From Garbage Data to Data on Garbage: Developing Custom Machine Learning Models for Sustainable Waste Classification

Final Exam – Report

Machine Learning & Deep Learning [KAN-CDSCO2004U]

Copenhagen Business School

M.Sc. Business Administration & Data Science

Github link

Authors: Moritz von Buchwaldt (Student Number: 175877)

Jasper Schwietring (Student Number: 175857)

Michael Webster (Student Number: 175868)

Moritz Kremmer (Student Number: 175862)

Examiners: Somnath Mazumdar

Number of Pages: 15

Number of Characters: 34,094 (2,273 characters per page)

Date of Submission: 16.05.2025

Abstract

Previous research has shown that pre-trained models are able to accurately classify waste images. Given the increasing amount of waste produced per year and influence on climate change it represents, it is an imperative to address the problem of waste management in a reliable process. Using various custom CNN architecture, this paper analyzes to what extent non-pre-trained models can reliably classify waste images from the RealWaste dataset into nine waste classes. Furthermore, it analyzes to what extent the success of pre-trained models can be replicated. Using three CNN-models and one hybrid CNN-SVM model, we find that replication is, in parts, successful. While plain CNNs struggled to match the 0.89 F1-score set by the pre-trained models (Single et al., 2023) with the highest score being 0.86 set by a model inspired by the VGG-Architecture but enhanced though more modern CNN techniques. Only the hybrid model consisting of the best performing CNN as a feature extractor and a SVM achieved a F1-score of 0.88, close to the 0.89 F1-score. While the dataset provides a controlled benchmark for initial evaluation, future research should explore how lightweight custom CNN architectures perform under more realistic conditions introducing greater variability in input data and testing robustness in industrially relevant environments.

Keywords: Waste Type Identification, Convolutional Neural Network, Image Classification, Data Augmentation, Depthwise Separable Convolution

Table of Contents

1. Introduction & Motivation

Efficient waste management is an increasingly critical global challenge, directly tied to environmental sustainability and reducing climate impact. Mismanaged waste significantly contributes to pollution and greenhouse gas emissions, with landfills and wastewater accounting for approximately 20% of global anthropogenic methane emissions (Global Methane Assessment, 2021). Despite ongoing efforts, recycling rates remain insufficient, with only around 19% of municipal solid waste effectively recycled worldwide in 2020 (Global Waste Management Outlook, 2024). Improving waste classification accuracy is essential to divert more recyclables from landfills and mitigate these negative environmental impacts.

At the same time, the volume of waste produced globally is rapidly increasing, pressuring traditional sorting processes currently employed. These manual systems are not only labor-intensive, but prone to errors, and increasingly unable to keep pace with municipal waste levels, which are expected to rise

by 87% between 2020 and 2040 (Global Waste Management Outlook, 2024). Automated classification methods leveraging convolutional neural networks (CNNs) offer significant potential to scale waste sorting operations efficiently and accurately, surpassing manual methods in both throughput and consistency.

2. Context & Research Imperatives

Recent studies have explored the application of CNNs for waste classification, addressing both performance and real-world applicability. Single et al. (2023) introduce the RealWaste dataset, a collection of landfill waste images, to examine whether CNN models trained on lab-assembled datasets can generalize effectively to real-world conditions. Their work compares several CNN architectures, including VGG-16, InceptionResNetV2, DenseNet121, Inception V3, and MobileNetV2, and evaluates their performance on the RealWaste dataset. The models mentioned all achieved an accuracy between 85–90% on the test set. The study demonstrates the limitations of synthetic datasets and validates the RealWaste dataset's utility for realistic waste classification.

Malik et al. (2022) present a deep learning-based image recognition approach for waste classification aimed at supporting sustainable development. They evaluate the performance of the EfficientNet-B0 architecture and compare it to deeper variants such as EfficientNet-B3. The latter model variant achieved a classification accuracy of 85%. The study emphasizes computational efficiency, demonstrating that lightweight models like EfficientNet-B0 can achieve high classification accuracy across region-specific datasets. Their findings support the adaptability and sustainability of CNN-based methods in municipal solid waste management.

Given the limitations of previous work, which often rely on simplified datasets featuring generalized stock images of waste (Kumsetty & Nekkare, 2022), we extend on the literature by using a more generalizable dataset. Simplified approaches struggle with the complexity and variability of real-world waste conditions. In contrast, our research investigates custom CNN architectures specifically designed and trained on the RealWaste dataset (Single et al., 2023), consisting of authentic landfill images. By doing so, we aim to evaluate whether tailored CNN models can effectively address the real-world challenges of waste classification and potentially match or surpass the accuracy of traditional pretrained methods (Single et al., 2023), on representative datasets.

To explore these possibilities, this report addresses two primary research questions:

1. *To what extent can self-trained models reliably classify images from the authentic RealWaste dataset?*
2. *To what extent can self-trained models match pre-trained CNN performance in waste classification?*

3. Conceptual Framework

3.1 Data Preparation

This project is based on the "RealWaste" dataset from Single et al. (2023), which contains real-world landfill waste images organized into nine classes (Figure 1). The dataset was split into training, validation, and test sets to ensure proper, unbiased evaluation. To avoid data leakage, the splitting was done prior to any data augmentation. The validation set was strictly used to tune the hyperparameters of the models, while the test set was solely used for a final evaluation.

Given the consistent image dimensions and aspect ratios, we focused preprocessing on improving model robustness through augmentation. These augmentations increased the size of the training dataset, helping the models learn diverse visual patterns and improving their ability to generalize. Before loading, the images were resized according to the specific input requirements of each model. Additionally, input values were rescaled and class weights were computed and applied during training to address imbalances across the nine waste categories.

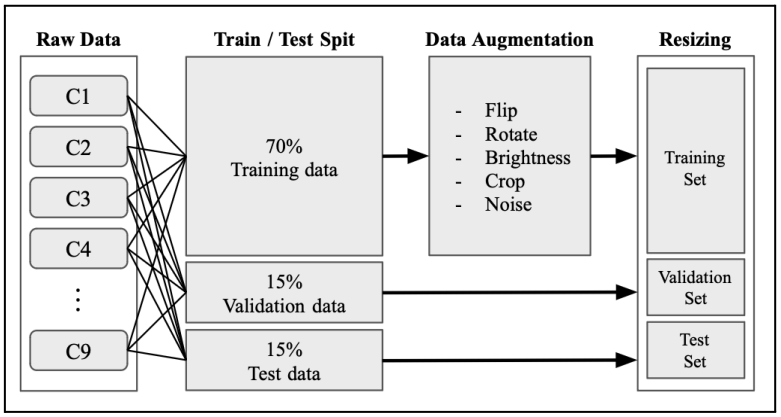

Figure 1: Data Preparation Process

3.2 Training Strategy

Our training strategy involved comparing three different CNN architectures and one hybrid model combining deep learning with classical machine learning (Figure 2). All models were trained on the same augmented dataset to ensure comparability. Each model's performance was measured on the validation and test sets using accuracy, precision, recall, and F1-score. A random search was conducted for hyperparameter tuning of both CNN-2 and CNN-3 architectures. The search was run for 60 iterations, with each model trained for ten epochs per iteration. For instance, in the case of CNN-2, it was applied on the learning rate, dense layer size, and the dropout rate in the dense layer. Only the CNN-2 + SVM setup, a combined approach of random search for the CNN component and grid search for the SVM classifier was applied. This multi-model training allowed exploring both the architectural and hybrid dimensions of CNN-based waste classification.

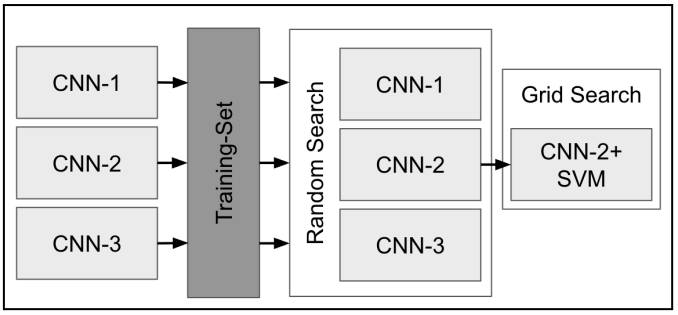

Figure 2: Training Strategy

Batch sizes were adjusted to the demands of the individual models. Adam was kept as an optimizer and cross-entropy loss (with logits) as the loss function for all models, while the learning rate was adjusted respectively. Early stopping was applied with a patience of five on validation loss. As mentioned above, class weights were computed and applied during training to tackle class imbalance in the dataset. Furthermore, callbacks were employed to stabilize and optimize the training process.

4. Methodology

4.1 Data Description

The RealWaste dataset (Single et al., 2023) contains a total of 4,752 high-resolution RGB images of waste items across nine classes. These are *Cardboard*, *Food Organics*, *Glass*, *Metal*, *Miscellaneous Trash*, *Paper*, *Plastic*, *Textile Trash*, and *Vegetation*. Each image has a fixed resolution of 524×524 pixels and follows a standardized setup: a single object placed centrally on a uniform background - usually concrete

Despite a generalized image structure, classification remains non-trivial. A challenge stems from an ambiguity in classification of certain objects. Several images exhibit features that plausibly belong to more than one class e.g., a plastic dinosaur toy classified as “Miscellaneous Trash” (Figure A1). This is particularly pronounced in the “Miscellaneous Trash” class, which acts as a residual category for items that are clearly identifiable as objects or do not clearly match predefined material groups. Such examples are found in Figure A2. Further analysis will be conducted in the discussion section**.**

The dataset exhibits a class imbalance (Figure A3). Plastic is the most represented category with over 900 samples, followed by Metal (791), with all other classes consisting of between 300 and 500 samples. This imbalance poses a risk of bias during model training, especially in underrepresented classes where fewer examples can lead to poorer generalization and reduced recall.

Furthermore, although the backgrounds are generally standardized across classes, items within the same class occasionally share further visual context cues. These include attributes such as similar lighting conditions or placement on differently textured concrete surfaces. This may introduce biases

driven by features unrelated to the object itself (lighting/ background), particularly when class attributes are generally visually subtle.

Pixel-level analysis of the dataset further supports this view. As shown in Figure A4, the brightness distribution is centered around a mean normalized pixel intensity of 0.61 (std: 0.17), indicating generally well-lit images with low variability in exposure.

4.2 Data Preprocessing

As a precursor for the analysis, data was preprocessed to ensure stability across the models. Given the susceptibility of CNNs to biased, mismanaged inputs and susceptibility of training duration on image size, the images need to be standardized into one recurrent form. The following steps were taken: (1) splitting into training, test and validation sets, (2) adding augmentations of images to the training data and (3) resizing of images dependent on the model architecture. To allow comparison of models, the augmented images were saved. These newly constructed image folders consisting of the original and augmented images, were then used as inputs for all models. Due to varying image resolutions across models, the images' maximum scale of 524×524 was preserved.

Firstly, to allow validation of the training outcomes, validation and test datasets were constructed on the initial dataset of 4,752 images through a 70-15-15 split. The overall dataset thus allowed the adjustment of hyperparameters through the validation set, and unbiased testing across classes. In total, the preliminary training set consisted of 3,326 images, the validation set of 710 images, and the test set of 719[1] images (Figure A6).

Then, data augmentation was used to extend the training dataset and create new instances of images. The operations included flipping, rotating, cropping, introduction of noise and adjustment of brightness. Given the general homogeneity of the images and limited size of the dataset, this extended the scope and variability of the data, allowing models to generalize more easily. Consequently, normalization was not conducted, as heterogeneity was the limiting factor, rather than homogeneity. The exact values of each operation are summarized in Table A1. Exemplary outcomes of the augmentations are represented in Figure A5**.** In total this extended the training set to 19,938 images[2].

Finally, before training, all images were rescaled from 524×524 to the format in accordance with each model's architecture. The exact scale for each of the models was 200×200 pixels for CNN-1 and 256×256 for all other models, which is further discussed in Section 4.3**.**

[1] The test set is slightly larger than the validation set due to rounding in the data splitting.
[2] No oversampling or class weighting was applied, as the goal was to evaluate how well the models could generalize under natural class imbalance conditions.

4.3 Modelling Framework

The underlying architecture varied across models. This extended from the input resolution, number of layers, types of layers, to the final activation functions. How each model differs is outlined in the following section.

4.3.1 CNN-1 - VGG & Le-Net5 Inspired

As a baseline model (hereinafter CNN-1), we implemented a CNN inspired by the VGG and LeNet-5 architectures. These represented simple, widely-used CNN-models, with the goal of evaluating the classification performance on a real-world dataset before exploring more advanced architectures.

The CNN-1 architecture comprises four convolutional layers with 16, 32, 64 and 128 filters respectively, each with a receptive field of 3×3 and a ReLU activation function. Following each of the convolutional layers is a max-pooling function to decrease the spatial-dimensionality. The architecture ends with one flattening layer and one fully connected dense layer with 128 neurons and a ReLU activation function. To mitigate overfitting, dropout with a rate of 0.5 was applied before the final output layer, which is a dense layer producing raw logits for the nine waste classes. A rescaling layer at the input initially normalizes the pixel values.

4.3.2 CNN-2 - Improved Architecture with Additional Convolutional Layer

The CNN-2 architecture leverages CNN-1 as a foundation and incorporates a range of architectural best practices to improve learning capacity and generalization. The input resolution was increased to 256×256 to help the model capture finer textures in waste classification, where surface details often matter more than shape, which is in line with Sabottke and Spieler's (2020) findings. To ensure more abstract and higher level features could be explored, we also deepened the architecture by adding a fifth convolutional layer with 256 filters.

Additionally, instead of flattening the feature maps as in CNN-1 through fully connected layers (dense layer accounting for over 1.6 million parameters), CNN-2 uses a Global Average Pooling layer. This roughly keeps the model's complexity, but decreases overfitting. To counteract exploding and vanishing gradient problems, batch normalization layer after each convolutional layer was added. Furthermore, L2 kernel regularization is used to penalize large weights and enforce smoother decision boundaries. This allows the model to generalize better to unseen data by reducing the risk of overfitting, especially in deeper architectures.

4.3.3 CNN-3 - Depthwise Separable Convolution

Building on the foundations laid by CNN-1 and CNN-2, the third model (CNN-3) was designed to investigate whether increased depth could improve classification performance by learning more

abstract, high-level representations of waste items. To avoid overfitting while scaling depth, the model uses a lightweight yet expressive building block: Depthwise Separable Convolutions (DSC). This approach significantly reduces computational complexity compared to standard convolutions by splitting spatial and channel-wise filtering into separate operations (Chollet, 2017).

The CNN-3 architecture consists of one convolutional layer followed by four consecutive DSC blocks, each comprising a 3×3 depthwise convolution followed by a 1×1 pointwise convolution, with batch normalization and ReLU activations applied after both steps. To encourage generalization, L2 kernel regularization is applied to all pointwise convolutions, and each block concludes with max pooling to reduce spatial resolution. The number of filters progressively doubles from 16 to 256 across the five blocks, allowing the network to build increasingly abstract representations.

CNN-3 continues the trend toward depth in our set of models, aiming to enhance classification accuracy while retaining efficiency through the use of separable convolutions.

4.3.4 CNN-2 + SVM

Across academia, it has been shown that replacing a CNN's softmax head with a Support Vector Machine (SVM) classifier can improve classification performance (e.g., Xue et al., 2016). Drawing inspiration from this previous work, the approach was adopted for waste classification. The model uses the best performing CNN (CNN-2).

The dense layers are removed from the model and the features are then extracted from the remaining network. Standard scaling is applied for use in the SVM. The resulting feature vectors are passed to the SVM RBF kernel to capture nonlinear class boundaries. Class weighting is also passed as a parameter to compensate for class imbalance in the dataset, ensuring that underrepresented classes have a proportionally greater influence on the decision boundary. To find the most effective C and gamma parameters for our data we use a grid search. From this, the optimal combination (C=10, gamma=0.01) is derived. The SVM uses a one-versus-rest (OvR) strategy by default, training one binary classifier per class to handle the multi-class nature of the problem.

4.4 Evaluation Metrics

To evaluate the final model performance, the weighted F1-score of the test set was used, as it captures the harmonic mean of precision and recall while accounting for class imbalance in the dataset. This metric is particularly relevant in the context of automated waste sorting, where both precision (contamination rate) and recall (recovery rate) are operationally critical. High precision is essential to minimize contamination, which increases post-sorting costs and reduces the marketability of recycled materials. *"How many of the items that we sorted into plastic are actually plastic?"* A high recall is

necessary to judge the efficiency of the recycling process, as only correctly classified materials can be recovered and reused. *"How many of our plastic items actually end up in plastic?*

During training, however, the F1-score is not available. Therefore, we additionally track validation accuracy and loss to monitor model convergence and potential overfitting. While less nuanced than the F1-score, these metrics offer valuable insights into the training dynamics and early stopping. Notably, in our experiments, accuracy, loss, and weighted F1-score on the test set never differed by more than one percentage point, further justifying the use of accuracy and loss as reliable proxies during training. Hereinafter, the weighted F1-score will be referred to as F1-score.

5. Results

This section presents a detailed comparison of the developed CNN models in terms of classification performance, training behaviour, and architectural trade-offs. All information and statistics on the models referenced in this section can be found in the Appendix between Figure A7 and A23.

5.1 Performance Evaluation & Training Analysis

Metric \ Model	**CNN-1**	**CNN-2**	**CNN-3**	**CNN-2 + SVM**
F1-Score (weighted)	0.70	0.86	0.80	0.88
Metal (precision || recall)	0.67 || 0.78	0.84 || 0.87	0.80 || 0.83	0.87 || 0.87
Plastic (precision || recall)	0.71 || 0.64	0.85 || 0.86	0.77 || 0.77	0.83 || 0.89
Miscellaneous (precision || recall)	0.51 || 0.60	0.71 || 0.68	0.71 || 0.60	0.75 || 0.73
Textile (precision || recall)	0.60 || 0.53	0.80 || 0.82	0.72 || 0.78	0.88 || 0.88

Table 1: Model Test Performance

Among the evaluated models, the CNN-2 (moderate depth with regularization) combined with a Support Vector Machine (SVM) classifier achieved the highest weighted F1-score of 0.88. The plain CNN-2 followed closely with a score of 0.86. CNN-3, which employed DSCs, reached a weighted F1-score of 0.80, while CNN-1 (VGG inspired), yielded the lowest score at 0.70 (Table 1).

The training dynamics differed notably across models. CNN-1 started with high validation accuracy at 0.48 that increased until reaching a peak at 0.73 in epoch nine. Training accuracy surpassed validation accuracy in epoch four and increased the gap by more than 0.1. Training loss continued to decrease almost linearly throughout training while validation loss reached its minimum at epoch five. This growing gap between training and validation loss indicates signs of overfitting beginning around epoch four. The early overfitting observed in CNN-1 can be attributed to the model's limited depth, which restricts its ability to extract abstract and generalizable features. Moreover, the use of a flatten layer followed by a dense layer introduces over 1.6 million trainable parameters in a single layer. This

high parameter count renders the model overly expressive, increasing the risk of memorizing training data rather than learning representations that generalize well to unseen examples. Lastly, CNN-1 was also only model trained on a pixel size 200×200 resulting in ~39% less pixels to extract features out of which could be a reason for the lower prediction capabilities of the model than 256×256 pixels.

CNN-2 almost achieved a validation accuracy of 0.50 after the first epoch and continued to improve until epoch 25 where it stabilized at 0.88. Validation loss continued to decline until epoch 25, leading to a final training cutoff at epoch 35. Unlike CNN-1, CNN-2 employed a combination of architectural and regularization that helped prevent overfitting and increase stability of the model. Specifically, the use of Batch Normalization throughout the convolutional blocks combatted exploding and vanishing gradient problems, while L2 kernel regularization penalized excessively large weights, further promoting generalization. Furthermore, GlobalAveragePooling replaced flatten, reducing the parameter count by a factor of nearly 3.8 while preserving feature strength, which enabled a compact yet effective representation of spatial features. Over the 35 epochs the learning rate decay schedule helped the model converge more gradually resulting in finer adjustments that continued to increase the validation accuracy.

CNN-3 showed a noticeably smaller gap between training and validation performance compared to the other models. The validation accuracy increased to 0.63 after the third epoch and finally peaked at 0.83. Training accuracy continued to increase up to 0.86. This indicates relatively stable generalization, though absolute performance remained lower than CNN-2. Beyond epoch 22, validation loss plateaued, indicating limited improvement. Although CNN-3 used DSCs to drastically reduce parameter count by about 80% compared to CNN-2, the reduced representational capacity likely limited its ability to distinguish across the nine waste classes showing less overfitting but also constrained performance.

The CNN-2 + SVM-based classification on extracted features further improved the test weighted F1-score of the plain CNN-2 by 0.02. While both models use the same convolutional feature extractor, CNN-2 applies a softmax layer that optimizes for probabilistic confidence, whereas the SVM constructs decision boundaries in the feature space. This enables the SVM to better separate overlapping or underrepresented classes, particularly in edge cases, leading to more robust generalization and sharper classification performance (Xue et al., 2016).

The softmax probability overview gives us an insight of how confident the CNN models were in their prediction. CNN-2 exhibited the highest probability, with approximately 60% of predictions yielding a softmax probability above 0.95, and 69% above 0.85 false classifications remained stable across all prediction probability intervals. CNN-1 assigned high softmax probabilities very similar to CNN-3, at

around 45% exceeding a softmax probability of 0.85 compared. However, its overall accuracy was lower, indicating overconfidence or poor calibration.

Probability intervals could not be calculated for the CNN-2 + SVM model, as the SVM does not produce softmax probabilities.

When examining the confusion matrix of the worst performing model, CNN-1, structural misclassifications are evident. The most prominent confusion occurred between the metal and plastic classes, with 18 of plastic misclassified as metal and 13 instances of metal misclassified as plastic. Additional frequent misclassifications included plastic predicted as miscellaneous trash, cardboard, and food organics (ten, six, and five instances, respectively), and paper misclassified as metal (ten instances). Textile waste and miscellaneous trash were misclassified across the majority of all categories, leading to particularly low recall scores for these underrepresented classes. While similar misclassification patterns were observed across the other models, their frequency and severity diminished as overall performance improved. For the CNN-2 + SVM model, the most persistent misclassification was the bidirectional confusion between metal and plastic. Also, miscellaneous was still confused across multiple categories but less often in any other model.

5.2 Complexity & Run Time Analysis

Metric \ Model	**CNN-1**	**CNN-2**	**CNN-3**	**CNN-2 + SVM**
Training Time (s)	65	247	258	319
# of Trained Epochs*	11	36	38	30 + SVM training
# of Trainable Params	1,737,129	462,697	82,029	462,697
# Convolutional Layers	4	5	9	5
Image size in pixels	200×200	256×256	256×256	256×256

Table 2: Model Complexity

While evaluation metrics reflect model performance, they do not capture the computational cost of achieving it. To assess overall efficiency, we consider runtime, architectural design, and convergence speed. All models were trained on a Google Colab A100 GPU with 40 GB of VRAM.

CNN-1 converged fastest, completing training in 65 seconds over 11 epochs (Table 2). Despite having only four convolutional layers, it had the highest parameter count (1.74M), mainly due to its dense head with 12,800 flattened inputs. The combination of high parameter density, shallow feature extraction, and lack of normalization led to rapid overfitting. This led to low training time, but also limiting generalization.

CNN-2 took 247 seconds and 35 epochs to train despite having only five convolutional layers. Its moderate parameter count (463k) was balanced by one more convolution and a Global Average Pooling head.

The CNN-2 + SVM variant introduced only 72 seconds of additional post-training time to fit the SVM classifier. Given that the convolutional backbone was reused without retraining, this marginal increase in complexity represents an efficient extension with high return on performance.

CNN-3 required 258 seconds and 38 epochs. Although it had the deepest architecture (nine convolutional layers), it maintained the lowest trainable parameter count (82 thousand) due to the use of DSCs. While computationally lightweight, this deep yet slim design may have lacked sufficient representational power for a 9-class task. Larger input resolution (256×256) used in both CNN-2 and CNN-3 architectures also contributed to longer per-epoch times compared to CNN-1.

Overall, training efficiency was not solely reliant on depth or parameter count. CNN-1 appeared computationally efficient but lacked robustness. CNN-3 demonstrated architectural efficiency but underperformed due to limited capacity. CNN-2, although slower, achieved stronger generalization, and its SVM extension offered the best trade-off between runtime, complexity, and performance.

6. Discussion

The results show that custom architectures achieve performance on par with pretrained models when developed iteratively from a baseline model. To analyze the differences of the model architectures, limitations and implications are discussed in the following section.

6.1 Performance Reflections & Implications

Despite its simplicity, CNN-1 demonstrated that a shallow model with minimal tuning can achieve a test F1-score of 0.70 while requiring only a fraction of the development time and computational effort. As such, CNN-1 provides a baseline that balances resource expenditure with reasonable predictive accuracy.

Architectural refinements in CNN-2 allowed it to better capture the dataset's complexity and generalize more effectively, achieving the best F1-score of our plain CNN-s at 0.86. Training patterns in both CNN-1 and CNN-2 suggest that earlier stopping could have reduced unnecessary compute.

CNN-3 improves efficiency by reducing the parameter count by about 80% compared to CNN-2. However, this comes at the cost of a performance drop compared to CNN-2. For the RealWaste dataset, the trade-off between efficiency and accuracy proves steep at 0.06 points of F1-score. More effective paths, such as pruning, could be explored in future work.

Lastly, introducing a hybrid solution between a CNN feature extractor and a Support Vector Machine (SVM) classifier revealed the value of decoupling representation learning from classification. By replacing the softmax head with an SVM trained on extracted features, the model gained the ability to draw sharper, non-probabilistic boundaries in high-dimensional space particularly beneficial in edge cases with low inter-class separability increasing F1-score by another 0.02 to 0.88. A caveat to this is that we lose the information of the softmax probability.

The findings indicate that custom-designed CNNs, when appropriately tuned and regularized, perform remarkably well on the RealWaste dataset. This supports the case for domain-specific modeling, where architecture and training strategy are optimized around task constraints and data characteristics rather than relying solely on transfer learning.

The performance of the CNN-2 + SVM model is also comparable to the top-performing models presented by Single et al. (2023) on the same dataset (see Table 4 in their paper). The training of CNN-2 follows similar patterns to the high-performing models from Single et al. The training accuracy converges quickly, while validation accuracy tracks the progression of the training with slight overfitting. Our custom CNN with the addition of a SVM classifier can almost match the performance of pretrained Image-Net baselines, with CNN-2 + SVM achieving 0.88 F1-score compared to the 0.89 from Inception V3. These findings affirm both research questions, that custom CNNs are not only effective on the RealWaste dataset, but can also achieve performance on par with leading pretrained models.

6.2 Error Analysis

Figure 3 shows representative errors across the nine waste categories from CNN-2. The misclassified notebook was understandably placed in the paper class instead of miscellaneous trash. Both the vegetation images could easily be swapped for food organics.

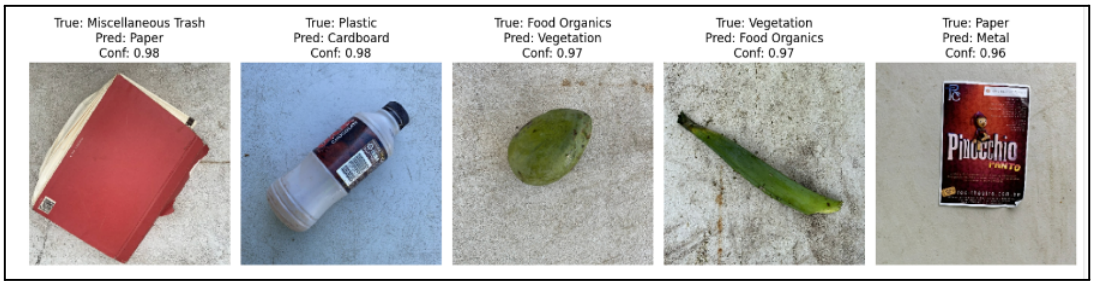

Figure 3: Representative Errors

These confusions point to not only the model's limitations but also to inconsistent labelling in the source data. As mentioned in 4.1, miscellaneous trash can be ambiguous in its interpretation. Low recall can be observed for the miscellaneous trash class across the CNN models. However, using CNN-2's features, the SVM classifier showed greater performance on the class. This could be

attributed to the SVM's ability to learn a maximum margin boundary around an outlier class, effectively isolating heterogeneous examples. In future iterations of this classification, a review of the classification should improve consistency. Further, class imbalance exists in the dataset, influencing the classification. However, the performance of the underrepresented textile trash class indicates that class imbalance was not a leading factor in misclassifications for this dataset.

6.3 Ethical concerns

The use cases for the classification models are centred around sustainability goals, such as landfill management or recycling promotion. However, the use of deep CNNs for these sustainability goals is somewhat paradoxical, as energy consumption increases with more complex models. This underlines the necessity for continued research into lightweight models for training.

There is also an absence of explainable artificial intelligence tools used in this analysis. Policymakers in waste management will require clarity around decisions made by these models, especially given the negative environmental effects of misclassification. Bias in models toward specific classes could negatively affect sustainability goals. For example, the lower recall in the textile trash category may in practice result in more environmental harm if it results in contaminated textile fibres entering recycling streams. In practice, some misclassifications carry higher stakes than others. For example, plastics misclassified as glass can damage sorting equipment. A cost-sensitive evaluation or class-weighted loss may better align model training with sustainability goals. To address these issues, future work could integrate tools like Grad-CAM to visualize model attention, identify correlations, and build a misclassification cost matrix that guides both annotation and model optimization toward sustainability goals.

7. Conclusion & Future Work

This study set out to evaluate whether custom-designed CNN models, trained exclusively on authentic waste imagery, could rival the performance of pretrained architectures in a real-world classification task. The findings confirm this hypothesis: with thoughtful architectural adjustments and hybridization, even lightweight models can deliver competitive accuracy. However, performance gains remain tightly coupled to design choices, and complexity does not guarantee superiority. Moreover, limitations such as class imbalance, ambiguous labeling, and the sustainability paradox of computational cost underscore the need for more nuanced evaluation criteria.

To build on these findings, future work should focus on testing the generalizability of these models across more diverse and realistic scenarios.

First, expanding the dataset to include more noise, variation in lighting, object positioning, and background variation would provide a more challenging but industrially relevant benchmark. This

would test the robustness of lightweight models in less controlled environments and be closer to real-world deployment conditions, such as in smart waste-sorting systems.

Second, introducing data from multiple sources or geographic regions could help assess the adaptability of the classifiers across different waste classification standards or cultural patterns of waste generation.

Moreover, exploring additional hybrid architectures such as CNNs paired with other classical ML techniques beyond SVM i.e., Random Forest or KNN could yield further performance gains without significantly increasing computational cost.

In summary, while the current results are promising, truly scalable impact lies in validating lightweight models under real-world constraints. The next step is to stress-test these architectures under higher dataset complexity to unlock their full potential for robust and efficient deployment.

References

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. IEEE Xplore, 1800–1807. https://doi.org/10.1109/cvpr.2017.195

Malik, M., Sharma, S., Uddin, M., Chen, C.-L., Wu, C.-M., Soni, P., & Chaudhary, S. (2022). Waste classification for sustainable development using image recognition with deep learning neural network models. *Sustainability, 14*(12), 7222. https://doi.org/10.3390/su14127222

Kumsetty, N., Nekkare, A., (2022). TrashBox. IEEE Dataport. https://dx.doi.org/10.21227/csg6-h017

Sabottke, C. F., & Spieler, B. M. (2020). The effect of image resolution on deep learning in radiography. Radiology: Artificial Intelligence, 2(1), e190015. https://doi.org/10.1148/ryai.2019190015

Single, S., Iranmanesh, S., & Raad, R. (2023). RealWaste: A novel real-life data set for landfill waste classification using deep learning. *Information, 14*(12), 633. https://doi.org/10.3390/info14120633

Single, S., Iranmanesh, S., & Raad, R. (2023). RealWaste [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5SS4G.

United Nations Environment Programme, & Climate and Clean Air Coalition. (2021). Global methane assessment: Benefits and costs of mitigating methane emissions. United Nations Environment Programme. https://www.unep.org/resources/report/global-methane-assessment-benefits-and-costs-mitigating-methane-emissions

United Nations Environment Programme. (2024). Global waste management outlook 2024: Beyond an age of waste – Turning rubbish into a resource. United Nations Environment Programme. https://doi.org/10.59117/20.500.11822/44939

Xue, D.-X., Zhang, R., Feng, H., & Wang, Y.-L. (2016). CNN-SVM for microvascular morphological type recognition with data augmentation. Journal of Medical and Biological Engineering, 36(6), 755–764. https://doi.org/10.1007/s40846-016-0182-4

Appendix

Figure A1: Plastic Dinosaur Toy Classified as Miscellaneous Trash

Figure A2: Miscellaneous Trash

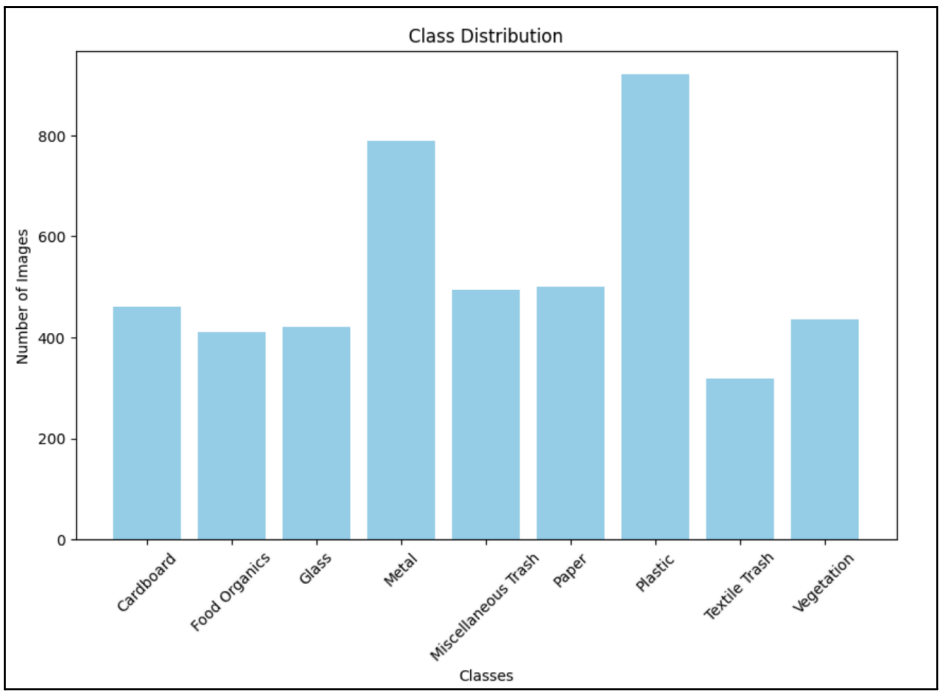

Figure A3: Class Distribution

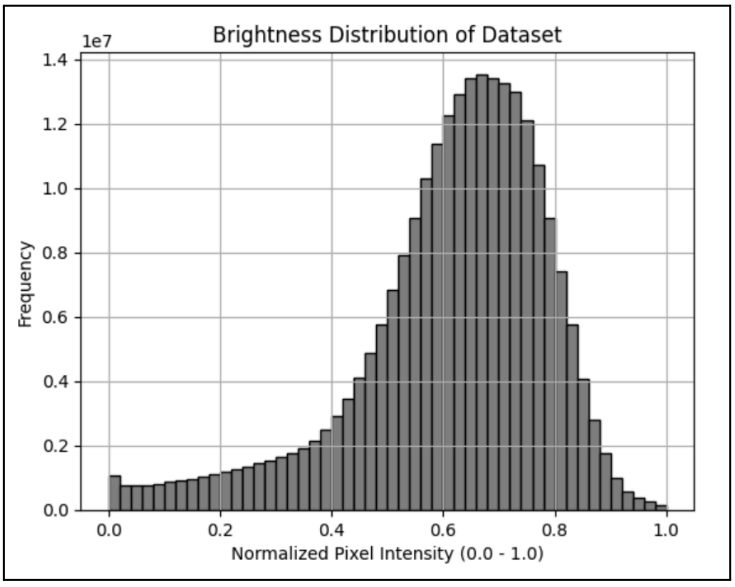

Figure A4: Pixel Distribution

Type	Description	Range
Flip	Horizontally flips the image	N/A
Rotate	Rotates the image by a random angle	-30° to +30°
Brightness	Randomly adjusts image brightness	-0.3 to +0.3 delta
Crop	Centrally crops and resizes image	80% of center
Noise	Adds random Gaussian noise	Stdev = 15.0

Table A1: Augmentation Operations

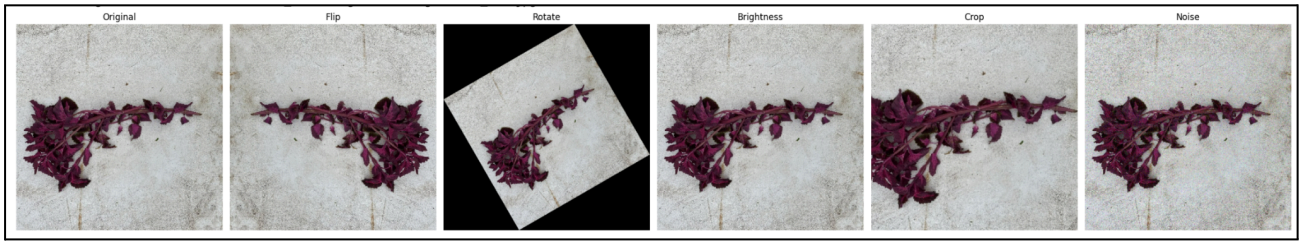

Figure A5: Augmentation Types

```
Found 19938 files belonging to 9 classes.
Found 710 files belonging to 9 classes.
Found 719 files belonging to 9 classes.
```

Figure A6: Train/Validation/Test Data After Augmentation

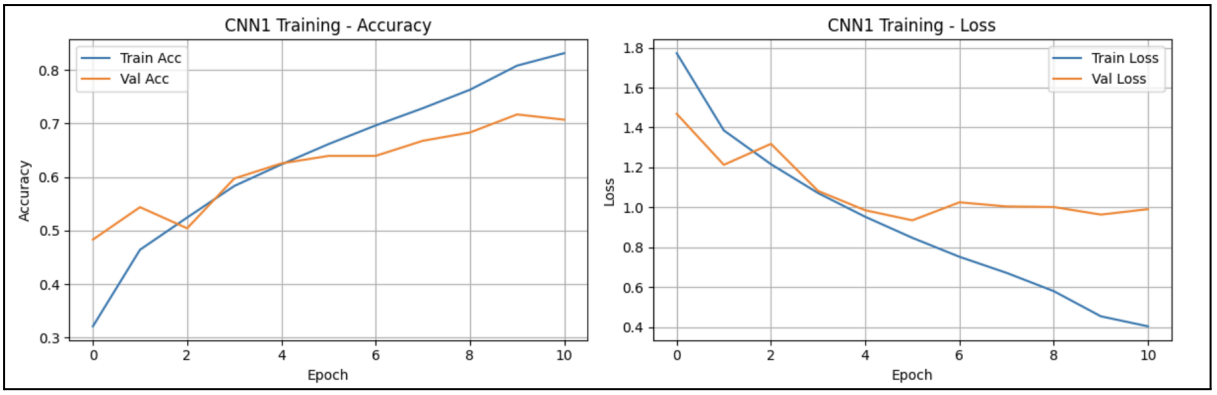

Figure A7: CNN-1 Training Accuracy and Loss

Test Classification Report:

	precision	recall	f1-score	support
Cardboard	0.70	0.64	0.67	70
Food Organics	0.79	0.90	0.84	63
Glass	0.83	0.71	0.77	63
Metal	0.67	0.78	0.72	119
Miscellaneous Trash	0.51	0.60	0.55	75
Paper	0.71	0.63	0.67	75
Plastic	0.71	0.64	0.67	139
Textile Trash	0.60	0.53	0.57	49
Vegetation	0.86	0.89	0.87	66
accuracy			0.70	719
macro avg	0.71	0.70	0.70	719
weighted avg	0.71	0.70	0.70	719

Figure A8: CNN-1 Classification Report

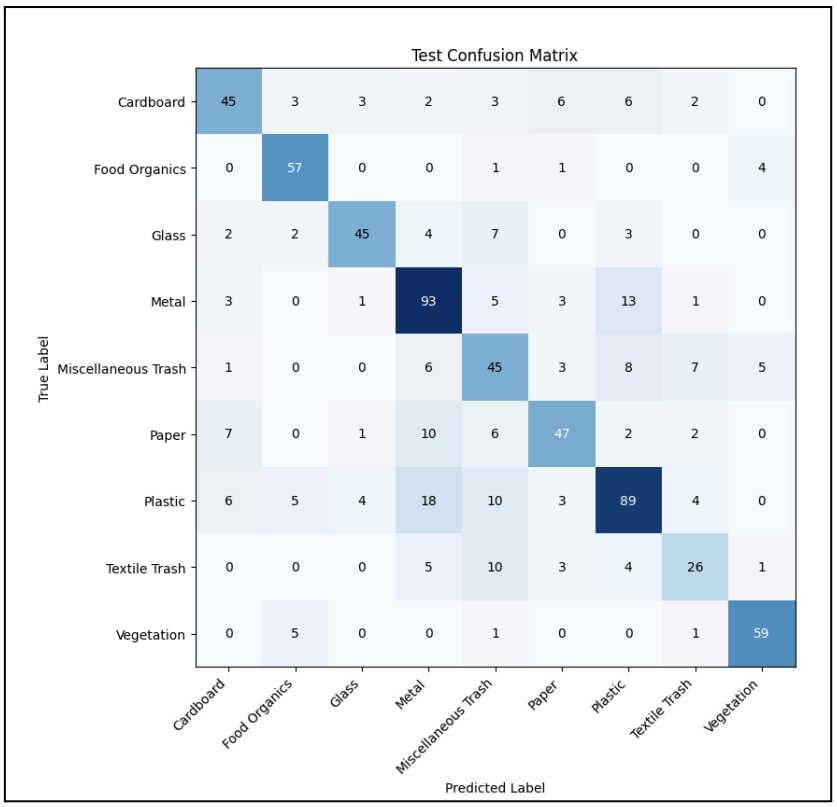

Figure A9: CNN-1 Confusion Matrix

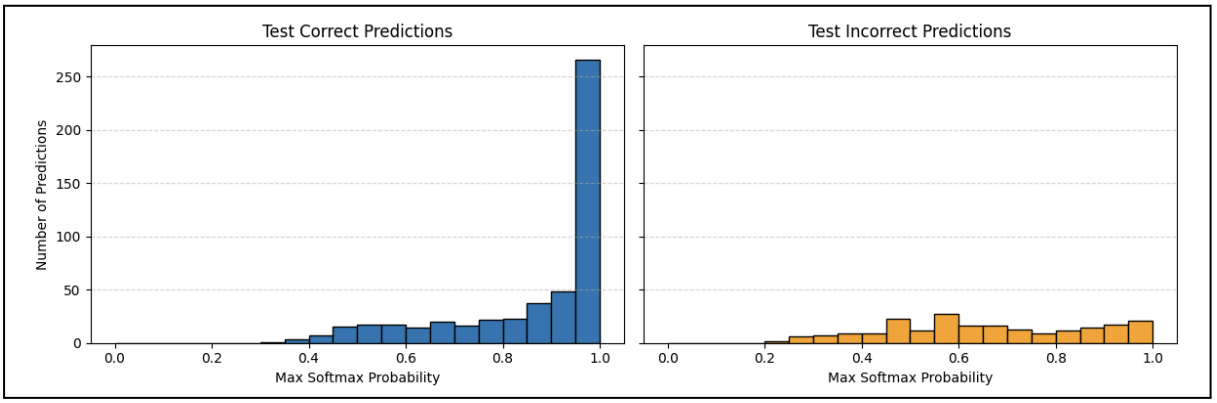

Figure A10: CNN-1 Softmax Probabilities

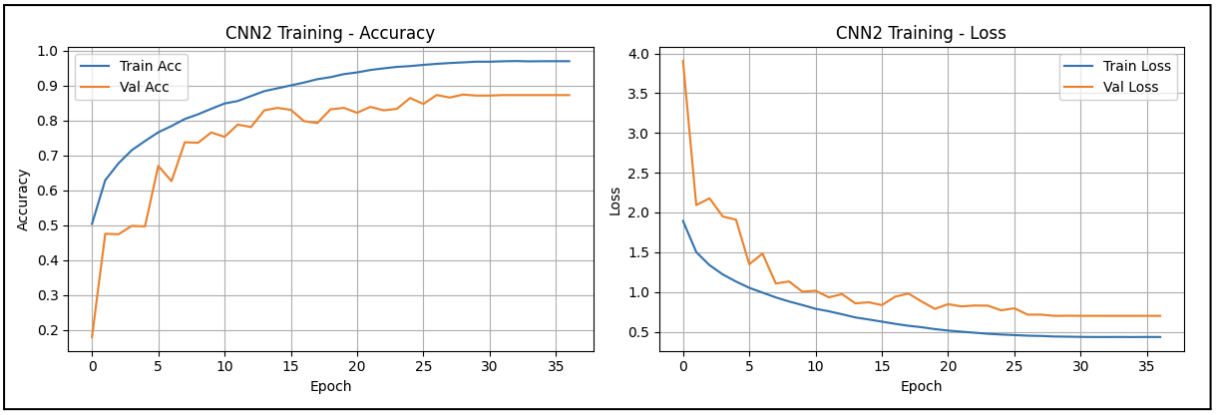

Figure A11: CNN-2 Training Accuracy and Loss

Test Classification Report:

	precision	recall	f1-score	support
Cardboard	0.91	0.91	0.91	70
Food Organics	0.86	0.94	0.89	63
Glass	0.93	0.90	0.92	63
Metal	0.84	0.87	0.85	119
Miscellaneous Trash	0.71	0.68	0.69	75
Paper	0.93	0.87	0.90	75
Plastic	0.85	0.86	0.85	139
Textile Trash	0.80	0.82	0.81	49
Vegetation	0.94	0.92	0.93	66
accuracy			0.86	719
macro avg	0.86	0.86	0.86	719
weighted avg	0.86	0.86	0.86	719

Figure A12: CNN-2 Classification Report

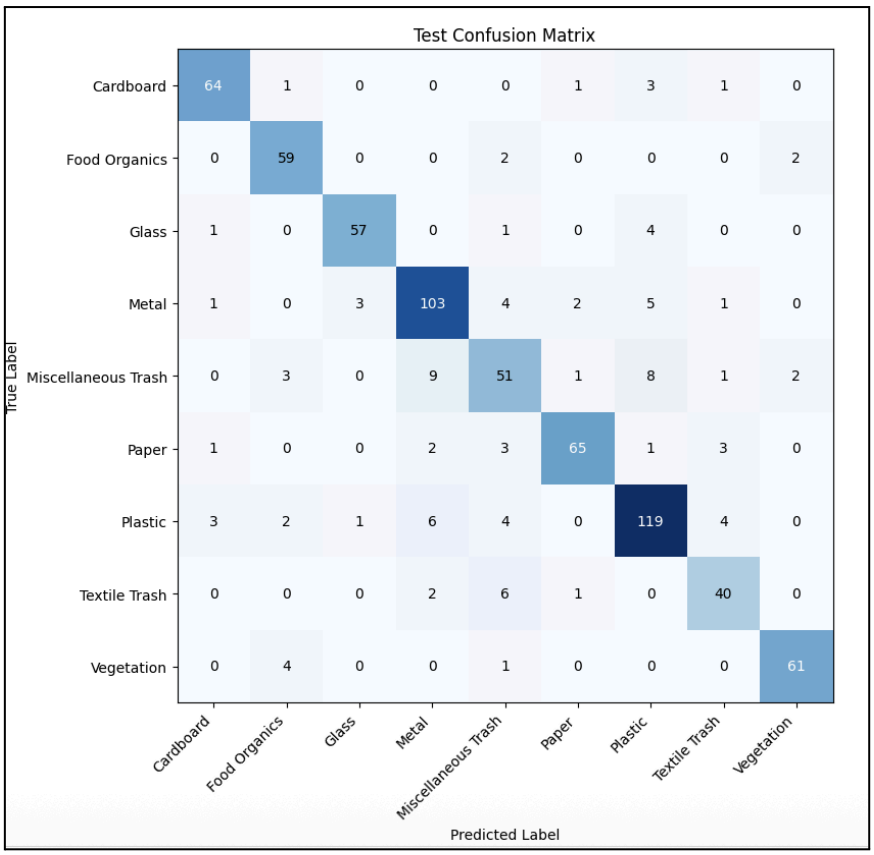

Figure A13: CNN-2 Confusion Matrix

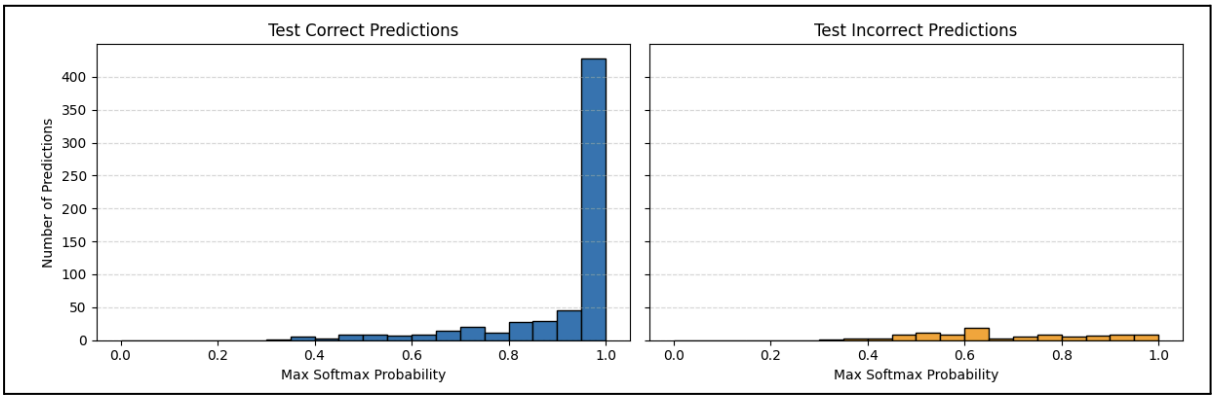

Figure A14: CNN-2 Softmax Probabilities

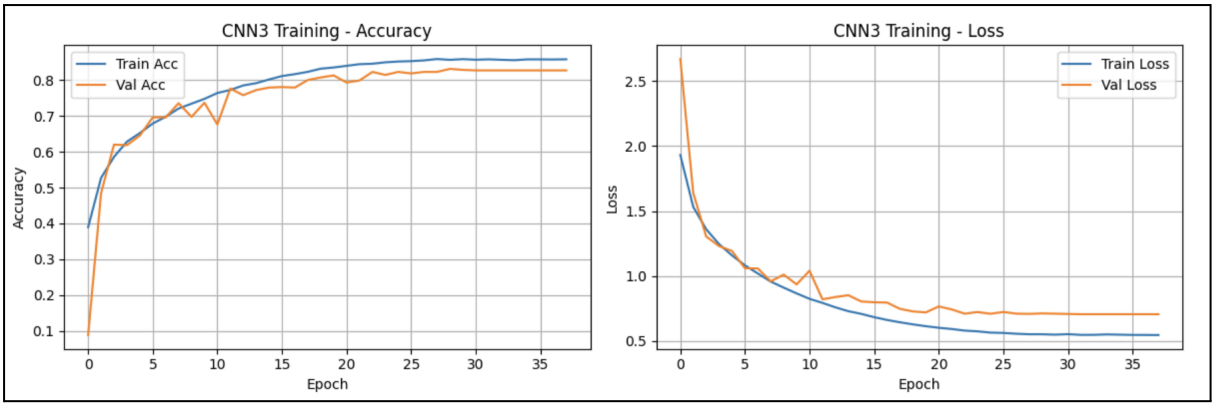

Figure A15: CNN-3 Training Accuracy and Loss

Test CNN3 Classification Report:

	precision	recall	f1-score	support
Cardboard	0.80	0.81	0.81	70
Food Organics	0.89	0.87	0.88	63
Glass	0.82	0.86	0.84	63
Metal	0.80	0.83	0.82	119
Miscellaneous Trash	0.71	0.60	0.65	75
Paper	0.87	0.79	0.83	75
Plastic	0.77	0.77	0.77	139
Textile Trash	0.72	0.78	0.75	49
Vegetation	0.85	0.95	0.90	66
accuracy			0.80	719
macro avg	0.80	0.81	0.80	719
weighted avg	0.80	0.80	0.80	719

Figure A16: CNN-3 Classification Report

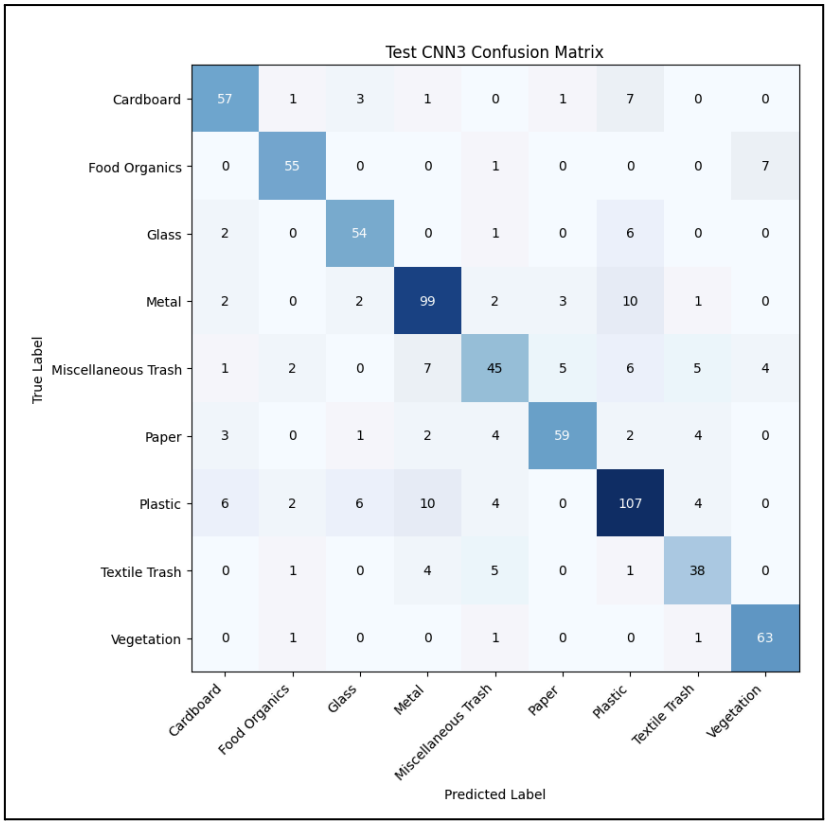

Figure A17: CNN-3 Confusion Matrix

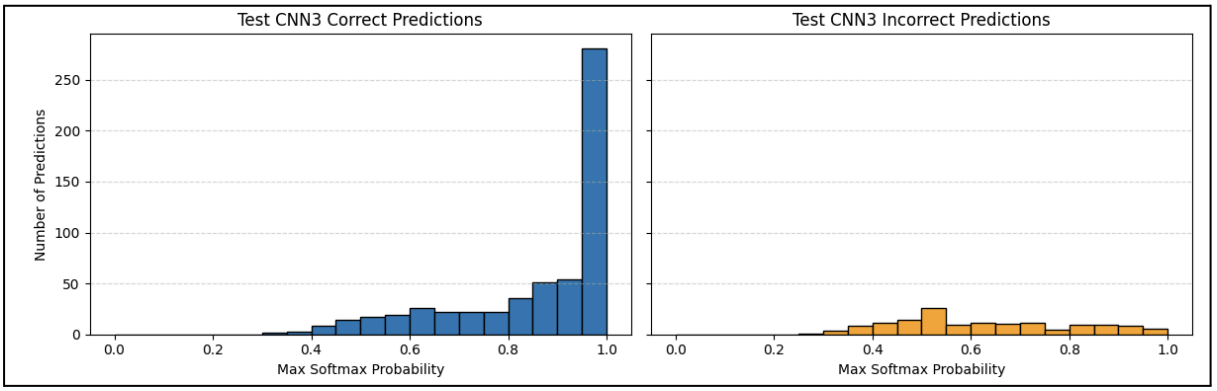

Figure A18: CNN-3 Softmax Probabilities

SVM2 test Classification Report:

	precision	recall	f1-score	support
Cardboard	0.93	0.90	0.91	70
Food Organics	0.92	0.95	0.94	63
Glass	0.98	0.87	0.92	63
Metal	0.87	0.87	0.87	119
Miscellaneous Trash	0.75	0.73	0.74	75
Paper	0.91	0.91	0.91	75
Plastic	0.83	0.89	0.86	139
Textile Trash	0.88	0.88	0.88	49
Vegetation	0.95	0.94	0.95	66
accuracy			0.88	719
macro avg	0.89	0.88	0.89	719
weighted avg	0.88	0.88	0.88	719

s

Figure A19: CNN-2 + SVM Classification Report

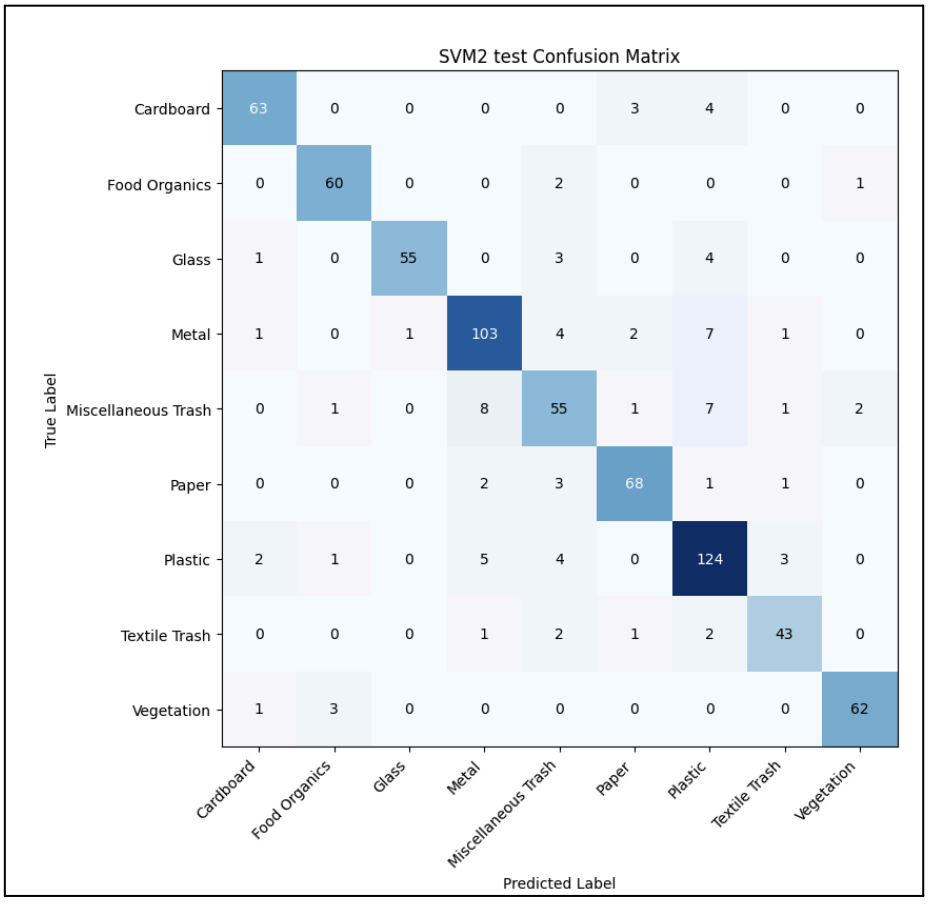

Figure A20: CNN-2 + SVM Confusion Matrix

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 200, 200, 3)	0
conv2d (Conv2D)	(None, 198, 198, 16)	448
max_pooling2d (MaxPooling2D)	(None, 99, 99, 16)	0
conv2d_1 (Conv2D)	(None, 97, 97, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 64)	0
conv2d_3 (Conv2D)	(None, 21, 21, 128)	73,856
max_pooling2d_3 (MaxPooling2D)	(None, 10, 10, 128)	0
flatten (Flatten)	(None, 12800)	0
dense (Dense)	(None, 128)	1,638,528
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 9)	1,161

Total params: 1,737,129 (6.63 MB)
Trainable params: 1,737,129 (6.63 MB)
Non-trainable params: 0 (0.00 B)

Figure A21: CNN-1 Architecture

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 256, 256, 3)	0
conv2d_5 (Conv2D)	(None, 256, 256, 16)	448
batch_normalization_5 (BatchNormalization)	(None, 256, 256, 16)	64
activation_5 (Activation)	(None, 256, 256, 16)	0
max_pooling2d_5 (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_6 (Conv2D)	(None, 128, 128, 32)	4,640
batch_normalization_6 (BatchNormalization)	(None, 128, 128, 32)	128
activation_6 (Activation)	(None, 128, 128, 32)	0
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_7 (Conv2D)	(None, 64, 64, 64)	18,496
batch_normalization_7 (BatchNormalization)	(None, 64, 64, 64)	256
activation_7 (Activation)	(None, 64, 64, 64)	0
max_pooling2d_7 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_8 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_8 (BatchNormalization)	(None, 32, 32, 128)	512
activation_8 (Activation)	(None, 32, 32, 128)	0
max_pooling2d_8 (MaxPooling2D)	(None, 16, 16, 128)	0
conv2d_9 (Conv2D)	(None, 16, 16, 256)	295,168
batch_normalization_9 (BatchNormalization)	(None, 16, 16, 256)	1,024
activation_9 (Activation)	(None, 16, 16, 256)	0
max_pooling2d_9 (MaxPooling2D)	(None, 8, 8, 256)	0
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65,792
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 9)	2,313

Total params: 462,697 (1.77 MB)
Trainable params: 461,705 (1.76 MB)
Non-trainable params: 992 (3.88 KB)

Figure A22: CNN-2 Architecture

Model: "sequential_3"

Layer (type)	Output Shape	Param #
rescaling_3 (Rescaling)	(None, 256, 256, 3)	0
conv2d_16 (Conv2D)	(None, 256, 256, 16)	448
batch_normalization_29 (BatchNormalization)	(None, 256, 256, 16)	64
activation_29 (Activation)	(None, 256, 256, 16)	0
max_pooling2d_16 (MaxPooling2D)	(None, 128, 128, 16)	0
depthwise_conv2d_13 (DepthwiseConv2D)	(None, 128, 128, 16)	160
batch_normalization_30 (BatchNormalization)	(None, 128, 128, 16)	64
activation_30 (Activation)	(None, 128, 128, 16)	0
conv2d_17 (Conv2D)	(None, 128, 128, 32)	544
batch_normalization_31 (BatchNormalization)	(None, 128, 128, 32)	128
activation_31 (Activation)	(None, 128, 128, 32)	0
max_pooling2d_17 (MaxPooling2D)	(None, 64, 64, 32)	0
depthwise_conv2d_14 (DepthwiseConv2D)	(None, 64, 64, 32)	320
batch_normalization_32 (BatchNormalization)	(None, 64, 64, 32)	128
activation_32 (Activation)	(None, 64, 64, 32)	0
conv2d_18 (Conv2D)	(None, 64, 64, 64)	2,112
batch_normalization_33 (BatchNormalization)	(None, 64, 64, 64)	256
activation_33 (Activation)	(None, 64, 64, 64)	0
max_pooling2d_18 (MaxPooling2D)	(None, 32, 32, 64)	0
depthwise_conv2d_15 (DepthwiseConv2D)	(None, 32, 32, 64)	640
batch_normalization_34 (BatchNormalization)	(None, 32, 32, 64)	256
activation_34 (Activation)	(None, 32, 32, 64)	0
conv2d_19 (Conv2D)	(None, 32, 32, 128)	8,320
batch_normalization_35 (BatchNormalization)	(None, 32, 32, 128)	512
activation_35 (Activation)	(None, 32, 32, 128)	0
max_pooling2d_19 (MaxPooling2D)	(None, 16, 16, 128)	0
depthwise_conv2d_16 (DepthwiseConv2D)	(None, 16, 16, 128)	1,280
batch_normalization_36 (BatchNormalization)	(None, 16, 16, 128)	512
activation_36 (Activation)	(None, 16, 16, 128)	0
conv2d_20 (Conv2D)	(None, 16, 16, 256)	33,024
batch_normalization_37 (BatchNormalization)	(None, 16, 16, 256)	1,024
activation_37 (Activation)	(None, 16, 16, 256)	0
max_pooling2d_20 (MaxPooling2D)	(None, 8, 8, 256)	0
global_average_pooling2d_3 (GlobalAveragePooling2D)	(None, 256)	0
dense_6 (Dense)	(None, 128)	32,896
dropout_3 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 9)	1,161

Total params: 83,849 (327.54 KB)
Trainable params: 82,377 (321.79 KB)
Non-trainable params: 1,472 (5.75 KB)

Figure A23: CNN-3 Architecture