ECOSORT: GARBAGE CLASSIFICATION

Hassan Qayyum
Student# 1008063836
hassan.qayyum@mail.utoronto.ca

Briana Vieira
Student# 1007790437
briana.vieira@mail.utoronto.ca

Areeba Mobeen
Student# 1007096786
areeba.mobeen@mail.utoronto.ca

Rayan Mustafa
Student# 1008224191
rayan.mustafa@mail.utoronto.ca

ABSTRACT

The following provides an overview of the advancements made in the EcoSort project, aimed at developing an AI model for classifying garbage classes to facilitate efficient waste disposal. It highlights the progress achieved thus far, focusing on key aspects such as data processing and cleaning as well as current best models. We discuss the baseline model established to set a benchmark for the project's performance and elaborate on the current performance of the primary model, which leverages transfer learning to enhance classification accuracy.

—-Total Pages: 8

1 INTRODUCTION

The average person will spend one-third of their lifetime at work and at the same time generate 0.74kg of daily waste, contributing to roughly 2 billion tonnes globally every year (Li & Chen, 2023). Workplaces are challenged with managing a significant amount of waste, our team poses the question of how successfully this is being done. If executed correctly, waste sorting enables the recovery and recycling of valuable materials like metals, paper, and plastics, which conserves natural resources, reduces energy consumption, and lowers greenhouse gas emissions.

Currently, these systems are entirely manual and rely on the consumer to correctly sort their waste, and are consequently error-prone. As is demonstrated by the millions of tons of garbage that is not sorted correctly and end up in improper disposal sites, contributing to environmental degradation and climate change each year (Deer, 2021). In Toronto alone, the blue bin program manages 180,000 tonnes of recyclables annually, yet 30 percent (54,000 tonnes) is sent to landfills due to contamination (Zettler, 2019). This fully manual method has plenty of room for improvement and hinders progress toward efficient and accurate waste classification by not employing data-driven approaches.

Deep learning, particularly CNNs, offers a promising approach due to its ability to automatically learn discriminative features from raw data, making it well-suited for image-based tasks like garbage classification. Other machine learning approaches like RNNs, SVMs, and Random Forests are less suitable for image-based tasks such as garbage classification. RNNs are designed for sequential data tasks, while SVMs and Random Forests rely on handcrafted features or predefined rules, which may not capture the diverse visual characteristics of waste materials as effectively. Overall, CNNs offer a powerful and efficient solution for garbage classification, leveraging the capabilities of deep learning to automatically learn and extract discriminative features from raw image data.

Employing this CNN model approach, our project, EcoSort, addresses the pressing issue of improper waste management by developing a robust garbage classification system that will facilitate proper disposal efforts. We plan for our model to classify waste into the following six classes: cardboard, glass, metal, paper, plastic, and trash, of which the outputs can then be grouped into umbrella categories as needed. By leveraging the power of deep learning, we aim to enhance waste management practices, mitigate environmental pollution, and contribute to the fight against climate change.

2 Data Processing

2.1 Data Collection

To collect data, our team used 3 large datasets from Kaggle with images of garbage correctly sorted within their class. One of the datasets is correctly aligned with our intended classes for the model and the other dataset has additional classes.

2.2 Preliminary Data Cleaning

To ensure our data effectively trained our model, we undertook a rigorous cleaning process to remove duplicates and outliers. Each team member was responsible for cleaning an assigned class, and meticulously checking images one by one. Duplicate images, identified by identical orientation, size, and padding, were eliminated. While many images initially appeared to be duplicates due to file name similarities, each dataset was refined to contain only one copy of each unique image.

Our team also identified and removed outlier—images that did not fit any garbage class or were excessively blurry, dark, or unclear. This thorough cleaning process concluded our Preliminary Data Cleaning phase, ensuring the quality and integrity of our dataset for training.

2.3 Combing the Datasets

Once the preliminary dataset and image folders had been cleaned the next step was to import the images into a dataset (ImageFolder) and create data loaders for training, validation, and testing. To combine the 3 datasets into 1 dataset, we created a Python script to copy the images for the intended class from each dataset into corresponding class folders in a single dataset. The Python script got all the file names of each image within each dataset and then checked the label/class of the image within the name created the corresponding directory within the final dataset and copied the image. Once all the images were imported into a single folder, it was simple to create an ImageFolder from that directory and determined the number of images in each class, as seen in Figure 1. We realized the dataset was imbalanced with a smaller number of samples for the trash class than the other classes. To mitigate these issues we applied extra data augmentation, specifically for the trash class, and attempted to double the number of images.

2.4 Data Augmentation

Determining the number of images in each class, as shown in Figure 1, highlighted that the dataset was imbalanced with fewer samples in the `trash` class compared to the others. The augmentations applied were randomly selected from one of three: rotate by 30 degrees, resize, and flip the image horizontally. This allowed us to create a much more balanced dataset as highlighted by Figure 2. We proceeded to add more augmentations to the data such as color distorting, cutout, rotation, and noise blur to prevent overfitting as well as expand our dataset size. The final dataset composition can be seen in Figure 2.

Class	Number of Examples	
Cardboard	4038	17.01%
Glass	5099	21.48%
Paper	3458	14.57%
Plastic	4965	20.92%
Metal	4371	18.42%
Trash	1805	7.60%
Total	**23736**	**100.00%**

Figure 1: Original imbalanced dataset

Class	Number of Examples	
Cardboard	3945	16.62%
Glass	3719	15.67%
Paper	4054	17.08%
Plastic	3923	16.53%
Metal	3889	16.38%
Trash	4206	17.72%
Total	**23736**	**100.00%**

Figure 2: Balanced dataset after augmentations

2.5 Splitting the Data

After creating the image folder, splitting the dataset into training, validation, and testing sets was simple. We utilized the `train_test_split` function from the `sklearn` library to perform this task. This function is essential for maintaining dataset integrity, as it preserves the distribution of classes across each dataset, ensuring the model training is as unbiased as possible. We chose a split

ratio of 70%, 15%, and 15% for training, validation, and testing datasets, respectively. To allow sufficient time for model training, we proceeded with the imbalanced dataset as the final dataset was not yet ready.

2.6 Additional Data for testing

We currently have around 5500 images in our testing dataset. However, we believe that additional images from outside our original dataset may be necessary for a comprehensive evaluation of our model. Kaggle offers a variety of garbage classification datasets that could provide valuable additional testing data. Once the training phase of our model is complete, we plan to select a suitable dataset from Kaggle to perform a final evaluation of our model.

3 Baseline Model

EcoSort aims to classify images into six classes of garbage: cardboard, glass, metal, paper, plastic, and trash. Therefore, a Convolutional Neural Network (CNN) is the most suitable baseline model. CNNs excel at feature learning in classification tasks by employing mathematical constructs such as convolution filters and pooling. These techniques enable the model to discern distinct and recognizable features within smaller sections of the data.

Our team chose a 3-layer CNN as a baseline model for our image classification project. The architecture of the model is depicted in Figure 3. We chose this simple CNN as a starting point to establish a performance benchmark to which we can compare more complex models' performances. Our primary expectation from the baseline model is that it can generalize and accurately classify images in our datasets. Hence, a CNN was the perfect choice for image classification. In the 'Gesture Recognition' lab, using a similar CNN architecture led to high accuracy results. Therefore, we think this CNN is a good blueprint for classifying images for our project EcoSort, aligning with our established expectations of the model.

We used three convolutional layers, each layer further capturing more complex patterns to produce feature maps of the input images. Our chosen kernel size is 3x3 since any large kernel can be approximated with 3x3 kernels, ultimately being the most efficient choice as proven by the GoogLeNet. The first kernel is 5x5 to resize the input image as it is convenient to use a kernel of this size on the first layer. Additionally, a max pooling layer is used in between each convolutional layer to reduce the spatial dimensions of the feature maps. Finally, the convolution and pooling layers are followed by two fully connected layers for our model's purpose of image classification.

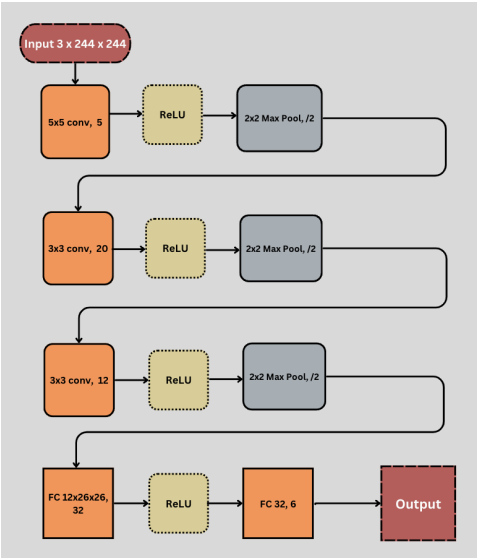

Figure 3: Baseline model

3.1 Comparing the Baseline Model with the Primary Model

To evaluate the accuracy of our primary model, we will compare it to a benchmark baseline model. Testing only the primary model's performance accuracy is insufficient, as high accuracy could result from either effective classification or the problem's simplicity. The CNN serves as an ideal baseline model, offering a straightforward and computationally efficient comparison point with minimal fine-tuning required. By comparing the primary model to this benchmark, we can assess its performance based on how much it outperforms the baseline. If the primary model does not significantly outperform the baseline, it suggests that it is underperforming and not meeting our expectations

3.2 Best Results

3.2.1 Quantitative: Validation Accuracy vs. Training Accuracy

The baseline model is a feasible benchmark model to achieve our project objectives of accurate image classification. In Figure 4, we can see that as the number of epochs increased, the model's training and validation loss steadily declined, indicating successful training of the model. Additionally, with an increase in epochs, the training and validation accuracy shows signs of improvement. The model performed well for the given time; with a validation and training accuracy of 65%, respectively. We can judge the performance of our primary model based on these results. While the baseline model did exhibit areas where the validation loss exceeded the training loss, suggesting overfitting and data memorization in those specific areas, the overall trend of both loss curves is downward. This indicates that the baseline model performs well and serves as a viable benchmark for comparing more complex models.

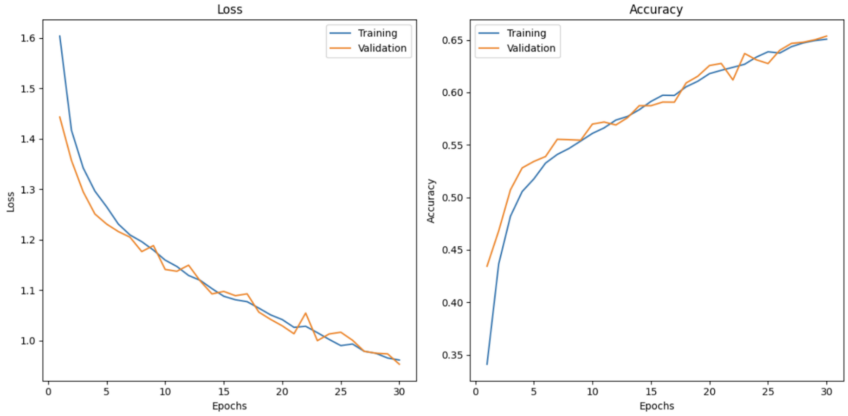

Figure 4: Training and validation loss vs. accuracy

3.2.2 Qualitative: Confusion Matrix

Following the training phase of the baseline model, we evaluated how well the model can perform on unseen images of different classes in the test data set. From the model's confusion matrix in Figure 5, we made several observations about how well the model performs with certain classes over others. A value closer to 500, signifies a higher number of correct predictions being made by the model. We noted that for every row, the cell with the most predictions accurately matches the correct category labels, indicating mostly accurate predictions for those classes. It is important to note that this is not the most reliable measure of accuracy as this was calculated using the dataset before it was balanced dataset, as shown in Figure 1. We noticed that the two classes that the model mixes up the most are glass and metal. There are approximately 7% more images of metal than glass in the test dataset. Despite this, our confusion matrix indicates that with a test set of unseen images, the model is still able to perform well and is a solid benchmark model for the project.

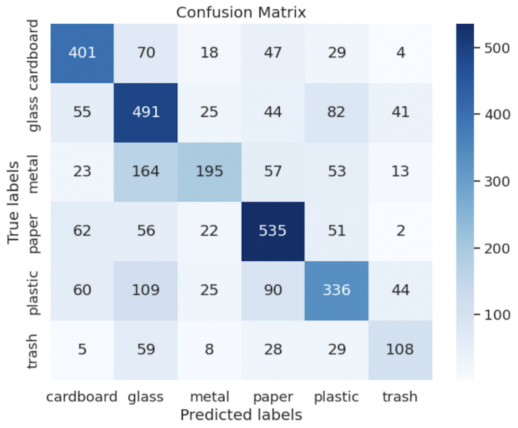

Figure 5: Confusion matrix with test dataset

3.2.3 Other Qualitative Observations

In Figure 6, we displayed several images that our baseline model classified, along with their ground truth labels and the class that the model predicted. By comparing the truth label to the predicted class, we can observe whether or not the baseline model was able to classify that image correctly. Of the 9 images displayed, 4 of them were accurately predicted by the model. Our goal moving forward is to surpass the performance of this benchmark CNN model to achieve higher accuracy results.

3.3 Challenges Faced

Training the baseline model was a time and effort-consuming task primarily due to the sheer size of the training dataset, which comprised over 16,000 images. This led to long training times and difficulty in hyperparameter tuning. Each iteration demanded hours of computational time, impeding the optimization process. Despite using CUDA for faster computations, its limited availability hindered the full utilization of resources. Furthermore, diagnosing the model's performance issues proved to be a complex and time-consuming endeavor. One of the major challenges was identifying the classes that were being confused by the model and determining whether there was a discernible pattern to these errors. This debugging process required meticulous examination and analysis, consuming a significant amount of time and effort.

Analyzing the data with a confusion matrix and printing labels for predicted images was challenging due to the large dataset. Generating and interpreting the confusion matrix required substantial computational resources and mapping the predicted labels back to their corresponding class names for printing added complexity, as misinterpretation could lead to incorrect analysis. Additionally, integrating the analysis into the existing workflow, especially in a collaborative environment like Google Colab, presented logistical challenges. Coordinating the analysis with other collaborators

Figure 6: Baseline model classifications

and ensuring that everyone had access to the necessary resources and tools required careful planning and communication.

Overall, these challenges underscore the complexities involved in training and optimizing deep learning models on large datasets, highlighting the importance of efficient debugging and optimization strategies in such endeavors.

4 Primary Model

EcoSort aims to classify images of garbage into six classes: cardboard, glass, metal, paper, plastic, and trash. To accomplish this, we plan to leverage transfer learning, utilizing a pre-trained Convolutional Neural Network (CNN). CNNs are particularly well-suited for image classification tasks due to their ability to learn distinctive features from images using convolution filters and pooling operations.

In the realm of CNNs, successful models have been developed with exceptional feature-extraction capabilities. Among these, AlexNet stands out as one of the pioneering models, featuring 5 convolution layers followed by 3 fully connected layers, totaling 8 layers and 62.8 million training parameters (Sinha, 2021). Another notable model is ResNet, designed to tackle the issue of vanishing gradients by incorporating Residual Networks and skip connections, allowing for much deeper networks. ResNet comes in several versions with varying depths, such as 18, 34, and 50 layers.

To select the most suitable model for our task, we will explore the capabilities of ResNets, AlexNet, and other models, utilizing transfer learning to train our classifier. Transfer learning is particularly efficient in this context as it leverages features learned by pre-trained models like AlexNet and ResNet. These models have been trained on extensive datasets like ImageNet, enabling them to capture generic features such as edges, textures, and shapes that are valuable for various computer vision tasks. By employing transfer learning, we can initiate training with these pre-trained models and fine-tune them on our specific dataset, achieving high performance with less computational resources and training time compared to training from scratch. Furthermore, transfer learning helps mitigate overfitting, as the pre-trained models have already learned meaningful representations from vast amounts of data.

Our classifier comprises a single convolutional layer utilizing 3x3 filters, which enables us to replicate the impact of larger resolution filters while minimizing adjustable parameters. It also includes max pooling and two fully connected layers. To date, our team has trained the classifier using high-level features extracted using AlexNet.

4.1 Best Result

After training the classifier with the AlexNet embedding and experimenting with various hyperparameters, we identified the best model to use the ReLU activation function, a learning rate of 0.001, and a batch size of 128 after 28 epochs. This optimal model consists of a total of 8 layers: 5 convolutional layers from AlexNet, 1 convolutional layer from the classifier, and 2 fully connected layers in the classifier. In addition to the trainable parameters from AlexNet, the classifier contains 5,647,078 parameters. The full model architecture is shown in Figure 7.

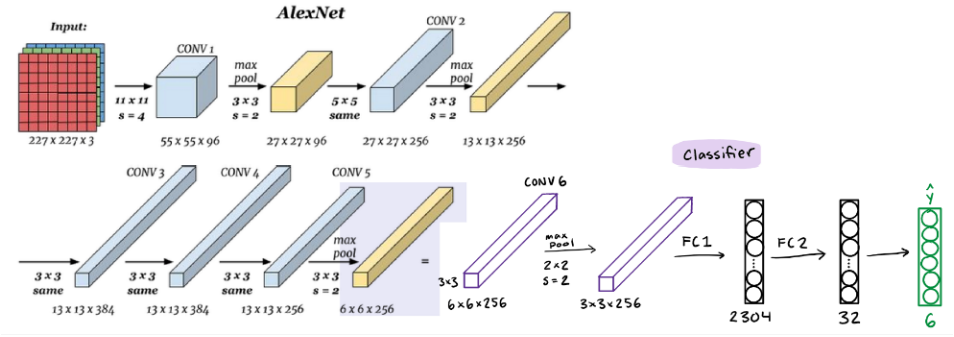

Figure 7: Full model architecture using AlexNet

This model achieves a validation accuracy of 87.192% (Figure 8), demonstrating strong performance and surpassing the baseline model by approximately 22%

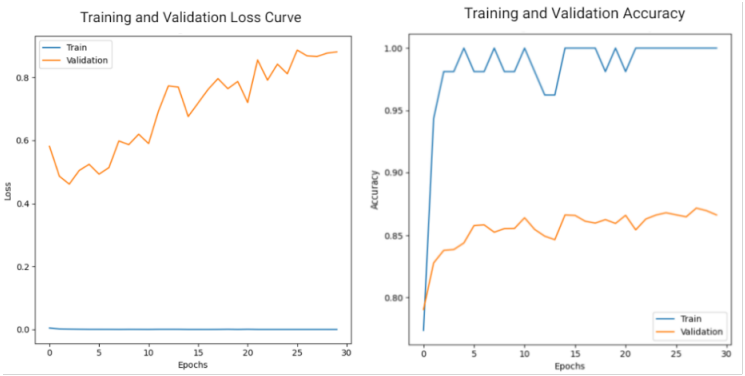

Figure 8: Loss and accuracy training curves over thirty epochs

However, our model appears to be overfitting to the training data as the training loss is zero while the validation loss has an increasing trend. Going forward we will try to combat this issue by employing a learning rate scheduler, weight decay, and dropout techniques to introduce more noise and regularize when training the model, which should result in better results on the validation data.

The confusion matrix in Figure 9 provides insights into our model's performance across classes 1 through 6 (cardboard, glass, metal, paper, plastic, trash), allowing us to compare its performance across different classes. On unseen data, our model demonstrated strong performance, with the highest accuracy on paper (92.62% of predictions matching the true label) and the lowest on trash (79.63%). However, the model struggled in some areas, such as predicting plastic as glass and glass as metal.

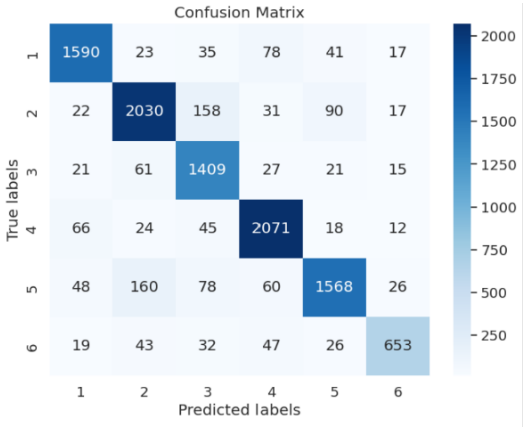

Figure 9: Confusion matrix with the test dataset

Additionally, we have yet to try using ResNets and are unaware of how they will compare to the AlexNet results. Especially with the deeper ResNets that have more layers, the model should be able to extract higher quality embeddings, which our classifier should then be able to learn more accurately and have increased score results.

4.2 Challenges

Developing our primary model posed challenges akin to those encountered with the baseline model. Training the classifier with 16,000 images was time-consuming, compounded by the additional time needed to extract embeddings from all 23,000 images using AlexNet.

Logistical challenges in Google Colab persisted, particularly regarding data preparation before training and the necessity of baseline results for comparison. Each step was executed in separate Colab files, underscoring the importance of continuous communication and task tracking throughout the process. Furthermore, rectifying the issue or model with overfitting is a challenge we will continue to try to overcome in the next stage of the project. At this point we have also struggled to successfully extract embeddings using ResNet as the method is different from the one used for AlexNet. These are challenges we will continue to navigate to put forward an even better model at the end of this project.

5 Google Colab

The following folder (link to folder) contains all the Google Colab notebooks used to make progress, including those for data processing, the baseline model, and the primary model files.

REFERENCES

Ryan Deer. Landfills: We're running out of space. *ROADRUNNER*, 2021.

Ninghui Li and Yuan Chen. Municipal solid waste classification and real-time detection using deep learning methods. *Urban Climate*, 2023.

Sajal Sinha. Alexnet architecture. *Medium*, 2021.

Melanie Zettler. Toronto recycling: Why so much material still goes to landfill. *Global News: Environment*, 2019.