Deep Learning for Real Waste Image Classification

Alessio Lani, Enrico Giordani, Lorenzo Arcioni, Marta Lombardi, Valeria Avino

December 28, 2024

Abstract

Improper waste management poses a significant environmental challenge, necessitating innovative solutions for efficient classification and recycling. This study introduces a deep learning approach for the classification of waste images using the Real Waste Dataset[1], a publicly available dataset on Kaggle[2] comprising 4752 images categorized into 9 waste classes. Leveraging pretrained models (ResNet50[3], DenseNet121[4], MobileNetV2[5], VGG-16[6], EfficientNetV2[7], and Swin Transformer[8]) alongside a custom CNN, the proposed pipeline integrates data augmentation, class weighting, and hyperparameter optimization to address class imbalance and enhance model robustness. Among the models tested, MobileNetV2 achieves a remarkable accuracy of 93.28%, making it a highly efficient choice for resource-constrained applications. Despite these successes, challenges persist in the classification of heterogeneous categories like "Miscellaneous Trash", highlighting areas for future improvement to refine accuracy across all waste types.

1 Introduction

Waste management has become one of the most critical environmental challenges of our time. Every year, millions of tons of waste are improperly disposed of, leading to widespread pollution, depletion of natural resources, and damage to ecosystems. Addressing this issue requires not only better infrastructure but also innovative tools to streamline the process of waste sorting and recycling.

A key challenge lies in the classification of waste. Real-world scenarios often involve diverse materials, unpredictable conditions, and overlapping categories, making this task particularly complex. Automating waste classification offers the potential to significantly reduce human error and improve the efficiency of recycling systems. By harnessing modern technological solutions, we can take meaningful steps toward minimizing the environmental footprint of waste and fostering sustainable practices.

This project aims to tackle these challenges by exploring cutting-edge approaches that can enable more accurate and efficient waste classification.

2 Related Works

Recent efforts in waste classification aim to overcome dataset limitations and improve model performance in real-world applications. A notable study by [1] introduced two datasets, DiversionNet and RealWaste, to test waste classification models under different conditions. DiversionNet features pristine samples from datasets like TrashNet, while RealWaste consists of real-world images from landfills, reflecting contamination and material deformation.

Five CNN architectures—VGG-16, DenseNet121, Inception V3, InceptionResNet V2, and MobileNetV2—were evaluated. Models trained on RealWaste significantly outperformed those trained on DiversionNet. Inception V3 emerged as the top performer with 89.19% accuracy and balanced metrics (precision: 91.34%, recall: 87.73%, F1-score: 90.25%) on RealWaste, demonstrating its effectiveness in handling the complexity of real-world waste.

The study revealed that training on pristine datasets like DiversionNet causes overfitting to idealized features, limiting generalization. Performance gains on RealWaste, such as a 58.83% accuracy improvement for VGG-16 and 31.50% for Inception V3, highlight the importance of realistic data. Deeper architectures like DenseNet121 and Inception V3 showed superior adaptability to the diverse feature space of waste compared to shallower networks like VGG-16.

This research underscores the critical role of authentic datasets like RealWaste in developing reliable waste classification systems, offering a strong foundation for advancing sustainable waste management technologies.

Similarly, the work by [9] explores the use of pre-trained CNN models, including InceptionV3, MobileNet, and ResNet50, for classifying waste into seven categories.

3 Proposed Methods

To tackle the challenges of waste classification, we designed a comprehensive approach that combines advanced deep learning architectures with robust preprocessing and optimization techniques. The proposed method leverages state-of-the-art pretrained convolutional neural networks (CNNs) and transformer-based models, starting from a baseline custom CNN tailored for the task.

3.1 Data Preprocessing and Augmentation (*Lorenzo*)

The process begins with preprocessing the images from the Real Waste Dataset. Each image is resized and transformed into tensors to ensure uniformity and compatibility with deep learning frameworks. Following preprocessing, the dataset is divided into three subsets:

- **Training set (80%)**: Used to train the models.
- **Validation set (10%)**: Used to fine-tune hyperparameters and evaluate model performance during training.
- **Test set (10%)**: Held out for final performance evaluation.

Data augmentation is applied exclusively to the training set to improve generalization and robustness. Augmentation techniques include random flipping, rotation, and zooming, which simulate variations encountered in real-world scenarios. These augmentations effectively increase the training set size by a factor of four, enhancing diversity and reducing overfitting. Thus, each image in the training set is accompanied by three additional augmented versions, generated using random transformations.

To visualize the impact of augmentation, bar charts are included in Figure 3 to show the class distribution of the dataset before and after augmentation: Figure 1 representing the overall dataset and Figure 2 illustrating the augmented training set distribution.

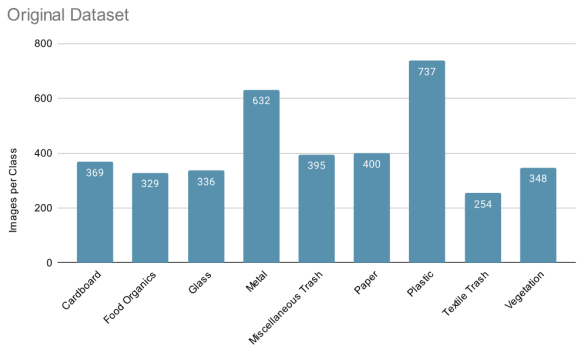

Figure 1: Class distribution in the original dataset.

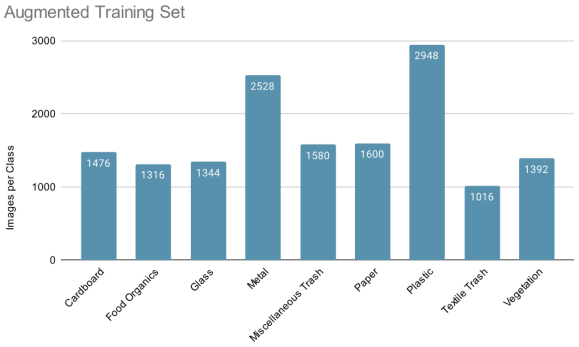

Figure 2: Class distribution in the augmented training set.

Figure 3: Class distributions before and after data augmentation.

To handle image loading and preprocessing efficiently, a custom PyTorch[10] Dataset class, named *RealWasteDataset*, was developed to efficiently handle image loading and preprocessing. This class dynamically retrieves images, applies transformations such as resizing and normalization, and supports mapping class labels to descriptive names. To enhance flexibility, the dataset leverages Pandas[11] DataFrames for augmentation, associating each image path with its label and a transformation pipeline. This approach avoids saving augmented images locally, reducing storage overhead and streamlining experimentation.

3.2 Architectures and Models

Pretrained Models We fine-tuned the following state-of-the-art pretrained architectures by replacing their final classification layers to adapt to the nine target waste categories: ResNet50 *(Lorenzo)*, DenseNet121 *(Lorenzo)*, MobileNetV2 *(Lorenzo)*, VGG16 *(Lorenzo)*, EfficientNetV2 *(Enrico)*, Swin Transformer (Base implementation in PyTorch) *(Marta)*, Vision Transformers (ViT) large 16 (PyTorch implementation)*(Valeria)*.

Custom CNN (*Alessio*) In addition to pretrained models, a custom CNN was designed from scratch to address the unique challenges of waste classification. An illustration of the custom CNN architecture is available on our GitHub repository.

3.3 Addressing Class Imbalance (*Lorenzo*)

Class imbalance in the dataset is addressed by calculating class weights based on the inverse frequencies of each category. These weights are incorporated into the `CrossEntropyLoss` function to ensure balanced learning across all classes. The formula used to compute the weights is as follows:

$$w_i = \frac{\frac{1}{c_i}}{\sum_{j=1}^{K} \frac{1}{c_j}} \cdot K \quad \forall i \in \{1, 2, \ldots, K\} \tag{1}$$

where c_i and w_i represent the frequency[1] and the weight of class i, respectively; and K is the total number of classes. This formula ensures:

- **Inverse Frequency**: $\frac{1}{c_i}$ assigns higher weights to underrepresented classes.
- **Normalization**: The denominator $\sum_{j=1}^{K} \frac{1}{c_j}$ normalizes the weights to maintain consistency across classes.
- **Scaling**: Multiplication by K ensures the weights are scaled proportionally to the number of classes.

By using this approach, the model is incentivized to prioritize learning features relevant to minority classes, which helps in achieving better overall performance and fairness across all classes.

3.4 Optimization Strategy

The models are optimized using the Adam optimizer, a widely adopted method in deep learning for its computational efficiency and ability to handle sparse gradients. The optimization process involves careful tuning of hyperparameters such as the learning rate and weight decay to achieve optimal performance. Specifically, the learning rate is selected from the range 1×10^{-5} to 1×10^{-4}, while the weight decay is fixed at 1×10^{-2}. These values are determined through iterative evaluations based on accuracy and loss metrics computed on the validation set, ensuring a balance between convergence speed and model generalization.

The models are trained for a total of 15 epochs, a duration chosen to optimize performance without overfitting the training data. This training regime, combined with the hyperparameter tuning strategy, promotes efficient convergence and minimizes the risk of overfitting, leading to robust and reliable model performance across all classes.

4 Experimental Results

Figure 4 presents the training and validation performance metrics (loss and accuracy) for architectures introduced earlier, measured over the course of 15 epochs. The top-left sub-figure shows how *ResNet50* and *MobileNetV2* rapidly minimize the training loss, highlighting their efficient learning capabilities. By contrast, the *Custom CNN* experiences a slower decrease in loss, suggesting that it either struggles to extract the most salient features or requires more epochs to converge.

In the top-right sub-figure, the validation loss curves confirm that *ResNet50* and *MobileNetV2* generalize well to unseen data, since their validation loss remains comparatively low throughout training. Meanwhile, the *Custom CNN* shows a consistently higher validation loss, hinting at overfitting or insufficient representational power.

The bottom-left sub-figure, illustrating training accuracy, again places *ResNet50* and *MobileNetV2* in the lead, as they quickly achieve high accuracy on the training set. The *Custom CNN* and the *Swin_B* transformer model, on the other hand, display slower improvements, suggesting they require more fine-tuning or different hyperparameters to reach similar performance levels.

Finally, the bottom-right sub-figure shows the validation accuracy. Here, *ResNet50* and *MobileNetV2* emerge as the top performers, reflecting strong generalization and robust feature extraction. In comparison, the *Custom CNN* and *Swin_B* lag behind, indicating that they struggle to maintain high accuracy on unseen samples.

Given space constraints, we focus on the best-performing (*MobileNetV2*) and the weakest (*Custom CNN*) models for the remainder of the discussion. However, we computed the same set of metrics for all other models; the results are fully available in our GitHub repository. We assessed both on the test set using multiple metrics (confusion matrices, precision, F1-score, recall) to elucidate class-level strengths and weaknesses. For completeness, the corresponding images (e.g., confusion matrices and performance curves) for these two models are provided in the Appendix (figures 6 and 7).

MobileNetV2 consistently surpasses the *Custom CNN* on all classes, achieving higher accuracy and significantly reducing confusion among similar categories like *Plastic* vs. *Miscellaneous Trash*. Notably, *MobileNetV2*

[1] The number of samples of class i.

demonstrates a remarkable increase in *Metal* recall (from 67.1% in the *Custom CNN* to 91.1%) as well as a better distinction between *Miscellaneous Trash* and *Plastic*, thereby improving both precision and recall.

Contrary to our initial expectations, the least-represented class (*Textile Trash*) is not the most misclassified by the *Custom CNN*. Instead, the network struggles heavily with discriminating between *Miscellaneous Trash* and *Plastic*, resulting in low precision for these two classes. This could be rooted in their overlapping visual characteristics (e.g., color or reflective surfaces). Additionally, the *Metal* class—though more frequent in the dataset—exhibits low recall under the *Custom CNN*, suggesting intraclass variability or subtle features that the network fails to capture. *MobileNetV2*, aided by its deeper architecture and pretrained weights, addresses these limitations by identifying more nuanced cues and minimizing confusion across categories. All these insights can be drawn from the confusion matrices shown in Figure 10, with Figures 8 and 9 corresponding to MobileNetV2 and the Custom CNN, respectively.

5 Future Works

A promising direction for future research is to combine the strengths of our top-performing models through an ensemble approach. In particular, a voting classifier that integrates the predictions of *MobileNetV2*, *ResNet50*, *DenseNet121* and *EfficientNetV2*, promising architectures could capitalize on diverse feature representations and decision boundaries, potentially boosting overall accuracy. By aggregating multiple model outputs, such an ensemble often mitigates each classifier's individual weaknesses, leading to more consistent performance across challenging classes like *Miscellaneous Trash*.

Additionally, careful design of the voting mechanism can balance accuracy with computational efficiency, making ensemble models suitable for real-world applications such as automated waste-sorting systems. Techniques like weighted voting, where models that consistently perform well on specific classes are given greater influence, may further optimize classification results without incurring substantial additional complexity.

6 Conclusions

Classifying waste from images remains a challenging task, mainly due to the diversity of materials and ambiguous visual cues. A comprehensive plot (Figure 5) comparing the final test accuracies (evaluated on the test set) of all the models is provided in the Appendix 6. Our experiments show that both *MobileNetV2* and *ResNet50* achieve an accuracy of 93.28%. In contrast, *Inception V3*—the best model used in our reference article [1]—yields 89.19% accuracy, making our solutions more effective on this dataset. Although *ResNet50* matches *MobileNetV2* in accuracy, we highlight *MobileNetV2* because it requires less training time (36 minutes 58 seconds vs. 48 minutes 3 seconds for *ResNet50*), offering efficiency without compromising accuracy.

Despite typically excelling on large-scale datasets, transformer-based architectures underperformed in this scenario. We also observed that *Miscellaneous Trash* remains the most misclassified class across all models, suggesting that additional data or a refined definition of this category could further improve results. Future research may focus on more granular class definitions, extended data augmentation strategies, or domain adaptation techniques to address these persistent classification ambiguities. Such efforts could be particularly valuable for challenging categories like *Miscellaneous Trash* and *Plastic*, as observed during visual inspection (some images are difficult to differentiate even to the human eye), as mentioned in the README file in our GitHub repository.

References

[1] Sam Single, Saeid Iranmanesh, and Raad Raad. Realwaste: A novel real-life data set for landfill waste classification using deep learning. *Information*, 14(12):633, November 2023.

[2] RealWaste Image Classification — kaggle.com. https://www.kaggle.com/datasets/joebeachcapital/realwaste.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[7] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021.

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[9] Sujan Poudel and Prakash Poudyal. Classification of waste materials using cnn based on transfer learning. In *Proceedings of the 14th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '22, page 29–33. ACM, December 2022.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[11] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

Appendix

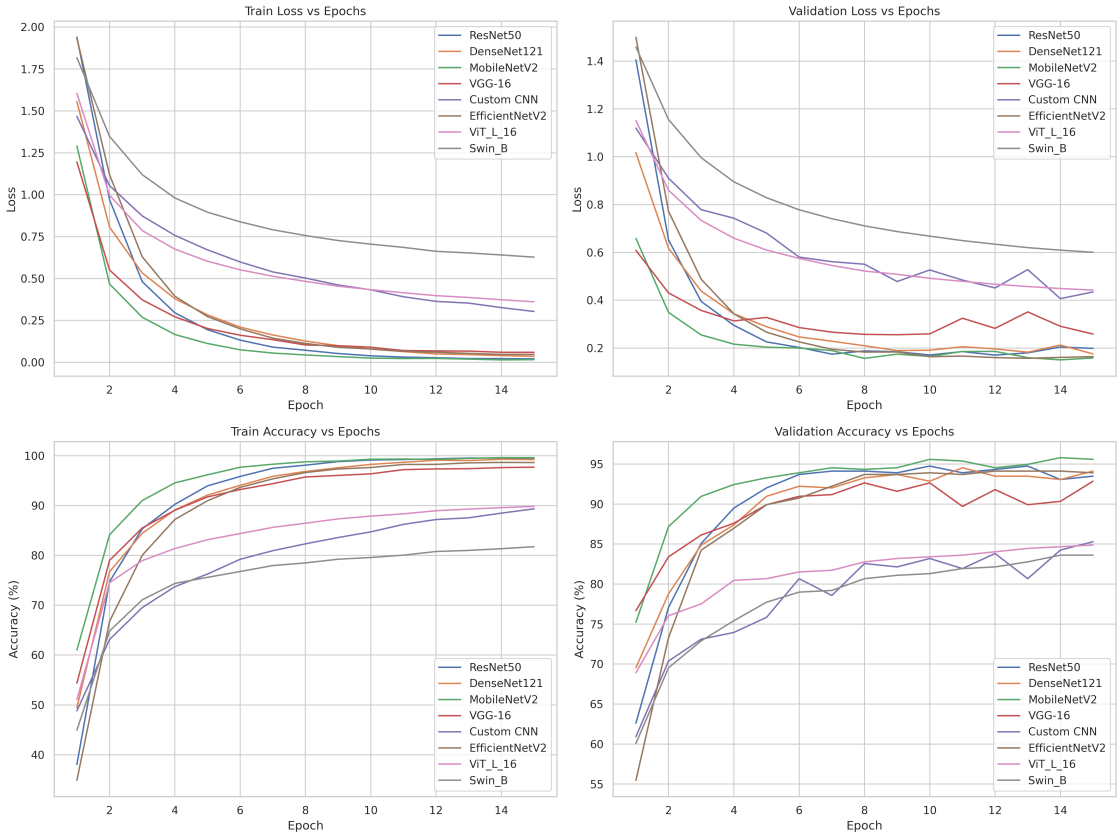

Figure 4: Train and Validation Loss and Accuracy for Each Model Across Epochs

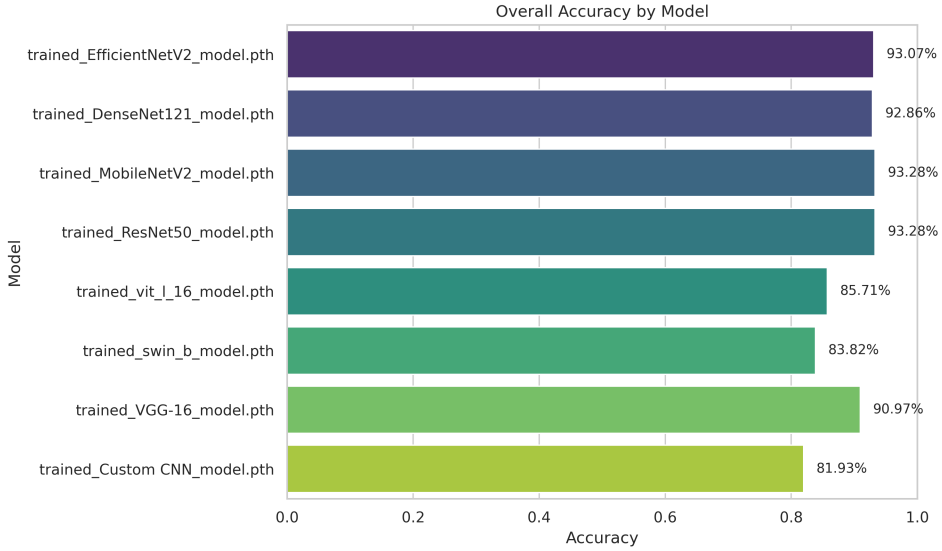

Figure 5: Test Accuracy by Model

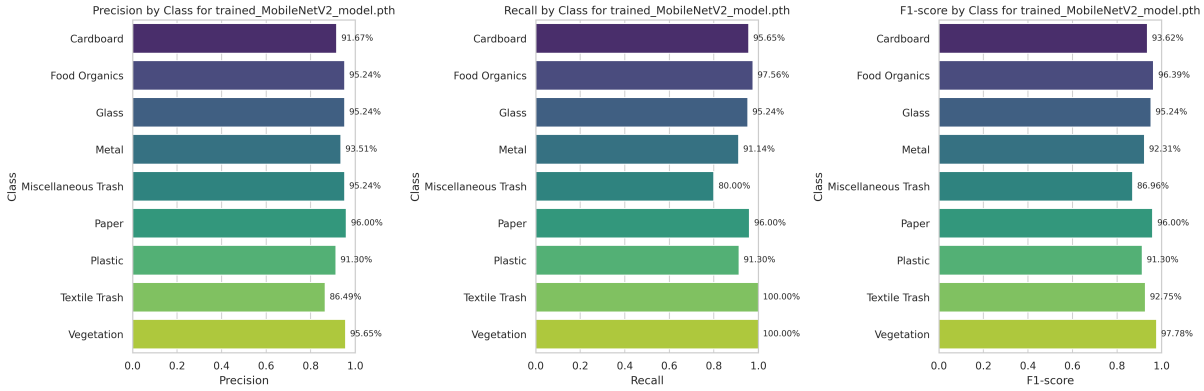

Figure 6: MobileNetv2 Test Metrics: Precision, Recall and F1-Score

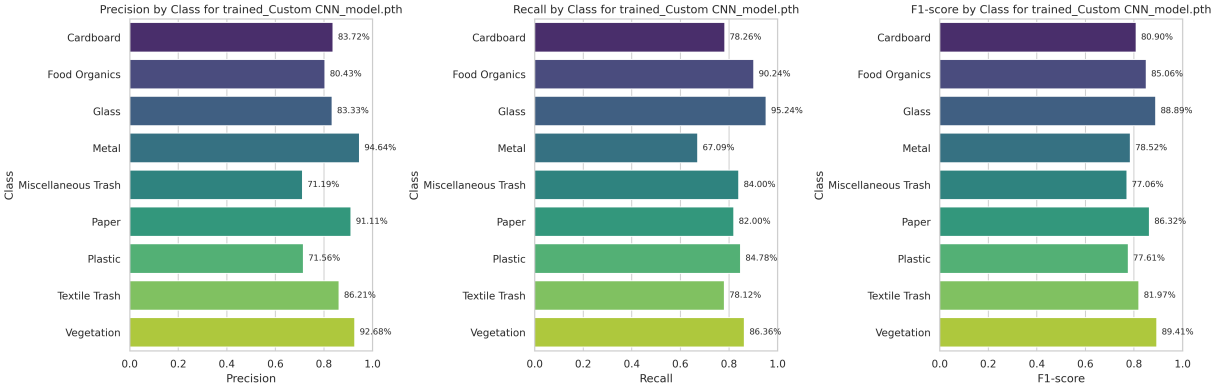

Figure 7: Custom CNN Test Metrics: Precision, Recall and F1-Score

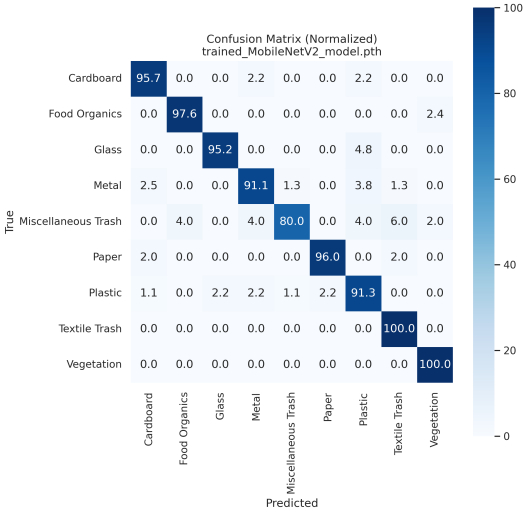

Figure 8: MobileNetV2 Confusion Matrix

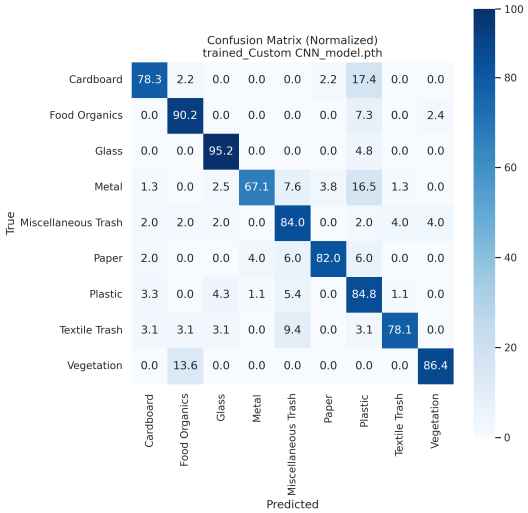

Figure 9: Custom CNN Confusion Matrix

Figure 10: Comparison of confusion matrices for the Custom CNN and MobileNetV2 models