WASTE SEGREGATION USING CNN MODEL

Athavan Therunavukarasu
Student# 1005363143
athavan.therunavukarasu@mail.utoronto.ca

Shadman Kaif
Student# 1005303137
shadman.kaif@mail.utoronto.ca

Abdurrafay Khan
Student# 1005215517
abdurrafay.khan@mail.utoronto.ca

Nafio Miah
Student# 1005235522
nafio.miah@mail.utoronto.ca

ABSTRACT

The Group 27 Project Team plans to create a Deep Learning application using a Convolutional Neural Network model that can predict the waste disposal category that any given input image of a waste item belongs to. In this report, we reiterate our rationale for this project, discuss our progress, explain our data processing steps, our baseline model, and our primary model. —-Total Pages: 9

1 BRIEF PROJECT DESCRIPTION

As global temperatures rise, they bring along a variety of environmental issues that are of concern to humanity. The improper disposal of waste plays a factor in this increase. Hence, the main objective of our project is to design a system that aids in and encourages the correct disposal of waste by helping users correctly categorize their waste.

Applications such as TOwaste currently exist to help users correctly dispose of their waste items. However, these apps are restricted to the contents of their databases. These databases become increasingly outdated as new products emerge that vary in their material composition. The existing databases cannot take into consideration these new products until they are manually updated to include them.

Our model aims to eliminate the need for manual database updates by utilizing the predictive capabilities of Deep Learning. The Convolutional Neural Network (CNN), which is a specific type of Deep Learning architecture, is known for solving classification problems when given image data. The manual construction of a database that contains a classification for every single waste material that could exist is impossible as the possibilities are endless. Deep learning solves the classification problem more efficiently as it will be trained using a number of images to learn how to make predictions to classify new and unseen images in the future.

Figure 1 provides a high-level overview of our model's input and outputs. The input to our model will be an image of a waste item, and the output of the model will be the disposal bin category that our model believes is appropriate for that waste item.

2 INDIVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

The team believes we are working at a good pace to complete this project by its deadline. We have taken a collaborative and democratic approach for this project, and believe that we have been collaborating well thus far. Using a democratic approach ensures that each member's opinions are heard. Using a collaborative approach allows for a better learning experience for everyone as they get to contribute to different parts of the project and can receive support as we each have our different strengths and weaknesses. Our communications are mostly conducted through a Facebook Messenger group chat where we communicate through text messages and group calls. This allows us to

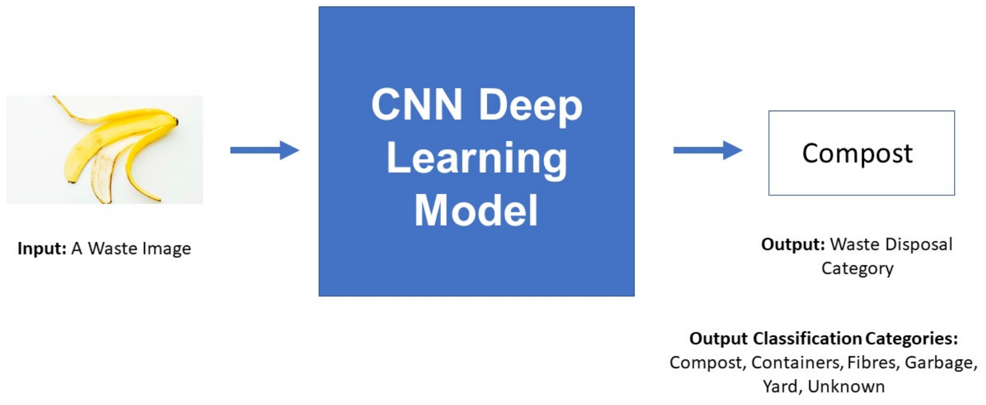

CNN Deep Learning Model
Compost
Input: A Waste Image
Output: Waste Disposal Category
Output Classification Categories:
Compost, Containers, Fibres, Garbage, Yard, Unknown

Figure 1: High Level Overview of Model's Input and Output

collaborate and provide updates. We also utilize Trello, which is an online project management tool, that allows us to view our tasks and progress in a visual dashboard. The code for our project resides in shared Google Colab Notebooks, and their associated files and folders are all uploaded to a shared Google Drive. This enables us to see changes in real time. This also allows for collaborative coding sessions which we conduct through Facebook Messenger's screenshare and video call features.

Table 1: **Completed Tasks by Progress Report Deadline**

Tasks	**Objective**	**Contributors**	**Start Date**	**Date of Completion**
Data Processing	Ensure project's data is of good quality and split data into training, validation and testing datasets	Abdurrafay, Nafio	October 13	October 18
Implement the Baseline Model	To create a working baseline model	Athavan, Abdurrafay	October 19	October 22
Testing the Baseline Model	Gather results using Baseline model to compare against our CNN model	Athavan, Shadman	October 23	October 24
Determine and design a CNN Architecture	Determine the number of convolutional layers, Fully Connected layers, kernel size, activation functions and a suitable optimizer	Everyone	October 21	October 21
Write the code for the CNN Architecture	Construct a Python Class that represents our initial CNN model	Shadman, Athavan	October 22	October 23

Creating the associated functions required to train and validate the CNN	Program the necessary Python functions to train the CNN on the training set, to tune its hyperparameters on the validation set, and to plot accuracies.	Shadman, Nafio	October 23	October 25
Train the CNN and tune the hyperparameters	Determine an optimal set of hyperparameters that does not result in overfitting or underfitting	Nafio, Abdurrafay	October 25	November 4

Table 2: **Remaining Tasks After Progress Report Deadline**

Tasks	**Objective**	**Contributors**	**Start Date**	**Deadline**
Creating the Disposal Bin Classifier Function	Create function to display CNN model output in a useful form for end user	Abdurrafay, Nafio	November 5	November 7
Testing the CNN Model	Obtain testing accuracies to evaluate the performance of our model	Athavan, Shadman	November 11	November 14
Create Project Presentation Script	Compile what the group will share in the project presentation	Everyone	November 14	November 18
Record Project Presentation	Have a recorded presentation following course's guidelines	Everyone	November 19	November 24
Write Project Report/Final Deliverable	Summarize our project, and final results into a final deliverable for submission	Everyone	November 25	December 1

3 Data Processing

As mentioned in the Project Proposal, we will be using the Waste Segregation Image Dataset from Kaggle [1]. This dataset has been split into two primary categories of either Biodegradable or Non-biodegradable items. Each of those contain four sub-categories for different kinds of biodegradable items (Food, Leaf, Paper, Wood) and non-biodegradable items (Electronic, Metal Cans, Plastic Bags, Plastic Bottles).

From this dataset, we are using a total of 9197 images for training and 3083 images for validation. As mentioned as a comment in the published dataset, there are two defective files which cause errors during modelling. We have discarded those two files from our training and validation subsets. Some categories can vary heavily in the general appearance of their images (such as food) compared to others (such as plastic bags). To account for this, we have split the training and validation datasets proportionally to allow more data for highly variable classifications for proper learning and classification purposes, as shown in the training and validation testing data split shown in Figure 2. As for testing, we plan on using a subset of the remaining images in the Kaggle Dataset as well as a combination of images from other datasets to ensure our model's accuracy with data that it has not been seen before. As the classification categories vary within different datasets, we are collecting data and manually sorting them into the classification categories we have. Some datasets such as the trashnet dataset [2] already are classified in ways similar to ours. We believe that using images from datasets such as this as well as the remaining subset from the Kaggle dataset will be sufficient to test our model's performance. To account for the variability in categories mentioned above (highly variable images of food versus similar looking images of plastic bags), we are creating the testing dataset following the same proportions of categories used during training, as shown in the testing data split shown in Figure 3.

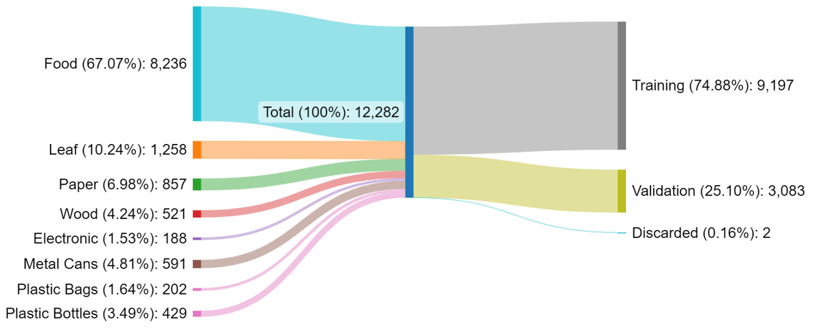


Figure 2: Training and Validation Data Breakdown

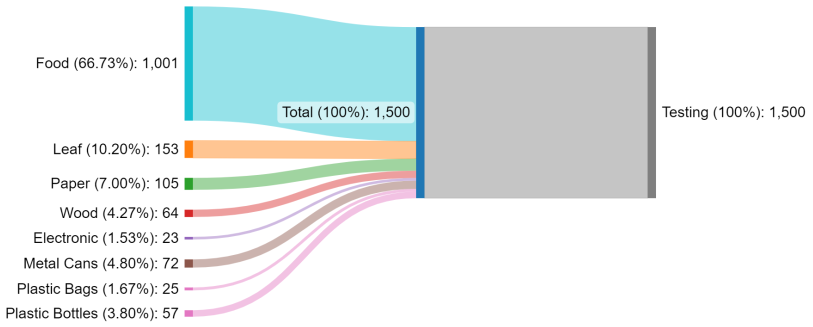


Figure 3: Test Data Breakdown

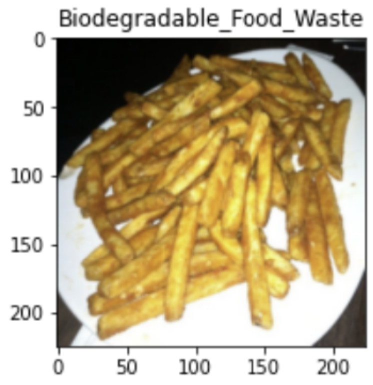


Figure 4: Example of Biodegradable Food Waste

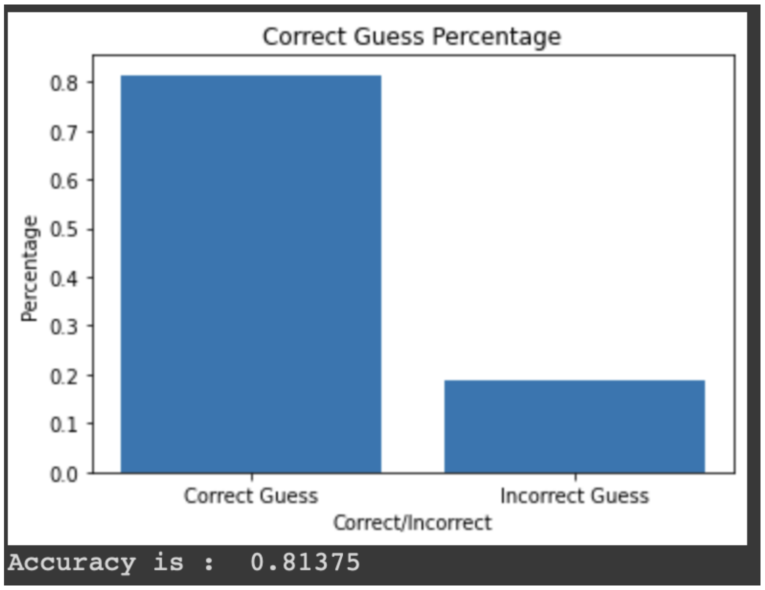
Due to the various sizes of the images in the datasets, we are resizing them to 224x224x3 when training them in the model. 224x244 ensures the image quality is enough to be distinguishable and it is not too much that would make our model inefficient in terms of time and processing. Figure 4 illustrates an example of a preprocessed image from our dataset.

4 Baseline Model

The random forest classifier algorithm is the baseline model that the team used to compare against the neural network. The random forest is a simple machine learning model created using multiple decision trees. Random subsets of the training dataset are used to create each unique tree. Once the random forest is created, validation/testing can be done to verify that the algorithm performs accurately. Each input that is inputted into the random forest traverses through each decision tree. Every tree will then output a class and the class with the most tallies is the random forest's prediction. The low correlation between the models(tree) allows the random forest to ideally converge on the correct class making it a strong baseline for classification problems. Another benefit of this algorithm is that it works well with large datasets. This is because the random subset sampling done in the beginning stages allows the data to be partitioned.

The algorithm was implemented using the RandomForestClassifier module located in the scikit-learn python package. The first step the team did was to load the train, validation and testing datasets. The next step was to format the data in order for it to be compatible with the RandomForestClassifier. We converted each 3x244x244(RGB) image to a single one-dimensional tensor for the classifier to use. The classifier was then trained using the training set followed by using the validation/testing set for the results. One hyperparameter for the RandomForestClassifer is the number of trees. The team tested the algorithm using various numbers of trees and discovered that the validation accuracy started to plateau at around 40 trees. This resulted in our decision to use 50 trees for the final model.

Figure 5 shows that the testing accuracy from the model was 81.375% with 50 trees. Therefore, the baseline goal for the primary model is to accomplish a testing accuracy that exceeds 81.375

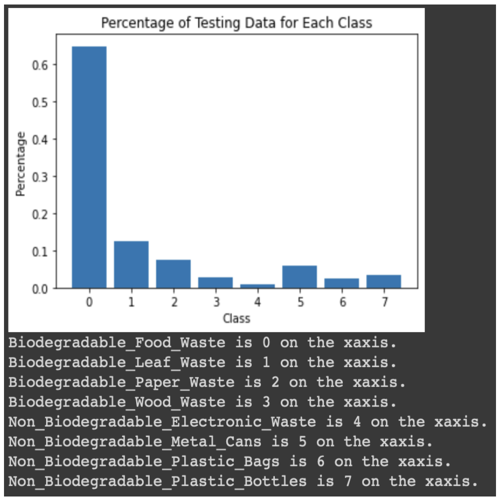


```
Accuracy is :  0.81375
```

Figure 5: Random Forest Classifier Testing Accuracy with 50 trees

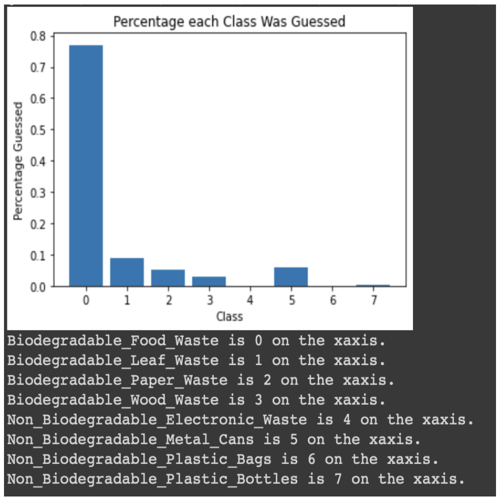
Furthermore, the team looked into the results to understand any trends that can be useful to pay attention to while working on the primary model. Figure 6 below shows the percentage of the testing data that each class accounts for. The food waste class is heavily frequent in the dataset that is being used. To check if the model is not predicting the most frequent class, the percentage of times that each class was guessed was also plotted. Figure 7 indicates that there may be a relationship between the number of data for each class and the number of predictions for each class. There is a slight bias towards guessing food waste compared to the other classes and that is the class that is most frequent in the dataset. Figure 8 plots the percentage of times each class was guessed correctly. It shows that the classes with less data are less likely to be predicted correctly. This is something the baseline model struggles to learn since the model is simple. Another measure of success for the primary model is to improve the accuracy for classes with less data compared to other classes.

Some struggles that the team faced when working on the baseline model had to do with low validation accuracy results from classes with less data points. Even after tuning the hyper parameters, we achieved low validation accuracy. The team concluded that this was due to the low complexity of the model. It was also decided that if the accuracy for the low frequency classes continued to be low for the primary model, changes would have to be made on the dataset itself in order to have an equal distribution of classes in the dataset.



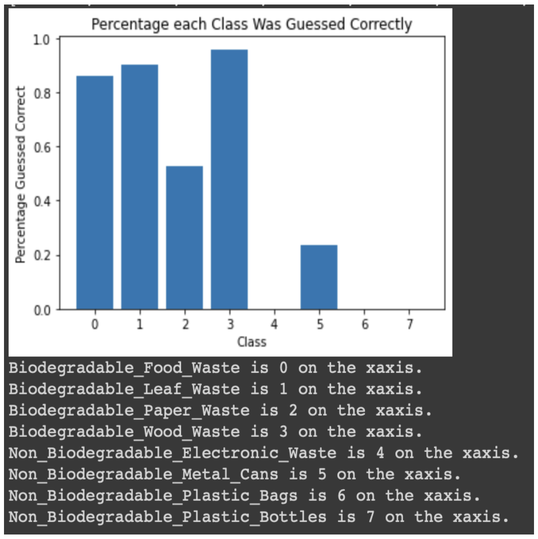
```
Biodegradable_Food_Waste is 0 on the xaxis.
Biodegradable_Leaf_Waste is 1 on the xaxis.
Biodegradable_Paper_Waste is 2 on the xaxis.
Biodegradable_Wood_Waste is 3 on the xaxis.
Non_Biodegradable_Electronic_Waste is 4 on the xaxis.
Non_Biodegradable_Metal_Cans is 5 on the xaxis.
Non_Biodegradable_Plastic_Bags is 6 on the xaxis.
Non_Biodegradable_Plastic_Bottles is 7 on the xaxis.
```

Figure 6: Class Distribution in the Testing Data



```
Biodegradable_Food_Waste is 0 on the xaxis.
Biodegradable_Leaf_Waste is 1 on the xaxis.
Biodegradable_Paper_Waste is 2 on the xaxis.
Biodegradable_Wood_Waste is 3 on the xaxis.
Non_Biodegradable_Electronic_Waste is 4 on the xaxis.
Non_Biodegradable_Metal_Cans is 5 on the xaxis.
Non_Biodegradable_Plastic_Bags is 6 on the xaxis.
Non_Biodegradable_Plastic_Bottles is 7 on the xaxis.
```

Figure 7: Percentages of Guesses for Each Class



```
Biodegradable_Food_Waste is 0 on the xaxis.
Biodegradable_Leaf_Waste is 1 on the xaxis.
Biodegradable_Paper_Waste is 2 on the xaxis.
Biodegradable_Wood_Waste is 3 on the xaxis.
Non_Biodegradable_Electronic_Waste is 4 on the xaxis.
Non_Biodegradable_Metal_Cans is 5 on the xaxis.
Non_Biodegradable_Plastic_Bags is 6 on the xaxis.
Non_Biodegradable_Plastic_Bottles is 7 on the xaxis.
```

Figure 8: Percentages of Correctly Classified Occurrences by Class

5 Primary Model

We opted to use a Convolutional Neural Network (CNN) as our primary model for the task of waste segregation. CNNs are extremely useful for tasks of image classification. CNNs work in a compatible way with images as input data: using filters on image results in feature maps. CNNs do not process data in a forward-facing way, which is a characteristic of artificial neural networks, but rather refer to the same data multiple times when creating maps [3]. These differentiate random forest trees, support vector machines and artificial neural networks from the CNN, which is currently state-of-the-art. Compared to its predecessors, the main advantage of CNN is that it automatically detects the important features without any human supervision [3].

The architecture that we implemented was: 5 convolution layers, 5 max pooling layers of stride 2 and kernel 2, and 4 fully-connected layers with the Rectified Linear Unit (ReLU) activation function. Each convolution layer has a kernel size of 3 and stride and padding of 1 respectively. A diagram of the architecture is included in Figure 9.

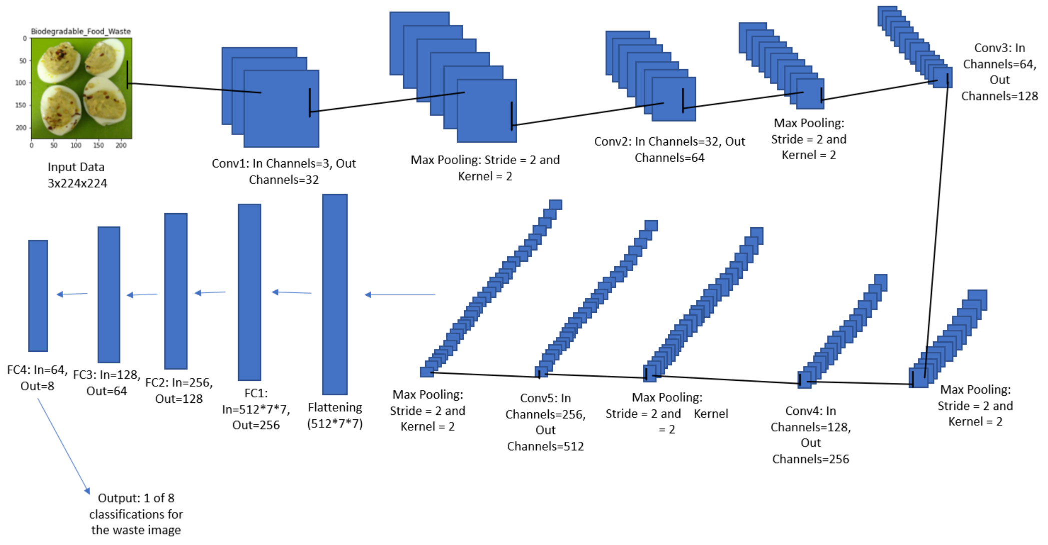

Biodegradable_Food_Waste
Input Data 3x224x224
Conv1: In Channels=3, Out Channels=32
Max Pooling: Stride = 2 and Kernel = 2
Conv2: In Channels=32, Out Channels=64
Max Pooling: Stride = 2 and Kernel = 2
Conv3: In Channels=64, Out Channels=128
Max Pooling: Stride = 2 and Kernel = 2
Conv4: In Channels=128, Out Channels=256
Max Pooling: Stride = 2 and Kernel = 2
Conv5: In Channels=256, Out Channels=512
Max Pooling: Stride = 2 and Kernel = 2
Flattening (512*7*7)
FC1: In=512*7*7, Out=256
FC2: In=256, Out=128
FC3: In=128, Out=64
FC4: In=64, Out=8
Output: 1 of 8 classifications for the waste image

Figure 9: CNN Architecture

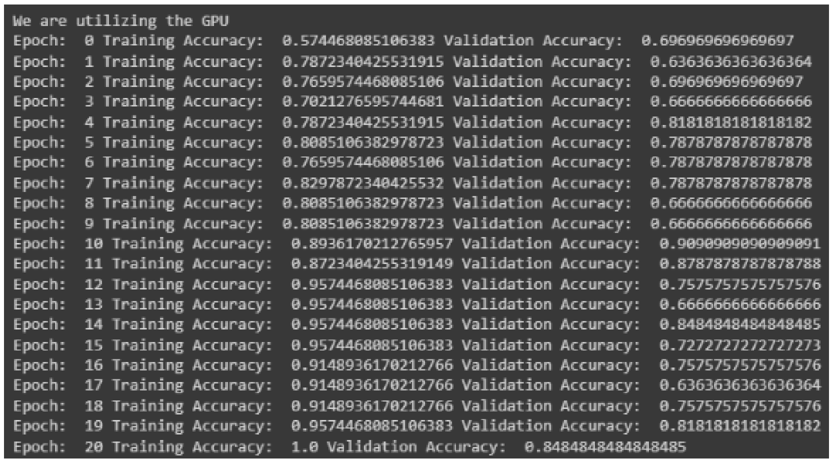
Note that this image is not to scale and simply shows the kernel size and stride for the pooling layers and input and output channels for the convolution and fully-connected layers. The more abstractions that the CNN is able to derive from image data, the more channels it has available. Since the CNN receives raw pixel data at the input layer, the number of filters tends to increase. Image data is particularly noisy. As a result, we let CNNs first extract some pertinent data from noisy, raw pixel data. After extracting the pertinent traits, we ask the CNN to develop more intricate abstractions. Thus, we increase the number of channels as we go deeper in this CNN. Eventually, we notice a plateau in terms of output for training and validation accuracy after 512 output channels in the fifth convolution layer. We chose to use 4 fully-connected layers as it increases the capacity of the network which helps as we have the highest number of parameters after flattening (512*7*7). Finally, we have the output as 1 of 8 classifications for the waste image. These are: Biodegradable Leaf Waste, Biodegradable Food Waste, Biodegradable Paper Waste, Biodegradable Wood Waste, Non-Biodegradable Electronic Waste, Non-Biodegradable Metal Waste, Non-Biodegradable Plastic Bottles, Non-Biodegradable Plastic Bags. These were loaded with our training and validation data.

The hyperparameters we finalized on were batch size of 50, learning rate of 0.03 and 20 epochs. We also used the Stochastic Gradient Descent (SGD) optimizer over the Adam optimizer, and validated its success in image classification tasks as the Adam optimizer performs aggressive training loss optimization, but loses on performance ultimately [4]. We tuned our hyperparameters while keeping the others consistent in order to generate reliable results. Table 3 illustrates our tuning and the respective highest accuracy results.

Table 3: **Hyperparameter Tuning Results**

Batch Size	**Learning Rate**	**Number of Epochs**	**Train Accuracy**	**Val. Accuracy**
32	0.03	5	71.9%	78.5%
64	0.03	5	76.6%	78.5%
50	0.03	5	81.8%	78.7%
50	0.01	5	75.3%	71.4%
50	0.05	5	72.1%	67.6%
50	0.03	20	100%	90.9%

As we can see, the hyperparameters were tuned depending on the results we were getting. Figure 10 and Figure 11 show the training and validation accuracy per epoch and the learning curve for our best hyperparameters respectively. The graph illustrates that underfitting and overfitting aren't issues in this case as the validation accuracy is usually under the training accuracy. For next steps, we will increase the number of epochs such that the training and validation will plateau. The 100% training accuracy we achieved is not an indication of memorization, but rather a testament to the plethora of data we are training the model with. This will allow us to achieve high testing accuracy.



```
We are utilizing the GPU
Epoch:  0 Training Accuracy:  0.574468085106383 Validation Accuracy:  0.696969696969697
Epoch:  1 Training Accuracy:  0.7872340425531915 Validation Accuracy:  0.6363636363636364
Epoch:  2 Training Accuracy:  0.7659574468085106 Validation Accuracy:  0.696969696969697
Epoch:  3 Training Accuracy:  0.7021276595744681 Validation Accuracy:  0.6666666666666666
Epoch:  4 Training Accuracy:  0.7872340425531915 Validation Accuracy:  0.8181818181818182
Epoch:  5 Training Accuracy:  0.8085106382978723 Validation Accuracy:  0.7878787878787878
Epoch:  6 Training Accuracy:  0.7659574468085106 Validation Accuracy:  0.7878787878787878
Epoch:  7 Training Accuracy:  0.8297872340425532 Validation Accuracy:  0.7878787878787878
Epoch:  8 Training Accuracy:  0.8085106382978723 Validation Accuracy:  0.6666666666666666
Epoch:  9 Training Accuracy:  0.8085106382978723 Validation Accuracy:  0.6666666666666666
Epoch:  10 Training Accuracy:  0.8936170212765957 Validation Accuracy:  0.9090909090909091
Epoch:  11 Training Accuracy:  0.8723404255319149 Validation Accuracy:  0.8787878787878788
Epoch:  12 Training Accuracy:  0.9574468085106383 Validation Accuracy:  0.7575757575757576
Epoch:  13 Training Accuracy:  0.9574468085106383 Validation Accuracy:  0.6666666666666666
Epoch:  14 Training Accuracy:  0.9574468085106383 Validation Accuracy:  0.8484848484848485
Epoch:  15 Training Accuracy:  0.9574468085106383 Validation Accuracy:  0.7272727272727273
Epoch:  16 Training Accuracy:  0.9148936170212766 Validation Accuracy:  0.7575757575757576
Epoch:  17 Training Accuracy:  0.9148936170212766 Validation Accuracy:  0.6363636363636364
Epoch:  18 Training Accuracy:  0.9148936170212766 Validation Accuracy:  0.7575757575757576
Epoch:  19 Training Accuracy:  0.9574468085106383 Validation Accuracy:  0.8181818181818182
Epoch:  20 Training Accuracy:  1.0 Validation Accuracy:  0.8484848484848485
```

Figure 10: Epochs and Respective Training and Validation Accuracy

Challenges we faced consisted primarily with the data. Initially, we used 12,282 images for train data and 3,084 images for validation data. Although lots of data gives us higher training and validation accuracy, it can also be an issue given the constraint of using a free Google Colab account and the time crunch for submitting this progress report with metrics. Thus, we often timed out with GPU usage on Google Colab and found it difficult to execute our models with different hyperparameters. In order to combat this, we reduced the number of training and validation images to a number which still allows us to reach a high accuracy while giving us quicker results and not timing out. Furthermore, another preliminary challenge we noticed was with our TrashCan 1.0 test dataset [5]. When trying some unit tests by passing in one image from this dataset to see if we can generate a classification, we noticed that the model had a difficult time predicting the correct result. This is because the TrashCan 1.0 dataset only has images of underwater waste. We deemed this out of the scope of our model as it was trained and validated exclusively with waste images not from underwater. To resolve this, we found the TrashNet dataset [2] that provided better data that was more transferable to practical use for the everyday household and our model alike.

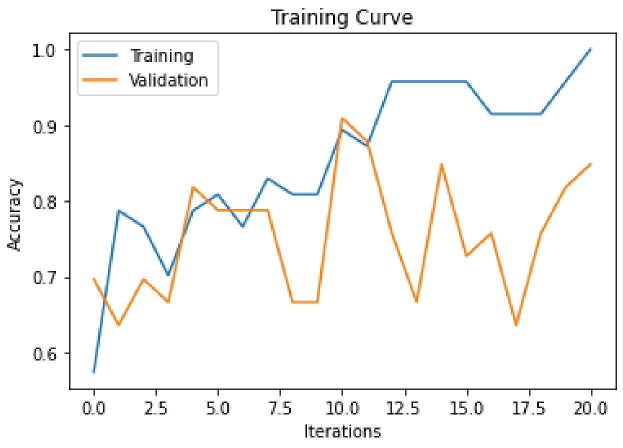


Figure 11: Training and Validation Learning Curves

6 Link to Github or Colab Notebook

Here is the link to the Colab Notebook:
Primary Model Colab Notebook

Here is the link to the Baseline Model Colab Notebook:
Base Model Colab Notebook

References

[1] A. Dutt, "Waste segregation image dataset," Sep 2022.

[2] G. Thung, "Garythung/trashnet: Dataset of images of trash; torch-based cnn for garbage image classification."

[3] A. Kumar, "Machine learning - training, validation, and test data set," Jun 2021.

[4] A. Miko, "Agamiko/waste-datasets-review: List of image datasets with any kind of litter, garbage, waste and trash."

[5] P. Mishra, "Why are convolutional neural networks good for image classification?," Jul 2019.