



**AiCE Undergraduate Research Project  
Final Report**

**Spring 2023 Semester**

***Trash Master***

**Team Members**

**Inkaphol Siriwongwattana, Ink, [isiriwon@cmkl.ac.th](mailto: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***

**28/04/2023**

<b>Contents</b>	
<b>Contents</b> .....	<b>2</b>
<b>Chapter 1</b> .....	<b>4</b>
<b>1.1 Problem Statement</b> .....	<b>4</b>
<b>1.2 Project Solution Approach</b> .....	<b>4</b>
<b>1.3 Project Objectives</b> .....	<b>4</b>
<b>Chapter 2</b> .....	<b>5</b>
<b>2.1 Fundamental Theory and Concepts</b> .....	<b>5</b>
2.1.1 Digital Image Structures .....	5
2.1.2 Image Operations .....	6
2.1.3 Artificial Neural Network (ANN) or Forward Feed Neural Network.....	7
2.1.4 Backpropagation and Learning .....	8
2.1.5 Transfer Learning .....	8
2.1.6 Convolutional Neural Networks .....	8
2.1.7 Evaluation Metrics .....	12
2.1.8 Web Scraping .....	13
2.1.9 Native and Cross Platform Mobile Application .....	13
2.1.10 Backend as a Service .....	14
<b>2.2 Technologies</b> .....	<b>14</b>
2.2.1 PyTorch.....	14
2.2.2 ResNet.....	15
2.2.3 Google Lens.....	16
2.2.4 DownThemAll .....	16
2.2.5 Apex.....	17
2.2.6 Keras.....	17
2.2.7 SciKitLearn .....	17
2.2.8 Flutter/Dart .....	17
2.2.10 Firebase .....	18
2.2.11 TensorFlow Lite .....	19
2.2.12 Onnx.....	19
<b>2.3 Related Research and Development</b> .....	<b>19</b>
2.3.1 Recycle Mate .....	19
2.3.2 Trash Bot .....	20
<b>Chapter 3</b> .....	<b>22</b>
<b>3.1 Architecture</b> .....	<b>22</b>
<b>3.2 User Flow</b> .....	<b>22</b>

3.3 Model Transion.....	23
3.4 Firebase Connection for Android .....	23
3.5 Firebase Connection and Further Set-up for IOS Testing .....	25
Chapter 4 .....	26
4.1 Previous Results.....	26
4.2 Screens and User Flow .....	27
4.3 Implementation.....	30
4.4 Mobile Model Results.....	34
Chapter 5 .....	36
5.1 Summary of Accomplishments.....	36
5.2 Issues and Obstacles.....	36
5.3 Future Directions.....	36
5.4 Lessons Learned .....	37
References .....	38

## **Chapter 1**

### **Introduction**

#### **1.1 Problem Statement**

Mismanaged trash is 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 has not been a way for people who may be new to recycling to quickly and intuitively participate in the new system. Aside from lack of facilities, barriers to individuals' recycling include the lack of convenience and the process itself consuming too much time.

#### **1.2 Project Solution Approach**

Our project's approach to solving this problem is to create 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. Since 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 allow the user to interact with the system 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**

Our main objective in the first semester was to create a machine learning model that could, to a certain benchmark, predict the material make up of a given image and its recyclability. We eventually completed this task, creating a model consisting of a total of 10 classes and achieving an accuracy rate of roughly 90%. 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. After the creation of this machine learning model, our current goal for the project shifted to the development of a mobile application that can host the previously stated model. The app will be connected to the user's camera, which will make it possible for the user to take a picture directly within the app itself. Subsequently, the pictures will be processed through the model and the prediction displayed to the user, while a copy of the picture will be sent to an external database to add to the pre-existing dataset. The user will then get a chance to evaluate the prediction result if the outcome is not consistent with the actual category of the input.

## Chapter 2 Background

### 2.1 Fundamental Theory and Concepts

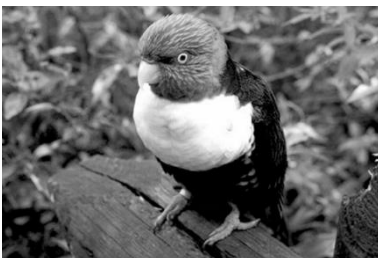
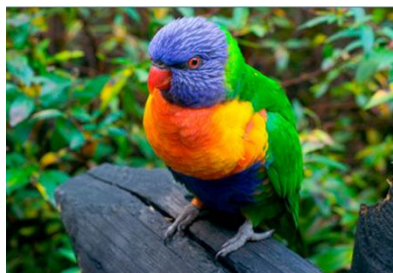
#### 2.1.1 Digital Image Structures

A pixel is the smallest subset within a picture that stores a value associated with its color intensity, 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. In figure 2.1.1.1, notice the vividly blue feathers on the bird's head in the original picture and how it registers as a lighter color in the blue channel and darker in others. Similarly, the reddish beak and breast appear bright in the red channel but dark in others.

Original Image [a]

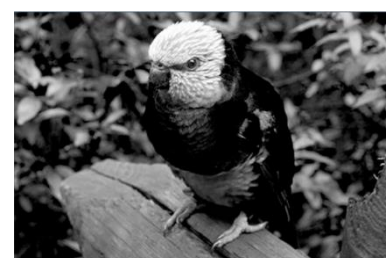
[1]



Red  
[b]



Green  
[c]



Blue  
[d]

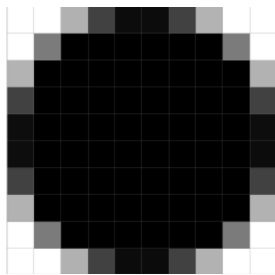
Figure 2.1.1.1 – Color image [a] and individual channels ([b] through [d])

## 2.1.2 Image Operations

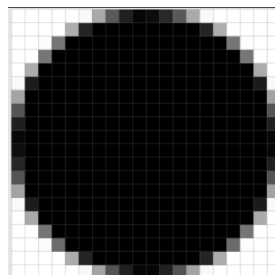
### Resizing

The resizing process is the practice of either increasing or decreasing the number of pixels that comprise 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 poor since the image will go through a process called resampling. 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.

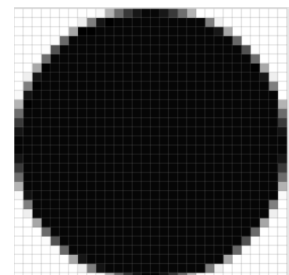
[4]



(10 x 10px)  
[Figure 2.1.2.1]



(20 x 20px)  
[Figure 2.1.2.2]



(30 x 30px)  
[Figure 2.1.2.3]

### Grayscale

Grayscale is the process where a colored image is converted into a picture with only one color channel, turning the picture into shades of gray between black and white. There are generally two main approaches to turning a picture into a grayscale image. 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. The second, called the Luminosity Method, also takes an average of the three colors but considers how perceptible each color is. Since the human eye is more sensitive to the color green when compared to others like red or blue, the value associated with the color green is given much more weight within the equation, which can be seen in figure 2.1.2.5.

[5]

$$I = 0.33 * R + 0.33 * G + 0.33 * B$$

Average Method  
[Figure 2.1.2.4]

$$I = 0.299 * R + 0.587 * G + 0.114 * B$$

Luminosity Method  
[Figure 2.1.2.5]



Base image  
[Figure 2.1.2.6]



Average method  
[Figure 2.1.2.6a]



Luminosity Method  
[Figure 2.1.2.6b]

### 2.1.3 Artificial Neural Network (ANN) or Forward Feed Neural Network

An Artificial Neural Network is a web of interconnected nodes with a structure reminiscent to that of a brain. An ANN is made up of computing elements (called neurons) which can be adapted to recognize patterns or classify inputs. A neuron is a point where one or multiple inputs are given, processed, and an output as a singular value is passed on. Artificial neural networks get their name from how it works by making minor adjustments to these interconnected neurons inside their system. ANNs are usually used in large projects with a large amount of data to parse through since an ANN excels at pattern recognition and ignoring minor blemishes in the dataset. The structure of an ANN consists of an input layer of neurons, 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 Feed Forward Neural Networks since the data can only move in one direction. While there is always only one input and one output layer, the researcher can set the number of hidden layers and is usually changed depending on the use case. The neurons are then bound together by a web of connections, each of which has an associated numeric bias and a set of weights, which are modifiers that effect the output of the node. Weights refer to a number that is multiplied to one of the node's input to determine its significance; this means that if the node has 15 inputs, there will be a set of 15 weights to correspond to the number of inputs. On the other hand, each node contains only one bias taking the form of a constant number. This number is then added to offset the output of that node. 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.

Due to its architecture, an artificial neural network 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 layer may detect lines, others may detect other traits like shadows.

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 weighing 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 and Learning**

ANNs learn via the process of backpropagation. This is a mechanism that adjusts the weights on connections between neurons, from one trial to the next, in order to improve accuracy and reduce the loss. 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 determined by the model's predicted number when compared to the actual answer that the model was supposed to come to. If these values differ, then the loss value will be non-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 loss value.

#### **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 embodied in the starting network, this means that the model will not 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 use 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 performs, 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. Each cell in the filter matrix (also known as a kernel element) 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 one group of pixels at a time; the process starts from the top left with the filter moving rightwards until it is no longer possible then downwards, repeating until the bottom right of the image is reached. The set of numbers that determine the speed at which the filter moves through the image is called the stride and is usually stored using the format of (x,y), with the first number signifying its movement horizontally and the second vertically. As the stride controls the movement of the filter after each calculation of the convolution, larger numbers as the stride value consequently means a smaller output shape and faster computation time.

For a 7x7 image to be processed by a 3x3 kernel, the convolutional views the picture in 9-pixel groups, corresponding to the size of the kernel as seen in figure 2.1.6.1. Referring to figure 2.1.6.3 a-c, the kernel is then overlaid onto each of these groups 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. This example uses a stride value of 1 in both horizontal and vertical directions.

0.0	0.0	0.35	0.35	0.35	0.0	0.0
0.0	0.35	0.65	0.65	0.65	0.35	0.0
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.0	0.35	0.65	0.65	0.65	0.35	0.0
0.0	0.0	0.35	0.35	0.35	0.0	0.0

Original Image  
[Figure 2.1.6.1]

1.0	-1.0	0.0
1.0	-1.0	0.0
1.0	-1.0	0.0

Kernel  
[Figure 2.1.6.2]

AA	AB	AC				
BA	BB	BC				
CA	CB	CC				

[Figure 2.1.6.3a]

	AA	AB	AC			
	BA	BB	BC			
	CA	CB	CC			

[Figure 2.1.6.3b]

		AA	AB	AC		
		BA	BB	BC		
		CA	CB	CC		

[Figure 2.1.6.3c]

### Kernel's Progression Through the Original Picture

0.0	0.0	0.35	0.35	0.35	0.0	0.0
0.0	0.35	0.65	0.65	0.65	0.35	0.0
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.0	0.35	0.65	0.65	0.65	0.35	0.0
0.0	0.0	0.35	0.35	0.35	0.0	0.0

[Figure 2.1.6.4]

1.0	-1.0	0.0
1.0	-1.0	0.0
1.0	-1.0	0.0

[Figure 2.1.6.5]

### Kernel Overlaid onto the Original Image

$$\text{Output Pixel} = 1.0[\text{AA}] - 1.0[\text{AB}] + 0.0[\text{AC}] + 1.0[\text{BA}] - 1.0[\text{BB}] + 0.0[\text{BC}] + 1.0[\text{CA}] - 1.0[\text{CB}] + 0.0[\text{CC}]$$

$$\text{Output Pixel} = 1.0[0.0] - 1.0[0.0] + 0.0[0.35] + 1.0[0.0] - 1.0[0.35] + 0.0[0.65] + 1.0[0.35] - 1.0[0.65] + 0.0[1.0]$$

$$\text{Output Pixel} = 0.05$$

### Convolution Equation for the First Pixel

[Figure 2.1.6.5]

0.0	0.0	0.35	0.35	0.35	0.0	0.0
0.0	0.35	0.65	0.65	0.65	0.35	0.0
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.35	0.65	1.0	1.0	1.0	0.65	0.35
0.0	0.35	0.65	0.65	0.65	0.35	0.0
0.0	0.0	0.35	0.35	0.35	0.0	0.0



	0.05					

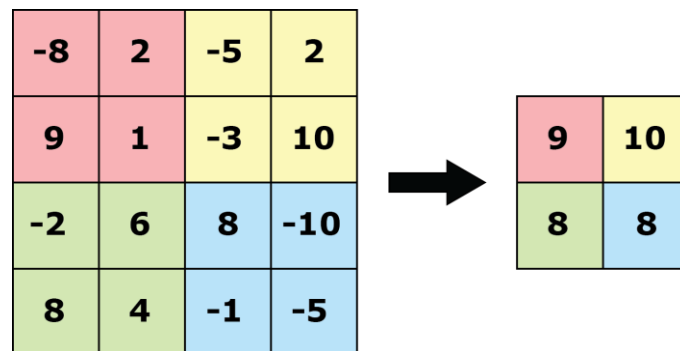
### Convolution Overview

[Figure 2.1.6.6]

## 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 alterations including shifting may produce a different feature map.

To illustrate, figure 2.1.6.7 shows an input picture of size 4x4 being processed through a max pooling layer of size 2x2. Each 2x2 section of pixels will be grouped together without any overlap, and the pixel with the high value within each group of 4 pixels will become the output pixel.



Max Pooling Process  
[Figure 2.1.6.7]

## 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 often follow immediately after convolutional layers, as the ReLU layer will be able to catch any negative values before they enter into the pooling layers.

## Fully connected layer

The output from the pooling layer will eventually 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.

### 2.1.7 Evaluation Metrics

#### Confusion Matrix

A confusion matrix is a table of size  $N \times N$  ( $N$  being the number of classes we have within the dataset), which is used to measure the performance of our model. The rows in the table indicate the true classes. The columns indicate the assigned classes. The value in cell  $[I, J]$  is the number of items from class  $I$  that were labeled as being class  $J$ . In this table, each cell 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. These are cells where the row and column indices are the same.

**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		
Actual	0	TP	FN	FN		Actual	TN	FP	FN		0
	1	FP	TN	FN			FN	TP	FN		
	2	FP	FN	TN			FN	FP	TN		

Confusion Matrix  
[Figure 2.1.6.7]

#### Precision Score

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. The value of this score ranges from 0 to 1 where 1 is the best and 0 the worst.

$$\text{Precision Score} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

### 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. Once again, the value ranges from 0 to 1 with 1 being perfect recall.

$$\text{Recall Score} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

### 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.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

## 2.1.8 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. Specifically, web scraping issues HTTP requests to target sites, then analyzes the results returned from a site, which will usually be in HTML format. The user can then parse through the returned results using commands to retrieve any relevant or desirable information.

Screen-scraping libraries like BeautifulSoup [17] have always been popular for their ease of use and uncomplicated syntax. Extensions also exist specifically for scraping pictures, including ones like the Firefox extension DownThemAll [12], 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, 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.

## 2.1.9 Native and Cross Platform Mobile Application

Each mobile device requires an operating system, or a program which acts as an interface between the user and the device, to function, with the most popular ones currently being Android and IOS. While both of these systems are targeted towards mobile use, they do not occupy the same ecosystem; this means that an app intended for one is often incompatible with the other, as different operating systems run off different functions and systems. Applications on Android or IOS operate through an underlying code base; this code needs to be written in a language that is compatible to the target platform, giving rise to two methods when developing a mobile application: native and cross-platform. The first, as the name implies, runs languages or systems that are native to the targeted operating systems. An example of this concept is the Swift [18] coding language, which corresponds to the IOS operating system. This language includes functions and capabilities that

allow it to fully access the systems and functionalities of the operating system. On the other hand, cross-platform refers to a system that can be used on two or more platforms such as languages like Kotlin [19] or SDKs such as Flutter. Furthermore, there are many ways the cross-platform status can be achieved, one of which is by having a single codebase which can then be compiled into different operating system ecosystems individually; this example is how Flutter operates. On the other hand, frameworks such as Cordova [20] accomplish this by rendering the application on an integrated native browser, meaning Android and IOS phones will likely be running their applications on Chrome and Safari respectively, using standard web application languages such as HTML [21], CSS [22], and JavaScript [23]. Although this method is less complex than others that have been covered thus far, the framework's lack of a consistent browser across different platforms does pose some obstacles when developing certain functionalities. While on the surface, cross-platform languages seem more desirable based on their versatility alone, both options have their unique advantages.

Being linked to only one operating system allows for easier access to functionalities of the device such as their cameras, microphones, and GPS systems. Furthermore, apps developed via native languages are usually smoother to operate as it can lean heavily into the platform's architecture being much more form fitting. However, two codebases are needed to replicate what cross-platform languages can achieve with just one, making the cross-platform option more attractive for larger apps with larger and more complex functionalities. Due to the other option entailing changes on both codebases, which usually means developing the changes in two different ways to account for the specifications of different operating systems.

### **2.1.10 Backend as a Service**

BaaS works as an alternative to traditional backend architecture, purchasing rather than building architecture such as databases or in the case of Firebase, Real-Time Databases which synchronize data across all the serviced devices. Even though storage is one of the main functionalities that a BaaS provides, the service provided could be anything from database storage to push notifications. Furthermore, most of these functionalities often require very few lines of code to implement in the client application, making it much easier and more time efficient when compared to building the services from the ground up. The more individualized nature of these services also makes them more modular, which benefits developers as they can pick and pay for only the services they use, rather than paying for unused functionalities; this means that its also easier to incrementally introduce features without a large spike in subscription cost, as each of them stands on their own.

## **2.2 Technologies**

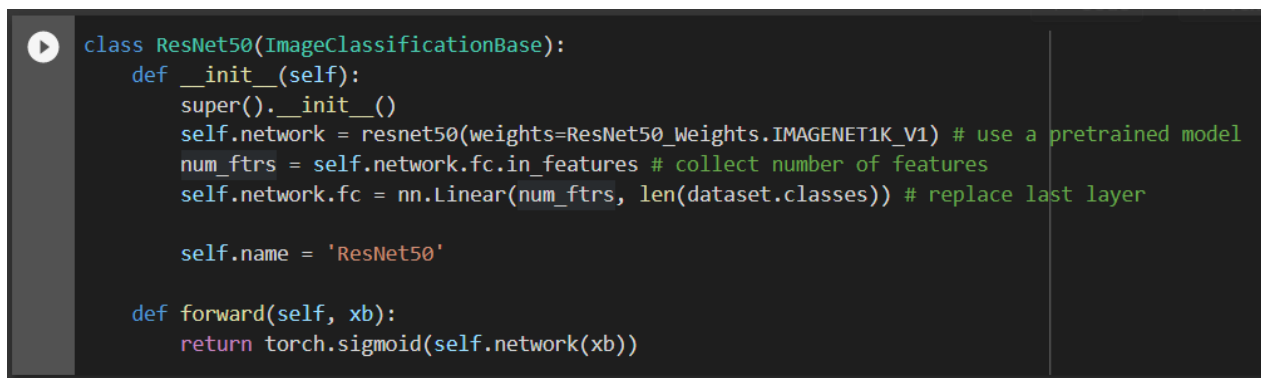
### **2.2.1 PyTorch**

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, with the models and their functionalities, such as the training and validation steps, being implemented as a set of Python classes. Its ease of use and readability compared to TensorFlow made it a solid choice for this project. Furthermore, this ability to easily import and use partially trained models in a process called transfer learning as shown in figure 2.2.1.1, which is highly useful for projects with small datasets such as ours.

Additional capabilities include its ability to pre-process the images without the use of a third-party library; this being functions that resize, rotate, greyscale and add random noise the input image.

After a model has been trained, it can be processed through a function to make it exportable. Packaged as a part of PyTorch, Torch Script provides a way to save and export PyTorch models with the function calls `jit` and `save` respectively. The exported file becomes Python independent, which means that a model can be developed in a Python environment but exported to run on another environment where running Python might be counter-intuitive or just not possible.

Lastly, a library is needed to import the packaged model back into our Flutter application. PyTorch Mobile [24], a Flutter plug-in created by Fynn Maarten and Elbek Khoshimjonov, allows an exported PyTorch model to be called directly within a Dart program. Furthermore, the plug-in can be used with models that use images as their inputs which is what we are interested in. The exported model needs to be in the format `.pt` rather than other common formats such as `.pth`, since the first format is more suitable for mobile application; this is because the file extension `.pth` is also used by Python for their path configuration files.



```
class ResNet50(ImageClassificationBase):
    def __init__(self):
        super().__init__()
        self.network = resnet50(weights=ResNet50_Weights.IMAGENET1K_V1) # use a pretrained model
        num_fters = self.network.fc.in_features # collect number of features
        self.network.fc = nn.Linear(num_fters, len(dataset.classes)) # replace last layer

        self.name = 'ResNet50'

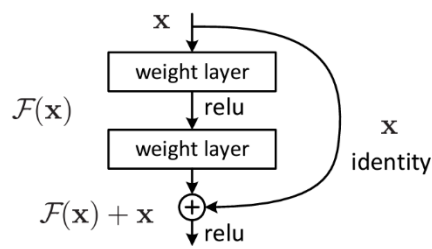
    def forward(self, xb):
        return torch.sigmoid(self.network(xb))
```

Importing Resnet  
[Figure 2.2.1.1]

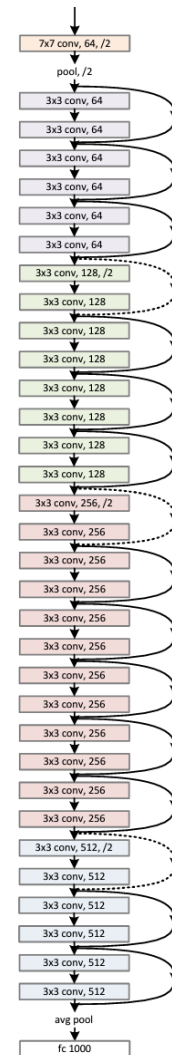
## 2.2.2 ResNet

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 is a metric which reflects the probability that the model assigns the highest score to an incorrect class.



Convolution Architecture  
[Figure 2.2.2.1]



[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 an open-source, 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 BeautifulSoup 4, the technique fell through when calls made to a Google-Lens URL would default back to the landing page. DownThemAll, 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.

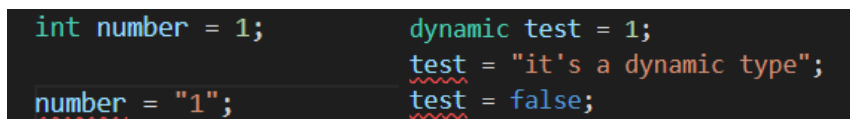
### **2.2.8 Flutter/Dart**

Developed by Google at the tail end of 2018, Flutter is an open source UI framework that enables the creation of applications on different platforms such as IOS, Android, and Desktop. The framework is usually used in tandem with the Dart programming language, which is also created by Google in 2011. Since Flutter can compile a code base for multiple different native formats, it is appropriate for building cross platform applications.

Dart is a language that is strongly typed, meaning that variables must be declared as a particular data type before they are used, and using them in a context where another data type is required will generate errors as the variable will not be implicitly converted to a usable form. Furthermore, the language is somewhat unique in the sense that it is statically typed but its variables can be manually set to be dynamic as well. Statically-typed languages check for their type errors during the compile time, while dynamic ones check during the run time itself; the stricter nature of static languages make it marginally harder to write, but also significantly safer. In Figure 2.2.8.1 differences of the two types can be observed; it should be noted that while the dynamic example does alert the user, the code is still perfectly runnable. The alert stems from the fact that even though dynamic typing is a part of Dart, its use is discouraged due to the possible runtime issues that may occur due the variable's non-static nature.

Dart is a compiled language. An application created within the Flutter framework has a single codebase. Compatibility to different operating systems is achieved by compiling the codebase into different machine code formats. Unlike languages like C or Haskell, Dart is an object-oriented programming language; this means that Dart works with objects, which are user generated classes with their own variables and functions associated with them, meaning that variables which are assigned a certain class will inherit the variables and functions of that class.

A Flutter application is created by assembling and customizing widgets. A widget in this case just pertains to a part of the user interface; this can range from simple images and texts to more complex functionalities such as buttons or scroll bars. Many widgets are already being built into the framework. While these standard widgets may be employed across different operating systems without issue, there are some operating system dependent functionalities which are not usable on every system. To deal with this, Flutter provides packages that can access more specific systems such as cameras, geolocation, and even systems like Siri. Once a package of this sort has been installed, it can be used directly in the codebase. Furthermore, the framework has hot-reloading capabilities, which allows the developer to see the adjustments made to the codebase in real time within the already running application without the need to recompile or reload. This feature makes it significantly faster to track in real-time how the newly written code interacts with the existing codebase and provides for a smoother workflow without the need to constantly close and re-run the testing application.



```
int number = 1;           dynamic test = 1;
number = "1";             test = "it's a dynamic type";
                           test = false;
```

Static vs Dynamic Typing  
[Figure 2.2.8.1]

### 2.2.10 Firebase

Firebase is an SDK created by Google in 2011 which provides many services ranging from real time databases to website hosting and many other things in between: push notification, authentication, and advertisements [25]. The main draw of Firebase is its user friendly nature, as some of the functionalities can be managed directly from their online page, along with its ability to be used with virtually every operating platform, while at the same time being extremely light on back-end code. In this project, Firebase was selected due to its compatibility with the Flutter framework, as both come from the same company, Firebase is already equipped with a wide range

of plug-ins which make the set-up process on Flutter much less complex than other competing platforms such as AWS Amplify [27] or MongoDB [26]. In terms of pricing, Firebase provides the user with two pricing options: the Spark and the Blaze plans. The former is a no-cost plan with a few set limitations regarding advanced functionalities, user counts, and other general limits, while the latter operates on a “pay as you go” model. The Blaze plan provides the same functionalities as the Spark plan, along with other services such as cloud functions, with the cost being calculated by the number of uses the functionality experiences rather than stated as a set price.

The main Firebase functionality used in this project is the cloud storage function, which allows for the transfer of images directly from the user’s device to our cloud databases. Aside from that, the only other service we will be using is the no-SQL database to store the usernames for a more personalized experience. No-SQL databases differ from traditional SQL databases due to its key-value system rather than the normal relational system; this key-value system allows for schema-less databases, which means that each data item may have a unique set of attributes, unlike SQL systems where all items of a particular category have the same attributes (table columns). The unstructured nature makes it safer to apply structural changes to the table as there is no threat of data loss, unlike more traditional tables. Although no-SQL databases offer some convenience, they also have drawbacks in terms of efficiency. Because of its non-standardized nature, the query time for selecting a batch of items based on a field will be significantly longer than a normal SQL database.

#### **2.2.11 TensorFlow Lite**

TensorFlow Lite is a file format used to store a compressed version of a TensorFlow machine learning model by freezing the model’s graph along with its weights and biases [28]. It is conceptually similar to TorchScript. First released in 2015 by Google, TensorFlow Lite is used to operate machine learning models, which are usually large, in smaller devices such as mobile phones. While normally used as a part of the TensorFlow ecosystem, there is a way to convert a PyTorch file into the TensorFlow Lite format by using an open source library named Onnx. This project will also make use of a Flutter library named tflite [29], which is a plugin that enables us to import and use a TensorFlow Lite model inside the Flutter application.

#### **2.2.12 Onnx**

Developed in a joint effort between Facebook and Microsoft and released in 2017, Onnx is a specific format made to store machine learning models, which works mainly as an intermediary and a transition phase between two different machine learning formats [30]; this functionality makes it possible to convert models into a different format to avoid the issue where some environments only support certain machine learning formats and not others.

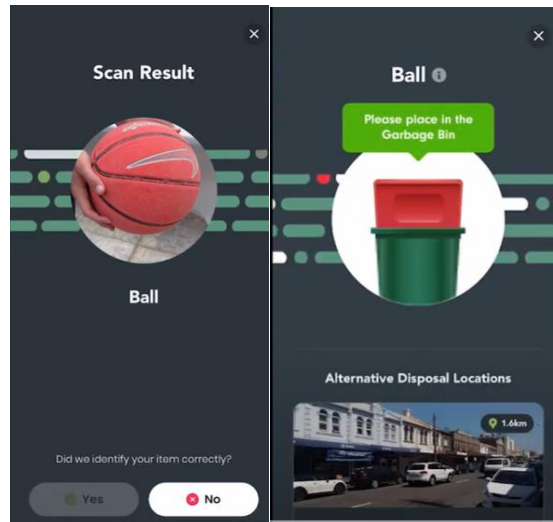
### **2. 3 Related Research and Development**

#### **2.3.1 Recycle Mate**

Recycle Mate is an artificial intelligence-based recycling application designed and developed by DreamWalk, an app development agency. Released in 2019 as a beta on both IOS and Android, the app works by letting the user take an image then analyzing it with their own custom- built TensorFlow model. Finally, a prediction is displayed to the user whether the item scanned is

suitable for the yellow bins, which denotes the recycling bins as specified by the Australian Government. In addition to the smooth transitions between the app's different pages, the app itself has a friendly user flow, as it mainly consists of just three main screens: photo taking, analyzing, and prediction. Currently, the app can only be used in Australia as it is a part of a domestic government backed program.

[16]



[Figure 2.3.1.1]

### 2.3.2 Trash Bot

Developed by the Clean Robotics team and based in the United States, Trash Bot is a trash receptacle that uses an artificial intelligence model to sort discarded items into different bins. While the internal model of the product is not clearly described, the company does claim that it is able to correctly sort given items accurately 95% of the time. In addition to that the bins themselves have multiple chutes for the different categories, along with application which can display whether the bins are full. The product works by first lowering the product into a small temporary bin where an internal camera takes an image then processes said image. Depending on the prediction, the temporary box then maneuvers itself above the matching chute, and the item is then dropped inside the specified box through a trapdoor like mechanism on the floor of the temporary box. Although the company has yet to place these unit in public places, except for events such as conferences and a handful of festivals, the company has announced a deal worth USD 4.5 million with an investment company in Hong Kong with the name Melco International Development Limited. Furthermore, they state that they are currently looking at the possibility of partnerships within countries like China, Australia, and Singapore.



[Figure 2.3.1.2]

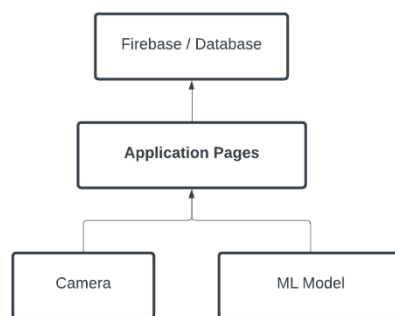
## Chapter 3

### Methodology

This chapter will discuss the details regarding the application's development, including its architecture, user flow, and Firebase connection. Furthermore, additional information will be given about the model's transformation process from a PyTorch model to one using TensorFlow Lite.

#### 3.1 Architecture

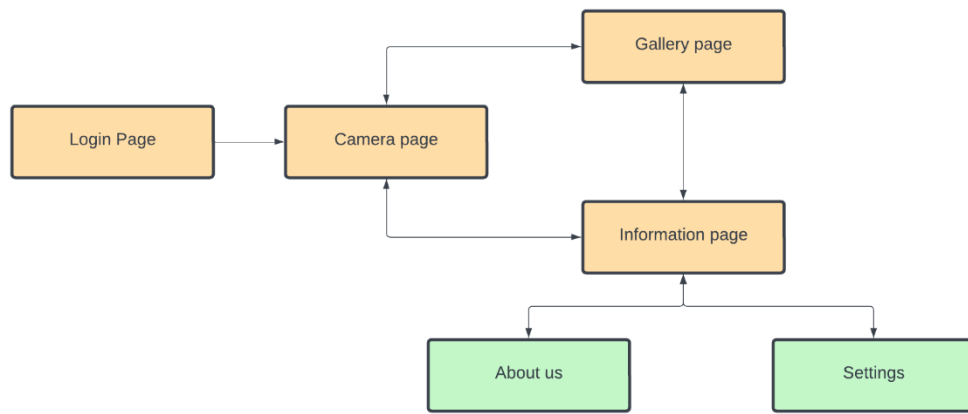
As seen in figure 3.1.1, our current application includes only four main components, which include the application pages, ML model, camera, and database. Similarly, the flow of information is also relatively linear with an input in the form of an image coming into the application from the camera then subsequently transported to the model for prediction. The result is shown to the user and the image is then duplicated and sent to an external database, which in this case is Firebase, with the user's consent.



Architecture Overview  
[Figure 3.1.1]

#### 3.2 User Flow

Referring to figure 3.2.1, our user flow is generally straightforward and can be segmented into three main segments: login, camera, and information. When a user first enters the application, they are greeted with the login page. The name given to the page will not be made into an account but will be displayed in the loading screen in subsequent visits to provide the app with a sense of personalization. After the login page, the user will then be introduced to the camera page; this page will be where the main functionality of taking images and getting their prediction takes place. Optionally, the user can then choose to move to the information page or gallery by navigating to it via the selectors located at the bottom of the page. In the information page, the user will be able to access the app's Settings and About Us pages, along with seeing the number of items the application as scanned since its release. On the other hand, the gallery will be occupied by the pictures the user has previously taken within the application.



User Flow Overview  
[Figure 3.2.1]

### 3.3 Model Transion

As stated in the architecture section, due to an unforeseen problem with the PyTorch library that we intended to use with Flutter, we decided to switch to another machine learning framework which is supported by working libraries. In this process, a transformation is needed, and this section will go over the details of that process.

As seen in figure 3.3.1, our model file is first saved through torch jit, which is a way to save the current biases and weights of the current model so that it can be exported; this call will produce a file with the .pt format. However, we could not find a way to directly translate a .pt into a .tflite format which is what is used by TensorFlow Lite. Due to this, the file needs to first be changed into a format readable by our intended receiver; this is where Onnx comes in, being an intermediary framework, it is able to read and be read by both PyTorch and TensorFlow. By calling the export function in Onnx while giving the model and input parameters as the arguments, an output file with the .onnx format will be created. The file can then be treated like a TensorFlow file and be translated into TensorFlow Lite through the normal means.



Model Transformation Process  
[Figure 3.3.1]

### 3.4 Firebase Connection for Android

To connect a project to Firebase's services, it must first be registered to the platform, which can be done on the Firebase website with a name that can be linked with the application. In the application directory, the `build.gradle` file is accessed by following the `android > app > build.gradle` path, and the same name given to the project is then set as the

application identifier as seen in figure 3.4.1. After which a json will have to be downloaded and placed within the same directory as `build.gradle`.

```
defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
    applicationId "com.example.flutter_app"
    // You can update the following values to match your application needs.
    // For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-gradle-build-configuration.
    minSdkVersion 21
    targetSdkVersion flutter.targetSdkVersion
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
}
```

Setting the Application Identifier  
[Figure 3.4.1]

The classpath is identified in the `android > build.gradle`, which is not the same file as the one where the application name is set, providing a path back to Firebase's resources. Plug-ins and other dependencies can then be initialized in `android > app > build.gradle` shown in figure 3.4.3 – 3.4.4.

```
dependencies {
    classpath 'com.android.tools.build:gradle:7.2.0'
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    classpath 'com.google.gms:google-services:4.3.15'
}
```

Setting Class Paths  
[Figure 3.4.2]

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation platform('com.google.firebase:firebase-bom:31.5.0')
    implementation 'com.google.firebase:firebase-analytics-ktx'
}
```

Importing Plug-ins and Dependencies  
[Figure 3.4.3]



### 3.5 Firebase Connection and Further Set-up for IOS Testing

Making the connection to Firebase IOS wise includes only two main steps, the first being to add the registered name of the project to `flutter.app > ios > Runner.xcodeproj > project.Pbxproj` as shown in Figure 3.5.1; this is then followed by downloading the corresponding google services file, which will be imported into the `flutter_app > ios > Runner > GoogleService-Info.plist` directory within our application.



Firestore Connection  
[Figure 3.5.1]

Even though the connection to Firebase requires significantly fewer steps than that of its Android counterpart, further changes would have to be made to run the application on an iOS device. The codebase is first moved from Virtual Studio Code into Xcode; this is preformed as the option to initialize the application on a pre-existing iOS device is supported by Xcode. However, to work with Flutter in this new environment a package called `homebrew` [33] must first be installed which acts as a channel to download dependencies that cannot be directly called natively by MacOS otherwise. `flutter-app/ios/Runner.xcworkspace` will then be accessed and a log in process will have to be performed using an existing iCloud account. Lastly, a field within `flutter-app/ios/Runner/infoplist` is changed which is seen in Figure 3.5.2; this change will make it so that permission is asked when accessing the device's camera.

Information Property List		Dictionary	(19 items)
Privacy - Camera Usage Description	String	\$(PRODUCT_NAME) camera use	
CADisableMinimumFrameDurationOnPhone	Boolean	YES	
Development localization	String	\$(DEVELOPMENT_LANGUAGE)	

Accessing the Camera / Privacy Settings  
[Figure 3.5.2]

## Chapter 4 Results

### 4.1 Previous Results

In the past semester, our team worked on developing a transfer learning-based machine learning model that is able to discern the recyclability of an object, resulting in 2 different models. The first was a ResNet50 based model that was able to achieve an average accuracy of more than 90% and an F1 score of around 85%, while the second model was a Keras model constructed from scratch that achieved slightly lower accuracy with 67% for both metrics. Both of the model's confusion matrices and metrics are displayed below in figures 4.1.1 – 4.1.4. Due to its performance, our team decided to choose the ResNet based model for the base for our application going forward.

Version No.	Accuracy	Recall	Precision	F1 Score
12	97.01%	85%	94%	88%
13	93.88%	77%	89%	81%
14	93.88%	75%	87%	79%
15	91.81%	82%	90%	85%

Figure 4.1.1 ResNet Evaluation Metrics

		Actual											
		wood	metal	kitchen ware	photo graphs	plastic	ceramic	styro foams	paper	glass	packing peanuts	light blub	
Predicted	wood	113	0	0	0	0	0	0	0	0	0	0	
	metal	12	185	0	1	13	0	0	0	2	0	0	
	kitchen ware	0	0	180	1	0	0	0	0	0	1	0	
	photo graphs	0	0	0	111	0	0	0	0	0	0	0	
	plastic	10	28	0	0	287	3	0	0	38	3	3	
	ceramic	0	0	0	0	0	41	0	0	0	1	0	
	styro foams	1	0	0	5	0	2	220	22	9	37	2	
	paper	0	0	0	0	0	0	0	74	0	0	0	
	glass	4	47	0	2	38	5	20	4	247	12	1	
	packing peanuts	0	0	0	0	0	9	0	0	4	146	0	
	light blub	0	0	0	0	2	0	0	0	0	0	34	

Recall: 82%  
Precision: 90%  
F1\_score: 85%

Figure 4.1.2 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 4.1.3 Keras Evaluation Metrics



Figure 4.1.4 Keras Confusion

## 4.2 Screens and User Flow

The final iteration of our user flow is similar to the one we previously described in the methodology section, and is shown in Figure 4.2.1.

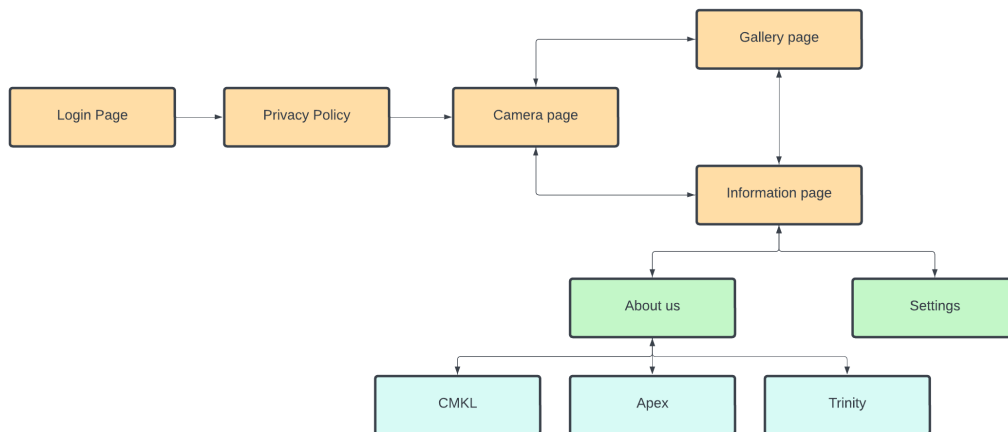
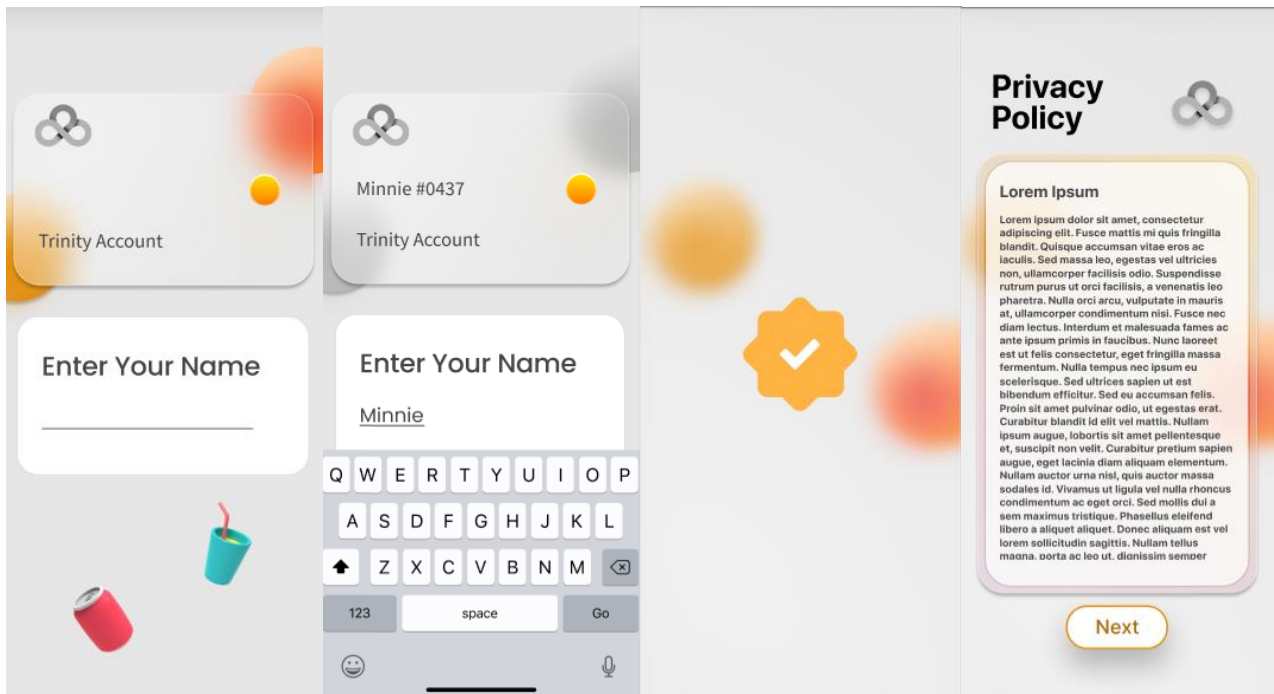
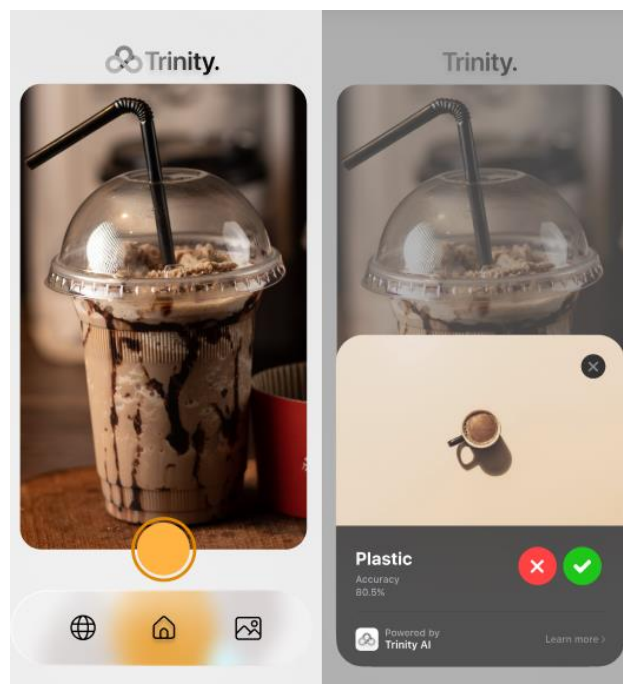


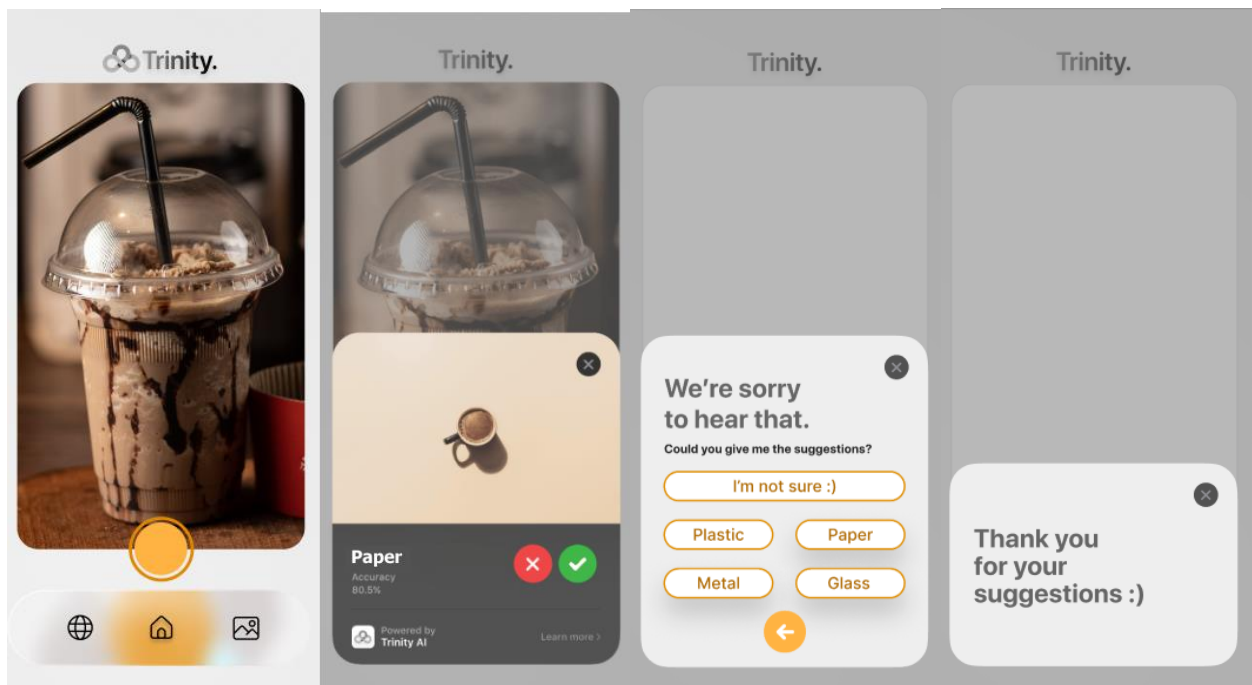
Figure 4.2.1 User Flow Overview

The login page still serves the same purpose. The user inputs a name which the application will later refer them by and the user will then be transferred to the privacy policy page; this chain of events is preformed if this is the first time the user accesses the application. The privacy policy page contains information on how the retained data will be used by the application, along with any other facet that pertains to the user's data privacy. As this page is of high importance, the user must scroll down to the end of the policy prompt before the consent can be given; this measure is an attempt at ensuring that the user has fully read and has agreed to all the conditions listed. If the user does choose to agree to the policy, they are then subsequently moved to the main camera page.

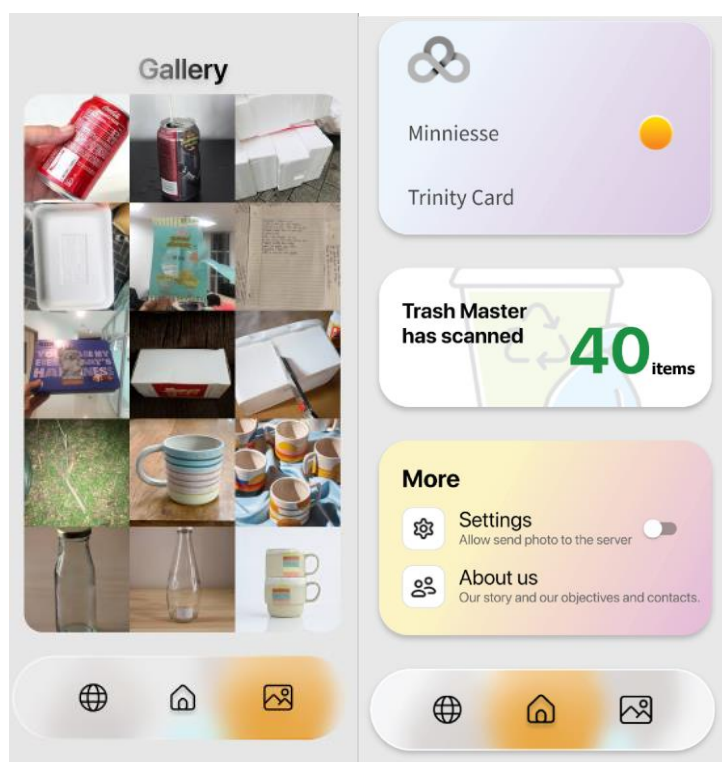


After consenting to the presented policies, the user is given access to the main loop of the application, which is made up of three main pages: camera, information, and gallery. The page that the user finds themselves in first is the camera page, where the user can take pictures and receive a prediction on it. A pop-up will then appear with the prediction along with a feedback form in which the user can press on options corresponding to whether the prediction was made correctly; this process determines which folder the image will later be moved into. From this page the user can access the other two main pages and vice versa, by using the navigation bar located at the bottom of each page.





One of the options within the navigation bar is the gallery page, which as the name suggests stores the images the user has taken previously, displaying them in rows of three. The other option is that of the information page, containing 3 main sections: object statistics, settings, and about us. Being second from the top, the object statistics section displays the number of objects processed by the model since the launch of the application. As our settings page is set to contain only the capability of enabling and disabling the collection of images from the user, our team chose to make the option a slidable toggle rather than constructing an entirely new page.





On the other hand, the About Us section does have multiple sub-pages which can be accessed by interacting with the icon. After pressing the icon, the user is moved to the overview page which contains 3 further pages each hosting information on topics relating to team, organization, and technologies relating to the application. Pressing one of the sections moves the user into a separate page containing further details regarding the desired topic.



### 4.3 Implementation

As stated in the architecture section of the report, the main core of this application revolves around three main segments, which make up the overall flow of the program. Starting with the first object in the flow, the camera functionality is imported into the Flutter application using a package named `camera` [31]. `camera` is an official package released by the same developers who made Flutter and is used to take new images using the camera. After a valid image is produced, it will then be fed into the machine learning model. At first we were looking to use an open source library

to run the model on our application, but after experiencing several issues which will be further touched upon in chapter 5 we decided to change our model format to that of TensorFlow Lite; this change required us to re-format the model which required a few attempts, but the model was eventually made compatible with TensorFlow Lite using an intermediate framework called Onnx. After the transition, we can now import both the prediction labels and model itself by calling `loadLabels` and `loadModel` as shown in figure 4.3.1. Subsequently, the input image can then be passed into the model for a prediction by using the `predict` function.

```
static Future<Classifier?> loadWith({  
  required String labelsFileName,  
  required String modelFileName,  
}) async {  
  try {  
    final labels = await _loadLabels(labelsFileName);  
    final model = await _loadModel(modelFileName);  
    return Classifier._(labels: labels, model: model);  
  } catch (e) {
```

Importing the Model  
[Figure 4.3.1]

```
void _analyzeImage(File image) {  
  _setAnalyzing(true);  
  
  final imageInput = img.decodeImage(image.readAsBytesSync());  
  
  final resultCategory = _classifier.predict(imageInput);  
  
  final result = resultCategory.score >= 0.8  
    ? _ResultStatus.found  
    : _ResultStatus.notFound;  
  final plantLabel = resultCategory.label;  
  final accuracy = resultCategory.score;
```

Prediction Step  
[Figure 4.3.2]

Before the next step, we first must determine whether the user allows the application to collect their images. To achieve this, a package named `SharedPreferences` is imported as shown in figure 4.3.3 into the camera and information pages; this package is used to store key and data pairs which can later be referenced in other pages.

```
import 'package:shared_preferences/shared_preferences.dart';
```

Importing Shared Preferences  
[Figure 4.3.3]

In the information page, there will be a section at the bottom with the name settings, which contains a slider that the user can use to toggle their consent regarding the collection; this slider is connected to the function displayed in figure 4.3.4, setting a variable named `buttonEnabled` to a Boolean value consistent with the current state of the slider.

```
Future<void> _saveButtonState(bool isEnabled) async {  
  SharedPreferences prefs = await SharedPreferences.getInstance();  
  await prefs.setBool('buttonEnabled', isEnabled);  
}
```

Setting Button Variable  
[Figure 4.3.4]

Conversely in the camera page, the program will then use the set variable to determine the next course of action after the prediction stage is complete. If the user allows for the transfer of the image, it will be saved into the gallery and subsequently transferred into Firebase for storage. If no permission is given, the image will only be saved within the gallery. To save the taken image into the gallery, the application uses the function `_saveImageAndNavigate`, which sets a name for the image, finds the image directory for the application, creates a destination point in the directory, and saves the image to said destination point; the path to this image is then saved into the a global variable containing every single image's path.

```
Future<String> _saveImageAndNavigate(XFile image) async {  
  final Directory appDir = await getApplicationDocumentsDirectory();  
  final String imageName = DateTime.now().millisecondsSinceEpoch.toString();  
  final String finalImagePath = join(appDir.path, '$imageName.jpg');  
  
  await image.saveTo(finalImagePath);  
  print('Image saved to: $finalImagePath');  
  
  SharedPreferences prefs = await SharedPreferences.getInstance();  
  List<String> imagePaths = prefs.getStringList('imagePaths') ?? [];  
  imagePaths.add(finalImagePath);  
  await prefs.setStringList('imagePaths', imagePaths);  
  
  return finalImagePath;  
}
```

Saving to Gallery  
[Figure 4.3.5]



As for transporting the image to Firebase, whose connection will later be elaborated upon in one of the following sections, a function starting with `_uploadImage` followed by typing and results are called; the typing consists of the classes within the dataset and the results signify the correctness of the prediction, where T is correct and F is wrong. So you'll end up with function names such as `_uploadImage_metal_T` or `_uploadImage_others_F`.

```
Future<void> _uploadImage_plastic_F(File imageFile) async {
  try {
    String fileName = DateTime.now().millisecondsSinceEpoch.toString();
    FirebaseStorage storage = FirebaseStorage.instance;
    Reference ref = storage.ref().child('Wrong/Plastic_F/$fileName');
    UploadTask uploadTask = ref.putFile(imageFile);
    await uploadTask.whenComplete(() => print('Image uploaded successfully'));
  } catch (e) {
    print('Error uploading image: $e');
  }
}
```

To be Saved in Firebase (images that are plastic, but the model presented the user with can  
incorrect prediction)  
[Figure 4.3.6]

```
Future<void> _uploadImage_plastic_T(File imageFile) async {
  try {
    String fileName = DateTime.now().millisecondsSinceEpoch.toString();
    FirebaseStorage storage = FirebaseStorage.instance;
    Reference ref = storage.ref().child('Correct/Plastic_T/$fileName');
    UploadTask uploadTask = ref.putFile(imageFile);
    await uploadTask.whenComplete(() => print('Image uploaded successfully'));
  } catch (e) {
    print('Error uploading image: $e');
  }
}
```

To be Saved in Firebase (images that are plastic and are predicted correctly as plastic)  
[Figure 4.3.7]






The images are then sent to their respective folders, which manifests in the database as two main directories, denoting either correctly or incorrectly predicted, with several sub-folders containing all the other different classes.

gs://tinity-73d37.appspot.com				Upload file	+	⋮
<input type="checkbox"/>	Name	Size	Type	Last modified		
<input type="checkbox"/>	Correct/	—	Folder	—		
<input type="checkbox"/>	Wrong/	—	Folder	—		

gs://tinity-73d37.appspot.com > Correct				Upload file	+	⋮
<input type="checkbox"/>	Name	Size	Type	Last modified		
<input type="checkbox"/>	Glass_T/	—	Folder	—		
<input type="checkbox"/>	Metal_T/	—	Folder	—		
<input type="checkbox"/>	Others_T/	—	Folder	—		
<input type="checkbox"/>	Paper_T/	—	Folder	—		
<input type="checkbox"/>	Plastic_T/	—	Folder	—		

gs://tinity-73d37.appspot.com > Correct > Paper_T				Upload file	+	⋮
<input type="checkbox"/>	Name	Size	Type	Last modified		
<input type="checkbox"/>	 1682096346047	128.23 KB	image/jpeg	Apr 21, 2023		
<input type="checkbox"/>	 1682096427986	120.04 KB	image/jpeg	Apr 22, 2023		
<input type="checkbox"/>	 1682321166557	82.37 KB	image/jpeg	Apr 24, 2023		
<input type="checkbox"/>	 1682323739862	82.57 KB	image/jpeg	Apr 24, 2023		
<input type="checkbox"/>	 1682323765549	82.12 KB	image/jpeg	Apr 24, 2023		

To be Saved in Firebase (images that are plastic, but the model presented the user with can incorrect prediction)  
[Figure 4.3.6]

## 4.4 Mobile Model Results

Preliminary evaluation of our transformed model is done using two methods, the first being calculating the F1 score and the second being its consistency value. Currently, our F1 Score is calculated by using an externally procured set of 40 images, while keeping the distribution between the different classes relatively similar; this set of images resulted in an F1 Score of 76%. However, this metric displays our accuracy in just one dimension, which introduces the need for another metric. The consistency metric allows for the quantification of the likelihood that our model will produce the correct results when viewing the same object. Using the same test set, our model was able produce a consistency rate of 67.44%. While the statistic isn't high, there are patterns within the consistency rate between the different objects; this pertains to some objects not being within the current dataset, causing the model to infer a prediction based on the current set. These inferences caused some of the image's predictions to fluctuate significantly, resulting in a drop in the overall

rate. The consistency between individual objects therefore is either extremely low or high, with the final rate being an average of the two.



F1 Score: 76%  
Test Set Confusion Matrix  
[Figure 4.4.1]

## **Chapter 5**

### **Conclusions**

#### **5.1 Summary of Accomplishments**

We were able to build on-top of our prior work in the first semester and to construct a Flutter application to house our previously created machine learning model. The mentioned application receives an image as its input; the image can either be taken from the user's gallery or freshly taken by the user's camera. Subsequently, a prediction of the picture can be made and displayed to the user, while a copy of the image is saved into an external database for future training if user consent is given. Although the current version of the app is not in any of the stores, our prototype has been successfully installed and ran on both Android and IOS devices for testing purposes.

#### **5.2 Issues and Obstacles**

The only major issue we've faced in this semester is the PyTorch\_mobile library just blatantly not working, which proved to be a massive hassle to fix and is something that we could not sidestep seeing as having a working model in our application is one of the criteria we'd have to meet in order to achieve a bare minimum product. At first we had operated on the assumption that all the libraries would work and that it was just a matter of assembly. The issue with PyTorch-mobile was that its last update and maintenance cycle was around eight months old, meaning that any bugs introduced with newer versions of Dart would go unfixed. The problem encountered was that predictions made with the model imported using this package would return the same result regardless of the base image. While it was a setback, our team didn't think that it would be that difficult to find a solution to our current predicament, as there was a well-documented way to transform the PyTorch file format into TensorFlow; the transformation was decided upon because we saw that the TensorFlow library for Flutter was much more well-maintained than the one associated with PyTorch.

At first, this approach seemed to work with the transition being done without any major obstacles. However the file we ended up with would still not work. Through even more tests, one of our group members discovered that the reason that the model would just make haphazard predictions comes down to the formatting of the input tensor. A model generated by PyTorch follows the NCWH format, each of the letters meaning batch size, channel number, width, and height respectively. On the other hand, TensorFlow runs on the NWHC format, meaning while the image can be passed through to the model, the model would be trying to make predictions using unrelated data, which resulted in the random predictions generated. A fix was eventually found by permuting the input parameters within the original PyTorch file before transforming it into the TensorFlow format, which resolved this issue altogether.

Another roadblock was encountered when our code base was moved into Xcode, which came in the form of a discontinued library that was unable to successfully interface with some of the newer libraries. To fix it, individual dependencies of the discontinued library had to be manually over-ridden, which while not optimal does allow for the deprecated library to function.

#### **5.3 Future Directions**

This will likely be the farthest our group will go in working on this project, as every member of the group has decided that they would like to pursue other project ideas in the upcoming

semester. Due to this, any further development will be halted for the foreseeable future, however the codebase of the project will be stored on GitHub along with the dataset used in training the model used [32]. Although, there are still several possible improvements that could be made to the project, including issues with the dataset which have been minimized somewhat but are still present. Furthermore, future work in transitioning the project into a form where the model is hosted on a server should be considered. While this may introduce more moving components into the system, as a request system will have to be implemented to handle retrieving images from the user device and sending out prediction statuses from the model back to said device, it does provide us with the capability to make constant updates to the model without needing the user to update the application on their end.

The Dart codebase for the mobile application could and should be given another cycle of reviews to trim down the number of lines needed along with removing parts of the code that are redundant. Lastly, further optimizations can be made to the model itself by experimenting with the size of the input images. Overall, it is definitely a project worth developing due to most of the current solutions out there either still being in their infancy or only targeting specific areas.

## **5.4 Lessons Learned**

The lessons learned this semester stem from the obstacles we had to tackle, in this case most of it comes from the package issue that ended up costing us weeks of time diverted to find a solution to this problem. Therefore, use official libraries or packages whenever possible as those are the ones that are most likely to be well maintained. Furthermore, version histories of prospective third party packages should always be checked to ensure that they are up to date. The other lesson is that it never hurts to ask, the current solution in use to the issue involving the input dimensions was gleamed from asking a graduate student teaching machine learning about the issue; this problem ended up costing us a significant amount of time, but it would have been even more lengthy without his contribution. The last one is to try and use all the resources you have at hand; an expert specializing in a field that you are currently working on can and should be logged as a potential person to consult if the project is stuck at a bottleneck.

## 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: .
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: .
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: .
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](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.
15. Trashbot: The smart recycling bin that sorts at the point of disposal (2023) CleanRobotics. Available at: <https://cleanrobotics.com/trashbot/> (Accessed: April 17, 2023).
16. Recycle mate (no date) Recycle Mate. Available at: <https://recyclemate.com.au/> (Accessed: April 17, 2023).
17. Beautiful Soup Documentation - Beautiful Soup 4.4.0 documentation. Available at: <https://beautiful-soup-4.readthedocs.io/en/latest/> (Accessed: April 21, 2023).

18. Inc., A. (no date) Swift, Apple Developer. Available at: <https://developer.apple.com/swift/> (Accessed: April 21, 2023).
19. Get started with Kotlin: Kotlin (no date) Kotlin Help. Available at: <https://kotlinlang.org/docs/getting-started.html> (Accessed: April 21, 2023).
20. Get started fast (no date) Apache Cordova. Available at: <https://cordova.apache.org/> (Accessed: April 21, 2023).
21. HTML: Hypertext markup language (no date) MDN. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTML> (Accessed: April 21, 2023).
22. CSS reference – CSS: Cascading style sheets: MDN (no date) CSS: Cascading Style Sheets | MDN. Available at: <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference> (Accessed: April 21, 2023).
23. JavaScript (no date) MDN. Available at: <https://developer.mozilla.org/en-US/docs/Web/javascript> (Accessed: April 21, 2023).
24. Pytorch\_mobile: Flutter Package (2022) Dart packages. Available at: [https://pub.dev/packages/pytorch\\_mobile](https://pub.dev/packages/pytorch_mobile) (Accessed: April 21, 2023).
25. Firebase documentation (no date) Google. Google. Available at: <https://firebase.google.com/docs/> (Accessed: April 21, 2023).
26. The developer Data Platform (no date) MongoDB. Available at: <https://www.mongodb.com/> (Accessed: April 21, 2023).
27. Hollands, M. (2016) Amplify, Amazon. Mark Hollands. Available at: <https://aws.amazon.com/amplify/> (Accessed: April 21, 2023).
28. Tensorflow Lite: ML for Mobile and edge devices (no date) TensorFlow. Available at: <https://www.tensorflow.org/lite> (Accessed: April 21, 2023).
29. Tflite\_flutter: Flutter Package (2023) Dart packages. Available at: [https://pub.dev/packages/tflite\\_flutter](https://pub.dev/packages/tflite_flutter) (Accessed: April 21, 2023).
30. Onnx (no date) Onnx/onnx: Open standard for machine learning interoperability, GitHub. Available at: <https://github.com/onnx/onnx> (Accessed: April 21, 2023).
31. Camera: Flutter package (2023) Dart packages. Available at: <https://pub.dev/packages/camera> (Accessed: April 26, 2023).
32. Minniesse (2023) Trinity/install.bat at Main · Minniesse/Trinity, GitHub. Available at: <https://github.com/Minniesse/Trinity/blob/main/install.bat> (Accessed: April 22, 2023).
33. Documentation (no date) Homebrew Documentation. Available at: <https://docs.brew.sh/> (Accessed: April 27, 2023).