Convolutional Neural Networks for Efficient Garbage Classification

1st Łukasz Chrostowski
Department of Mathematics (DMat)
University of Aveiro (UA)
Aveiro, Portugal
lukasz.chrostowski@ua.pt

2nd Dominika Eder
Department of Mathematics (DMat)
University of Aveiro (UA)
Aveiro, Portugal
dominika.eder@ua.pt

***Abstract*—This document presents a convolutional neural network (CNN) model designed to classify images of garbage into six categories. The goal is to improve the efficiency of waste management and recycling processes by automating the classification of garbage. We introduce a sequential CNN model trained on a dataset comprising images of cardboard, glass, metal, paper, plastic, and trash, showing promising results in automated waste sorting.**

***Index Terms*—convolutional neural network, image classification, garbage classification, waste management, recycling**

I. Introduction

Garbage classification refers to the process of segregating waste into distinct categories, enabling effective recycling and reduction of waste sent to landfills. This practice is critical for managing the world's increasing waste production, which is intensified by population growth, urbanization, and economic development. Effective garbage classification is essential for minimizing environmental impacts, conserving natural resources, and enhancing the efficiency of waste management systems.

Proper waste management is pivotal in preventing pollution, reducing greenhouse gas emissions, and conserving raw materials through recycling. It also plays a crucial role in public health, helping to prevent diseases spread by inadequate waste handling. Despite these benefits, effective implementation faces numerous challenges, including technological, economic, and social barriers.

Technological challenges involve the complexity of accurately identifying and separating different types of waste materials. Traditional methods often rely on manual sorting, which is labor-intensive and not always efficient or feasible. Additionally, the variations in waste composition require sophisticated technologies to ensure accurate classification and recycling.

Economic challenges include the high costs associated with developing and maintaining advanced waste management systems. These systems often require significant capital investment in technology and infrastructure, as well as ongoing operational costs. Social challenges revolve around consumer behavior and awareness. Effective garbage classification requires public participation, which can be hindered by a lack of awareness, resistance to change, or misunderstanding about how to classify waste correctly.

To address these challenges, recent advancements have focused on the integration of machine learning and artificial intelligence. These technologies offer promising solutions for automating the classification process, increasing accuracy, and reducing the reliance on manual labor. Machine learning models can analyze and learn from large datasets of waste images, enabling the automatic identification and sorting of waste types based on visual characteristics.

II. Methodology

A. Dataset

For the purpose of the Garbage classification automation we have used data from kaggle. Dataset contains images distributed among six classes: cardboard (403), glass (501), metal (410), paper (594), plastic (482), and trash (137). These images are used to train a model to automatically classify waste into these categories.

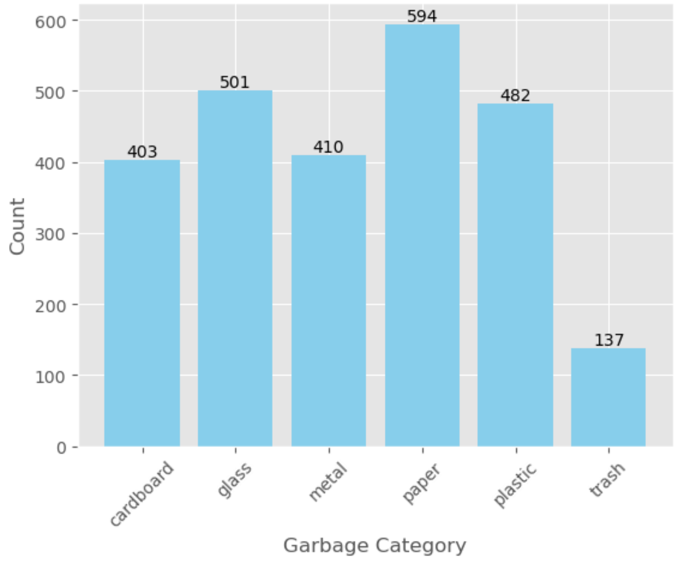


Fig. 1: Caption for the example image.

During data preprocessing, we divided the set of photos into training and test data, which is a normal practice when training machine learning models. The data was divided in a ratio of 80:20, which means that we used 80% of the photos to train the model to correctly classify photos, and 20% to validate the model, i.e. the ability to correctly classify photos that were not ”visible” to the algorithm. Images 2a, 2b, 2c, 2d, 2e, 2f present example pictures of the photed garbage from the dataset.

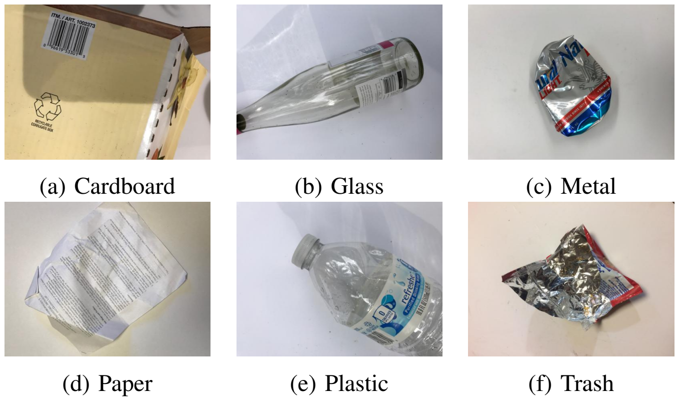


Fig. 2: Various types of recyclable materials and waste

B. Initial Data Preprocessing

In order to differentiate the data and enlarge the collection, we performed preliminary operations on the data. First of all, we performed data augmentation, in other words we applied transformations to the original images by scaling, cropping, and rotating them to introduce more variability into our dataset. This process not only helps in simulating different viewing conditions but also prevents the model from learning and memorizing specific details of the training images, thereby enhancing its ability to generalize to new, unseen images. These steps are critical in preparing a robust dataset that helps improve the accuracy and efficacy of a Machine Learning models.

C. Model Architecture

To automate the entire process, we have taken advantage of the still-discovered power of artificial neural networks. Although it is a relatively young branch of science, there are already many different developments and architectures based on this machine learning solution. In the context of image processing, so-called Convolutional Neural Networks have gained particular popularity in recent years.

Convolutional Neural Networks (CNNs) are a specialized type of artificial neural network designed for processing data that has a grid-like topology, such as images. CNNs are particularly powerful in tasks related to image recognition, classification, and analysis, which makes them integral to many modern artificial intelligence applications. There exist many different architectures of these type of Artificial Neural Networks (ANNs) but in general one can split into the following layers:

1) Convolutional Layers: These layers apply a number of filters to the input. Each filter detects different features such as edges, colors, or textures. The output of these layers is a set of feature maps.
2) ReLU Layers: Standing for Rectified Linear Unit, ReLU layers apply a non-linear activation function that introduces non-linearity to the system, allowing the network to learn more complex patterns.
3) Pooling Layers: These layers reduce the spatial size of the feature maps, making the detection of features invariant to scale and orientation, and reducing the computational complexity.
4) Fully Connected Layers: At the end of the network, fully connected layers analyze the features that have been identified by the previous layers to make the final classification decision.

There are also, of course, layers of other types such as DropOut layers. At the 3 one can look on the general flow of ANN layers in CNN models.

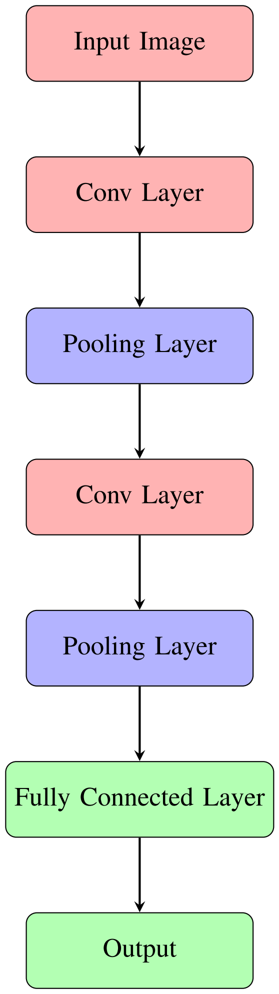


Fig. 3: Flow Diagram of a Convolutional Neural Network

In our approach, we built 4 models based on convolutional neural networks. Details about their architectures are included in Appendix. Two of them were designed directly by us and we name them as Not Regularized and Regularized (by adding

DropOut layer) Custom Models. The remaining two use ready-made CNN architectures known as ResNet (Residual Networks) and MobileNet. Let us briefly describe their properties.

D. ResNet

ResNet have been introduced by Kaiming He et al. in their 2015 paper, ”Deep Residual Learning for Image Recognition,” is a type of convolutional neural network (CNN) that addresses the problem of training very deep networks. As networks grow deeper, they often run into the vanishing gradient problem, which makes them hard to train. ResNet addresses this by introducing ”residual blocks” with skip connections, or shortcuts that allow the gradient to flow through the network without being dampened by deep layers.

E. MobileNet

On the other hand MobileNets, that have been introduced by Google researchers Andrew G. Howard et al. in their 2017 paper, ”MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” are a set of CNN architectures designed for mobile and resource-constrained environments. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build lightweight deep neural networks.

F. Training

It is also worth describing the hyperparameters that we defined while training the models. This is primarily about the choice of the loss function that we want to minimize, or the value of the learning coefficient, which determines the step length during learning and affects the time complexity and the found minimum.

Hyperparameter	Value
Learning Rate	0.0001
Batch Size	32
Epochs	14
Dropout Rate	0.5
Loss Function	categorical crossentropy
Optimizer	ADAM
Activation	relu or softmax

TABLE I: Hyperparameters and their Values

III. Experiments and Results

Describe the experimental setup, including how the data was split between training and validation, the training process, and the metrics used for evaluation. Present the results, including training and validation accuracy over epochs.

A. Evaluation

To assess the quality of the model, we will use popular metrics for classification problems. First, let’s briefly discuss what they mean and how we obtain them. These are statistics such as

1) Precision, that measures the accuracy of positive predictions. Formally, it is the ratio of correctly predicted positive observations to the total predicted positives. This metric is particularly useful when the costs of False Positives are high.

$$\text{Precision} = \frac{TP}{TP + FP}$$

2) Recall, also known as Sensitivity or True Positive Rate, that measures the ability of a model to find all the relevant cases (i.e., True Positives) within a dataset. It is the ratio of correctly predicted positive observations to all observations in actual class - yes

$$\text{Recall} = \frac{TP}{TP + FN}$$

3) F1 Score, that is a way of combining both precision and recall into a single measure that captures both properties. It is the harmonic mean of precision and recall.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Model	Precision	Recall	F1	Accuracy
Custom CNN 1	0.75	0.73	0.73	0.74
Custom CNN 2	0.69	0.63	0.65	0.64
ResNet50	0.36	0.36	0.34	0.41
MobileNetV2 4	0.85	0.80	0.81	0.82

TABLE II: Model Evaluation Metrics

B. Comparison to the other articles

We decided to compare our best result obtained using the MobileNet network with other studies on garbage classification. The results are in III. The model called DLA (Deep-Learning Algorithm) is based on ResNexT architecture, that is built on the innovations of the ResNet model by introducing a more modular architecture, while Transfer Learning is based on EfficientNetV2M architecture, that is extension and improvement of the EfficientNet architecture, which focuses on optimizing training speed and parameter efficiency.

Model	Precision	Recall	Accuracy	No Classes
Proposed Model (Mobile Net)	0.85	0.80	0.82	6
Transfer Learning	0.96	0.96	0.96	10
DLA	0.84	0.88	0.89	7

TABLE III: Model Evaluation Metrics with the other articles

IV. Conclusion

In this project, we embarked on a quest to enhance waste management and recycling processes through the deployment of a Convolutional Neural Network (CNN) tailored for the classification of garbage images. The CNN model demonstrated the capability to efficiently categorize waste into six distinct groups: cardboard, glass, metal, paper, plastic, and trash. This automation aims to address the escalating issues of waste management exacerbated by global population growth, urbanization, and economic activities. We showed that in this particular case MobileNet Architecture overcomes ResNet or customized CNN architectures. On the other hand, it is a bit short of beating those indicated in other articles, which may become the topic of next projects.

V. Future Work

In order to improve accuracy of the models and get better results more pictures of garbage should be gathered, especially for a 'trash' group. For now different groups are imbalanced and not big enough. More various photos should be collected and based on that even more, additional variations should be artificially generated. Oversampling for small groups doesn't give such good results, especially later models won't be good enough while using it for real cases. When there is more data, more complicated models can be used too. We can fine-tune learning rates, number of layers, number of units per layer, types of optimizers etc. to find the best configuration for the model. We can also try more popular already designed architectures for image processing tasks and fine-tune them.

References

[1] M. Malik, S. Sharma, M. Udin, 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," *MDPI*, April 2022. DOI: 10.3390/su14127222. Available: https://www.mdpi.com/2071-1050/14/12/7222.

[2] A. U. Gondal, M. I. Sadiq, T. Ali, M. Irfan, A. Shaf, M. Aamir, M. Shoaib, A. Glowacz, R. Tadeusiewicz, and E. Kantoch, "Real Time Multipurpose Smart Waste Classification Model for Efficient Recycling in Smart Cities Using Multilayer Convolutional Neural Network and Perceptron," *MDPI*, June 2021. DOI: 10.3390/s21144916. Available: https://www.mdpi.com/1424-8220/21/14/4916.

[3] J. Gunaseelan, S. Sundaram, and B. Mariyappa, "A Design and Implementation Using an Innovative Deep-Learning Algorithm for Garbage Segregation," *MDPI*, July 2023. DOI: 10.3390/s23187963. Available: https://www.mdpi.com/1424-8220/23/18/7963.

[4] S. Kunwar, "Managing Household Waste Through Transfer Learning," January 2024. DOI: 10.48550/arXiv.2402.09437. Available: https://arxiv.org/abs/2402.09437.

VI. Appendix

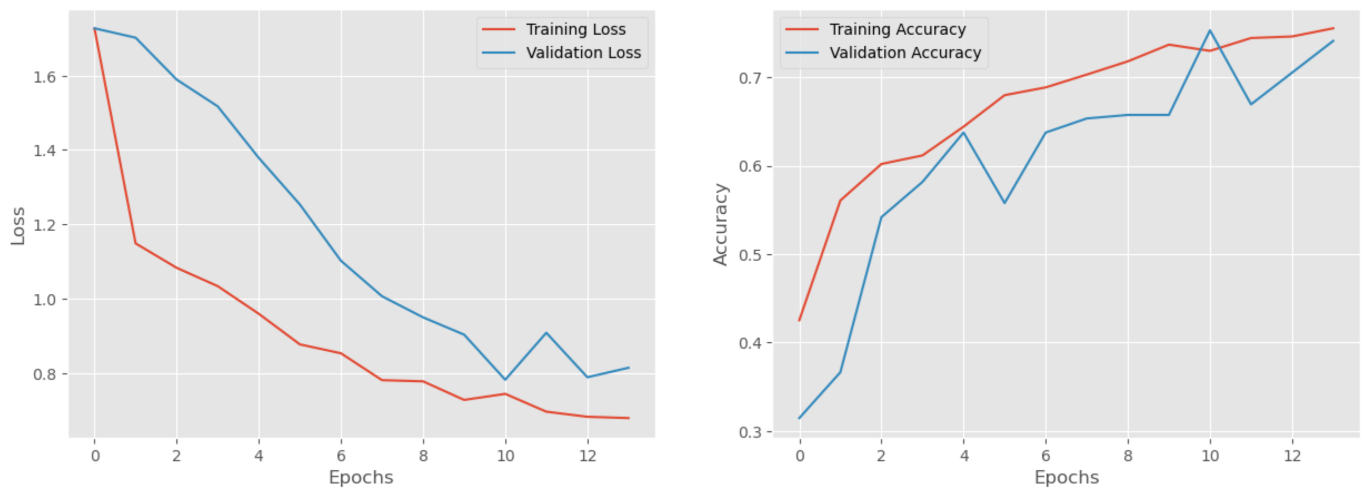


Fig. 4: Loss and Accuracy for Not Regularized Custom Model

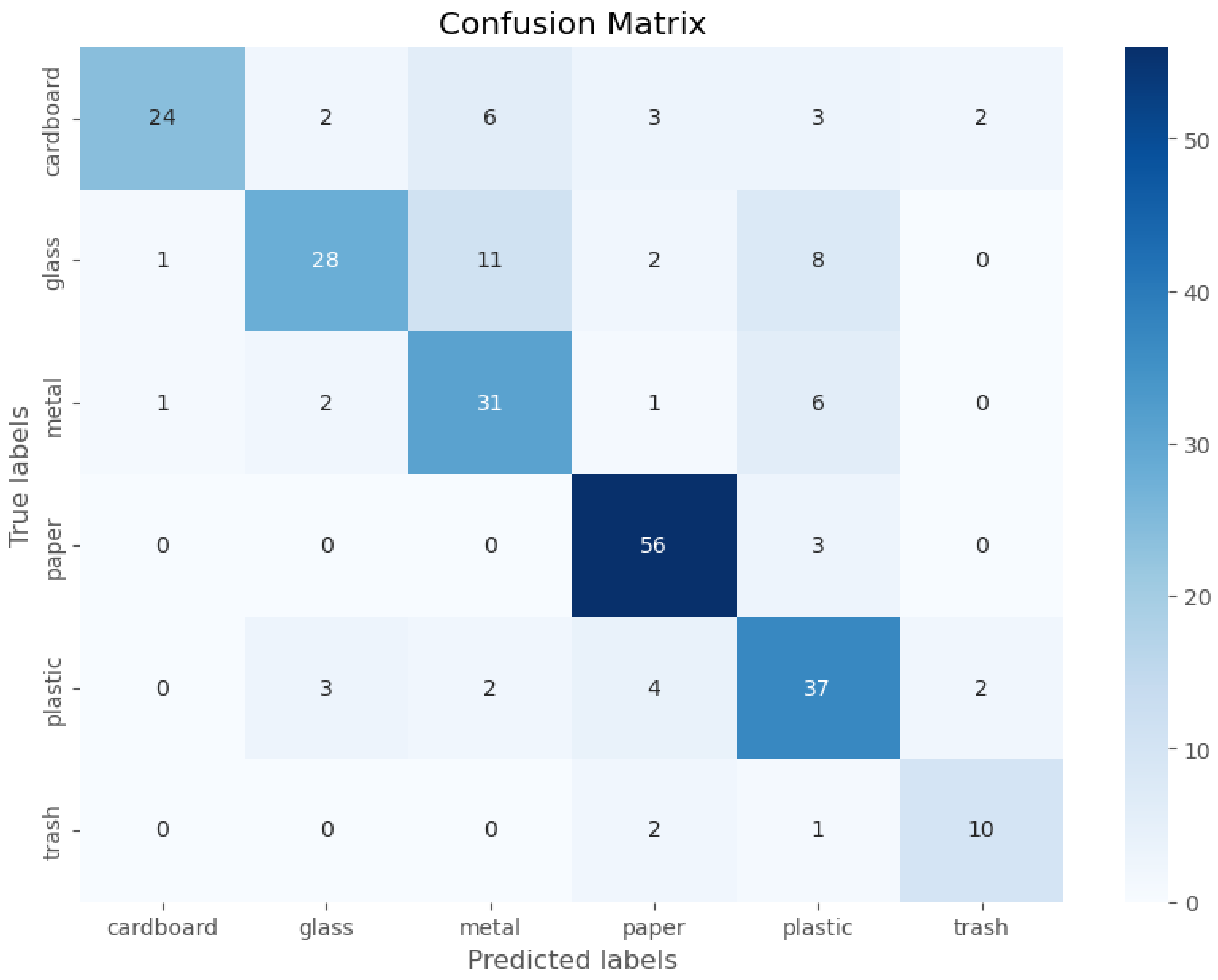
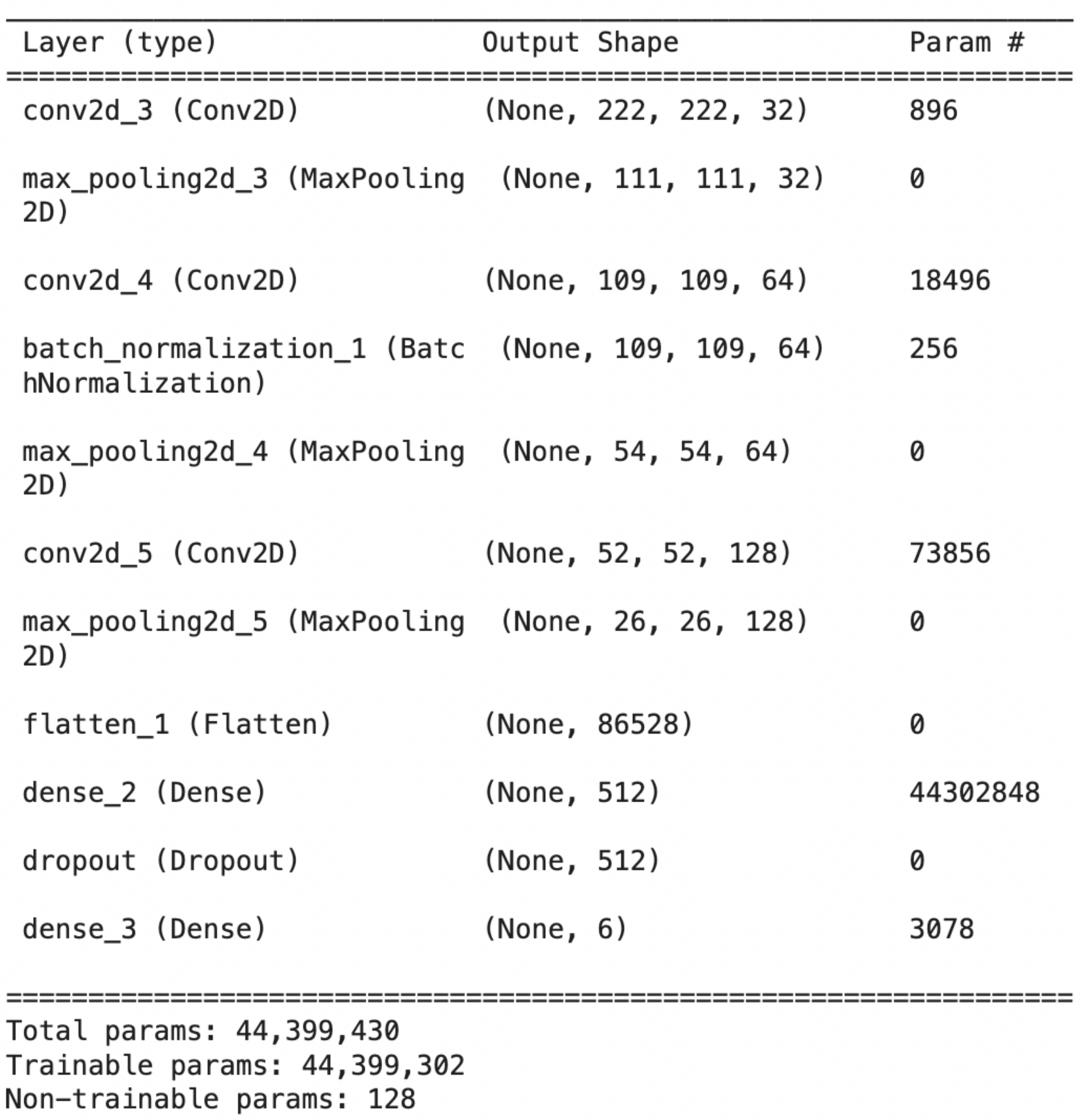


Fig. 5: Confusion Matrix for Not Regularized Custom model



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 222, 222, 32)      896

 max_pooling2d_3 (MaxPooling  (None, 111, 111, 32)     0
 2D)

 conv2d_4 (Conv2D)           (None, 109, 109, 64)      18496

 batch_normalization_1 (Batc  (None, 109, 109, 64)     256
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 52, 52, 128)       73856

 max_pooling2d_5 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 86528)             0

 dense_2 (Dense)             (None, 512)               44302848

 dropout (Dropout)           (None, 512)               0

 dense_3 (Dense)             (None, 6)                 3078

=================================================================
Total params: 44,399,430
Trainable params: 44,399,302
Non-trainable params: 128
```

Fig. 6: Regularized Custom CNN model architecture

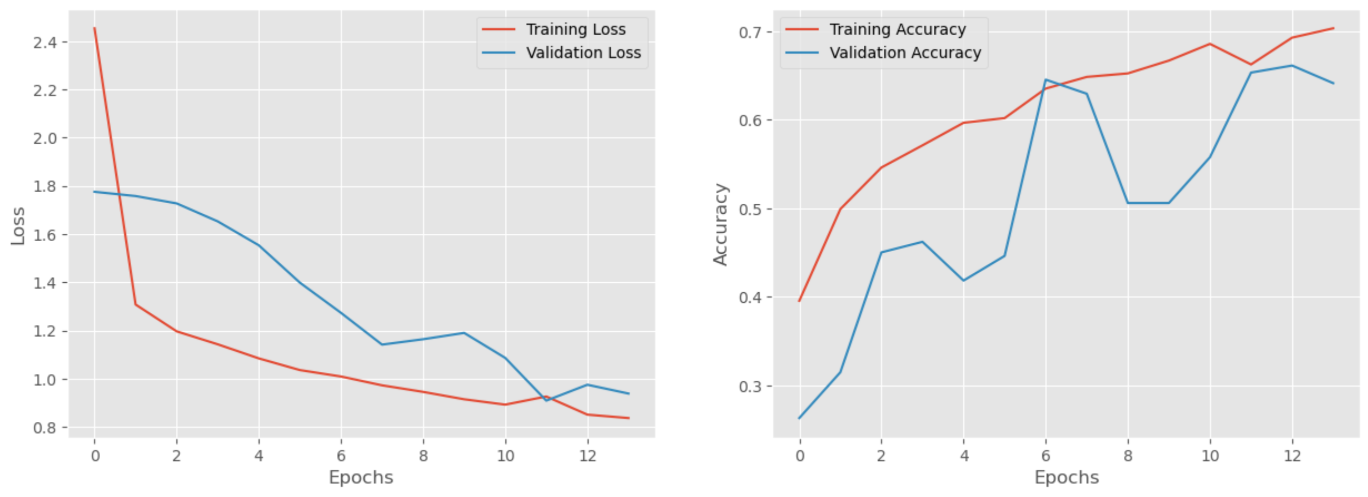
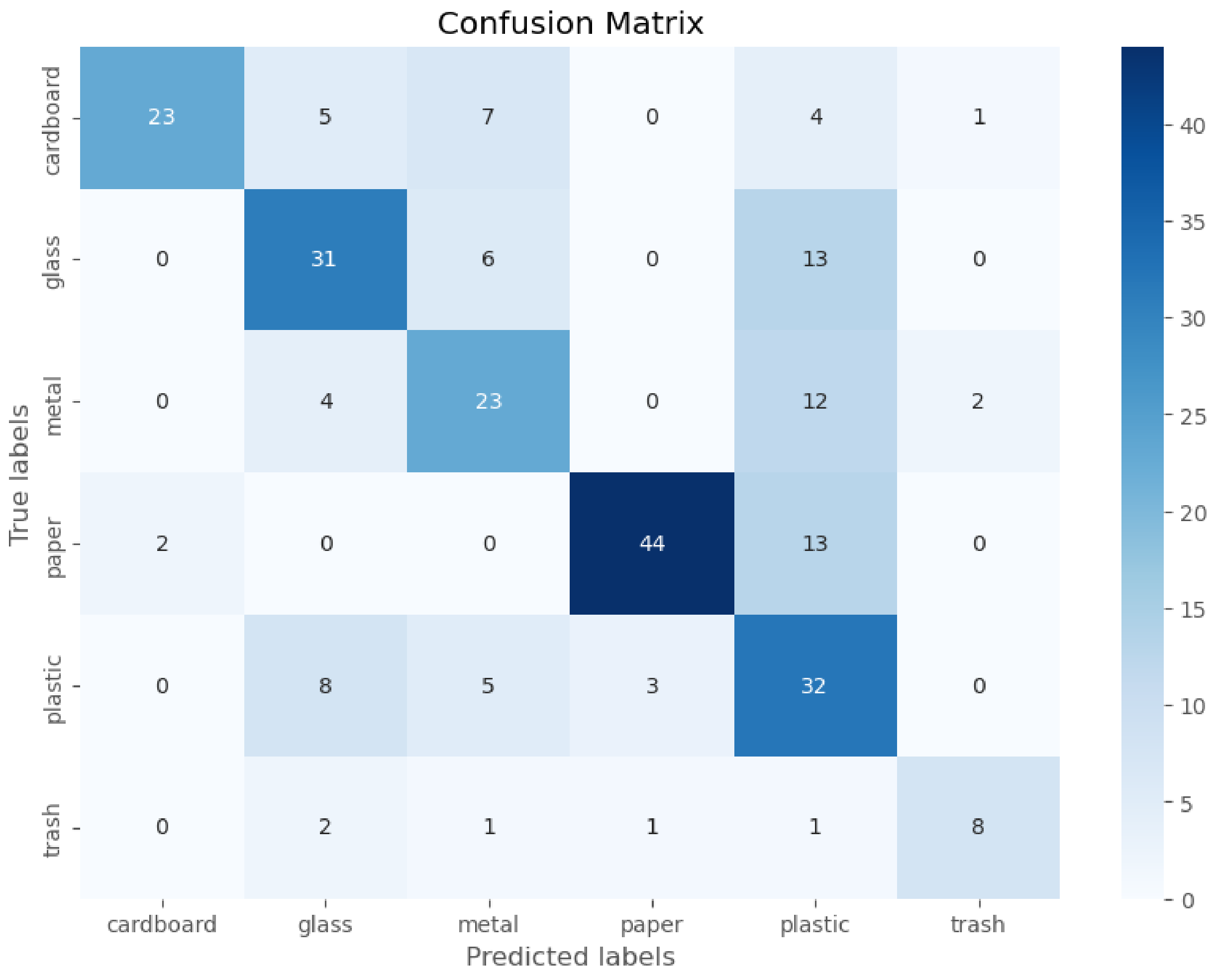
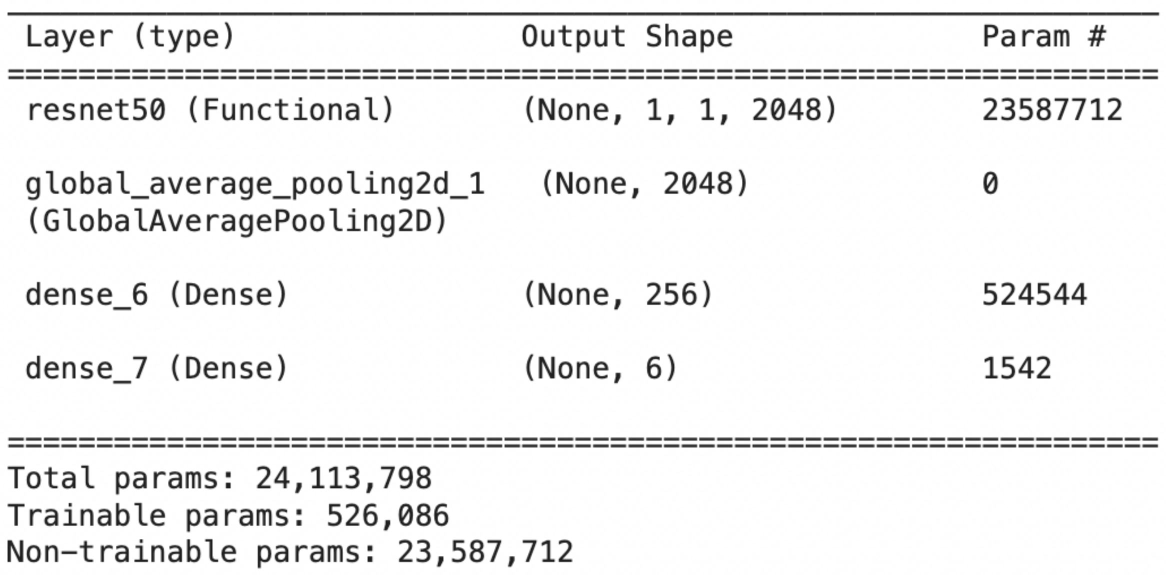


Fig. 7: Loss and Accuracy for Regularized Custom Model



True \ Predicted	cardboard	glass	metal	paper	plastic	trash
cardboard	23	5	7	0	4	1
glass	0	31	6	0	13	0
metal	0	4	23	0	12	2
paper	2	0	0	44	13	0
plastic	0	8	5	3	32	0
trash	0	2	1	1	1	8

Fig. 8: Confusion Matrix for Regularized Custom model



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 1, 1, 2048)        23587712

 global_average_pooling2d_1   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_6 (Dense)             (None, 256)               524544

 dense_7 (Dense)             (None, 6)                 1542

=================================================================
Total params: 24,113,798
Trainable params: 526,086
Non-trainable params: 23,587,712
```

Fig. 9: ResNet CNN model architecture

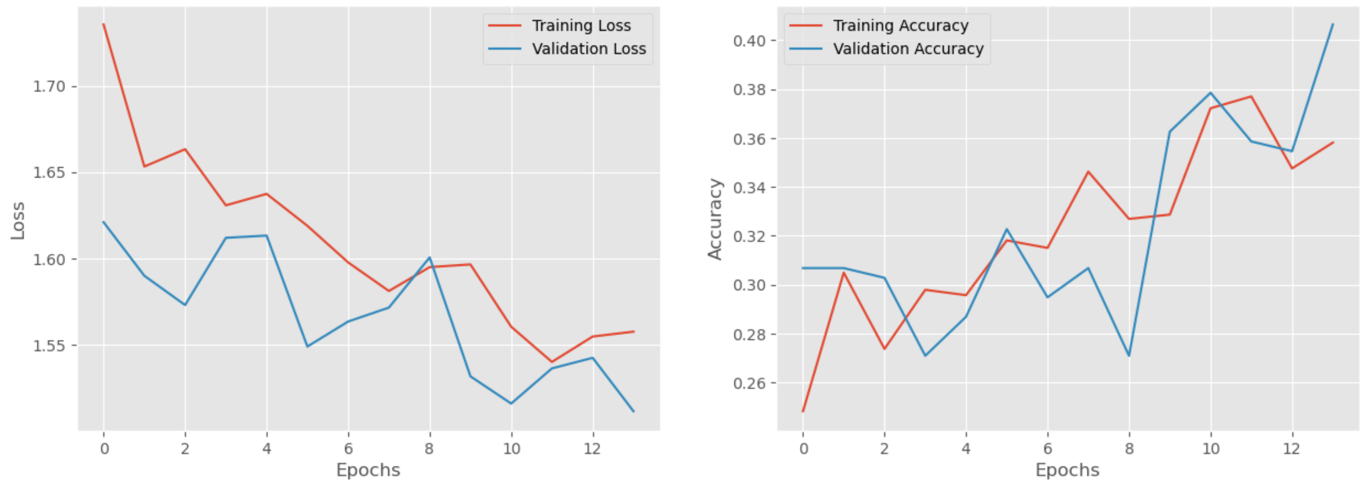


Fig. 10: Loss and Accuracy for ResNet Model

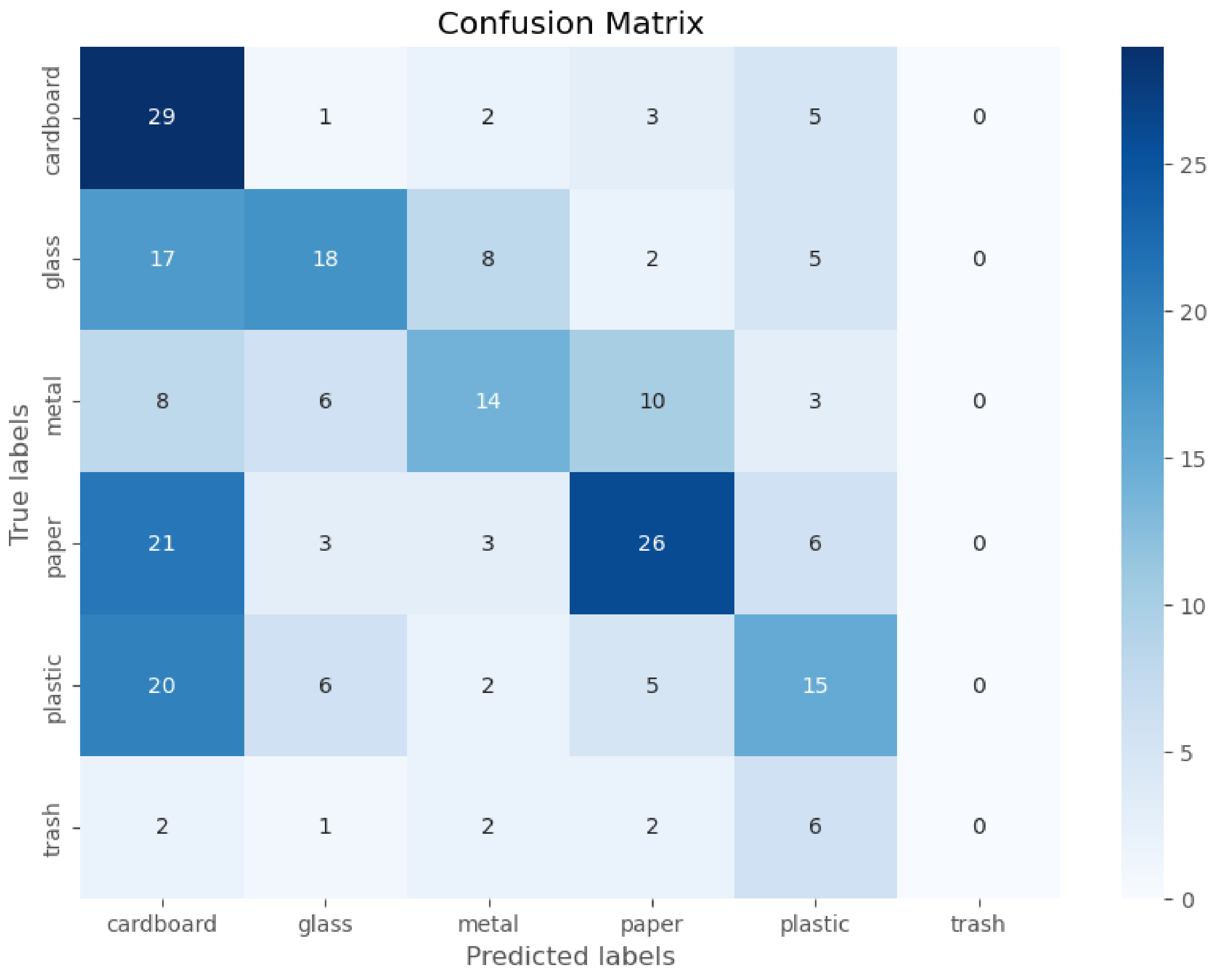
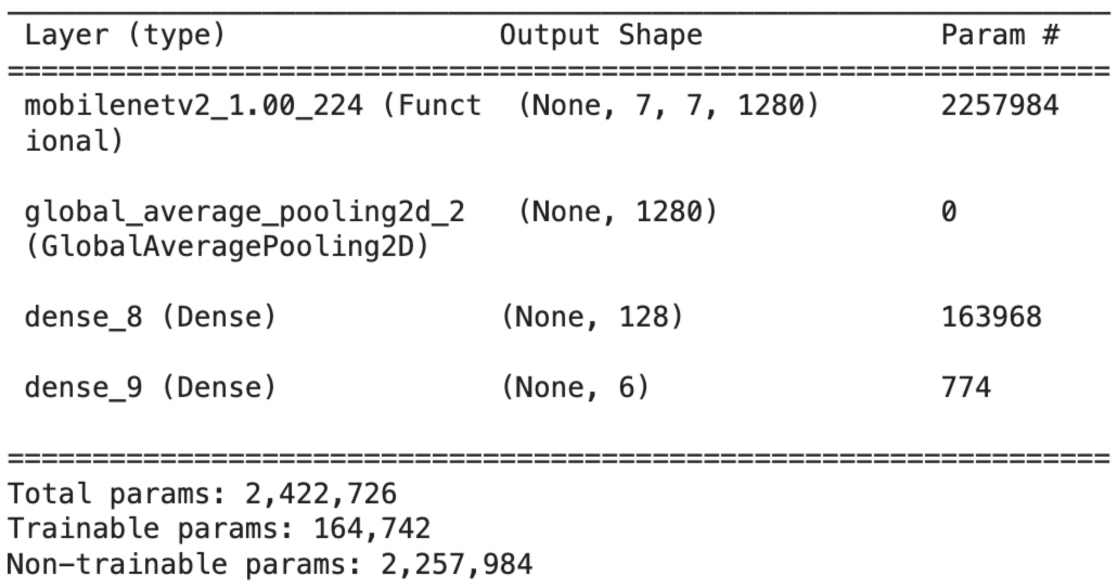


Fig. 11: Confusion Matrix for ResNet model



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 7, 7, 1280)       2257984
 ional)

 global_average_pooling2d_2   (None, 1280)             0
 (GlobalAveragePooling2D)

 dense_8 (Dense)             (None, 128)               163968

 dense_9 (Dense)             (None, 6)                 774

=================================================================
Total params: 2,422,726
Trainable params: 164,742
Non-trainable params: 2,257,984
```

Fig. 12: MobileNet CNN model architecture

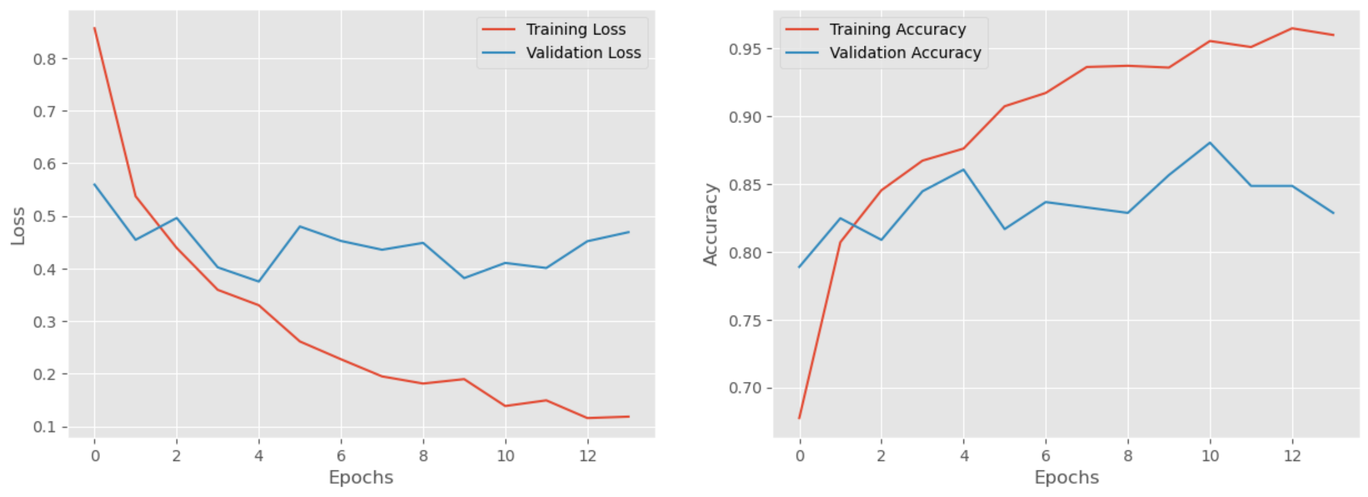


Fig. 13: Loss and Accuracy for MobileNet Model

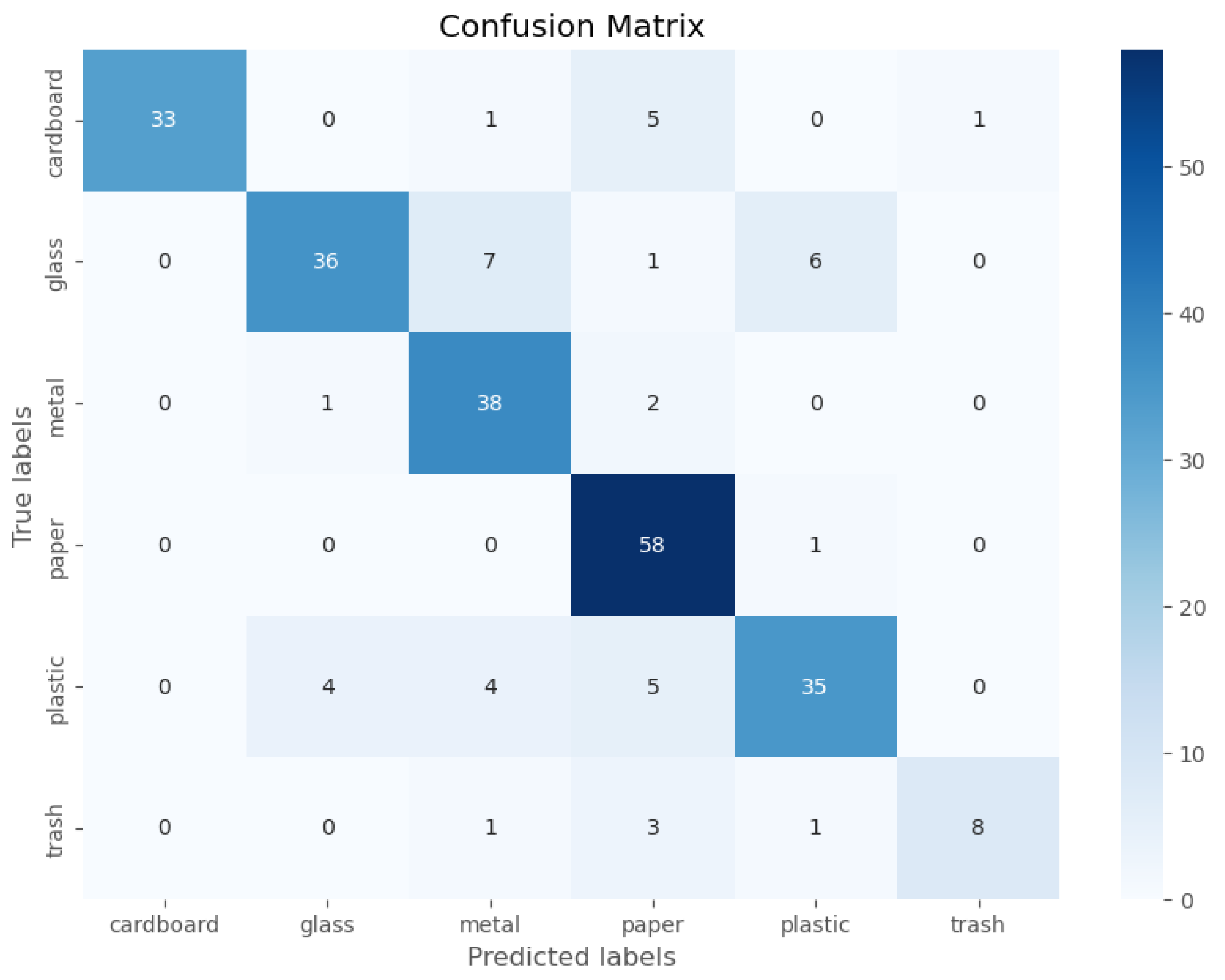


Fig. 14: Confusion Matrix for MobileNet model