Automated intelligent Waste management system in smart cities: enhanced recycling facilities

A project report submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

by

Gullapalli Krishna Priya (21BF1A3352)
Ande Gayathri (21BF1A3301)
Bijivemula Srikanth Reddy (21BF1A3328)
Kalavagunta Nithya Santhosh (21BF1A3363)

Under the guidance of
Dr. A. Revathi
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI & ML)
SRI VENKATESWARA COLLEGE OF ENGINEERING
(AUTONOMOUS)
(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
Accredited by NBA, New Delhi & NAAC with 'A' grade)
Karakambadi Road, TIRUPATI – 517507

2021 - 2025

SRI VENKATESWARA COLLEGE OF ENGINEERING (AUTONOMOUS)

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
Accredited by NBA, New Delhi & NAAC with 'A' Grade)
Karakambadi Road, TIRUPATI – 517507.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI & ML)

CERTIFICATE

This is to certify that the project work entitled, "**Automated Intelligent Waste management system in smart cities: enhanced recycling facilities**" is a bonafide record of the project work done and submitted by

Gullapalli Krishna Priya (21BF1A3352)
Ande Gayathri (21BF1A3301)
Bijivemula Srikanth Reddy (21BF1A3328)
Kalavagunta Nithya Santhosh (21BF1A3363)

under my guidance and supervision for the partial fulfillment of the requirements for the award of B.Tech degree in **COMPUTER SCIENCE AND ENGINEERING (AI & ML).** This project is the result of our own effort and that it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above

GUIDE **HEAD OF THE DEPARTMENT**

External Viva-Voce Exam held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the project report entitled **"Automated intelligent Waste management system in smart cities: enhanced recycling facilities"** done by us is submitted in partial fulfilment of the requirements for the award of the Bachelor's of Technology in **OF COMPUTER SCIENCE AND ENGINEERING (AI & ML).**

Gullapalli Krishna Priya (21BF1A3352)

Ande Gayathri (21BF1A3301)

Bijivemula Srikanth Reddy (21BF1A3328)

Kalavagunta Nithya Santhosh (21BF1A3363)

ACKNOWLEDGEMENT

We are thankful to our guide **Dr. A. Revathi**, Associate Professor for her valuable **guidance** and encouragement. Her helping attitude and suggestions have helped us in the successful completion of the project.

We would like to express our gratefulness and sincere thanks to **Dr. R. Swathi, Head of the DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI & ML)** for her kind help and encouragement during the course of our study and in the successful completion of the project work.

We have great pleasure in expressing our hearty thanks to our beloved **Principal Dr. N. Sudhakar Reddy**, for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank our **Management** for providing all our necessary facilities during the course of study.

We would like to thank our parents and friends, who have the greatest contributions in all our achievements, for the great care and blessings in making us successful in all our endeavors.

Gullapalli Krishna Priya **(21BF1A3352)**

Ande Gayathri **(21BF1A3301)**

Bijivemula Srikanth Reddy **(21BF1A3328)**

Kalavagunta Nithya Santhosh **(21BF1A3363)**

ABSTRACT

With increasing urbanization, waste has become a major problem in the present world. Therefore, proper waste management is a must for a healthy and clean environment. The proposed system is not requiring any prior tagging of items which makes it closer to the current end-user habit. Object recognition is a technique to detect the semantic of objects in digital images then to identify those objects into a particular class. A convolutional neural network is first trained to recognize and classify fundus images of biodegradable and non-biodegradable item. The network is then visualized, using a technique of pixel optimization, to discover the features that the trained network looks for to classify the image. According to a survey conducted by the World Bank organization, it is stated that annually the world generates more than 2 billion metric tons of solid waste and at least 33% of that is not managed in an environmentally friendly manner.

The world bank organization further states that, if no proper actions are taken, this solid waste generation will be increased 3 times. Due to the various impacts of environmental pollution, people have been more focused on proper waste management. Waste management involves processes of waste identification, collection, transportation, recycling of valuable items, and disposal of unwanted wastes. Recently, fully connected and convolutional neural networks have been trained to achieve state-of-the-art performance on a wide variety of tasks such as speech recognition, image classification, natural language processing, and bioinformatics. In our approach, we trained a deep Convolutional Neural Network model on a large dataset to achieve higher accuracy.

TABLE OF CONTENTS

List of Figures

LIST OF ABBREVATIONS

Name	ABBREVATION
API	Application Programming Interface
AODE	Averaged One-Dependence Estimators
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
DL	Deep Learning
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
I Python	Interactive Python
Jupiter	(I Python Notebook)
KDD	Knowledge Discovery in Databases
ML	Machine Learning
NB	Naive Bayes
PCA	Principal Component Analysis
Pippi	Python Package Index
RBF	Radial Basis Function
RNN	Recurrent Neural Network
SVM	Support Vector Machine
UML	Unified Modelling Language
VGG	Visual Geometry Group
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

INTRODUCTION

With rapid urbanization, waste management has emerged as a critical challenge for maintaining a clean and healthy environment. The proposed Automated Intelligent Waste Management System aims to address this issue by leveraging advanced machine learning techniques for efficient waste classification and recycling. By utilizing convolutional neural networks (CNNs), the system can accurately identify and categorize waste materials, such as biodegradable and non-biodegradable items, without requiring prior tagging. This innovative approach not only streamlines waste segregation but also enhances recycling efforts, contributing to sustainable urban living. As cities continue to grow, implementing such intelligent systems will be essential for effective waste management and environmental protection, ultimately fostering cleaner and smarter urban spaces.

1.1 MOTIVATION

The increasing volume of waste generated in urban areas poses significant environmental challenges, including pollution and resource depletion. Traditional waste management methods are often inefficient and labor-intensive, leading to improper disposal and recycling practices. Our motivation stems from the urgent need for innovative solutions that leverage technology to enhance waste management processes. By developing an Automated Intelligent Waste Management System, we aim to promote sustainable practices, improve recycling rates, and foster a cleaner environment. This project aspires to contribute to the creation of smarter cities that prioritize ecological balance and public health.

1.2 PROBLEM STATEMENT

Urban areas generate diverse waste types, including biodegradable and non-biodegradable materials, complicating effective waste management. Traditional methods often fail to accurately classify and segregate these materials, leading to increased pollution and inefficient recycling. Our project addresses this challenge by developing an automated system for multi-label waste classification and management.

1.3 OBJECTIVE

The primary objective of this project is to develop an Automated Intelligent Waste

Management System that efficiently classifies and segregates various types of waste in urban environments. By utilizing advanced machine learning techniques, particularly convolutional neural networks (CNNs), the system aims to accurately identify multiple waste categories, including biodegradable and non-biodegradable materials. This automation will reduce the reliance on manual sorting, thereby minimizing human error and labor costs. Additionally, the system will enhance recycling efforts by providing real-time data on waste composition, enabling better resource recovery. Ultimately, the project seeks to contribute to sustainable waste management practices, promote environmental awareness, and support the development of smarter, cleaner cities. Through this initiative, we aim to foster a healthier urban ecosystem and improve overall waste management efficiency.

1.4 SCOPE

The scope of this project encompasses the design and implementation of an Automated Intelligent Waste Management System tailored for urban environments. It includes the development of a machine learning model capable of classifying various waste types, such as plastics, metals, and organic materials. The system will facilitate real-time monitoring and reporting, enhancing recycling processes and promoting sustainable waste management practices in smart cities.

1.5 INTRODUCTION

The rapid growth of urban populations has led to an unprecedented increase in waste generation, posing significant challenges for effective waste management.

Traditional methods often fall short in accurately classifying and segregating diverse waste types, resulting in environmental pollution and resource wastage. This project introduces an Automated Intelligent Waste Management System that leverages advanced machine learning techniques, specifically convolutional neural networks (CNNs), to enhance waste classification. By automating the identification of biodegradable and non-biodegradable materials, the system aims to streamline recycling efforts, reduce manual labor, and promote sustainable practices, ultimately contributing to cleaner and smarter urban environments.

1.6 METHODOLOGY

1.6.1 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) is a deep learning model widely used for image classification tasks, including waste classification. In this methodology, the images collected from the dataset are first resized to a standard dimension of 128x128 pixels and normalized for uniformity. Preprocessing is followed by data augmentation techniques such as rotation, flipping, and noise addition to improve model generalization and prevent overfitting. The dataset is then split into training and testing subsets in an 80:20 ratio. The CNN model consists of several convolutional layers with increasing filter sizes (32, 64, 128, 256, and 1024), each followed by a Rectified Linear Unit (ReLU) activation function that introduces non-linearity. Max pooling layers are applied to reduce the spatial dimensions of the feature maps, helping to minimize computation. A global average pooling layer is used before the fully connected layers to reduce the number of trainable parameters and further avoid overfitting. Finally, a dense layer with a Softmax activation function classifies the image into waste categories. The trained model is evaluated using performance metrics like accuracy, precision, recall, and confusion matrix. The CNN provides an efficient end-to-end solution for automatic feature extraction and classification from waste images.

1.6.2 VGG-16

VGG-16 is a pre-trained Convolutional Neural Network model used in this project through a transfer learning approach for waste classification. The model takes images resized to 224x224 pixels as input. As part of transfer learning, the convolutional base of VGG-16—trained on the ImageNet dataset—is reused, while the top layers are replaced with custom dense layers suited for the specific waste categories. The initial convolutional layers are kept frozen to retain their pre-learned features, allowing the model to focus on training the new classification layers. The architecture consists of 13 convolutional layers followed by 3 fully connected layers, using small 3x3 filters, same padding, and ReLU activation functions throughout. Max pooling layers are applied after every few convolutional blocks to reduce the spatial dimensions. The final fully connected layers end in a Softmax layer that predicts the waste category. VGG-16 is preferred for its high accuracy and ability to generalize well even with limited training data. The model is evaluated using standard performance metrics, and its weights are saved for real-time waste classification.

1.6.3 Workflow

The workflow for waste classification using CNN and VGG-16 begins with collecting a labeled image dataset of various waste types, including biodegradable and non-biodegradable materials. These images are preprocessed by resizing and normalizing to ensure uniform input, followed by data augmentation techniques like rotation and flipping to improve model generalization. The dataset is then split into training and testing sets in an 80:20 ratio. Feature extraction is performed using either a custom-built CNN model or a pre-trained VGG-16 network. These features are passed through dense layers and a Softmax activation function for classification. The model is evaluated using metrics like accuracy and confusion matrix, and once satisfactory performance is achieved, the trained model is saved for real-time waste classification on new images. Below Fig: 1 shows the workflow of the waste classification system.

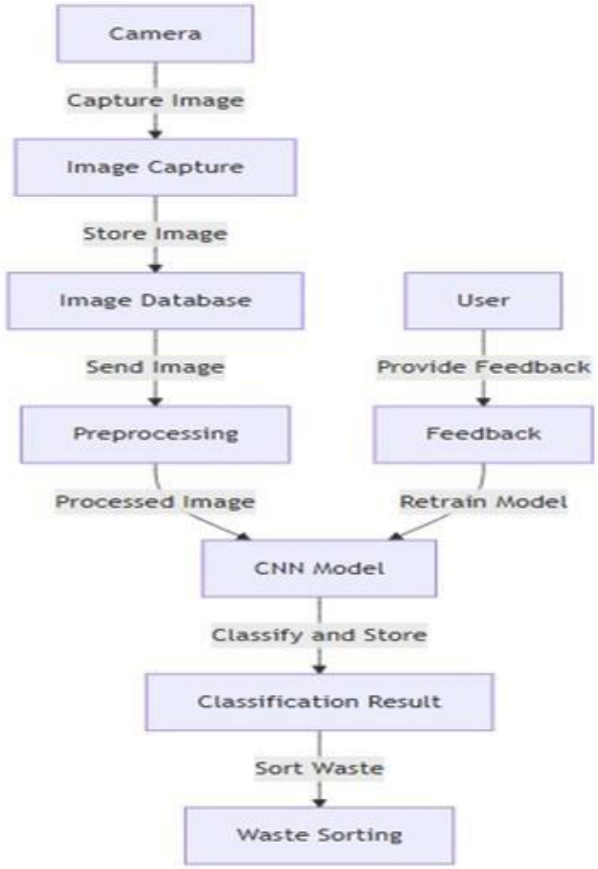

Fig 1:Workflow of the Waste Classification System

Chapter 2

LITERATURE SURVEY

2.1 RELATED WORK

[1] Abeygunawardhana A. G. D. T., Shalinda R. M. M. M., "Introducing Innovative Item Management Process Towards Providing Smart Fridges," 2020.

This research presents an AI-driven smart waste bin designed to classify common solid waste materials, including Metal, Glass, and Plastic. Utilizing image processing and machine learning algorithms, the smart bin separates waste and continuously monitors waste levels with ultrasonic sensors. A dedicated mobile application optimizes collection routes for waste collectors. By relying on visual data for waste recognition, the system eliminates the need for expensive sensors, allowing for efficient waste categorization and real-time notifications of container levels, thus providing a portable waste management solution.

[2] Akshay Dabholkar, Bhushan Muthiyan, "Smart Illegal Dumping Detection," 2017.

This paper addresses the persistent issue of illegal dumping in urban areas, which poses health risks and degrades city aesthetics. The authors propose a deep learning approach to identify frequently dumped waste types, enhancing detection accuracy compared to existing manual monitoring systems. By employing edge computing, the system minimizes unnecessary image transfers to servers, processing images locally and only sending relevant data when illegal dumping is detected. The study demonstrates high recognition accuracy with a compact memory footprint, showcasing the effectiveness of deep learning in waste management applications.

[3] Berker Arslan, Sefer Memiş, "Fine-Grained Food Classification Methods on the UEC FOOD-100 Database," 2022.

This article explores automatic food recognition systems, emphasizing their applications in waste food management and dietary monitoring. The authors review common deep learning methods for food classification and present benchmark results on the UEC Food-100 database, achieving a state-of-the-art accuracy of 90.02%. The study highlights the ensemble method's effectiveness, combining predictions from ResNeXt and DenseNet models. By averaging results over multiple trials, the paper provides a comprehensive analysis of food classification performance, contributing valuable insights to the field of automatic food recognition.

[4] Andrew E. Bruno, Patrick Charbonneau, Janet Newman, Edward H. Snell, David R. So, Vincent Vanhoucke, Christopher J. Watkins, Shawn Williams, Julie Wilson, "Classification of Crystallization Outcomes using Deep Convolutional Neural Networks," 2018.

The MARCO initiative compiles approximately half a million annotated images from macromolecular crystallization experiments. This study employs advanced machine learning algorithms to classify these images, achieving over 94% accuracy in labeling test images regardless of their experimental origin. The findings underscore the potential of machine learning in enhancing high-density screening and systematic analysis of crystallization experiments, paving the way for both industrial and fundamental research applications.

[5] Farzana Shaikh, Nagma Kazi, Farheen Khan, "Waste Profiling and Analysis using Machine Learning," 2018.

This paper addresses the significant waste generation in India, focusing on the need for effective waste segregation. The authors propose a machine learning-based system to classify waste as dry or wet based on images of garbage. By simplifying the process for civic bodies, the system aims to enhance waste disposal practices and create awareness about waste management. The study emphasizes the importance of accurate waste detection in improving disposal habits and optimizing waste treatment budgets, contributing to more efficient waste management strategies.

Chapter 3

EXISTING SYSTEM AND PROPOSED METHOD

3.1 EXISTING SYSTEM

A dumping site detection by using features extracted from images that are taken from the sky using images. To correctly distinguish dumping site from various types of clean areas, such as wheat field, ground, lake and forest, they proposed to use not only spectral information but also spatial information extracted from the images. The extracted features are fed to a Bayesian classifier to automatically distinguish illegal dumping spots from clean areas.

3.2 DISADVANTAGES

1. The Bayesian algorithm takes more time to train the model and find the best hyper parameter.
2. This is text class classification problem. The time complexity is very high compared to other classification tasks.
3. Collecting the data and convert the text data into numerical is very hard.

3.3 PROPOSED SYSTEM

The proposed system aims to develop an Automated Intelligent Waste Management System that utilizes advanced machine learning techniques for multi-class waste classification. By employing Convolutional Neural Networks (CNNs) and transfer learning with pre-trained models like VGG-16, the system will accurately identify various waste categories, including plastics, metals, and organic materials. The model will be trained on a diverse dataset of waste images, enabling real-time classification and segregation. This innovative approach will enhance recycling efforts, reduce manual sorting, and promote sustainable waste management practices in urban environments, contributing to cleaner and smarter cities.

3.4 ADVANTAGES

1. We can predict the result with best accuracy.
2. Time complexity also low compared to other machine learning algorithms and we can use this in low latency situation also.
3. By using CNN algorithm, we can predict disease very accurately.

Chapter 4

HARDWARE AND SOFTWARE REQUIREMENTS

4.1 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

The system encompasses both functional and non-functional requirements to ensure effective waste classification and management, enhancing user experience and operational efficiency.

4.1.1 Functional Requirements:

The system must accurately classify various waste types, including plastics, metals, and organic materials, using machine learning algorithms. It should provide real-time waste level monitoring and notifications for collection. Users must be able to upload images of waste for classification, and the system should generate reports on waste composition and recycling rates. Additionally, the system should support a user-friendly mobile application for waste management tracking and route optimization for waste collectors.

4.1.2 Non-functional requirements:

The system should ensure high accuracy in waste classification, achieving at least 90% accuracy in identifying waste types. It must be scalable to accommodate increasing data and user demands. The response time for image classification should be under five seconds, ensuring real-time processing. The system should maintain data security and user privacy, complying with relevant regulations. Additionally, it should be user-friendly, with an intuitive interface for both collectors and end-users

4.2 HARDWARE REQUIREMENTS

The system requires a computer or server with a minimum of 16 GB RAM, a multi-core processor, and a dedicated GPU for efficient model training and inference. Additionally, ultrasonic sensors for waste level monitoring are needed.

PROCESSOR	:	Intel I5
RAM	:	4GB
HARD DISK	:	500 GB

4.3 SOFTWARE REQUIREMENTS

The system will utilize Python as the primary programming language, along with libraries such as TensorFlow or PyTorch for machine learning. A database management system, like MySQL or Sqlite, will be required for data storage. The mobile application will be developed using React Native or Flutter for cross-platform compatibility, ensuring accessibility for users.

PYTHON IDE	**:**	Anaconda Jupyter Notebook, VScode
PROGRAMMING LANGUAGE	:	Python
SERVER-SIDE SCRIPT	**:**	HTML, CSS

Chapter 5

MODULE DESCRIPTION

5.1 DATA COLLECTION

Data collection is a critical step in developing an effective Automated Intelligent Waste Management System. For this project, a diverse dataset of waste images is essential to ensure the model can accurately classify various waste types, including plastics, metals, and organic materials.

5.1.1 Sources of Data:

1. **Public Datasets**: Utilize existing public datasets related to waste classification, such as the WasteNet dataset or other relevant image repositories.
2. **Custom Data Collection**: Capture images using smartphones or cameras in various urban environments, ensuring a wide variety of waste types and conditions (e.g., different lighting, angles, and backgrounds).
3. **Collaboration with Local Authorities**: Partner with local waste management authorities to access their data and images of waste collected from different locations.

5.1.2 Data Characteristics

- **Image Format**: Collect images in common formats such as JPEG or PNG.
- **Resolution**: Ensure images are of high resolution (at least 1280x720 pixels) to capture sufficient detail for classification.
- **Labeling**: Each image must be accurately labeled with its corresponding waste category to facilitate supervised learning as show in the fig 2: labels classification of the dataset images

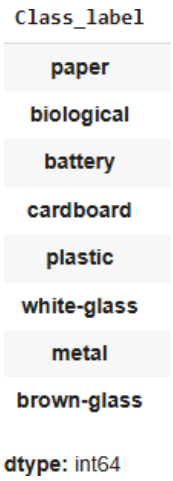

Fig 2 : labels classification of the dataset images

5.2 Image Preprocessing

Image preprocessing is essential to prepare the collected images for effective model training. This step involves several techniques to enhance image quality and ensure uniformity across the dataset.

Steps in Image Preprocessing

1. **Resizing**: All images are resized to a standard dimension (e.g., 128x128 pixels) to ensure consistency in input size for the model.
2. **Normalization**: Pixel values are normalized to a range of 0 to 1 by dividing by 255. This helps in speeding up the convergence of the model during training.
3. **Color Space Conversion**: Convert images to grayscale if color information is not critical for classification, reducing computational complexity.
4. **Noise Reduction**: Apply techniques such as Gaussian blur to reduce noise and improve image quality.

5.3 Data Augmentation

Data augmentation is a technique used to artificially expand the size of the training dataset by creating modified versions of images. This helps improve the model's generalization capabilities and reduces overfitting.

Augmentation Techniques

1. **Rotation**: Randomly rotate images within a specified range (e.g., -30 to +30 degrees) to simulate different orientations.

2. **Flipping**: Horizontally flip images to introduce variability in the dataset.
3. **Zooming**: Apply random zoom to images to simulate different distances from the camera.
4. **Brightness Adjustment**: Randomly adjust the brightness of images to account for varying lighting conditions.
5. **Shearing**: Apply shearing transformations to create slanted versions of the images as shown in the fig 3: data augmentation on the images

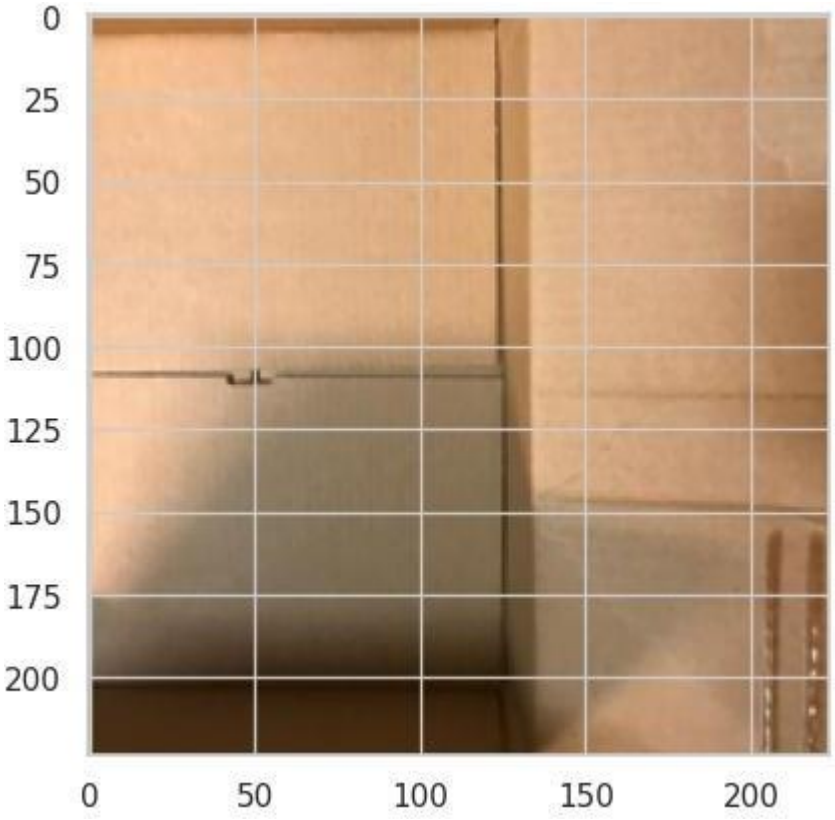

Fig 3 : Data augmentation on the images

5.4 Data Splitting

Data splitting is crucial for evaluating the model's performance. The dataset is divided into training, validation, and testing subsets to ensure that the model can generalize well to unseen data.

Splitting Strategy

1. **Training Set**: Typically, 80% of the dataset is used for training the model. This set is used to teach the model to recognize different waste categories.

2. **Validation Set**: 10% of the dataset is reserved for validation during training. This set helps in tuning hyperparameters and preventing overfitting.
3. **Testing Set**: The remaining 10% is used for final evaluation after the model has been trained. This set provides an unbiased assessment of the model's performance as show in the below fig 4: pie chart of the training, validation and testing of the dataset.

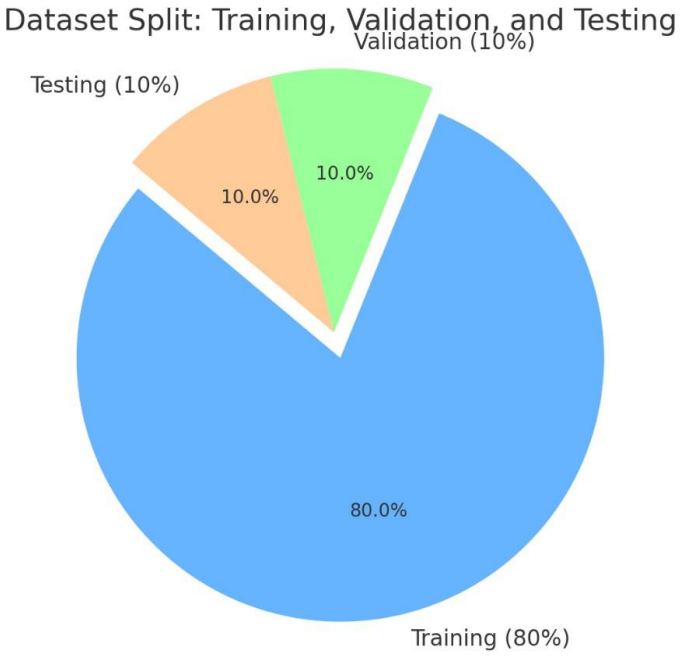

Fig 4 :Pie chart of the training, validation and testing of the dataset

5.5 Modeling

Modeling involves selecting and training the appropriate machine learning algorithms to classify the waste images. In this project, we will utilize Convolutional Neural Networks (CNNs) and transfer learning with pre-trained models like VGG-16.

5.5.1 CNN Architecture

1. **Input Layer**: Accepts preprocessed images of size 128x128 pixels.
2. **Convolutional Layers**: Multiple convolutional layers with increasing filter sizes (e.g., 32, 64, 128) to extract features from images.
3. **Activation Function**: Use Rectified Linear Unit (ReLU) activation to introduce Non-linearity.
4. **Pooling Layers**: Max pooling layers to reduce the spatial dimensions of feature maps, minimizing computation.

5. **Fully Connected Layers**: Dense layers that connect all neurons from the previous layer culminating in a Softmax layer for multi-class classification.

5.5.2 Transfer Learning with VGG-16

1. **Pre-trained Model**: Utilize the VGG-16 model, which has been trained on a large dataset (ImageNet), to leverage its learned features for waste classification.

2. **Fine-tuning**: Replace the final classification layer with a new layer that corresponds to the number of waste categories in our dataset. Freeze the initial layers to retain learned features while training the new layer.

3. **Training**: Train the model using the training dataset, applying techniques such as early stopping and learning rate scheduling to optimize performance as shown in the below Fig 5 :Diagram of the CNN architecture and transfer learning process.

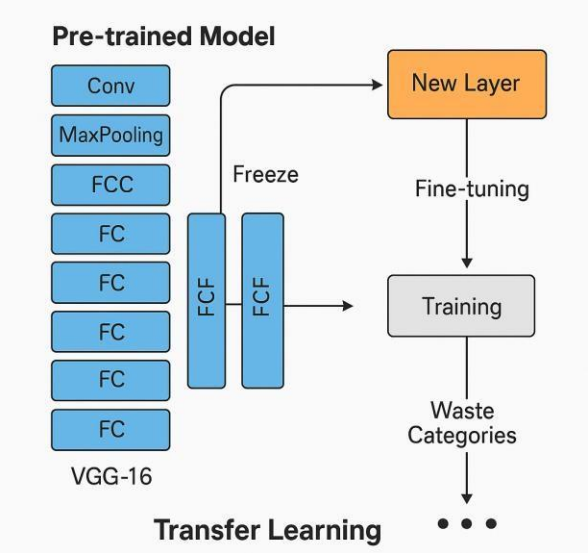

Fig 6 :Diagram of the CNN architecture and transfer learning process.

5.6 Prediction and Evaluation

After training the model, the next step is to evaluate its performance on the test dataset and make predictions on new images.

5.6.1 Prediction Process

1. **Input New Images**: Preprocess new images using the same techniques applied during training (resizing, normalization).
2. **Model Inference**: Use the trained model to predict the waste category of the input images, generating probabilities for each class.

5.6.2 Evaluation Metrics

1. **Accuracy**: Calculate the overall accuracy of the model by comparing predicted labels with true labels in the test set.
2. **Confusion Matrix**: Generate a confusion matrix to visualize the performance across different waste categories, identifying misclassifications.
3. **Precision, Recall, and F1-Score**: Compute these metrics to assess the model's performance in detail, particularly for imbalanced datasets.

Chapter 6

SYSTEM DESIGN

6.1 ARCHITECTURE

The architecture of the Automated Intelligent Waste Management System is designed to facilitate efficient waste classification and management through a well-structured integration of hardware and software components. This architecture can be divided into three main layers: Data Acquisition, Processing, and User Interface, each playing a crucial role in the overall functionality of the system as shown in the below Fig 7: System Design Automated Intelligent Waste Management System.

The Data Acquisition Layer is responsible for collecting images of waste materials. It includes high-resolution cameras or smartphones that capture images of waste in various environments. These devices can be strategically mounted on waste bins or utilized by waste collectors to take pictures of waste items. Additionally, ultrasonic sensors are integrated into this layer to monitor the fill levels of waste bins, providing real-time data on when bins need to be emptied. This information is essential for optimizing collection routes and schedules, ensuring that waste management operations are both efficient and timely.

The Processing Layer serves as the core of the system, where data is analyzed and classified. This layer consists of an image preprocessing module that handles the initial processing of captured images, including resizing, normalization, and noise reduction to ensure uniformity and quality. To enhance the training dataset, a data augmentation module applies various transformations to the images, such as rotation, flipping, and brightness adjustments, thereby improving the model's robustness. The heart of this layer is the machine learning model, which can be a Convolutional Neural Network (CNN) or a pre-trained model like VGG-16. This model is trained on the processed dataset to classify waste into categories such as plastics, metals, and organic materials. Finally, a prediction module is responsible for making predictions on new images and providing real-time classification results.

The User Interface Layer allows users to interact with the system effectively. It includes a user-friendly mobile application that enables waste collectors and users to upload images, view classification results, and receive notifications about bin status and collection schedules.

Additionally, a web-based dashboard is available for waste management authorities to monitor waste levels, analyze data trends, and optimize collection routes based on real-time information. This comprehensive architecture ensures a seamless flow of data from acquisition to processing and user interaction, ultimately contributing to more efficient waste management practices.

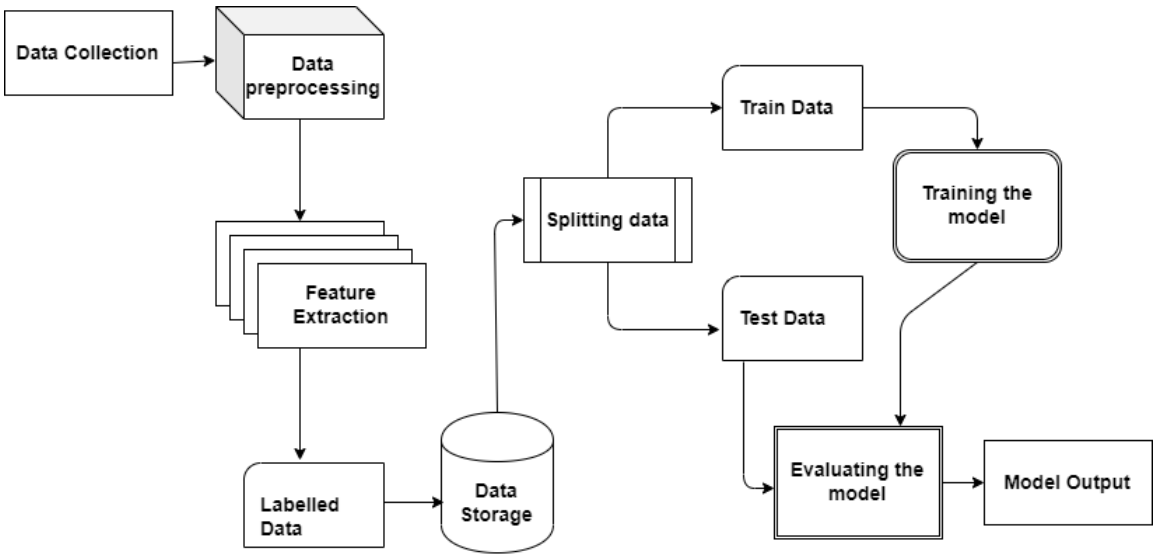

Fig 7:System Design Automated Intelligent Waste Management System

6.2 UML DIAGRAMS:

6.2.1 Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The below Fig 8 :Use case diagram of the Waste system is a visual representation of the interactions between actors (users or external systems) and the system under consideration. It illustrates the functionality provided by a system from the perspective of its users, helping to capture the requirements and scope of the system.

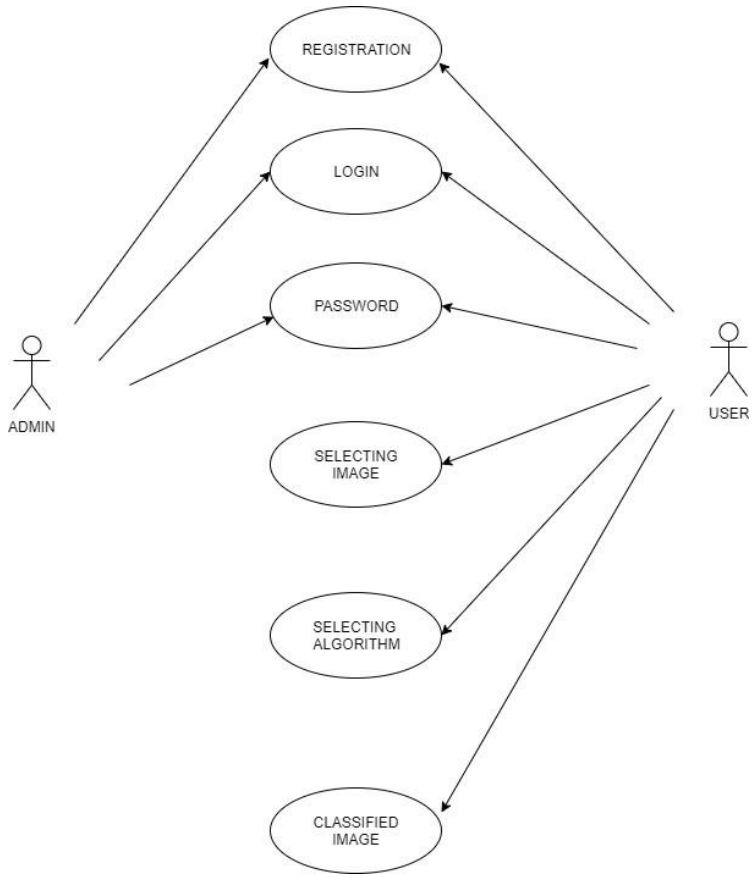

Fig 8 :Use case diagram of the Waste system

6.2.2 Class Diagram:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. The below Fig:9 Class Diagram of the Waste System is a visual representation of the static structure of a system, illustrating the classes, attributes, methods, and relationships among them.

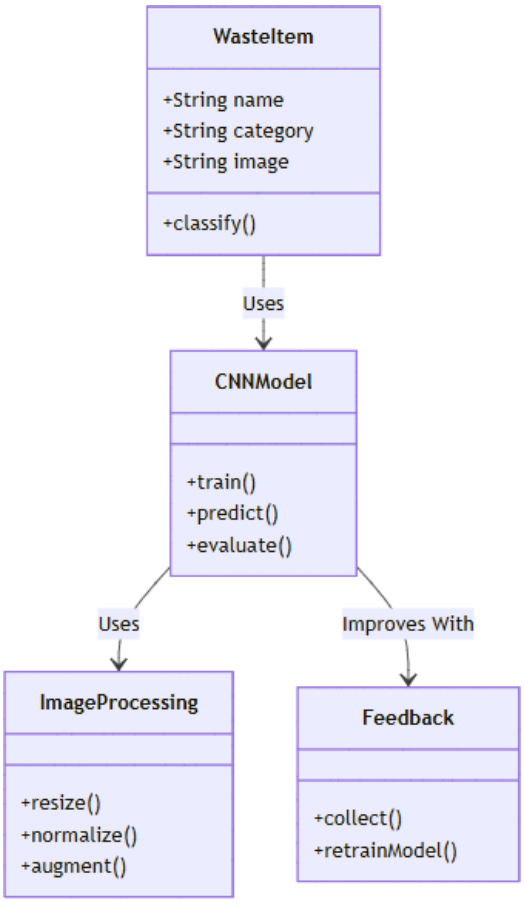

Fig 9 : Class Diagram of the Waste System

6.2.3 State Diagram:

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language. Then, all of the possible existing states are placed in relation to the beginning and the end as shown in the below

Fig 10: State Diagram of the Automated Intelligent Waste Management System

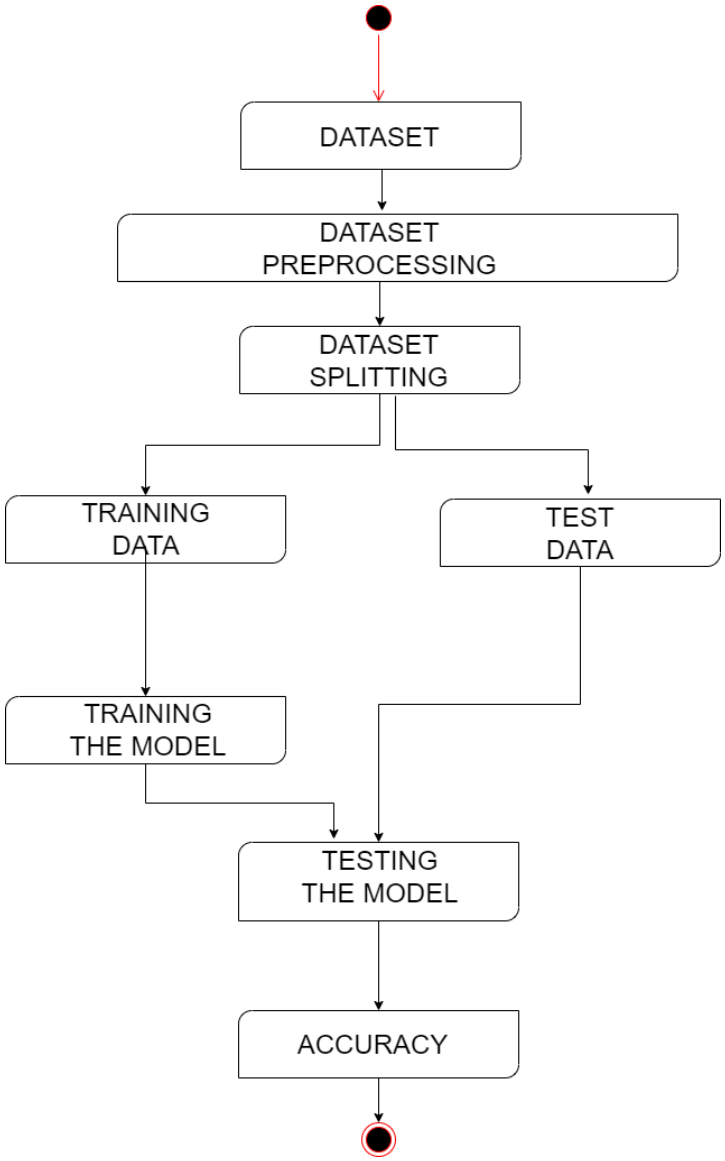

Fig 10 :State Diagram of the Automated Intelligent Waste Management System

6.2.4 Activity Diagram:

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination as shown in the below Fig 12 : Activity Diagram of the Automated Intelligent Waste Management System.

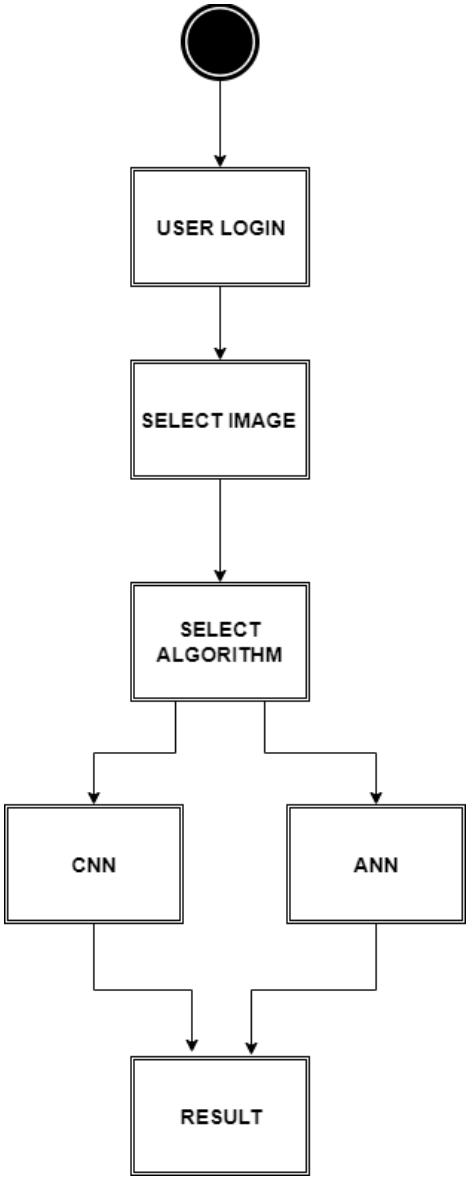

Fig 11 :Activity Diagram of the Automated Intelligent Waste Management System

Chapter 7

IMPLEMENTATION AND RESULTS

7.1 SOURCE CODE

7.1.1 MODEL CODE

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, GlobalMaxPooling2D
from tensorflow.keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

Images = []
for dirname, _, filenames in os.walk(data1):
    for filename in filenames:
        img = os.path.join(dirname, filename)
        print(dirname)
        Images.append(img)
Class_label = []
for i in Images:
  j = i.split("/")

#  print(j)
  Class_label.append(j[-2])

import random
temp = list(zip(Images, Class_label))
```

```
random.shuffle(temp)
Images, Class_label = zip(*temp)

data = pd.DataFrame(list(zip(Images, Class_label)), columns =['Image_path', 'Class_label'])

from sklearn.preprocessing import LabelBinarizer
enc = LabelBinarizer()
Y = enc.fit_transform(df['Class_label'])
from keras.utils.np_utils  import to_categorical
y = to_categorical(Y)

vgg=VGG19(weights='imagenet',include_top=False,input_shape=(224,224,3))
model_2=Sequential()
model_2.add(vgg)
model_2.add(Flatten())
model_2.add(Dense(128, activation='relu'))
model_2.add(Dropout(0.2))
model_2.add(Dense(2, activation='softmax'))
model_2.summary()
vgg=VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
model_1=Sequential()
model_1.add(vgg)
model_1.add(Flatten())
model_1.add(Dense(128, activation='relu'))
model_1.add(Dropout(0.2))
model_1.add(Dense(2, activation='softmax'))
model_1.summary()
History_1 = model_2.fit(X_train, y_train, epochs = 3, validation_data = (X_test,y_test),batch_size
= 128)

# plot the accuracy plot
plt.plot(History_1.history['accuracy'], 'r')
plt.plot(History_1.history['val_accuracy'], 'b')
```

```
plt.legend({'Train Accuracy': 'r', 'Test Accuracy':'b'})
plt.show()
```

7.1.2 BACKEND-SERVER CODE

models.py

```
from django.db import models

class User(models.Model):
    email = models.EmailField(unique=True)
    name = models.CharField(max_length=255)
    password = models.CharField(max_length=255)

    def __str__(self):
        return self.name

class Prediction(models.Model):
    text = models.CharField(max_length=200)
    prediction = models.CharField(max_length=30)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.text
```

serializers.py

```
from rest_framework import serializers
from .models import User
from .models import Prediction

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['email', 'name', 'password']

# class ResultSerializer(serializers.ModelSerializer):
```

```
#     class Meta:
#         model = Result
#         fields = '__all__'

class PredictionSerializer(serializers.ModelSerializer):
    class Meta:
        model = Prediction
        fields = ['id','text', 'prediction', 'created_at']
views.py
from rest_framework import status
from rest_framework.decorators import api_view
from rest_framework.response import Response
from .models import User
from .serializers import UserSerializer
from .serializers import PredictionSerializer
from .models import Prediction
from rest_framework.views import APIView
from rest_framework import authentication, permissions
from rest_framework import generics

class UserView(APIView):
    authentication_classes = [authentication.SessionAuthentication]
    permission_classes = [permissions.IsAuthenticated]

    def get(self, request):
        if request.user.is_authenticated:
            return Response({'username': request.user.username})
        else:
            return Response({'error': 'User not authenticated'})

@api_view(['POST'])
def user_signup(request):
    serializer = UserSerializer(data=request.data)
```

```
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['POST'])
def user_login(request):
    email = request.data.get('email')
    password = request.data.get('password')
    try:
        user = User.objects.get(email=email, password=password)
    except User.DoesNotExist:
                    return Response({'error': 'Invalid email or password'},
status=status.HTTP_401_UNAUTHORIZED)
    serializer = UserSerializer(user)
    return Response(serializer.data)

@api_view(['POST'])
def prediction_view(request):
    serializer = PredictionSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class PredictionList(generics.ListAPIView):
    queryset = Prediction.objects.all()
    serializer_class = PredictionSerializer
```

7.1.3 FRONTEND CODE:

./App.js

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Footer from './components/footer/Footer';
```

```
import Navbar from './components/navbar/Navbar';
import HomePage from './components/homepage/HomePage'
import Signup from './components/signup/Signup';
import Login from './components/login/Login';
import WasteVideo2 from "./components/wastevideo/WasteVideo2";
import WasteVideo from './components/wastevideo/WasteVideo'
import Landing from "./components/Landing";
import './App.css'
import PredictionList from "./components/predictions/PredictionList";

function App() {
  return (
    <div className="App">

      <BrowserRouter>
      <Navbar/>
        <Routes>
          <Route path="/" element={<Landing/>}/>
          <Route path="/home" element={<HomePage/>} />
          <Route path="/signup" element={<Signup />} />
          <Route path="/login" element={<Login />} />
          <Route path="/video" element={<WasteVideo2/>}/>
          <Route path="/image" element={<WasteVideo/>}/>
          <Route path="/predictionlist" element={<PredictionList/>}/>
        </Routes>
      <Footer/>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

./index,js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

App.css

```
html{
    scroll-behavior: smooth;
}
```

Index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

./reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
```

```
};

export default reportWebVitals;
```

./HomePage.jsx

```
import React from "react";
import dustbin from "./dustbin.png";
import truck from "./Truck.png";
// import video1 from "./wastvideo.mp4";

import { useState } from "react";
import { BsChevronCompactLeft, BsChevronCompactRight } from "react-icons/bs";
import { RxDotFilled } from "react-icons/rx";
import { useEffect } from "react";
import "./homepage.css";

import { Link } from "react-router-dom";
import ImageOne from "./photos/Bg11.jpg";
import ImageTwo from "./photos/li-hao-b25tsR8dBh0-unsplash.jpg";
import ImageThree from "./photos/jas-min-CIItgnBEOgw-unsplash.jpg";
import NewWaste from "../wastevideo/NewWaste";

const images = [ImageOne, ImageTwo, ImageThree];
const HomePage = () => {
  const [currentImage, setCurrentImage] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setCurrentImage((currentImage + 1) % images.length);
    }, 5000);
    return () => clearInterval(interval);
  }, [currentImage]);
```

```
const [text, setText] = useState("");
const fullText = "WELCOME TO ECOSAVVY";

useEffect(() => {
  let i = 0;
  const timer = setInterval(() => {
    if (i === fullText.length) {
    clearInterval(timer);
    } else {
      setText(fullText.slice(0, i + 1));
      i++;
    }
  }, 100);
  return () => clearInterval(timer);
}, []);

return (
  <div>
    <div className="relative h-screen w-screen">
      {images.map((image, index) => (
        <div
          key={index}
          className={`saa absolute inset-0 h-full w-full transition-opacity duration-2000 ${
            index === currentImage ? "opacity-100" : "opacity-0"
          }`}
          style={{ paddingBottom: "6.5%", paddingRight: "13px" }}
        >
          <h1 className="italic text-6xl font-bold absolute top-1/2 left-1/2 transform -translate-x-
1/2 text-green-500  underline     -translate-y-1/2 z-10  ">
            {text}
             {/* <span>W</span>ELCOME<span className="text-yellow-300"> HOME</span>
*/}
          </h1>
```

```
      <img
        src={image}
        alt={`Image ${index}`}
        className="absolute inset-0 h-full w-full object-cover object-center"
      />
    </div>
  ))}
</div>

{/* <div className="">
<video className="" src={video1} autoPlay={true} onError={(e) => console.log(e)} />
</div> */}

<section className="text-gray-600 body-font scroll-smooth" id="shawn">
 <div className="container mx-auto flex px-5 py-24 items-center justify-center flex-col">
   <img
     className=" lg:w-2/6 md:w-3/6 w-5/6 mb-10 object-cover object-center rounded"
     alt="hero"
     src={dustbin}
   />
   <div className="text-center lg:w-2/3 w-full">
     <h1 className="title-font sm:text-4xl text-3xl mb-4 font-medium text-gray-900">
       ML-based Garbage Segregation
     </h1>
     <p className="mb-8 leading-relaxed">
       The ML-based garbage segregation process using webcam and image
       upload involves capturing live images or uploading individual
       images of the waste material, followed by pre-processing and
       standardizing the images for training the supervised machine
       learning model. The trained model can then identify and segregate
       the waste materials based on their visual features, such as color,
       texture, and shape. The system provides feedback to the user,
```

```
            including a label or description of the waste material, to help
            them segregate their waste properly.{" "}
          </p>
          <div className="flex justify-center">
                <button className="inline-flex text-white bg-green-500 border-0 py-2 px-6
focus:outline-none hover:bg-green-600 rounded text-lg">
              <Link to="/video">Web Cam</Link>
            </button>
             <button className="ml-4 inline-flex text-gray-700 bg-gray-100 border-0 py-2 px-6
focus:outline-none hover:bg-gray-200 rounded text-lg">
              <Link to="/image">Image Upload</Link>
            </button>
          </div>
        </div>
      </div>
    </section>

    {/* next section */}

    <section className="text-gray-600 body-font">
     <div className="container mx-auto flex px-5 py-24 items-center justify-center flex-col">
       <img
         className="lg:w-2/6 md:w-3/6 w-5/6 mb-10 object-cover object-center rounded"
         alt="hero"
         src={truck}
       />
       <div className="text-center lg:w-2/3 w-full">
         <h1 className="title-font sm:text-4xl text-3xl mb-4 font-medium text-gray-900">
           Why EcoSavvy?
         </h1>
         <p className="mb-8 leading-relaxed">
           Our app can help you identify recyclable items and properly
           dispose of them, reducing waste and saving money. By doing so, you
```

```
            can make a positive impact on the environment and support the
            recycling industry, which is facing economic and
            diplomatic challenges. Our app can help address this challenge by
            providing users with a simple and intuitive way to identify
            recyclable items and learn how to dispose of them properly. This
            can have a significant positive impact on the environment by
            reducing waste and conserving resources, as well as supporting the
            recycling industry, which is facing economic and
            diplomatic challenges.
          </p>
          {/* <div className="flex justify-center">
                <button className="inline-flex text-white bg-indigo-500 border-0 py-2 px-6
focus:outline-none hover:bg-indigo-600 rounded text-lg">
              Button
            </button>
             <button className="ml-4 inline-flex text-gray-700 bg-gray-100 border-0 py-2 px-6
focus:outline-none hover:bg-gray-200 rounded text-lg">
              Button
            </button>
          </div> */}
        </div>
      </div>
    </section>
    <NewWaste />
  </div>
 );
};

export default HomePage;
```

./PredictionsList.jsx

```
import { PieChart, Pie, Legend, Tooltip } from 'recharts';
```

```
import { useState, useEffect } from 'react';
import axios from 'axios';

function PredictionList() {
  const [predictions, setPredictions] = useState([]);
  const [chartData, setChartData] = useState([]);

  useEffect(() => {
    axios.get('http://localhost:8000/api/predictions/')
      .then(response => {
        const data = response.data;
        setPredictions(data);
        // Extract percentages from prediction data
        const percentages = data.reduce((acc, prediction) => {
          const category = prediction.prediction.split(' - ')[0];
          const percentage = parseInt(prediction.prediction.split(' - ')[1]);
          if (!acc[category]) {
            acc[category] = percentage;
          } else {
            acc[category] += percentage;
          }
          return acc;
        }, {});
        // Convert percentages object to array of objects for Recharts
        const chartData = Object.keys(percentages).map(category => ({
        value: percentages[category],
          name: category,
        }));
        console.log(chartData);
        setChartData(chartData);
      })
      .catch(error => {
        console.log(error);
```

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

    return (
      <div>
        <h1>List of all Wastes</h1>
        <ul>
          {predictions.map(prediction => (
            <li key={prediction.id}>
              {prediction.text} - {prediction.prediction}
            </li>
          ))}
        </ul>
        <PieChart width={400} height={400}>
            <Pie data={chartData} dataKey="value" nameKey="name" cx="50%" cy="50%" outerRadius={80} fill="#8884d8" label />
          <Tooltip />
          <Legend />
        </PieChart>
      </div>
    );
  }

  export default PredictionList;

  ./Temp.jsx
  import { useState, useEffect } from 'react';
  import axios from 'axios';
  import { Chart } from "react-google-charts";

  function PredictionList() {
    const [predictions, setPredictions] = useState([]);
    const [chartData, setChartData] = useState([['Category', 'Value']])
```

```
const options = {
  title: "Predictions piechart",
};

useEffect(() => {
  axios.get('http://localhost:8000/api/predictions/')
    .then(response => {
      const data = response.data;
      setPredictions(data);
      // Extract percentages from prediction data
      const percentages = data.reduce((acc, prediction) => {
        const category = prediction.prediction.split(' - ')[0];
        const percentage = parseInt(prediction.prediction.split(' - ')[1]);
        if (!acc[category]) {
          acc[category] = percentage;
        } else {
          acc[category] += percentage;
        }
        return acc;
      }, {});
      console.log(percentages);
      // Convert percentages object to array of objects for Recharts
      const chartData = Object.keys(percentages).map(category => {
      return [category, percentages[category]];
      });
      console.log(chartData);
      setChartData(chartData);
    })
    .catch(error => {
      console.log(error);
    });
}, []);
```

```
  return (
    <div>
      <h1>List of all Wastes</h1>
      <ul>
        {predictions.map(prediction => (
          <li key={prediction.id}>
            {prediction.text} - {prediction.prediction}
          </li>
        ))}
      </ul>
      <Chart
        chartType="PieChart"
        data={chartData}
        options={options}
        width={"100%"}
        height={"400px"}
      />

    </div>
  );
}

export default PredictionList;
```

./Chart.jsx

```
import React from "react";
import GaugeChart from "react-gauge-chart";

const Chart = (props) => {
  const data = props.data;
  const label = data.label;
  const confidence = parseFloat(data.confidence.toFixed(2));
```

```
      console.log(label, confidence);
 return (
  <div className="mt-4">
     <h3 className="text-center text-2xl text-bold">Prediction: <span className="text-bold
uppercase font-extrabold text-4xl underline">{label}</span></h3>
    <GaugeChart
     id="gauge-chart3"
     nrOfLevels={3}
     colors={["#FF5F6D", "#FFC371", "rgb(26 202 26)"]}
     arcWidth={0.3}
     percent={confidence}
     style={{ color: "red" }} // <-- set the color to red
    />
  </div>
 );
};
export default Chart;
```

./NewWaste.jsx

```
import React from 'react'
import P1 from './p1.png'
import P2 from './p2.png'
import P3 from './p3.png'

const NewWaste = () => {
 return (
  <div>
   <section className="text-gray-400  body-font">
    <div className="container px-5 py-24 mx-auto">
     <div className="flex flex-col">
      <div className="h-1 bg-gray-800 rounded overflow-hidden">
       <div className="w-24 h-full bg-green-500"></div>
```

```
          </div>
          <div className="flex flex-wrap sm:flex-row flex-col py-6 mb-12">
            <h1 className="sm:w-2/5 text-black font-medium title-font text-3xl mb-2 sm:mb-
0">Previous Predictions</h1>
          </div>
        </div>
        <div className="flex flex-wrap sm:-m-4 -mx-4 -mb-10 -mt-4">
          <div className="p-4 md:w-1/3 sm:mb-0 mb-6">
            <div className="rounded-lg h-64 overflow-hidden">
              <img alt="content" className="object-cover object-center h-full w-full" src={P2}
/>
            </div>
            <h2 className="text-xl font-medium title-font text-white mt-5">Metal</h2>

          </div>
          <div className="p-4 md:w-1/3 sm:mb-0 mb-6">
            <div className="rounded-lg h-64 overflow-hidden">
              <img alt="content" className="object-cover object-center h-full w-full" src={P3}
/>
            </div>
            <h2 className="text-xl font-medium title-font text-white mt-5">Glass</h2>

          </div>
          <div className="p-4 md:w-1/3 sm:mb-0 mb-6">
            <div className="rounded-lg h-64 overflow-hidden">
              <img alt="content" className="object-cover object-center h-full w-full" src={P1}
/>
            </div>
            <h2 className="text-xl font-medium title-font text-white mt-5">Paper</h2>

          </div>
        </div>
      </div>
```

```
      </section>
    </div>
  )
}

export default NewWaste
```

./Landing.jsx

```
import React from 'react'
import Image from '../components/nareeta-martin-FoG7PKNYjpM-unsplash.jpg'
import { Link } from 'react-router-dom'

const Landing = () => {
  return (
    <div>
      <div>
        <section class="mb-20">
          <div class="px-6 py-12 md:px-12 bg-gray-50 text-gray-800 text-center lg:text-left">
            <div class="container mx-auto xl:px-32">
              <div class="grid lg:grid-cols-2 gap-12 items-center">
                <div class="mt-12 lg:mt-0">
                 <h1 class="text-5xl md:text-6xl xl:text-7xl font-bold tracking-tight mb-12">"Sort Smarter,recycle better - <br /><span class="text-green-600">lets keep our planet greener!"</span></h1>
                  <Link to="/signup" class="inline-block px-7 py-3 mr-2 bg-green-600 text-white font-medium text-sm leading-snug uppercase rounded shadow-md hover:bg-green-700 hover:shadow-lg focus:bg-green-700 focus:shadow-lg focus:outline-none focus:ring-0 active:bg-green-800 active:shadow-lg transition duration-150 ease-in-out" data-mdb-ripple="true" data-mdb-ripple-color="light" href="#!" role="button">SignUp</Link>
                  <Link to="/login" class="inline-block px-7 py-3 bg-transparent text-green-600 font-medium text-sm leading-snug uppercase rounded hover:text-green-700 hover:bg-gray-100 focus:bg-gray-100 focus:outline-none focus:ring-0 active:bg-gray-200 transition duration-150
```

```
ease-in-out" data-mdb-ripple="true" data-mdb-ripple-color="light" href="#!"
role="button">Login</Link>
              </div>
              <div class="mb-12 lg:mb-0">
                <img
                  src={Image}
                  class="w-full rounded-lg shadow-lg"
                  alt=""
                />
              </div>
            </div>
          </div>
        </div>
      </section>
    </div>
  </div>
 )
}

export default Landing;
```

./Login.jsx

```
import React, { useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import { Link } from "react-router-dom";

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();

  const handleEmailChange = (event) => {
```

```
  setEmail(event.target.value);
};

const handlePasswordChange = (event) => {
  setPassword(event.target.value);
};

const handleSubmit = async(event) => {
  event.preventDefault();
  await axios
    .post("http://localhost:8000/api/login/", {
      email: email,
      password: password,
    })
    .then((response) => {
      console.log(response.data);
      navigate("/home");
    })
    .catch((error) => {
      console.log(error);
    });
};

return (
  <div className="relative flex flex-col justify-center min-h-screen overflow-hidden">
    <div className="w-full p-6 m-auto bg-white rounded-md shadow-md lg:max-w-xl">
      <h1 className="text-3xl font-semibold text-center text-green-500 underline">
        Sign in
      </h1>
      <form className="mt-6">
        <div className="mb-2">
          <label
            htmlFor="email"
```

```
            className="block font-semibold text-gray-800 text-2xl"
          >
            Email
          </label>
          <input
            type="email"
            value={email}
            onChange={handleEmailChange}
            className="block w-full px-4 py-2 mt-2 text-green-600 bg-white border rounded-
md focus:border-green-400 focus:ring-green-300 focus:outline-none focus:ring focus:ring-
opacity-40" required
          />
        </div>
        <div className="mb-2">
          <label
            htmlFor="password"
            className="block text-2xl font-semibold text-gray-800"
          >
            Password
          </label>
          <input
            type="password"
            value={password}
            onChange={handlePasswordChange}
            className="block w-full px-4 py-2 mt-2 text-green-600 bg-white border rounded-
md focus:border-green-400 focus:ring-green-300 focus:outline-none focus:ring focus:ring-
opacity-40" required
          />
        </div>
        <Link to='/login' className="text-xl text-purple-600 hover:underline">
          Forget Password?
        </Link>
        <div className="mt-6">
```

```
            <button
              type="submit"
              onClick={handleSubmit}
               className="w-full px-4 py-2 tracking-wide text-white transition-colors duration-
  200 text-xl transform bg-green-500 rounded-md hover:bg-green-400 focus:outline-none
  focus:bg-green-600"
            >
              Login
            </button>
          </div>
        </form>

        <p className="mt-8 text-xl font-light text-center text-gray-700">
          {" "}
          Don't have an account?{" "}
          <Link
            to='/signup'
            className="text-xl font-medium text-purple-600 hover:underline"
          >
            Sign up
          </Link>
        </p>
      </div>
    </div>
  );
};

export default Login;
```

./homepage.css

```
.saa{
   transition-duration: 1500ms ;}
```

7.2 OUTPUT SCREENS

7.2.1 Accuracy Graph:

The accuracy graph shows steady improvement in both training and validation accuracy over 38 epochs. Training accuracy reaches approximately 83%, while validation accuracy peaks around 89%. This indicates effective learning and strong generalization of the model as shown in the below Figure 12: Accuracy graphs of VGG16 Model.

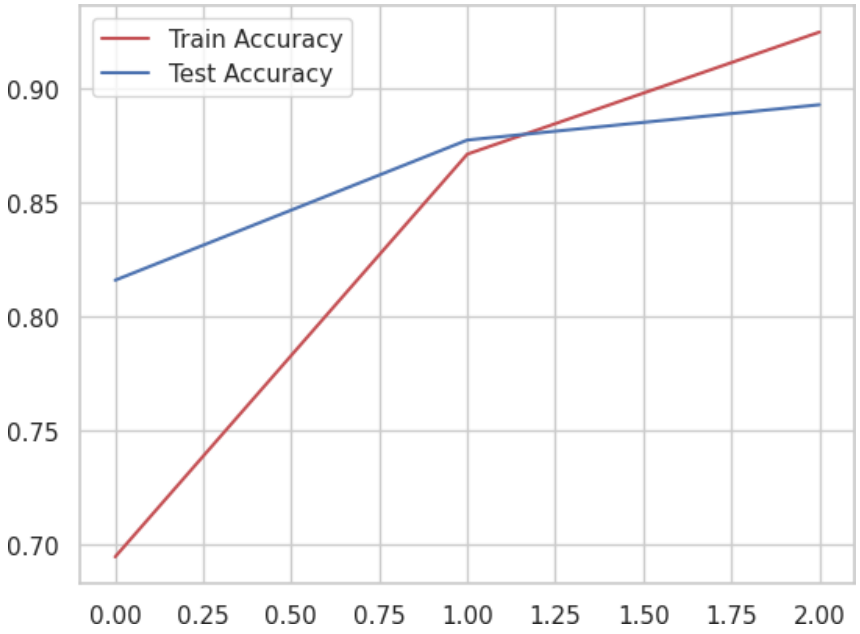

Fig 12 : Accuracy graphs of VGG16 Model

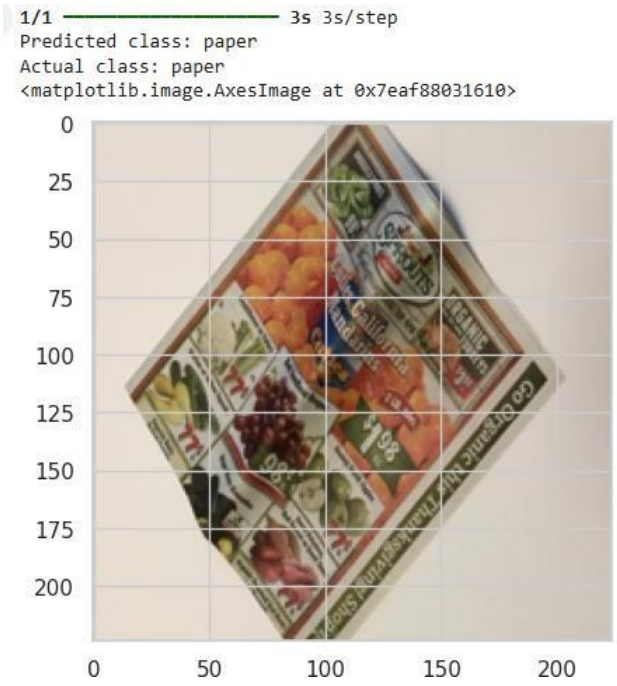

Fig 13:Example of Correct Classification: Predicted and Actual Class are "paper"

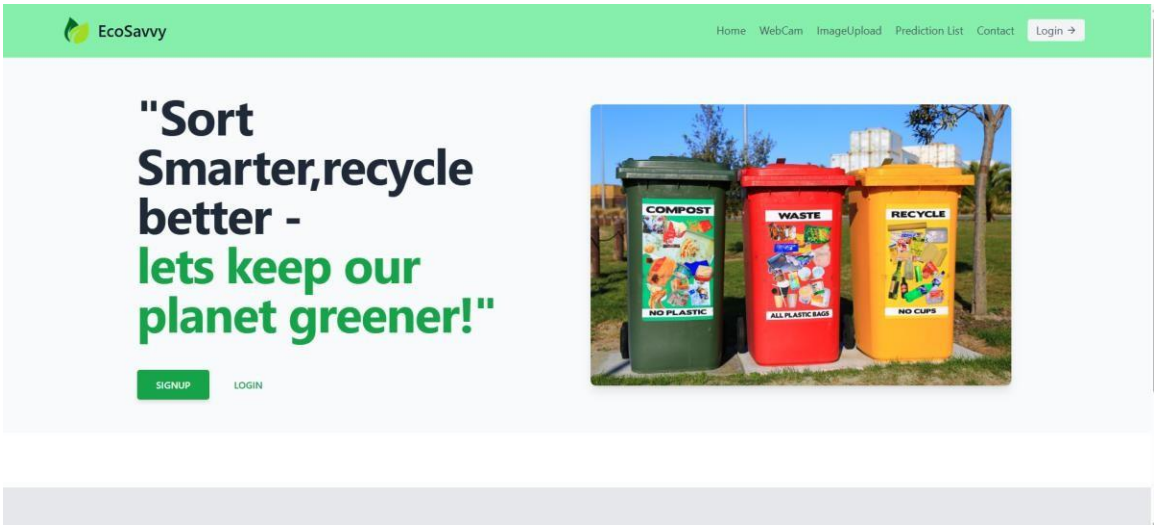

Fig 14:Homepage of the Web Application

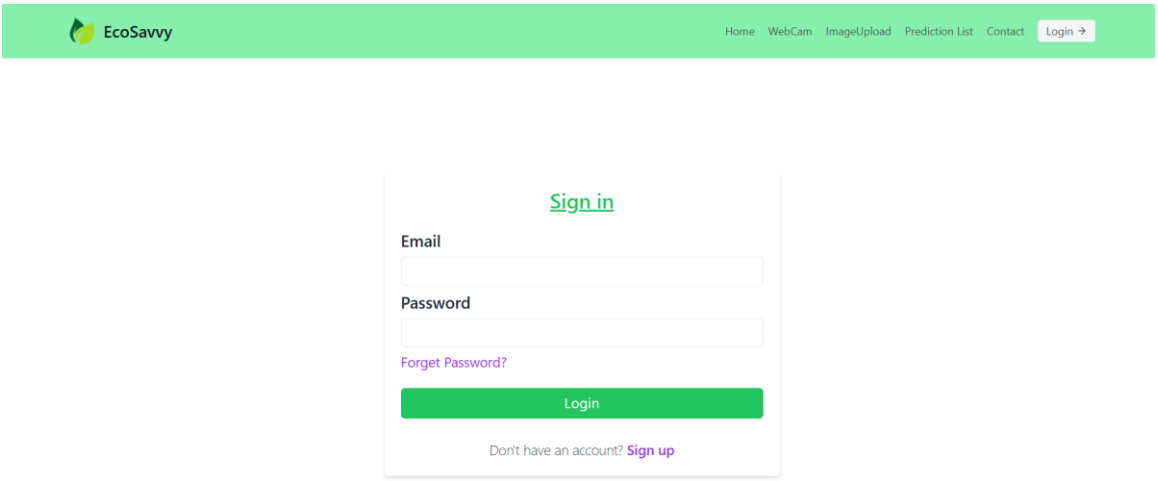

Fig 15:Login Page of the Web Application

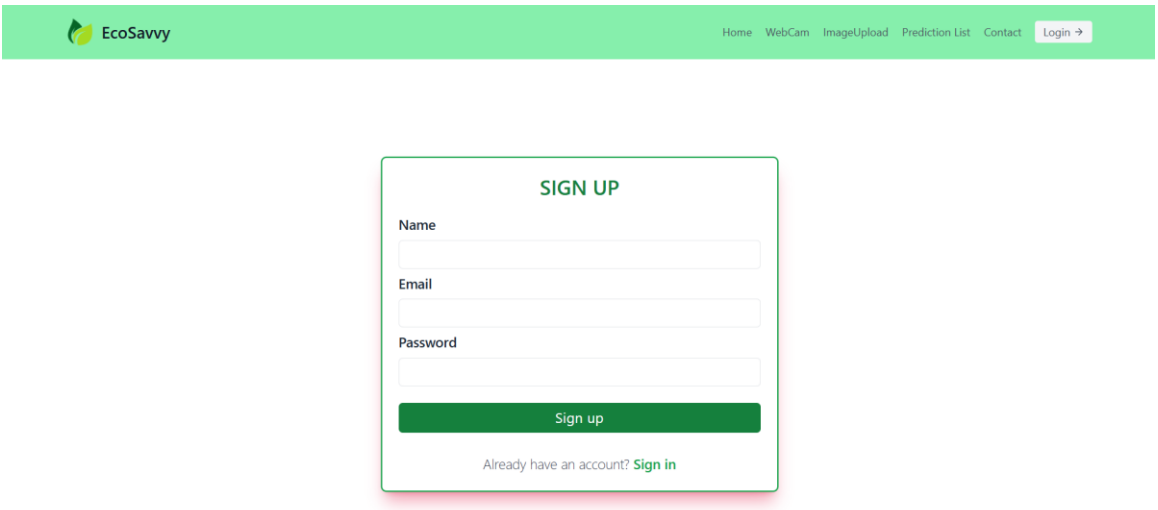

Fig 16 : Sign-Up Page of the Web Application

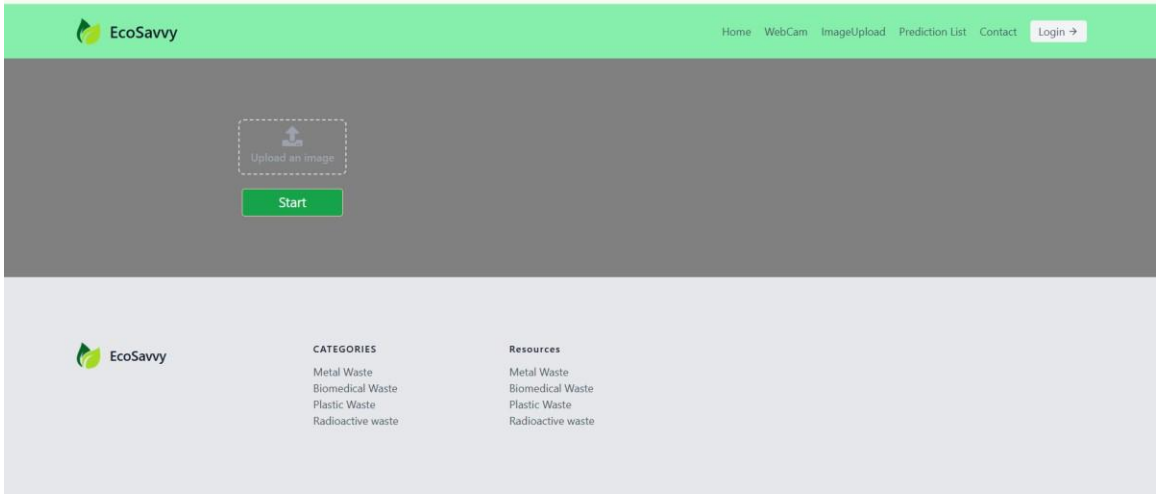

Fig 17 : Image Upload Page of the Web Application

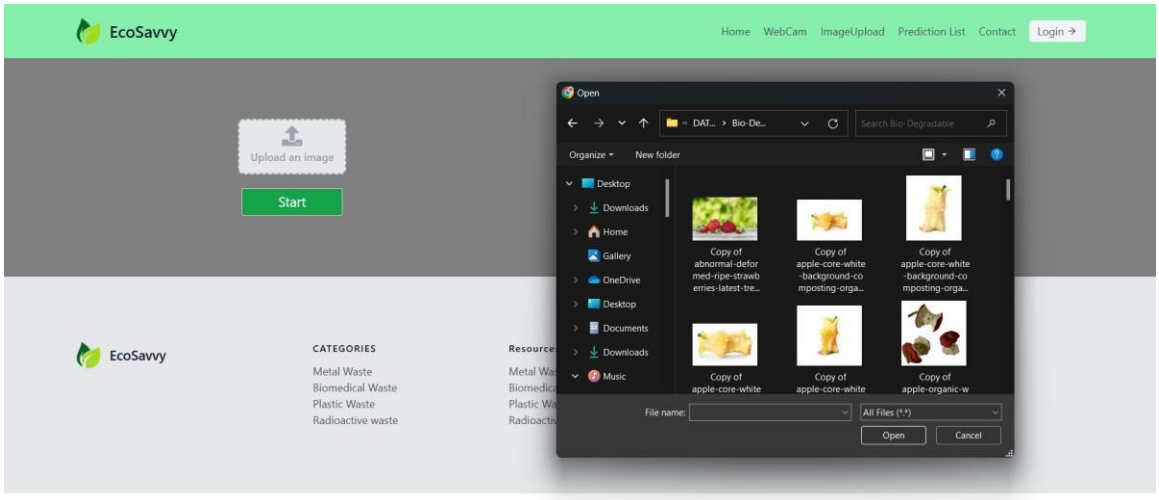

Fig 18: File Upload Dialog Box Overlaying the Image Upload Page

Fig 19:Prediction Result Page of the Web Application Showing "ORGANIC" with 100% Confidence

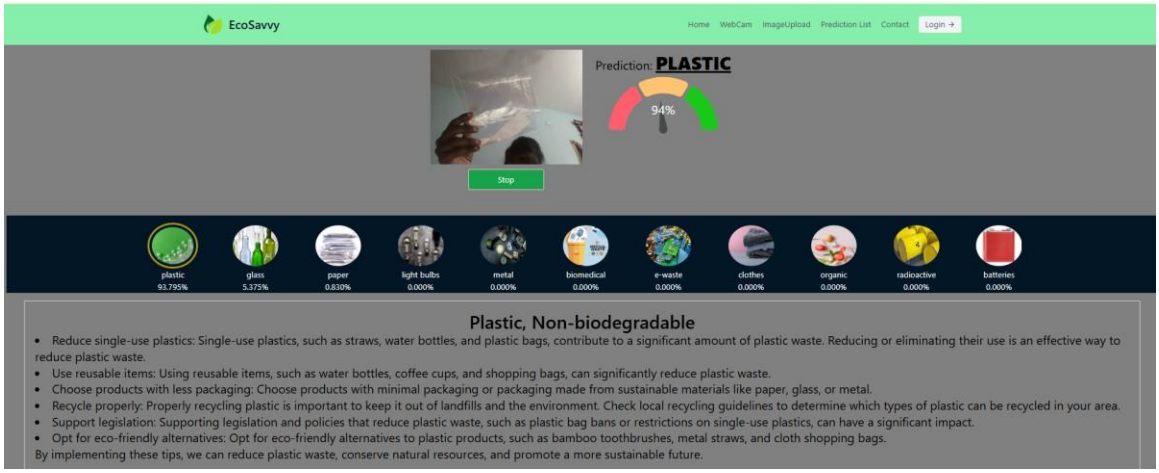

Fig 20 : Prediction Result Page of the Web Application Showing "PLASTIC" with 94% Confidence

CHAPTER 8

CONCLUSION, FUTURE SCOPE AND REFERENCES

8.1 CONCLUSION:

On successful implementation of our project, our trained VGG16-based model accurately classifies waste into categories such as e-waste, batteries, metals, paper, plastic, and more. The system demonstrates strong feature-learning capabilities using a generalized dataset without manual preprocessing. It can analyze thousands of images in real-time, enabling rapid and efficient waste identification. The generated reports help streamline recycling and disposal processes. With further training on diverse images, the model's accuracy and reliability can be enhanced.

8.2 FUTURE SCOPE:

In the future, this intelligent waste management system can be enhanced by integrating IoT-enabled smart bins for real-time monitoring and automated collection. The model can be expanded to support multi-class waste classification, improving recycling efficiency. A mobile application with GPS and user feedback can provide location-based alerts and continuous learning. Integration with cloud and edge computing will ensure scalability and faster processing. Advanced AI models can be adopted to increase accuracy, making the system more robust and suitable for deployment in smart cities.

8.3 REFERENCE:

[1] World Bank, "Trends in Solid Waste Management", Datatopics.worldbank.org, 2019. [Online]. Available:https://datatopics.worldbank.org/what-a-waste/trends-in-solid-waste-management.html [Accessed: 17-Jun-2020].

[2] Wikipedia Contributors, "Convolutional Neural Network", Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network [Accessed: 7-Jun-2020].

[3] Yann Glouche, Paul Couderc, "A Smart Waste Management with Self-Describing Objects", The Second International Conference on Smart Systems, Devices and Technologies (SMART'13), IARIA, June 2013, Rome, Italy. [Online]. Available: https://hal.inria.fr/hal-00924270/document [Accessed: 20-Jun-2020].

[4] Fetulhak Abdurahman, Sileshi Aweke, Chera Assefa, "Automated Garbage Monitoring System Using Arduino", IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 20, No.1, Jan-Feb 2018.

[5] Kannan G. S., Kumar S. S., R. R., M. B., "Automatic Garbage Separation Robot Using Image Processing Technique", International Journal of Scientific and Research Publications, Vol. 6, No. 4, Apr 2016, pp. 326–328. [Online]. Available: http://www.ijsrp.org/research-paper-0416.php?rp=P525265 [Accessed: 20-Feb-2020].

[6] Abdulrahman Alkandari, "Trash Basket Sensor Notification Using Arduino with Android Application", 2018. [Online]. Available: https://www.researchgate.net/publication/324138992 [Accessed: 10-Jun-2020].

[7] Nagaraju Urlagunta, "Smart Dustbin for Economic Growth", 2017. [Online]. Available: https://www.researchgate.net/publication/316700582 [Accessed: 10-Feb-2020].

[8] A. Gitz-Johansen, et al., "A Practical Delivery Route Planning System", 2019 20th IEEE International Conference on Mobile Data Management (MDM), Hong Kong, 2019, pp. 349–350. [Online]. Available: https://ieeexplore.ieee.org/document/8788747 [Accessed: 20-Apr-2020].

[9] A. Renugambal, V. Adilakshmi Kameswari, "Finding Optimal Vehicular Route Based on GPS", International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 5, No. 2, 2014. [Online]. Available: https://pdfs.semanticscholar.org/de0f/04565e702059404a9cb8ec8dc4f58ce3f3e9.pdf [Accessed: 20-Feb-2020].

[10] Keras Team, "Keras Documentation: Layer Activation Functions", 2020. [Online]. Available: https://keras.io/api/layers/activations/ [Accessed: 7-Jun-2020].