****

**AiCE Undergraduate Research Project**

**Final Report**

**Fall 2022 Semester**

***Trash Masters***

**Team Members**

**Inkaphol Siriwongwattana, Ink, isiriwon@cmkl.ac.th**

**Paksaran Kongkaew, Mammoth,** [**pkongkae@cmkl.ac.th**](mailto:pkongkae@cmkl.ac.th)

**Chan Bormei Suy, Peach,** [**csuy@cmkl.ac.th**](mailto:csuy@cmkl.ac.th)

***Advisor***

***Dr. Sally Goldin***

**9/12/2022**

**Table of Contents**

[**Table of Contents** 2](#_Toc121476012)

[**Chapter 1** 4](#_Toc121476013)

[**1.1** **Problem Statement** 4](#_Toc121476014)

[**1.2 Project Solution Approach** 4](#_Toc121476015)

[**1.3 Project Objectives** 4](#_Toc121476016)

[**Chapter 2** 5](#_Toc121476017)

[**2.1 Fundamental Theory and Concepts** 5](#_Toc121476018)

[**2.1.1 Digital Image Structures** 5](#_Toc121476019)

[**2.1.2 Image Operations** 5](#_Toc121476020)

[**2.1.3 Artificial Neural Network [ANN] or Forward Feed Neural Network** 7](#_Toc121476021)

[**2.1.4 Backpropagation** 7](#_Toc121476022)

[**2.1.5 Transfer Learning** 7](#_Toc121476023)

[**2.1.6 Convolutional Neural Networks** 8](#_Toc121476024)

[**2.1.7 Web Scraping** 11](#_Toc121476025)

[**2.2 Technologies** 12](#_Toc121476026)

[**2.2.1 PyTorch** 12](#_Toc121476027)

[**2.2.2 Resnet** 12](#_Toc121476028)

[**2.2.3 Google Lens** 13](#_Toc121476029)

[**2.2.4 DownThemAll** 14](#_Toc121476030)

[**2.2.5 Apex** 14](#_Toc121476031)

[**2.2.6 Keras** 14](#_Toc121476032)

[**2.2.7 SciKitLearn** 14](#_Toc121476033)

[**2. 3 Related Research** 15](#_Toc121476034)

[**2.2.1 The use of machine learning methods to classify images of trash** 15](#_Toc121476035)

[**2.2.2 Using Resnet50 to classify images with similar qualities** 15](#_Toc121476036)

[**Chapter 3** 16](#_Toc121476037)

[**3.1 Dataset Collection** 16](#_Toc121476038)

[**3.2 Structures for the Models** 17](#_Toc121476039)

[**3.3 Experimentation and Parameters for each of the Versions** 18](#_Toc121476040)

[**3.3.1 Background** 18](#_Toc121476041)

[**3.3.2 Dataset Versions** 18](#_Toc121476042)

[**3.3.3 Version History** 19](#_Toc121476043)

[**Chapter 4** 21](#_Toc121476044)

[**Chapter 5** 23](#_Toc121476045)

[**5.1 Summary of Accomplishments** 23](#_Toc121476046)

[**5.2 Issues and Obstacles** 23](#_Toc121476047)

[**5.3 Future Directions** 23](#_Toc121476048)

[**5.4 Lessons Learned** 24](#_Toc121476049)

[**References** 25](#_Toc121476050)

# **Chapter 1**

**Introduction**

* 1. **Problem Statement**

The concept of mismanaged trash has been one of Thailand’s most persistent and significant environmental issues. As the world’s tenth largest producer of plastic-based oceanic waste [2], Thailand needs a system which facilitates proper recycling habits among the Thai population. A recycling solution is hard to implement since the system will only work when it is embraced by both the government and the community. While the government in provinces like Bangkok has been pushing for the renewal of the city’s recycling effort, there hasn’t been a way for people who may be new to recycling to quickly and intuitively participate in the new system, due to the fact that the most prominent barriers for individuals to start recycling other than the lack of facilities are the lack of convenience and the process itself consuming too much time.

## **1.2 Project Solution Approach**

The project’s approach to solving this problem is creating a machine learning model that can sort between recyclables and non-recyclables; furthermore, if the object is recyclable, the model will then state which class of object it belongs in. Although the scope of the project has been set to objects that are commonly used and of a smaller size, home appliances and larger items will not be included within the dataset. The final iteration of the project will have the user interact by inputting an image into the model through a mobile application interface. The results will help lower the barrier of entry for recycling as the application will increase the convenience of recycling and decrease the time taken to sort the object correctly. The pictures input by mobile users can then be further analyzed to improve the model’s accuracy or developed to store and sort the classes, which can be used to tune and optimize the recycling process.

## **1.3 Project Objectives**

This project aims to create a supervised learning model that is able to predict the make-up of the material of an object in the input picture to a satisfactory degree of accuracy. The completed model will be able to discern between recyclables and non-recyclables. The recyclables will then be sub-categorized into four different groups: glass, metal, plastic, and paper, while the non-recyclables will be separated into six different classes: ceramics, kitchenware, light bulb, photographs, Styrofoam, wood. The final version of this project will be made in the form of a mobile application to increase the availability and convenience of the product.

# **Chapter 2**

**Background**

## **2.1 Fundamental Theory and Concepts**

### **2.1.1 Digital Image Structures**

A pixel is the smallest subset within a picture that stores a value associated with its color saturation, which means a digital image is just a group of pixels. How many pixels are present in an individual image determines its quality, meaning the higher the density of pixels in the picture, the higher the quality; this concept is called resolution. A higher resolution presents more detail, but on the other hand, it also increases the time it takes to process the image and the amount of disk or memory space required to store it.

Digital images can be classified into grayscale and color images. In every picture, the value representing a color's intensity is denoted from 0 -255 because images usually store their color information in 8-bit numbers. A channel is a grayscale image of the same size as the original image, which can be associated with one of the primary colors. A grayscale image, therefore, will have pixels that range from 0 being black to 255 being white, and the picture itself will have only one channel. In comparison, color pictures will have three channels corresponding with red, green, and blue, which are then overlaid on top of each other.

[1]

Original Image

****

****

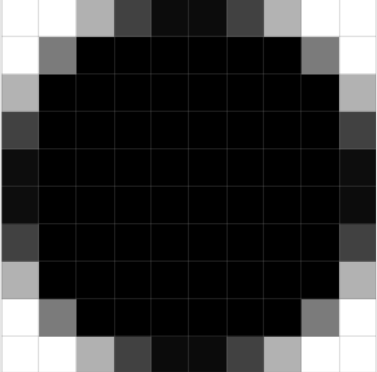
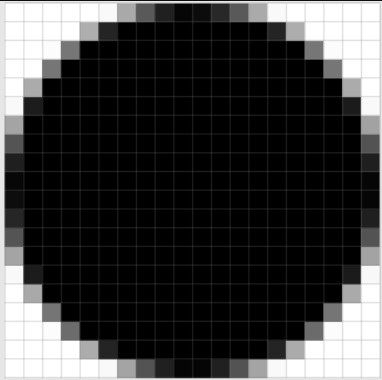
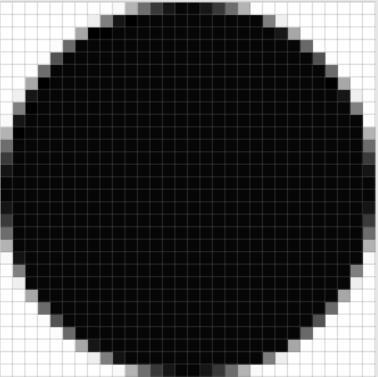
Red Green Blue

### **2.1.2 Image Operations**

**Resizing**

The resizing process is the practice of either increasing or decreasing the number of pixels inside a given image. There are two ways to change the image using this method. The first is upscaling, where the image’s resolution is increased, while the other, downscaling, decreases the targeted image’s resolution. Keep in mind that increasing the pixel count of an image will not directly correlate to an increase in quality if the starting quality of the image is subpar since the image will go through a process called nearest neighbor resampling. Nearest neighbor resampling occurs either when the base image is upscaled or downscaled and imbues the colors of surrounding pixels into a newly formed pixel. This may happen when the image is upscaled and new pixels are formed or downscaled, where already existing pixels are merged.

[4]



(10 x 10px)

(20 x 20px)

(30 x 30px)

**Grayscaling**

Grayscaling is the process where a colored image is converted into a picture with only one color channel, turning the picture black and white. There are generally two main approaches to turning a picture black and white, where the first is to average out all the RGB values, while the second places unique weights on each of the values. The first is called the Average Method, which as the name implies, takes and averages out the three color values, giving each of them the same level of importance. While the second, called the Luminosity Method, also takes an average of the three colors and considers how perceptible each color is. This gives each color a weight associated with it, giving a more accurate translation into grayscale.

[5]

I = 0.33∗R + 0.33∗G + 0.33∗B

Average Method

I = 0.299∗R + 0.587∗G + 0.114∗B

Luminosity Method

****

[6]

Base image

****

Average method

Luminosity Method

### **2.1.3 Artificial Neural Network [ANN] or Forward Feed Neural Network**

Artificial neural networks get their name from how it works by making minor adjustments to the various interconnected neurons inside their system; this web of neurons gives the system its name as it looks like an artificial brain. ANNs are usually used in large projects with a large amount of data to parse through since an ANNs excels at pattern recognition and ignoring minor blemishes in the dataset. The structure of an ANN consists of an input layer, multiple hidden layers, and an output layer where the input data will flow through these layers with respect to their order. This is why ANNs are sometimes called Forward Feed Neural Networks since the data can only move in one direction. While there is always only one input and output layer, the researcher can set the number of hidden layers and is not usually assigned to any particular number beforehand. Due to its architecture, artificial neural networks can process any data as long as it can be made into a numerical format. When used with image data, the hidden layers detect specific characteristics of the picture itself; while one may detect lines, others may detect other traits like shadows. The neurons are then bound together by a web of connections, each of which has an associated numeric weight. Because neurons in a layer are usually only connected to their immediate neighbor in an adjacent layer, the input and output layer are never in direct contact with each other.

Input from the user is first received by the input layer, after which it is weighted and sent off to the first hidden layer. This process of weighting the input before passing it onto the subsequent layer repeats until the transformed input arrives at the output layer as the model’s prediction. It should be noted that the number of input and output neurons does not need to be the same. The user will feed into the model as input parameters represent the different inputs, while the output parameters are just the number of possible results given by the user.

### **2.1.4 Backpropagation**

Loss is another word for error in an ANN prediction. Backpropagation is a process in which the loss is calculated by starting from the output layer and returning layer by layer to the input layer. The loss is also determined by the model's predicted number when compared to the actual answer that the model was supposed to come to. If both of these numbers are different, then the value of loss will not have the value of zero and its magnitude will increase proportionally to how far the answer was from the target. It will then be further used to correct the weights and biases of each layer so that the model may be a little bit more accurate in the next iteration and produce a smaller amount of loss.

### **2.1.5 Transfer Learning**

The premise of transfer learning revolves around re-purposing a model pre-trained for one task for another similar task. Since some of the pattern recognition capability needed for the completion of the job is already there, this means that the model won’t need to spend as much time nor as many resources in re-learning concepts. This usually ends with the transfer learning-based model using fewer resources while still producing a competitive solution within a short time. This type of learning is common within pre-trained models used for recognition or classification since using a model already proficient at extracting features will lessen the time needed for a satisfactory set of weights to present itself immensely.

### **2.1.6 Convolutional Neural Networks**

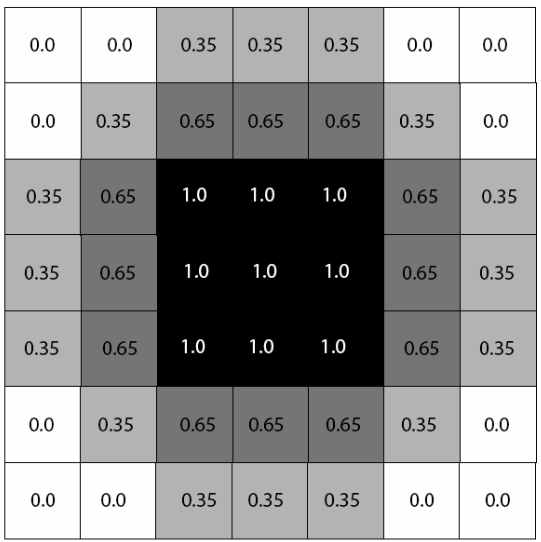
**Overview**

CNNs are multilayer neural networks that excel in image recognition tasks. Due to their function, CNNs have a unique and sometimes daunting structure. They are made up of a cycle of three different types of layers: convolution, pooling, and ReLU layers, after which the input is processed through either one or multiple fully connected layers. Convolutional layers represent the core of the extraction process in every CNN. Features from the images given to the model are extracted at this layer. After which, the features are stored by compressing it using either max or min pooling, then the matrix will then go through a ReLU layer which will remove all the non-positive numbers from the matrix. The output is then passed into the fully connected layers, compiling it and then mapping it with the corresponding output variable. Like classical ANNs, CNNs are equipped with backpropagation, making it possible for the model to improve constantly upon itself from its previous predictions.

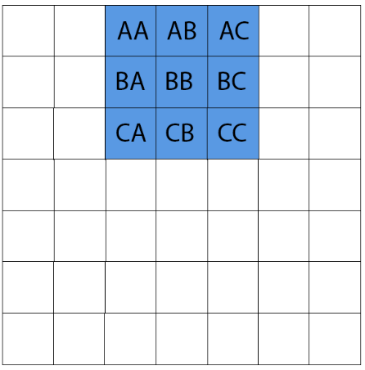
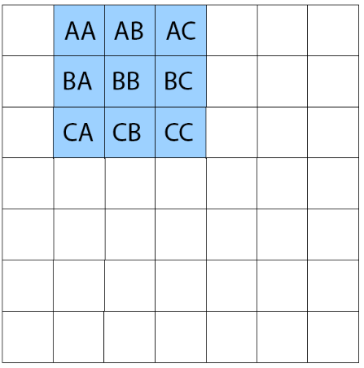
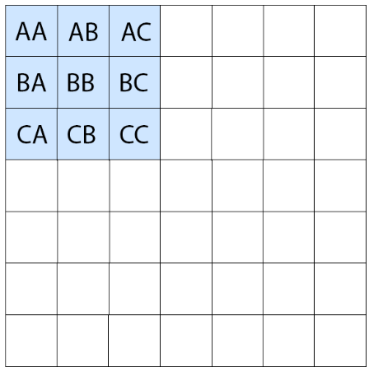
**Convolution layer**

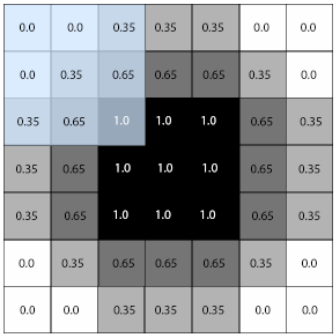
The convolutional layer derives its name from the process it preforms, that being convolutions, which is the process of creating an output by merging a filter with an input matrix. A filter, also called a kernel, is a matrix of fixed size that is often found in convolutional layers as a way to detect the image's features; Each cell in the filter matrix will have a value. The pattern of these values will determine what the kernel will detect, whether that be corners, shadows, edges, or lines. In a typical CNN, the kernel elements will start as random values. As the network learns, the kernel will adapt to extract a specific feature type; this filter will take the form of a value matrix which will be overlaid onto the group of pixels, while the input will be a matrix of pixels from the image of the same size. Kernel sizes are usually much smaller than the images they process, meaning that the image will be handled in batches rather than all at once.

For a 7x7 image to be processed by a 3x3 kernel, the convolutional layer must segment the picture into 9-pixel batches, corresponding to the size of the kernel. After which the kernel is overlaid onto each of these batches which creates an output pixel; the equation for said output is the sum of the filter’s value at that pixel multiplied by the value of the underlying pixel for each pixel in the batch. The process will then repeat until the convolution reaches the end of the image and no further pixels are found, and the output will be passed onto the next layer in the form of a 5x5 matrix.





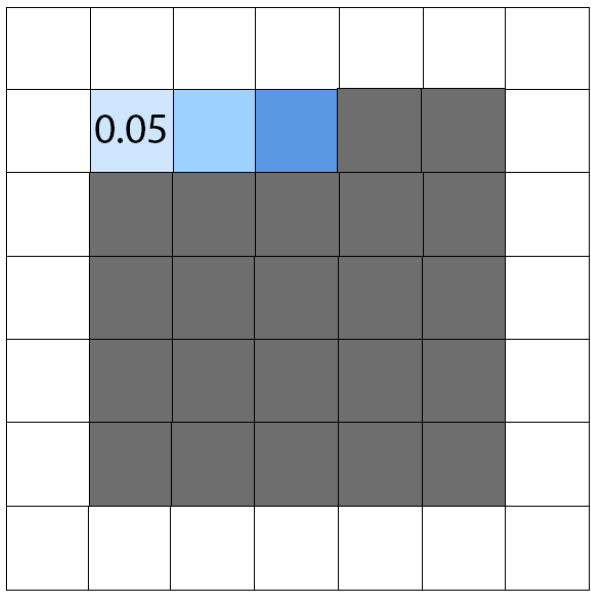
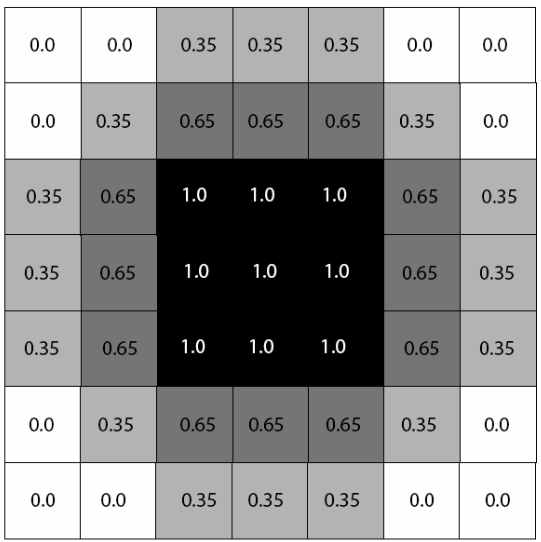




Output Pixel = 1.0[AA] – 1.0[AB] + 0.0[AC] + 1.0[BA] – 1.0[BB] + 0.0[BC] + 1.0[CA] – 1.0[CB] + 0.0[CC]

Output Pixel = 1.0[0.0] – 1.0[0.0] + 0.0[0.35] + 1.0[0.0] – 1.0[0.35] + 0.0[0.65] + 1.0[0.35] – 1.0[0.65] + 0.0[1.0]

Output Pixel = 0.05



**Pooling layer**

The concept of pooling revolves around compressing an image into a specific size while retaining important information. Out of the three main types of pooling, which include max-pooling, min-pooling, and avg-pooling, the one that is the most commonly used is max-pooling. Max-pooling gets its name from the fact that when the group of pixels are compressed into one, the resulting pixel will have the value of the highest value pixel within the previous group. Min-pooling operates on the same concept, but instead of being the maximum value, it is the minimum, and the average pool averages out each pixel’s value. The process is used to stabilize the position of the feature or object inside the image, as without the pooling layer, slight altercations including shifting may produce a different feature map.

**ReLU layer**

The ReLU layer works to clean up the current image; it does this by selecting every negative number and setting it equal to zero, while positive numbers stay as they were. The process is done to refine the output of the previous layer, usually a convolutional layer, as a negative number means that it has low relevance towards the identification of a targeted feature. Because of this ReLU layers are often stitched together with convolutional layers, as the ReLU layer will be able to catch any negative values before it enters into the pooling layers.

**Fully connected layer**

The output from the pooling layer will then arrive in the fully connected layer as a 1-D vector. Due to the nature of the layer being fully connected as in the neurons are connected to every weight from the previous layer, the output that is distilled from these layers will have taken into account every element from the entire picture taken into account and not just a portion of it.

**Evaluation Metrics**

**Confusion Matrix**

A confusion matrix is a table of size N x N (N being the number of classes we have within the dataset), which is used to measure the performance of our model. In this table, each cell will have its own label, which can fall into one of four categories: true positives, false positives, false negatives, and true negatives.

True Positive [TP]: When the items inside the targeted class are identified properly.

False Positive [FP]: When the items from other classes are mistakenly identified as ones from the target class.

False negative [FN]: When the item from the targeted class gets mistakenly identified as one from another.

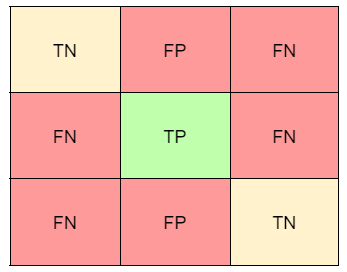
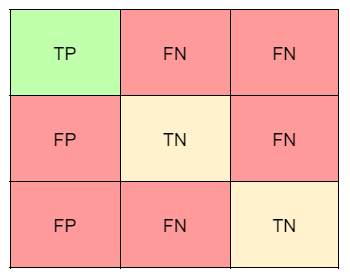
True Negative [TN]: When items outside the targeted class are identified correctly

Predicted

Predicted

0 1 2

0 1 2



0

1

2

0

1

2

Actual

Actual

**Precision Score**

In this case, the precision score is an evaluation metric which measures the probability that a result from another class will not be classified as the targeted class; this metric is therefore found by dividing the true positives with the sum of the true and false positives. Similarly to the other two metrics listed below, the value of this score ranges from 0 to 1 where 1 is the best and 0 the worst.

Precision Score =

TP

TP + FP

**Recall Score**

The recall score calculates the percentage chance that the items inside the targeted class will be correctly identified. Even though the equation superficially looks identical to the one used to calculate the precision score, the equation sets the denominator as the sum of the false negatives and the true positives.

Recall Score =

TP

TP + FP

**F1 Score**

A combination of the two metrics listed above, the F1 score is the harmonic mean of the precision and recall score or the average of the two rates; being the average makes the score less susceptible to variations such as an imbalanced dataset.

F1 Score =

2 \* Precision \* Recall

Precision + Recall

### **2.1.7 Web Scraping**

Web scraping is the process of combing through the internet with a script to find wanted information; the script used can either be a user-made or a library or an extension that another person created. When using a library, a request first has to be made to access the targeted site, usually in the form of a http link; it then extracts the HTML and any other underlying source code and returns it in an HTML format, which the user can then parse through using commands to retrieve any relevant or desirable information. Screen-scraping libraries like Beautiful Soup have always been popular for their ease of use and uncomplicated syntax. Still, extensions also exist specifically for scraping pictures, including ones like the Firefox extension DownThemAll, which provides the user with an interface; This allows the user to scrape items like images and links without needing to get tangled in coding a script for such purposes. Although convenient, there is a contention that the act of web scraping strays into the gray areas of legality since web scraping does not differentiate between public and copyrighted material, caution must be taken to receive data that is not only usable but ethical as well.

## **2.2 Technologies**

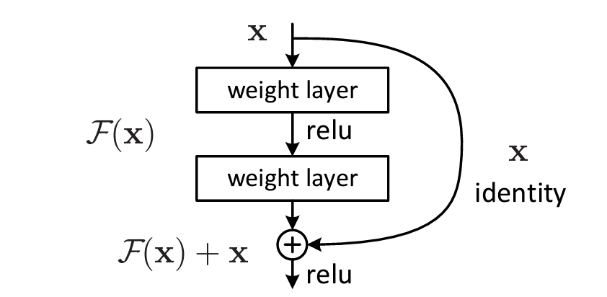
### **2.2.1 PyTorch**

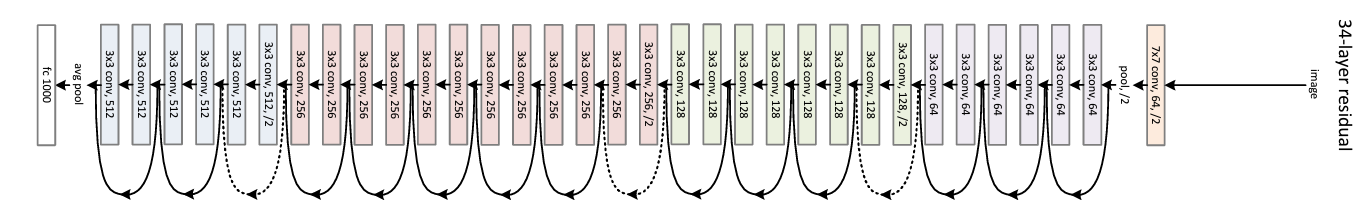
PyTorch is an open-source machine learning framework developed initially as a part of MetaAI by Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan [10]. The framework was initially released to the public on September 2016. Its ease of use and readability compared to TensorFlow made it a solid choice for this project, along with the fact that it gave the project access to models such as ResNet50.

### **2.2.2 Resnet**

Developed by a team of researchers from Microsoft in 2016, Residual Neural Networks or Resnet was created as an alternative model for recognizing images [9]. The model structurally is similar to other Convolutional Neural Networks and contains many of the same layers like convolution, pooling, and fully connected layers. However, unlike other models, Resnet’s structure is grouped into residual blocks; these blocks are made up of two convolutional layers, where a connection point is made, which bypasses both layers. This “direct connection” differentiates Resnet from other CNN models as it gives the structure an alternative route to travel through the system, which can help alleviate an issue experienced by CNNs with a large number of layers called vanishing gradients; this happens when the model discards a distinguishing feature due to the over-processing of an image caused by constructing a model with too many convolutional layers. Resnet versions are usually tied to the number of layers inside that particular version, so Resnet50 will have 50 layers, making it easy to keep track of the different versions continuity-wise. Through testing by feeding the Resnet50 model 1.28 million images differentiated into 1000 different classes, the top-1-error of the model was calculated out to be 22.85% placing the model higher than established models such as VGG-16 and PReLU-net which scored 28.07% and 24.27% respectively [9]. The top-1-error being a metric which calculates the probability that the model does not assign the highest score to the correct class.

[9]





### **2.2.3 Google Lens**

Google Lens is an image recognition platform developed by Google that uses a machine learning-based neural network for parsing visual data [11]. The program was launched in 2017 during Google’s annual I/O fair with a wide range of capabilities; one relevant to the project is the image search function. An image can be placed into the engine; subsequently, a group of pictures similar to the one given will be produced and presented to the user. Since the images curated are based on an original image, there is a significant decrease in the number of pictures with unsatisfactory characteristics: stark white background, multiple items, etc. This has made the procurement of a dataset of decent size with no compromise to quality much less time-consuming and more streamlined.

### **2.2.4 DownThemAll**

The browser extension DownThemAll is a downloadable add-on in the web browser FireFox used specifically to download all the links from a particular page. It was developed by Federico Parodi, Stefano Verna, and Nils Maier and is used in this project to scrape pictures from the Google Lens page quickly [12]. Although we first thought of using the traditional method of scraping a website by using Beautiful Soup 4, the technique fell through when calls made to a Google-Lens URL would default back to the landing page. This open-sourced product, along with Google-Lens, allowed the project to construct appropriate databases quickly and methodically since every picture can be expanded into 50 others, and each of those can be expanded as well.

### **2.2.5 Apex**

Apex is a supercomputer and storage platform designed explicitly for processing intensive activities like training a complex machine learning model. It is currently hosted at CMKL University itself and contains Nvidia DGX A100 48x GPUs, which have a high amount of processing power, and a Tensor core GPU which is more optimized for training when compared to a Cuda core GPU which is standard on most laptops. Suppose multiple models are assigned to run at once instead of splitting the processing power of one GPU to run several models, which would happen if a laptop is used. Each of the models will receive its own GPU core, drastically decreasing the time needed to train multiple models. Apex’s processing power was used to consecutively train variations of our Resnet model, which made it possible for us to experiment and make minute adjustments to our training variables, as it shortened the training time needed to complete all the training cycles significantly.

### **2.2.6 Keras**

Keras is a library written in Python as an interface for TensorFlow, aiming to provide a more user-friendly option for users looking to construct a deep learning model. It was released in early 2015 as an open-source project [12]. The original creator named Francois Chollet wanted the library to contain all the functionalities of TensorFlow, such as the ability to construct a new model and continuously add onto the previously constructed model, but without the hard-to-understand syntax of the original library. The previously mentioned function of adding on additional layers not only covers the basic layers like ReLU but also layers geared more towards utilities like normalizations and dropouts.

### **2.2.7 SciKitLearn**

SciKitLearn is a Python library initially developed by David Cournapeau, who was a part of Google at the time. It contains a multitude of features like unsupervised learning models, low-level supervised learning models, data visualization, and evaluation metrics [13]. Furthermore, this model is built upon multiple other libraries, such as matplotlib and pandas, which make up the core of some of SciKitLearn’s functionalities. First developed in 2007, it was released to the public in early 2010 and remained one of the most popular machine-learning libraries for Python, with regular updates still being released.

## **2. 3 Related Research**

### **2.2.1 The use of machine learning methods to classify images of trash**

A paper released by Stanford University showed that it is possible to use machine learning models to classify images of trash by employing two different methods with varying degrees of success [7]. Besides the “trash” group, which has 100 pictures, every other class has between 400 to 500 photos, making the dataset approximately 2,400 images. Furthermore, all images used had a stark white background, but their lighting and orientation were varied to introduce some diversity into the dataset. The first approach was using an SVM model, which plots the image onto a graph based on its most notable features, after which a hyperplane is derived. The hyperplane is an algorithm that finds a line that gives the most separation between the plotted classes. In comparison, the second approach used a CNN, which had a similar structure to AlexNet, containing a total of 11 layers: 5 convolutional layers, three pooling layers, and three fully connected layers. Another difference between the CNN and SVM models was that with the CNN model, the trash class was removed entirely since it only contained 100 images which didn’t give the model enough data and disrupted the accuracy of the model as a whole.

The researchers found that the SVM model produced accuracy rates of around 63%, although they also speculated that the model’s simplicity might have caused this relatively low accuracy. However, the CNN model only achieved results of 22%, with the loss value still fluctuating wildly, meaning that there is either an issue with the dataset or the model’s parameters. The report’s conclusion shows that it is possible to train a machine learning model to recognize and classify images of trash, although parameters and datasets must be chosen carefully. This is to avoid the possibility that the model will become too restricted by the model and end up not learning features that are relevant in classifying the images. [7]

### **2.2.2 Using Resnet50 to classify images with similar qualities**

Three researchers from Turkey published a report that displayed the ability of ResNet, specifically ResNet50, to recognize and classify chest x-rays which contain signs of pneumonia. The report documented the accuracy and loss of various models: Alexnet, Googlenet, Densenet201, etc. Furthermore, the team also constructed a modified version of the Resnet model, which added five additional layers, one of which was a dropout layer. A dropout layer is a layer whose sole job is to prevent the model from becoming overly attached to the training dataset by selecting a neuron periodically and making the value of that neuron zero which will cascade down the network.

A library from Kaggle was used, which contains approximately 5,000 images at a roughly 3 to 1 ratio of pneumonia images to pneumonia-free images. The results found that every model successfully classified the infection; they all achieved at least a 90% accuracy identification rate in their respective tests. The model with the highest accuracy is the modified version of Resnet, surpassing the original by roughly 3%. The conclusion above shows that Resnet is capable of classifying images with similar qualities and can find a significant amount of success in the topic. [8]

# **Chapter 3**

**Methodology**

This chapter will be discussing about the development process of the final model, with Figure 3.1 showing the main stages of the project being dataset collection, parameter setting, and model comparison. We will also go into details including the acquisition methods for the dataset, what we changed for each of the model versions, and the structure for both Keras and Resnet50.

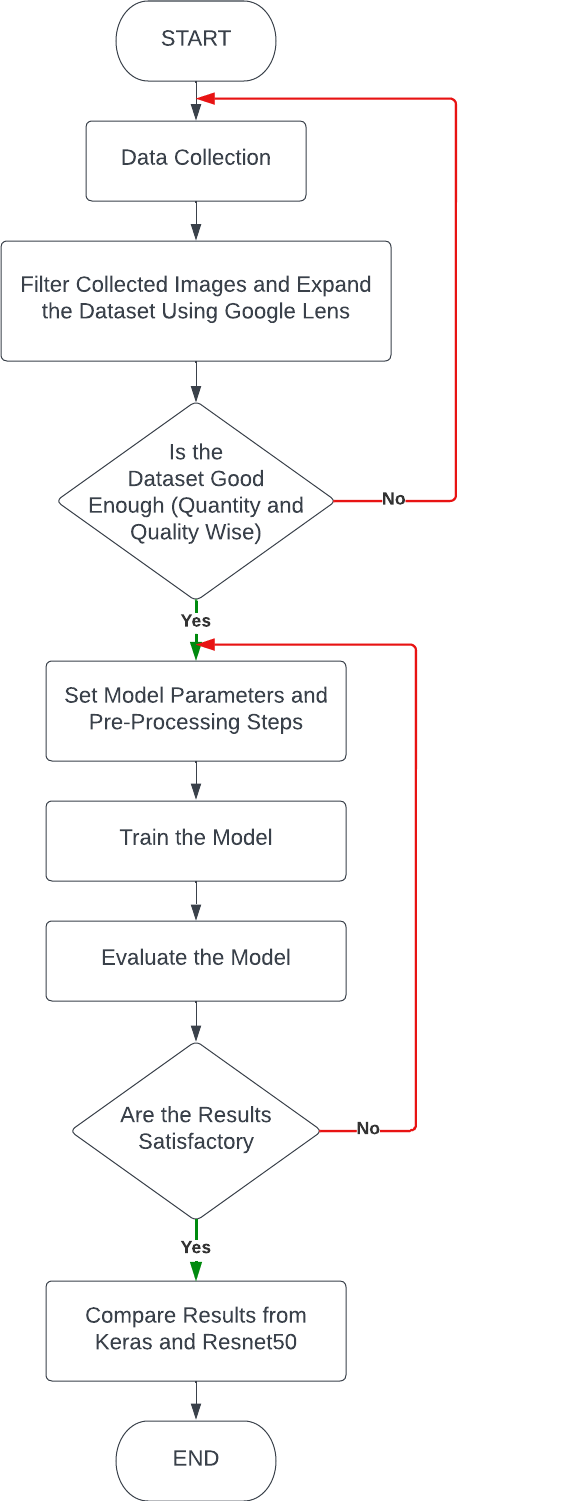


Figure 3.1 Overview of the working progress

## **3.1 Dataset Collection**

In the process of data collection, we first tried to find a dataset deposit from Kaggle that we could adapt to fit our current model. Sadly it was not possible as the dataset found contained features that did not match the user input; this is because the images from the Kaggle dataset had white backgrounds, which would not be the case in a real-life scenario. Therefore, we needed to construct our dataset from scratch, which took some time to compile. Our first choice was to create a form where students could submit pictures, but the dataset acquired proved to be incomplete; this is because the dataset retrieved only contained around 200 images over the entire dataset. Due to this setback, the dataset had to be expanded by using Google Lens and passing the uploaded photos from the dataset into it to create the first official version of the dataset.

The second version is just a filtered down version of the first, but since the second dataset’s images mostly contained pictures of recyclable objects, we had to create another version with non-recyclable objects; this version would from now will be referred to as the third version. The fourth is a revised version of the third, which saw the removal of some of the classes from the non-recyclable section because the removed section did not fit into the project’s scope. While some were discarded from the dataset entirely, others were merged due to their material make-up, an example would be the combination of the classes of Styrofoam and packing peanuts. Lastly is the fifth version, which is again built upon the third version and not the fourth because of an issue detected within the fourth version after testing; the issue being after combining some of the classes, specifically Styroform and packing peanuts, the model began to regress in accuracy.

## **3.2 Structures for the Models**

At the start, the group debated on which model to use as the base version; the first suggestion was to construct a model from the ground up, while the second was to import a model and use transfer learning to shorten the amount of data needed. After more research, the second option was picked after taking our available non-recyclable training data into account, which accumulated to us choosing Resnet as our base with a Keras model made from scratch for comparison. The comparison was made to experiment with how much improvement the transfer learning model has over the traditional model. Resnet, specifically Resnet 50, was chosen for this project because the project’s scope aligns with the model’s objective since Resnet’s model was made to recognize and classify images specifically. Even though the accuracy values for the models were all around 91 – 95%, the reasoning for choosing Resnet 50 over other versions like Resnet 101 or 152 came down to the diminishing returns we received from the latter two versions. While Resnet18’s accuracy was around the same as the other three versions, moving to Resnet50 stabilized the learning line over the epochs significantly, translating to an increase in accuracy. On the other hand, further changes to Resnet 101 and 152 produced much of the same result as the stability issue was already fixed; the only difference experienced by moving the model onto Resnet 101 and 152 is a marginal increase in its accuracy. Since updating to a version with more layers also increases computational resources and time, the extra resources needed are not justifiable when compared to the benefit brought by the alteration itself.

Throughout the different iterations of Resnet, preprocessing methods are added to the model to safeguard against variations a real photo may possess. The first processing technique applies random rotation to the image, which counteracts the possibility that photos taken may come in sometimes unconventional angles. Jitter acts similarly, but it adjusts the brightness instead of the angle, simulating the various light conditions in which a picture can be taken.

In the case of Keras, the structure was constructed without a base model, with means that its structure will consist of layers expected in a convolutional network: convolution, ReLU, pooling, and fully connected layers. Specifically, the final model contains eight convolutional layers, each with a ReLU layer attached, four dropout layers, and two dense layers. The dropout layers ensure that the model will be less prone to over-fitting by setting the output of its layer to zero in intervals, making it more difficult for the model to tailor itself to the training dataset; these dropout layers are placed after the convolutional and ReLU layers, and also after each of the dense layers. On the other hand, a dense layer is a layer where its neurons are connected to every node of the input and output layer, making it more of a transitional layer used to specify the following layer's input size.

Both models use the stochastic gradient descent (SDG) optimizer function, an equation the model uses to guide the weight adjustment. Optimizer functions in general act as a force of influence on the learning progress by manipulating the weight’s rate of change according to the previously calculated loss. The reasoning for why we chose to use a SDG instead of a normal gradient descent function is that the standard method updates the parameters every training cycle, but the SDG optimizer updates itself every epoch. In other words, an SDG optimizer can adjust itself from just one data point, whereas the usual method will not modify its value until it has run through the entire training set; this gives the SDG optimizer a better chance of arriving at a suitable set of weights.

Equally important is the loss function, which in this case is cross-entropy; this loss function is used across both models to measure the error in the model’s predictions. However, the cross-entropy method differs from standard entropy in terms that it does not only take into account the ratio of images between classes in a dataset but also how far the predicted result is from the actual target.

Although the models for both Keras and Resnet display accuracy values for each of their iterations, the metric could be somewhat misleading as the equation used to find the accuracy value is too broad. In order to find a more accurate representation of the model's success, other metrics must be used in tandem with accuracy; therefore, three other metrics will be used to evaluate the different versions: precision, recall, and f1 scores. These three scores will be able to determine if there are inconsistencies within the prediction model when used together with the confusion matrix. As for the training environment, the Resnet model is trained on the Apex system which uses a Nvidia DGX A100, while the Keras model is trained by a NVIDIA GTX 1060; the Keras model is not run on the Apex system due to the version difference between the two system’s versions of Tensorflow.

## **3.3 Experimentation and Parameters for each of the Versions**

### **3.3.1 Background**

The datasets below were used in both the Resnet and Keras model, with each of the categories receiving an 80-20 split between the training and validation sets; this split is consistent between the two models. Furthermore, the status of filtered implies that the dataset was partially refined, meaning that images that did not fit the scope of the project or those that do not fit the criteria are removed; the criteria being no solid white backgrounds and the majority of the object must be in the image.

### **3.3.2 Dataset Versions**

**Version 1**

Size: 3,287

Filtered: No

Classes: [Plastic (1,025), Glass (735), Metal (1070), Paper (457)]

**Version 2**

Size: 2,539

Filtered: Yes

Classes: [Plastic (674), Glass (538), Metal (886), Paper (441)]

**Version 3**

Size: 4,133

Filtered: No

Classes: [Plastic (674), Glass (538), Metal (886), Paper (441), Ceramics (195), Kitchenware (685), Light Bulb (177), Packing Peanuts (99), Plastic Toys (162), Styrofoam (148), Wood (128)

**Version 4**

Size: 3,660

Filtered: Yes

Classes: [Plastic (674), Glass (538), Metal (886), Paper (441), Ceramics (175), Kitchenware (301) Light Bulb (177), Photographs (87), Styrofoam (253), Wood (128)]

**Version 5**

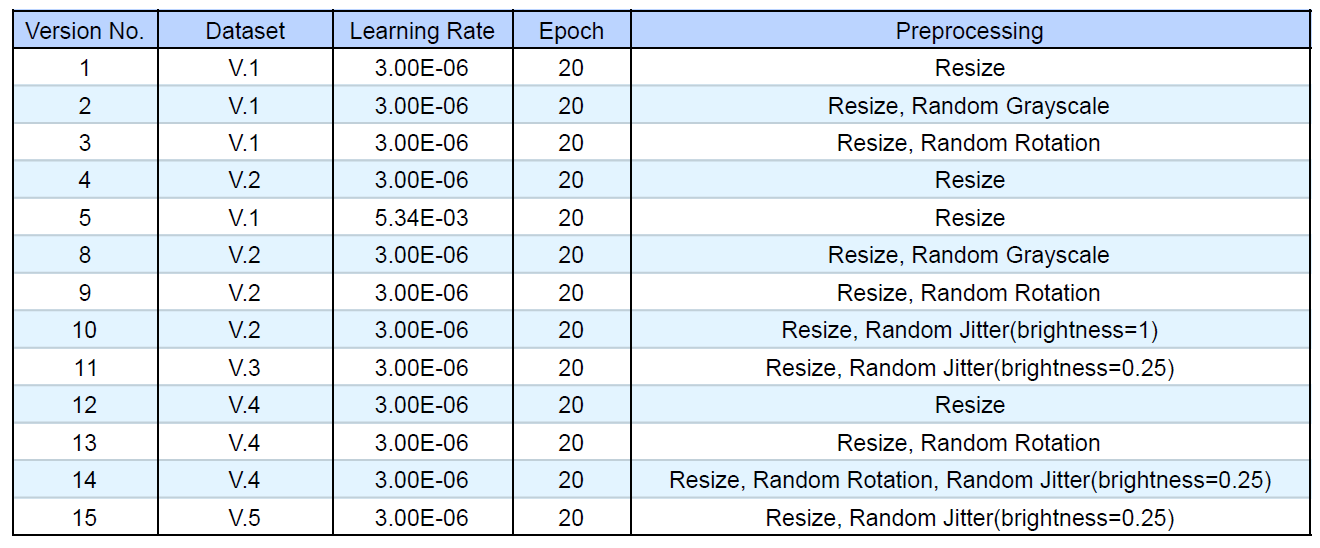
Size:

Filtered: Yes

Classes: [ Ceramics (175), Glass (538), Kitchenware (390), Light Bulb (178), Metal (887), Paper (445), Photographs (87), Plastic (681), Styrofoam (168), Wood (129)]

### **3.3.3 Version History**

**Resnet**



Over the list of versions, the constant variables include only the epochs and learning rate; however, the learning rate was briefly changed during the fifth version to test its effects on the model. The change was done in an attempt to find a more suitable learning rate, but eventually the new rate had to be discarded as it decreased the model’s evaluation metrics substantially: Accuracy, precision, and recall. Other than that, the versions are just a systematic addition and subtraction of features to find one that can comfortably identify the given classes at a satisfactory accuracy rate.

**Keras**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version No. | Dataset | Learning Rate | Epoch | Preprocessing |
| 1 | V.4 | 0.03 | 100 | Resize |
| 2 | V.5 | 0.03 | 200 | Resize, Random Contrast, Random Shear |

Due to Keras being built with no underlying structure, many versions produced results that caused a negligible change to the evaluation metrics. Therefore, two versions that showed the most promise are displayed, with the second being the final version. Both of the structure’s layers are the same, with preprocessing techniques applied to the second version.

# **Chapter 4**

**Results**

There are currently a few concerns regarding the dataset's quality, mainly geared toward new classes added in version three since more time will be needed to refine the dataset. Figure 4.1 will show the comparison between version 15 and other similar versions, while Figure 4.2 shows the confusion matrix along with other evaluation means of version 15 of the Resnet model. From the data presented, strong results show that machine learning, especially residual neural networks, can be used in the systematic classification of discarded objects. The reason for selecting version 15 and not 12 is because even though the accuracy for the latter is higher, it struggles significantly against external testing data taken in sub-optimal lighting. When the image taken isn’t in good lighting, we observed that the model becomes unable to differentiate between certain classes including paper and plastic. Currently, the hypothesis is that the lower brightness causes normally semi-transparent objects to become more opaque, which causes the aforementioned issue.

Results from the second Keras model displayed in Figure 3 are similar to that of Resnet’s, although to a lesser degree because of its less complex structure. Interestingly, Resnet holds the highest overall accuracy among the two models, while Keras has the least variance between the four-evaluation metrics, as seen in Figure 4.

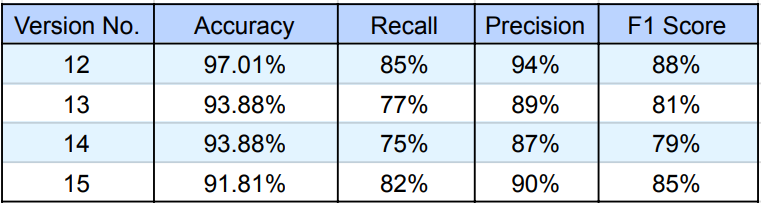
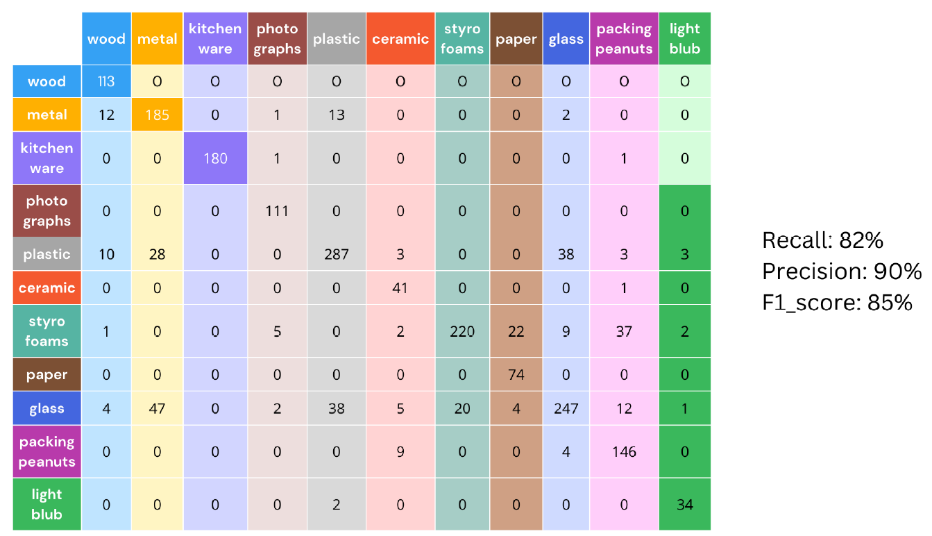


Figure 4.1 Resnet Evaluation Metrics



Actual

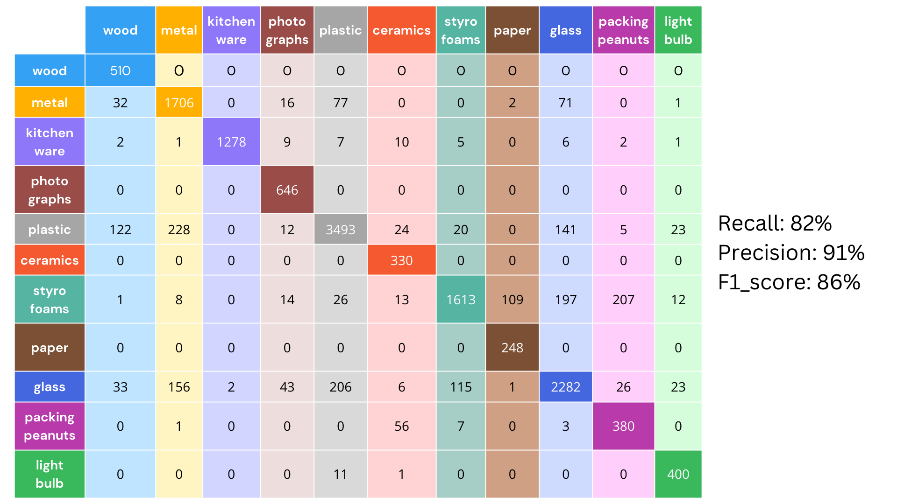




Figure 4.2 Resnet Version 15 Confusion Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version No. | Accuracy | Recall | Precision | F1 Score |
| 1 | 64.84% | 65% | 66% | 64% |
| 2 | 67.70% | 68% | 68% | 67% |

Figure 4.3 Keras Evaluation Metrics

Actual

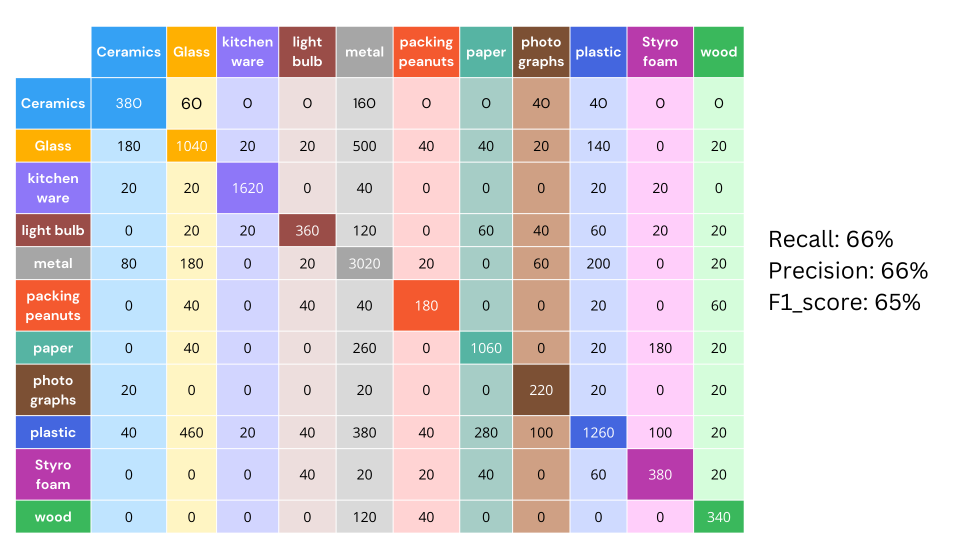




Figure 4.4 Keras Confusion Matrix

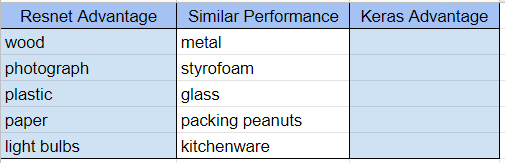


Figure 4.5 Resnet & Keras Comparison 3

When comparing the confusion matrixes of Figures 4.2 and 4.4, Resnet performed better overall, which is to be expected. There are multiple classes, such as paper and light bulbs, where the Resnet is already able to sort the categories with a high level of accuracy, while the Keras model is still struggling to classifying the same classes. However, there are also classes that both models had issues with, these being metal, Styrofoam, and packing peanuts; this is due in some part to the dataset itself. While the only category that both models were able to identify comfortably is kitchenware, with each of the models’ rate of accuracy being more than 90%.

Overall, the confusion matrixes make it clear that the still un-optimized dataset is causing issues with the model’s learning ability since certain classes have a high percentage of misclassifications, like foam and paper. However, this problem occurs in some categories and not others, suggesting either an imbalance within the dataset or a lack of identifying features between some of the classes. One instance of the latter is the problem of misclassification between paper and Styrofoam, since both of these classes have similar features: white shade, completely opaque, and mostly smooth. Furthermore, most of the distinguishing feature which separate these two classes is their texture, which is not fully captured as down-sampling has to be completed before the picture can be processed by the model.

# **Chapter 5**

**Conclusions**

## **5.1 Summary of Accomplishments**

We successfully constructed a Resnet model capable of sorting discarded items at an acceptable accuracy rate, with each testing metric returning a result of at least 80%. Furthermore, due to the version testing conducted, we could clearly attribute improvements in performance to refinements in the dataset. As almost every version that saw a sharp rise in accuracy or precision is one that received a new dataset, this can be observed in both versions 12 and 15 of the Resnet model. Changes relating to the set should be later implemented in a newer version of the model to increase its performance, both in accuracy and other evaluation metrics.

## **5.2 Issues and Obstacles**

The group set out to construct a model that can discern the difference between recyclable and non-recyclable materials while giving the user an output consisting of the class and recyclability of the material. On that front, it can be considered a success; although a satisfactory result was found, the way to the said solution was not straightforward. Several issues have been attached to the project; some would be solved relatively quickly, while others would remain unsolved until the last weeks of the semester. The first issue pertains to the flow of ideas and information between the group members and the broad responsibilities given to each member; this caused multiple members to work on the same issue while other matters were not attended to. A clear example of this is the dataset that was supposed to be the first topic finished within the timeline. Still, due to misunderstandings and unclear communication, the dataset took months longer to complete than it should have.

Secondly, we debated and created a form where our classmates could contribute to the database. Even though the strategy was successful initially, the flow of incoming pictures slowed and eventually just stopped entirely; this led us to use Google Lens to expand the dataset. However, even Google Lens had its own set of problems since the Beautiful Soup model would go to their landing page every time a Google Lens URL was produced. Beautiful Soup 4 being unusable pushed us towards downloading an extension called DownThemAll that would allow us to scrape the wanted images. Further touching on the subject of issues relating to the dataset, because the original dataset was so limited in quantity, we had to find Kaggle deposits that contained items missing from our dataset: kitchenware, lightbulbs, Styrofoam, etc. The addition of which caused further issues such as the decline of quality of the images, which impacted the ability for the model to distinguish previously sortable classes. The last significant problem encountered by the team was the issue of an inappropriate learning rate; this is due to the opinion that the program should be finding the optimal learning rate using LRFinder, which is imported from the Torch library. The method did not work, but we believed the issue came from introducing a new dataset at the time. Consequently, this oversight delayed the completion of the base model by around a week, but the group eventually found a more suitable learning rate.

## **5.3 Future Directions**

The project will likely be worked on further in the second term to fully flesh out the model with a UX/UI interface. Furthermore, a feature that allows the pictures given by the end user to be added to the model’s dataset has been planned and will be attempted within the next semester; for this to be possible, our team will have to create a full-fledged application to host the model. The new feature would help in resolving the imbalance in our current dataset and help increase its overall quality. In addition, a platform will have to be chosen as a base since the process will involve constructing front and back-end elements that support the saved Torch model. The decision has to be made because the different operating systems, such as Android and Apple, use different languages in the development of their applications and have their own set of requirements to complete to earn the eligibility to publish the app.

## **5.4 Lessons Learned**

Through the progression of this project, we have experienced our fair share of successes and failures, but through all of it, we have learned many things. These lessons ranged from technical aspects, like machine learning concepts, to more interpersonal aspects, such as the importance of group communication. As this was the first time our group members had an extended interaction with the concepts and inner workings of systems such as CNNs and RNNs, we had to learn many different foreign concepts, from basic vocabulary, such as what an epoch was, to understanding more complex systems, like how was the structure of RNNs different from CNNs. Furthermore, there were also the various libraries and pre-trained models which we had to research. Although we were able to absorb these concepts with time passively, what needed active attention from us was the communications aspect of the project; this was the first time many of us had worked with the same group for an extended period.

Group meetings were at times more important than the coding itself or sometimes even more so, as we learned that without these meetings, there was no real sense of progress and direction. So we had to learn how to explain concepts, progress, and meeting times briefly but also clearly since we have lost time throughout this project to misunderstandings in the segmentation of tasks and other areas. It was also shocking to discover the existence of pre-trained models, especially ones for CNNs and RNNs, as everyone went into the project thinking we would have to construct a model from the ground up. Lastly, we found that pinpointing an optimized learning rate was a deciding factor for whether or not the model could produce any results. This is because we used a pre-set learning rate during our earlier versions, which caused our model's accuracy to hover around 60%, which only increased after we manipulated the learning rate value. The issues experienced regarding the size and quality of the dataset also made it clear that we should have had a more active role in its acquisition, since it was much more difficult to produce a coherent model without an acceptable dataset.

# **References**

1. Patterson, S. (2020) RGB and color channels in Photoshop explained, Photoshop Essentials. Available at: https://www.photoshopessentials.com/essentials/rgb/ (Accessed: October 19, 2022).
2. Chanthamas, Y. (2021) Disparity Worsens Ocean Pollution, Thailand Development Research Institute. TDRI Insight. Available at: https://tdri.or.th/en/2021/06/disparity-worsens-ocean-pollution/ (Accessed: October 20, 2022).
3. Saponara, S. and Elhanashi, A. (2022) “Impact of image resizing on deep learning detectors for training time and model performance,” Lecture Notes in Electrical Engineering, pp. 10–17. Available at: <https://doi.org/10.1007/978-3-030-95498-7_2>.
4. Inc., A. Images, Images - Foundations - Human Interface Guidelines - Design - Apple Developer. Available at: https://developer.apple.com/design/human-interface-guidelines/foundations/images/ (Accessed: November 22, 2022).
5. Güneş, A., Kalkan, H. and Durmuş, E. (2015) “Optimizing the color-to-grayscale conversion for Image Classification,” Signal, Image and Video Processing, 10(5), pp. 853–860. Available at: <https://doi.org/10.1007/s11760-015-0828-7>.
6. Antoniadis, P. (2022) How to convert an RGB image to a grayscale, Baeldung on Computer Science. Available at: https://www.baeldung.com/cs/convert-rgb-to-grayscale (Accessed: November 22, 2022).
7. Yang, M. and Thung, G. (2017) “Classification of Trash for Recyclability Status,” CS229 Project Report 2016 [Preprint]. Available at: https://cs229.stanford.edu/proj2016/report/ (Accessed: November 29, 2022).
8. Çınar, A., Yıldırım, M. and Eroğlu, Y. (2021) “Classification of pneumonia cell images using improved RESNET50 model,” Traitement du Signal, 38(1), pp. 165–173. Available at: <https://doi.org/10.18280/ts.380117>.
9. He, K. et al. (2015) Deep residual learning for image recognition, arXiv.org. Available at: https://arxiv.org/abs/1512.03385 (Accessed: December 3, 2022).
10. Warsaw, A.P.U.of et al. (2019) Pytorch: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Guide Proceedings. Available at: https://dl.acm.org/doi/10.5555/3454287.3455008 (Accessed: December 8, 2022).
11. Anushya , A. (2019) Google Lens as an image classifier - ijsrcsams.com, International Journal of Scientific Research in Computer Science Applications and Management Studies. Available at: https://www.ijsrcsams.com/images/stories/Past\_Issue\_Docs/ijsrcsamsv8i6p2.pdf (Accessed: December 7, 2022).
12. Maier, Federico Parodi and Stefano Verna, N., Parodi , F.M. and Verna, S. (no date) Downthemall, DownThemAll! Available at: https://www.downthemall.net/ (Accessed: December 8, 2022).
13. Chollet , F. (no date) Simple. flexible. powerful., Keras. Available at: https://keras.io/ (Accessed: December 8, 2022).
14. Hackeling, G. (2014) Mastering machine learning with scikit-learn apply effective learning algorithms to real-world problems using scikit-learn. Birmingham: Packt Publ.