Waste Classification Vision System

For Information Systems 315: Exercise 3

By: Micaela Klink

Due: 17th May 2025

Contents

Introduction

Waste continues to play an impact on the environment through pollution. A waste classifier can assist in reducing these effects by assisting workers in the recycling industry through proper waste categorization.

The aim of this project is to build a vision system that classifies different sorts of waste in the real-world accurately.

Methodology

Data collection

All datasets were retrieved from Kaggle and is as follows:

- The first dataset was retrieved from Suman Kunwar with the title of ‘Garbage Dataset’ which contained 10 directories (Battery, Biological, Cardboard, Clothes, Glass, Metal, Paper, Plastic, Shoes and Trash)
- The second dataset was retrieved from Akshat Tamrakar with the title of ‘E Waste Image Dataset’ which contained 10 directories (PCB (Print Circuit Board), Player, Battery, Microwave, Mobile, Mouse, Printer, Television, Washing Machine and Keyboard)
- The third dataset was retrieved from Cchangcs titled ‘Garbage Classification’ which included 6 directories (Cardboard, Glass, Metal, Paper, Plastic and Trash)
- The last dataset was retrieved from Angelika Ouedragogo titled ‘Waste Images’ which included 9 directories (Aluminium, Carton, E-Waste, Glass, Organic Waste, Paper and Cardboard, Plastics, Textiles and Wood)

These datasets were well thought of and combined to achieve realism for a real world. For example, the idea of merging categories such as clothes with shoes under the textile category was thought of but they are not always recycled together due to materials like rubber on shoes. Furthermore, adding a carton category was important as cartons contain aluminium, so in the recycling process, it may be split from cardboard.

With all this information, the following directories accumulated is as follows:

1. **Cardboard –** from ‘Garbage Dataset’
2. **Carton –** from ‘Waste Images’

3. **Clothes –** from ‘Garbage Dataset’
4. **Battery –** from ‘Garbage Dataset’
5. **E-Waste -** All ‘E Waste Image Dataset’ directories besides the battery directory
6. **Glass -** glass directory from ‘Garbage Classification’ with glass directory from ‘Garbage Dataset’
7. **Metal -** metal directory from ‘Garbage Classification’ with metal directory from ‘Garbage Dataset’
8. **Organic Waste -** organic waste directory from ‘Waste Images ‘with biological directory from ‘Garbage Dataset’
9. **Paper -** paper directory from ‘Garbage Classification’ with paper directory from ‘Garbage Dataset’
10. **Plastic -** plastics directory from ‘Waste Images with plastics directory from ‘Garbage Dataset’
11. **Shoes –** from ‘Garbage Dataset’
12. **Trash -** from ‘Garbage Dataset’

Appendix A displays the outcome of the dataset main details and appendix B shows the distribution of the data before pre-processing in a visual aspect. As observed, the dataset is imbalanced which can create a bias.

Pre-processing

Cleaning dataset

- The first action taken was to decrease the battery directory through python and combine it with E-Waste manually.
- The second action was to remove all files that were not jpgs which was done using python. Removing non jpg files helped reduced the dataset by a small bit.
- The third action was to remove duplicates in each directory which was done through DupeGuru. After this step, labelling the images correctly in each directory was important since various datasets directories were combined. For example, all plastic images would have the syntax of ‘plastic’, and so on.

Handling imbalances

- The fourth action was to apply under sampling to the dataset with the max folder being 500. This allowed for the dataset to be reduced so it could be manageable, but random sampling is also a technique to balance datasets that prevents bias. The outcome of under sampling is displayed in appendix C.

Data splitting

- Before normalization, data splitting occurs first to prevent information leakage from the testing set into the training set which can lead to unrealistic performance evaluations (Pinjosovsky, 2023).
- Data splitting is essential for the model's accuracy of performance in unseen data; hence, it ensures effectiveness in the real-world (alooba, n.d.).
- The dataset was split into 'Training', 'Validation', and 'Test' using the 80:10:10 ratio, respectively. The training dataset was set at a higher percentage rate so it could heavily learn patterns. In fine-tuning, the split was split into the 60:20:20 ratio as objects struggled to be detected in the real-world. This could have been because the 80:10:10 split lead each category to display an average of 50 images in each of the 11 directories. Furthermore, random sampling was applied as it ensures the split is unbiased (Buhl, 2024). Appendix D displays the data splitting output.

Normalization and Data Augmentation

- The next step was to calculate the mean and standard deviation so it could be utilized in normalizing the dataset. Normalizing is a crucial step in pre-processing. As Lin (2024) states, data normalization assists when the data volume is huge, and it assists in accuracy and reducing redundancy. The calculated mean and standard deviation results are shown in appendix E.
- Alongside normalization, data augmentation was applied. Data augmentation is applied to the training aspect of the dataset where it increases the size and diversity of the dataset (Hallaj, 2023). Hallaj (2023) further explains that it improves generalisation and accuracy, and additionally, it reduces overfitting (when the model learns training data too well, so it is unable to generalise new data). Appendix F displays the augmentations applied to the training data set. In appendix F, the RandomeResizedCrop learns to focus on different portions of an image whereas the ColorJitter ensures the model can generalize images captured in different lighting conditions.

Model development

Machine learning framework

According to snowflake (n. d.), machine learning frameworks contains pre-built functions. Additionally, it provides libraries and tools needed to build, train, and optimize machine learning models and it assists in developing and deploying machine learning systems.

The machine learning framework that was utilized for this project is PyTorch. According to snowflake (n. d.), PyTorch is easier to use for beginners than Tensorflow. Additionally, PyTorch was chosen over TensorFlow because it is suitable for rapid prototyping (Atreya, 2024).

In addition to PyTorch, Sklearn was applied to provide classification reports (appendix I) and visualisations like the confusion matrix (appendix K) to evaluate the performance of the model.

Neural network architecture

The neural network architecture that will be utilized is convolutional neural network (CNN) which is useful for data patterns and visual datasets like images and videos (geekforgeeks, 2025).

At first, a CNN model was in the process of being created from scratch, but due to the constant low accuracy (60% for the most) regardless of changes made, the decision of using a pre-trained model was established. ResNet18 was utilized over ResNet50 due to computational resources being limited such as GPU (Product Teacher, n.d.). Additionally, ResNet18 was utilized as it can handle complex and large datasets (Iterate.ai, n.d.).

Training and validation

Training and validating the model is a vital step. The training dataset is utilized for the model to understand patterns on the dataset whereas validating the dataset is used to assess the performance of the model and tell us how the model is learning and adapting (Kili, n.d.). Appendix G and I displays the epochs of 15 (so the model can learn the pattern) that was ran. Appendix G displays that the training loss decrease was consistent which indicates the model is learning effectively but the validation loss trend varies which means that model performance on unseen data is not improving and the gap between the training and validation indicates the model isn’t overfitting. Appendix H also indicates that the model is learning effectively but the gap between the two line graphs indicates potential overfitting.

Evaluation

In this phase, testing is run for the model to test unseen data. Figure I to K displays the performance of the test. Appendix I shows the precision, recall, F1-score and support for each category. Cardboard has the highest f1-score of 0.98 and metal has the lowest f1-score of 0.78. Additionally, cardboard has the highest precision of 1.00 and metal with the lowest precision of 0.67. The overall test accuracy is of 89.10% which means that 89.10% of the test samples were correctly classified. The lowest recall is from carton of 0.72 which means the model struggled to correctly identify the samples. Appendix J displays the outcome of randomly retrieving test images to see if the prediction was right to the actual category. Appendix K displays a confusion matrix which indicates that there is frequent plastic and glass misclassifications which can be due to the transparency of those objects. Additionally, e-waste gets confused with metal which can be because e-waste contains metal.

Additionally, object detection was utilized to test new and unseen data in real time. My results were not too optimistic as it identified myself as shoe and trash. Additionally, it tended to mix up plastic with glass so fine-tuning was needed.

Fine-Tuning

In this phase, I used my reduced dataset and built from there. I created a new folder for the fine-tuning dataset. I manually deleted similar images and took images of my own and added it to the respective directories. From there, I continued the usual process of splitting the data, but I decided to split the dataset into a 60:20:20 ratio (appendix L) as mentioned earlier. After that, calculating the new mean and standard deviation (appendix M) for normalization was established. In addition, data augmentation (appendix N) was added too, but the only difference from the first time was adding a hue and some values. The reason behind increasing the ColorJitter values was so it can be more robust for the real-world lighting variations.

Appendix O displays the training and validation logs over 10 epochs which was reduced due to my data split ratio and for it to run faster. The final epoch training and validation accuracy is also below 10% but it is close to it which can represent that the model is overfitting which can also be reviewed in appendix P and appendix Q. Appendix P displays that the training loss consistently decreases which indicates that the model is learning the patterns effectively, but the validation loss decreases initially, stabilise then fluctuates up and then down again which suggests that the model performance on unseen data improves early. Appendix Q training displays that the training accuracy increases which indicates the model is learning patterns effectively.

Appendix R displays that the test overall accuracy was 89.89% which means 89.89% of the test samples were classified correctly. Appendix S displays that the plastic class is performing purely due to it being the lowest recall (0.71), precision (0.86) and F1-score (0.78). Appendix T displays a confusion matrix which indicates that glass is being identified as metal and plastic is being identified as trash. Appendix S displays predicted vs actual results from randomly retrieving images.

Real-world applications, Limitations and Ethics

Real-world applications

The waste classification vision system can be deployed in recycling facilities. This could assist the facilities in identifying the waste category and ensure waste is separated correctly for recycling. Additionally, it can be deployed in public spaces to guide citizens on recycling separation, so essentially, citizens would be helping recycling facilities.

Limitations

The vision system did not overall perform a 100% accurate. This could be because of the data selection in the datasets and underrepresented categories. Additionally, items like electronics are made from metal and plastic materials which does hinder the system from identifying the object as an electronic item.

Additionally, in the real-world, objects tend to be more distorted, dirty and broken. As reviewed in my object detection phase, some items were classified as trash although my trash dataset did not resemble the other directories.

Furthermore, hardware such as a lack of GPU lead me to using ResNet18 instead of ResNet50.

And lastly, lightning and background does hinder object detection. For example, bad lighting can perceive paper as cardboard. Additionally, from experience, the white background can hinder object detection as it will perceive the object is paper.

Ethics

Bias

Bias can lead to inaccuracy in identifying objects in the real-world. There are ways of mitigating it such as through data augmentation, random sampling, under sampling or oversampling. Additionally, outliers, missing data and irrelevant data should be removed to help reduce bias. Lastly, the dataset should be diverse to avoid redundancy and duplicates.

Privacy

Visions systems pick up objects but also people. Additionally, items like order boxes from Takealot contains citizens addresses which can violate citizens privacy and confidentiality. Finding ways such as blurring address details should be implemented.

Methods for responsible use of vision system

Vision systems involve

Bias mitigation

Data diversity is recommended to ensure the training is diverse and represents the real-world. Pre-processing methods such as data augmentation, random sampling and removing duplicates reduce bias.

Privacy protection

Confidential information such as addresses should be blurred.

Environment setup

Natural lighting is recommended for object detection.

Conclusion

Despite of the draw backs from this vision system due to challenges such as misclassification, the waste classification vision system still has the potential of improving through more testing and adding different images to the dataset.

By addressing the short falls such as bad lighting, the waste classification vision system can one day assist in reducing environmental harm.

References

alooba. (n.d.). Data Splitting. Retrieved from: https://www.alooba.com/skills/concepts/deep-learning/data-splitting/

Atreya, M. (2024, September 17). PyTorch vs. TensorFlow: A Comprehensive Comparison.

Retrieved from Rafay: https://rafay.co/the-kubernetes-current/pytorch-vs-tensorflow-a-comprehensive-comparison/#:~:text=Ultimately%2C%20the%20choice%20between%20PyTorch%20and%20TensorFlow,TensorFlow%20is%20likely%20to%20be%20more%20suitable.

Buhl, N. (2024, November 19). Training, Validation, Test Split for Machine Learning Datasets.

Retrieved from ENCORD: https://encord.com/blog/train-val-test-split/

Cchangcs. (2018). Garbage Classification [Data set]. Retrieved from Kaggle:

https://www.kaggle.com/ds/81794

Hallaj, P. (2023, September 20). Data Augmentation: Benefits and Disadvantages. Retrieved

from: https://medium.com/@pouyahallaj/data-augmentation-benefits-and-disadvantages-38d8201aead

Iterate.ai. (n.d.). ResNet: The Definition, Use Case, and Relevance for Enterprises. Retrieved

from: https://www.iterate.ai/ai-glossary/resnet-deep-network-architecture

Kili. (n.d.). Training, Validation and Test Sets: How To Split Machine Learning Data. Retrieved

from: https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data

Kunwar, S. (2024). Garbage Dataset [Data set]. Retrieved from Kaggle:

https://www.kaggle.com/dsv/10182596

Lin, P. (2024, March 20). Why is Data Normalization Important? Retrieved from:

https://www.computer.org/publications/tech-news/trends/importance-of-data-normalization

Ouedraogo, A. (2023). Waste Images [Data set]. Retrieved from Kaggle:

https://www.kaggle.com/datasets/angelikasita/waste-images

Pinjosovsky, S, B. (2023, July 5). Normalize data before or after split of training and testing data?

Retrieved from Medium: https://medium.com/@spinjosovsky/normalize-data-before-or-after-split-of-training-and-testing-data-7b8005f81e26

Product Teacher. (n.d.). ResNet18 & ResNet50 in Computer Vision. Retrieved from:

https://www.productteacher.com/quick-product-tips/resnet18-and-resnet50

snowflake. (n, d.). Machine Learning Frameworks: Streamlining ML Model Development.

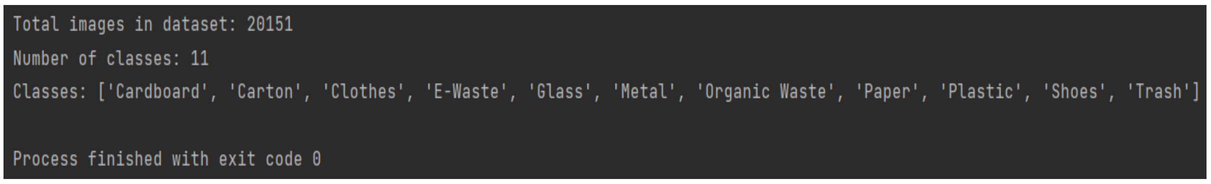
Retrieved from: https://www.snowflake.com/guides/machine-learning-frameworks/#:~:text=What%20Is%20a%20Machine%20Learning,preprocessing%2C%20model%20building%20and%20optimization.

Tamrakar, A. (2024). E Waste Image Dataset. Retrieved from Kaggle:

https://www.kaggle.com/datasets/akshat103/e-waste-image-dataset

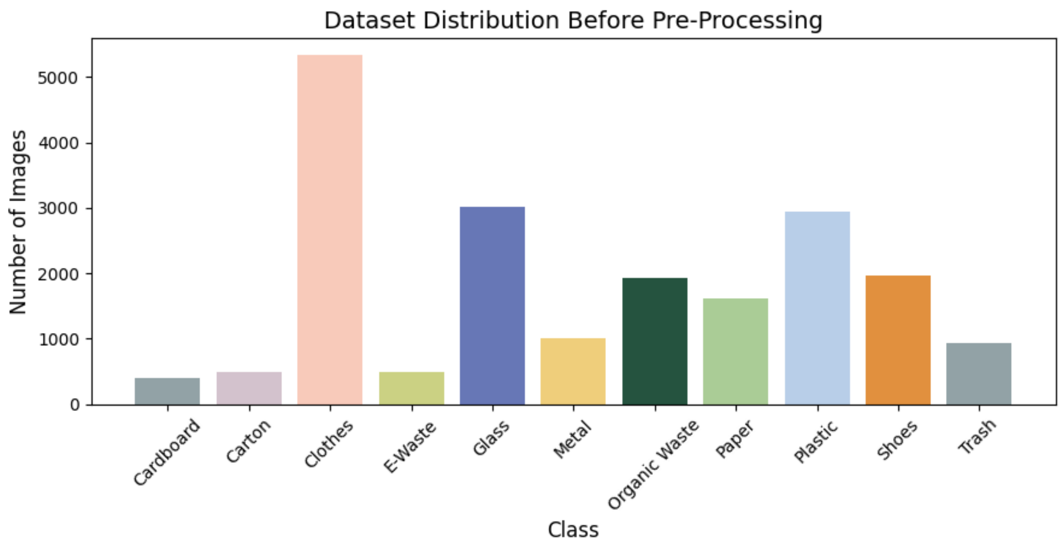
Appendices

Appendix A

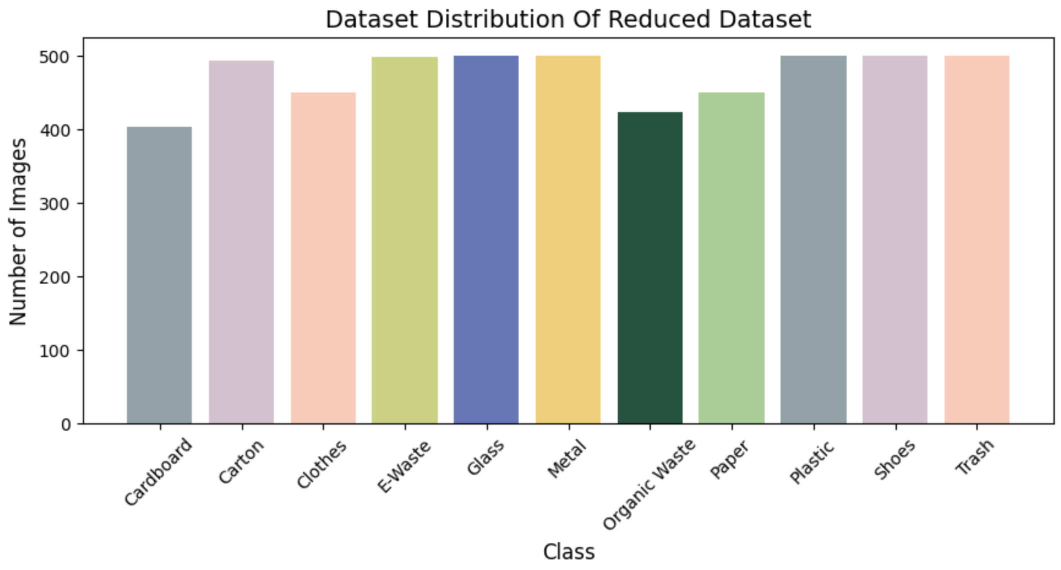


```
Total images in dataset: 20151
Number of classes: 11
Classes: ['Cardboard', 'Carton', 'Clothes', 'E-Waste', 'Glass', 'Metal', 'Organic Waste', 'Paper', 'Plastic', 'Shoes', 'Trash']

Process finished with exit code 0
```

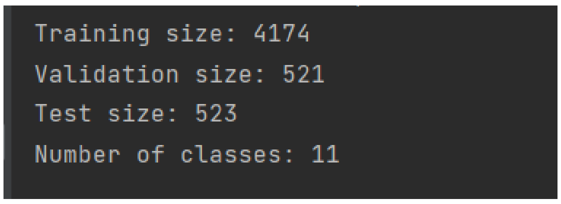
Appendix B



Appendix C

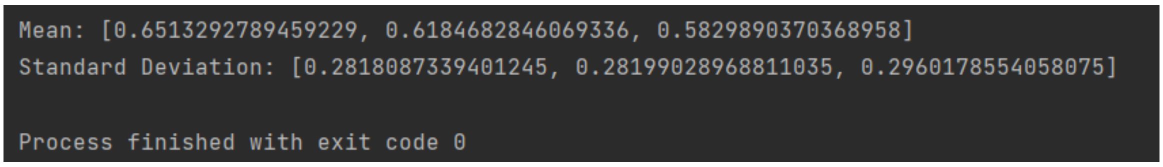


Appendix D



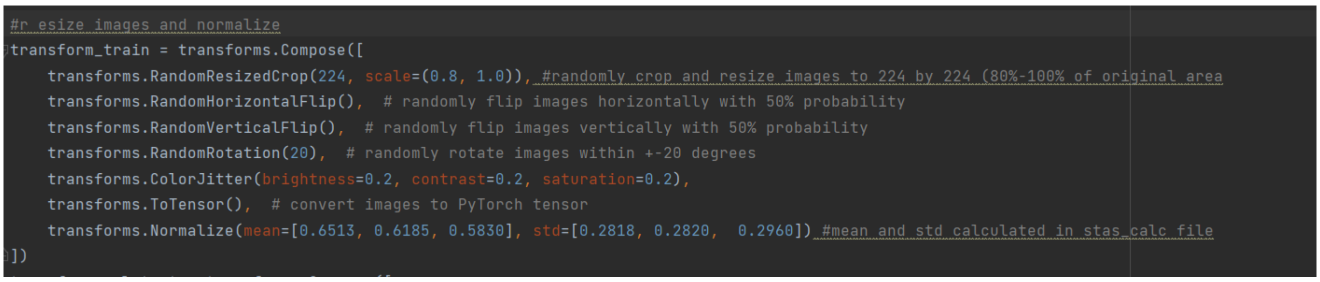
```
Training size: 4174
Validation size: 521
Test size: 523
Number of classes: 11
```

Appendix E



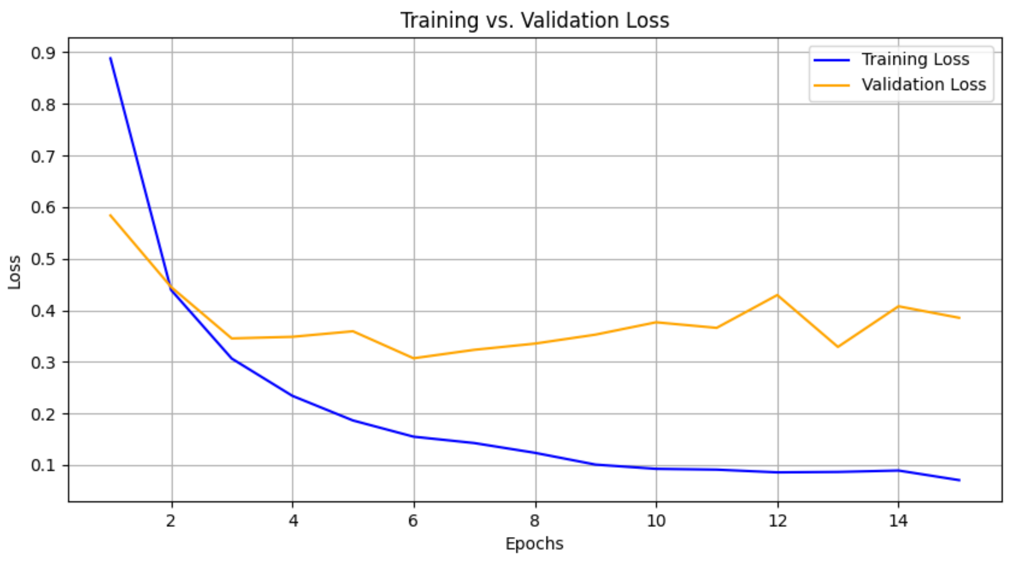
```
Mean: [0.6513292789459229, 0.6184682846069336, 0.5829890370368958]
Standard Deviation: [0.2818087339401245, 0.28199028968811035, 0.2960178554058075]

Process finished with exit code 0
```

Appendix F

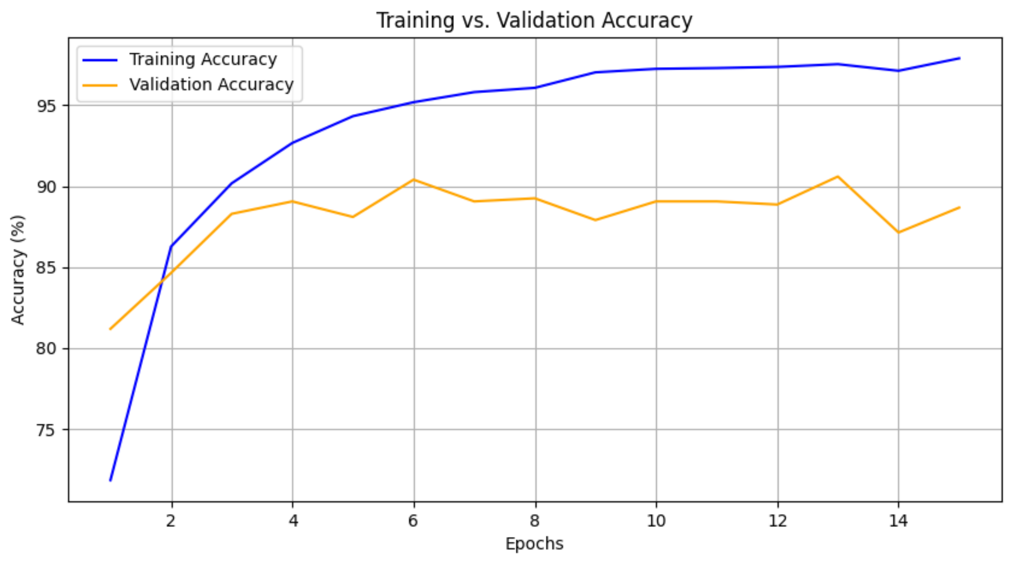


```
#r esize images and normalize
transform_train = transforms.Compose([
    transforms.RandomResizedCrop(224, scale=(0.8, 1.0)), #randomly crop and resize images to 224 by 224 (80%-100% of original area
    transforms.RandomHorizontalFlip(),  # randomly flip images horizontally with 50% probability
    transforms.RandomVerticalFlip(),  # randomly flip images vertically with 50% probability
    transforms.RandomRotation(20),  # randomly rotate images within +-20 degrees
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),
    transforms.ToTensor(),  # convert images to PyTorch tensor
    transforms.Normalize(mean=[0.6513, 0.6185, 0.5830], std=[0.2818, 0.2820,  0.2960]) #mean and std calculated in stas_calc file
])
```

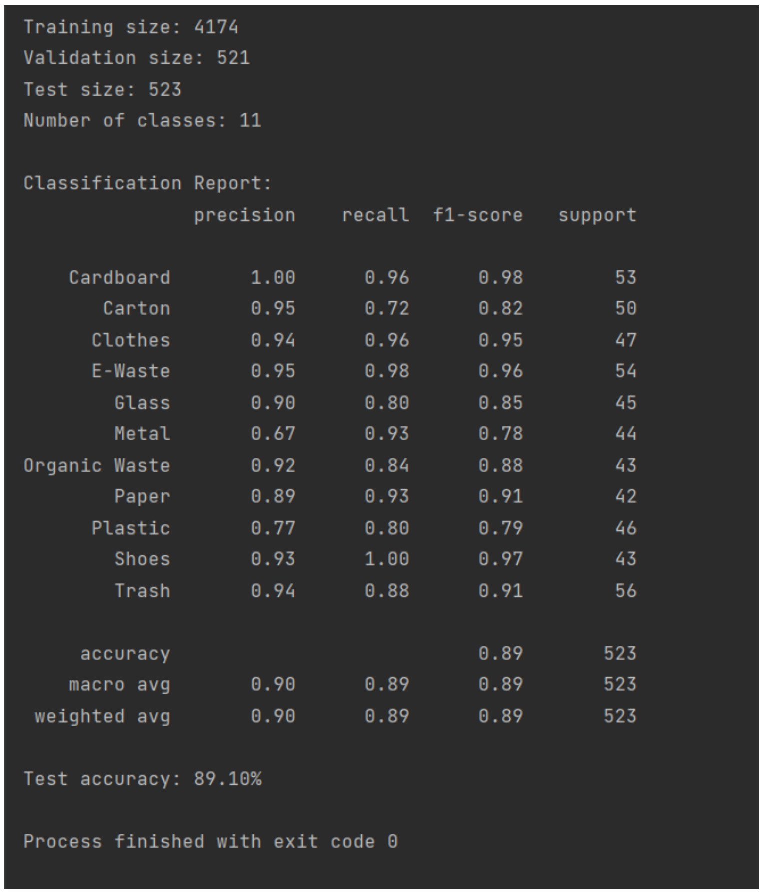
Appendix G



Appendix H



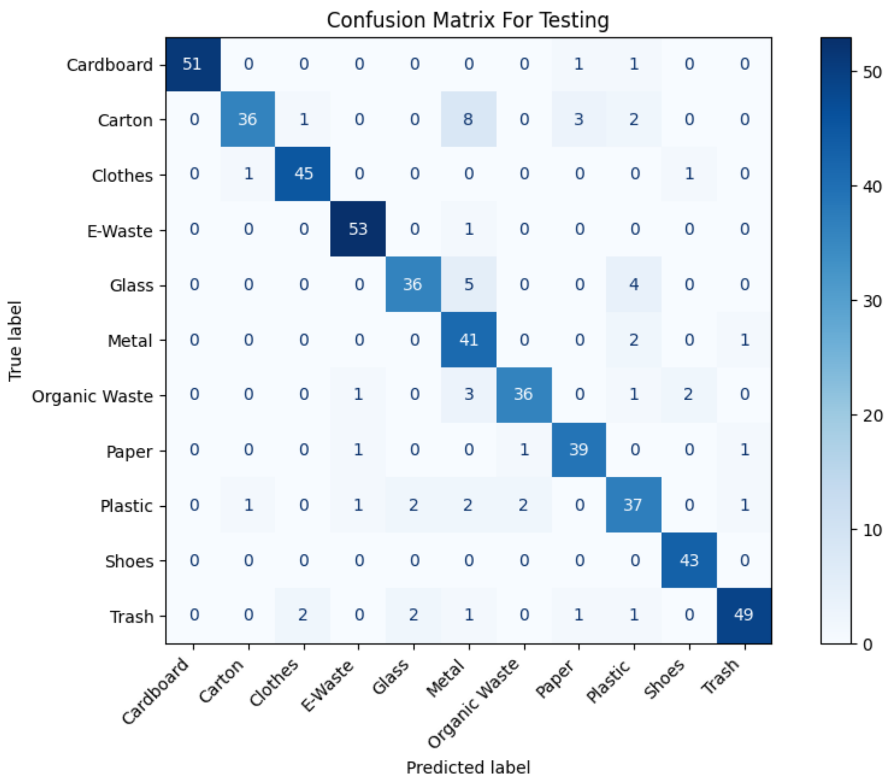
Appendix I



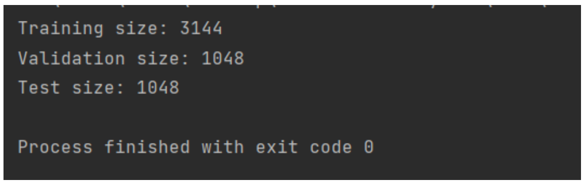
```
Training size: 4174
Validation size: 521
Test size: 523
Number of classes: 11

Classification Report:
               precision    recall  f1-score   support

    Cardboard       1.00      0.96      0.98        53
       Carton       0.95      0.72      0.82        50
      Clothes       0.94      0.96      0.95        47
      E-Waste       0.95      0.98      0.96        54
        Glass       0.90      0.80      0.85        45
        Metal       0.67      0.93      0.78        44
Organic Waste       0.92      0.84      0.88        43
        Paper       0.89      0.93      0.91        42
      Plastic       0.77      0.80      0.79        46
        Shoes       0.93      1.00      0.97        43
        Trash       0.94      0.88      0.91        56

     accuracy                           0.89       523
    macro avg       0.90      0.89      0.89       523
 weighted avg       0.90      0.89      0.89       523

Test accuracy: 89.10%

Process finished with exit code 0
```

Appendix J

Appendix K

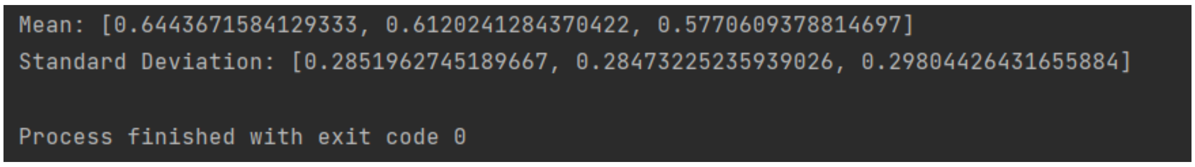


Appendix L



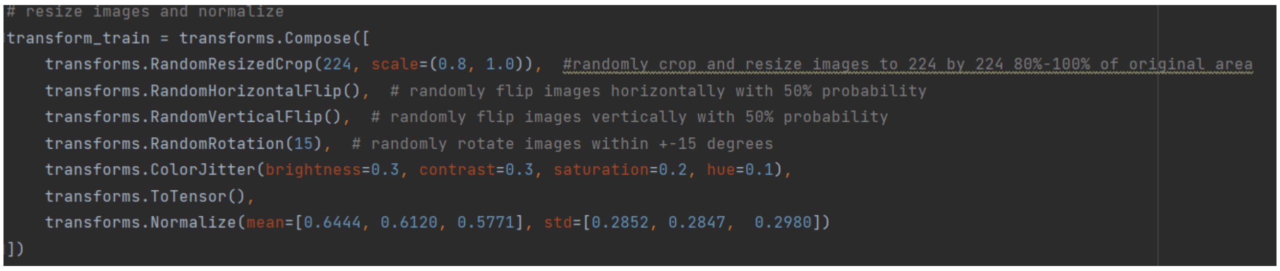
```
Training size: 3144
Validation size: 1048
Test size: 1048

Process finished with exit code 0
```

Appendix M



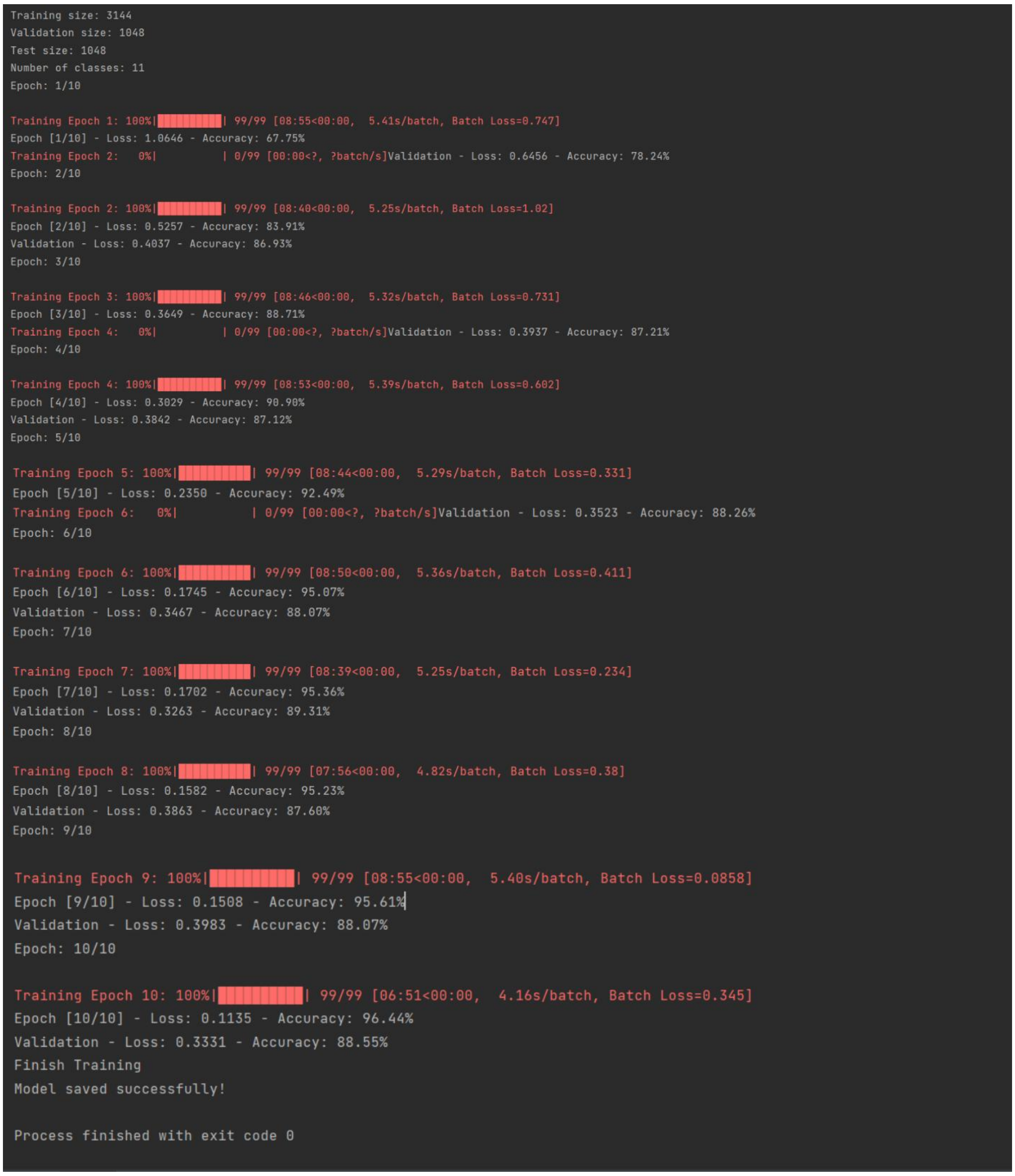
```
Mean: [0.6443671584129333, 0.6120241284370422, 0.5770609378814697]
Standard Deviation: [0.2851962745189667, 0.28473225235939026, 0.29804426431655884]

Process finished with exit code 0
```

Appendix N



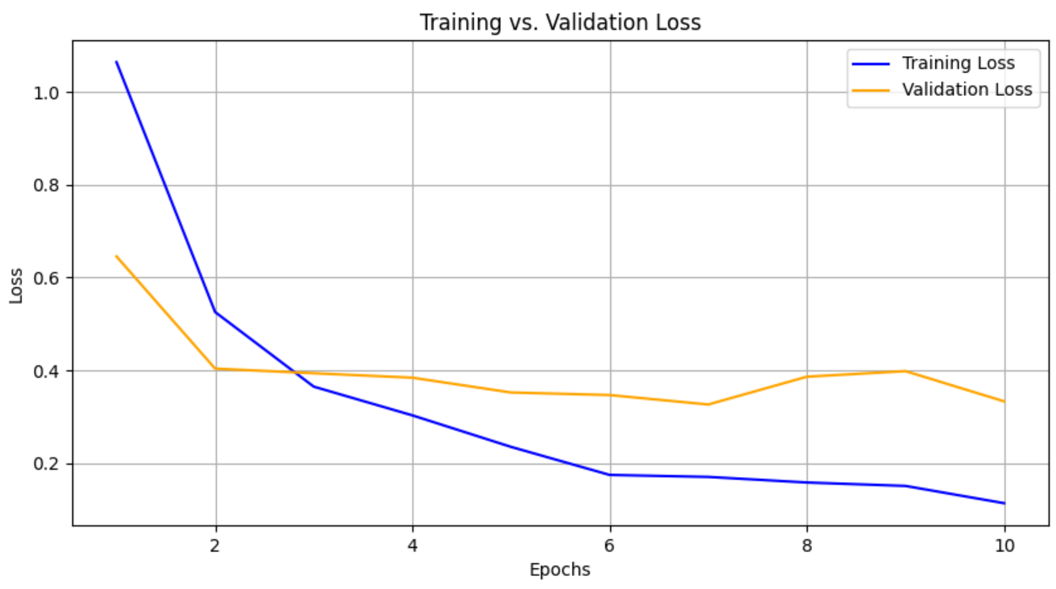
```
# resize images and normalize
transform_train = transforms.Compose([
    transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),  #randomly crop and resize images to 224 by 224 80%-100% of original area
    transforms.RandomHorizontalFlip(),  # randomly flip images horizontally with 50% probability
    transforms.RandomVerticalFlip(),  # randomly flip images vertically with 50% probability
    transforms.RandomRotation(15),  # randomly rotate images within +-15 degrees
    transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.6444, 0.6120, 0.5771], std=[0.2852, 0.2847,  0.2980])
])
```

Appendix O

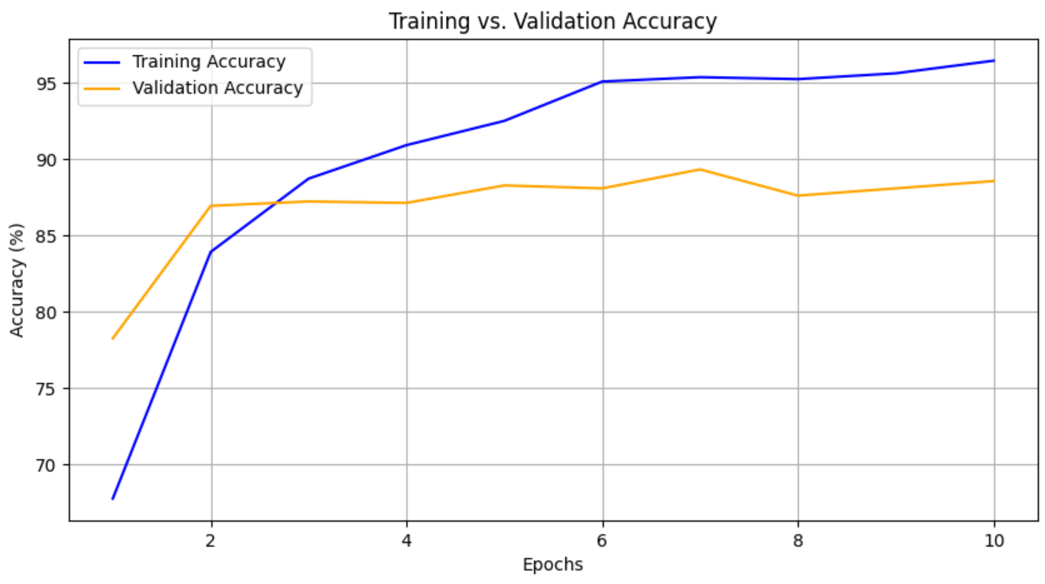


```
Training size: 3144
Validation size: 1048
Test size: 1048
Number of classes: 11
Epoch: 1/10

Training Epoch 1: 100%|██████████| 99/99 [08:55<00:00,  5.41s/batch, Batch Loss=0.747]
Epoch [1/10] - Loss: 1.0646 - Accuracy: 67.75%
Training Epoch 2:   0%|          | 0/99 [00:00<?, ?batch/s]Validation - Loss: 0.6456 - Accuracy: 78.24%
Epoch: 2/10

Training Epoch 2: 100%|██████████| 99/99 [08:40<00:00,  5.25s/batch, Batch Loss=1.02]
Epoch [2/10] - Loss: 0.5257 - Accuracy: 83.91%
Validation - Loss: 0.4037 - Accuracy: 86.93%
Epoch: 3/10

Training Epoch 3: 100%|██████████| 99/99 [08:46<00:00,  5.32s/batch, Batch Loss=0.731]
Epoch [3/10] - Loss: 0.3649 - Accuracy: 88.71%
Training Epoch 4:   0%|          | 0/99 [00:00<?, ?batch/s]Validation - Loss: 0.3937 - Accuracy: 87.21%
Epoch: 4/10

Training Epoch 4: 100%|██████████| 99/99 [08:53<00:00,  5.39s/batch, Batch Loss=0.602]
Epoch [4/10] - Loss: 0.3029 - Accuracy: 90.90%
Validation - Loss: 0.3842 - Accuracy: 87.12%
Epoch: 5/10

Training Epoch 5: 100%|██████████| 99/99 [08:44<00:00,  5.29s/batch, Batch Loss=0.331]
Epoch [5/10] - Loss: 0.2350 - Accuracy: 92.49%
Training Epoch 6:   0%|          | 0/99 [00:00<?, ?batch/s]Validation - Loss: 0.3523 - Accuracy: 88.26%
Epoch: 6/10

Training Epoch 6: 100%|██████████| 99/99 [08:50<00:00,  5.36s/batch, Batch Loss=0.411]
Epoch [6/10] - Loss: 0.1745 - Accuracy: 95.07%
Validation - Loss: 0.3467 - Accuracy: 88.07%
Epoch: 7/10

Training Epoch 7: 100%|██████████| 99/99 [08:39<00:00,  5.25s/batch, Batch Loss=0.234]
Epoch [7/10] - Loss: 0.1702 - Accuracy: 95.36%
Validation - Loss: 0.3263 - Accuracy: 89.31%
Epoch: 8/10

Training Epoch 8: 100%|██████████| 99/99 [07:56<00:00,  4.82s/batch, Batch Loss=0.38]
Epoch [8/10] - Loss: 0.1582 - Accuracy: 95.23%
Validation - Loss: 0.3863 - Accuracy: 87.60%
Epoch: 9/10

Training Epoch 9: 100%|██████████| 99/99 [08:55<00:00,  5.40s/batch, Batch Loss=0.0858]
Epoch [9/10] - Loss: 0.1508 - Accuracy: 95.61%
Validation - Loss: 0.3983 - Accuracy: 88.07%
Epoch: 10/10

Training Epoch 10: 100%|██████████| 99/99 [06:51<00:00,  4.16s/batch, Batch Loss=0.345]
Epoch [10/10] - Loss: 0.1135 - Accuracy: 96.44%
Validation - Loss: 0.3331 - Accuracy: 88.55%
Finish Training
Model saved successfully!

Process finished with exit code 0
```

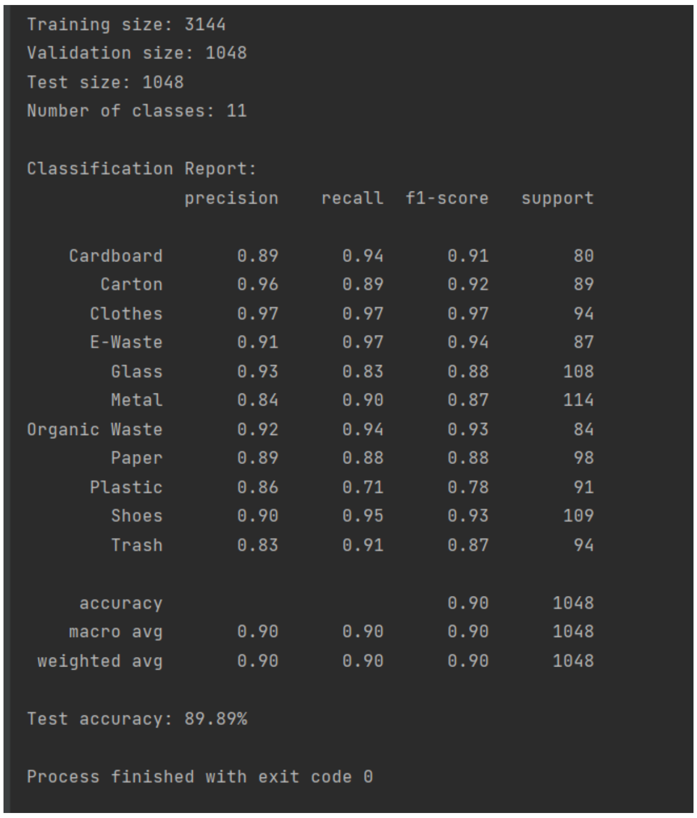
Appendix P



Appendix Q

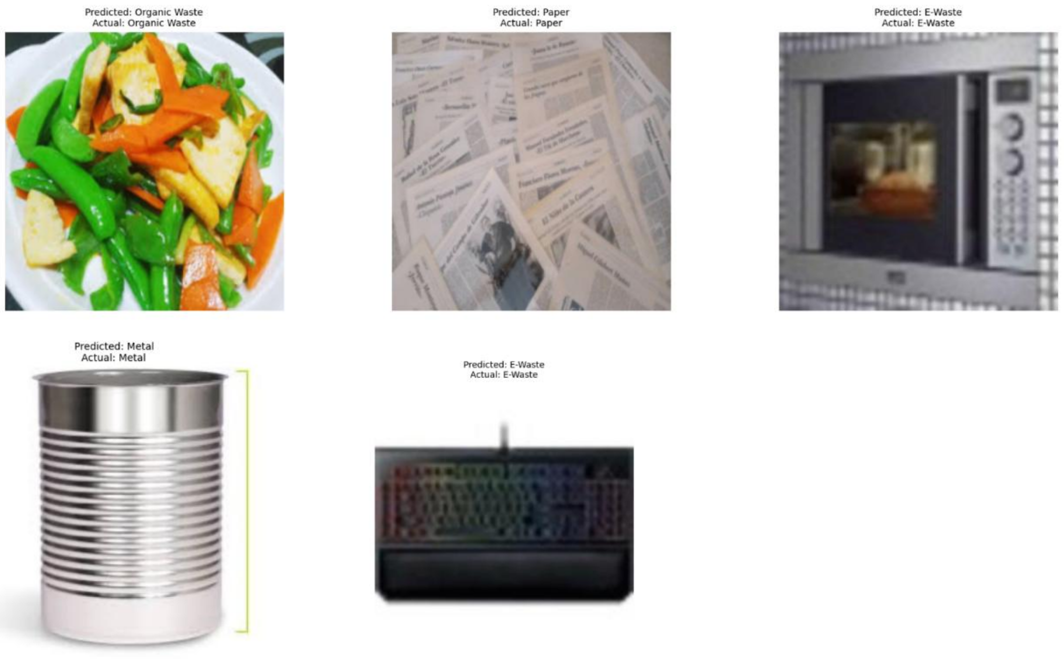


Appendix R

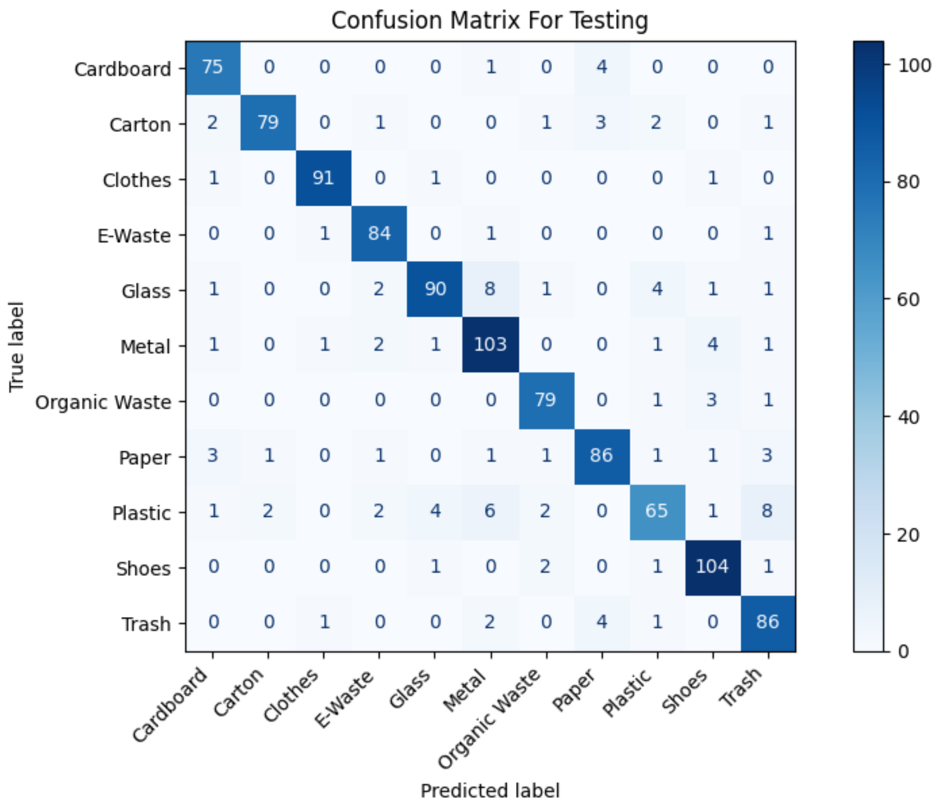


```
Training size: 3144
Validation size: 1048
Test size: 1048
Number of classes: 11

Classification Report:
               precision    recall  f1-score   support

    Cardboard       0.89      0.94      0.91        80
       Carton       0.96      0.89      0.92        89
      Clothes       0.97      0.97      0.97        94
      E-Waste       0.91      0.97      0.94        87
        Glass       0.93      0.83      0.88       108
        Metal       0.84      0.90      0.87       114
Organic Waste       0.92      0.94      0.93        84
        Paper       0.89      0.88      0.88        98
      Plastic       0.86      0.71      0.78        91
        Shoes       0.90      0.95      0.93       109
        Trash       0.83      0.91      0.87        94

     accuracy                           0.90      1048
    macro avg       0.90      0.90      0.90      1048
 weighted avg       0.90      0.90      0.90      1048

Test accuracy: 89.89%

Process finished with exit code 0
```

Appendix S



Appendix T



True label \ Predicted label	Cardboard	Carton	Clothes	E-Waste	Glass	Metal	Organic Waste	Paper	Plastic	Shoes	Trash
Cardboard	75	0	0	0	0	1	0	4	0	0	0
Carton	2	79	0	1	0	0	1	3	2	0	1
Clothes	1	0	91	0	1	0	0	0	0	1	0
E-Waste	0	0	1	84	0	1	0	0	0	0	1
Glass	1	0	0	2	90	8	1	0	4	1	1
Metal	1	0	1	2	1	103	0	0	1	4	1
Organic Waste	0	0	0	0	0	0	79	0	1	3	1
Paper	3	1	0	1	0	1	1	86	1	1	3
Plastic	1	2	0	2	4	6	2	0	65	1	8
Shoes	0	0	0	0	1	0	2	0	1	104	1
Trash	0	0	1	0	0	2	0	4	1	0	86