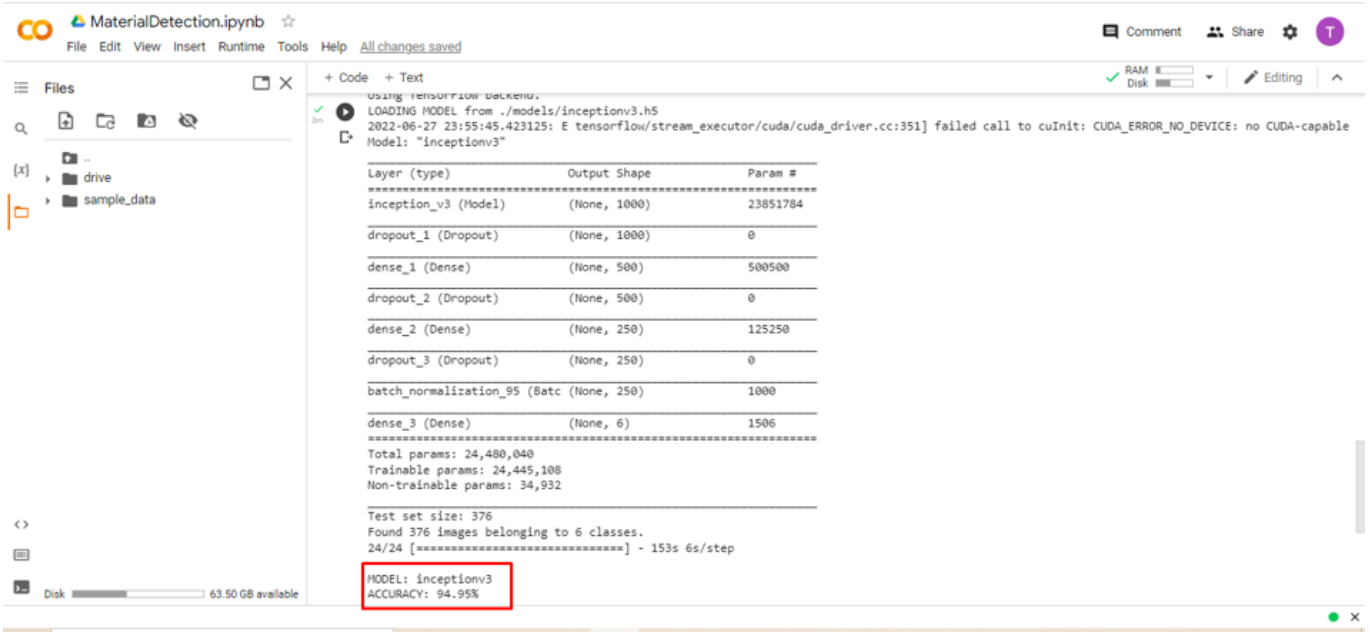

Figure 13: InceptionV3 model accuracy

While running this pre-trained MobileNetV2 model, Total params were 4,167,240, among them, Trainable params were 5,917,734 and Non-trainable params were 37,238 there. The test set size was 376 which were belonging to 6 classes.

And accuracy of this model was 93.62 percent.

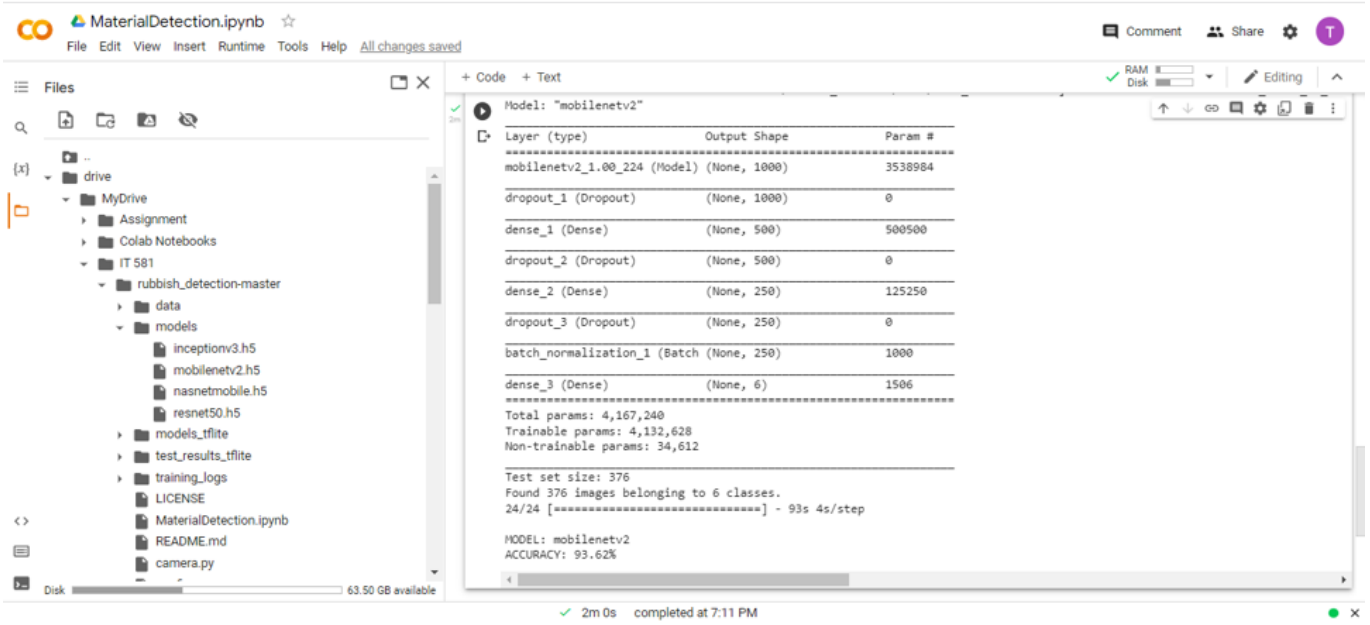

Figure 14: MobileNetV2 model accuracy

While running this pre-trained NASNetMobile model,Total params were 5,954,972, among them, Trainable params were 5,917,734 and Non-trainable params were 37,238 there. The test set size was 376 in pre-trained model. which were belonging to 6 classes.

And accuracy of this model was 93.62 percent.

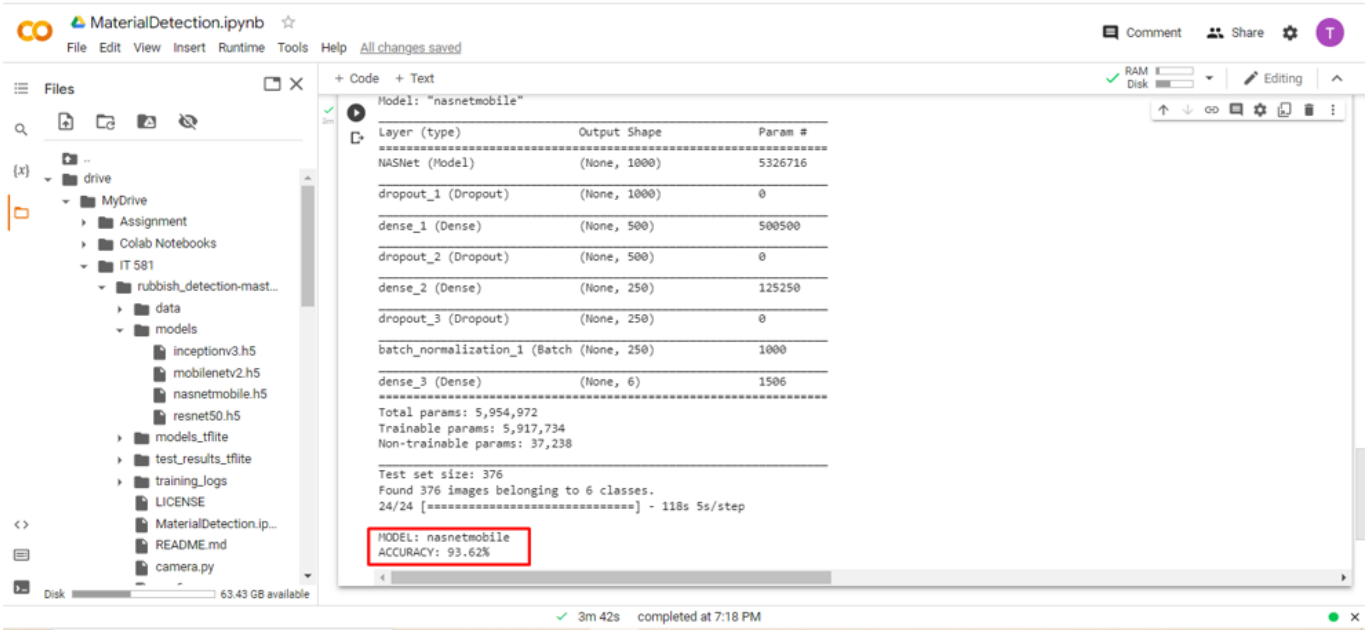

Figure 15: NASNetMobile model accuracy

By running this experiment on these three models it can be concluded that the inceptionV3 model is the most accurate to get a better result for material detection.

Table 1: Training results.

Keras model	Accuracy
InceptionV3	94.95%
MobileNetV2	92.55%
NASNetMobile	93.62%

After converting the model into tflite version it's optimized like the following:

Table 2: Optimization results.

TFLite Model	Accuracy
InceptionV3	94.95%
InceptionV3 float16 quantization	94.95%
InceptionV3 weights quantization	94.41%
InceptionV3 integer quantization	94.41%
MobileNetV2	92.55 %
MobileNetV2 float16 quantization	92.55%
MobileNetV2 weights quantization	34.04%
MobileNetV2 integer quantization	92.02%
NASNetMobile	93.62%
NASNetMobile float16 quantization	93.62%
NASNetMobile weights quantization	92.82%
NASNetMobile integer quantization	91.76%

5.2 Dataset

The dataset contains six classes: glass, paper, paperboard, plastic, metal, and garbage. Currently, the dataset consists of 691 images.

1. Test
   (a) 60 cardboard
   (b) 75 glasses
   (c) 61 metal
   (d) 6 papers
   (e) 21 plastic
   (f) 20 trash
2. Train
   (a) 5 cardboard
   (b) 6 glasses
   (c) 5 metal
   (d) 4 papers
   (e) 6 plastic
   (f) 42 trash

3. Validation
    (a) 61 cardboard
    (b) 76 glasses
    (c) 62 metal
    (d) 90 papers
    (e) 71 plastic
    (f) 20 trash

The image was taken with the object placed on a transparent white background and using sunlight or room lighting. With a regular camera on a mobile device. The pictures were also taken from different angles for better recognition. The size of the original dataset is 935MB.

6 Conclusions

To limit the impact of improper waste disposal, this project introduced a practical waste disposal method for homes and small businesses that uses deep learning algorithms and image processing techniques in addition to the integration of TensorFlow Lite.Therefore, the implementation framework worked with a large dataset of images, training algorithms, and prediction patterns for object detection and classification.

In this paper, we used models such as InceptionV3, MobileNetV2, NASNetMobile to classify waste into 6 categories (paperboard, metal, glass, paper, plastic, garbage) for various objects in the form of tensors.

Historically, most of the experiments related to this study were performed on specific models and included object detection using machine learning techniques such as SVM. However, this article focuses on comparing which model is more accurate and which model is for material detection of the detected object.

The methodology used in this paper helps reduce pollution levels and focuses on the further development of long-term universal waste management systems. Therefore, we can conclude that this project will be of great benefit to society.

The main issues with this project were accuracy and some limitations, like :- Detection works better with a clean background, proper lighting conditions, and a single object in front of the right angle.

Future work to consider is similar, but improve the dataset by capturing images of locally collected waste. You need to attach an image of the dirty and seemingly dirty waste in your training dataset.

This helps the model predict actual local waste. This mainly includes dirty household items. This will help improve the classification with higher accuracy. The dataset should also contain many images, including images of multiple wastes. This makes it easy to train the framework to predict multiple objects in a single image without causing errors in the discovery process.Further research on this project should also consider adding various other types of bulky waste categories to the dataset. Strengthening the list of categories will definitely help this framework develop further and improve proper waste management processes. You can also analyze and compare various models such as Faster R-CNN, SSD Mobile Net, and YOLO in the future.This analysis applies different classification algorithms for each model to this framework individually, identifying the most suitable model for final use, and making more accurate object detection and prediction in a short amount of time. You can do it with.

The future work also aims to implement this technology for all mobile platforms like, all mobile platforms such as Android, iOS and Chrome-OS, making it very easy for users to classify waste and dump it in the appropriate waste bins to protect their environment. I also aim to do it. Reduce pollution.

References

[1] Singh S, Mamatha KR, Ragothaman S, Raj KD, Anusha N, SusmiZacharia. Waste segregation system using artificial neural networks. HELIX. 2017 Jan 1;7(5):2053-8.

[2] Ranada, P. Why PH is the world's 3rd biggest dumper of plastics in the ocean. Rappler Blog. Retrieved October 6, 2015, 2015.

[3] Sousa, J.; Rebelo, A.; Cardoso, J.S. Automation of Waste Sorting with Deep Learning. 2019 XV Workshop de Visão Computacional (WVC). IEEE, 2019, pp. 43–48.

[4] Devi, R.S.; Vijaykumar, V.; Muthumeena, M. Waste Segregation using Deep Learning Algorithm.

[5] Bircanog˘ lu, C.; Atay, M.; Bes¸er, F.; Genç, Ö.; Kızrak, M.A. Recyclenet: Intelligent waste sorting using deep neural networks. 2018 Innovations in Intelligent Systems and Applications (INISTA). IEEE, 2018, pp. 1–7.

[6] Mitra, Arghadeep. Detection of Waste Materials Using Deep Learning and Image Processing. Diss. California State University San Marcos, 2020.

[7] Yang, M.; Thung, G. Classification of trash for recyclability status. CS229 Project Report 2016, 2016.

[8] Lowe, D.G. Distinctive image features from scale-invariant keypoints. International journal of computer vision 2004, 60, 91–110.

[9] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. Communications of the ACM 2017, 60, 84–90.

[10] Sakr, G.E.; Mokbel, M.; Darwich, A.; Khneisser, M.N.; Hadi, A. Comparing deep learning and support vector machines for autonomous waste sorting. 2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET). IEEE, 2016, pp. 207–212.

[11] Arebey, M.; Hannan, M.; Begum, R.; Basri, H. Solid waste bin level detection using gray level co-occurrence matrix feature extraction approach. Journal of environmental management 2012, 104, 9–18.

[12] Costa, B.S.; Bernardes, A.C.; Pereira, J.V.; Zampa, V.H.; Pereira, V.A.; Matos, G.F.; Soares, E.A.; Soares, C.L.; Silva, A.F. Artificial intelligence in automated sorting in trash recycling. Anais do XV Encontro Nacional de Inteligência Artificial e Computacional. SBC, 2018, pp.198-205.

[13] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 2014.

[14] Donovan, J. Auto-trash sorts garbage automatically at the techcrunch disrupt hackathon. Techcrunch Disrupt Hackaton, San Francisco, CA, USA, Tech. Rep. Disrupt SF 2016, 2016.

[15] Awe, O.; Mengistu, R.; Sreedhar, V. Smart trash net: Waste localization and classification. arXiv preprint 2017.

[16] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015, pp. 91–99.

[17] Dabholkar, A.; Muthiyan, B.; Srinivasan, S.; Ravi, S.; Jeon, H.; Gao, J. Smart illegal dumping detection. 2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, 2017, pp. 255–260.

[18] Zhao, Z.Q.; Zheng, P.; Xu, S.t.; Wu, X. Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems 2019, 30, 3212–3232.

[19] Çevik, E.; Zengin, K. Classification of Skin Lesions in Dermatoscopic Images with Deep Convolution Network. Avrupa Bilim ve Teknoloji Dergisi 2019, pp. 309–318.

[20] John, N.E.; Sreelakshmi, R.; Menon, S.R.; Santhosh, V. Artificial Neural Network based Intelligent Waste Segregator.

[21] Albeahdili, H.M.; Han, T.; Islam, N.E. Hybrid algorithm for the optimization of training convolutional neural network. International Journal of Advanced Computer Science and Applications 2015, 1, 79–85.

[22] Rosebrock, A. Histogram of Oriented Gradients and Object Detection. línea]. Disponible en:Blog(http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-objectdetection) 2014.

[23] Girshick, R. Fast r-cnn. Proceedings of the IEEE international conference on computer vision, 2015, pp.1440–1448.

[24] Bobulski, J.; Kubanek, M. Waste classification system using image processing and convolutional neural networks. International Work-Conference on Artificial Neural Networks. Springer, 2019, pp. 350–361.

[25] Rad, M.S.; von Kaenel, A.; Droux, A.; Tieche, F.; Ouerhani, N.; Ekenel, H.K.; Thiran, J.P. A computer vision system to localize and classify wastes on the streets. International Conference on computer vision systems. Springer, 2017, pp. 195–204.

[26] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. MobileNetV2: inverted residuals and linear bottlenecks. In: Proc. 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit, CVPR 2018. Salt Lake City, UT, United States, pp. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474.

[27] Yu, D., Xu, Q., Guo, H., Zhao, C., Lin, Y., Li, D., 2020. An efficient and lightweight convolutional neural network for remote sensing image scene classification. Sensors 20, 1999. https://doi.org/10.3390/s20071999.

[28] Buiu, C., Danaila, V., Raduta, C.N., 2020. MobileNetV2 ensemble for cervical precancerous lesions classification. Processes 8, 595. https://doi.org/10.3390/pr8050595.

[29] Liu, J., Wang, X., 2020. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. Plant Methods. 16, 83. https://doi.org/10.1186/s13007-020-00624-2.

[30] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: 26th Annu. Conf. Neural Inf. Process. Syst. 2012, NIPS 2012. Sydney, Australia, pp. 1097–1105. Retrieved from https://papers.nips. cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks. pdf.

[31] Zhou Z, Song Z, Fu L, Gao F, Li R, Cui Y. Real-time kiwifruit detection in orchard using deep learning on Android™ smartphones for yield estimation. Computers and Electronics in Agriculture. 2020 Dec 1;179:105856.

[32] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proc. 29th IEEE Conf. Comput. Vis. Pattern Recognit, CVPR 2016. Las Vegas, NV, USA, pp. 2818–2826. https://doi.org/ 10.1109/CVPR.2016.308.

[33] Shen, Y., Yin, Y., Zhao, C., Li, B., Wang, J., Li, G., Zhang, Z., 2019. Image recognition method based on an improved convolutional neural network to detect impurities in wheat. IEEE Access 7, 162206–162218. https://doi.org/10.1109/ ACCESS.2019.2946589.

[34] Zhuang, X., Zhang, T., 2019. Detection of sick broilers by digital image processing and deep learning. Biosyst. Eng. 179, 106–116. https://doi.org/10.1016/j.biosystemseng.2019.01.003.

[35] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria and J. Hemanth, "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2", Sustainable cities and society, pp. 102692, 2020.

[36] B. Zoph, V. Vasudevan, J. Shlens and Q. V. Le, "Learning transferable architectures for scalable image recognition", Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8697-8710, 2018.

[37] Addagarla SK, Chakravarthi GK, Anitha P. Real time multi-scale facial mask detection and classification using deep transfer learning techniques. International Journal. 2020 Jul;9(4):4402-8.

[38] Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; others. Speed/accuracy trade-offs for modern convolutional object detectors. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7310–7311.