Multi-Class Garbage Identifier

Rene Lamb - A20457909
Nishanth Rao - A20466002
Jean Haberer - A20496422

November 2021

Abstract

Waste management has been a global challenge for many years, and it will continue to be so unless we recycle the waste efficiently. Many of the inefficiencies within waste management are due to the mixing of various waste types that should be separated. This is often a result of ignorance or confusion regarding how to separate waste. In this project, we aim to aid individuals in their disposal of waste by developing a machine learning algorithm that identifies eight different waste classes. To develop our classifier we designed a custom convolution neural network which can identify waste items within the waste categories at approx. 72% accuracy. The results of our project show that a fairly accurate machine learning model can be developed to help identify waste. The ultimate output of our experimentation is a model that can eventually be applied to a computer vision application to identify waste in real-time.

1 Introduction

The average American discards seven and a half pounds of garbage every day [1]. The amount of wasted resources that are sent to landfills continues to put a strain on our environment. Recycling material, as opposed to discarding it within a landfill, can help reduce this environmental burden by reducing the amount of raw materials extracted and in turn reduce the energy cost for extracting these materials.

Much of the responsibility in recycling lies with the individual. Individuals need to know how to separate different types of waste when discarding them. Mixing recyclables with non-recyclables can lead to inefficiencies in waste management. Oftentimes, individuals may incorrectly dispose of waste due to apathy or lack of awareness. To help solve this issue, we have developed a machine learning algorithm to identify different types of garbage for individuals.

2 Related Work

There are some similar applications for using computer vision to classify garbage, however, the classification methods and how the models in these previous applications are utilized ultimately will differ from our approach.

In Yang and Thung's research paper [2], garbage is classified into glass, paper, metal, plastic, cardboard, and trash. We expand these classifications to include "organic", "E-waste" and "Chemical Waste" categories as well. Additionally, the training images are pieces of trash that are overlaid on a white background. The goal of our application is to utilize a machine learning algorithm to provide visual feedback to a person throwing an individual piece of trash away. Therefore, our approach is more robust since it identifies different types of garbage regardless of the background.

In Wang and Zang's research paper [3], collected garbage in large quantities is identified in images with various backgrounds. Unlike their application, our application is only concerned with identifying a single piece of garbage at a time. Additionally, our application specifies the category of garbage for that individual piece.

In Baras, Ziouzios, Dasygenis, and Tsanaktisidis' research paper [4], they outline a "smart recycling bin" prototype that identifies and classifies waste. However, in their application, the waste enters a single bin and is not identified until it is already inserted into the bin.

3 Problem Description

In this paper we define "garbage" and "waste" as used items to be discarded within a trash bin. Our application utilizes a convolution neural network to classify garbage into eight classes: plastic, glass, metal, paper, cardboard, organic waste, E-waste, and chemical waste (see classes in the figure below).

Figure 1: Garbage Classes

There is a lot of diversity in shapes, colors, and sizes of garbage. For example,

glass can come in the form of a bottle, cup, vase, etc. Organic waste can be a variety of different foods. Therefore, to solve the problem of identifying waste, we needed to develop a model that is able to identify key features that are common between items of the same class. Additionally, we needed to collect a large and diverse data set of images so that our model can correctly learn those features.

3.1 Theory

There are a lot of ways to predict a garbage class using machine learning. Here is a part of the theory to know behind our approach. Our task is an image processing task, we are working with pixel matrices.

3.1.1 Convolution Neural Network

This is a deep learning algorithm that takes an image as input and assigns weights and biases to the features in the image to help differentiate it from other images.

3.1.2 Max Pooling

This is one of the neural network layers in CNN. It takes the highest element in the matrix of pooled numbers for further processing that serves as an input to its subsequent layers. It is used to decrease the processing load in the subsequent layers.

3.1.3 Dense Layer

This neural network layer receives input from the previous layers to each neuron and performs matrix vector multiplication on the elements. The obtained values are parameters that can be trained. These trained values are updated through back propagation.

3.1.4 Batch Normalization

Batch normalization layers are utilized in our model to normalize the data at each layer. it is implemented by using the mean and standard deviation of each batch that passes through the layer. It helps in speeding up the convergence.

3.1.5 Flattening layer

Flatten layers are used to convert the data into one-dimensional array for the subsequent layers. The matrix is converted into a single feature vector after the flattening layer.

3.1.6 Gradient Descent

Gradient descent is an optimization function used to reduce the value of the loss function in the training of a machine learning model. Optimization techniques like this are helpful when the parameters cannot be calculated analytically.

3.2 Application

An additional problem when developing our application was dealing with images. Processing images can be very computationally demanding due to the fact that each pixel can be considered a feature and each has a separate red, blue, and greed numerical value. To train the model in a reasonable amount of time, we knew we would need to take advantage of GPUs. We, therefore, developed and trained our model on a Google Colab notebook which gave us access to powerful GPUs from remote servers. In creating our application, the data pipeline is as follows. We obtained our data set from an existing online source (discussed in further detail later in the paper). We generated a data-frame structure and generate labels using the pandas library. We split the data using an 80-20 Train/Validation distribution using the scikit library. From there we perform image pre-processing and feed the data into our k-fold training/validation module to train the model. The pre-processing, training, and model creation are all performed using the Tensorflow library as a back end that is accessed through the Keras API. From there, our script produces an output weights file which we use to generate predictions and analyze our results. See the figure below for the overall classifier pipeline.

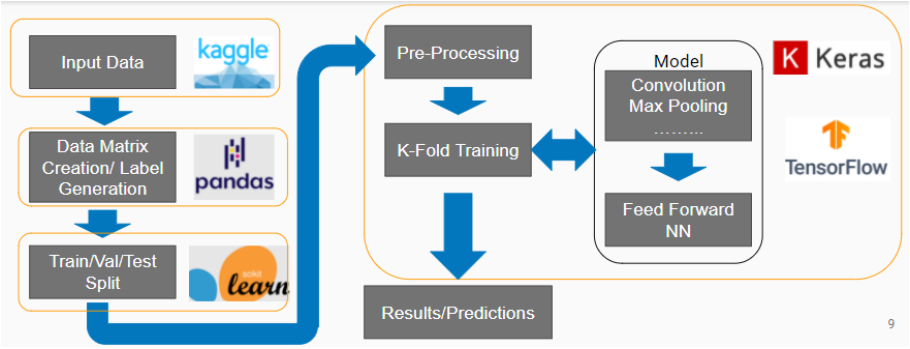

Figure 2: Classifier Pipeline

4 Data gathering

4.1 Where does the data come from?

As a reminder, we want to categorize garbage according to those 8 classes: Cardboard / Chemical waste (Batteries) / E-Waste / Glass / Metal / Organic

/ Paper / Plastic.

We had to find data sets that could give us a large set of images for all those categories. After going through Kaggle, we decided to pick data from two particularly representative data sets:

The first one (https://www.kaggle.com/mostafaabla/garbage-classification) already has a lot of data on the different categories we wanted to differentiate. It is well classified, has a lot of examples, and its size is not too large. Images are already compressed and downscaled.

The second one (https://www.kaggle.com/kaustubh2402/ewaste-dataset) fills the gap of the first one giving us examples of E-Wastes. The downside of this data set is that its size is big (more than 1GB) and images are full-sized.

4.2 How did we create our data set?

After gathering these two data sets, we had to normalize the size of each example. We didn't want our images to be too large to simplify our model and reduce the number of parameters, but we still wanted to have enough resolution to be able to distinguish easily the different classes. We found that a good compromise would be 256x256 pixels. Finally, to normalize the data set, we ran the command `sips -Z 256 *.jpg` so that each image has a size of 256x256.

We could now use this data set to train and test models by importing them into our colab project. To do so, we transformed the data set tree into a data frame that contains the image matrix and also the label of each image.

4.3 Initial Model

For the initial model, we started with a modest CNN model with a 2D Convolution layer, a Max Pooling layer, a Flattening layer, and two Dense layers with the last layer being the output layer with a softmax function. The initial model summary is in the appendix section. To train the model, we used adam for the optimizer since it is a decent default optimizer, to begin with. It was later abandoned since the stability of adam optimizer was not favorable and gave us inconsistencies in the validation accuracy values. We used categorical_crossentropy as the loss function. The number of epochs initially was set as 20. We had experimented on the number of epochs set to 30 to increase the accuracy of the model.

We tried different optimizers such as SGD. However, we abandoned that since the convergence rate in SGD was very slow. Adadelta was also considered, but the stability of the adadelta optimizer was not good, so it had to be discarded.

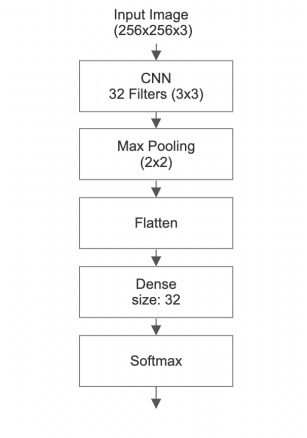

Figure 3: Initial CNN Model

Parameter Name	Selection
Optimizer	Adam
Loss	Categorical Crossentropy
Metrics	Accuracy
Epochs/Fold	20
Batch Size	16
Steps/epoch	Training Data Size/Batch Size

Table 1: Hyperparameters for initial model.

4.4 Final Model

The final model is a modification of the model developed by Jason Brownlee [5]. It consists of multiple iterations of convolution, batch normalization, and max pooling layers before it is flattened and fed to a feed forward neural network with multiple dense and batch normalization layers. The output layer consists of a soft max function. See image below. For the full model, see appendix at the end of the paper.

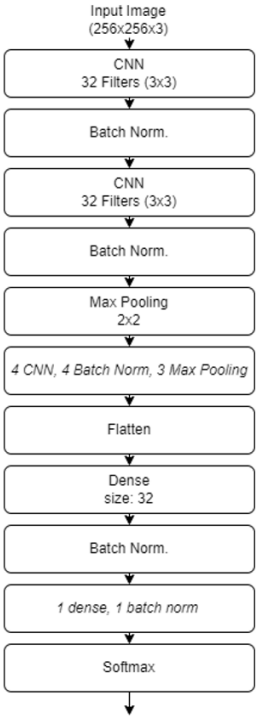

Figure 4: Final CNN Model

Parameter Name	Selection
Optimizer	Adagrad
Loss	Categorical Crossentropy
Metrics	Accuracy
Epochs/Fold	15
Batch Size	16
Steps/epoch	Training Data Size/Batch Size

Table 2: Hyperparameters for final model.

For training, the adagrad optimizer was selected with an initial learning rate set to equal 0.001. This adaptive learning rate allowed us to reach convergence in a shorter amount of time. Because this is a multi-class classification problem, loss for each iteration was calculated using categorical crossentropy. The number of epochs was chosen based on experimentation. We found that our accuracy leveled off at about 15 epochs. To increase training speed, we set our batch size to be 16 images (i.e. 16 images are processed before weight parameters are updated)

5 Results

5.1 Model training and K-Fold Validation

For training the model, we used K-Fold validation with 5 splits. In the training, we obtained validation accuracy consistently in the range of 70% to 75%. and the validation loss in the range of 1.0000 to 0.8000. The pictures below are some of the training graphs obtained.

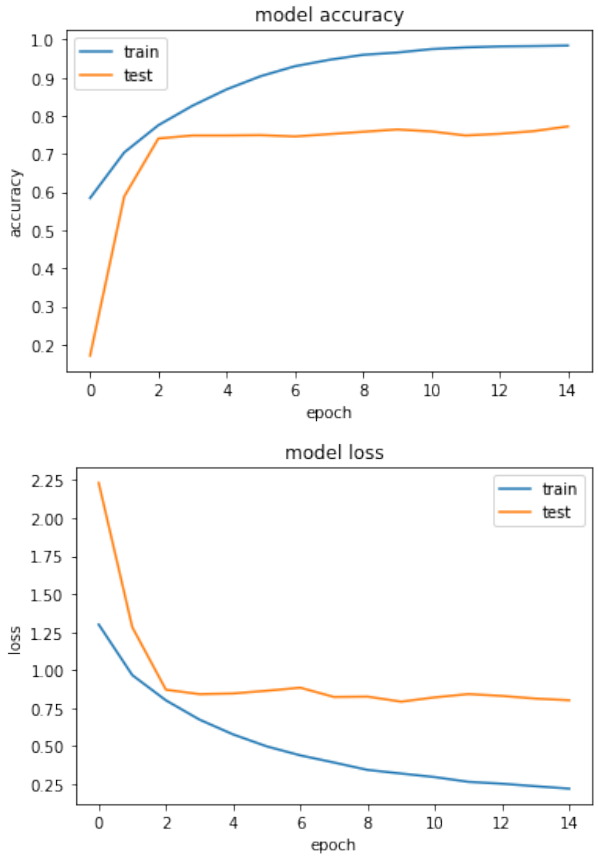

Figure 5: Model Accuracy and Model Loss for model generated by fold 5

5.2 Model Accuracy and Confusion Matrix

The above models were tested on an unseen dataset and the values and the model accuracy computed using evaluate_generator function turned out to be around 70%-75%. The pictures below are some of the confusion matrices obtained by the model when the true label was compared against the predicted label.

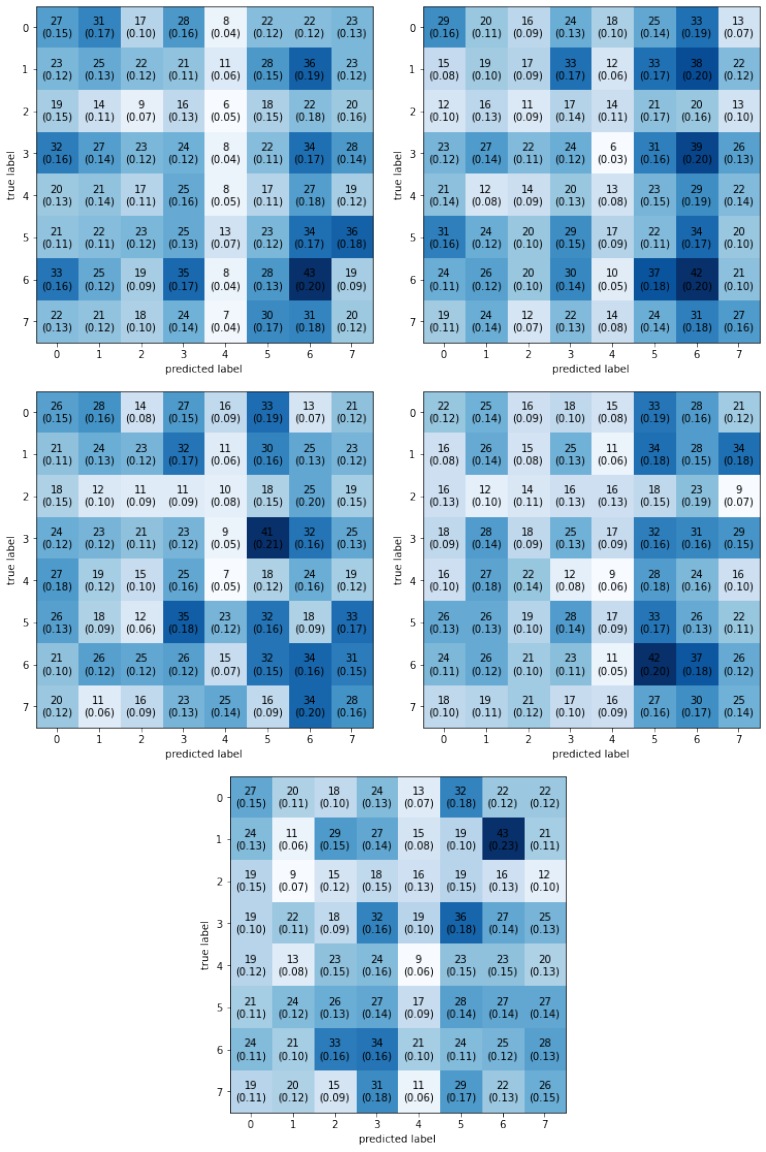

Figure 6: Confusion Matrices for all the models generated by kfold

5.3 Sample Test Results

To give a general idea of the accuracy of the model, we took a few pictures of items available to us and tested the model to get an idea of how the model performed with some of the unseen data as well as some of the invalid data, which includes pictures of ourselves. Some of the results are displayed below.

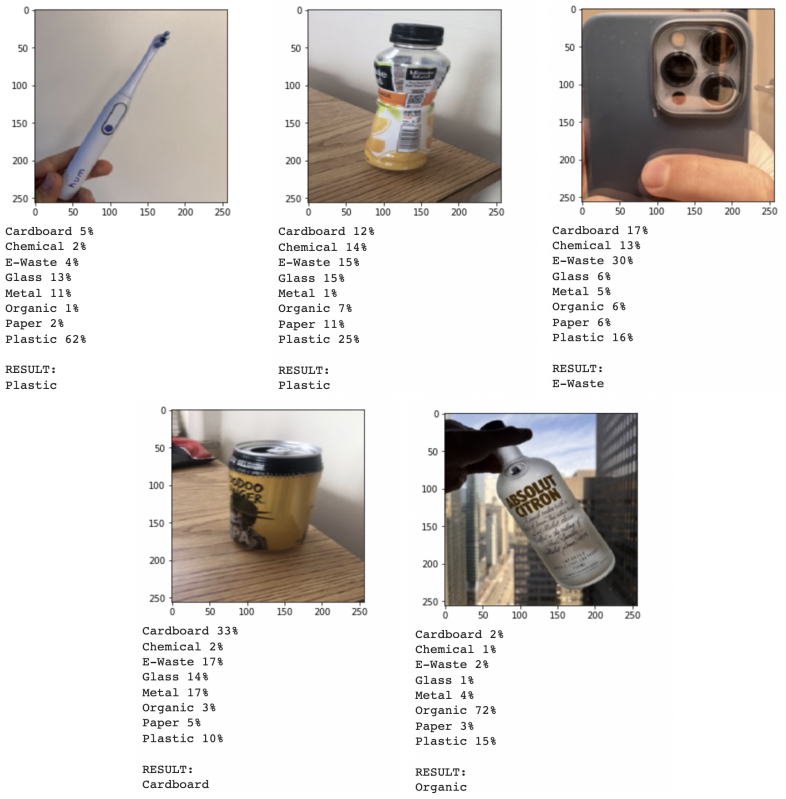

Figure 7: Sample images with their classification

6 Future work

A lot of work would still be necessary to bring this application to users. Especially to improve our model performance and our application process. Our future tasks include improving our model accuracy, generating a custom data set that consists of users "holding" garbage, and adapting the model to an application to give feedback in real time.

6.1 Improve performances

A way to improve the performances of our model would be to use transfer learning. There are already a lot of models that predict an object type with high accuracy. We could use these trained models to perform better in our tasks.

Another way to improve our model performances would be to add an object detection pipeline before our model so that the model is provided with only the object that it has to classify and no other random space.

6.2 Improve process

The current way to predict a garbage class with our application is to run lines of code. This cannot be brought like this to production. Ultimately, we would want the user to hold his/her garbage in front of a camera that would provide an image flow to our model which would predict the garbage class immediately. This feedback can be in the form of LED's on a multi-class trash bin that light up the corresponding receptacle for which to deposit the identified garbage.

7 Conclusion

Garbage recycling is a challenge everywhere in the world, we tried to create an application that could ease the task of classifying wastes. In our project, Machine Learning proves to be a powerful and very advanced tool to create such an application. But there is still a long road before distributing our smart-feedback application to full scale production. We would likely need to improve our image database by gathering a lot more images that are representative of real world scenarios in order for our model to be more accurate.

Ultimately, we believe our application is a good first step in helping to improve recycling habits. We believe our model can be adapted further so that it can be used in a real-time garbage identification system.

This paper is meant to show that we could actually develop such a solution today, and we should. Every step to a cleaner world is a step to a better world.

8 Appendix

8.1 Initial Model

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)            (None, 254, 254, 32)      896

 max_pooling2d_1 (MaxPooling  (None, 127, 127, 32)      0
 2D)

 flatten_1 (Flatten)          (None, 516128)            0

 dense_2 (Dense)              (None, 30)                15483870

 dense_3 (Dense)              (None, 8)                 248

=================================================================
Total params: 15,485,014
Trainable params: 15,485,014
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: Initial CNN Model

8.2 Full Model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (BatchNormalization)	(None, 254, 254, 32)	128
conv2d_1 (Conv2D)	(None, 252, 252, 32)	9248
batch_normalization_1 (BatchNormalization)	(None, 252, 252, 32)	128
max_pooling2d (MaxPooling2D)	(None, 126, 126, 32)	0
conv2d_2 (Conv2D)	(None, 124, 124, 64)	18496
batch_normalization_2 (BatchNormalization)	(None, 124, 124, 64)	256
conv2d_3 (Conv2D)	(None, 122, 122, 64)	36928
batch_normalization_3 (BatchNormalization)	(None, 122, 122, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 61, 61, 64)	0
conv2d_4 (Conv2D)	(None, 59, 59, 128)	73856
batch_normalization_4 (BatchNormalization)	(None, 59, 59, 128)	512
conv2d_5 (Conv2D)	(None, 57, 57, 128)	147584
batch_normalization_5 (BatchNormalization)	(None, 57, 57, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 32)	3211296
batch_normalization_6 (BatchNormalization)	(None, 32)	128
dense_1 (Dense)	(None, 16)	528
batch_normalization_7 (BatchNormalization)	(None, 16)	64
dense_2 (Dense)	(None, 8)	136

Figure 9: Full CNN Model

References

[1] Lbre.stanford.edu. 2021. Frequently Asked Questions: More on Recycling — Land, Buildings Real Estate. [online] Available at: <https://lbre.stanford.edu/pssistanford-recycling/frequently-asked-questions/frequently-asked-questions-more-recycling>

[2] M. Yang and G. Thung, "Classification of trash for recyclability status," CS229 Project Report 2016, 2016.

[3] Y.Wang and X.Zhang "Autonomous garbage detection for intelligent urban management" Shanghai University, 2018

[4] N. Baras, D. Ziouzios, M. Dasygenis, C. Tsanaktsidis "A cloud based smart recycling bin for waste classification" University of Western Macedonia, 2020

[5] Brownlee, J. (2020, August 27). How to develop a CNN from scratch for CIFAR-10 photo classification. Machine Learning Mastery. Retrieved December 3, 2021, from < https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification> .

[6] Cchangcs. (2018, November 24). Garbage classification. Kaggle. Retrieved December 4, 2021, from < https://www.kaggle.com/asdasdasasdas/garbage-classification/code.>