WASTE SEGREGATION USING CNN AND ALEXNET MODELS

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

This report summarizes Group 27's Deep Learning project. We explored 2 neural network architectures (our own CNN and an AlexNet) 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, highlight some similar existing work in the field, explain our data split and processing, explain our architectures, explain our baseline model, summarize our results, evaluate our model on new data, and discuss takeaways, ethical dilemmas and our project's difficulty level for this course. —-Total Pages: 7

1 INTRODUCTION

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 These databases become increasingly outdated as new products emerge that vary in their material composition.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.

2 ILLUSTRATION

Figure 1 offers a high-level overview of our system. As we explored 2 main architectures in our project, the Deep Learning Neural Network model represented either our own Convolutional Neural Network (CNN) architecture design, and a standard AlexNet architecture design. We describe these architectures in detail in the Architecture section of this report. The model's input is a 3x224x224 image of a waste item, while the model's output is a disposal bin category.

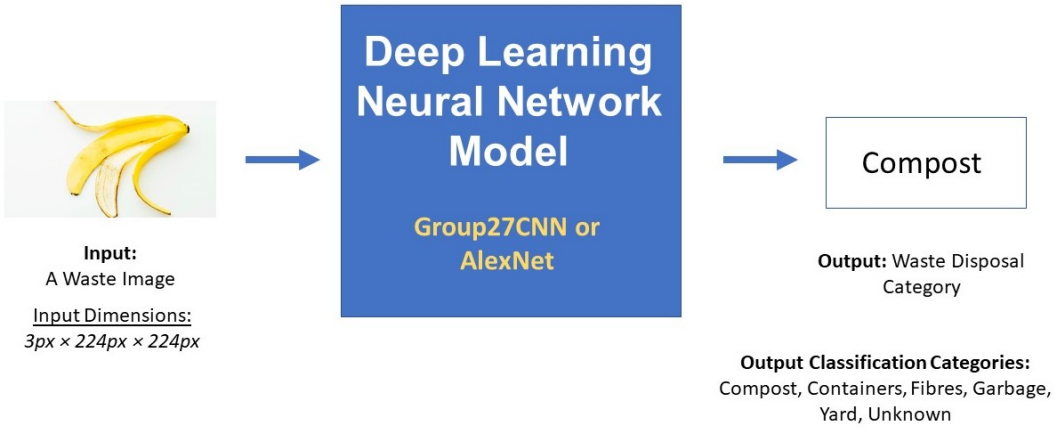

Figure 1: High Level Overview of Model's Input and Output

3 Background and Related Work

4 Data Processing

We used the Waste Segregation Image Dataset from Kaggle. The dataset consists of 2 primary categories: biodegradable and non-biodegradable. The biodegradable category consists of the Food, Leaf, Paper, and Wood classes, while the non-biodegradable category consists of Electronics, Metal Cans, Plastic Bags and Plastic Bottle classes. We then further map these 8 classes into the various methods of disposal such as the Green Compost Bin, the Containers bin, and Garbage.

This dataset was imbalanced as the Food class constituted the majority of the dataset. We presume that this was due to the nature of variability in how food's appearance. In addition, Google Colab's GPU limitations limited us on adding data from many other datasets such as the Trashnet dataset. Those other datasets were then used for testing. We used 9197 images for training, and 3083 images for validation. The remaining images were used for testing. Two images required removal as it caused errors during model training.

Before feeding data into our model, some processing was required. As the images varied in size, we resized all the images to be 224x224x3. Figure 2 shows an example of a preprocessed image in the training dataset. A resolution of 224x224 was chosen to ensure that the image contains enough detail to be identifiable while not being too fine to the extent that it results in an inefficient model.

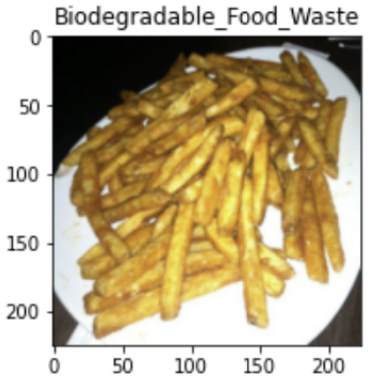

Figure 2: Example of Biodegradable Food Waste

Furthermore, the dataset also contained many cartoon images. We believe cartoon images are not the best representation of real-world waste items. However, they can be used to extract high-level features. Hence, we limited their quantity.

To explore the effect of the class imbalance, we created a secondary dataset consisting of duplicated images to normalize the classes. We initially thought of increasing the sizes of the non-food classes, thinking the model would perform better with more data. However, we recognized that this would result in longer training times and we were restricted by Google Colab's GPU limitations. Hence, we had to decrease the size of our Food class, while increasing the non-food classes to move towards a more normalized distribution.

5 ARCHITECTURE

We explored two main neural network architecture models. Figure 3 illustrates the first of the two models which is our own CNN architecture, while Figure 4 illustrates our second model which follows the standard AlexNet Architecture.

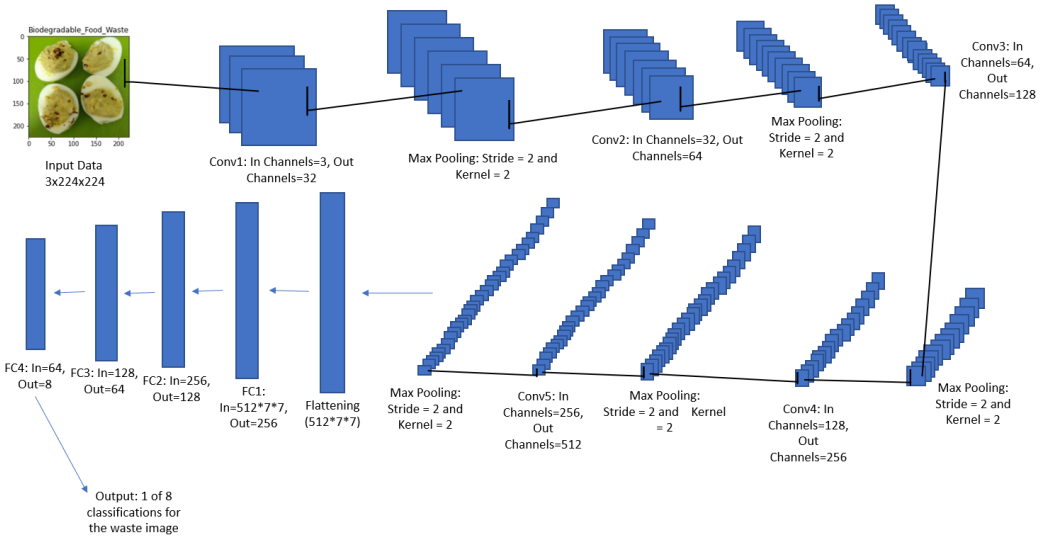

Figure 3: CNN Architecture

As illustrated in Figure 3, our own CNN architecture contains 5 convolution layers, 5 max pooling layers of stride 2 and padding 2, and 4 fully connected layers with ReLU activations. Each convolution layer has a 3x3 kernel filter with stride and padding of 1, respectively.

As we trained our own CNN model to maximize its performance, we thought we could do better. With AlexNet being a pre-trained model that would be much more efficient in terms of performance, we decided to use an AlexNet architecture as well [AK1].

As illustrated in Figure 4, the AlexNet model has 5 convolutional layers, and 3 fully connected layers. Our first convolution has an 11x11 filter size, the second convolution has a 5x5 filter size, and the 3rd, 4th, and 5th convolution layers have 3x3 filter sizes. We also use 3x3 max pooling with stride 2 after the first, second and fifth convolution. The fully connected hidden layers have 4046 neurons and the output layer has only 8 neurons as we have 8 different classifications.

6 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

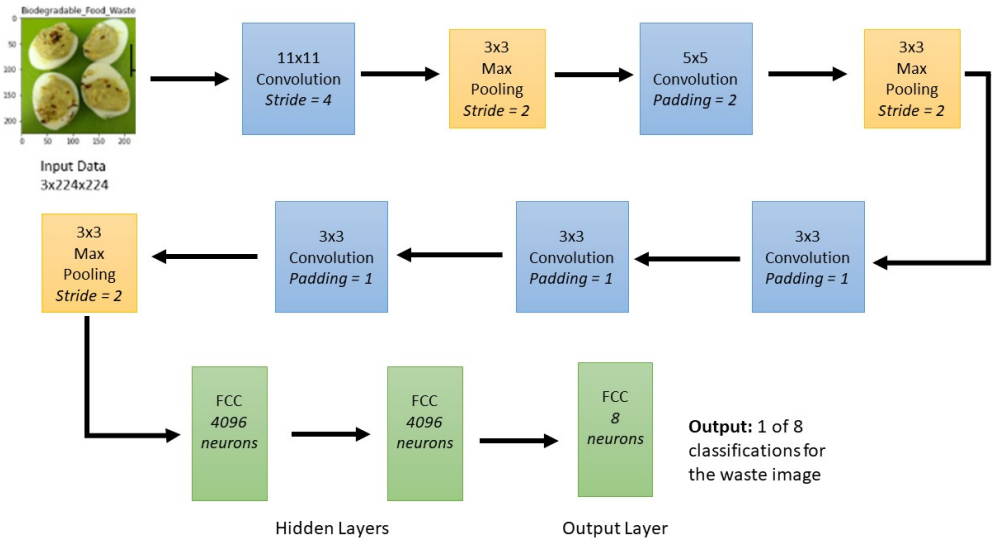

Figure 4: AlexNet Architecture

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

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

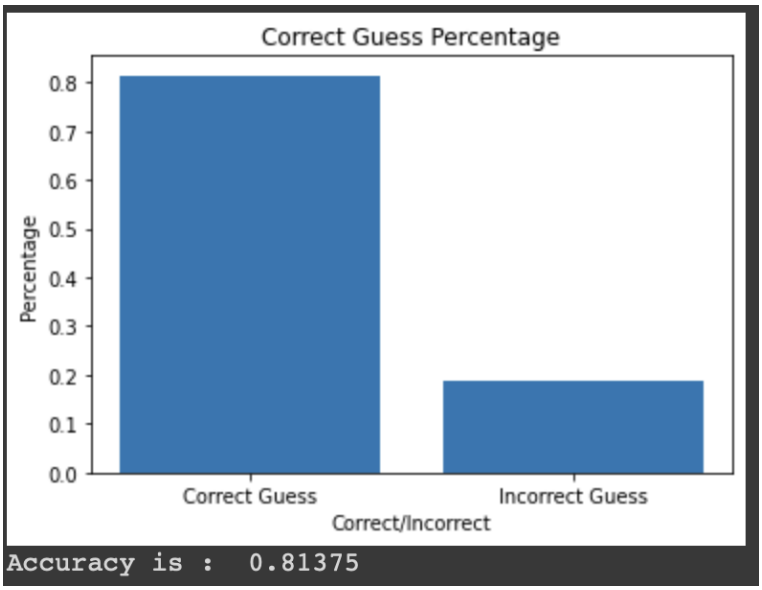

Figure 5: Random Forest Classifier Testing Accuracy with 50 trees

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.

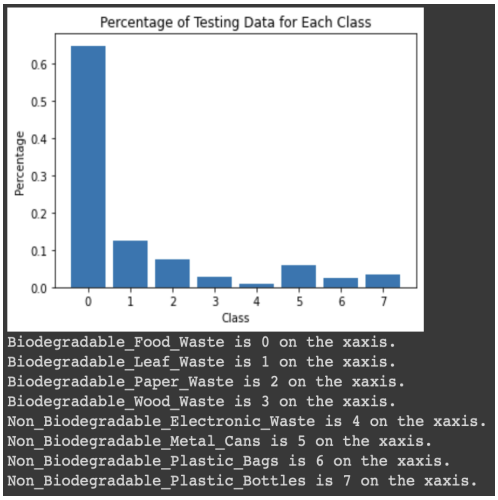

Figure 6: Class Distribution in the Testing Data

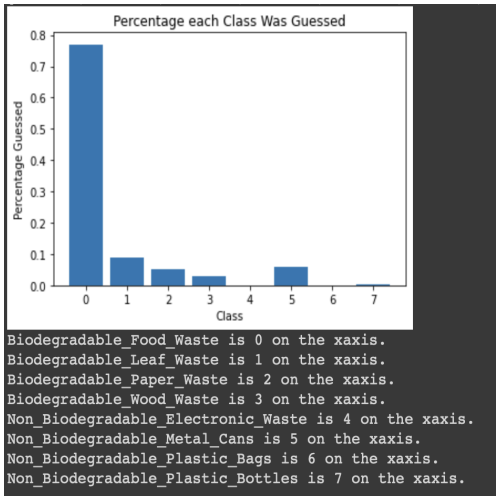

Figure 7: Percentages of Guesses for Each Class

7 Quantitative and Qualitative Results

To measure how our models perform, we used training, validation, and test accuracy as our main metrics. We also found the ratio of correctly predicted classifications for every class. Upon training the graphs, we displayed the train and validation accuracy for every epoch as it shows how our models perform incrementally. Initially, we trained our CNN model with 15 epochs without data duplication. The results for the CNN model trained with 15 epochs are illustrated in Figure XX. The accuracies are calculated as an average over its last 5 trained epochs. For the curve without data duplication, there is fluctuation for the validation curve, which resulted in us performing data augmentation to mitigate the fluctuation and improve steadiness. After data duplication, we notice more plateau for both the training and validation curves, as well as less jagged fluctuation for the

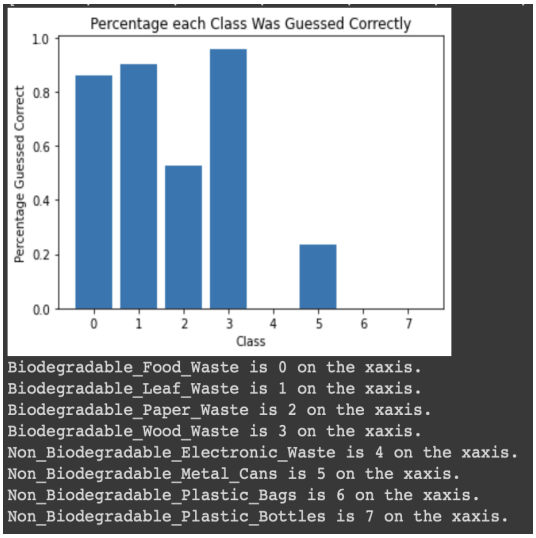

Figure 8: Percentages of Correctly Classified Occurrences by Class

training loss. We achieved approximately 4% higher training accuracy, but lost 8.87% validation accuracy, shown in Table YY.

We also trained two instances of our AlexNet model - one with data duplication and one without. Similar to our CNN model results, we noticed improved steadiness on both training and validation curves after implementing data duplication. We also gained almost 8% in training accuracy; the trade-off however was that we lost 8.72% in validation accuracy shown in Table YY. It is interesting to note how data duplication increased training accuracy for both our models, yet sacrificed almost an identical 9% in validation accuracy.

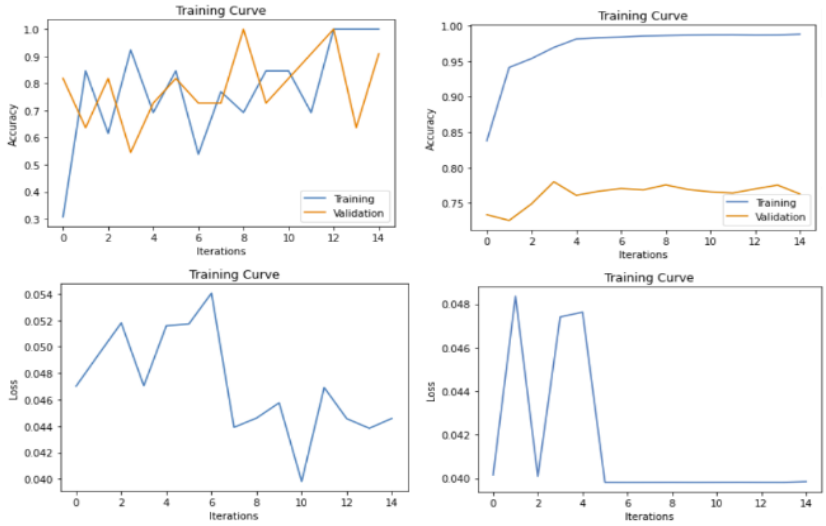

Figure 9: Percentages of Correctly Classified Occurrences by Class

After attaining training and validation accuracy, we found the test accuracy for the models which had the highest validation accuracy: CNN and AlexNet models without data duplication. For the CNN and AlexNet models we attained 85.5% and 94.1% on unseen test data respectively. These results were satisfactory as it demonstrates good classification performance on test data that comes from a different source than our initial training and validation data. We further wanted to see the

ratio of correct classification for every class, as that would give us a better idea of which classes obtained the most incorrect predictions.

8 Evaluate model on new data

9 Discussion

10 Ethical Considerations

Our project aims to enhance waste disposal applications that exist currently. These applications may need the user's permission to use their device's camera and access their geographic locations to know of the disposal methods of their location. Our project deals solely with the classification of waste item images, hence, gathering of user information that could possibly result in privacy breaches is not required by the application.

In addition, our model cannot claim to successfully classify every waste item due to the impracticality of training on every possible waste item. Training is limited to the images found in the training dataset. For example, our model struggles to correctly classify the disposal of stuffed animals as it is not trained for that task by including a number of stuffed animal images in the training dataset. The model is also only trained to classify images into the 8 categories described by the Waste Segregation Image Dataset. Therefore, misclassifications are possible, and we cannot guarantee that the model can always give a correct classification. Misclassifications may negatively impact the environment which is an ethical concern that we strived to mitigate by maximizing our model's performance.

11 Project Difficulty / Quality

12 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