TrashCan

Application to segregate Waste using Deep Learning

A MINOR PROJECT REPORT

Submitted By

Naman Jain (RA1911003010090)
Suvodeep Sinha (RA1911003010108)

Under the guidance of

Dr. S. Babu
Associate Professor

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M.Nagar, Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that 18CSP107L minor project report titled "**TrashCan-Application to segregate Waste using Deep Learning**" is the bonafide work of "**Naman Jain (RA1911003010090)**", "**Suvodeep Sinha (RA1911003010108)",** who carried out the minor project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. S. Babu
GUIDE
Associate Professor
Department of Computing Technologies

SIGNATURE

Dr. M. Pushpalatha, Ph.D.
HEAD OF THE DEPARTMENT
Professor
Department of Computing Technologies

Signature of the Panel Head
Dr. S. Babu
Associate Professor
Department of Computing Technologies

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to our guide, Dr. S. Babu his valuable guidance, consistent encouragement, personal caring, timely help, and for providing us with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to us for completing this project work.

Naman Jain
Suvodeep Sinha

ABSTRACT

Poor waste handling and increasing pollution have affected several countries in the environmental performance index 2021. With the emergence of several waste collection policies, there has been a surge in garbage production and collection. Even after such reforms, many individuals lack the ability to differentiate between different categories of waste. To manage a range of waste products, it's essential to have a smart waste management system. The process of separating the waste into its many components is one of the most crucial parts of waste management, and it is typically carried out manually by hand-picking. We provide an efficient waste material sorting system to streamline the process. segregate waste into 12 different categories using Deep Learning Models (Xception model in CNN) which is embedded into a mobile application. We use image classification techniques on a dataset of more than 15000 images of different types of waste to categorize the results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CNN	Convolutional Neural Network
SVM	Support Vector Machine
YOLO	You Only Look Once
VGG	Visual Geometry Group

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The rapidly growing population is to blame for the highest level of waste generation ever. In 2020, it was predicted that 2.29 billion tonnes of garbage would be produced, or 0.79 kg per person each day. By 2050, this amount is projected to rise by 73% to 3.88 billion tonnes as a result of population growth and increasing industrialization[1]. Due to the rise in trash generation, waste segregation has become a very laborious and challenging process. This process is typically carried out manually with the aid of human intelligence and physical work.

There are numerous types of waste, including paper, metal, plastic, food scraps, industrial waste, and others. Every piece of garbage needs to be accurately classified into a different types since each type requires a different processing method. Various categories, including dry and wet garbage, recyclable and non-recyclable waste, biodegradable and non-biodegradable waste, etc., are used by modern systems to classify rubbish. Although some level of segregation is permitted under this classification, it is insufficient to properly handle the rubbish.

Even though the public is becoming more aware of the necessity to separate waste, some items can be confusing during the process and result in inaccurate classification[2]. Since wrongly segregated waste cannot be recycled or disposed of using techniques that work for other types of rubbish, faulty waste segregation is another major issue. This could cause a large financial loss because the recyclable parts of the poorly divided rubbish cannot be recovered. Incorrectly categorized trash at landfills can have an effect on both the natural world and people's lives. Landfills produce greenhouse gasses that are bad for the environment[3].

1.2 Objectives and Proposed Work

For a long time, waste segregation and management have had a significant impact on a major fraction of the ecosystem. Waste management will be facilitated when technology is used correctly. There are various works and references where the above problem has been considered. However, no matter what techniques were added, it was tough to break through the 80% threshold with the scratch model. This finding prompts us to suggest an InceptionNet-inspired deep convolutional neural network design, in which depth-separable convolutions are used in place of the InceptionNet's Inception modules. (XceptionNet)[4] that divides the waste into 12 different kinds using a dataset of more than 15000 photos and a mobile application.

For the task of classifying our garbage into categories, we propose to use the Xception Architecture by Google Research[4]

1. XceptionNet

Convolution layers that are distinct from one another in terms of their depth are crucial components of the Xception concept. It is feasible to draw a distinct line within this architecture between the mapping of cross-channel connections and spatial connections in convolutional neural network property maps. This distinction can be made with regard to convolutional neural network property maps. The property extraction basis of the network of Xception, which is a more powerful version of the fundamental architecture of Inception, is made up of thirty-six convolutional layers. Xception is a more recent advancement in the field of artificial intelligence. Except for the very first and very last modules, each convolutional layer comprises a total of fourteen individual modules. The exceptions to this rule are the very first and very last modules. Each module is surrounded on all sides by corresponding linear remnants that run in a circle around it. Because it is a linear heap of depth-wise separable convolution layers with residual relations, this architecture is especially simple to explain and alter [4]. This is owing to the fact that it has residual relations.

1.3 Software Requirements Specification

1. User Interfaces
    a. Frontend Software - Flutter Framework
    b. Backend Software - Kaggle (Dataset and Train), Tensorflow

2. Hardware Interfaces
    a. Android Operating System
    b. Support for Internet Connectivity
    c. Support for Camera

3. Software Interface
    a. Operating System - It works on Android-based devices with a future possibility to run on iOS devices as well.
    b. Connectivity - It requires an internet connection for URL-based features in the mobile application but doesn't require the same for the main image classification.
    c. Camera - It needs access to the camera and device image storage to capture/upload it to the application.

4. Communication Interface

    It supports all types of Android-based mobile devices having a camera built and an android version of a minimum of 4 and above.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Review

We looked at a lot of different works and put together a complete summary of all the research that had been done on the subject. We looked at surveys, academic articles, books, and other sources that were relevant to our topic. The above review enumerates, evaluates objectively, and clarifies this prior research. It provides a theoretical foundation for the research and aided us in determining the nature of our investigation. We acknowledge the work of previous researchers, ensuring that our work is well-conceived by reading, evaluating, and incorporating their work into our own. Our goal is to communicate the established knowledge and ideas on a topic, as well as their strengths and weaknesses.

This literature review is intended to provide a critical overview of the present state of research on a certain issue, and it addresses the following:

- Identifies prior research areas.
- Puts each source within the context of its contribution.
- Describes the relationship between each source and the others we have chosen.
- Identifies novel ways to interpret prior research and sheds light on any omissions.

The 2016 TechCrunch Disrupt Hackathon entry "Auto-Trash" has the most similarities to this study [5]. The idea was an auto-sorting trash can with a camera and Raspberry Pi-powered module that could distinguish between compost and recycling. Utilizing Google's TensorFlow, Auto-Trash was expertly built [5]. Their system's flaw was that it could only distinguish between compost materials. When studying the mobile application for android and ios devices, we discovered a project that is similar. This study's objective was to approximately categorize large heaps of trash shown in a picture. The main aim of the mobile application is to allow users to track down and report trash they notice in their neighborhood. The dataset was built by using Bing Image Search, and they trained their network using the photos that were returned [6].

In the paper [7], four distinct measures are used to compare a suggested model called WasteNet to other models including VGG, AlexNet, ResNet, DenseNet, and SqueezeNet. Six separate categories have been used to classify the garbage. The proposed model was found to have a 97% accuracy and to perform better on all four measures. As a result, it was used to create a smart trash can that sorts the trash that is placed in it. The research [8] suggests a neural network with an accuracy of 81.22% for dividing trash into the three categories of organic waste, recyclables, and non-recyclables. The paper also compares other models including VGG16, Inception-Net, Dense-Net, and Mobile-Net. Mobile-Net had the highest accuracy of all the transfer learning models compared to before, with a score of 92.65%.

In the Paper [9], a Convolutional Neural Network and a Support Vector Machine technique are recommended for better garbage classification. Due to the short dataset, they utilized ResNet-50 as the transfer learning model. Glass, paper, plastic, and paper are the four categories into which garbage is separated. They obtained an accuracy of 87% using this method for Epoch 12. In this article, the effectiveness of four CNN-based trash classifiers, namely ResNet-50, VGG16, MobileNet V2, and DenseNet-121, is assessed [10]. The trash was separated into four categories: hazardous waste, ordinary waste, recyclable waste, and compostable trash. Hazardous waste was the first category, followed by ordinary waste, recyclable waste, and compostable trash. ResNet-50 achieved the highest level of accuracy, which was 94.8 percent.

The paper [11] investigates a novel method for waste sorting for effective recycling and disposal using a deep learning algorithm. A self-created dataset was trained in the Darknet framework using the YOLOv3 algorithm. Six separate types of object recognition have been taught to the network. Additionally, YOLOv3-tiny was used for the detection test in order to compare results and assess the effectiveness of the YOLOv3 algorithm.

A garbage container that uses the technological advancements of IoT with devices and machine learning algorithms to automatically sort rubbish is a crucial part of the system described in [28]. Due to the bin's connection to the cloud, various data points for each bin are recorded and uploaded, which helps with waste collection. Two different iterations of the system are discussed in this paper. The first one correctly categorizes trash as either wet or dry with a 75 percent accuracy rate, while the second one

correctly divides trash into six different categories with a 90 percent accuracy rate. Waste was divided by the report's authors [29] into four independent categories: cardboard, paper, metal, and plastic. For photos with a resolution of 50 by 50 pixels, they developed a convolutional neural network that, after 100 iterations, had the best testing accuracy of 76%.

2.2 Comparison of Existing Methods

We have compared a number of works containing topics or concepts relevant to our project. It was helpful to plan, develop, and compose a comparative analysis of the entire corpus by focusing on different aspects of these papers.

Approach and Algorithm	**Accuracy**	**Notable Achievements**	**Remarks**
Comparison between VGG 16, ResNet, AlexNet[3]	94.9 % (ResNet-50)	25,077 images were used in the dataset Compilation of various techniques over the years	Segregation based on only 2 categories - Organic and Recyclable
CNN with last 2 layers replaced by a classifier (softmax or SVM)[23]	79.94 % (AlexNet with SVM as the last layer)	Clear and simple implementation of only 2 methods Less computation power required	Categories are limited to 6 Sample size of the dataset is only 2527
InceptionNet Neural Network Architecture using a Real-time embedded system[24]	96.2 % (Inception Net)	Implemented hardware solution consisting of Raspberry Pi, IR sensor and PI Camera	Only 6 classes are used to categorize garbage

Zero-Shot Learning and Object Detection Algorithms[25]	83.60 % (YOLOv3)	Unique approach to task Algorithm mentioned clearly with plans of proposed system	Focus on Object Detection rather than classification Trained on COCO Dataset
Improvement on InceptionNet[4]	------	On a larger image classification dataset with 350 million images and 17,000 classes, XceptionNet performs much better than the traditional Inception Net	Better use of model parameters, not more capacity, is what led to the performance gains.
SVM with SIFT features CNN with Torch7 framework[27]	63 % (SVM)	Use of a simpler algorithm as SVM Implemented an eleven layer CNN that is very similar to AlexNet	Small dataset of 2527 images DIfferentiated on the basis of 6 classes
Inception Resnet V2, Xception architectures[26]	94 % (Inception Resnet V2)	Adam and Adadelta were used as the optimizer in neural network models	Limited samples of the Trashnet dataset

Table 2.1 Comparison of Existing Approaches

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Diagrams

A good diagram provides a concise overview of a system. We can see at a glance which building blocks are being implemented, how they are connected, and how data flows among them. It will be useful in a variety of situations and promote a unified understanding of a system. It can be used to evaluate the system's design.

We have made an architectural and flow diagram to represent the process and structure of the system. It is to demonstrate how objects interact with one another and in what order. It is essential to note that they depict interactions for a specific scenario. The processes are depicted in a rectangular box with arrows representing their interactions.

3.1.1 ARCHITECTURE DIAGRAM

An architectural diagram is a picture that shows how the parts of a software system are physically put together. It shows how the software system is put together as a whole, as well as the connections, limits, and boundaries between each part. The diagram provides an overview of the system, allowing users to comprehend how the system's various components interact and to assess the impact of updates and new features.

A simple architecture diagram assists system designers and developers in visualizing the high-level structure of their system or application in order to ensure that it meets the needs of the users. It can also be used to describe the patterns implemented throughout the design. Similar to a blueprint, we used our simple architecture diagram as a guide to discuss and enhance our product with ease.

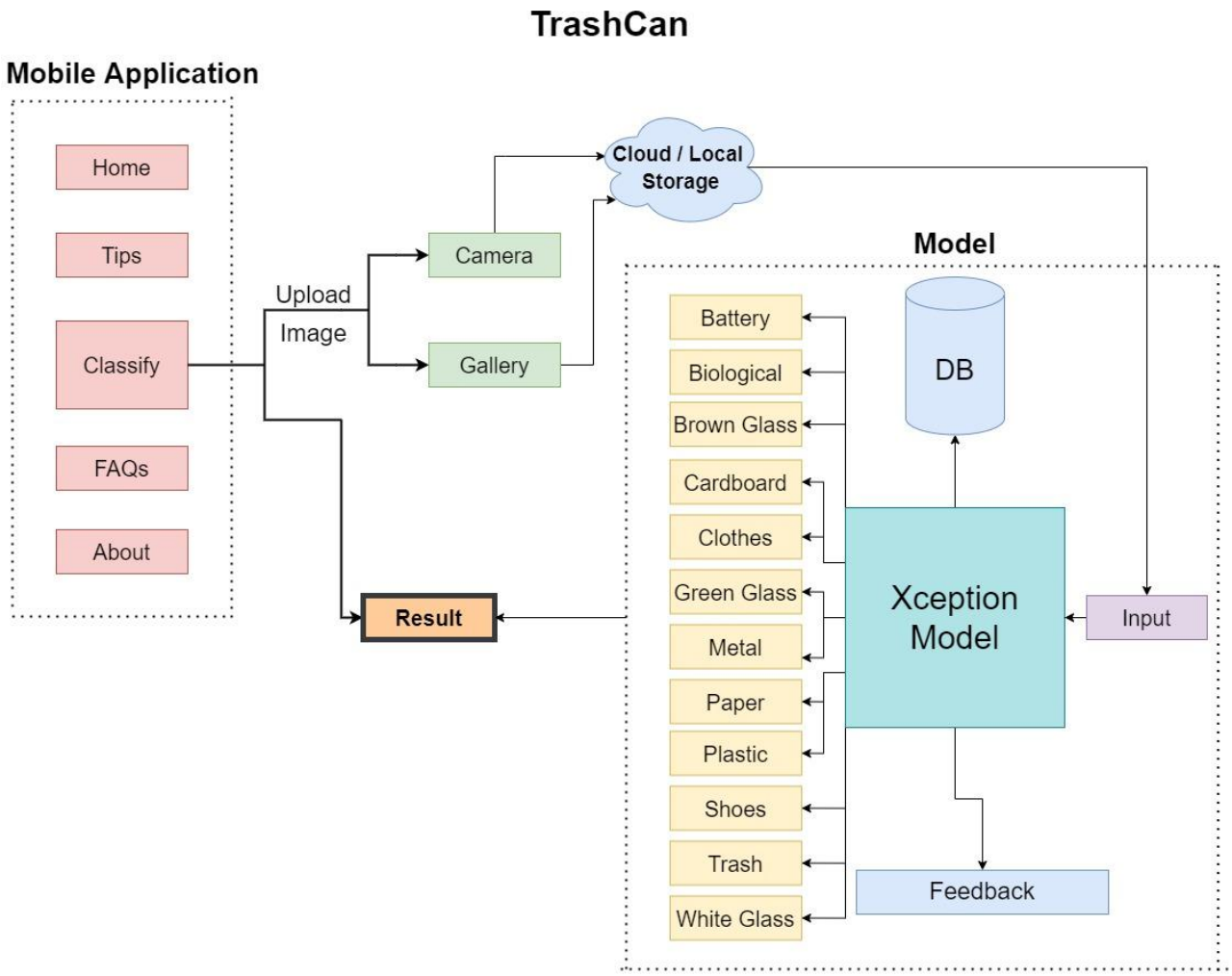


Figure 3.1 Architecture Diagram

3.1.2 FLOW DIAGRAM

A flow diagram is a visual representation of a series of actions, system movements, and/or decision points. They are a comprehensive explanation of each step in a procedure, regardless of its complexity. Flow diagrams, which are also called flowcharts, are useful tools for improving how people, things, or information move through a system or process. The meaning of a flow diagram is based on how the diagram's lines and symbols work together to show the direction of a movement and what needs to be done to make it happen.

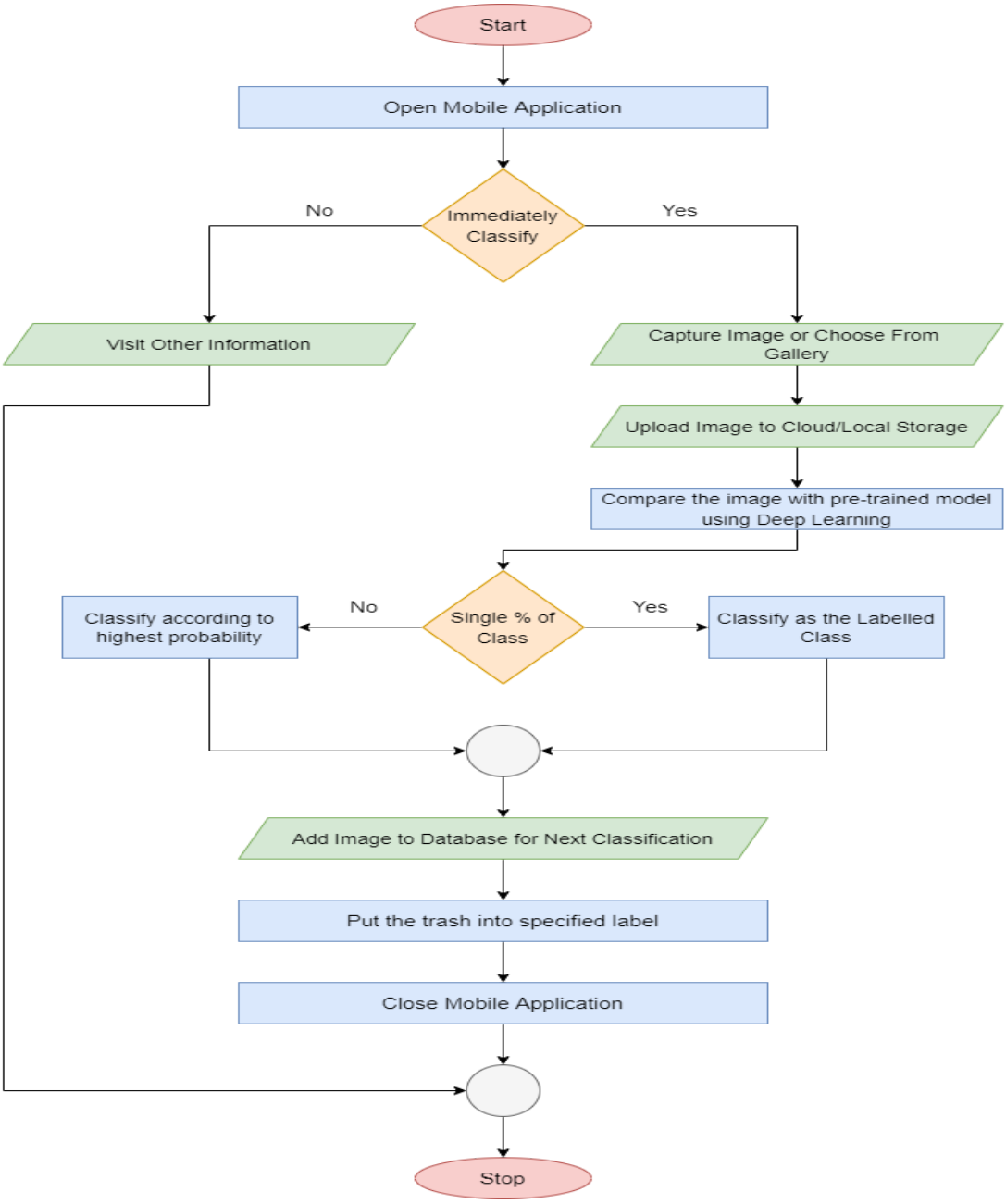


Figure 3.2 Flow diagram

3.2 Design Modules

Various parameters influence the model's performance. A model is deemed effective if it achieves high accuracy with production or test data and can predict effectively unknown data. If it is easy to implement in production and scalable. The machine learning model's parameters determine how to input data is transformed into the desired output, while the model's hyperparameters determine its shape. Before a model can be trained, hyperparameter attributes must be specified for the vast majority of standard learning methods.

```
# Increasing the image size didn't result in increasing the training accuracy
IMAGE_WIDTH = 320
IMAGE_HEIGHT = 320
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS = 3


# Path where our data is located
base_path = "../input/garbage-classification/garbage_classification/"

# Dictionary to save our 12 classes
categories = {0: 'paper', 1: 'cardboard', 2: 'plastic', 3: 'metal', 4: 'trash', 5: 'battery',
              6: 'shoes', 7: 'clothes', 8: 'green-glass', 9: 'brown-glass', 10: 'white-glass',
              11: 'biological'}

print('defining constants successful!')
```

Merging multiple datasets create redundancies and duplicates in the data, which must then be eliminated. Moreover, incorrect and inadequately collected datasets frequently result in models learning inaccurate representations of the data, thereby diminishing their decision-making capabilities. Modeling noise in the data must be avoided in order to create a model that generates trustworthy future insights, as it reduces model precision. Unstructured and poorly formatted data cannot be sorted efficiently by computers. It transforms the data into a format that can be efficiently and effectively processed by data mining, machine learning, and other data science tasks. Typically, the techniques are implemented at the earliest stages of the development pipeline for machine learning and artificial intelligence to ensure precise results.

```
# list conatining all the filenames in the dataset
filenames_list = []
# list to store the corresponding category, note that each folder of the dataset has one class of data
categories_list = []

for category in categories:
    filenames = os.listdir(base_path + categories[category])

    filenames_list = filenames_list  +filenames
    categories_list = categories_list + [category] * len(filenames)

df = pd.DataFrame({
    'filename': filenames_list,
    'category': categories_list
})

df = add_class_name_prefix(df, 'filename')

# Shuffle the dataframe
df = df.sample(frac=1).reset_index(drop=True)

print('number of elements = ' , len(df))
```

Data visualization is a technique that employs a variety of static and dynamic visuals within a specific context to assist individuals in comprehending and making sense of vast amounts of data. Frequently, the data is presented in a narrative format that highlights patterns, trends, and correlations that might otherwise go unnoticed. Data as a product is frequently monetized through the use of data visualization. Algorithms learn and automate data recognition and pattern identification, resulting in predictive analysis, through machine learning. Machine learning saves time and improves data quality because it employs algorithms to acknowledge and store data patterns. It's not just about tracking the user's data; it's also about understanding the user on a more granular level and how they will engage most effectively. It engenders a higher level of confidence among decision-makers and stakeholders.

```
# see sample image, you can run the same cell again to get a different image
random_row = random.randint(0, len(df)-1)
sample = df.iloc[random_row]
randomimage = image.load_img(base_path +sample['filename'])
print(sample['filename'])
plt.imshow(randomimage)
```

```
clothes/clothes4042.jpg
<matplotlib.image.AxesImage at 0x7f203ec3cb90>
```

CHAPTER 4

METHODOLOGY

4.1 Algorithm and Method

Because of the varying orientations of the waste items, the approach of augmenting the data given was applied to the photos in the data pre-processing stage. A few of the methods include selecting random images, translating the images, scaling the images arbitrarily, and shearing the images. The dataset size is maximized using this method. For the task of classifying our garbage into categories, we propose to use the Xception Architecture by Google .

1. XceptionNet

Inception inspired the architecture of a deep convolutional neural network. In this architecture, the Inception modules have been replaced by depth-wise separable convolutions. We show that our architecture, dubbed Xception, outperforms Inception V3 in some ways. Because the Xception design has the same number of parameters as the Inception V3 architecture, performance gains are achieved not through increased capacity but rather through more efficient parameter use.

Since a convolution layer seeks to learn filters in a three-dimensional environment with two spatial dimensions—the height and width, respectively—and a channel dimension, a single convolution kernel is responsible for simultaneously mapping cross-channel and spatial correlations. This is due to the fact that a convolution layer contains a channel dimension.

The purpose of the Inception module is to explicitly separate this process into a number of operations that would each independently evaluate cross-channel correlations and spatial correlations in order to make it easier to use and more efficient. To be more specific, the typical Inception module splits the input data into three or four smaller 3D spaces than the original input space, analyzes cross-channel correlations with a set of 1x1 convolutions, and then maps all correlations in these smaller 3D spaces

using conventional 3x3 or 5x5 convolutions. The foundation of Xception is a set of depth-aware separable convolutional layers.

In summary, what we are saying is that the cross-channel mapping and the spatial correlation mapping in the feature maps of convolutional neural networks can be totally dissociated from one another.

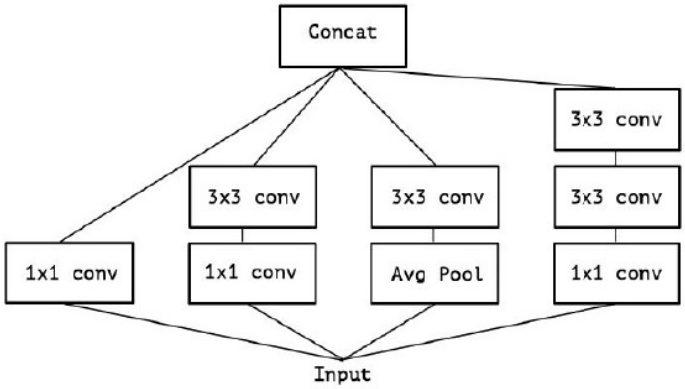


Figure 4.1 Inception V3 Module - Canonical

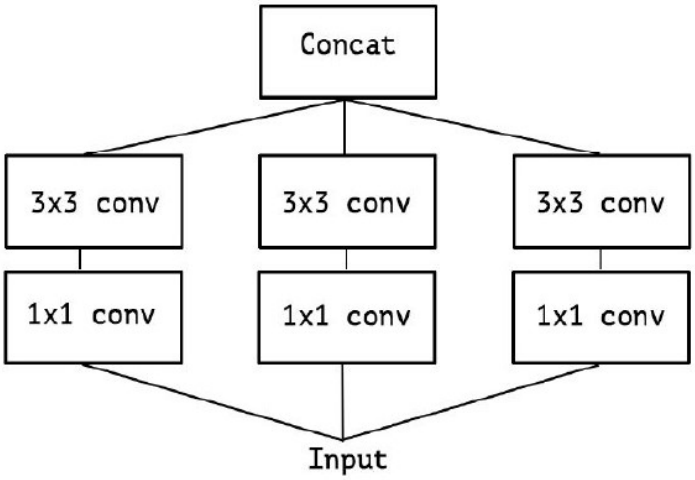


Figure 4.2 Inception Module - Simple

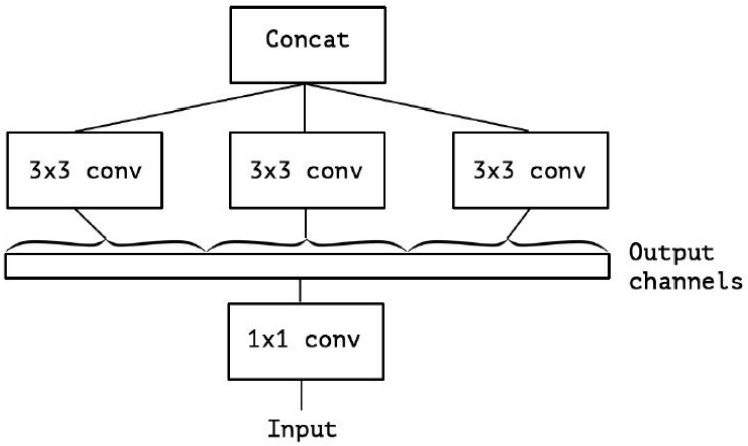


Figure 4.3 A formulation that is identical to the InceptionNet- Simple

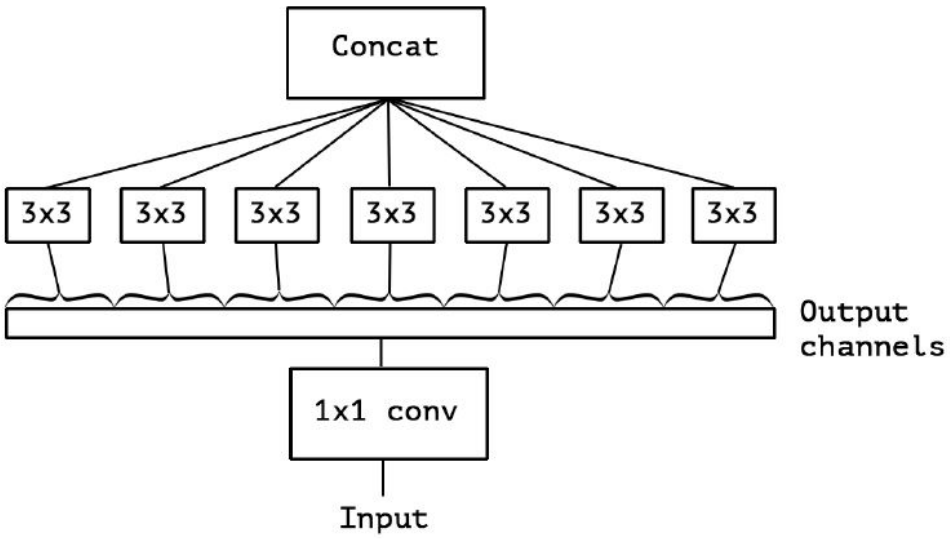


Figure 4.4 An "extreme" variant of our Inception Net

The data first crosses the input flow, then the duplicated eight-times center flow, and finally the exit flow. Noting that each Convolution and Separable Convolution layer is followed by batch normalization. Each layer of Separable Convolution has a depth multiplicator of 1.

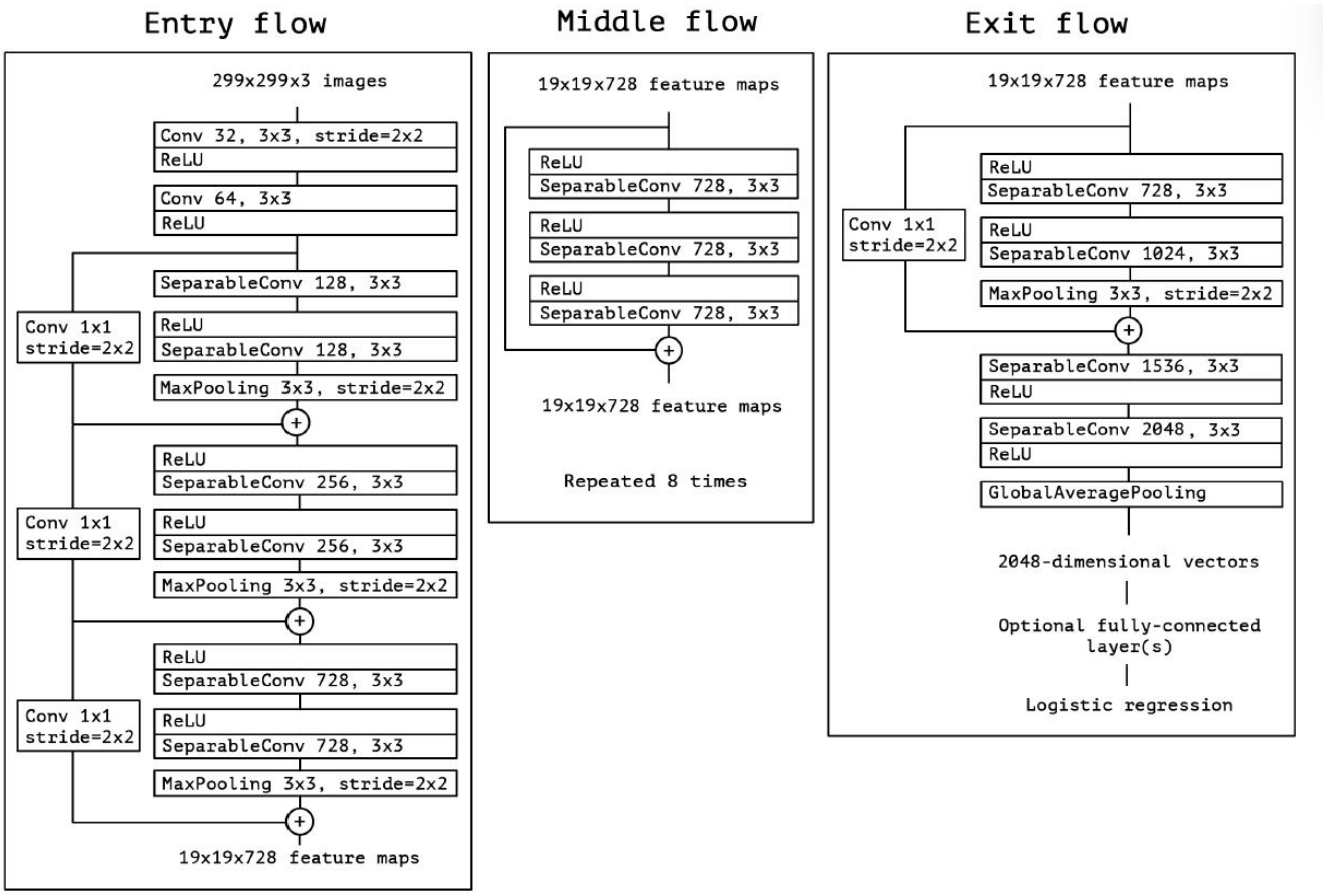


Figure 4.5 The Xception architecture

4.2 Implementation

The mobile application is a portable and simple method for evaluating the efficacy of the strategy and enabling its deployment in the real world. The application has a user-friendly design and other features that can be used to boost one's understanding of trash segregation. The program is developed using Flutter[15], making it compatible with both Android and iOS devices.

The application consists primarily of a bottom navigation bar that guides the user to four primary pages with distinct functionality. These are the available tabs:

- Home
- Tips
- Classify

- FAQs

A. Home

The home page is the page that loads when the application is launched. This page provides the user with vital information on waste segregation, educating them on the significance of waste segregation and the advantages of good waste segregation.

B.Tips

This page provides information on the various sorts of waste that exist, as well as an explanation of the various methods by which one can dispose of the aforementioned waste categories. It includes information that is particular to the five different forms of waste, which are dry waste, wet waste, hazardous waste, electronic waste, and biological waste.

C.Classify

This is the most important page on the application as it performs the function of getting the image that has to be classified and runs it onto the model proposed above and displays the results to the user. The image that has to be sent can either be clicked directly within the app or any image present on the device can also be used for this purpose. The app runs a local Tflite file in the backend to produce the results and reduce the time and resources that are required by cloud-based solutions.

D.FAQs

This page consists of the common questions that users of the application may have while segregating waste on their own. This page has in-depth explanations of some of the questions that might be asked by a user that wants to know more about waste management.

The application makes use of various flutter plugins that aid in the process of working the app by performing important functions that otherwise would have been very difficult to perform.

The plugins are as follows:

- [16]google_fonts: ^3.0.1
- [17]just_audio: ^0.9.30
- [18]url_launcher: ^6.1.6
- [19]carousel_slider: ^4.1.1
- [20]image_picker: ^0.8.6
- [21]flutter_tflite: ^1.0.1
- [22]font_awesome_flutter: ^10.2.1

CHAPTER 5

CODING AND TESTING

1. Mobile Application Code - Main

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:trash/waste.dart';
import 'about.dart';
import 'detect.dart';
import 'faq.dart';
import 'home.dart';

void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {

// This widget is the root of your application.
@override
Widget build(BuildContext context) {
  return MaterialApp(
    debugShowCheckedModeBanner: false,
    title: 'Trash Can',
    theme: ThemeData(),
    home: Nav(),
  );
}
}

class Nav extends StatefulWidget {
  @override
```

```
  _NavState createState() => _NavState();
}

class _NavState extends State<Nav> {
  int currentIndex = 0;
  String currentTitle='';

  setBottomBarIndex(index) {
    setState(() {
      currentIndex = index;
      if(index==0)
        currentTitle="Home";
      else if(index==1)
        currentTitle="Classify";
      else
          currentTitle='About';

    });
  }

  @override
  Widget build(BuildContext context) {
    final Size size = MediaQuery.of(context).size;
    final  List<Widget> _pages = <Widget>[
      HomePage(),
      Waste(),
      Predict(),
      FAQs(),
      About(),
    ];
    return Scaffold(
      backgroundColor: Color(0xFFFFFDD0),
      body: Stack(
        children: [
          Center(
            child: _pages[currentIndex],
          ),
          Positioned(
```

```
            bottom: 0,
            left: 0,
            child: Container(
              width: size.width,
              height: 80,
              child: Stack(
                clipBehavior: Clip.none,
                children: [
                  CustomPaint(
                    size: Size(size.width, 80),
                    painter: BNBCustomPainter(),
                  ),
                  Column(
                    children: [
                      Center(
                        heightFactor: 0.6,
                        child: FloatingActionButton(
backgroundColor:Colors.black.withOpacity(0.7),child: Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Image.asset('assets/logo.png',fit:
BoxFit.fill,),
                        ) ,onPressed: () {
                          setBottomBarIndex(2);
                        },),
                      ),
                      SizedBox(height: 26,),
                      Text('Classify',style: GoogleFonts.getFont('Didact
Gothic',color:Colors.black,fontWeight: FontWeight.bold,fontSize: 14),),
                    ],
                  ),
                  Container(
                    width: size.width,
                    height: 80,
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                      children: [
                        Column(
                          mainAxisAlignment:MainAxisAlignment.center,
```

```
                        children: [
                          IconButton(
                            icon: Icon(
                              FontAwesomeIcons.house,
                              color: currentIndex == 0 ? Colors.grey :
Colors.black,
                              size: 25,
                            ),
                            onPressed: () {
                              setBottomBarIndex(0);
                            },
                            splashColor: Colors.white,
                          ),
                          Text('Home',style: GoogleFonts.getFont('Didact
Gothic',color: currentIndex == 0 ? Colors.grey : Colors.black,fontWeight:
FontWeight.bold),),
                        ],
                      ),
                      Column(
                        mainAxisAlignment:MainAxisAlignment.center,
                        children: [
                          IconButton(
                            icon: Icon(
                              FontAwesomeIcons.glassWater,
                              color: currentIndex == 1 ? Colors.grey :
Colors.black,
                              size: 25,
                            ),
                            onPressed: () {
                              setBottomBarIndex(1);
                            },
                            splashColor: Colors.white,
                          ),
                          Text('Tips',style: GoogleFonts.getFont('Didact
Gothic',color: currentIndex == 1 ? Colors.grey : Colors.black,fontWeight:
FontWeight.bold),),
                        ],
                      ),
```

```
                        Container(
                          width: size.width * 0.2,
                        ),
                        Column(
                          mainAxisAlignment: MainAxisAlignment.center,
                          children: [
                            IconButton(
                                icon: Icon(
                                  FontAwesomeIcons.circleInfo,
                                  color: currentIndex == 3 ? Colors.grey :
Colors.black,
                                  size: 25,
                                ),
                                onPressed: () {
                                  setBottomBarIndex(3);
                                }),
                            Text('  FAQs',style: GoogleFonts.getFont('Didact
Gothic',color: currentIndex == 3 ? Colors.grey : Colors.black,fontWeight:
FontWeight.bold),),
                          ],
                        ),
                        Column(
                          mainAxisAlignment: MainAxisAlignment.center,
                          children: [
                            IconButton(
                                icon: Icon(
                                  FontAwesomeIcons.peopleGroup,
                                  color: currentIndex == 4 ? Colors.grey :
Colors.black,
                                  size: 25,
                                ),
                                onPressed: () {
                                  setBottomBarIndex(4);
                                }),
                            Text('  About',style: GoogleFonts.getFont('Didact
Gothic',color: currentIndex == 4 ? Colors.grey : Colors.black ,fontWeight:
FontWeight.bold),),
                          ],
```

```
                        ),
                      ],
                    ),
                  )
                ],
              ),
            ),
          )
        ],
      ),
    );
  }
}

class BNBCustomPainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    Paint paint = new Paint()
      ..color = Color(0xFFF7CC00)
      ..style = PaintingStyle.fill;

    Path path = Path();
    path.moveTo(0, 20); // Start
    path.quadraticBezierTo(size.width * 0.20, 0, size.width * 0.35, 0);
    path.quadraticBezierTo(size.width * 0.40, 0, size.width * 0.40, 20);
    path.arcToPoint(Offset(size.width * 0.60, 20), radius:
Radius.circular(20.0), clockwise: false);
    path.quadraticBezierTo(size.width * 0.60, 0, size.width * 0.65, 0);
    path.quadraticBezierTo(size.width * 0.80, 0, size.width, 20);
    path.lineTo(size.width, size.height);
    path.lineTo(0, size.height);
    path.lineTo(0, 20);
    canvas.drawShadow(path, Colors.black, 5, true);
    canvas.drawPath(path, paint);
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
```

```
    return false;
  }
}
```

2. Mobile Application Code - Detection

```
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:flutter_tflite/flutter_tflite.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:image_picker/image_picker.dart';
import 'package:just_audio/just_audio.dart';
import 'package:trash/wasteinfo.dart';

class Predict extends StatefulWidget {
  const Predict({Key? key}) : super(key: key);
  @override
  _PredictState createState() => _PredictState();
}
class _PredictState extends State<Predict> {
  String x='';
  late int y;
  bool loading = false;
  late File _image;
  var _output ;
  final imagepicker = ImagePicker();

  @override
  void initState() {
    super.initState();
    loading= true;
    loadmodel().then((value) {
      print(value);
      setState(() {});
    });
  }
```

```
detectimage(File image) async {
  var prediction = await Tflite.runModelOnImage(
      path: image.path,
      numResults: 12);
  setState(() {
    _output = prediction;
    print(_output);
    loading = false;
  });
}
  loadmodel() async {
  await Tflite.loadModel(
    model: 'assets/unquantized.tflite',
    labels: 'assets/labels.txt',
   ) ;
}
@override
void dispose() {
  Tflite.close();
  super.dispose();
}
pickimage_camera() async {
  var image = await imagepicker.getImage(source: ImageSource.camera);
  if (image == null) {
    return null;
  } else {
    _image = File(image.path);
  }
  detectimage(_image);
}
pickimage_gallery() async {
  var image = await imagepicker.getImage(source: ImageSource.gallery);
  if (image == null) {
    return null;
  } else {
    _image = File(image.path);
  }
  detectimage(_image);
```

```
  }
  @override
  Widget build(BuildContext context) {
    var h = MediaQuery.of(context).size.height;
    var w = MediaQuery.of(context).size.width;
    return Scaffold(
      backgroundColor: Color(0xFFFFFDD0),
      body: Container(
        height: h,
        width: w,
        child: Column(
          children: [
            SizedBox(
              height: 90,
            ),
            Container(
                child: Text('Garbage Segregator',
                    style: GoogleFonts.getFont('Didact
Gothic',color:Colors.black,fontWeight: FontWeight.bold,fontSize: 30),)),
            SizedBox(
              height: 20,
            ),
            Container(
              height: 150,
              width: 150,
              decoration: BoxDecoration(
                color: Colors.black,
                borderRadius: BorderRadius.circular(15),
              ),
              //color: Colors.black,
              padding: EdgeInsets.all(10),
              child: Image.asset('assets/logo.png'),
            ),
            SizedBox(height: 30),
            Container(
              child: Row(
                mainAxisAlignment: MainAxisAlignment.center,
```

```
                children: [
                  Container(
                    padding: EdgeInsets.only(left: 10, right: 10),
                    height: 60,
                    width: 150,
                    child: ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.black,
                          shape: RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(10)),
                        ),
                        child: Text('Capture',
                            style: GoogleFonts.getFont('Didact
Gothic',color:Colors.white,fontWeight: FontWeight.bold,fontSize: 24),),
                        onPressed: () async{
                          pickimage_camera();
                        }),
                  ),
                  SizedBox(height: 10),
                  Container(
                    padding: EdgeInsets.only(left: 10, right: 10),
                    height: 60,
                    width: 150,
                    child: ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.black,
                          shape: RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(10)),
                        ),
                        child: Text('Gallery',
                            style:  GoogleFonts.getFont('Didact
Gothic',color:Colors.white,fontWeight: FontWeight.bold,fontSize: 24),),
                        onPressed: () async{
                          pickimage_gallery();
                        }),
                  ),
                ],
              ),
```

```
            ),
            SizedBox(
              height: 20,
            ),
            loading != true
                ? Container(
              child: Column(
                children: [
                  Container(
                    height: 200,
                    width: 200,
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(15),
                      color: Color(0xFFF7CC00),
                    ),
                    // width: double.infinity,
                    padding: EdgeInsets.all(15),
                    child: Image.file(_image,fit: BoxFit.fitWidth,),
                  ),
                  SizedBox(
                    height: 10,
                  ),
                  _output != null
                      ? Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          IconButton(onPressed: () async{
                            final player = AudioPlayer();
                            await
player.setUrl('asset:///assets/audio${_output[0]['index']}.mp3');
                            player.play();
                          }, icon: Icon(Icons.volume_up,size: 30,)),
                          Text(
                          'Classified as : ${_output[0]['label'].toString()}',
                          style: GoogleFonts.getFont('Didact
Gothic',color:Colors.black,fontWeight: FontWeight.bold,fontSize: 22),),
                        ],
                      )
```

```
                        : Text(''),
                    Padding(
                      padding: const EdgeInsets.all(20.0),
                      child: GestureDetector(
                        onTap:(){ Navigator.push(context,
MaterialPageRoute(builder: (context)=>Info(_output[0]['index'])));},
                        child: Container(
                          height: 60,
                          width: double.infinity,

                          decoration: BoxDecoration(
                            borderRadius: BorderRadius.circular(10),
                            color: Colors.black,
                          ),
                          child: Center(child: Text('Know more',style:
GoogleFonts.getFont('Didact Gothic',color:Colors.white,fontWeight:
FontWeight.bold,fontSize: 26),)),
                        ),
                      ),
                    ),
                  ],
                ),
              )
                  : Container()
            ],
          ),
        ),
      );
  }
}
```

3. Model - Importing Libraries

```
import numpy as np
import pandas as pd
import random
import os
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import keras.applications.xception as xception
import zipfile
import sys
import time
import tensorflow.keras as keras
import tensorflow as tf
import re

from PIL import Image
from keras.layers import Input, Conv2D, Dense, Flatten, MaxPooling2D, Input,
GlobalAveragePooling2D
from keras.layers.experimental.preprocessing import Normalization
from keras.models import Model, Sequential
from keras.preprocessing import image
from keras.utils import to_categorical
from keras.layers import Lambda
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

print('setup successful!')
```

4. Model - DataFrame Creation

```
# Add class name prefix to filename. So for example "/paper104.jpg" become
"paper/paper104.jpg"
def add_class_name_prefix(df, col_name):
    df[col_name] = df[col_name].apply(lambda x: x[:re.search("\d",x).start()]
+ '/' + x)
    return df

# list conatining all the filenames in the dataset
filenames_list = []
# list to store the corresponding category, note that each folder of the
dataset has one class of data
categories_list = []
```

```
for category in categories:
    filenames = os.listdir(base_path + categories[category])

    filenames_list = filenames_list  +filenames
    categories_list = categories_list + [category] * len(filenames)

df = pd.DataFrame({
    'filename': filenames_list,
    'category': categories_list
})

df = add_class_name_prefix(df, 'filename')

# Shuffle the dataframe
df = df.sample(frac=1).reset_index(drop=True)

print('number of elements = ' , len(df))

# see sample image, you can run the same cell again to get a different image
random_row = random.randint(0, len(df)-1)
sample = df.iloc[random_row]
randomimage = image.load_img(base_path +sample['filename'])
print(sample['filename'])
plt.imshow(randomimage)
```

5. Model - Visualization and Splitting

```
df_visualization = df.copy()
# Change the catgegories from numbers to names
df_visualization['category'] = df_visualization['category'].apply(lambda
x:categories[x] )

df_visualization['category'].value_counts().plot.bar(x = 'count', y =
'category' )
df_visualization['category'].value_counts()
# plt.xlabel("Garbage Classes", labelpad=14)
# plt.ylabel("Images Count", labelpad=14)
# plt.title("Count of images per class", y=1.02);
```

```
df_visualization = df.copy()
# Change the catgegories from numbers to names
df_visualization['category'] = df_visualization['category'].apply(lambda
x:categories[x] )

df_visualization['category'].value_counts().plot.bar(x = 'count', y =
'category' )

plt.xlabel("Garbage Classes", labelpad=14)
plt.ylabel("Images Count", labelpad=14)
plt.title("Count of images per class", y=1.02);
#Change the categories from numbers to names
df["category"] = df["category"].replace(categories)

# We first split the data into two sets and then split the validate_df to two
sets
train_df, validate_df = train_test_split(df, test_size=0.2, random_state=42)
validate_df, test_df = train_test_split(validate_df, test_size=0.5,
random_state=42)

train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)

total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
total_test = test_df.shape[0]

print('train size = ', total_train , 'validate size = ', total_validate, 'test
size = ', total_test)
```

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Model and Accuracy

On the Kaggle platform, Keras library and TensorFlow version 2.1.4 were used for all of the experiments in this work [14]. Using GPU P100, the trials were carried out. We employed weights, training data, completely new training models, validation during training, and training models for the studies.

The batch size and epoch count in the suggested model were both set to 64 and 20, respectively, but it was found that after epoch number 7, the model's accuracy did not increase. As a result, an early halting technique was used, and an accuracy of 94.91% was achieved. The test loss became stagnant at a same value and stopped decreasing further, which is the need for terminating after the seventh epoch.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, 320, 320, 3)       0
_________________________________________________________________
xception (Functional)        (None, 10, 10, 2048)      20861480
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 12)                24588
=================================================================
Total params: 20,886,068
Trainable params: 24,588
Non-trainable params: 20,861,480
```

Figure 6.1 Model Structure

To compute accuracy, we accounted for the following parameters, which helped make our successful implementation precise -

- Precision = TruePositives / (TruePositives + FalsePositives)
- Recall = TruePositives / (TruePositives + FalseNegatives)
- F1-score = (2 * Precision * Recall) / (Precision + Recall)

	precision	recall	f1-score	support
battery	0.93	0.97	0.95	102
biological	1.00	1.00	1.00	102
brown-glass	0.88	0.86	0.87	49
cardboard	0.96	0.94	0.95	93
clothes	0.98	0.99	0.99	533
green-glass	0.84	0.82	0.83	62
metal	0.88	0.87	0.87	83
paper	0.95	0.94	0.95	107
plastic	0.84	0.84	0.84	87
shoes	0.99	0.98	0.99	212
trash	0.96	0.96	0.96	57
white-glass	0.87	0.82	0.84	65
accuracy			0.95	1552
macro avg	0.92	0.92	0.92	1552
weighted avg	0.95	0.95	0.95	1552

Table 6.1 Accuracy Matrix

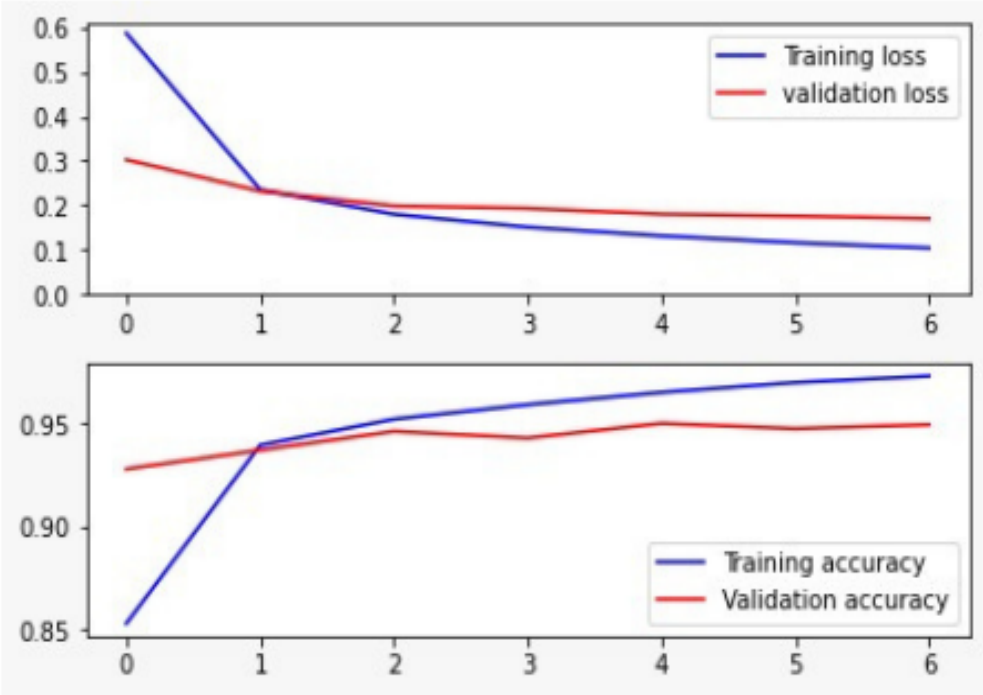
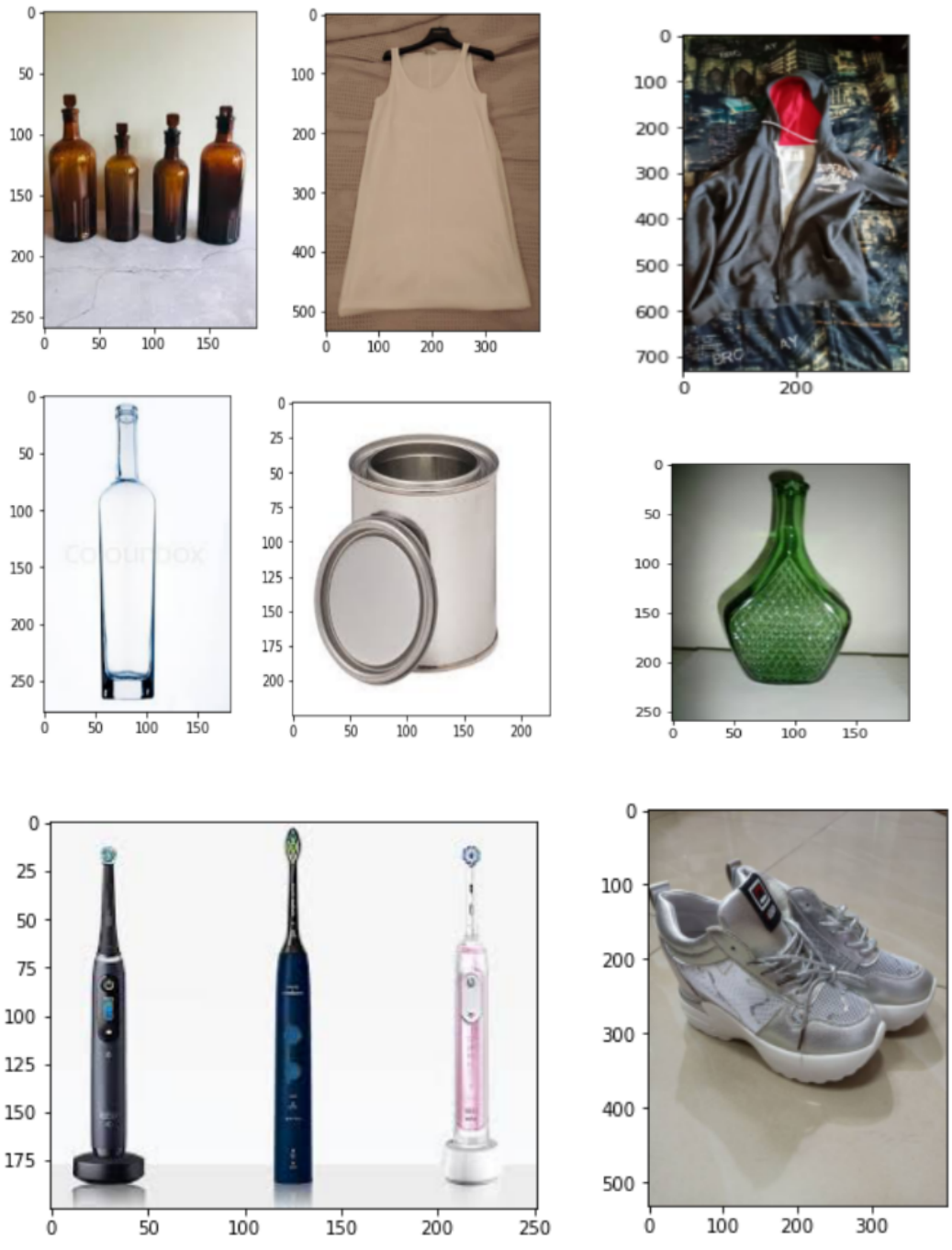


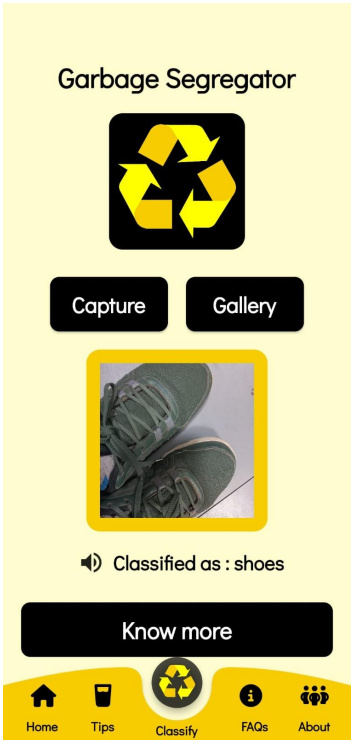
Figure 6.2 Accuracy and Loss Graph

6.2 Output Screenshots

1. Dataset Sample



2. Mobile Application



CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

To make garbage separation less onerous, we suggested a method of classifying waste into different categories. We also made this method accessible to regular users so they could use it to separate waste at the source. The proposed model uses the Xception model to accomplish this feat and has an accuracy that fluctuates randomly between 94 and 96 percent.

In the future, we hope to integrate an object detection model in addition to the classification to achieve waste segregation inside a single image. We also intend to enhance the dataset by including more recent photos, hence enhancing the model's precision. In addition, we want to add data in such a way as to lessen the highly skewed data set and make it more uniformly distributed.

REFERENCES

1) Trends in Solid Waste Management - https://datatopics.worldbank.org/what-a-waste/trends_in_solid_waste_management.html
2) Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network - Olugboja Adedeji, Zenghui Wang
3) A Method for Waste Segregation using Convolutional Neural Networks - Jash shah, Sagar Kamat
4) Xception: Deep Learning with Depthwise Separable Convolutions - Francois Chollet(Google Inc)
5) J. Donovan, "Auto-trash sorts garbage automatically at the techcrunch disrupt hackathon." [Online]. Available: https://techcrunch.com/2016/09/13/auto-trash-sortsgarbageautomatically-at-the-techcrunch-disrupt-hackathon/
6) G. Mittal, K. B. Yagnik, M. Garg, and N. C. Krishnan, "Spotgarbage: Smartphone app to detect garbage using deep learning," in Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ser. UbiComp '16. New York, NY, USA: ACM, pp. 940–945, 2016. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971731
7) S. Meng and W. -T. Chu, "A Study of Garbage Classification with Convolutional Neural Networks," 2020 Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN), 2020, pp. 152-157, doi: 10.1109/Indo-TaiwanICAN48429.2020.9181311.
8) Ziouzios, D.; Tsiktsiris, D.; Baras, N.; Dasygenis, M. A Distributed Architecture for Smart Recycling Using Machine Learning. Future Internet 2020, 12, 141. https://doi.org/10.3390/fi12090141
9) Gao, M.; Qi, D.; Mu, H.; Chen, J. A Transfer Residual Neural Network Based on ResNet-34 for Detection of Wood Knot Defects. Forests 2021, 12, 212. https://doi.org/10.3390/f12020212
10) K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in CVPR, 2016.
11) Youpeng Yu, , and Ryan Grammenos. "Towards artificially intelligent recycling Improving image processing for waste classification.", arXiv:2108.06274 [cs.CV],(2021).

12) https://www.kaggle.com/datasets/mostafaabla/garbage-classification
13) Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning: https://arxiv.org/abs/1602.07261
14) Kaggle: https://www.kaggle.com/
15) Flutter: https://flutter.dev/
16) Google Fonts: https://pub.dev/packages/google_fonts
17) Just Audio: https://pub.dev/packages/just_audio
18) URL Launcher: https://pub.dev/packages/url_launcher
19) Carousel Slider: https://pub.dev/packages/carousel_slider
20) Image Picker: https://pub.dev/packages/image_picker
21) Flutter Tflite: https://pub.dev/packages/flutter_tflite
22) Font Awesome: https://pub.dev/packages/font_awesome_flutter
23) Trash Classification Using Convolutional Neural Networks, Project Category: Computer Vision - Yujie He,Qinyue Gu,Maguo Shi
24) A deep learning approach based hardware solution to categorize garbage in an environment - Tanya Gupta · Rakshit Joshi· Devarshi Mukhopadhyay · Kartik Sachdeva · Nikita Jain1 · Deepali Virmani Laura Garcia-Hernandez
25) Waste Segregation Using Deep Learning - Meena. U, Shailaja. R, Swanit.R, Prajakta.P
26) Classification of TrashNet Dataset Based on Deep Learning Models : Rahmi Arda Aral; Şeref Recep Keskin; Mahmut Kaya; Murat Hacıömeroğlu
27) Classification of Trash for Recyclability Status: Mindy Yang, Gary Thung
28) S. Albawi, T. A. Mohammed and S. Al-Zawi, ”Understanding of a convolutional neural network,” 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTech-nol.2017.8308186
29) K. Ahmad, K. Khan, and A. Al-Fuqaha, ”Intelligent Fusion of Deep Features for Improved Waste Classification,” in IEEE Access, vol. 8, pp. 96495-96504, 2020, doi: 10.1109/ACCESS.2020.2995681