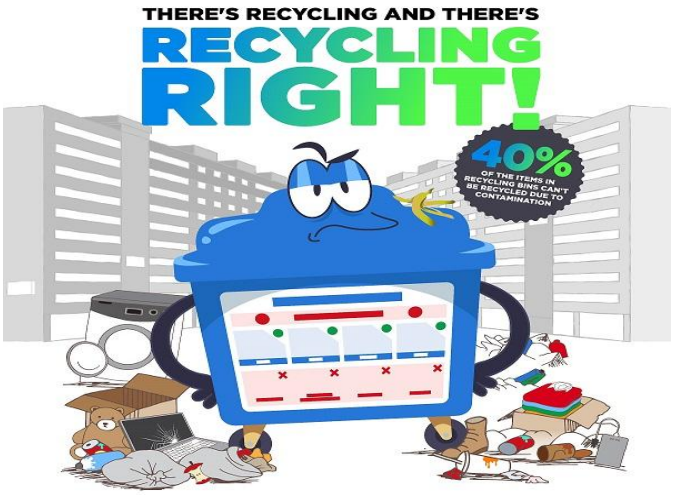


Smart Bin

Recycling Image Classification using neural networks

Group 5

Namrata Thakur	A0261619B
Ouyang Hui	A0261875U
See Jia Fong Grace	A0261797M
Wang Zhipeng	A0261980Y

Table of Contents

1.Introduction

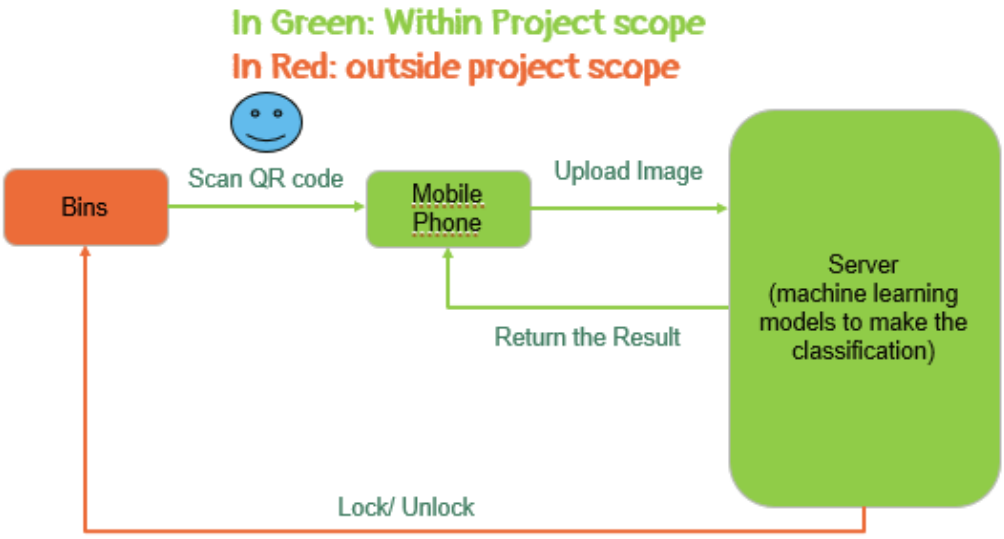
Singapore has been trying to raise recycling rates to promote sustainability. NEA has put in place 1 recycling bin for every HDB block to ensure accessibility to recycling channels. However domestic recycling rates remain low. The problem is further compounded by the fact that about 2 out of every 5 collected bins are contaminated. Contamination happens in the form of residents dumping non-recyclables or polluting substances into the recycling bins. Non-recyclables may be able to be separated out at a sorting facility. However, if polluting substances (such as leftover food and drink) are thrown into the bin, this may render the rest of the contents in the bin unrecyclable. This wastes the recyclables in the bins and effort to collect and sort them.

1.1. System Aim

The aim of the system is to build an **image classifier** that can differentiate between **clean recyclables, dirty recyclables and non recyclables**. The idea is that if the image classifier works well, we can eventually prevent dirty or non recyclables from entering the bin to contaminate other clean recyclables that are already present.

1.2. Project Scope

The scope of the project will be to build and demonstrate the image classifier. It will exclude the instrumentation of the Recycling bin.



2. Tools and Techniques

2.1. Data Crawling and Processing

We did a review of the available datasets and found that there were no datasets which addressed our problem directly. The ones on Kaggle/ Github were targeting problems such as material classification, identification of overflowing bins, garbage in the wild et cetera, but none were targeted toward determining the cleanliness of recycled items. We also contacted Recycleye to see if we could use their WasteNet images, but they did not get back to us. Hence we decided to crawl our own images.

Data was crawled from Google using a multithreaded, open source image crawler (CygnusX1).

We used as reference to a list of items curated by the National Environmental Agency (NEA) detailing whichitems were acceptable for recycling, and which ones were not. This list of items was further expanded in a keyword search to help gather more images for training. The complete list of keywords used can be found in the appendix.

After the data had been crawled, they were renamed for easy identification using Bulk Rename Utility, which is another freeware, and broadly classified into either “clean recyclable”, “dirty recyclable” or “non recyclable”

Next the dataset was split into 4 parts so that the group could collectively help review parts of the data. The aim of this review was to:

a. Remove all data of invalid formats (e.g. partially downloaded data, webpage data etc)

b. Ensure that the data sufficiently represented the keyword in real life. Images like infographics, or vector art were deleted. Images that were potentially confusing e.g. a clean container and a dirty container presented in picture in equal proportion, were deleted. However, if for example, the clean container was in the foreground and much larger and clearer than the dirty containers in the background, it would be retained

c. Ensure that images were correctly classified under the correct category. Sometimes looking for "clean" things, e.g. "clean PET bottle", would turn up "dirty PET bottle" instead. These had to be identified and reclassified. Some things required a bit more interpretation. For example, it was quite hard to tell the cleanliness of opaque objects like metal cans. Hence these were given the benefit of doubt and classified to "clean recyclable" because the software would be too restrictive otherwise, resulting in many metal cans being discarded instead of recycled. These still posed the risk of contamination but it was difficult to visually diagnose the contents in the can, hence potentially better way could be to have a bin design that could either weigh the can, or accept cans inverted to ensure no contents left etc. Also, in terms of classification, things like paper would usually be marked as "clean recyclable". However, dirty paper should not be regarded as "dirty recyclable" but "non recyclable" instead, as it is quite hard to clean dirty paper such that it reverts into a "clean recyclable" state.

After the entire cleaning process we have an image dataset of **41K images** (***entirely crawled and cleaned from scratch***). Dataset statistics are as follows:

Category	Before Cleaning	After cleaning
Clean Recyclables	21,157	14,421
Dirty Recyclables	8,140	10,192
Non Recyclables	34,270	16,305

Total Images	63,567	40,918

2.2. Data Preprocessing

Data preprocessing for model training included the following tools and techniques.

Technique	Library/ Method	Done for	Description
Check valid image extensions	None, self coded	Resnet	Done as part of preprocessing for resnet training
Unzip images into Colab environment	None, self coded	Resnet	Unzipping into Colab environment allowed for faster access to images as opposed to accessing from Drive
Training/ Validation/ Test set	Split-Folders	Ensemble CNN	Python library was used to split the dataset into training, validation and test set in the ratio (0.7, 0.1, 0.2)
Bootstrap data	Pandas	Ensemble CNN	Pandas dataframe sample with replacement. Two-thirds of the dataset was used for each model For more details, please refer to Ensemble CNN section
Image Augmentation	Keras ImageDataGenerator	Ensemble CNN	Attempted but not used finally due to low improvements
Image Resize	Tensorflow Image Resize method	Resnet/ Ensemble CNN	Final input dimensions for all images (200, 200, 3) 200x200 px was chosen because it seemed like the average, some crawled

			pictures were less than 200 on 1 or both axis

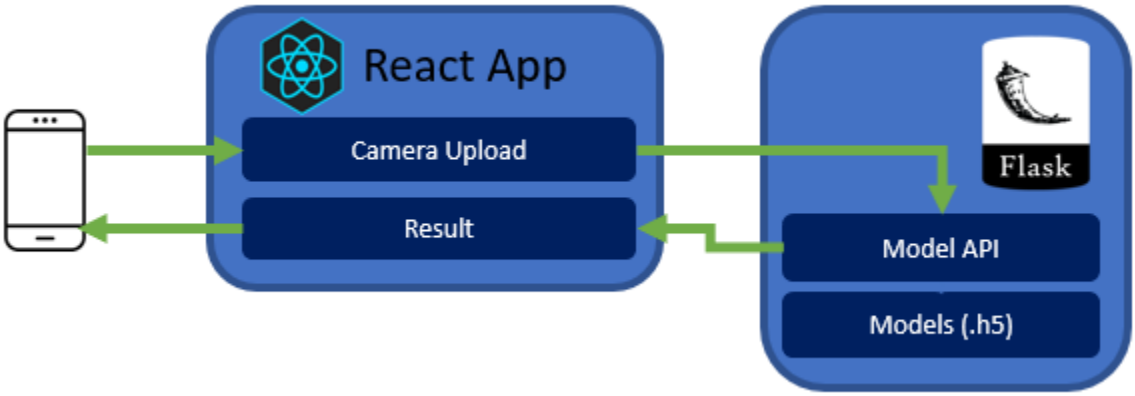
2.3. Model Training

Resnet was trained on Google Colab, while the CNN models were trained locally on Jupyter Notebooks running on NVIDIA GPUs. The resultant models were exported in .h5 format for deployment.

3. System Features

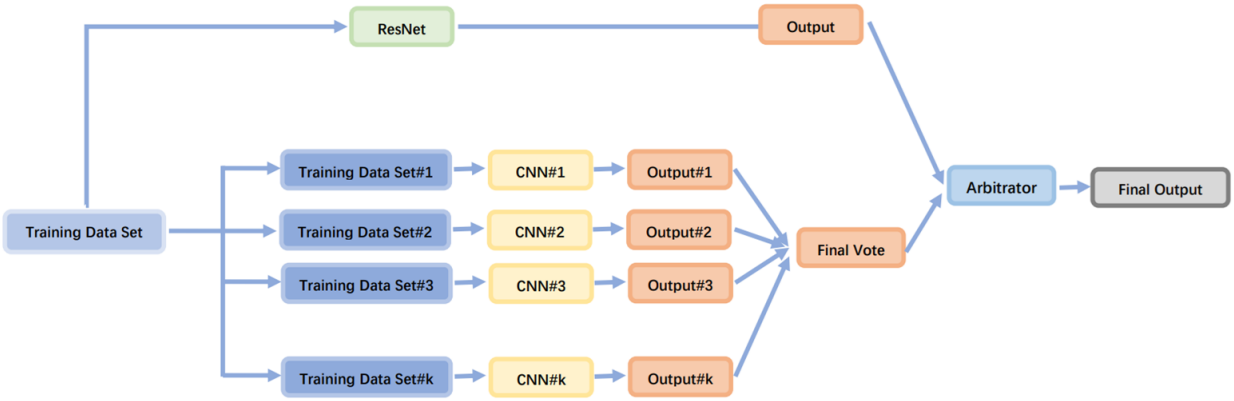
3.1. System Architecture

.



The system consisted of a React App which opened a camera in the user's phone when they navigated to the website. The user could take a picture and it would be submitted to the flask backend, which queried the models to get predictions. The predictions from the model endpoints were aggregated (see Section 3.2), and the final result was displayed to the user

3.2. Hybrid Architecture



Our solution had 2 stages. We built a residual CNN on the entire clean dataset as Approach 1. We also built an ensemble model with EfficientNet as base learners as Approach 2. The final output was the aggregated average of the two approaches.

We chose to design our system this way because we wanted to experiment with both building our models from scratch and transfer learning with pre-trained model. Since transfer learning involved much less work, we decided to ensemble the pretrained models to see if we could get better results.

4. Residual Neural Network

4.1. Model Introduction

Residual Net has been very popular in recent times due to the immense power it brings to the model performance. However, training a deep residual network from scratch is pretty challenging mainly because it tends to overfit (if the architecture is too complex). So, major real world applications use pretrained ResNet34 or ResNet50 etc to fit on their custom dataset. Since we had already decided to apply transfer learning for the ensemble, we tried training a residual network from scratch to solve our classification problem.

4.2. Model Construction

We designed a 12 layer vanilla residual neural network that has been trained on the entire train dataset. Residual architecture has two main parts: An Identity Block and a Convolutional Block.

A) Identity Block:

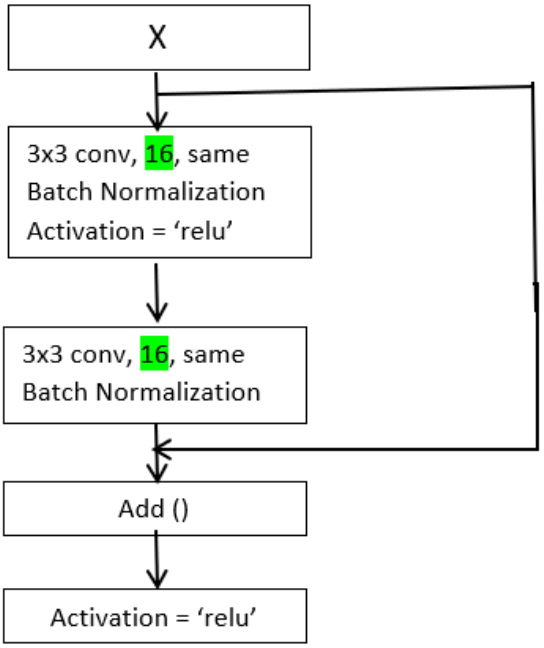
The identity block directly adds the residue to the output. All convolution layers were padded to maintain the shape of the input until it is merged with the residue.

For example: Say we have an input X = [50,50,16]. After 2 Convolution Layers with 16 filters and padding, the dimensions of output X_new should also be [50,50,16].

X_new is then added with the residue which was the same as the incoming image size (i.e. [50,50,16]). If we used padding='valid' then the shape of the two tensors would be different

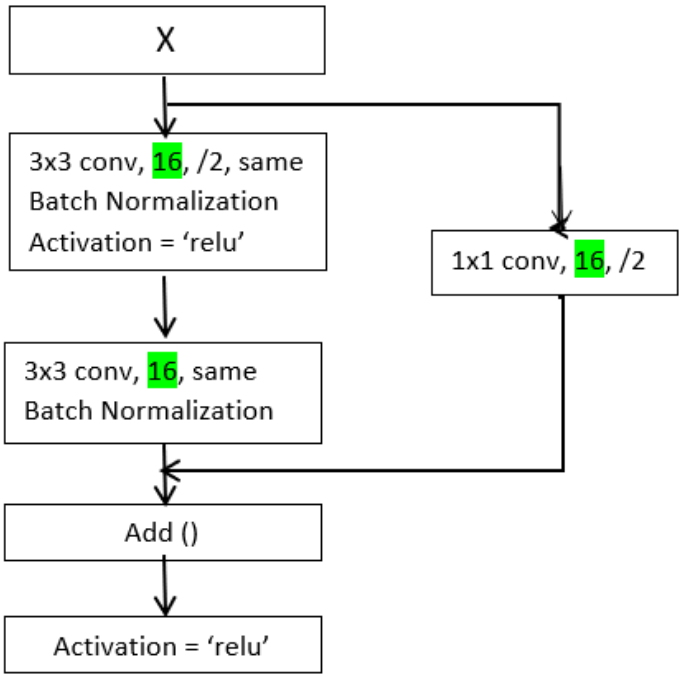
leading to an error as element wise addition operations could not be performed on arrays of different sizes.

Below is the image of the identity block construction. The GREEN marked part is the number of kernels. We will be reusing this structure for all the blocks, and we keep on changing the kernel size from 16 to 32.



B) Convolutional Block:

For the convolution block, the residue is not directly added with the output. It first passes through a 1x1 convolution block before getting added to the output. A stride of 2x2 is also used to downsample the image size. The strides are used on both paths to maintain the same image size after downsampling so that we can add the tensor later.



FINAL ARCHITECTURE:

Using the above identity and convolution block, we designed the below Residual architecture.

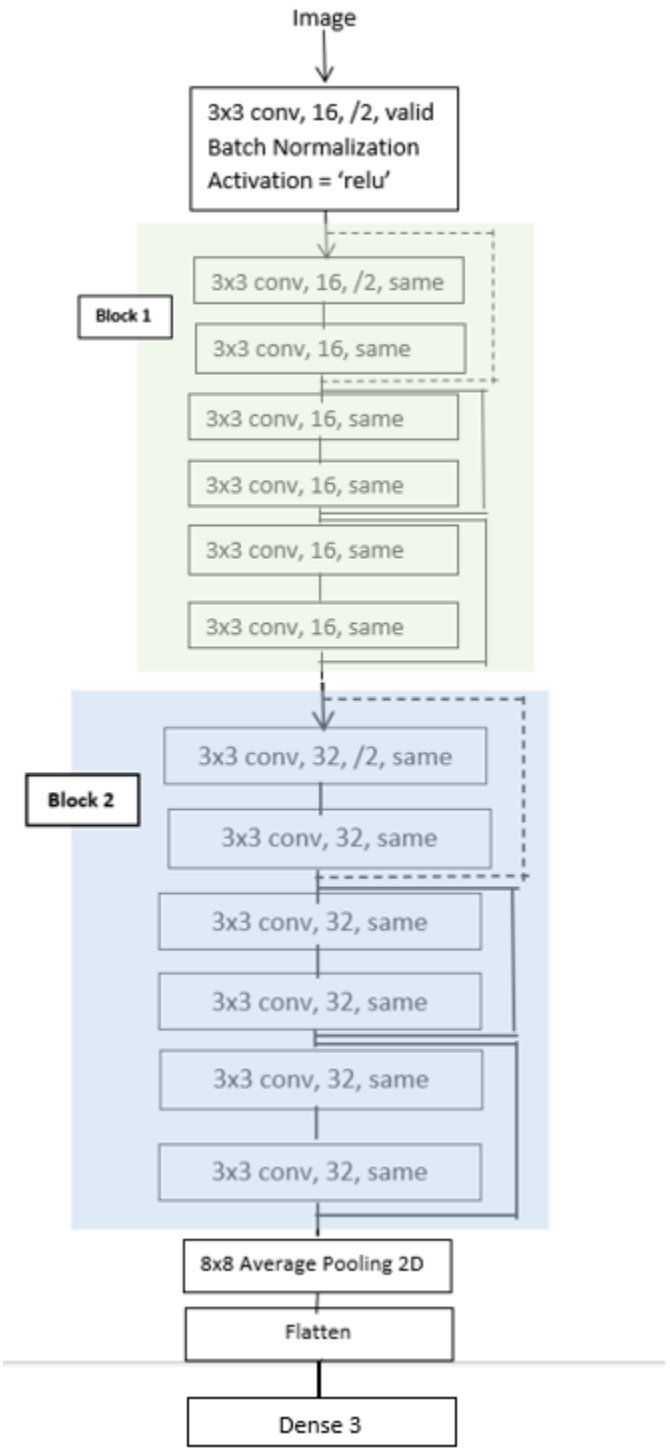


Image
3x3 conv, 16, /2, valid
Batch Normalization
Activation = 'relu'
Block 1
3x3 conv, 16, /2, same
3x3 conv, 16, same
3x3 conv, 16, same
3x3 conv, 16, same
3x3 conv, 16, same
3x3 conv, 16, same
Block 2
3x3 conv, 32, /2, same
3x3 conv, 32, same
3x3 conv, 32, same
3x3 conv, 32, same
3x3 conv, 32, same
3x3 conv, 32, same
8x8 Average Pooling 2D
Flatten
Dense 3

As shown in the image, we used **2 residual blocks** (shown in green and blue). Each block contains **3 sub-blocks** - ***1 convolution block followed by 2 identity blocks***. The convolution blocks are shown with --------- line. The identity blocks are shown with ——— line.

The initial input size is [200,200,3] as we are dealing with RGB images. The image is first passed through a 3x3 convolution layer with 16 kernels and padding as 'valid' with strides of 2. The convolution layer is followed by a batch normalization and relu activation layer. The output tensor size of the first convolution layer is [99,99,16]. This tensor is flown into the first residual block. The first residual block is using 16 kernels of size 3x3 for each convolution layer. The output of the first residual block is [50,50,16]. This tensor is then flowing into the second residual block that is using 32 kernels of size 3x3 for each convolution layer. The output of the second residual block is [25,25,32]. We are passing this tensor to an averagePooling layer of pool_size = 8 and followed by flatten layer. The flatten layer has 288 parameters that finally flows into the final output dense layer containing 3 softmax nodes for 3 classes.

We have a total **trainable parameters of 67,539 and 608 non-trainable parameters**. We tried to keep the parameters less so that it is easier to fit the model properly and overfitting is avoided.

4.3. Model Tuning & Results

We used data augmentations, learning rate scheduler, early stopping and model checkpoint to tune the network.

4.3.1. Data Augmentation

To increase the training dataset diversity, we applied online data augmentation using tf.image. We wanted to apply a different set of augmentations to each image during training. Hence, we created 5 different categories of augmentations and then applied each category randomly on each image. For example: Category 1 contained 3 augmentations like `stateless_random_flip_left_right`, `stateless_random_contrast and central_crop`. Category 2 contained another set of 3 other augmentations like `stateless_random_contrast`, `adjust_saturation`, `and stateless_random_brightness`. In this way we created 5 such categories.

During training, for every image for every epoch a random number is generated from 1 to 5. The category is then selected based on the random number generated and the corresponding augmentations are then applied to the incoming image.

In this way, we made sure that augmentations applied on each image are different and the model get to experience a diverse set of images during training. Alongside, we used the tf.data to create the pipeline for applying augmentations, repeating and shuffling the dataset (during training).

4.3.2. Learning Rate Scheduler

We used **Adam** as the optimizer with initial learning rate as **0.001**. We then decreased the learning rate by a factor of **1e-1** with increase of 15 epochs (for the first updation) and every 10 epochs for the rest of the training time. This managed to reduce the overfitting. Since we are progressively reducing the learning rate after every interval of 10-15 epochs, we needed a relatively higher learning rate as the initial value. A very low initial learning rate (for example: 0.0001) made the training process extremely slow and overfitting is noticed after once or twice updation.

4.3.3. Early Stopping

We used the Keras callback of EarlyStopping to monitor the validation accuracy at the end of every epoch. If the change in **validation accuracy** is **less than 0.0001** for **6 continuous epochs** then we stopped the training and kept the model with the last best weights update.

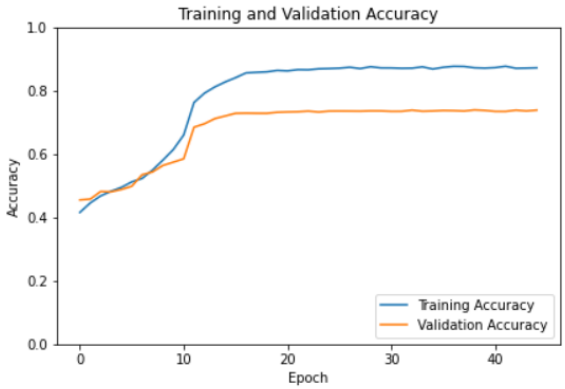
4.3.4. Model Checkpoint

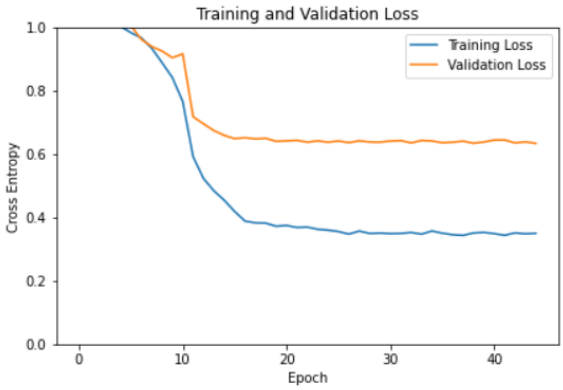
We created a model checkpoint to monitor the validation accuracy for every epochs. In case there is an increase in the metric value, the callback will save the model weights in an .h5 file. These callbacks helped to keep the latest model weights saved in the drive in case of a session disconnection.

4.3.5 Adding more residual block

We tried adding ***1 more residual block of 32 kernels***. The trainable parameter size increased to 123,699 and non trainable parameters increased to 992. Addition of an

extra residual block decreased the overfitting as evident in the below chart of Epoch vs Loss and Epoch vs Accuracy.





But this updation in the architecture didnt bring significant change in the performance metric values for the validation data. To be specific, it performed worse for dirty recyclable class where the precision, recall and f1 score fell (compared to model with 2 residual block).

Below are the classification report and confusion matrix for the model with 3 residual block.

Classification Report

```
Best accuracy (on validation dataset): 73.97%
                  precision    recall  f1-score   support

  non-recyclables     0.7555    0.7800    0.7675      3145
clean-recyclables     0.7001    0.8131    0.7523      2664
dirty-recyclables     0.7917    0.5700    0.6628      1900

         accuracy                         0.7397      7709
        macro avg     0.7491    0.7210    0.7276      7709
     weighted avg     0.7452    0.7397    0.7365      7709
```

Confusion Matrix

True \ Predict	Non Recyclable	Clean Recyclable	Dirty Recyclable
Non Recyclable	**2453**	528	164
Clean Recyclable	377	**2166**	121
Dirty Recyclable	417	400	**1083**

Detecting dirty recyclable is harder as compared to non recyclable and clean recyclable. Alongside, the main aim of our project was to reduce contamination within recyclables by removing dirty recyclables at the source. Therefore, we put more importance on high precision and recall for dirty recyclable than others. Compared to the model with 2 residual block, we see a reduction in correct classification of dirty recyclable by 200. That is a significant number considering dirty recyclable is a minority class. Hence, we didnot include this model for the final deployment.

4.4. Model Performance

We decided on the model with 2 residual blocks as the final architecture. We trained the model for **60 epochs**. We didn't train for longer epochs as we started to see the model getting overfitted for longer train time. The training finished at 35 epoch due to EarlyStopping callback as the validation accuracy didnt improve for more than 6 epochs.

The overfitiing issue is still there, but this model performed the best as compared to other versions. Below is the classification report and confusion matrix for 2 residual block architecture.

Classification Report

```
Best accuracy (on testing dataset): 76.79%
                   precision    recall  f1-score   support

  non-recyclables     0.7863    0.8156    0.8007      3145
clean-recyclables     0.7313    0.8450    0.7840      2664
dirty-recyclables     0.8064    0.5811    0.6754      1900

         accuracy                         0.7679      7709
        macro avg     0.7747    0.7472    0.7534      7709
     weighted avg     0.7723    0.7679    0.7641      7709
```

Confusion Matrix

True \ Predict	Non Recyclable	Clean Recyclable	Dirty Recyclable
Non Recyclable	**2451**	407	287
Clean Recyclable	430	**1981**	253
Dirty Recyclable	320	297	**1283**

The model performed better than all other versions on the validation data. It has been able to detect the highest number of dirty recyclable among all other versions that we tried.

The class probabilities are shown as list with the ***first probability*** belonging to ***class 'non recyclable'***, ***second probability*** to ***class 'clean recyclable'*** and ***third probability*** to ***class 'dirty recyclable'***.

Below are some of the images on which the model predicted accurately.

```
classify_output1 = pred.Predict('Test Images/xbox_2.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 34ms/step
[[0.83820987 0.16016127 0.00162896]]
```

```
classify_output1 = pred.Predict('Test Images/plastic_egg_tray_241.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 31ms/step
[[1.8957403e-04 4.2502912e-09 9.9981040e-01]]
```


```
classify_output1 = pred.Predict('Test Images/clean_crumpled_pet_bottle_61.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 67ms/step
[[1.5267455e-04 4.6028454e-02 9.5381886e-01]]
```

```
classify_output1 = pred.Predict('Test Images/leftover_bubble_tea_347.jpeg')
print(classify_output1)
```

```
WARNING:tensorflow:5 out of the last 12 calls to <function Model.make_predict_fun
1/1 [==============================] - 0s 316ms/step
[[0.12965584 0.00169269 0.86865145]]
```


```
classify_output1 = pred.Predict('Test Images/bodywash_bottle_257.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 56ms/step
[[0.30077907 0.68838    0.01084085]]
```

```
classify_output1 = pred.Predict('Test Images/aerosol_can_85.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 309ms/step
[[1.1544701e-02 9.8840618e-01 4.9111502e-05]]
```

```
classify_output1 = pred.Predict('Test Images/clean_plastic_container_266.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 301ms/step
[[0.19479132 0.7994439  0.0057648 ]]
```

```
classify_output1 = pred.Predict('Test Images/empty_condiment_bottle33.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 34ms/step
[[0.15989901 0.66281277 0.17728823]]
```

4.5. Limitations and Improvements

Before stating any limitations of the model, lets first check out the failure cases. Understanding the failure cases are important to understand where the model is lacking and what are the appropriate corrective steps.

Below are some of the images where the model predicted wrongly.

```
classify_output1 = pred.Predict('Test Images/dirty_aluminium_foil_8.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 31ms/step
[[9.3927133e-01 6.0702309e-02 2.6343831e-05]]
```

```
classify_output1 = pred.Predict('Test Images/contaminated_food_container_105.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 32ms/step
[[0.7794063  0.21603711 0.00455658]]
```

```
classify_output1 = pred.Predict('Test Images/dirty_drink_bottle_208.jpeg')
print(classify_output1)
```

```
1/1 [==============================] - 0s 68ms/step
[[0.7812333  0.21621946 0.00254731]]
```

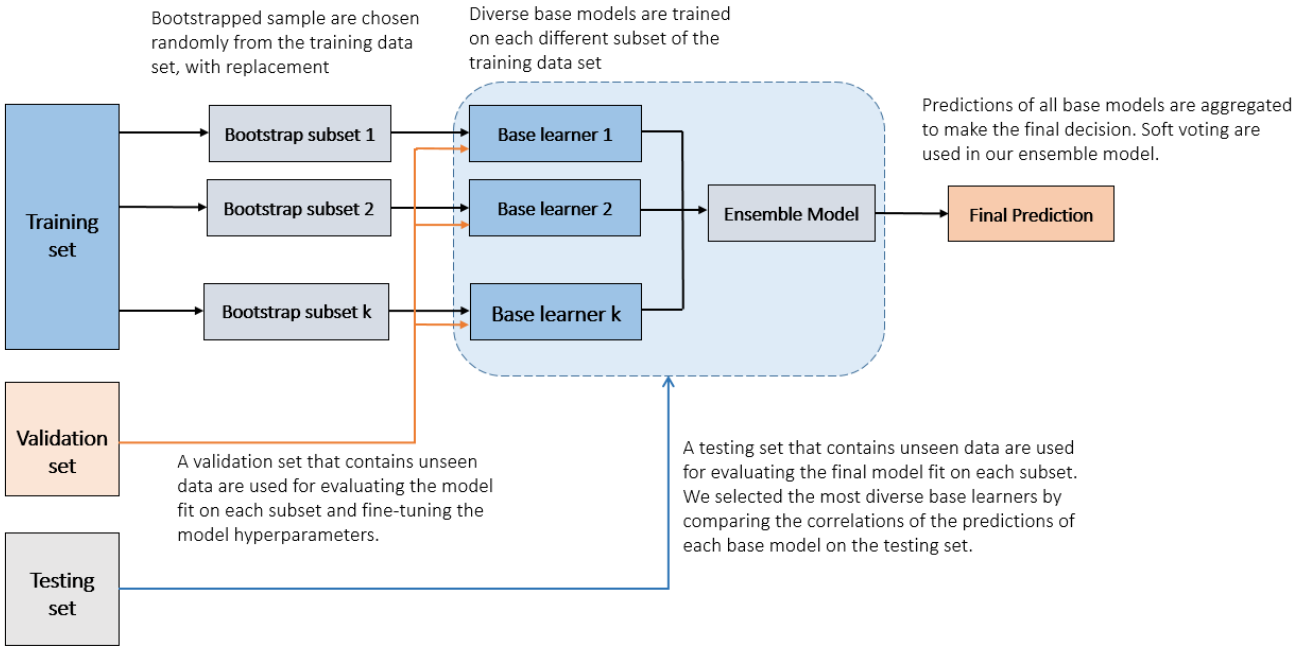
Some of the limitations and possible improvements are:

- Its performance on the dirty recyclable is still not extremely good. Most misclassifications are happening for this class. As a possible remedial, we need to collect more diverse images for the dirty recyclable class. Alongside, train the model further to reduce the overfitting that will reduce such misclassification.
- Perform better augmentations to diversify the dataset that the model sees.
- Tune the learning rate more aggressively to control overfitting.
- Increase the depth of the architecture without overfitting. That would help to extract finer details needed to classify dirty recyclables.
- Clean the dataset more vigorously as there are still some noise that is present.

5. Ensemble CNN

5.1. Model Introduction

In this project, we constructed our ensemble model by using bagging technique. Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. Bagging was done in three stages: bootstrapping, selecting optimal base learners and aggregation using soft voting.

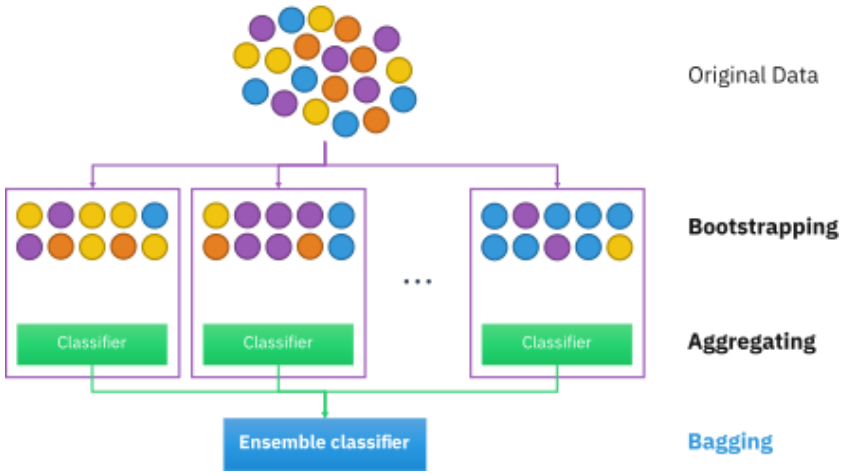


5.1.1 Train/ Validation/ Test Split

Before performing bagging, the data set was split into training, validation and testing sets in the ratio 7:1:2. Each set had the 3 subfolders of labeled images corresponding to “Clean Recyclables”, “Dirty Recyclables” and “Non Recyclables”.

5.1.2 Bootstrapping

Bootstrapping technique was used to create data subsets for training the base learners. Individual data points were randomly chosen from the training set with replacement, to make every subsequent selection independent from the previous selections. For different models, we randomly select 2/3 of the total training set for training each time to obtain diverse base learners.



5.1.3 Base Learner Selection

The component learners of the ensemble which are combined together strategically are referred to as base learners. Base (weak) learners must focus on correctly classifying the most highly weighted examples while avoiding overfitting.

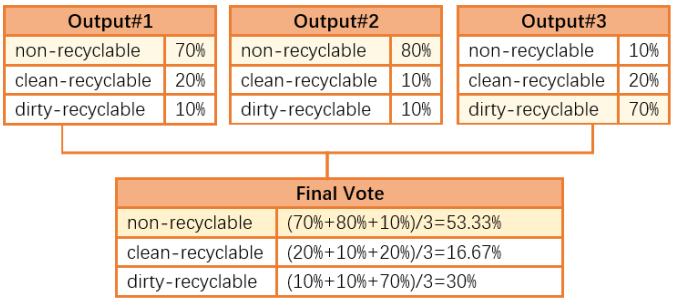
We tried 2 CNN models as candidates for the base learner to compare performance. EfficientNetV2 performed better than VGG16 as a base learner, which was in line with expectation as EfficientNetV2 has a higher reported test accuracy on ImageNet compared to VGG16.[1]

CNN Model Variant	Rationale for Selection	Model description
VGG16	We wanted a simple, shallow model with good runtime performance	VGG16 comprises of 5 blocks. Each block has 2 convolution layers (3x3 filter, stride 1, padded), each accompanied by its own ReLU activation layer, and 1 maxpool layer (2x2 filter, stride 2). At the end it has 2 fully connected layers (FC) followed by a softmax for output. This network is a pretty large network and it has about 138 million (approx) parameters.
EfficientNetV2B0/1/2/3	Relatively small model with good prediction performance	EfficientNet uniformly scales network width, depth and height with a compound coefficient. V2 is used because it is supposed to have better run time performance than the first version

5.1.4 Aggregation

We combine the multiple outputs of the classification by soft voting to derive the final vote.

[1] https://keras.io/api/applications/



5.2. Transfer Learning

5.2.1 VGG16 Modifications

In general, we modified the VGG16 using the two methods below:

Method	Description	Rationale
1	Replace Input layer to accept (200,200,3) instead of (224, 224, 3) Replace output layers with GlobalAveragePooling layer, several dense layers and one softmax layer.	Input size was different from VGG16's original VGG16 output up to 1000 categories but we only need 3
2	In addition to the changes in Method 1, we also added additional convolutional layers so that the model is able to learn on our dataset.	Adding the later convolutional layers preserves the training done for all the earlier layer but allows us to train the later layers for our use case (clean/dirty/non recyclable)

Results

The following table details the accuracy obtained from 4 different instances of VGG16. As you can see, there is a significant increase in accuracy after we unfroze the last two pretrained layers for training.

Model Name	Modification	Validation Accuracy
vgg_frozen	Froze all pretrained layers	0.578456072
vgg_frozen_1conv	Add 1 Conv layer after frozen pretrained layers	0.637890625
vgg_frozen_2conv	Add 2 Conv layers after frozen pretrained layers	0.640364583
vgg_unfrozen	Unfroze last 2 pretrained layers	0.705078125

5.2.1 EfficientNetV2 Modifications

Method	Description	Rationale
1	Replace Input layer to accept (200,200,3) instead of (224, 224, 3) Replace output layers with global average pooling layer, several dense layer and batch normalization layers and one softmax layer.	Input size was different from VGG16's original VGG16 output up to 1000 categories but we only need 3
2	In addition to the changes in Method 1, we also unfreeze the last 3 layers (convolution, batch normalization and activation) so that the model is able to learn on our dataset.	Unfreezing the later convolutional layers preserves the training done for all the earlier layer but allows us to train the later layers for our use case (clean/dirty/non recyclable)

Results

As per the VGG model, unfreezing later layers to facilitate model learning with our dataset yielded an increase as expected. EfficientNetV2B3 did particularly well, with almost 78% accuracy, which is much higher than the others. Other than this model, the other 3 EfficientNet variants did not perform significantly better than VGG, which was also surprising, since EfficientNetV2 does better on ImageNet than VGG.

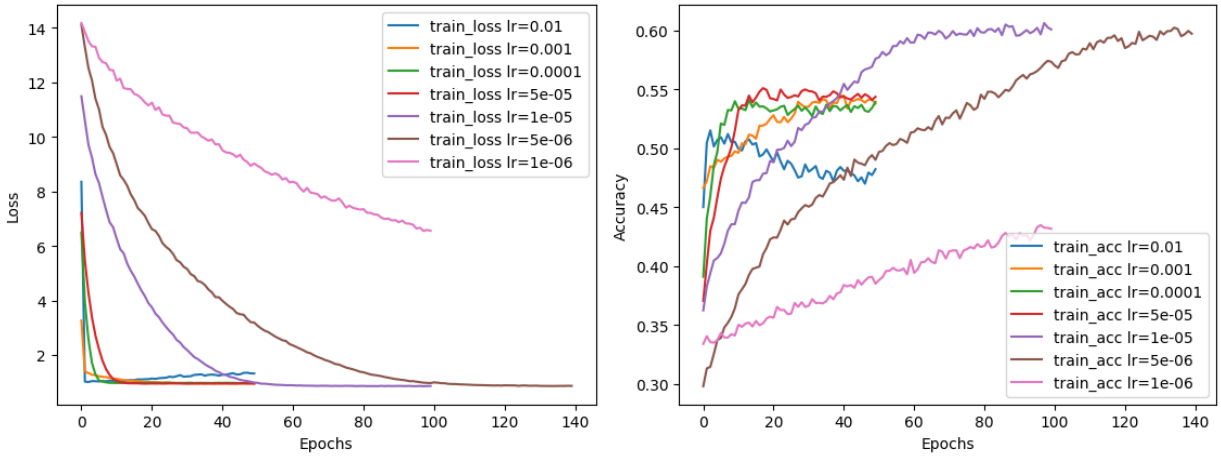
Model Name	Frozen Accuracy	Unfrozen Accuracy
EfficientNetV2B0	0.633478953	0.7012741264034
EfficientNetV2B1	0.639126373	0.7024094865649
EfficientNetV2B2	0.64287346	0.7165384130187
EfficientNetV2B3	0.70572916666	0.7796139775450

5.3. Model Tuning and Results

5.3.1. Learning rate

We experimented with different learning rates to see which one could give the best outcome. The optimal learning rate is identified by plotting graphs to compare the loss curve and accuracy

curve. For example, the following diagrams show the loss curves and accuracy curves of the model trained with different learning rates and Adam optimizer. The rapid decline of the curve indicated that learning rate 0.01, 0.001, 0.0001 and 5e-5 were too high for training this model, and their accuracy only reached around 55%. And the loss curve of learning rate 1e-6 failed to converge, which showed that learning rate 1e-6 was too low. It is indicated that curves of learning rate 1e-05 and 5e-06 managed to converge within a reasonable time. However both accuracy reached around 63% but training with learning rate 5e-06 will take longer time than 1e-5. Considering the time cost of training, 5e-06 is regarded as a better learning rate.



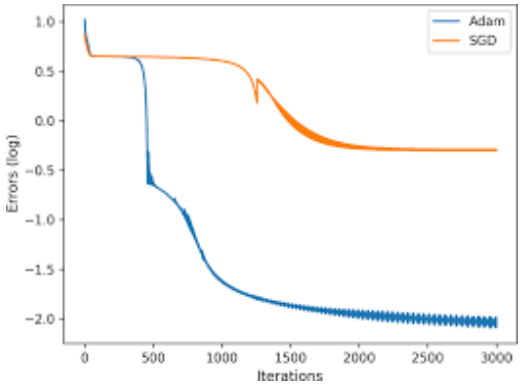
5.3.2. Optimizers

5.3.2.1. Stochastic Gradient Descent

Stochastic Gradient Descent is the most basic form of Gradient Descent. SGD subtracts the gradient multiplied by the learning rate from the weights. Despite its simplicity, SGD has strong theoretical foundations and is still used in training edge NNs.

5.3.2.2. Adam

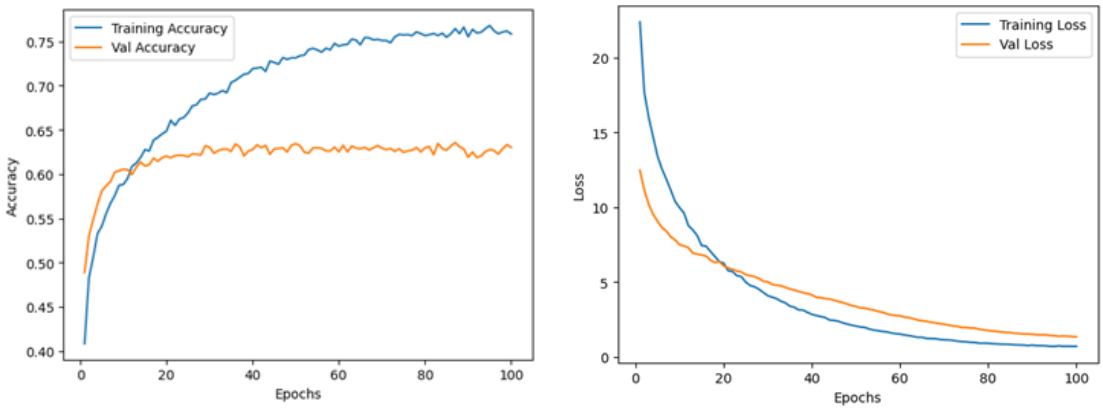
Adam is based on RMSProp but estimates the gradient as the momentum parameter to improve training speed. According to the experiments we have made, Adam outperformed all other methods in various training setups. Adam has become a default optimization algorithm regardless of fields.



5.3.3. Flatten vs GlobalAveragePooling2D

To reuse pretrained models, both Flatten and GlobalAveragePooling2D can be applied before feeding into a custom fully-connected layer. In this project, we examined the differences between Flatten and GlobalAveragePooling2D by comparing the training performance of

models reusing VGG16. It was observed that the model with Flatten implemented had an overfitting issue.



This may be because Flatten and GlobalAveragePooling2D represent features in different ways. Flatten will take the tensor of any shape and convert it into one dimensional tensor, while GlobalAveragePooling2D reduces spatial dimension by averaging the features and mapping to one. For example, given input size `(batch_size, height, width, channels)`, Flatten layer will have output with shape `(batch_size, height*width*channels)`, but the output shape of GlobalAveragePooling2D will be `(batch_size, channels)`. Therefore Flatten may result in a larger Dense layer which is more computationally expensive and may lead to overfitting (Verma, 2021).

5.3.4. Image augmentation

In addition to bootstrapping, online image augmentation was also used to diversify our data set to see if additional accuracy could be obtained. For the purpose of this project, we assume that the photos uploaded by users are taken in various environments and conditions. For example,

images could be taken either in daylight or at night, and the garbage may not be well centered and straightened in the image. Based on our assumptions, following operations were performed on the images from training data set:

- Randomly flip (reflect) an image horizontally.
- Randomly rotate an image clockwise or counter clockwise up to the 10 degree.
- Randomly adjust the brightness of an image to be in range 0.5 ~ 1.5.
- Randomly shift 0 ~ 0.1 of the width and height of an image.

In this project, we implemented real-time image augmentation by using keras ImageDataGenerator, below is the example of augmented image.

In left to right, top to down order:

1. Shift height + decrease brightness of egg carton
2. Increase brightness of various dirty water bottles
3. Rotate clockwise for aerosol cans
4. Decrease brightness for water bottle
5. Rotate anticlockwise for tissue paper
6. Rotate clockwise + decrease brightness for plates of food
7. Flip + Shift height for gift wrap
8. Crop + rotate clockwise for tissue wrap
9. Rotate clockwise for aluminium ball

Image augmentation was time consuming and did not improve accuracy significantly for us. This could be because we already had a very diverse dataset to begin with. In fact, we did observe lowered accuracy for overfitting, this could be because augmented images produced a black edge on rotation, which could have interfered with the training.

5.3.5. Increasing Base Learner Diversity

In order to construct a good ensemble model that can perform better than every single contributing model, we need to select a set of base learners as diverse as possible.

Smart Recycling Bin

We trained three CNN models that reused EfficientNet V2B3 and evaluated their performance on the testing set. It can be seen that all of these three models reached 70% accuracy but aggregating their results did not improve the accuracy very much.

Model	Accuracy
EfficientNetV2B3_instance1	0.705078125
EfficientNetV2B3_instance2	0.7046875
EfficientNetV2B3_instance3	0.7057291666666666
Ensemble	0.7065724738236407

Then we evaluated the diversity of these three base learners by finding the correlation of predicted values of these models on a testing set.

	base_model0	base_model1	base_model2
base_model0	1.000000	0.974781	0.976086
base_model1	0.974781	1.000000	0.970267
base_model2	0.976086	0.970267	1.000000

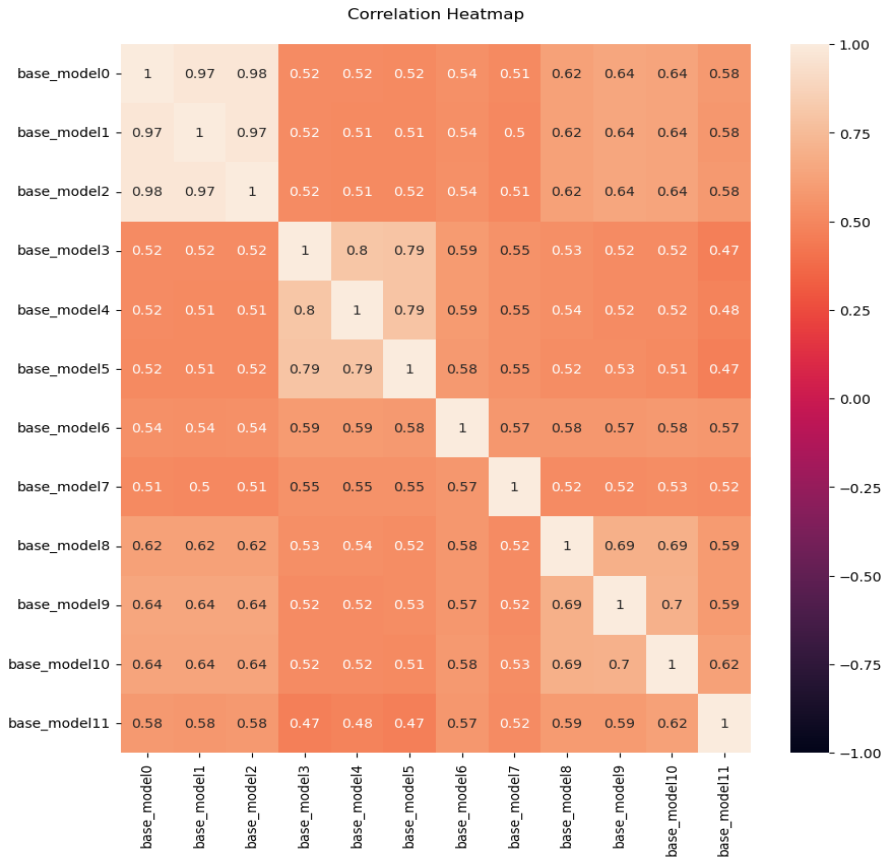
The correlation matrix showed that the predictions made by these three base learners were highly correlated, which means that they are making similar predictions. Consequently, the ensemble results did not improve as expected. For example, all three base learners predicted “dirty recyclables” for an image of aerosol cans whose label is “clean recyclables”. Therefore the aggregated result would also be “dirty recyclables”.

	clean recyclables	dirty recyclables	non recyclables
base_model_0	0.325287	0.600457	0.074256
base_model_1	0.336595	0.589074	0.074330
base_model_2	0.306299	0.622053	0.071648

"img_data_set\test\clean recyclables\aerosol_can_14.jpeg"

Knowing that the complementarity and diversity of individual base learners are desired to construct a good ensemble model, we examined different ways to increase diversity. In addition to resampling, image augmentation and trying different hyperparameters and optimizers, it is suggested that varying model types will also help with ensemble diversity (Brownlee, 2021). We examined the correlation matrix of all our base learner variants and found that between VGG16 and EfficientNet, the correlation is slightly lower, around 60%. We decided to try hybrid ensembling between VGG16 and EfficientNet to see if it could yield better ensemble results compared to just using the best EfficientNet variant (unfrozen V2B3). However, we observed a lower accuracy compared to just EfficientNet V2B3.

In the correlation heatmap below, base learners 0 to 3 are VGG-16, base learners 4 to 11 are the EfficientNetV2 variants (4-7 Frozen, 8-11 Unfrozen).



The heatmap showing the correlation between trained base learners.

We tried to adjust the learning rate, optimizer and batch size but the result of training was not much. We believe it is because most of the parameters have been frozen so the hyperparameters were so hard to improve the accuracy of the models.

We also tried with ensembling different EfficientNet variants (e.g. V2B0, V2B1, V2B2, V2B3). This was found to have higher accuracy than just V2B3. Hence this was our final approach for deployment.

Below is the list of final base learners in the ensemble model. It can be seen that the selected base learners exhibit modest positive correlation in their predictions on the testing set.

Model Name	Pretrained Model Used	Modification
base_model0	EfficientNetV2B3	Froze all pretrained layers
base_model1	EfficientNetV2B2	Unfroze last three pretrained layers
base_model2	EfficientNetV2B3	Unfroze last three pretrained layers

	base_model0	base_model1	base_model2
base_model0	1.000000	0.638400	0.578480
base_model1	0.638400	1.000000	0.624125
base_model2	0.578480	0.624125	1.000000

5.3.6. Result Aggregation using Hard/Soft voting

To evaluate the performance of the ensemble model, we examined if the final class label would be more accurate than any single base learners. In this project, we used two methods to

aggregate the results of base learners and compared their differences. The first method we used is hard voting, in which the predicted class label of each base model is the one with the highest probability, and the class label that has been predicted most frequently is considered as the final class label. Another method is soft voting, we first calculate the average class probabilities and select the class labels with highest probability as the ensemble class labels. Following is the example of how we conduct hard voting and soft voting:

'./img_data_set/test\dirty recyclables\clear_polymer_food_box_247.jpeg'

	clean recyclables	dirty recyclables	non recyclables
model0	0.476186	0.307428	0.216386
model1	0.409659	0.403094	0.187246
model2	0.016492	0.980102	0.003406

In hard voting/majority voting, each predicted label is the class with highest probability. Then the ensemble label would be “clean recyclables” as it is predicted most frequently. The table below illustrates the concept of hard voting:

Model	Result	Ensemble Result
Model0	clean recyclables	Clean Recyclables
Model1	clean recyclables	

Model2	dirty recyclables	

In soft voting/averaging probability, we compute the average probabilities for each class. The class which has the highest average probability is the final ensemble label.

Model	**Probabilities** **[Clean Recyclables, Dirty Recyclables, Non Recyclables]**	**Model Result**
Model0	[0.476186, 0.307428, 0.216386]	clean recyclables
Model1	[0.409659,0.403094,0.187246]	clean recyclables
Model2	[0.016492,0.980102,0.003406]	dirty recyclables

```
P(clean recyclables) = (0.476186+0.409659+0.016492)/3 = 0.3007793
P(dirty recyclables) = (0.307428+0.403094+0.980102)/3 = 0.5635414
P(non recyclables)   = (0.216386+0.187246+0.003406)/3 = 0.1356795
ensemble result -> dirty recyclables
```

Above example demonstrates how different voting methods give different ensemble results. In the above case, soft voting gives a correct prediction while hard voting does not. We examined which voting method is preferred to be implemented in our ensemble model by comparing the accuracy score on the testing data set. From the result it can be seen that with soft voting, the ensemble model will have better performance.

```
base_model0: 0.7057291666666666
base_model1: 0.7184895833333333
base_model2: 0.7766927083333334
```

```
hard_voting_acc: 0.7635416666666667

soft_voting_acc: 0.791015625
```

5.4. Limitations and improvements

①Although our hybrid model's accuracy rate is as high as 80%, there is still a lot of room for improvement, we will try to use more complex model to improve the accuracy of our model, we will also try to use more new technology such as transformer to let our model more reliable to our users.

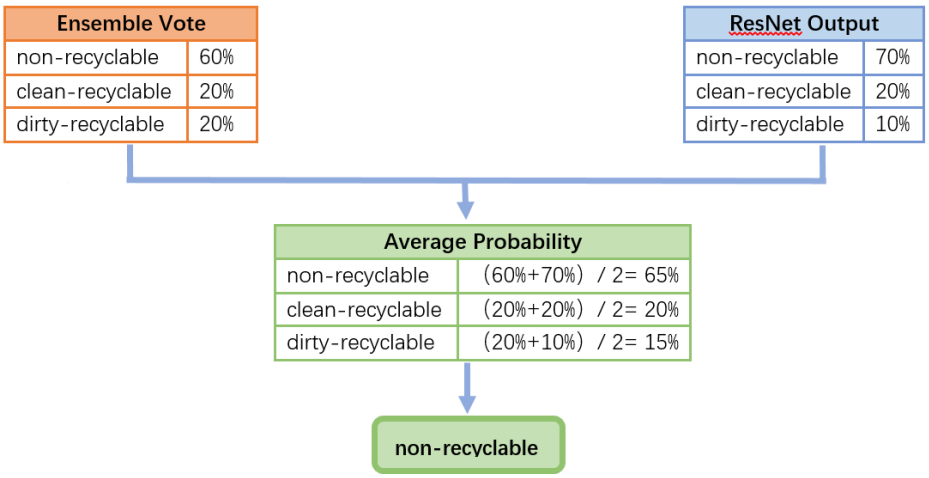
②Although our classification system is so powerful, we still need collect more data to improve our model, in the future after our user get the result after classification, they can apply to change the label if they think the item label is wrong, we will collect the image and label as the dataset to train the model and improve the model.

③There is no doubt our system combined with smart garbage cover will have a better effect to improve the garbage recycle rate, so if it is possible, we will integrate our system with hardware support, when the item is recyclable, the garbage cover will unlock to let our user throw the item in it, otherwise, the garbage cover will keep lock.

④Although we have used multiple thread to improve the processing speed, we still believe we can have more timely response if we have more advanced cloud computing resources.

6. Final Hybrid Model

Same as the previous step of voting, we combine the output of the ensemble vote and the output of Resnet, and calculate the average of the two sums to get the final average likelihood, of which the most likely is the final category.



7.Deployment considerations

The team considered 2 kinds of deployment patterns:

- Deploying the models to the edge (i.e. install a computer at the bloobin to run the model)
- Centralized deployment in the cloud

We opted for centralized deployment because of several reasons:

- Alleviate the power requirement to be installed at the bin to run a computer for the predictions. This would alleviate the retrofitting of existing bins with a smaller computer that would just be in charge just to receive the “open/close” command from the cloud when the algorithm “approves” the recyclable. We do not need a powerful computer to run the predictions.
- Easier update of algorithm. Running the algorithm centrally in the cloud will facilitate the easy deployment of more accurate models as and when they are trained, with less down-time as we do not have to collect back the edge devices to update their models. Vertex AI allows us to do traffic split to gradually roll out models.
- Central monitoring. We are able to monitor the model centrally and detect downtimes and failures. It will not be critical to instrument the edge devices too heavily, as they are only used for simple functions. If the model is failing, we can stub the model output to always “OPEN” while we rectify the issue, so as not to impede the disposal of recyclables.
- Possibility for better model accuracy using ensemble. Running in the cloud will allow us to run ensembles as there are less restrictions on computing resource.

- Elastic scaling of compute resource for the algorithm. If we are experiencing high numbers of requests for predictions, we scale the model more easily because it is cloud based.

The only drawbacks to this are that there is a need for wireless connection at every bin, and the network latencies might be an issue.

The team attempted several approaches:

At first the team attempted to deploy the tensorflow model to Google Colab using flask_with_ngrok. However, due to updates on the ngrok side, the flask_with_ngrok library did not work well after the authtoken had been installed, and kept crashing.

Hence, we next decided to deploy the tensorflow models to VertexAI. The models were converted from .h5 to SavedModel format and uploaded to Google cloud storage. From there, they were imported into VertexAI using a prebuilt Docker container, and deployed to a Model Endpoint for prediction serving over REST API. This approach worked well for VGG models, as they were smaller and managed to run on the prebuilt Docker containers. The EfficientNet models were larger and while failing on the regular prebuilt containers, could run on the Tensorflow “Optimized” containers. However, as we started to unfreeze and train the weights on the last 2 layers of the models, the final exported model increased in size from about 50mb to (70-140mb) and we had trouble deploying new model endpoints to the cloud, even after increasing the compute resource used to run the docker container.

Another approach we could have tried is to run the model locally on Tensorflow Serving, which is a server dedicated to serve Tensorflow model predictions through a REST API.. We think that Vertex AI was already hosting Tensorflow Serving behind the scenes, but with limitations on our end. For example, with Tensorflow Serving, we should be able to query model metadata for

hosted models, but we were not sure where Vertex AI was exposing this endpoint for query. In addition, as the docker containers on Vertex AI were already failing, we were not confident of creating one that could run successfully on Tensorflow Serving.

Another issue with remote hosting was the long latencies for query. A query took about 7 to 10 seconds to complete, whereas predictions run locally could return results under a second if the model was loaded. The recommended approach to solve latencies was to do peer Virtual Private Networks of Vertex AI, which was a bit too much exploration to do, given we were running out of time.

Hence we decided to roll back our efforts at cloud deployment and go for a local deployment. Two approaches were tried: Hosting models on the Jupyter Kernel Gateway, and to load and run the h5 models directly within Flask. Of the two, the Flask approach was simpler and ultimately chosen as the project final deployment approach.

8.Conclusion

We feel that the accuracy achieved (<80%) is still not yet good enough for our use case, but that is the best that we could achieve within the project timeframe. Other things that could have been tried to increase the accuracy include object detection prior to classification. Doing object detection prior to classification would reduce the amount of noise in the classified image. However, we did not try that yet. Nevertheless, an accuracy of 70% is still a start, and we are thinking to opensource our dataset to facilitate further such work.

In addition, we did not get to deploy our model to the cloud. Since we suspect that model size was an issue, perhaps something could be done to prune the model to reduce the size, as likely there might have been many redundant nodes. Pruning requires further experimenting to find the sweet spot of tradeoffs between accuracy and model size.

In conclusion, the team has developed and delivered a prototype classification system that is able to classify whether the garbage is recyclable and whether the recyclable is clean and dirty.

The team learnt how to use supervised learning to solve classification problems, build ResNet and perform transfer learning, learnt how to use image sensors to get the information from the camera and also learnt how to combine multiple models to evaluate the result by using ensemble approach and hybrid machine learning.

9. References

- Brownlee, J. (2021, April 26). *A gentle introduction to ensemble diversity for Machine Learning*. Machine Learning Mastery. Retrieved November 6, 2022, from https://machinelearningmastery.com/ensemble-diversity-for-machine-learning/
- Verma, Y. (2021, August 25). *Comprehensive guide to different pooling layers in deep learning*. Analytics India Magazine. Retrieved November 5, 2022, from https://analyticsindiamag.com/comprehensive-guide-to-different-pooling-layers-in-deep-learning/

Appendix I: Proposal

Date of proposal: Sep 2022

Project Title: Recycling Image Classification using neural networks

Group ID (As Enrolled in LumiNUS Class Groups): 5

Group Members (name , Student ID):

Group Member	**Student ID**
Namrata Thakur	A0261619B
Ouyang Hui	A0261875U
See Jia Fong Grace	A0261797M
Wang Zhipeng	A0261980Y

Sponsor/Client: *(Company Name, Address and Contact Name, Email, if any)*

Not Applicable

Smart Recycling Bin

Background

In land scarce Singapore, waste management has always been a problem. We have currently only 1 landfill, Semakau landfill (350ha, opened 1999), originally designed to last us till 2045, but is now filling up at an alarming rate and projected to fill up by 2035!

There is no known alternative yet beyond 2035; Singapore already has 2 filled up landfills in Lim Chu Kang (30 Ha, 1976-1992) and Lorong Halus (234Ha, 1970-1999), we are running out of space. [1]

Given our growing population, it is no wonder the government has constant emphasis on waste reductions through the 3Rs (Reduce, Reuse and Recycle) and through the Zero Waste Master Plan.

Singapore has been trying to raise recycling rates to promote sustainability. NEA has put in place 1 recycling bin for every HDB block to ensure accessibility to recycling channels [2]. In addition, the government partners with companies to do regular doorstep collection of recyclables.

All these in the efforts to increase our recycling rates, which are currently hovering about 60%. We achieve good recycling rates (~99%) for industrial waste; this is almost maxed out. From here, it is evident that if Singapore wants to further raise recycling rates to meet our goals in the Zero Waste Masterplan, we have to work on improving domestic recycling. It is easier to recycle industrial waste as it is more homogeneous (less sorting needed) compared to domestic waste. More significantly, for domestic waste, another problem is present: Contamination. [3]

Contamination happens in the form of residents dumping non-recyclables or polluting substances into the recycling bins. The problem is cleanliness. Non-recyclables may be able to be separated out at a sorting facility. However, if polluting substances (such as leftover food and drink) are thrown into the bin, this may render the rest of the contents in the bin unrecyclable. The problem of recycling contamination is so severe, it affects about 40% of all recycling bins. That means 2 in 5 collected bins will not be able to be recycled, and this wastes the recyclables in the bins and effort to collect and sort them.[4,5]

Aim

Smart Recycling Bin

Our team feels that more can be done to help residents ascertain what can go into the bin and what should stay out. Perhaps an algorithm to visually inspect recyclables, coupled with a bin design that only accepts clean recyclables (not our scope) can help to protect the clean recyclables that have been submitted, potentially reducing contamination rates, and increasing recycling rates.

Objectives

We aim to train an ensemble of neural networks to classify recyclables as polluted or clean, or whether it is non-recyclable. For example, this image classifier should be able to reject a PET bottle that still contains liquid, or dirty plastic food containers as dirty recyclables. It should also be able to reject other types of non recyclables, such as plastic cutleries or straws.

For some recyclables, it may not be apparent whether or not it is clean (e.g. an opaque drink can, or dark glass bottle). These can be classified as dirty recyclables, and subject to better bin design, for example remote unlocking when the user submits a clean recyclable.

Project Description:

Dataset

We will use existing garbage datasets available on Kaggle/Github, as these datasets were largely meant for material classification, and relatively clean, we will need to augment these datasets by trawling additional images representing polluted garbage and label clean/ dirty.

We may need additional labels for identification of garbage that are non-recyclables. For example, if an empty plastic cup (e.g. bubble tea cup) was presented to the camera, the model should differentiate if there is a straw present. If there is a straw present, the model should reject the item. If there is no straw present, the model will accept the item.[6]

Model

We will build an ensemble of Convolutional Neural Networks (CNN)

Smart Recycling Bin

Output

The model should output if the submitted item belongs to “Clean Recyclable”, “Dirty Recyclable” or “Non Recyclable”.

Implementation Process

1. We will first crawl data and manually label it according to the 3 outputs (“Clean Recyclable”, “Dirty Recyclable” and “Non Recyclable”) above
2. We will train a simple CNN model based on the crawled data. This will be our baseline to gauge if the data is sufficient to give a sensible output
3. Next we will apply data bagging and image augmentation to train multiple instances of the CNN
4. Finally, we will combine these models into an ensemble and apply majority voting to derive the output predictions of test data
5. The performance of the ensemble will be benchmarked against the initial CNN

Proposed System Workflow

Users can scan a QR code at the bin that will direct them to a website. Users can then take a photo of their recyclables and upload to our server. The server will run the image classification and tell the user if his item can be recycled. The server will also communicate with the bin to allow the user to put his item into the bin (out of project scope).

References

[1] https://zerowastecity.com/50-years-of-waste-management-in-singapore-landfills/

[2] https://www.towardszerowaste.gov.sg/recycle-right/

[3] https://www.nea.gov.sg/our-services/waste-management/waste-statistics-and-overall-recycling

[4] https://www.mse.gov.sg/resource-room/category/2021-02-16-written-reply-to-pq-on-hous

ehold-recycling/#:~:text=Currently%2C%20the%20contamination%20rate%20is,that%20should%20instead%20be%20donated.

[5]
https://www.straitstimes.com/singapore/environment/recycling-bins-to-be-given-to-each-household-to-raise-domestic-recycling-rate

[6]
[https://www.nea.gov.sg/docs/default-source/our-services/waste-management/list-of-items-that-are-recyclable-and-not.pdf]

Appendix II: Mapped Project Requirements

The project has fulfilled the requirements below:

Requirement (Highlighted=Fulfilled)	**System Description**
Supervised learning / unsupervised learning scenarios	The project uses Supervised Learning techniques as it uses labelled data
Machine learning / Deep learning techniques	The project trains ResNet and does transfer learning on VGG 16 to perform image classification
Hybrid machine learning / Ensemble approach	The project creates an ensemble of VGG16 to improve classification performance. The project combines the outputs from the ResNet and VGG16 ensemble to determine the final class of the image
Intelligent sensing / sense making techniques	The project accepts camera input from the frontend for users to submit photos of their recyclables

Appendix III: Data Crawling Keyword List

Item	Keywords
PRINTED PAPER (Glossy and non-glossy)	paper
WRITING PAPER	clean writing paper
NEWSPAPER	clean newspaper
FLYER (Glossy and non-glossy)	clean flyer
BROCHURE (Glossy and non-glossy)	clean brochure
MAGAZINE (Glossy and non-glossy)	clean magazine booklet
BOOKS TEXTBOOKS	clean books textbooks
TELEPHONE DIRECTORY	clean telephone directory
ENVELOPE (with and without plastic window)	clean envelope
RED PACKET	clean red packet
NAMECARD	clean namecard
CALENDAR	clean calendar

Smart Recycling Bin

GREETING CARD	clean greeting card
GIFT WRAPPING PAPER	clean gift wrapping paper
SHREDDED PAPER	clean shredded paper
PAPER RECEIPT	clean paper receipt
CARTON BOX, CARDBOARD BOX	clean carton box, cardboard box
PAPER PACKAGING PRINTED PAPER BOX PAPER BOX	clean paper packaging, printed paper box, paper box
EGG TRAYS	clean egg trays
BEVERAGE CARTON - Milk carton - Drink packet - Juice packet	clean beverage carton
PAPER TOWEL TUBE TOILET ROLL TUBE	PAPER TOWEL TUBE, TOILET ROLL TUBE
TISSUE BOX	Tissue box, kleenex box, empty tissue box
PAPER BAG	Paper bag, macdonalds paper bag, crushed paper bag
PAPER DISPOSABLES - Paper cup - Paper plate - Glitter paper - Crayon drawing	Paper cup, dirty paper cup, paper plate, dirty paper plate, glitter paper, wrapping paper, crayon drawing, greasy brown garbage crumpled
TISSUE PAPER	Tissue Paper, Crumpled tissue paper, dirty tissue paper

Smart Recycling Bin

PAPER TOWEL	dirty PAPER TOWEL, crumpled paper towel, dirty paper towel
TOILET PAPER	dirty TOILET PAPER
DISPOSABLE WOODEN CHOPSTICKS WOODEN CHOPSTICKS PIZZA BOXES	dirty chopsticks,empty pizza boxes, disposable chopsticks, dirty chopsticks, dirty pizza box, greasy pizza box
WAX PAPER	wax paper, brown wax paper sheets crumpled ball, empty cupcake liners dirty, empty cupcake wrappers dirty, half eaten muffin liner, wax paper cardboard food box, wax paper food liner
PAPER PACKAGING CONTAMINATED WITH FOOD	paper packaging with food, contaminated paper food packaging, dirty paper food packaging
PLASTIC BOTTLE PLASTIC CONTAINER - Mineral water bottle - Soft drink bottle - Carbonated drink bottle - Milk bottle - Water bottle - Medicine bottle	empty Mineral water bottle,empty Medicine bottle, empty milk bottle, half full water bottle, half full soft drink bottle, half empty water bottle, half empty soft drink bottle, dirty drink bottle, clean PET bottle, clean crumpled PET bottle, dirty crumpled PET bottle
SHAMPOO BOTTLE BODYWASH BOTTLE FACIAL CLEANSER BOTTLE DETERGENT BOTTLE SOAP BOTTLE	SHAMPOO BOTTLE, BODYWASH BOTTLE, FACIAL CLEANSER BOTTLE, DETERGENT BOTTLE, SOAP BOTTLE, hand soap bottle

Smart Recycling Bin

CD/DVD CD/DVD CASING	CD/DVD, CD Casing, CD ROM
PLASTIC BAG TOILET PAPER PACKAGING TISSUE BOX PACKAGING	plastic bag png, plastic bag, dirty plastic bag
PLASTIC FILM - Magazine wrapper - Plastic packaging for packet drink BUBBLE WRAP FRUIT BOX ZIPLOCK BAG	Magazine wrapper, cling wrap crumpled, plastic wrap crumpled, plastic wrap garbage, bubble wrap thrown away, crumpled bubble wrap, clear polymer food box, clean ziploc bag, dirty plastic box, old ziploc bag
PLASTIC PACKAGING BREAD BAG EGG TRAYS	PLASTIC PACKAGING, gardenia plastic bag, plastic egg tray, plastic egg carton, broken eggs in carton, clear plastic cup tea, leftover bubble tea, half drunk tea disposable cup, clean disposable cup, clean disposable plastic cup, clear plastic cup, dirty plastic cup, dirty disposable cup
PLASTIC CLOTHES HANGER	PLASTIC CLOTHES HANGER
PLASTIC TAKEAWAY FOOD CONTAINER PLASTIC CUPS BUBBLE TEA CUPS	FOOD CONTAINER, clean plastic container, clean food container, dirty plastic takeaway container, dirty food container
POLYSTYRENE FOAM PRODUCT STYROFOAM STYROFOAM CUP STYROFOAM CLAMSHELL	POLYSTYRENE FOAM PRODUCT, styrofoam, styrofoam cup, styrofoam box, styrofoam clamshell containers, dirty styrofoam clamshell containers, dirty styrofoam container, dirty styrofoam cup, dirty styrofoam trash

Smart Recycling Bin

CONTAINER	
PLASTIC DISPOSABLES PLASTIC CUTLERY PLASTIC CROCKERY	plastic cutlery trash, plastic fork trash, , plastic spoon trash, plastic bowl trash, plastic utensils trash, dirty plastic utensils, dirty plastic spoons
PLASTIC PACKAGING WITH FOIL POTATO CHIP BAGS EMPTY BLISTER PACK BLISTER PACK EXPIRED CREDIT CARDS	credit cards, blister pack, empt blister pack, foil packaging, potato chips
OXO-DEGRADABLE BAG BIO-DEGRADABLE BAG	bio degradable bag, oxo degradable bag
DRINKING STRAW	drinking straw
CASSETTE VIDEO TAPE	cassette, video tape
PLASTIC PACKAGING CONTAMINATED WITH FOOD/OIL STAINS	dirty plastic container
MELAMINE PRODUCTS - Melamine cups - Melamine plates	
BEVERAGE GLASS BOTTLE - Beer bottle - Wine bottle - Liquor bottle	clean wine glass, empty beer bottle, empty wine bottle

Smart Recycling Bin

FOOD GLASS BOTTLE - Sauce bottle - Condiment bottle - Jam spread bottle - Food jars	empty condiment bottle, empty food jars, empty sauce bottle, dirty jam bottle, dirty sauce bottle
COSMETIC GLASS BOTTLE PERFUME GLASS BOTTLE	empty perfume glass bottle,
MEDICINE GLASS BOTTLE SUPPLEMENT GLASS BOTTLE	
GLASSWARE - Glass cup - Glass plate - Drinking glass - Wine glass	
BOROSILICATE GLASSWARE PYREX GLASSWARE BAKEWARE TEMPERED GLASS OVEN-SAFE FOOD CONTAINERS CRYSTAL GLASS GLASS WITH METAL WIRES	
WINDOWS	
MIRRORS	
CERAMIC PRODUCT - Ceramic plate - Tea pot	

- Porcelain	
LIGHT BULB - Lamp - Tube - Incandescent lamp - Incandescent bulb - Fluorescent lamp - Fluorescent bulb - LED lamp - LED bulb	
BEVERAGE METAL CAN - Carbonated drink can - Soft drink can - Beer can	
FOOD METAL CAN - Biscuit tin - Food tin - Metal container	
MEDALS	
METAL BOTTLE CAP	
AEROSOL CAN	aerosol can, spray can, spray paint can
CLEAN ALUMINIUM TRAY CLEAN ALUMINIUM FOIL	clean aluminum tray, clean aluminum foil, clean tin foil, clean tin wrapper, aluminum foil ball
NON-FOOD METAL CONTAINER - Paint container - Paint cans	clean paint container, empty paint can, empty paint container

Smart Recycling Bin

RUSTY METAL CANS DIRTY ALUMINIUM FOIL DIRTY ALUMINIUM TRAY	rusty metal cans, dirty metal cans, dirty aluminum tray, dirty aluminum foil, dirty tin foil, dirty tin wrapper, half full paint can, half full paint container
METAL CUTLERY STEEL WOOL METAL ACCESSORIES	metal cutlery, steel wool, metal accessories, knife, metal spoon, metal fork, metal chopsticks, metal cup, thermal flask, metal straw, metal bottle, thermal cup, metal pots, metal pans, metal saucepans,
ICT EQUIPMENT - Desktop computer - Laptop - Tablet computer - Mobile phone - Computer battery - Mobile phone battery - Printer - Modem - Router - Desktop monitor - Docking station - Hard disk drive - Battery charger - Portable charger - Power bank, - Electronic cables - Computer mouse - Keyboard	desktop computer, laptop, tablet computer, mobile phone, smartphone, computer battery, laptop battery, mobile phone battery, printer, modem, router, desktop monitor, docking station, hard disk drive, battery charger, portable charger, power bank, electronic cables, computer mouse, keyboard, telephone, fax machine

Smart Recycling Bin

ELECTRONIC WASTE (e.g. Used mobile phone, tablet, laptop)	ipad, dell laptop, ibm, iphone, samsung galaxy, television, smart tv, speakers, soundbar, macbook, asus, alienware, floppy disk, kindle, lenovo, hp computer, hp laptop, cd, compact disc, cd rom, cd player, walkman, mp3 player, blackberry, personal tablet, handphone, flip phone, handset, nokia
RECHARGEABLE BATTERY	rechargeable battery
HOUSEHOLD BATTERY ALKALINE BATTERY	AA battery, AAA battery, alkaline battery, energizer battery, button battery, GP battery, watch battery, fuel battery, handphone battery, camera battery
REFRIGERATOR WASHING MACHINE DRYER TELEVISION AIR CONDITIONER ITEM CAN BE PLACED IN THE BLUE RECYCLING BINS CANNOT BE PLACED IN THE BLUE RECYCLING BINS	refrigerator, fridge, freezer, chest freezer, bar fridge, twin door fridge, large capacity fridge, washing machine, top load washer, front load washing machine, LG washing machine, dryer, front load dryer, CRT television, tv, samsung tv, panasonic tv, smart tv, lcd tv, aircon, air conditioner, mitsubishi aircon, daikin aircon, aircon compressor, midea aircon, midea washing machine, bosch dryer
ELECTRIC MOBILITY DEVICES - Personal mobility devices - Electric scooter - Electric bicycle - Power-assisted bicycle - Electric mobility scooter	personal mobility device, electric scooter, electric bicycle, power assisted bicycle, electric mobility scooter, electronic scooter, electronic bicycle, electric tricycle, electronic tricycle

Smart Recycling Bin

RICE COOKER MICROWAVE OVEN TOASTER OVEN ELECTRIC KETTLE FOOD PROCESSOR FOOD BLENDER ELECTRIC FAN STANDING FAN CD/DVD PLAYER MUSIC PLAYER SPEAKER AUDIO SOUND SYSTEM RADIO VACUUM CLEANER GAMING CONSOLE	rice cooker, electric multicooker, thermomix, microwave oven, oven, toaster over, airfryer, electric kettle, food processor, food blender, juice blender, blender, fan, electric fan, standing fan, ceiling fan, cd player, dvd player, music player, nintendo, playstation, ps2, ps3, ps4, game controller, xbox, audio system, radio, walkman, vacuum cleaner, dyson, gaming console, kinect
LAMP STAND LAMP FIXTURE	lamp stand, lamp fixture
SPORTS SHOES SCHOOL SHOES FOOTBALL SHOES (without metal studs)	sports shoes, school shoes, football shoes, sneakers, canvas shoes, adidas sneakers, nike sneakers, onitsuka tiger, canvas shoes, kappa shoes, new balance shoes, sports shoes, womens pumps, womens heels, platform shoes, slippers, sandals, crocs, flip flops, footwear,

Smart Recycling Bin

CLOTHES SHOES (other than those listed in #74) BAG TOY UMBRELLA SPECTACLES TEXTILE CURTAINS BEDSHEET BLANKET	clothes, jackets, shirts, tops, bottoms, skirts, pants, dresses, jumpsuits, crop tops, blouse, spaghetti straps, singlets, tshirts, dress shirts, sweaters, scarves, shawl, bag, backpack, sling bag, handbag, purse, wallet, luggage bag, messenger bag, shoulder bag, school bag, children's bag, bag with straps, tote bag, toy, children's toys, infant toys, baby toys, puzzle games, board games, monopoly, card games, spectacles, glasses, sunglasses, transition lenses, curtains, bedsheets, blankets, quilts, comforter, mattress protector, mattress cover, bedding, pillows, pillow cases, bolster, bolster cases, pillow sheets, pillow covers, cushion covers, bolster covers, sofa throw, umbrella
PLANT WASTE HORTICULTURAL WASTE	plant waste, horticultural waste, soil, compost, dead leaves, plant matter,
LUGGAGE BAG	luggage bag
BULKY WASTE FURNITURE	bulky waste, furniture, dining table, chairs, sofa, wardrobe, sideboard, cupboards, dresser table, bedside table, chest of drawers, coffee table
FOOD WASTE	food waste, restaurant waste, kitchen waste, vegetable peelings, coffee grounds carrot peelings, corn husks, corn cob, rotten vegetables, rotten food, rotten fruits, organic waste
LEFTOVER MEDICINE	medicine bottles, medicine tablets, panadol, capsule tablet, supplement tablets, supplement drinks, cough syrup
DIAPER SANITARY PAD	diaper, nursing pad, sanitary pad, kotex, laurier pad

Smart Recycling Bin

STATIONERY PEN PENCIL	stationery, pen, pencil, eraser, paintbrush, pencil sharpener, color pencils, highlighter pen

Appendix IV: Installation Guide

To install the current release:

$ git clone https://github.com/princepride/scanner

$ cd scanner

· To install the models:

go to the google driver to install the models and move it to the file path scanner/backend/

· To install node.js dependency packages

$ cd frontend

$ npm install

· To install python dependency packages

$ cd backend

$ pip install -r requirement.txt

Appendix V: User Guide

Follow the steps below to start the server and react.js

- $ cd backend
- $ python app.py
- $ cd frontend
- $ npm start

then, you can run it on your localhost.

if you want to run it on your mobile phone, you need to open your hot point and change the address of server address

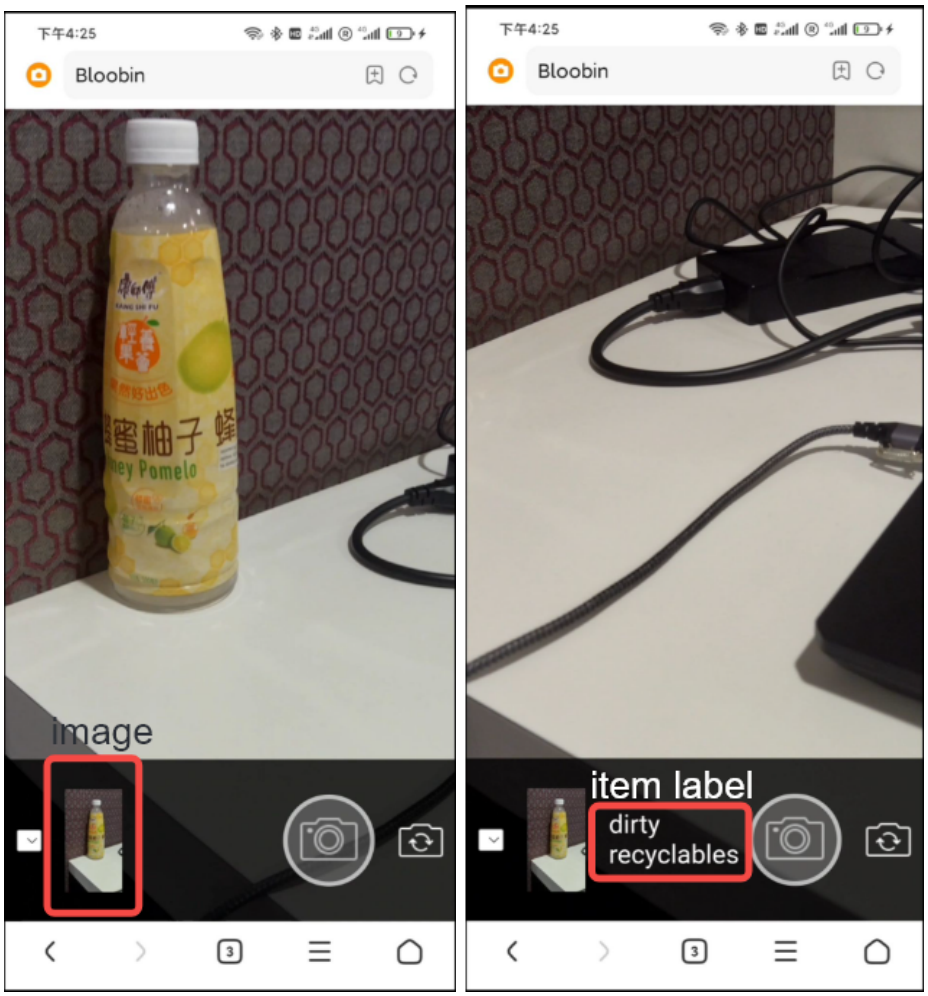
- open the file frontend/src/fetchMethod.js

  change `export const address = "http://localhost:5000/"` to `export const address = "http://your_hotpoint_address:5000/"`

button1: To switch the camera of your device. You can use either of your front or back camera.

button2: To take the picture, then the picture will be automatically sent to the back end to make the classification.

button3: To flip the camera if left and right are reversed.



After taking the picture, the image will be shown on the bottom side.

After our server run the models to make the classification, the label of the item will be shown on the bottom side.