Development of AI Algorithms to Detect Explosives in Waste Receptables Using Vision Transformers

Arusha Shahi *[1], Vinod *[2], Abrar *[3], Sayem *[4]

[1] mit240925@stud.mit.edu.au

[2] mit213892@stud.mit.edu.au

[3] mit233042@stud.mit.edu.au

[4] mit232456@stud.mit.edu.au

***Abstract*— In recent times, the production of waste has increased in mostly populated areas that affect the environment. So, sorting of waste management at the initial stage requires a long time to process and recycle it with the items. Isolation of waste is predicted to be 2.2 billion by 2025 in the world whereas plastic and oil fill has coverage of 20% among the landfills. Likewise, the improvised explosive devices from dustbins cause severe effects on people such as spread of disease, increased greenhouse effects, and so on. From the analysis of existing approaches, deep learning approaches and models will enhance a better classification of explosive wastes. The dataset contains more than 2500 images with different such as cardboard, plastic, etc. The vision transformer model (ViT_B16) is developed with the augmented dataset and accuracy is determined for different classification models. The result shows that Model 1 achieves an accuracy of 84% on testing data to predict and classify the waste.**

***Index Terms* – Deep learning, Waste management, Improvised Explosive Devices, Vision Transformer Model, Accuracy**

I. Introduction

Because of outdated infrastructure and technological aspects, waste management has faced significant challenges. Similarly, due to the growth of the urban population, the generation of waste has also increased in numbers. The developing countries are not able to segregate the waste together. With an increased waste, inexperienced workers were recruited to supplement the work. The combination of recycling waste with the ma power becomes an urged challenge [1]. So, the classification method can be deployed automatically to sort the wastes effectively in a shorter time. With the development of automation technologies and artificial intelligence, deep learning models have also become mainstream to classify and recycle waste.

In recent times, the vision transformer model has been a remarkable achievement in natural language processing and machine transactions [2]. The transformer can easily analyse all the words and segregate them in a sequential order. Likewise, this project also invokes the vision transformer model with different optimization and activation functions to classify the waste from the collected data. Based on the experimental analysis, the best and most effective model to classify the waste accordingly. The contribution of the project is:

- Collect the dataset from the GitHub repository to develop the deep learning vision transformer model.
- Analyse the dataset to determine any missing values.
- Develop and build the model to classify the image based on specific classes.
- Analyse the result based on performance metrics like accuracy, confusion matrix, and so on.

II. Literature Review

Explosive of hazardous materials cause significant cause to the environment because of high chemical reactions. The substances can harden and be classified into multiple forms. Likewise, explosive detection in different locations such as airports, government buildings, and public areas able to prevent criminal attacks. The author has exposed the analysis of different technologies through a literature review. The effectiveness of the detection mechanism can increase the detection speed and to ensure the public safety [3]. Through several analyses, artificial intelligence technologies have the potential to detect explosive waste materials in different fields from healthcare to finance. The artificial models support in detecting the explosive materials through training data collected from various resources. The author has also explained future research directions such as AI and machine learning, signal processing techniques, drone establishment, combining multiple detection technologies, and advanced sensor developments.

With the mass production and consumption of waste, the disposal process becomes a critical factor. The waste classification with the traditional method is a time-consuming process. The previous waste classification increases the cost of the waste sorting process and is inefficient. So, the author is supposed to enhance an automatic recyclable waste classification using the convolutional neural network. With the vision transformer model, the performance of the waste classification such as accuracy is determined with the different methods. Based on the GLASSENSE-VISION dataset, the accuracy reached 100% when testing after 80 steps, and the drinking water classification attained an accuracy of 99.29% after 180 steps [4]. The author has also classified the waste and shown in the interface design of an Android Mobile phone. The author has also recommended considering object detection to accurately predict the waste location in the future.

Nowadays, industrial and home solid wastes have been clustered in larger sizes which cannot be unsorted. The consequences of solid wastes lead to environmental pollution so, it is essential to recycle them by segregating them. The author has conducted an experimental analysis by three image classification algorithms. The implementation of such algorithms can reduce environmental pollution and create a safer environment. The author has implemented the CNN model for waste classification, the You Only Look Once algorithm for the object detection model and Faster RCNN for training and classification. From the analysis, the CNN model has an accuracy classification of 80% whereas the YOLO algorithm attains an accuracy classification of 88% with 40% loss [5]. Similarly, the faster RCNN object detection algorithm has received the highest accuracy of 91% with a 16% loss. The author also suggests that future analysis can be based on the combination of two or more computer vision algorithms to classify the waste from the smart dustbin.

Recycling of waste has become a serious cause and issue that affects the atmosphere as well as human health. Before the recycling process, the segregation of waste through an automatic approach can increase efficiency as well as cost-effectiveness. The problem arises with the classification of waste from the main sources such as fabric, plastic, explosive materials, and so on. The author uses the Convolutional Neural Network (CNN) which is a deep learning architecture. The use of the Dropout CNN model has addressed the binary classification problem to classify the waste. Based on the experimental analysis of data collection, the different model such as VGG16, VGG19, MobileNetV2, DenseNet121, and EfficientNet-B0 has an accuracy of 90.60%, 89.27%, 92.73%, and 49.43% respectively. This indicates that the CNN performs better pretraining with an accuracy of 93.28% from the household waste images for landfills [6].

In developing countries, waste management is one of the biggest challenges because it requires a longer process. Manual separation of waste becomes hazardous for the human and it's a time-consuming process. So, the author has approached artificial intelligence which can classify the images according to the waste types.

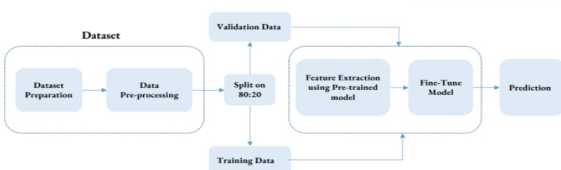

Fig. 1. CNN-based waste classification [7]

The author has used a supervised learning technique called convolutional neural network for image classification. With 3242 images, 103 images were added to the different classes to perform different pre-trained models. The image pre-processing has been conducted to transform the training images such as horizontal flipping and rotation. The image waste classification models such as InceptionResNetV2, ResNet50, and DenseNet201 are selected as the final model. According to the experiment, the DenseNet201 has the highest accuracy of 95.05% when compared to the other two models [7]. In the future, the author has approached the expansion of work by classifying the electronic and medical wastes with advanced models.

III. Materials and Methods

In this section, the dataset description, experimental setup, and model training were provided. The experimental analysis is based on Python programming language.

A. *Data Used for Model Training*

Detection of recycling materials from large bins is an important aspect of today's living environment. The dataset is collected from the open-source repository called "GitHub". The dataset is to build a multiclass classification that has two different categories such as IDE and non-IDE images. The dataset also contains trash data that is taken from the TrashNet dataset.

Dataset Source: https://github.com/jessieAmoakoh/I2Net33

The TrashNet dataset comprised 2527 images with 6 different classes which are shown in Table I.

TABLE I
DATASET DESCRIPTION

S.N.	Classes in dataset	Number of images
1	Glass	501
2	Paper	594
3	Cardboard	403
4	Plastic	482
5	Metal	410
6	Trash	137
7	IED	

The I2net database has also been downloaded from the GitHub repository and the size of the images is converted into 224x224 pixels, so that the deep learning model can be recommended.

B. *Models Used*

1) Vision Transformer Model (ViT): Unlike the traditional convolutional neural networks, the vision transformer (ViT-B16) model is designed to classify the images based on the transformer architecture. This vision transformer model will split the given image into a fixed size of 16x16 pixels [8]. Here each pixel is treated to perform a natural language processing to be embedded into a sequence of patches. The ViT B16 model has several advantages where it can generalize the large dataset to perform pretraining.

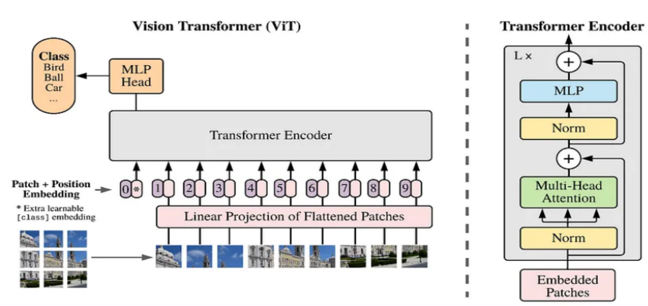

Fig. 2. Vision Transformer Model Architecture [9]

C. Experimental Setup

2)Tools used: Jupyter Notebook tool is used which is an open-source and web-based interactive tool where the code can be generated and visualized effectively. This tool also helps in sharing the document which contains the live code. mostly, this tool is widely used for data analysis and machine learning algorithms to develop the model. The advantage of using the Jupyter Notebook is that it can integrate the code execution for documentation purposes. This tool also supports various programming languages while Python is the most popularly used. Since the tool uses a cell-based structure, the user can easily write the code and run it specifically to determine the output in a visualization or table format. The platform also supports different programming languages such as R, Python, Julia, and so on. The versatility of the tool supports the developers and users to interact and compute the analysis process.

2) Language Used: To develop the model, the Python programming language is used which is simple and readable. The Python programming language can be ideal for both experienced and beginners to write a concise program. This language also supports different programming paradigms such as functional and object-oriented programming. The Python programming language has extensive libraries and frameworks such as pandas, NumPy, TensorFlow, PyTorch, and so on. Similarly, this project also utilizes the TrashNet dataset to develop the vision transformer model to classify the waste bin as real waste or potential explosives.

D. Model Training and Experiments

3) Model Experimentation: The experimental analysis has been conducted where the model is developed and evaluated based on the TrashNet dataset. The experimentation using Jupyter Notebook and Python programming language are shown in the screenshots below:

```
import zipfile
import os

zip_train = zipfile.ZipFile('A2_Data.zip', 'r')
zip_train.extractall('')
zip_train.close()
```

Fig. 3. Unzipping the dataset

The above figure represents the importing of the zip file and os library into the Jupyter notebook platform. the A2_Data.zip contains the images of different classes so, the file is extracted using the extractall () function.

```
import os
import random
import numpy as np
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from vit_keras import vit
from tensorflow.keras import layers, models, optimizers
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
import warnings
warnings.simplefilter("ignore")
```

Fig. 4. Importing Python Libraries

The above figure explains the importing of different libraries like OS, random, TensorFlow, numpy, matplotlib, and warnings. To develop the vision transformer model, these library files are essential to evaluate the model performances like accuracy, classification, confusion matrix, and so on [10].

```
root_dir = "Data"

data = {"Train": [], "Val": [], "Test": []}

for subdir in ["Train", "Val", "Test"]:
  subdir_path = os.path.join(root_dir, subdir)
  for folder in os.listdir(subdir_path):
    folder_path = os.path.join(subdir_path, folder)
    if os.path.isdir(folder_path):
      num_images = len(os.listdir(folder_path))
      data[subdir].append(num_images)
img_count_df = pd.DataFrame(data, index=os.listdir(os.path.join(root_dir, "Train")))
print(img_count_df)
```

```
           Train  Val  Test
cardboard    282   60    61
glass        350   75    76
ied          116   24    26
metal        287   61    62
paper        415   89    90
plastic      337   72    73
trash         95   20    22
```

Fig. 5: Splitting the dataset

The screen represents the code to spit the dataset into the training, validation, and testing dataset. Based on the image count, the data split is examined for the classification of waste based on their categories.

```
class_names = img_count_df.index.tolist()
class_names
```

```
['cardboard', 'glass', 'ied', 'metal', 'paper', 'plastic', 'trash']
```

Fig. 6: Displaying class categories

```
seed = 42
random.seed(seed)
np.random.seed(seed)
```

Fig. 7.Generating random numbers

The figure 6 indicates in displaying the Data Frame where the class names are created. The tolist () function will convert the index into the list format such as cardboard, glass, ied, metal, and so on. Similarly, Figure 7 represents the generation of random numbers using the NumPy library function.

```python
datagen_train = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   featurewise_center=True,
                                   featurewise_std_normalization=True)

datagen_test = ImageDataGenerator(rescale=1./255,
                                  featurewise_center=True,
                                  featurewise_std_normalization=True)
```

```python
img_width = 224
img_height = 224
batch_size = 32
```

Fig. 8. Converting the size of the image to fixed size

Figure 8 indicates the data augmentation using the ImageDataGenerator object for the training and testing process. The different parameter holds different values such as zooming of images, horizontal flips, dividing the data based on standard deviation, shear transformation, and so on. In addition, the height, width, and size of the image are represented as 224, 244, and 32 respectively [11].

```python
train_set = datagen_train.flow_from_directory(
    os.path.join(root_dir, "Train"),
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True,
    seed=seed
)

val_set = datagen_test.flow_from_directory(
    os.path.join(root_dir, "Val"),
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False,
    seed=seed
)


test_set = datagen_test.flow_from_directory(
    os.path.join(root_dir, "Test"),
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False,
    seed=seed
)
```

```
Found 1882 images belonging to 7 classes.
Found 401 images belonging to 7 classes.
Found 410 images belonging to 7 classes.
```

Fig. 9. Classification of dataset

The Figure 9 code snippet shows the generation of data for training, validation, and testing datasets. With the help of an image data generator, the images were resized and categorized according to the different categories.

```python
def create_model(base_model, activation='relu'):
  inputs = layers.Input(shape=(img_width, img_height, 3))
  x = base_model(inputs)
  x = layers.Reshape((1, 1, x.shape[-1]))(x)
  x = layers.GlobalAveragePooling2D()(x)
  x = layers.Dense(256, activation=activation)(x)
  x = layers.Dropout(0.2)(x)
  predictions = layers.Dense(len(class_names), activation='softmax')(x)
  model = models.Model(inputs=inputs, outputs=predictions)
  return model
```

```python
def plot_cm(y_pred):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

```python
def display_accuracy(y_pred):
    accuracy  = accuracy_score(y_true, y_pred)
    print("Test Accuracy : {:.2f}%".format(accuracy * 100))
```

Fig. 10. Developing model with confusion matrix and accuracy

The above code shows the creation of the model to examine the confusion matrix and accuracy level of waste image classification.

3) Model Experimentation: The ViT-B16 model is developed where it takes the input size of the image, pre-trains the model, and classifies it accordingly. The freezing of the model happens when the weight will not be updated during the training process. Similarly, the model will be trained for 10 epochs by assigning the true class labels from the test datasets [12].

```python
vit_b16_model = vit.vit_b16(
    image_size=img_width,
    pretrained= True,
    include_top=False,
    pretrained_top=False
)

for layer in vit_b16_model.layers:
  layer.trainable = False
```

```python
epochs = 10
train_steps = train_set.samples // batch_size
val_steps = val_set.samples // batch_size
y_true = test_set.classes
```

Fig. 11. Developing Vision Transformer Model

4) Model Summary: The summary of the vit_bit16 model is shown in the above figure. Based on the several later types, the output share is based on the size of the images. The total parameter obtained from the model is 85997575 where the trainable and non-trainable parameters are 198919 and s85798656 respectively.

```python
model_1 = create_model(vit_b16_model, activation='PReLU')
```

```python
model_1.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 vit-b16 (Functional)        (None, 768)               85798656

 reshape_1 (Reshape)         (None, 1, 1, 768)         0

 global_average_pooling2d (  (None, 768)               0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               197120

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 7)                 1799

=================================================================
Total params: 85997575 (328.05 MB)
Trainable params: 198919 (777.03 KB)
Non-trainable params: 85798656 (327.30 MB)
_________________________________________________________________
```

Fig. 12. Summary of Model 1

E. *Model Training and Evaluation*

The ViT model with the different optimizers and activation is enhanced to determine the classification of waste. The test accuracy of the Adam optimizer with PReLU activation is 82.93% whereas the optimizer without the PReLU also attains an accuracy of 84.39%. Similarly, the accuracy rate.

```python
model_1.compile(optimizer=optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_model_1 = model_1.fit(
    train_set,
    steps_per_epoch = train_steps,
    epochs=epochs,
    validation_data=val_set,
    validation_steps = val_steps
)
```

```
Epoch 1/10
58/58 [==============================] - 660s 11s/step - loss: 0.7781 - accuracy: 0.7200 - val_loss: 0.5384 - val_accuracy: 0.8281
Epoch 2/10
58/58 [==============================] - 673s 12s/step - loss: 0.3419 - accuracy: 0.8816 - val_loss: 0.4546 - val_accuracy: 0.8464
Epoch 3/10
58/58 [==============================] - 673s 12s/step - loss: 0.2796 - accuracy: 0.8995 - val_loss: 0.4818 - val_accuracy: 0.8542
Epoch 4/10
58/58 [==============================] - 679s 12s/step - loss: 0.2101 - accuracy: 0.9276 - val_loss: 0.4777 - val_accuracy: 0.8438
Epoch 5/10
58/58 [==============================] - 671s 12s/step - loss: 0.1776 - accuracy: 0.9395 - val_loss: 0.5061 - val_accuracy: 0.8594
Epoch 6/10
58/58 [==============================] - 682s 12s/step - loss: 0.1418 - accuracy: 0.9481 - val_loss: 0.4722 - val_accuracy: 0.8802
Epoch 7/10
58/58 [==============================] - 662s 11s/step - loss: 0.1243 - accuracy: 0.9551 - val_loss: 0.5858 - val_accuracy: 0.8620
Epoch 8/10
58/58 [==============================] - 654s 11s/step - loss: 0.1291 - accuracy: 0.9573 - val_loss: 0.4739 - val_accuracy: 0.8646
Epoch 9/10
58/58 [==============================] - 684s 12s/step - loss: 0.1102 - accuracy: 0.9659 - val_loss: 0.5334 - val_accuracy: 0.8516
Epoch 10/10
58/58 [==============================] - 647s 11s/step - loss: 0.1044 - accuracy: 0.9681 - val_loss: 0.4586 - val_accuracy: 0.8724
```

Fig. 13. Model 1 Adam optimizer, PReLU activation function

```
[ ] model_2 = create_model(vit_b16_model)

model_2.compile(optimizer=optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_model_2 = model_2.fit(
    train_set,
    steps_per_epoch = train_steps,
    epochs=epochs,
    validation_data=val_set,
    validation_steps = val_steps
)

Epoch 1/10
58/58 [==============================] - 659s 11s/step - loss: 0.7662 - accuracy: 0.7286 - val_loss: 0.5808 - val_accuracy: 0.8307
Epoch 2/10
58/58 [==============================] - 639s 11s/step - loss: 0.3659 - accuracy: 0.8670 - val_loss: 0.5209 - val_accuracy: 0.8516
Epoch 3/10
58/58 [==============================] - 653s 11s/step - loss: 0.2588 - accuracy: 0.9081 - val_loss: 0.5021 - val_accuracy: 0.8568
Epoch 4/10
58/58 [==============================] - 662s 11s/step - loss: 0.2003 - accuracy: 0.9249 - val_loss: 0.5160 - val_accuracy: 0.8568
Epoch 5/10
58/58 [==============================] - 1340s 23s/step - loss: 0.1801 - accuracy: 0.9368 - val_loss: 0.4603 - val_accuracy: 0.8802
Epoch 6/10
58/58 [==============================] - 686s 12s/step - loss: 0.1545 - accuracy: 0.9470 - val_loss: 0.4920 - val_accuracy: 0.8672
Epoch 7/10
58/58 [==============================] - 683s 12s/step - loss: 0.1498 - accuracy: 0.9449 - val_loss: 0.5032 - val_accuracy: 0.8568
Epoch 8/10
58/58 [==============================] - 641s 11s/step - loss: 0.1172 - accuracy: 0.9622 - val_loss: 0.5476 - val_accuracy: 0.8594
Epoch 9/10
58/58 [==============================] - 633s 11s/step - loss: 0.1022 - accuracy: 0.9665 - val_loss: 0.5038 - val_accuracy: 0.8672
Epoch 10/10
58/58 [==============================] - 674s 12s/step - loss: 0.0860 - accuracy: 0.9741 - val_loss: 0.4911 - val_accuracy: 0.8802
```

Fig. 14. Model 2 Adam optimizer, Without PReLU activation function

```
[ ] model_3 = create_model(vit_b16_model, activation='PReLU')

[ ] model_3.compile(optimizer=optimizers.SGD(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_model_3 = model_3.fit(
    train_set,
    steps_per_epoch = train_steps,
    epochs=epochs,
    validation_data=val_set,
    validation_steps = val_steps
)

Epoch 1/10
58/58 [==============================] - 705s 12s/step - loss: 2.3415 - accuracy: 0.1686 - val_loss: 1.8851 - val_accuracy: 0.2891
Epoch 2/10
58/58 [==============================] - 691s 12s/step - loss: 1.7948 - accuracy: 0.3146 - val_loss: 1.5683 - val_accuracy: 0.4323
Epoch 3/10
58/58 [==============================] - 668s 12s/step - loss: 1.5518 - accuracy: 0.4297 - val_loss: 1.3846 - val_accuracy: 0.5156
Epoch 4/10
58/58 [==============================] - 668s 12s/step - loss: 1.4248 - accuracy: 0.4859 - val_loss: 1.2508 - val_accuracy: 0.5573
Epoch 5/10
58/58 [==============================] - 661s 11s/step - loss: 1.2908 - accuracy: 0.5481 - val_loss: 1.1648 - val_accuracy: 0.6016
Epoch 6/10
58/58 [==============================] - 673s 12s/step - loss: 1.1996 - accuracy: 0.5708 - val_loss: 1.0914 - val_accuracy: 0.6224
Epoch 7/10
58/58 [==============================] - 663s 11s/step - loss: 1.1143 - accuracy: 0.6086 - val_loss: 1.0320 - val_accuracy: 0.6458
Epoch 8/10
58/58 [==============================] - 713s 12s/step - loss: 1.0414 - accuracy: 0.6272 - val_loss: 0.9824 - val_accuracy: 0.6823
Epoch 9/10
58/58 [==============================] - 1419s 24s/step - loss: 0.9831 - accuracy: 0.6595 - val_loss: 0.9443 - val_accuracy: 0.6979
Epoch 10/10
58/58 [==============================] - 1393s 24s/step - loss: 0.9466 - accuracy: 0.6757 - val_loss: 0.9101 - val_accuracy: 0.7109
```

Fig. 15. Model 3 SGD optimizer, PReLU activation function

```
model_4 = create_model(vit_b16_model)

[ ] model_4.compile(optimizer=optimizers.SGD(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

history_model_4 = model_4.fit(
    train_set,
    steps_per_epoch = train_steps,
    epochs=epochs,
    validation_data=val_set,
    validation_steps = val_steps
)

Epoch 1/10
58/58 [==============================] - 1478s 25s/step - loss: 2.1531 - accuracy: 0.1995 - val_loss: 1.7099 - val_accuracy: 0.3620
Epoch 2/10
58/58 [==============================] - 1041s 18s/step - loss: 1.7816 - accuracy: 0.3378 - val_loss: 1.4424 - val_accuracy: 0.4844
Epoch 3/10
58/58 [==============================] - 615s 11s/step - loss: 1.5391 - accuracy: 0.4216 - val_loss: 1.2789 - val_accuracy: 0.5443
Epoch 4/10
58/58 [==============================] - 617s 11s/step - loss: 1.3881 - accuracy: 0.4865 - val_loss: 1.1645 - val_accuracy: 0.5938
Epoch 5/10
58/58 [==============================] - 682s 12s/step - loss: 1.2832 - accuracy: 0.5389 - val_loss: 1.0782 - val_accuracy: 0.6198
Epoch 6/10
58/58 [==============================] - 669s 12s/step - loss: 1.1849 - accuracy: 0.5984 - val_loss: 1.0098 - val_accuracy: 0.6484
Epoch 7/10
58/58 [==============================] - 665s 11s/step - loss: 1.1200 - accuracy: 0.5973 - val_loss: 0.9587 - val_accuracy: 0.6797
Epoch 8/10
58/58 [==============================] - 692s 12s/step - loss: 1.0457 - accuracy: 0.6368 - val_loss: 0.9149 - val_accuracy: 0.6875
Epoch 9/10
58/58 [==============================] - 849s 15s/step - loss: 0.9952 - accuracy: 0.6622 - val_loss: 0.8813 - val_accuracy: 0.6953
Epoch 10/10
58/58 [==============================] - 1000s 17s/step - loss: 0.9484 - accuracy: 0.6838 - val_loss: 0.8508 - val_accuracy: 0.7109
```

Fig. 16. Model 4 SGD optimizer, Without PReLU activation function

IV. Results and Discussion

A. Model 1: Adam optimizer, PReLU activation function

5) Test Accuracy: The accuracy of Model 1 (Adam optimizer with PReLU activation) is examined to be 82.93%. This model has a better optimization to classify the wastes.

```
[ ] y_pred = model_1.predict(test_set)
    y_pred_model_1 = np.argmax(y_pred, axis=1)

    display_accuracy(y_pred_model_1)
    plot_cm(y_pred_model_1)
    classification_report(y_true, y_pred_model_1, target_names=class_names)

    plot_loss(history_model_1)
    plot_accuracy(history_model_1)

13/13 [==============================] - 97s 7s/step
Test Accuracy : 82.93%
```

Fig. 17. Accuracy of Model 1

6) Confusion matrix: The confusion matrix shows the classification performance of the different classes such as cardboard, glass, ied, metal, and so on. From the analysis of the matrix, glass is classified 67 times whereas the comparison of plastic and glass is 12 times because of confusion. Similarly, the paper is classified as 87 times and there is a misclassification between the plastic with glass or trash. There is a strong performance which indicates a higher classification rate in classes such as paper and glass [13].

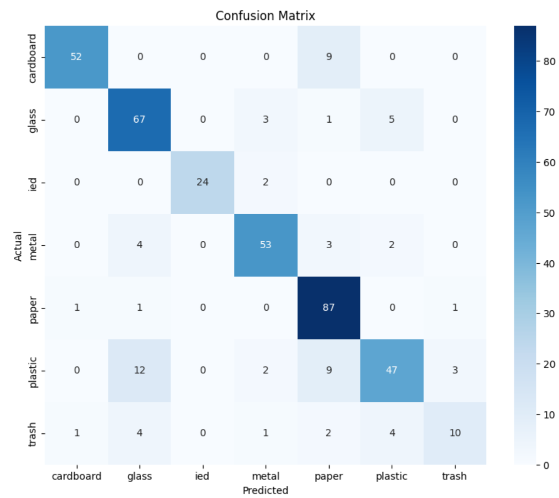

Fig. 18. Confusion matrix of Model 1

7) Model Accuracy and Loss:

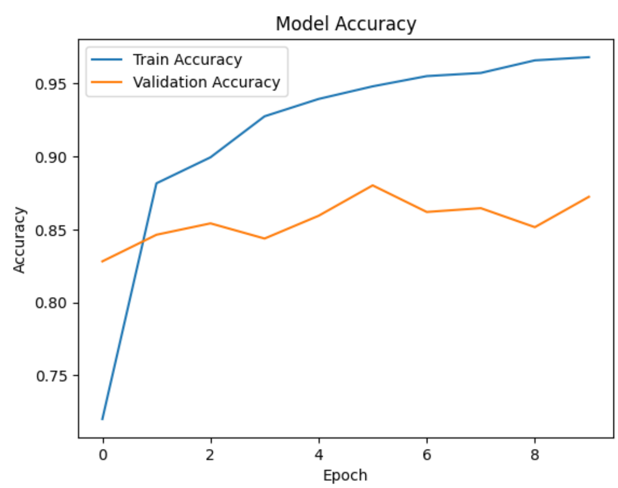

Fig. 19. Plot of Model 1 Accuracy

Figure 14 indicates the accuracy of the model between the train and validation accuracy. The training accuracy is determined to be 0.98 which indicates there is a higher classification.

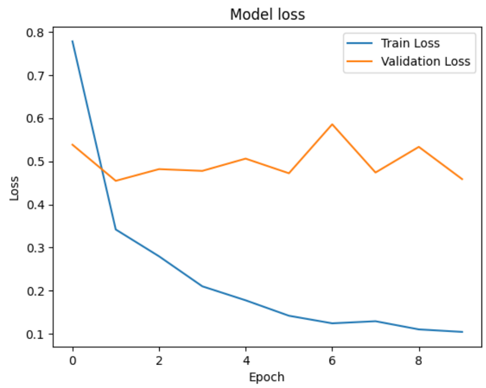

Fig. 20. Plot of Model 1 loss

The model loss is determined for the training model where the training loss is comparatively very low with the validation loss. The training model loss is 0.07 which is competitively low.

B. *Model 2: Adam optimizer, Without PReLU activation function*

8) Model Accuracy: The model 2 is enhanced to determine the classification of the waste based on the class types. The model prediction indicates that the Adam optimizer without the PReLU activation function has attained a classification accuracy of 84.39% [14].

```
# Predictions
y_pred = model_2.predict(test_set)
y_pred_model_2 = np.argmax(y_pred, axis=1)

display_accuracy(y_pred_model_2)
plot_cm(y_pred_model_2)
classification_report(y_true, y_pred_model_2, target_names=class_names)

plot_loss(history_model_2)
plot_accuracy(history_model_2)

13/13 [==============================] - 98s 7s/step
Test Accuracy : 84.39%
```

Fig. 21. Model 2 Accuracy

9) Confusion matrix: The above confusion matrix indicates that the paper waste is highly classified (87 times) next after the glass (70 times). This indicates that most of the waste were accurately classified except the comparison between glass and plastic.

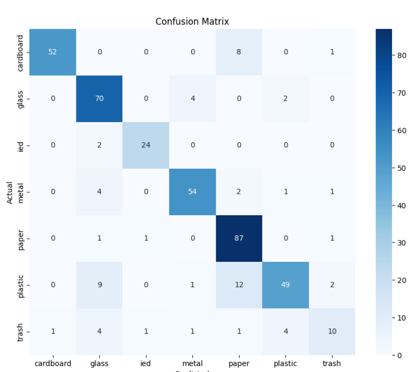

Fig. 22. Model 2 – Confusion matrix

10) Model Accuracy and Loss:

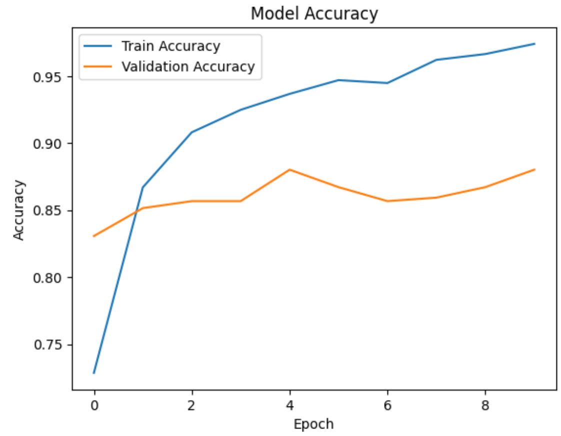

Fig. 23. Plot of Model 2 accuracy

The above figure shows the accuracy of Model 2 (Adam Optimizer without PReLU activation function). The accuracy of training data is examined as 0.97 which represents that the waste are effectively classified based on their types.

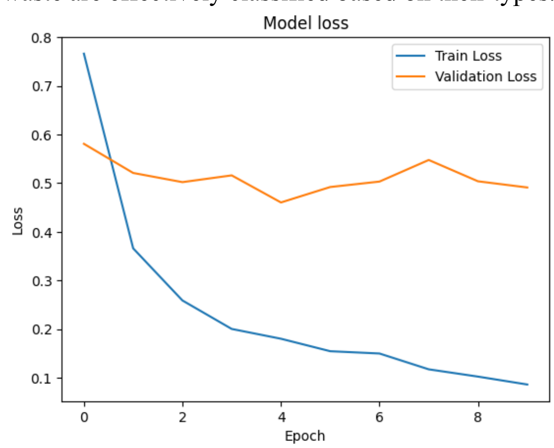

Fig. 24. Plot of Model 2 loss

The model loss for the model 2 is visualized in the above figure. This model has the least loss of about 0.08 which is comparatively very low. The lower indication of loss shows that more accuracy can be achieved with the waste classification.

C. *Model 3: SGD Optimizer, PReLU Activation Function*

11) Model Accuracy: The different classification optimizer is used with a PReLU activation function. The SGD optimizer is a global learning optimizer where it updates each parameter according to the learning rate [15]. Likewise, the model prediction to classify the waste using the SGD optimizer is 64.63%.

```
# Predictions
y_pred = model_2.predict(test_set)
y_pred_model_2 = np.argmax(y_pred, axis=1)

display_accuracy(y_pred_model_2)
plot_cm(y_pred_model_2)
classification_report(y_true, y_pred_model_2, target_names=class_names)

plot_loss(history_model_2)
plot_accuracy(history_model_2)
```

```
13/13 [==============================] - 98s 7s/step
Test Accuracy : 84.39%
```

Fig. 25. Model 3 Accuracy

12) Confusion matrix: The above confusion matrix indicates the classification of waste based on its types. Only paper is highly classified (84 times) whereas the glass, actual model, and plastic have the lowest classification count. For example, the paper and cardboard have a misclassification of 22 times which represents that the image is paper or cardboard.

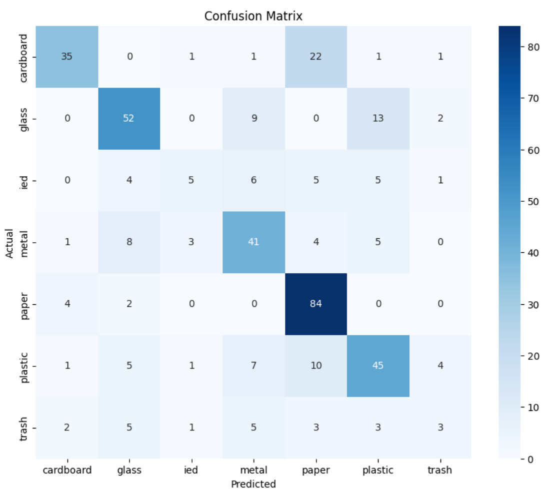

Fig. 26. Model 3 – Confusion matrix

13) *Model Accuracy and Loss:*

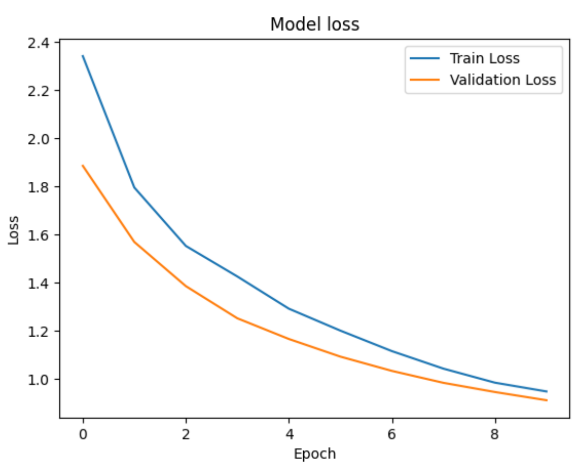

Fig. 27. Plot for Model 3 loss

The training loss on Model 3 is examined and visualized in the above screen. The training and validation loss is examined as 0.85 which indicates this model has a less performance in classification of waste [16].

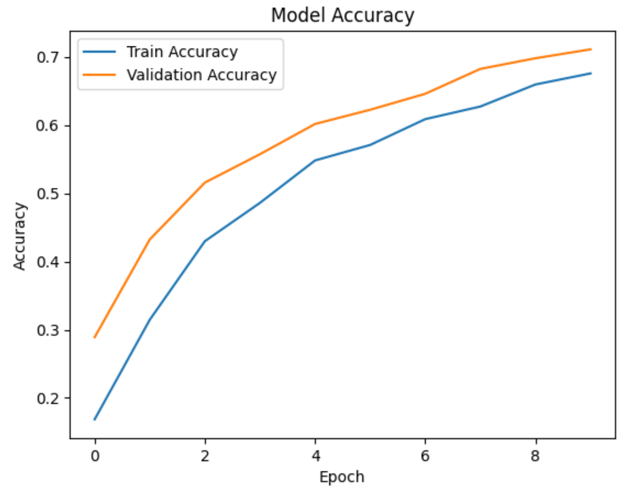

Fig. 28. Plot for Model 3 accuracy

The above visualization graph represents the model 3 accuracy based on the training data. the training and validation gradually increased with an accuracy of 0.72 or 72%.

D. *Model 4: SGD Optimizer, Without PReLU Activation Function*

1) Accuracy: The above code snippet is to predict the classification accuracy and generate a confusion matrix according to the waste classification class. The accuracy of this model is examined to be 65.37% on training and testing data.

```
# Predictions
y_pred = model_4.predict(test_set)
y_pred_model_4 = np.argmax(y_pred, axis=1)

display_accuracy(y_pred_model_4)
plot_cm(y_pred_model_4)
classification_report(y_true, y_pred_model_4, target_names=class_names)

plot_loss(history_model_4)
plot_accuracy(history_model_4)
```

```
13/13 [==============================] - 215s 16s/step
Test Accuracy : 65.37%
```

Fig. 29. Model 4 Accuracy

2) Confusion matrix: Similar to Model 3, Model 4 has also a similar prediction and classification of waste. The classification between glass and plastic is only 16 times which indicates it is difficult to classify between them. The paper is highly classified with 84 times with less rate.

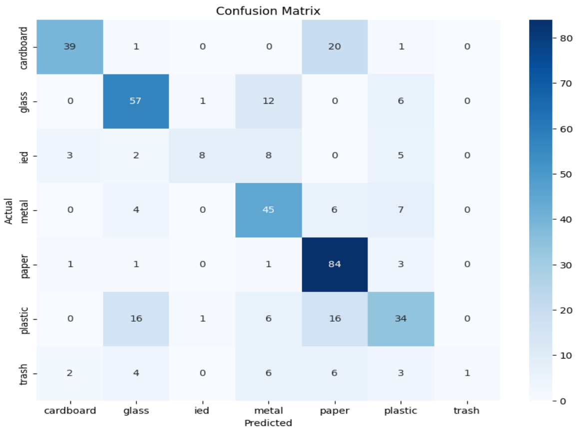

Fig. 30. Model 4 – Confusion matrix

3) *Model Accuracy and Loss:*

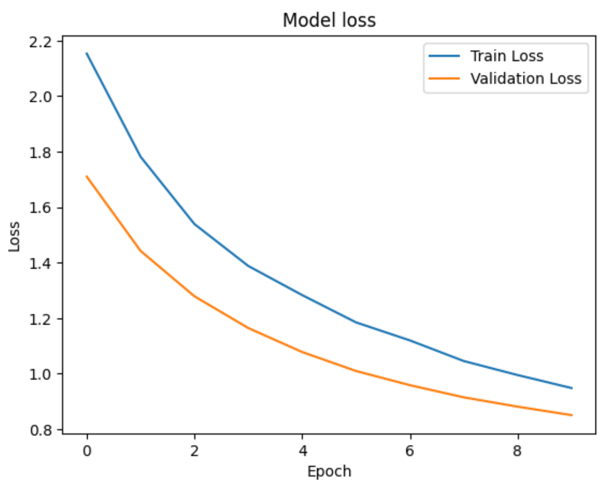

Fig. 31. Plot for Model 4 loss

The above graphs depict that the Model 4 accuracy and training competitively decreased when learning the dataset. The relationship between validation and training loss is narrow towards the end so, there is a generalized performance in waste classification.

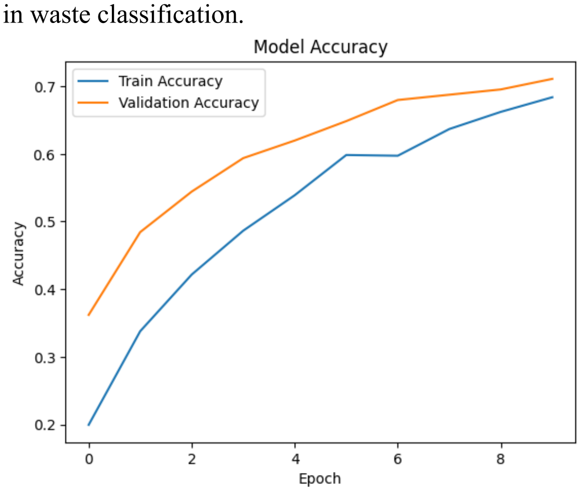

Fig. 32. Plot for Model 4 accuracy

At epoch 10, the model accuracy attained to classify the waste is 0.72 or 72%. The variation in the line represents the validation accuracy exceeds training accuracy because of data regularization.

4) *Accuracy on classification:*

```
eval_results = []

for model_name, model in [("model1", model_1), ("model2", model_2), ("model3", model_3), ("model4", model_4)]:
  for dataset_name, dataset in [("train", train_set), ("val", val_set), ("test", test_set)]:
    loss, accuracy = model.evaluate(dataset)
    eval_results.append([model_name, dataset_name, loss, accuracy])

eval_results_df = pd.DataFrame(eval_results, columns=["Model", "Dataset", "Loss", "Accuracy"])
pivot_result_df = eval_results_df.pivot(index='Model', columns='Dataset', values=['Accuracy', 'Loss'])
pivot_result_df = pivot_result_df.swaplevel(0, 1, axis=1).sort_index(axis=1)

display(pivot_result_df.style.format("{:.2f}").set_caption("Model Performance Metrics"))
```

```
59/59 [==============================] - 914s 15s/step - loss: 0.0670 - accuracy: 0.9809
13/13 [==============================] - 196s 15s/step - loss: 0.5443 - accuracy: 0.8529
13/13 [==============================] - 201s 15s/step - loss: 0.6401 - accuracy: 0.8293
59/59 [==============================] - 934s 16s/step - loss: 0.0758 - accuracy: 0.9729
13/13 [==============================] - 200s 15s/step - loss: 0.5721 - accuracy: 0.8628
13/13 [==============================] - 204s 16s/step - loss: 0.6076 - accuracy: 0.8439
59/59 [==============================] - 937s 16s/step - loss: 0.8540 - accuracy: 0.7226
13/13 [==============================] - 199s 15s/step - loss: 0.9766 - accuracy: 0.6833
13/13 [==============================] - 204s 16s/step - loss: 0.9963 - accuracy: 0.6463
59/59 [==============================] - 1334s 23s/step - loss: 0.8577 - accuracy: 0.7237
13/13 [==============================] - 162s 12s/step - loss: 0.9421 - accuracy: 0.6808
13/13 [==============================] - 119s 9s/step - loss: 0.9835 - accuracy: 0.6537
```

Model Performance Metrics

Dataset	test		train		val	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Model						
model1	0.83	0.64	0.98	0.07	0.85	0.54
model2	0.84	0.61	0.97	0.08	0.86	0.57
model3	0.65	1.00	0.72	0.85	0.68	0.98
model4	0.65	0.98	0.72	0.86	0.68	0.94

Fig. 33. Evaluation of All Models

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
  test_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 2513 files belonging to 2 classes.
Using 502 files for validation.
```

Figure 34. Loading and Splitting Dataset for Validation in TensorFlow

Here it loads the testing dataset in a similar manner as the training dataset. However, this dataset will be used strictly for evaluating the model after it has been trained.

```
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
X_train, y_train = next(iter(normalized_ds))
first_image = X_train[0]
print(np.min(first_image), np.max(first_image))
```

```
0.0074697463 1.0
```

Figure 35. Normalizing Dataset

This code normalizes the images in the training dataset, scaling the pixel values to a range between 0 and 1. Then, it extracts the first batch of images from the normalized dataset, retrieves the first image from this batch, and prints the minimum and maximum pixel values of this image. The normalization ensures that the model will have an easier time learning from the images, as it helps prevent issues with large or inconsistent pixel values affecting the learning process.

```
loss, accuracy = model.evaluate(X_train,y_train, verbose=Fals
e)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test,y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Training Accuracy: 1.0000
Testing Accuracy:  0.7188
```

Figure 36. Model Accuracy

The output shows that the training accuracy is 1.0000, meaning the model is perfectly predicting the labels on the training dataset. This suggests that the model has learned the training data very well. The testing accuracy is 0.7188, meaning the model correctly predicts around 71.88% of the test data. This is lower than the training accuracy, indicating that while the model performs perfectly on the training data, it may not generalize as well to new data, potentially due to overfitting.

```
def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```

Figure 37. Plot Training and Validation Metrics Over Epochs

This code defines function plot graphs to plot training and validation performance metrics (accuracy and loss) over epochs. It takes in the history object (from model training) and a metric name (accuracy or loss). It plots both the training metric (history.history[string]) and the validation metric (history.history['val_' + string]) over the number of epochs.

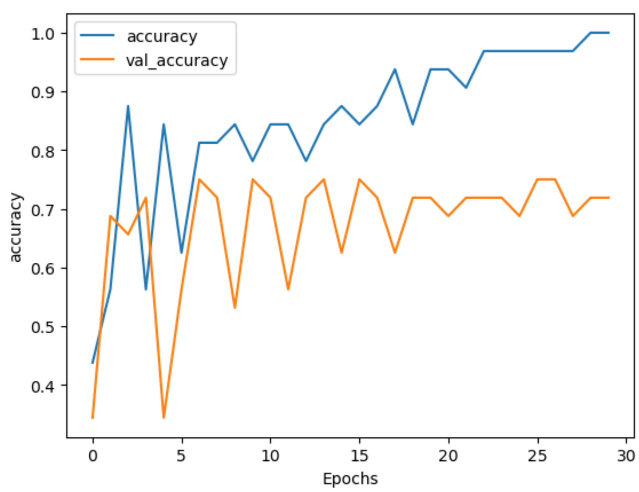

Figure 38. Training and Validation Accuracy Over 30 Epochs

The graph shows that training accuracy (blue line) improves steadily, reaching nearly 100%, while validation accuracy (orange line) fluctuates and stabilizes around 70-75%. The large gap between training and validation accuracy suggests overfitting, meaning the model is learning the training data well but struggling to generalize to unseen data.

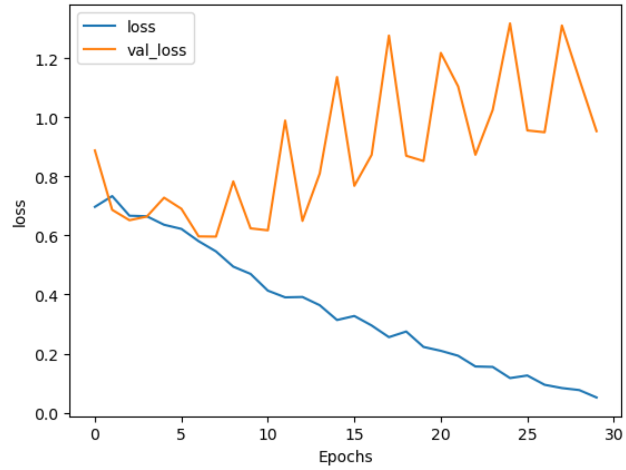

Figure 39. Training and Validation Loss Over 30 Epochs

The graph shows training loss steadily decreasing, while validation loss fluctuates and remains higher. This indicates overfitting, where the model learns the training data well but struggles to generalize to unseen validation data.

Conclusion

Waste classification using the Vision transformer model is proposed and deployed in this project. Since there are several deep learning models, the vision transformer model provides an efficient classification of waste according to its classes. The segregation of waste through automatic processes will reduce labor costs and improve the efficient disposal of waste. whaBy pre-training the TrashNet and I2Net datasets, the vision transformer model with different variations enhances different accuracy and model loss. The result obtained through an experimental analysis is that Model 1 has the highest accuracy across the entire dataset whereas Model21 also has similar accuracy with slight underperformance. Model 3 performs well on the test data where the accuracy attained is 100% but has a lower validation accuracy. When compared to all other models, Model 1 has a consistency in the classification of waste under different categories with the train, test, and validation dataset. In the future, an advanced combination of models can be used to effectively classify the waste.

References

[1] [1] N. S. Khan, Z. M. A. Amin and R. Hassan, "Waste Management Using Machine Learning and Deep Learning Algorithms," International Journal on Perceptive and Cognitive Computing, vol. 6, no. 2, pp. 97-106, 2020.

[2] M. Uddin, M. Malik, S. Sharma, C.-L. Chen, C.-M. Wu, P. Soni and S. Chaudhary, "Waste Classification for Sustainable Development Using Image Recognition with Deep Learning Neural Network Models," Sustainability , vol. 14, no. 12, 2022.

[3] A. Alshahrani, "Artificial intelligence technologies utilization for detecting explosive materials," International Journal of Data and Network Science , vol. 8, pp. 617-628, 2024.

[4] K. Huang, H. Lei, Z. Jiao and Z. Zhong, "Recycling Waste Classification Using Vision Transformer on Portable Device," Sustainability, vol. 13, no. 21, 2021.

[5] N. Narayanswamy, A. R. A. Rajak and S. Hasan, "Development of Computer Vision Algorithms for Multi-class Waste Segregation and Their Analysis," Emerging Science Journal, vol. 6, no. 3, pp. 631-646, 2022.

[6] M. Chhabra, B. Sharan, M. Elbarachi and M. Kumar, "Intelligent waste classification approach based on improved multi-layered convolutional neural network," Multimedia Tools and Applications , pp. 1-26, 2024.

[7] S. Poudel and P. Poudyal, "Classification of Waste Materials using CNN Based on Transfer Learning," in FIRE '22: Proceedings of the 14th Annual Meeting of the Forum for Information Retrieval Evaluation, Kolkata, India, 2023.

[8] *F. S. Alrayes, M. M. Asiri, M. S. Maashi, M. Rizwanullah, A. E. Osman, M. K. Nour, S. Drar and A. S. Zamani, "Waste classification using vision transformer based on multilayer hybrid convolution neural network," Urban Climate, vol. 49, 2023.*

[9] S.-H. Tsang, "Review: Vision Transformer (ViT)," medium, 4 Feb 2022. [Online]. Available: https://sh-tsang.medium.com/review-vision-transformer-vit-406568603de0. [Accessed 14 Sep 2024].

[10] K. Huang, H. Lei, Z. Jiao and Z. Zhong, "Recycling Waste Classification Using Vision Transformer on Portable Device," Sustainability , vol. 13, no. 21, 2021.

[11] M. Zhou, Z. Wang, B. Liu and T. Li, "Deep Image Steganography Using Transformer and Recursive Permutation," Entropy, vol. 24, no. 7, 2022.

[12] *W. Liu, H. Ouyang, Q. Liu, S. Cai, C. Wang, J. Xie and W. Hu, "Image Recognition for Garbage Classification Based on Transfer Learning and Model Fusion," Mathematical Problems in Engineering, vol. 1, 2022.*

[13] *K. An and Y. Zhang, "LPViT: A Transformer Based Model for PCB Image Classification and Defect Detection," IEEE Access, vol. 10, pp. 42542-42553, 2022.*

[14] *S. Junawane, "ViT: Vision Transformer," Medium, 2022 Apr 2022. [Online]. Available: https://medium.com/machine-intelligence-and-deep-learning-lab/vit-vision-transformer-cc56c8071a20. [Accessed 17 Sep 2024].*

[15] *K. Chaudhary, "Understanding Optimizers for training Deep Learning Models," Medium, 15 Nov 2023. [Online]. Available: https://medium.com/game-of-bits/understanding-optimizers-for-training-deep-learning-models-694c071b5b70#:~:text=SGD%20optimizer%20works%20with%20a,modification%20of%20the%20SGD%20algorithm.. [Accessed 17 Sep 2024].*

[16] A. A. M. Omer, "Image Classification Based on Vision Transformer," Journal of Computer and Communications, vol. 12, pp. 49-59, 2024.

[17] Shruti, "Satellite Image Classification Using Vision Transformers," Analytics Vidhya, 3 Oct 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/10/satellite-image-classification-using-vision-transformers/. [Accessed 19 Sep 2024]