ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY, KOTTAYAM

PROJECT REPORT ON

DEEP LEARNING APPROACHES FOR WASTE CLASSIFICATION: A COMPARATIVE STUDY OF RESNET152, INCEPTION V3, AND VGG19

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE IN
COMPUTER APPLICATIONS [TRIPLE MAIN]

Submitted By
Malavika. M
III B.Sc. Computer Applications [Triple Main]
Register No: SB21CA015

Under the guidance of
Ms. Harsha K M

DEPARTMENT OF COMPUTER APPLICATIONS

2021 – 2024

CERTIFICATE

THE FEAR OF THE LORD IS THE BEGINNING OF WISDOM

ST. TERESA'S COLLEGE
(AUTONOMOUS)
ESTD 1925

This is to certify that the project report entitled ***"Deep Learning Approaches For Waste Classification: A Comparative Study Of ResNet152, InceptionV3, VGG19"*** is a bona-fide record of the work done by **MALAVIKA. M (SB21CA015)** during the year 2021-2024 and submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Applications (Triple Main) under Mahatma Gandhi University, Kottayam.

ST. TERESA'S COLLEGE
DEPT. OF COMPUTER
ERNAKULAM

[illegible] 21/3/24

Head of the Department

for [illegible] 21/3/24

Internal Examiner

[illegible] 22.03.24

External Examiner

Date: 21-03-2024

DECLARATION

I, **MALAVIKA M** (Register no: **SB21CA015**), B.Sc. Computer Applications [Triple Main] final year student of St. Teresa's College (Autonomous), Ernakulum, hereby declare that the project submitted named **"Deep Learning Approaches for Waste Classification: A Comparative Study of ResNet152, InceptionV3, VGG19"** for the Bachelor's Degree in Computer Applications [Triple Main] is my original work. I further declare that the said work has not previously been submitted to any other university or academic body.

Date: 21-03-2024

Place: Ernakulam

[illegible]

MALAVIKA

ACKNOWLEDGEMENT

I would like to convey my heartfelt gratitude to **Rev. Dr. Sr. Vinitha** (CSST) Manager, Director **Rev. Sr. Emeline** (CSST), and Principal **Dr. Alphonsa Vijaya Joseph** for providing me with this wonderful opportunity to work on a project with the topic of Deep Learning Approaches for Waste Classification: A Comparative Study Of ResNet152, InceptionV3, VGG19.

I would like to express my profound gratitude to the Head of the Department of Computer Applications **Ms. Remya C J,** my project guide **Ms Harsha K M**, and all other faculty of the department for their contributions to the completion of my project. The completion of the project would not have been possible without their help and insights.

Finally, I take this opportunity to thank all of them who have directly or indirectly helped me with my project.

MALAVIKA M

SYNOPSIS

Efficient waste management is a critical challenge in urbanizing environments, necessitating accurate waste classification for optimal resource utilization and environmental preservation. This research investigates the application of deep learning algorithms, specifically ResNet-152, InceptionV3, and VGG19, for automated waste classification. Leveraging convolutional neural networks trained on extensive image datasets, particularly ResNet-152, we explore their efficacy in classifying diverse waste materials into categories such as paper, plastic, and metal. Through comparative analysis, this study evaluates the performance of ResNet-152 against InceptionV3 and VGG19 in real-time waste classification tasks. This research contributes to advancing computer vision techniques for enhancing waste management efficiency, with implications for sustainable urban development.

CONTENTS

1. INTRODUCTION

1.1 About Project

Rapid global urbanization, population growth, and industrialization have led to the accumulation of enormous amounts of solid waste. In India, waste consists of various materials, and the current waste management status is concerning, with a large portion ending up in landfill sites. The consequences of improper waste disposal are detrimental to the environment, economy, and public health. Thus effective waste classification plays a pivotal role in optimizing resource utilization and preserving the environment. For that effective waste management systems have to be implemented.

Through a comparative analysis of different deep learning models with the classification of wastes into different types, this study aims to evaluate the efficacy of ResNet-152, InceptionV3, and VGG19. By assessing factors such as classification accuracy and computational efficiency, and insights into the strengths and limitations of each model, we can identify the most effective among them.

The findings of this research are expected to contribute to the advancement of computer vision techniques for improving waste management efficiency. Ultimately, the implications extend to promoting sustainable urban development by enabling more effective resource utilization and environmental preservation practices.

1.2 Objectives of the Project

The primary objective of this paper presentation is to compare the performance metrics of three different machine learning models ResNet152, InceptionV3, and VGG19, in the task of classifying different waste materials. Thus also by generating a model that perfectly classifies wastes into different categories this paper compares and analyses the most efficient deep learning model among ResNet152, InceptionV3, and VGG19. Through this paper by accurate waste classification, we may enhance waste management practices by emphasizing segregation techniques and promoting efficient disposal methods.

2. SYSTEM ANALYSIS

2.1 Introduction

System analysis is conducted to comprehensively understand the structure, functionality, and performance of a system or process. It involves examining the existing system's components, interactions, and workflows to identify strengths, weaknesses, and areas for improvement. The results obtained from system analysis provide valuable insights into the current state of the system, including its functionality, usability, and effectiveness. Through system analysis, organizations can pinpoint inefficiencies, bottlenecks, and potential risks, enabling them to make informed decisions for enhancing efficiency, optimizing resources, and achieving strategic objectives.

2.2 Literature Survey

	TITLE	YEAR	AUTHOR(s)	SUMMARY	REFERENCE
1	A Novel YOLOv3 Algorithm-Based Deep Learning Approach for Waste Segregation: Towards Smart Waste Management	2020	Saurav Kumar, Drishti Yadav, Himanshu Gupta, Om Prakash Verma, Irshad Ahmad Ansari and Chang Wook Ahn	The paper has done a comparative analysis of the YOLOv3 algorithm and the YOLOv3-tiny algorithm to analyze which algorithm is more efficient. It was trained using 6437 waste images. It had 6 object classes. The results showed that the YOLOv3 algorithm is more efficient in complex garbage detection.	[1]
2	Waste Classification using ResNet-152	2023	Vijay Gadre, Sanika Sashte,	The pretrained ResNet-152 for waste	[2]

			Apoorva Sarnaik	classification has shown promising results with high levels of classification accuracy over the waste dataset which consists of 5078 images and is divided into 9 classes.	
3	A Multi-Crop Disease Detection and Classification Approach Using CNN	2021	Zubair Saeed, Ali Raza, Ans H. Qureshi, and Muhammad Haroon Yousaf	Examines the overall performance of InceptionV3 and ResNet152 shows the improved result by adding a Dence layer to both which results in the outperformance of ResNet over Inception which was trained over 4 different datasets consisting of less than 2000 images.	[3]
4	Identification of Leaf Diseases in Potato Crop Using Deep Convolutional Neural Networks (DCNNS)	2021	Zubair Saeed, Misha Urooj Khan, Ali Raza	The paper compares the performance of algorithms ResNet-152 and InceptionV3 in a dataset consisting of 1500 images and results in the outperformance of Inception over ResNet.	[4]
5	Open AccessArticle Comparing Inception V3, VGG 16, VGG 19, CNN, and ResNet 50: A Case Study on	2023	Syed Rehan Shah, Salman Qadri, Hadia Bibi, Syed Muhammad	Includes comparison between InceptionV3 and VGG19 under a	[5]

	Early Detection of a Rice Disease		Waqas Shah, Muhammad Imran Sharif, Francesco Marinello	dataset consisting of over 2000 images which results in the outperformance of InceptionV3 over VGG19.	

Table. 1 Literature review.

2.3 Disadvantages of Existing Works

The existing works on classifying images encounter several drawbacks. Firstly, the complexity of implementation possess a significant challenge, potentially hindering the adoption of solutions in real-world settings. Secondly, the availability of a limited dataset restricts the scope and generalizability of the findings, limiting the effectiveness of models trained on such datasets. Lastly, the absence of comparative analyses of algorithms on waste data undermines the ability to identify the most efficient and effective solutions. Addressing these disadvantages is crucial for advancing waste classification research and facilitating the development of robust and widely applicable models.

2.4 Proposed System

This research focuses on exploring the effectiveness of deep learning algorithms, specifically ResNet-152, InceptionV3, and VGG19, in automating waste classification tasks, therefore addressing the limitations of existing works in waste classification. These convolutional neural networks (CNNs) are trained on an extensive image dataset to accurately categorize various waste materials. Using the acquired dataset wastes can be classified into 9 different classes Aluminium, Carton, Glass, Organic wastes, other plastics, Paper and cardboard, Plastic, textiles, and Wood. The same dataset is used to train the three above-mentioned deep learning models. By comparing the performance metrics of these three models, the study aims to identify the most efficient deep learning architecture among them.

To overcome the disadvantages of existing works, the proposed system employs methods that reduce complexity, making it more accessible and practical for implementation in real-world scenarios. Additionally, the system utilizes a diverse dataset, ensuring that the models are trained on a wide range of waste materials, thus enhancing their generalizability and effectiveness.

Thus the proposed system endeavors to elevate waste management practices by prioritizing precise waste classification, segregation, and efficient disposal methods. By tackling the deficiencies observed in current methodologies and presenting a robust and comprehensive approach to waste classification, this system can substantially refine waste management processes and contribute to environmental sustainability.

2.5 Operating System

An operating system (OS) is a software program that manages computer hardware resources and provides common services for computer programs. It acts as an intermediary between computer hardware and software applications, providing an interface for software developers to interact with the hardware. It simply provides an environment within which other programs can do useful work.

The operating system that we have used for the completion of our project is Windows 11. However, the proposed research also works well in Windows 8 and higher OS

2.6 Languages and Packages Installed

- *Jupyter:* Jupyter is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It's widely used for interactive computing and data analysis.
- *Python:* Python is a popular programming language known for its simplicity and readability. It's widely used in various fields such as web development, data analysis, artificial intelligence, and scientific computing. Python's straightforward syntax makes it accessible to beginners and powerful enough for professionals to build complex applications.
- *NumPy:* NumPy is a Python library for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is the foundation for many other Python libraries in the scientific computing ecosystem.
- *TensorFlow:* TensorFlow is an open-source machine learning framework developed by Google. It's widely used for building and training deep learning models, including neural networks. TensorFlow provides a flexible and scalable platform for developing machine learning and artificial intelligence applications, with support for both research and production-level deployments.

- *Keras:* Keras is a high-level neural network API written in Python. It provides a user-friendly interface for building, training, and deploying deep learning models. Keras allows you to define neural networks using a simple, modular approach, making it easy to experiment with different architectures and techniques. It can run on top of TensorFlow, as well as other deep learning frameworks.
- *Transfer Learning:* Transfer learning is a machine learning technique where a model trained on one task or dataset is reused as the starting point for a different but related task. Instead of training a model from scratch, transfer learning leverages the knowledge and features learned by a pre-trained model, which can significantly reduce training time and resource requirements. It's commonly used in deep learning for tasks such as image classification, object detection, and natural language processing.

2.7 Methodology

The method consists of collecting data, selecting an appropriate method and algorithm for data analysis, optimizing the performance, training, evaluating the model, and finally visualizing it.

2.7.1 Data Collection:

The dataset is taken from GitHub which comprises a total number of 5078 images. The Image dataset consists of different types of wastes, it is classified into nine classes Aluminium, Cartons, Glasses, Organic waste, other Plastics, paper and cardboard, Plastic, Textile, and wood and it is pre-processed to prepare it for training the machine learning model.

2.7.2. Method Used:

After referring to different papers, it was found for appropriate waste classification the selected models must be first trained and then analyzed depending on different evaluation metrices to access the models' performances. Some common options include accuracy, precision, and recall. Through further research different models namely ResNet-152, InceptionV3, and VGG19 are selected and trained upon the prepared dataset.

2.7.3 ResNet-152:

ResNet-152 is a deep convolutional neural network known for its 152-layer architecture and groundbreaking residual learning approach. It was trained on the ImageNet dataset and has achieved state-of-the-art results in image classification tasks. The key innovation of ResNet-152 lies in its utilization of residual learning, which involves incorporating shortcut connections to bypass one or more layers to ease training and prevent vanishing gradients. The architecture consists of convolutional, batch normalization, and Rectified Linear Unit (ReLU) activation layers, followed by residual blocks. Each residual block contains two convolutional layers with batch normalization and ReLU activation, along with a shortcut connection that adds the input to the output of the block. Additionally, ResNet-152 includes a global average pooling layer and a fully connected layer for classification. With pre-trained models like ResNet-152, researchers can save time and resources by leveraging learned features for transfer learning on new datasets.

2.7.4 InceptionV3:

InceptionV3, developed by Google researchers, is a cutting-edge deep-learning algorithm for image recognition and classification. InceptionV3, like other deep learning models, can be utilized for transfer learning. This involves fine-tuning the pre-trained model on a smaller dataset or a different task. InceptionV3 architecture is characterized by its deep network structure with a high number of layers, enabling it to capture intricate features in images also various modules called inception modules help to capture intricate features in the images. Transfer learning with InceptionV3 enables researchers to leverage the learned features of the model and adapt it to specific classification tasks with less training data.

InceptionV3 has demonstrated state-of-the-art performance on various benchmark datasets, including ImageNet, achieving high accuracy rates in image classification tasks. Its efficient architecture allows it to achieve competitive results while maintaining relatively low computational requirements compared to deeper networks. InceptionV3 is widely used in various applications, including image classification, object detection, and image segmentation.

2.7.5 VGG19:

VGG19, an extension of the VGG16 architecture, is renowned for its simplicity and remarkable performance in image recognition tasks. With 19 layers including convolutional, pooling, and fully connected layers, it achieves high accuracy rates, especially when trained on large datasets like ImageNet. Due to its uniform structure and straightforward design, VGG19 is easy to implement and understand. Similar to other deep learning models, VGG19 can be used for transfer learning. This involves fine-tuning the pre-trained model on a new dataset or task.

Transfer learning with VGG19 allows researchers to leverage the learned features of the model and adapt it to specific classification tasks with less training data. Pre-trained models are readily available, enabling efficient transfer learning for various classification tasks with minimal data requirements. Widely applied in image analysis fields such as classification, detection, and segmentation, VGG19 stands as a reliable choice for diverse deep-learning projects.

2.7.6 Importing and preparing dataset

After accessing the data with the help of mounting Google Drive with Colab the dataset which contains a total of 5078 images belonging to nine different classes of wastes is split into 4571 images for training purposes and 507 for validation tasks.

2.7.7 Optimising training and testing performance:

The code initializes variables for storing class names and the number of classes in the training dataset. Then, it optimizes data loading and preprocessing performance using TensorFlow's AUTOTUNE mechanism, which dynamically adjusts parallel processing. Additionally, it prefetches data from both the training and test datasets, reducing I/O latency and enhancing overall model training and evaluation efficiency.

2.7.8 Data augmentation:

The data augmentation pipeline begins with rescaling, which normalizes pixel values to a range between 0 and 1. This step stabilizes training by ensuring consistent input data. Next, random horizontal and vertical flips are introduced using the RandomFlip layer, diversifying the dataset by creating mirrored versions of the images. Following this, random rotations within a specified range are applied with the RandomRotation layer, enhancing the model's robustness to variations in object orientations. Additionally, the “preprocess_input” function prepares the images for compatibility with the InceptionV3 model, applying necessary preprocessing steps such as mean subtraction and pixel value scaling. Finally, the GlobalAveragePooling2D layer computes average values across feature maps, reducing spatial dimensions while retaining essential information. This compact pipeline efficiently augments and preprocesses input images, enhancing model training and generalization for tasks like image recognition.

2.7.9 Training:

Each model is fine-tuned with different configurations to optimize performance on the specific dataset. Initially, the “tf.keras.applications.ResNet152” function is utilized to instantiate the base model, pre-trained on the ImageNet dataset. The “input_shape” parameter specifies the expected shape of input images, while “weights='imagenet'” loads the pre-trained weights of the ResNet152 model and all layers are frozen to prevent further training of the pre-trained features. A custom classification head consisting of a dropout layer, a flatten layer, and two dense layers is added on top of the ResNet base. The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss, and trained for 10 epochs with early stopping.

InceptionV3 is initialized similarly with pre-trained weights from ImageNet. However, only the top layers of the model are made trainable, while the earlier layers are kept frozen. Data augmentation techniques are applied to the model output before feeding it into the classification head, which includes dense layers and dropout regularization. The model is compiled with an Adam optimizer with a learning rate schedule and trained for 20 epochs.

VGG19 follows a similar approach, with the pre-trained model initialized with ImageNet weights and all layers frozen initially. The classification head consists of flattened and dense layers with dropout regularization. The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss, and trained for 10 epochs.

2.8 Hardware and Software Specifications

Hardware Requirements:

- Main Processor: Intel core i3 or above
- RAM 8 GB or Above
- Keyboard Standard 108 keys
- Mouse 3D Optical mouse
- Monitor 15” Standard
- Hard disk: 10 GB of available disk space minimum or above

Software Requirements:

- Operating System 64-bit Microsoft Windows 8/10/11
- Programming language Python
- Jupyter Notebook
- Development Environment Google Colab – Google Colab used external hardware specifications with System RAM:1.2 / 12.7 GB, GPU RAM, Disk:26.3 / 78.2 GB.

3. SYSTEM DESIGN

3.1 Introduction

In the system design, the architecture and functionality of the project are defined. A Data Flow Diagram (DFD) is used for this, which visually illustrates how data moves through the system. The DFD provides a high-level overview of processes, data stores, and data flows, facilitating a clear understanding of the system's operation.

3.2 Data Flow Diagram

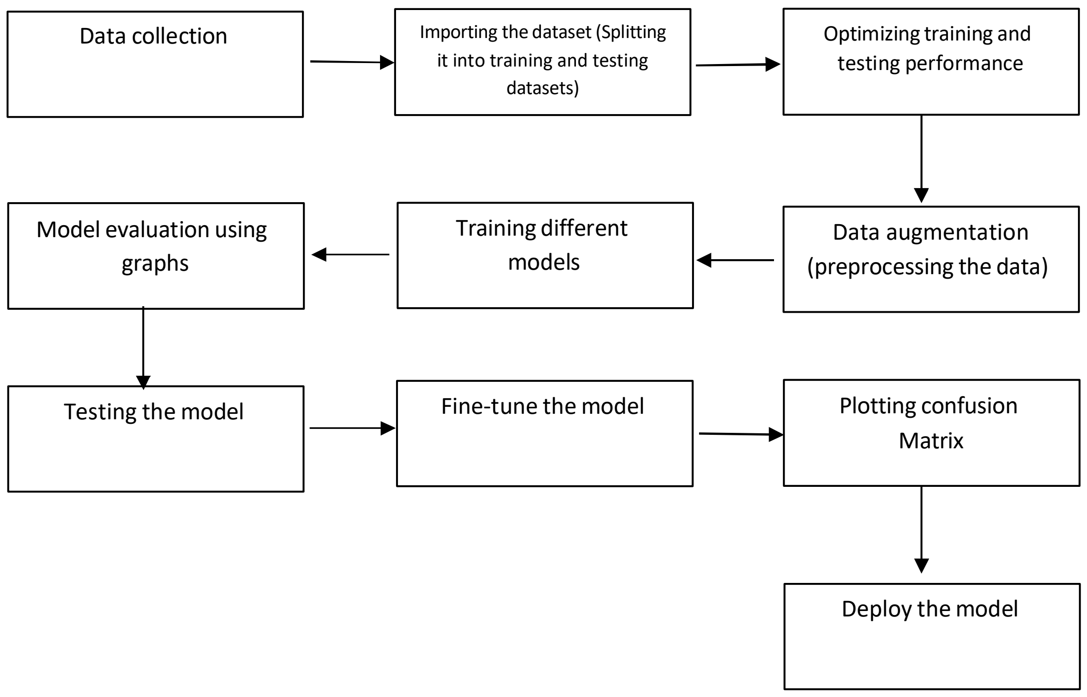

Data collection
Importing the dataset (Splitting it into training and testing datasets)
Optimizing training and testing performance
Data augmentation (preprocessing the data)
Training different models
Model evaluation using graphs
Testing the model
Fine-tune the model
Plotting confusion Matrix
Deploy the model

fig. 1 System flow chart.

4. SYSTEM DEVELOPMENT

4.1 Introduction

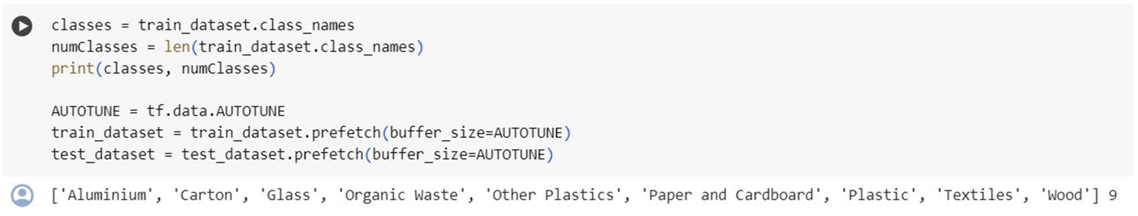
A comprehensive overview of the system development process undertaken as part of the study is depicted in this section. The system development process encompasses the steps involved in conceptualizing, designing, implementing, and evaluating the developed system. This section serves to elucidate the overall procedures visually.

4.2 Process Description

The dataset stored in Google Drive is accessed by mounting the drive. Then it imports the necessary libraries and unzips the dataset and the dataset is loaded and split into training and testing datasets. augmentation techniques like random flipping and rotation are applied to the training dataset to enhance model generalization. The models are trained on the training data for a specified number of epochs with early stopping to prevent overfitting. Training and validation accuracy and loss are monitored during training. The training and validation accuracy/loss are plotted to visualize model performance. The models are tested by importing new images. Finally, a confusion matrix is generated to evaluate the model's performance on the test dataset which provides insights into its classification accuracy.

4.3 Source Code

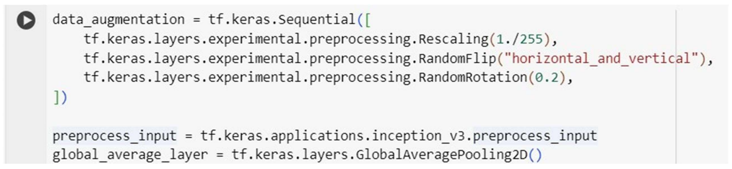
Optimise Training and testing Performance



```
classes = train_dataset.class_names
numClasses = len(train_dataset.class_names)
print(classes, numClasses)

AUTOTUNE = tf.data.AUTOTUNE
train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

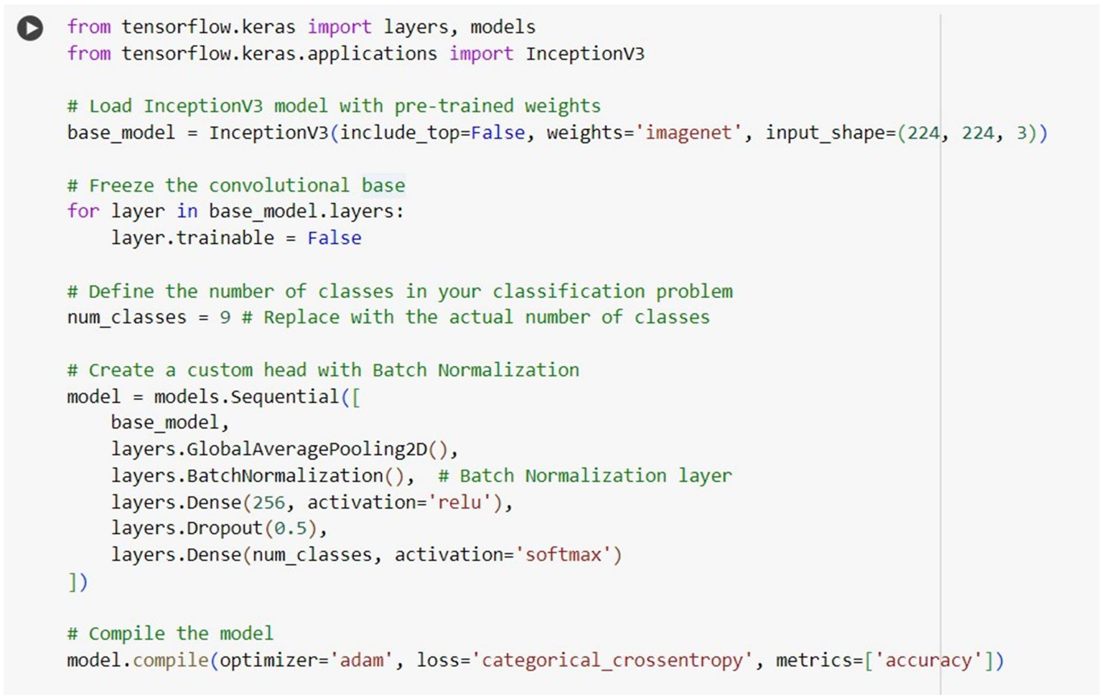
```
['Aluminium', 'Carton', 'Glass', 'Organic Waste', 'Other Plastics', 'Paper and Cardboard', 'Plastic', 'Textiles', 'Wood'] 9
```

Data Augmentation



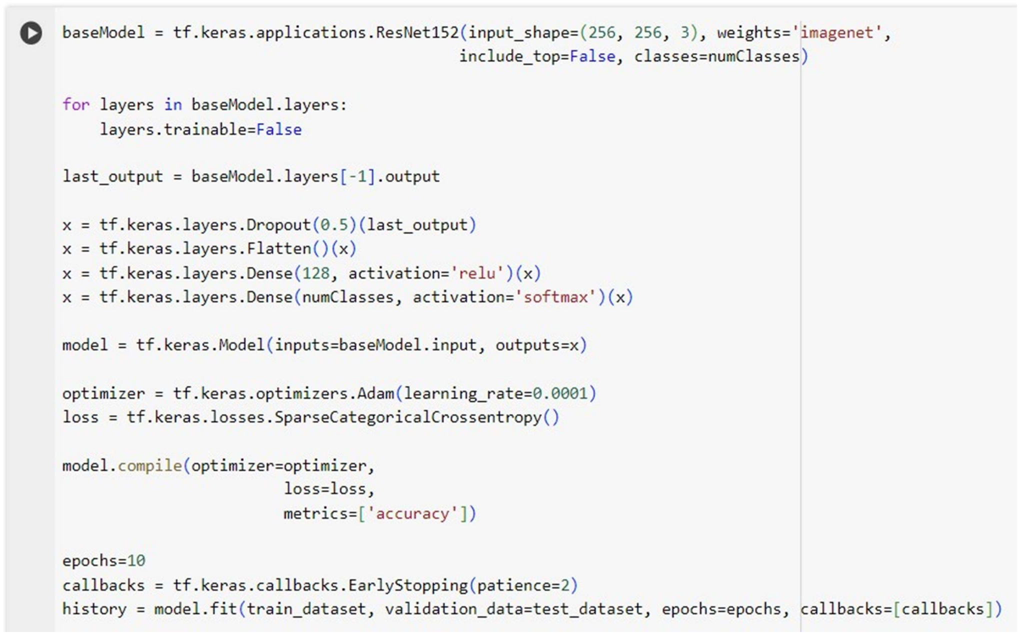
```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.Rescaling(1./255),
    tf.keras.layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
])

preprocess_input = tf.keras.applications.inception_v3.preprocess_input
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
```

Transfer Learning applied on inceptionV3



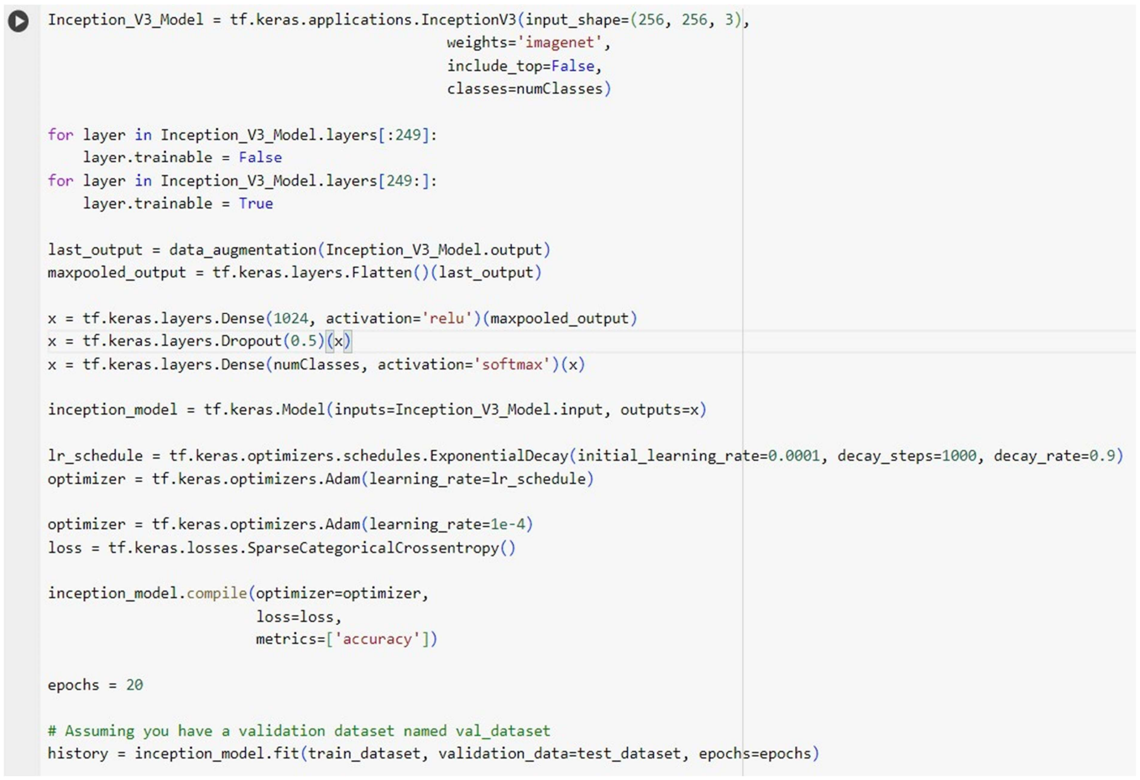
```python
from tensorflow.keras import layers, models
from tensorflow.keras.applications import InceptionV3

# Load InceptionV3 model with pre-trained weights
base_model = InceptionV3(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

# Freeze the convolutional base
for layer in base_model.layers:
    layer.trainable = False

# Define the number of classes in your classification problem
num_classes = 9 # Replace with the actual number of classes

# Create a custom head with Batch Normalization
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.BatchNormalization(),  # Batch Normalization layer
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Training of ResNet-152



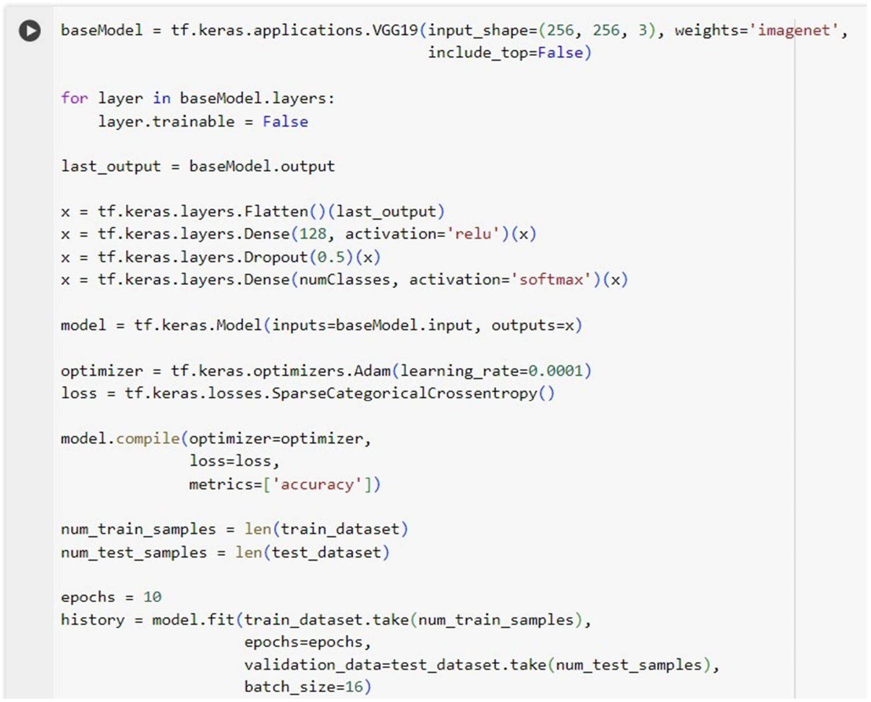
```python
baseModel = tf.keras.applications.ResNet152(input_shape=(256, 256, 3), weights='imagenet',
                                            include_top=False, classes=numClasses)

for layers in baseModel.layers:
    layers.trainable=False

last_output = baseModel.layers[-1].output

x = tf.keras.layers.Dropout(0.5)(last_output)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(128, activation='relu')(x)
x = tf.keras.layers.Dense(numClasses, activation='softmax')(x)

model = tf.keras.Model(inputs=baseModel.input, outputs=x)

optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
loss = tf.keras.losses.SparseCategoricalCrossentropy()

model.compile(optimizer=optimizer,
                        loss=loss,
                        metrics=['accuracy'])

epochs=10
callbacks = tf.keras.callbacks.EarlyStopping(patience=2)
history = model.fit(train_dataset, validation_data=test_dataset, epochs=epochs, callbacks=[callbacks])
```

Training of InceptinV3



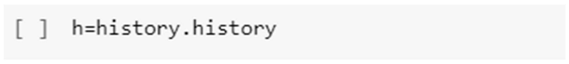
```
Inception_V3_Model = tf.keras.applications.InceptionV3(input_shape=(256, 256, 3),
                                              weights='imagenet',
                                              include_top=False,
                                              classes=numClasses)

for layer in Inception_V3_Model.layers[:249]:
    layer.trainable = False
for layer in Inception_V3_Model.layers[249:]:
    layer.trainable = True

last_output = data_augmentation(Inception_V3_Model.output)
maxpooled_output = tf.keras.layers.Flatten()(last_output)

x = tf.keras.layers.Dense(1024, activation='relu')(maxpooled_output)
x = tf.keras.layers.Dropout(0.5)(x)
x = tf.keras.layers.Dense(numClasses, activation='softmax')(x)

inception_model = tf.keras.Model(inputs=Inception_V3_Model.input, outputs=x)

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=0.0001, decay_steps=1000, decay_rate=0.9)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
loss = tf.keras.losses.SparseCategoricalCrossentropy()

inception_model.compile(optimizer=optimizer,
                        loss=loss,
                        metrics=['accuracy'])

epochs = 20

# Assuming you have a validation dataset named val_dataset
history = inception_model.fit(train_dataset, validation_data=test_dataset, epochs=epochs)
```

Training of VGG-19

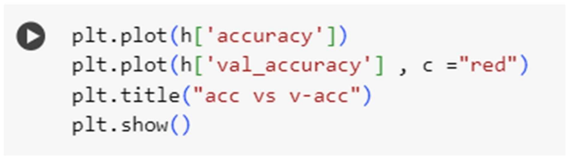


```
baseModel = tf.keras.applications.VGG19(input_shape=(256, 256, 3), weights='imagenet',
                                        include_top=False)

for layer in baseModel.layers:
    layer.trainable = False

last_output = baseModel.output

x = tf.keras.layers.Flatten()(last_output)
x = tf.keras.layers.Dense(128, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
x = tf.keras.layers.Dense(numClasses, activation='softmax')(x)

model = tf.keras.Model(inputs=baseModel.input, outputs=x)

optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
loss = tf.keras.losses.SparseCategoricalCrossentropy()

model.compile(optimizer=optimizer,
              loss=loss,
              metrics=['accuracy'])

num_train_samples = len(train_dataset)
num_test_samples = len(test_dataset)

epochs = 10
history = model.fit(train_dataset.take(num_train_samples),
                    epochs=epochs,
                    validation_data=test_dataset.take(num_test_samples),
                    batch_size=16)
```

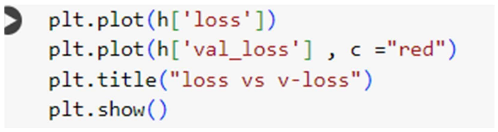
Model evaluation



```
[ ]  h=history.history
```



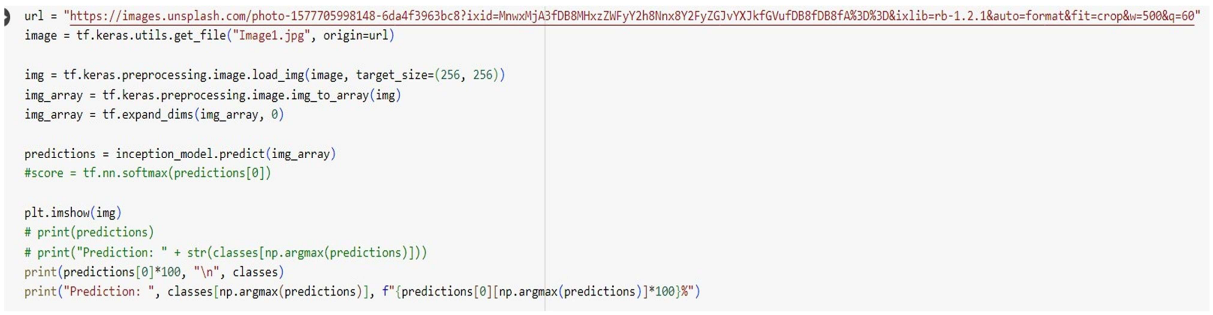
```
plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'] , c ="red")
plt.title("acc vs v-acc")
plt.show()
```



```
plt.plot(h['loss'])
plt.plot(h['val_loss'] , c ="red")
plt.title("loss vs v-loss")
plt.show()
```

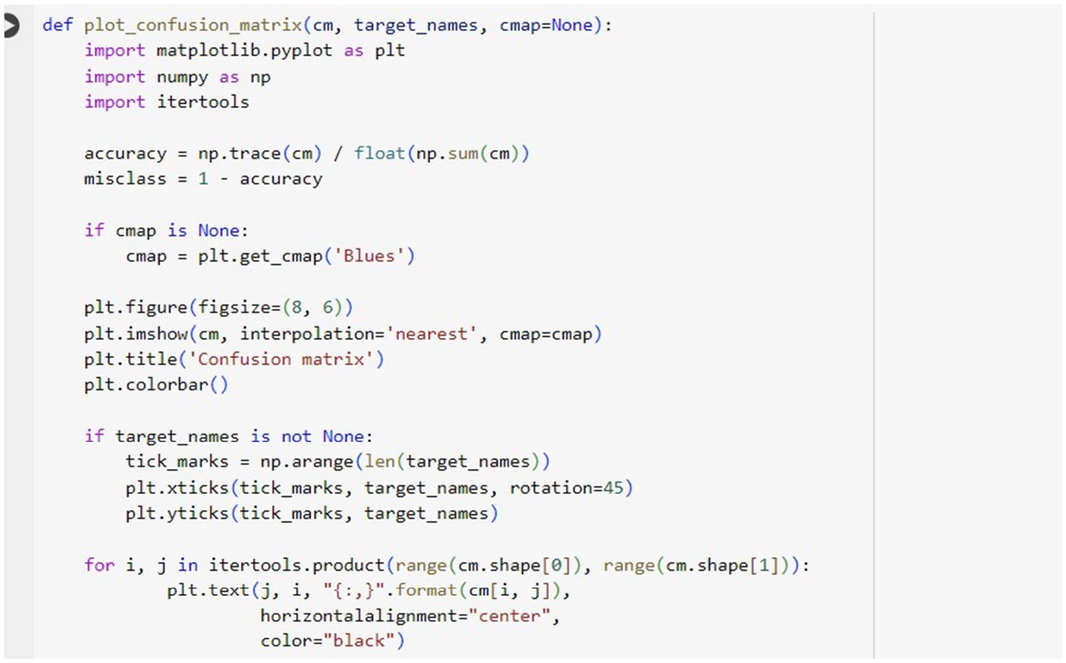
Testing

Testing Of ResNet-152

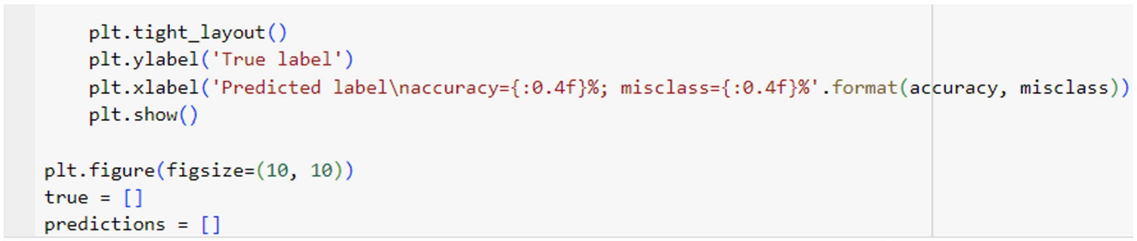


```
url = "https://images.unsplash.com/photo-1577705998148-6da4f3963bc8?ixid=MnwxMjA3fDB8MHxzZWFyY2h8Nnx8Y2FyZGJvYXJkfGVufDB8fDB8fA%3D%3D&ixlib=rb-1.2.1&auto=format&fit=crop&w=500&q=60"
image = tf.keras.utils.get_file("Image1.jpg", origin=url)

img = tf.keras.preprocessing.image.load_img(image, target_size=(256, 256))
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = inception_model.predict(img_array)
#score = tf.nn.softmax(predictions[0])

plt.imshow(img)
# print(predictions)
# print("Prediction: " + str(classes[np.argmax(predictions)]))
print(predictions[0]*100, "\n", classes)
print("Prediction: ", classes[np.argmax(predictions)], f"{predictions[0][np.argmax(predictions)]*100}%")
```

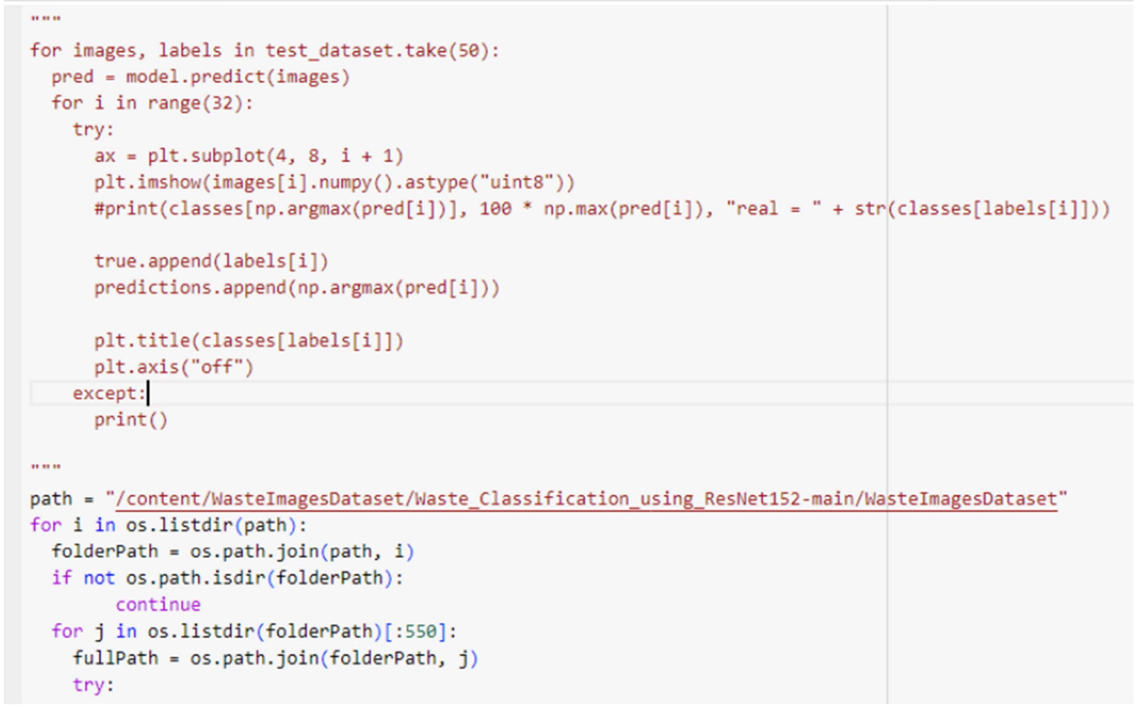
Confusion Matrix



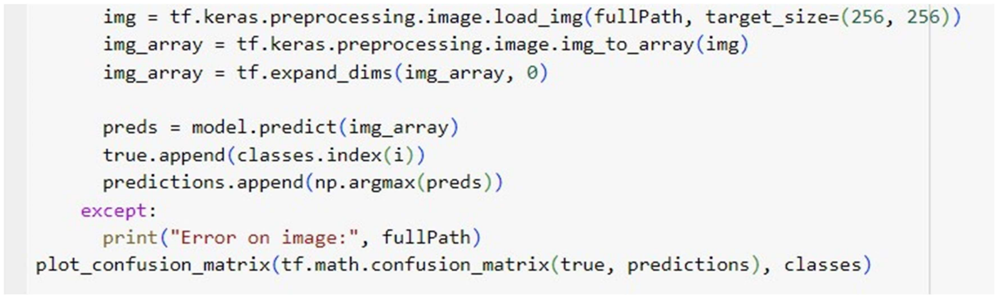
```
def plot_confusion_matrix(cm, target_names, cmap=None):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title('Confusion matrix')
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="black")
```



```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f}%; misclass={:0.4f}%'.format(accuracy, misclass))
    plt.show()

plt.figure(figsize=(10, 10))
true = []
predictions = []
```



```
"""
for images, labels in test_dataset.take(50):
  pred = model.predict(images)
  for i in range(32):
    try:
      ax = plt.subplot(4, 8, i + 1)
      plt.imshow(images[i].numpy().astype("uint8"))
      #print(classes[np.argmax(pred[i])], 100 * np.max(pred[i]), "real = " + str(classes[labels[i]]))

      true.append(labels[i])
      predictions.append(np.argmax(pred[i]))

      plt.title(classes[labels[i]])
      plt.axis("off")
    except:
      print()

"""
path = "/content/WasteImagesDataset/Waste_Classification_using_ResNet152-main/WasteImagesDataset"
for i in os.listdir(path):
  folderPath = os.path.join(path, i)
  if not os.path.isdir(folderPath):
        continue
  for j in os.listdir(folderPath)[:550]:
    fullPath = os.path.join(folderPath, j)
    try:
```



```
      img = tf.keras.preprocessing.image.load_img(fullPath, target_size=(256, 256))
      img_array = tf.keras.preprocessing.image.img_to_array(img)
      img_array = tf.expand_dims(img_array, 0)

      preds = model.predict(img_array)
      true.append(classes.index(i))
      predictions.append(np.argmax(preds))
    except:
      print("Error on image:", fullPath)
plot_confusion_matrix(tf.math.confusion_matrix(true, predictions), classes)
```

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 Introduction

In the system testing and implementation phase, we transition from development to deployment, focusing on validating the functionality, reliability, and performance of the developed system. This phase involves rigorous testing to ensure the system meets specified requirements and successful deployment into the operational environment. Results from testing and implementation activities are documented here.

5.2 Results and Discussion:

This study analyzed the classification of wastes into different classes using three different deep learning algorithms such as ResNet-152, InceptionV3, and VGG19. The evaluation was performed on a waste classification dataset obtained from GitHub. During the overall process of computing, we utilized Python 3 and employed tensorflow preprocessed functions on Google Colab. The training process benefited from a Tesla T4 GPU allocated by Colab. Based on the input data and operations performed, it was observed that ResNet152 outperformed the other 2 networks in the waste classification task. We performed Python's certain built-in augmentation processes on the three models to enhance its accuracy and performance. The augmentation processes involve Rescaling, RandomFlip, and RandomRotation.

The ResNet152 model achieved promising results with an accuracy of approximately 99.41% on the training set and 89.94% on the validation set after 6 epochs of training. It demonstrated rapid convergence and high accuracy, indicating its effectiveness in learning discriminative features from the dataset.

In contrast, the InceptionV3 model exhibited a slower convergence rate and lower accuracy compared to ResNet152, achieving an accuracy of approximately 96.98% on the training set and 51.48% on the validation set after 20 epochs of training. Despite the longer training duration, InceptionV3's performance plateaued at a lower accuracy level, suggesting potential challenges in capturing intricate features from the dataset.

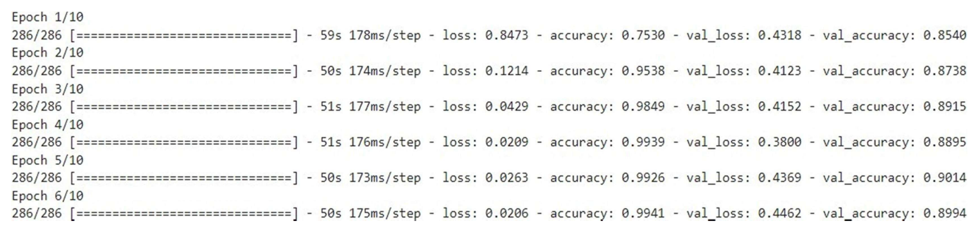
Lastly, the VGG19 model demonstrated competitive performance with an accuracy of approximately 87.38% on the training set and 83.04% on the validation set after 10 epochs of training. While VGG19 showed slightly lower accuracy compared to ResNet152, it achieved comparable results with fewer epochs, indicating its efficiency in learning features from the dataset.

DEEP LEARNING ALGORITHM	**ACCURACY**	**LOSS**	**VALIDATION ACCURACY**	**VALIDATION LOSS**
ResNet152	99.41%	2.06%	89.94%	44.62%
InceptionV3	96.98%	9.31%	51.48%	241.2%
VGG19	87.38%	36.55%	83.04%	71.87%

Table. 2 Performances of different models.

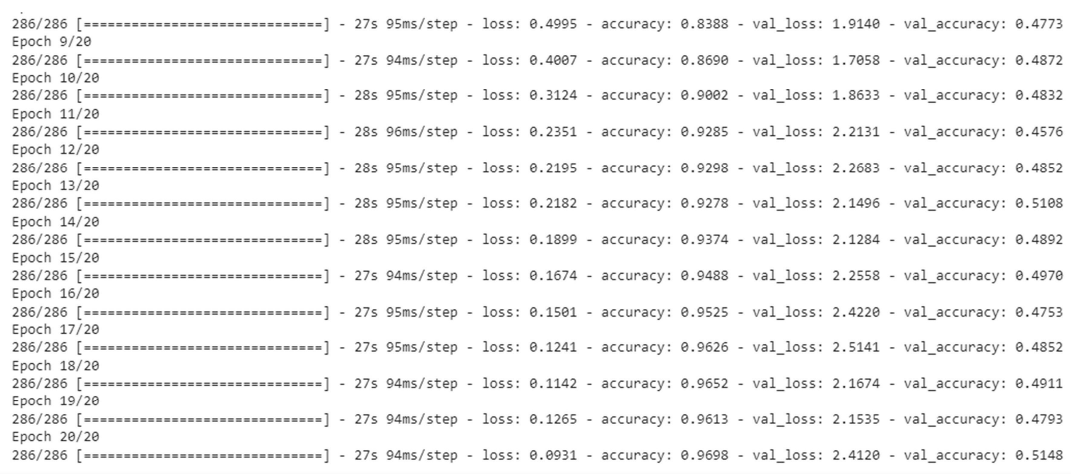
The table above summarizes the overall performance of the applied modules on the selected waste classification dataset. ResNet152 emerged as the top-performing model, achieving an impressive accuracy of 99.41%. This model also demonstrated the lowest loss during training, indicating its ability to effectively learn from the dataset. Furthermore, ResNet152 exhibited strong validation accuracy at 89.94%, with a comparatively lower validation loss of 44.62%. In contrast, InceptionV3 and VGG19 showed lower overall accuracy rates, with InceptionV3 achieving an accuracy of 96.98% and VGG19 achieving 87.38%. However, InceptionV3 suffered from a higher validation loss of 241.2%, suggesting potential overfitting or difficulty in generalization. VGG19, although trailing behind ResNet152 in accuracy, showed moderate validation accuracy at 83.04% and a validation loss of 71.87%. Overall, ResNet152's superior performance in both training and validation metrics underscores its effectiveness in waste image classification tasks.

ResNet-152



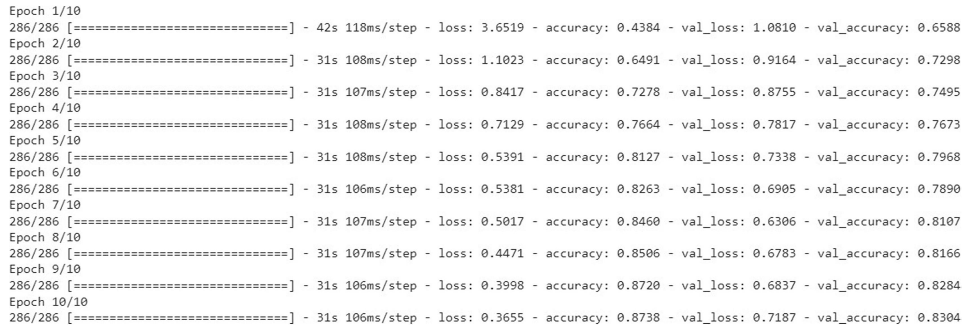
```
Epoch 1/10
286/286 [==============================] - 59s 178ms/step - loss: 0.8473 - accuracy: 0.7530 - val_loss: 0.4318 - val_accuracy: 0.8540
Epoch 2/10
286/286 [==============================] - 50s 174ms/step - loss: 0.1214 - accuracy: 0.9538 - val_loss: 0.4123 - val_accuracy: 0.8738
Epoch 3/10
286/286 [==============================] - 51s 177ms/step - loss: 0.0429 - accuracy: 0.9849 - val_loss: 0.4152 - val_accuracy: 0.8915
Epoch 4/10
286/286 [==============================] - 51s 176ms/step - loss: 0.0209 - accuracy: 0.9939 - val_loss: 0.3800 - val_accuracy: 0.8895
Epoch 5/10
286/286 [==============================] - 50s 173ms/step - loss: 0.0263 - accuracy: 0.9926 - val_loss: 0.4369 - val_accuracy: 0.9014
Epoch 6/10
286/286 [==============================] - 50s 175ms/step - loss: 0.0206 - accuracy: 0.9941 - val_loss: 0.4462 - val_accuracy: 0.8994
```

InceptionV3



```
286/286 [==============================] - 27s 95ms/step - loss: 0.4995 - accuracy: 0.8388 - val_loss: 1.9140 - val_accuracy: 0.4773
Epoch 9/20
286/286 [==============================] - 27s 94ms/step - loss: 0.4007 - accuracy: 0.8690 - val_loss: 1.7058 - val_accuracy: 0.4872
Epoch 10/20
286/286 [==============================] - 28s 95ms/step - loss: 0.3124 - accuracy: 0.9002 - val_loss: 1.8633 - val_accuracy: 0.4832
Epoch 11/20
286/286 [==============================] - 28s 96ms/step - loss: 0.2351 - accuracy: 0.9285 - val_loss: 2.2131 - val_accuracy: 0.4576
Epoch 12/20
286/286 [==============================] - 28s 95ms/step - loss: 0.2195 - accuracy: 0.9298 - val_loss: 2.2683 - val_accuracy: 0.4852
Epoch 13/20
286/286 [==============================] - 28s 95ms/step - loss: 0.2182 - accuracy: 0.9278 - val_loss: 2.1496 - val_accuracy: 0.5108
Epoch 14/20
286/286 [==============================] - 28s 95ms/step - loss: 0.1899 - accuracy: 0.9374 - val_loss: 2.1284 - val_accuracy: 0.4892
Epoch 15/20
286/286 [==============================] - 27s 94ms/step - loss: 0.1674 - accuracy: 0.9488 - val_loss: 2.2558 - val_accuracy: 0.4970
Epoch 16/20
286/286 [==============================] - 27s 95ms/step - loss: 0.1501 - accuracy: 0.9525 - val_loss: 2.4220 - val_accuracy: 0.4753
Epoch 17/20
286/286 [==============================] - 27s 95ms/step - loss: 0.1241 - accuracy: 0.9626 - val_loss: 2.5141 - val_accuracy: 0.4852
Epoch 18/20
286/286 [==============================] - 27s 94ms/step - loss: 0.1142 - accuracy: 0.9652 - val_loss: 2.1674 - val_accuracy: 0.4911
Epoch 19/20
286/286 [==============================] - 27s 94ms/step - loss: 0.1265 - accuracy: 0.9613 - val_loss: 2.1535 - val_accuracy: 0.4793
Epoch 20/20
286/286 [==============================] - 27s 94ms/step - loss: 0.0931 - accuracy: 0.9698 - val_loss: 2.4120 - val_accuracy: 0.5148
```

VGG19



```
Epoch 1/10
286/286 [==============================] - 42s 118ms/step - loss: 3.6519 - accuracy: 0.4384 - val_loss: 1.0810 - val_accuracy: 0.6588
Epoch 2/10
286/286 [==============================] - 31s 108ms/step - loss: 1.1023 - accuracy: 0.6491 - val_loss: 0.9164 - val_accuracy: 0.7298
Epoch 3/10
286/286 [==============================] - 31s 107ms/step - loss: 0.8417 - accuracy: 0.7278 - val_loss: 0.8755 - val_accuracy: 0.7495
Epoch 4/10
286/286 [==============================] - 31s 108ms/step - loss: 0.7129 - accuracy: 0.7664 - val_loss: 0.7817 - val_accuracy: 0.7673
Epoch 5/10
286/286 [==============================] - 31s 108ms/step - loss: 0.5391 - accuracy: 0.8127 - val_loss: 0.7338 - val_accuracy: 0.7968
Epoch 6/10
286/286 [==============================] - 31s 106ms/step - loss: 0.5381 - accuracy: 0.8263 - val_loss: 0.6905 - val_accuracy: 0.7890
Epoch 7/10
286/286 [==============================] - 31s 107ms/step - loss: 0.5017 - accuracy: 0.8460 - val_loss: 0.6306 - val_accuracy: 0.8107
Epoch 8/10
286/286 [==============================] - 31s 107ms/step - loss: 0.4471 - accuracy: 0.8506 - val_loss: 0.6783 - val_accuracy: 0.8166
Epoch 9/10
286/286 [==============================] - 31s 106ms/step - loss: 0.3998 - accuracy: 0.8720 - val_loss: 0.6837 - val_accuracy: 0.8284
Epoch 10/10
286/286 [==============================] - 31s 106ms/step - loss: 0.3655 - accuracy: 0.8738 - val_loss: 0.7187 - val_accuracy: 0.8304
```

5.3 Data Visualization

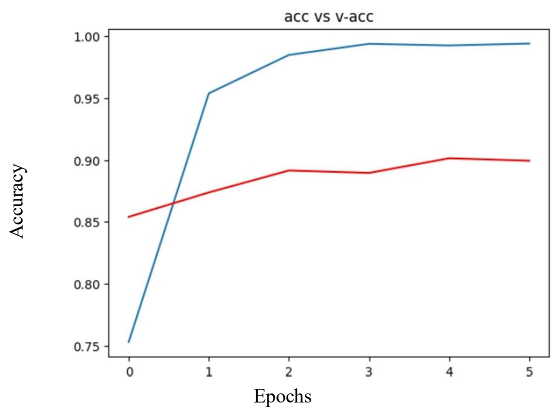


Fig. 2 Graph comparing training and validation accuracy in ResNet152

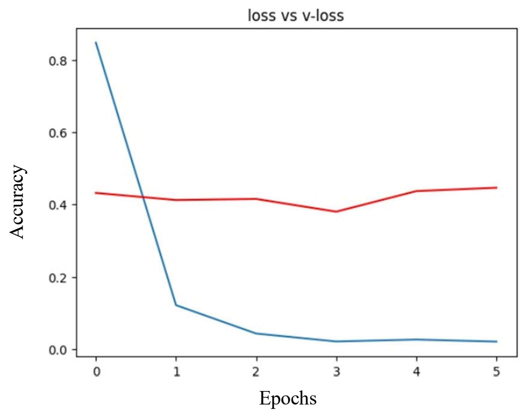


Fig. 3 Graph comparing training and validation loss in ResNet152

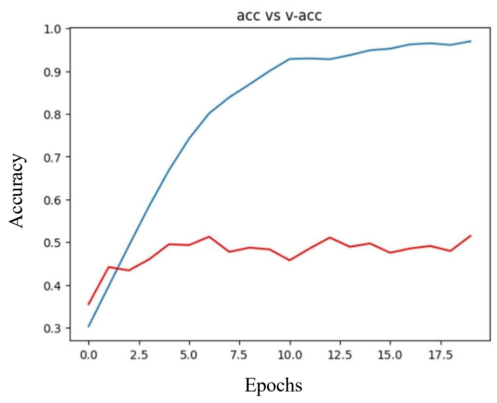


Fig. 4 Graph comparing the training and validation accuracy in InceptionV3

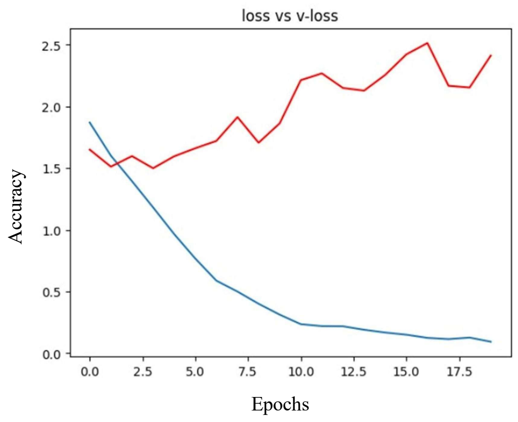


Fig. 5 Graph comparing the training and validation loss in InceptionV3

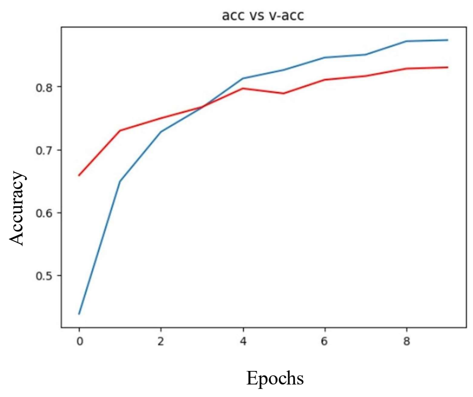


Fig. 6 Graph comparing training and validation accuracy of VGG1

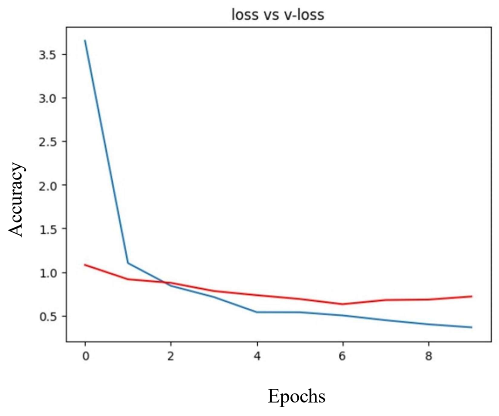
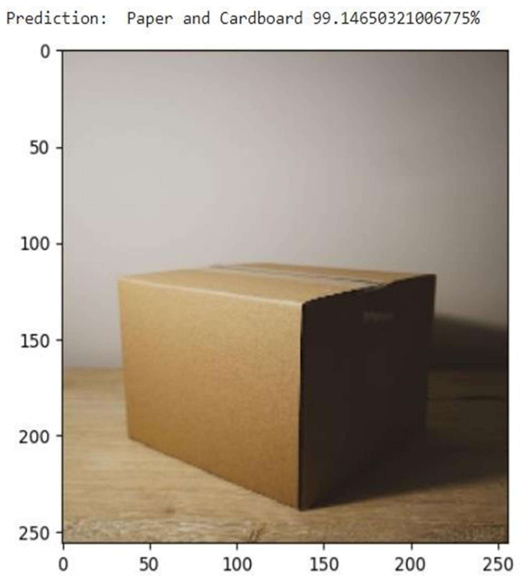


Fig. 7 Graph comparing training and validation loss in VGG19

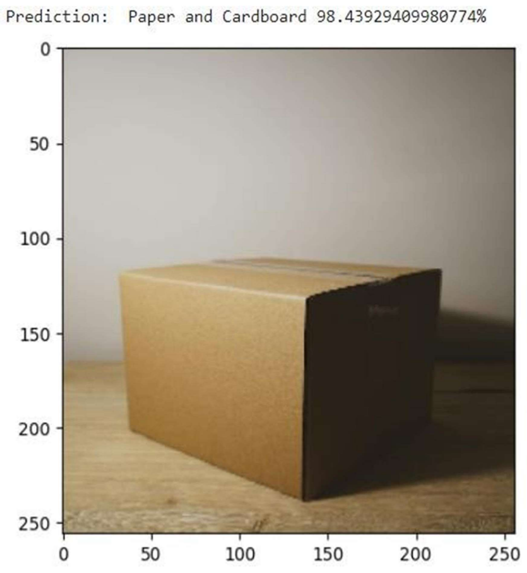
acc vs v-acc graph:
Accuracy - 🟦
Val- Accuracy - 🟥

loss vs v-loss graph:
Loss - 🟦
Val- Loss - 🟥



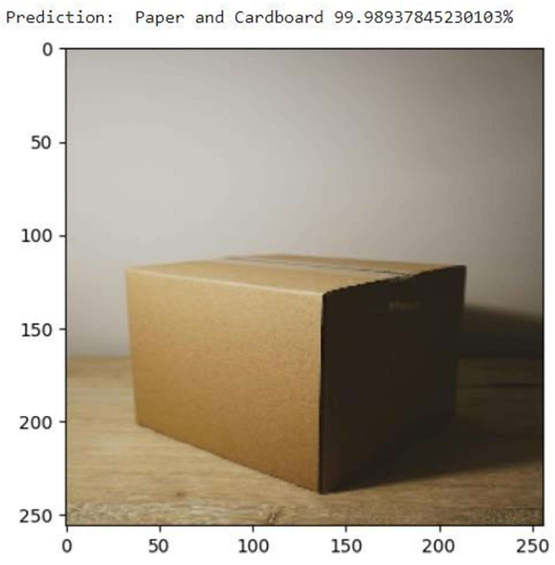
```
Prediction:  Paper and Cardboard 99.14650321006775%
```

Fig. 8 Testing using ResNet152



```
Prediction:  Paper and Cardboard 98.43929409980774%
```

Fig. 9 Testing using InceptionV3



```
Prediction:  Paper and Cardboard 99.98937845230103%
```

Fig. 10 Testing using VGG19

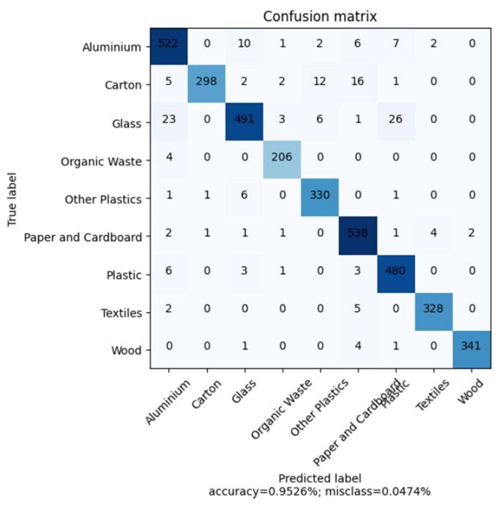


Fig 11. Confusion matrix of resNet152

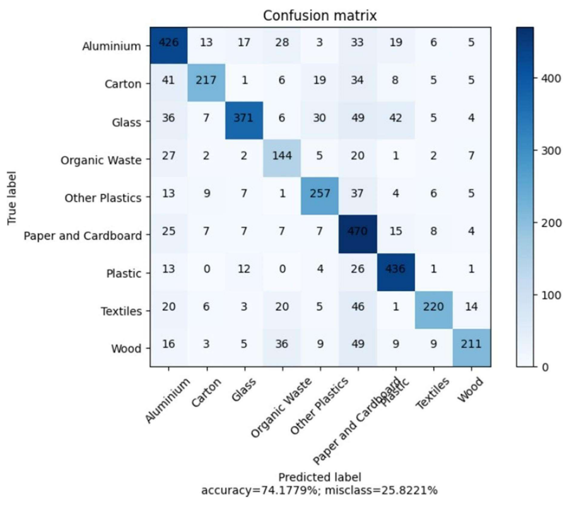


Fig 12. Confusion matrix of InceptionV3

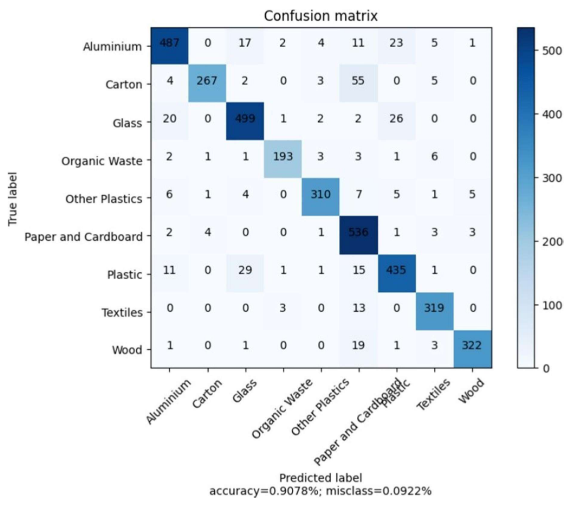


Fig 13. Confusion matrix of VGG19

- ResNet152 (fig 11) achieved the highest prediction accuracy of 95%, with a relatively low misclassification rate of 4%. This indicates robust performance in correctly identifying most classes.
- InceptionV3 (fig 12) exhibited a prediction accuracy of 74% and a higher misclassification rate of 25.8%. While the accuracy is lower compared to ResNet152 and VGG19, the misclassification rate suggests a significant number of incorrect classifications.
- VGG19 (fig 13) demonstrated a prediction accuracy of 90.78% and a misclassification rate of 9%. Despite a slightly lower accuracy compared to ResNet152, VGG19 shows a relatively lower misclassification rate, indicating better overall performance.

ResNet152 shows strong performance in training accuracy and prediction accuracy. InceptionV3 and VGG19 show better performance in training but InceptionV3 performs poorly with testing data.

5.4 Different Testing Techniques Applied

Unit Testing: This code snippet can be tested independently to ensure that each function (e.g., image loading, preprocessing, model prediction) works as expected.

Integration Testing: It's crucial to test the integration of different components, such as image loading, model prediction, and result interpretation, to ensure they work together seamlessly.

System Testing: The entire image classification system, including data fetching, preprocessing, model prediction, and result visualization, should be tested to validate its functionality and performance.

Regression Testing: Any changes or updates made to the code should be tested to ensure that they do not adversely affect existing functionality.

Performance Testing: This involves evaluating the speed and efficiency of the image classification process, including model prediction time and resource consumption.

5.5 Scope for Future Enhancement

Our project offers ample room for improvement in various aspects. We can enhance existing models through optimization techniques to boost classification accuracy and generalization. Exploring new model architectures may unveil more efficient solutions for image classification. Furthermore, expanding and diversifying our dataset will enhance our model's ability to generalize and improve robustness. These efforts will significantly advance the effectiveness and applicability of our image classification system.

6. CONCLUSION

In conclusion, our comparative analysis of deep learning algorithms for waste image classification highlights that ResNet152 outperforms InceptionV3 and VGG19 in terms of both training and testing accuracy. These findings underscore the effectiveness of ResNet152 in accurately identifying and discriminating between different types of waste materials, demonstrating its potential for practical applications in waste management and recycling initiatives.

However, while ResNet152 shows promising results, there remains ample opportunity for further improvement. Enhancing the performance of existing models and exploring new architectures could lead to even more accurate and robust waste classification systems. Additionally, expanding and refining the dataset used for training is crucial for improving the generalization and reliability of the models.

As urbanization and waste management challenges continue to escalate, it is imperative to pursue innovation in this field to create a greener and more sustainable planet. By leveraging deep learning and transfer learning techniques, particularly through implementations like ResNet-152, we can achieve high levels of classification accuracy and contribute meaningfully to the advancement of waste management practices.

7. REFERENCES

1. Kumar, Saurav, et al. "A novel yolov3 algorithm-based deep learning approach for waste segregation: Towards smart waste management." *Electronics* 10.1 (2020): 14.
2. Huang, Rui, et al. "A rapid recognition method for electronic components based on the improved YOLO-V3 network." *Electronics* 8.8 (2019): 825.
3. Gadre, Vijay, et al. "Waste Classification using ResNet-152" IJSREM, 2023
4. Huang, Rui, et al. "A rapid recognition method for electronic components based on the improved YOLO-V3 network." *Electronics* 8.8 (2019): 825.
5. Saeed, Zubair, et al. "Identification of leaf diseases in potato crop using Deep Convolutional Neural Networks (DCNNs)." *2021 16th International Conference on Emerging Technologies (ICET)*. IEEE, 2021.
6. Saeed, Zubair, et al. "A multi-crop disease detection and classification approach using CNN." *2021 International Conference on Robotics and Automation in Industry (ICRAI)*. IEEE, 2021.
7. Kang, Zhuang, et al. "An automatic garbage classification system based on deep learning." *IEEE Access* 8 (2020): 140019-140029.
8. Shah, Syed Rehan, et al. "Comparing Inception V3, VGG 16, VGG 19, CNN, and ResNet 50: A Case Study on Early Detection of a Rice Disease." *Agronomy* 13.6 (2023): 1633.
9. Abhirami, K., et al. "Assessment of municipal solid waste management in Kochi city." *Carbon* (1983): 4-21.
10. Srinilta, Chutimet, and Sivakorn Kanharattanachai. "Municipal solid waste segregation with CNN." *2019 5th International conference on engineering, applied sciences and technology (ICEAST)*. IEEE, 2019.
11. Tiryaki, Burcu, et al. "Dental implant brand and angle identification using deep neural networks." *The Journal of Prosthetic Dentistry* (2023).