IMAGE BASED WASTE CLASSIFICATION SYSTEM

A PROJECT REPORT

Submitted by

GROUP - 03

NISHIT NAKRANI

KRUPAL VORA

NIDHI JOHNSON

ANISHKA TAHILIANI

Studying in

IST 718 Big Data Analytics M002

Spring 2020

School of Information Studies

Syracuse University

Table of Contents

1. INTRODUCTION

The place of recycling in modern society is very important. In recycling process for today, separation of waste with manpower must take place in order to make a series of large filters. People may be confused about how products they consume are considered to be garbage. Within the scope of this study, it was to develop an algorithm for classification of garbage. Our aim is to increase efficiency of waste processing facilities and to identify non-recyclable wastes because garbage separation process is very difficult to separate garbage with 100% accuracy. The proposed method will be designed not only for environmental benefits but also for saving time and manpower.

The process of this project is only to classify whether or not garbage is suitable for recycling. Another project related to recycling is to categorize garbages as a smartphone application by utilizing imaging method. With the application carried out, it is to ensure that citizens are followed up for recycling garbage in the neighborhood. The accuracy rate obtained after training phase is approximately 88%. Using pre-trained model, a faster and higher accuracy network training was conducted.

2. PROBLEM STATEMENT

Waste Management is a big problem in the United States of America. Most waste is not handled efficiently and ends up in landfills.

This leads to many issues like:

- Increase in landfills
- Eutrophication
- Consumption of toxic waste by animals
- Leachate
- Increase in toxins
- Land, water and air pollution

3. AIM OF PROJECT

Our aim is to increase efficiency of waste processing facilities and to identify non-recyclable wastes because garbage separation process is very difficult to separate garbage with 100% accuracy. The proposed method will be designed not only for environmental benefits but also for saving.

4. OUR PROPOSED MODEL

Our study and research related to this is for United States Environmental Protection Agency and other countries similar agency to start building smart waste collection. The system comprises of a sensor that is situated on top of the trash opening which will use an IoT system to classify the waste into organic and recyclable. This bin will be connected to the wifi.

Figure: Left side Image show old trash bin and Right side show the new trash bin with sensor

4.1. ADVANTAGE OF OUR PROPOSED PROJECT:

1. Automate process
2. Time efficient
3. Less system loss
4. Predictable output
5. Automated work assignment
6. Creating more job opportunity

4.2. APPROACH:

• Studied white papers on waste management

• Analysed the components of household waste

• Segregated the waste matter into two classes (Organic and recyclable)

• Automated the process by using IOT and machine learning

• Reduce toxic waste ending in landfills

5. WORKFLOW

1. Examine and understand data

2. Build an input pipeline

3. Build the model

4. Train the model

5. Test the model

6. Improve the model and repeat the process

7. Evaluate the model

5.1. EXAMINE AND UNDERSTAND DATA

The dataset we used for this project is been taken from Kaggle. The dataset comprises of : **Training data - 22564 images (85%) & Test data - 2513 images (15%).**

Data Preparation Steps:

1. Read images from the disk.

2. Decode contents of these images and convert it into proper grid format as per their RGB content.

```
total training R images: 10000
total training O images: 12566
total test R images: 1112
total test O images: 1401
--
Total training images: 22566
Total testing images: 2513
```

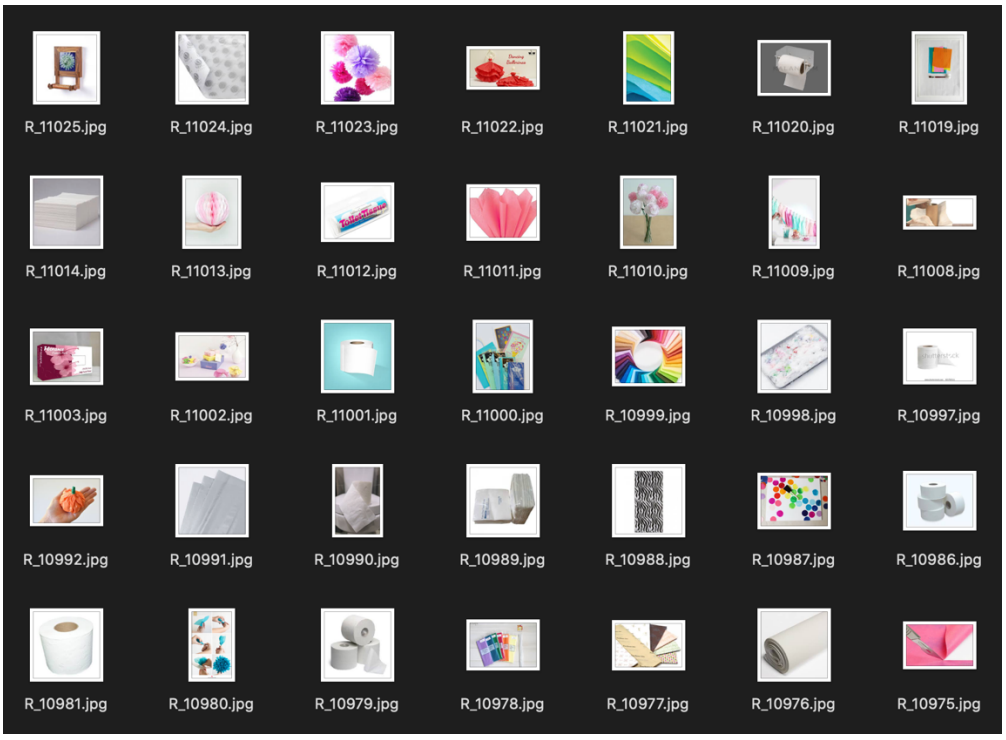


Figure: Recyclable Dataset Images

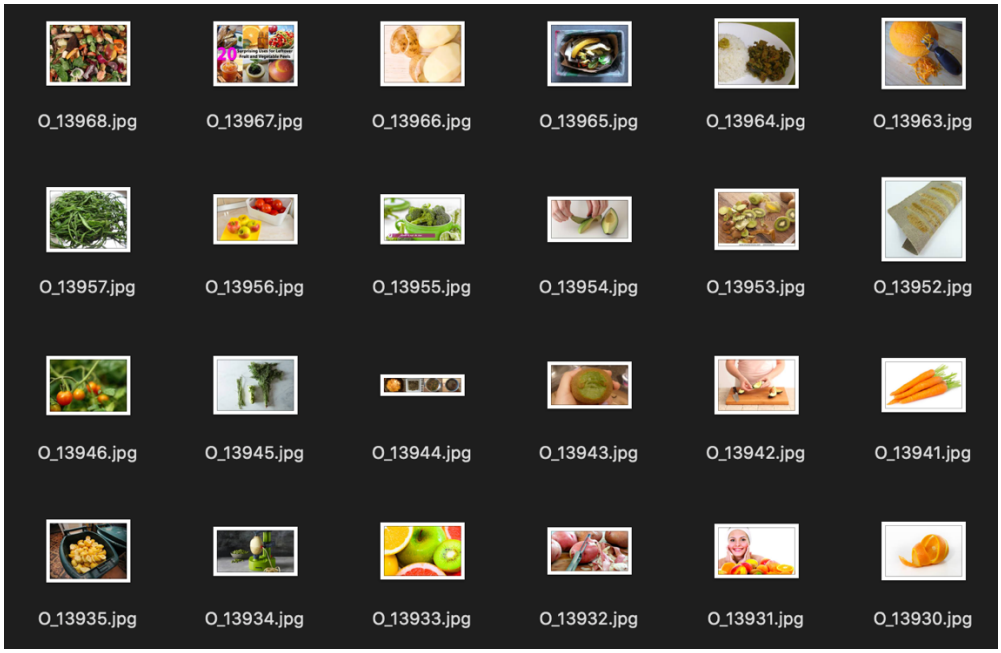


Figure: Organic Dataset Images

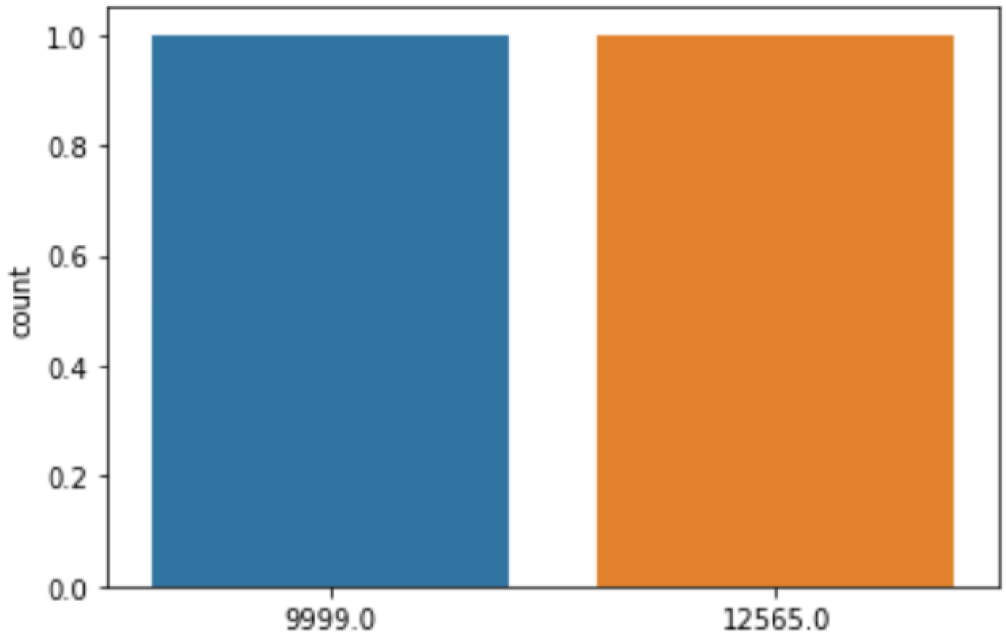
5.2. BUILDING INPUT PIPELINE:

1. We created y_test and y_train array to label images as 0(Recyclable) or 1(Organic).

2. After defining the generators for training and validation images, the flow_from_directory method load images from the disk, applies rescaling, and resizes the images into the required dimensions.

3. Visualize the training images by extracting a batch of images from the training generator — then plot five of them with matplotlib.

4. We have total 12565 Organic images and 9999 Recyclable images in train directory.



5.3. BUILDING MODEL:

Deep learning, called hierarchical learning, aims to analyse structure of data from simple to complex by using multilayer structures. In particular, effective features obtained by Convolutional Neural Network (CNN) make classification process much easier. In order to better understand convolutional neural networks, first layer on convolutional network identifies the simplest structures contained in the image, for example, through familiar two-dimensional images. Each of many filters in layer is obliged to find one of edges in a given image. This means that when an edge information contained in the image is taken into account, different filters serve for different angles of an edge, or for different shapes with other edges. The output of the first layer is feature maps that contain the information of structures that these filters detect. These outputs include information about various edge structures associated with the image. The next evolution layer reveals relationship edges that have been detected on the previous layer on features maps. Each convolution layer analyses correlations of combinatorial structures detected in previous layer with image. This flow, in which complexity increases with the number of layers, facilitates the acquisition of semantic information from structural information.

Convolutional neural networks (CNN) is a specialized state of multilayer neural network and is designed to detect geometric shape in image processing. In conventional multilayer neural

network, a neuron in first layer is connected with all neurons in next layer; convolutional layer establishes local connections on the output of previous layer. The fully connected layer performs a matrix multiplication.

The models for this project were constructed using the Neural Networks. These models were used to train the on the given image dataset with multiple waste images. The Convolutional Neural Network (CNN) models with multilayer neural networks were designed to detect the geometric shape in image processing.

MODEL 1: Selflaid CNN model with 3 layers

- This model contained 3 convolutional blocks to process the dataset.
- This model has one fully connected layer with 512 units, i.e. it contains 512 hidden layers which was activated by the Rectified Linear Unit (ReLU) activation Function.
- This model was compiled with ADAM optimizer with Binary Cross Entropy loss function.
- Total params: 10,641,441 Trainable params: 10,641,441 Non-trainable params: 0

MODEL 2: Selflaid CNN model with 10 layers

- This model contained 10 convolutional blocks to process the dataset.
- This model has one fully connected layer with 130 units, i.e. it contains 130 hidden layers which was activated by the Sigmoid activation Function.
- This model was compiled with RMSprop optimizer with Binary Cross Entropy loss function.
- Total params: 542,738 Trainable params: 542,738 Non-trainable params: 0

MODEL 3: Inception Resnet V2 model

- This is a heavily engineered model which process the data with multiple filters in the same level.
- This model contained 203 convolutional blocks to process the dataset.
- This model has one fully connected layer with 258 units, i.e. it contains 258 hidden layers which was activated by the Sigmoid activation Function.
- This model was compiled with RMSprop optimizer with Categorical Cross Entropy loss function.

- Total params: 54,534,242 Trainable params: 54,473,442 Non-trainable params: 60,800

5.4. TRAINING AND TESTING MODELS:

Model 1 is train in 15 epochs in batch of 128 and took almost 3 hours to train and validate test images.

Model 2 is train in 10 epochs in batch of 256 and took almost 6 hours to train and validate test images.

Model 3 is train in 10 epochs in batch of 64 and took almost more than 24 hours (1 day) to train and validate test images.

5.5. IMPROVING THE MODEL BY DATA AUGMENTATION:

In the plots above of Model 1, the training accuracy is increasing linearly over time, whereas validation accuracy stalls around 80% in the training process. Also, the difference in accuracy between training and validation accuracy is noticeable—a sign of **overfitting.**

There are multiple ways to fight overfitting in the training process. We will use data augmentation and add dropout to our model.

One way to fix this problem is to augment the dataset so that it has a sufficient number of training examples

This helps expose the model to more aspects of the data and generalize better.

1. Apply horizontal flip

2. Take one sample image from the training examples and repeat it five times so that the augmentation is applied to the same image five times.

3. Randomly rotate the image

4. Apply zoom augmentation

5. Create validation data generator

6. Creating a new network with Dropouts : When applying 0.2 dropout to a certain layer, it randomly kills 20% of the output units in each training epoch.

5.6. EVALUATE THE MODEL:

MODEL 1: Selflaid CNN model with 3 layers

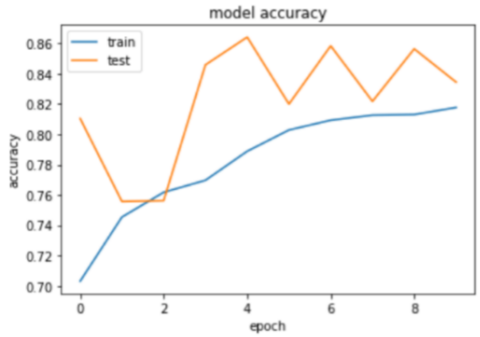
Model 1:

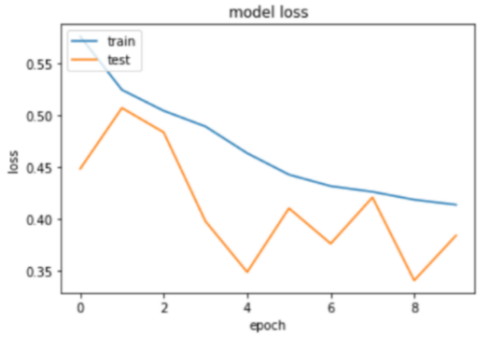

```
Epoch 10/15
176/176 [==============================] - 354s 2s/step - loss: 0.3366 - accuracy: 0.8503 - val_loss: 0.3693 - val
_accuracy: 0.8487
Epoch 11/15
176/176 [==============================] - 339s 2s/step - loss: 0.3307 - accuracy: 0.8551 - val_loss: 0.3029 - val
_accuracy: 0.8684
Epoch 12/15
176/176 [==============================] - 357s 2s/step - loss: 0.3315 - accuracy: 0.8538 - val_loss: 0.3130 - val
_accuracy: 0.8816
Epoch 13/15
176/176 [==============================] - 346s 2s/step - loss: 0.3304 - accuracy: 0.8558 - val_loss: 0.3099 - val
_accuracy: 0.8795
Epoch 14/15
176/176 [==============================] - 333s 2s/step - loss: 0.3223 - accuracy: 0.8600 - val_loss: 0.3114 - val
_accuracy: 0.8840
Epoch 15/15
176/176 [==============================] - 305s 2s/step - loss: 0.3181 - accuracy: 0.8603 - val_loss: 0.3140 - val
_accuracy: 0.8643
```

Model is underfitting as validation loss is less than training loss. Fluctuation in validation set, this basically means, that some portion of examples are classified randomly, which produces fluctuations, as the number of correct random guesses always fluctuate (imagine accuracy when coin should always return "heads"). Basically sensitivity to noise (when classification produces random result) is a common definition of overfitting. We tried to solve overfitting earlier by data augmentation but still there is little overfitting. This can be solved if we train the model for more epoches, try to play with learning rate and also by getting more data of waste.

MODEL 2: Selflaid CNN model with 10 layers

Model 2:





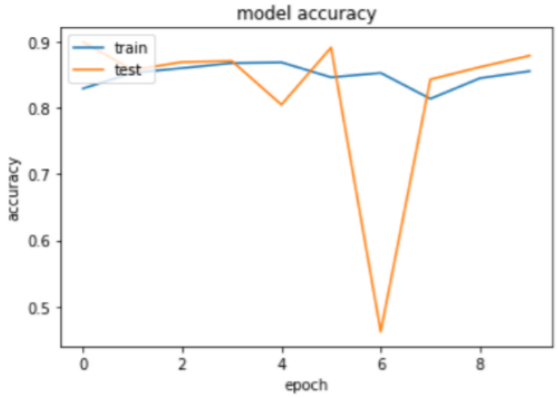

```
8 - val_accuracy: 0.8458
Epoch 5/10
22564/22564 [==============================] - 1707s 76ms/step - loss: 0.4636 - accuracy: 0.7888 - val_loss: 0.348
7 - val_accuracy: 0.8641
Epoch 6/10
22564/22564 [==============================] - 1644s 73ms/step - loss: 0.4429 - accuracy: 0.8028 - val_loss: 0.410
4 - val_accuracy: 0.8199
Epoch 7/10
22564/22564 [==============================] - 1981s 88ms/step - loss: 0.4318 - accuracy: 0.8093 - val_loss: 0.376
2 - val_accuracy: 0.8583
Epoch 8/10
22564/22564 [==============================] - 1576s 70ms/step - loss: 0.4263 - accuracy: 0.8126 - val_loss: 0.420
9 - val_accuracy: 0.8217
Epoch 9/10
22564/22564 [==============================] - 1580s 70ms/step - loss: 0.4186 - accuracy: 0.8130 - val_loss: 0.340
6 - val_accuracy: 0.8563
Epoch 10/10
22564/22564 [==============================] - 1844s 82ms/step - loss: 0.4138 - accuracy: 0.8177 - val_loss: 0.384
0 - val_accuracy: 0.8345
```

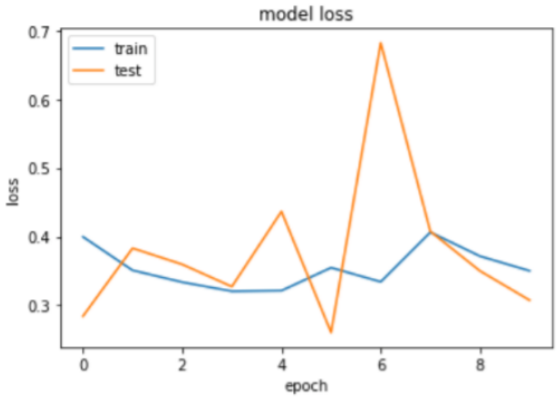
Model is underfitting as validation loss is less than training loss. In addition, the following ways can also be used to tackle underfitting.

- Increase the size or number of parameters in the ML model.
- Increase the complexity or type of the model.
- Increasing the training time until cost function in ML is minimised.

MODEL 3: Inception Resnet V2 model

Model 3:



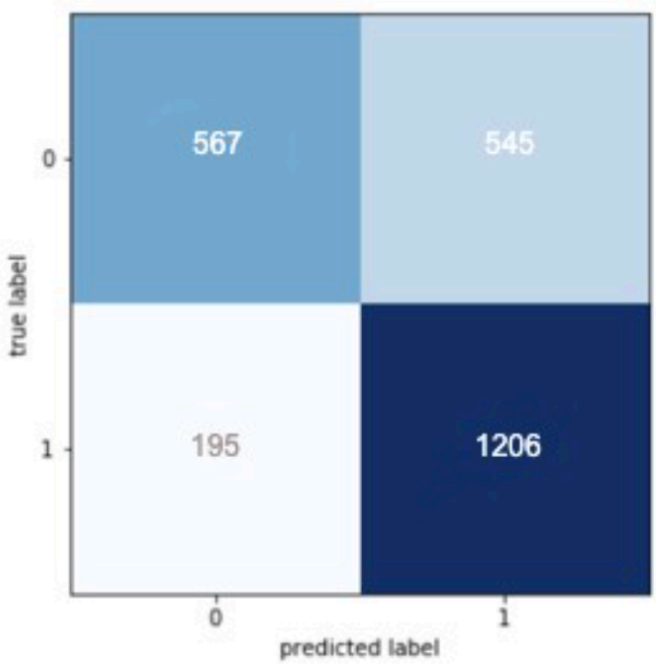



```
Epoch 6/10
22564/22564 [==============================] - 7529s 334ms/step - loss: 0.3548 - accuracy: 0.8460 - val_loss: 0.26
05 - val_accuracy: 0.8910
Epoch 7/10
22564/22564 [==============================] - 7835s 347ms/step - loss: 0.3342 - accuracy: 0.8526 - val_loss: 0.68
21 - val_accuracy: 0.4620
Epoch 8/10
22564/22564 [==============================] - 9580s 425ms/step - loss: 0.4064 - accuracy: 0.8136 - val_loss: 0.40
75 - val_accuracy: 0.8428
Epoch 9/10
22564/22564 [==============================] - 12369s 548ms/step - loss: 0.3715 - accuracy: 0.8447 - val_loss: 0.3
500 - val_accuracy: 0.8615
Epoch 10/10
22564/22564 [==============================] - 20089s 890ms/step - loss: 0.3502 - accuracy: 0.8553 - val_loss: 0.3
074 - val_accuracy: 0.8786
```

- Here model is sometime overfitting and some time underfitting.
- The problem is here validation data is biased.
- When validation data is of only one class mostly test accuracy fluctuates.
- We can try to fix this with decreasing learning rate as learning rate must be big and also the size of validation set may be too small, such that small changes in the output causes large fluctuations in the validation error.

5.7. CONFUSION MATRIX:

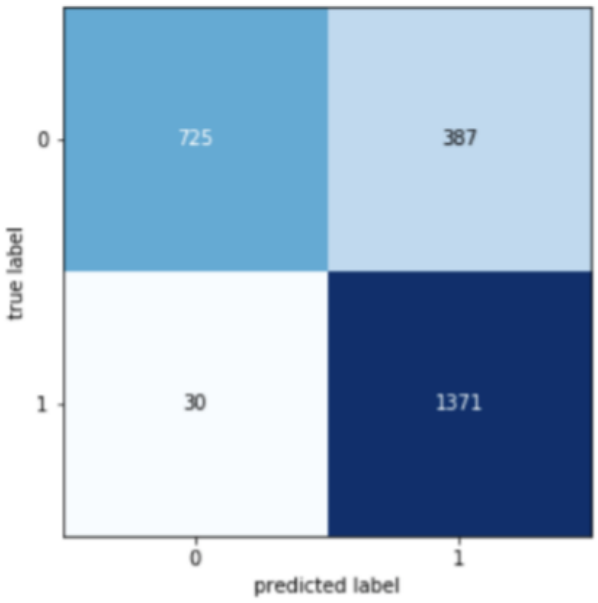
MODEL 1: Selflaid CNN model with 3 layers



	precision	recall	f1-score	support
0	0.74	0.50	0.59	1112
1	0.68	0.86	0.74	1401
accuracy			0.70	2513
macro avg	0.75	0.69	0.71	2513
weighted avg	0.74	0.70	0.72	2513

MODEL 2: Selflaid CNN model with 10 layers

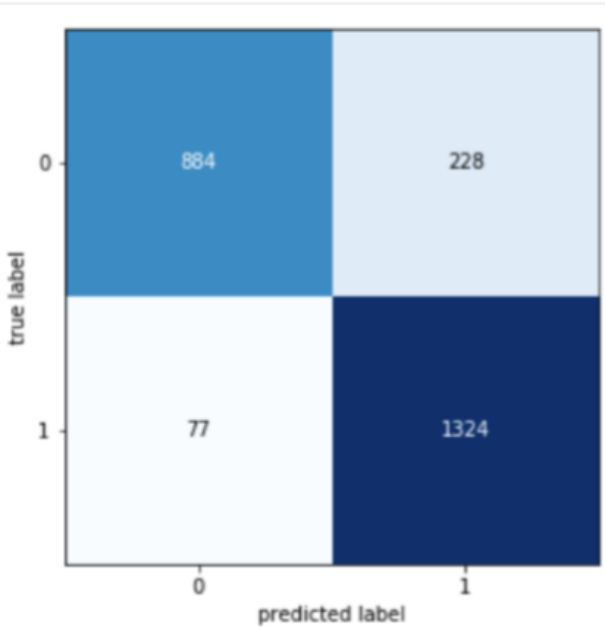
Model 2:



	precision	recall	f1-score	support
0	0.96	0.65	0.78	1112
1	0.78	0.98	0.87	1401
accuracy			0.83	2513
macro avg	0.87	0.82	0.82	2513
weighted avg	0.86	0.83	0.83	2513

MODEL 3: Inception Resnet V2 model

Model 3:



	precision	recall	f1-score	support
0	0.92	0.79	0.85	1112
1	0.85	0.95	0.90	1401
accuracy			0.88	2513
macro avg	0.89	0.87	0.87	2513
weighted avg	0.88	0.88	0.88	2513

Although high Accuracy scores have a comforting effect on our intuition — leading us to false claims that our machine learning model is performing very well, we often forget to focus on the little things like Precision and Recall that sometimes matter the most.

A low False Positive value results in a much higher Precision score and low FN value results in a much higher Recall score.

The 3 Pillars of Binary Classification: Accuracy, Recall, and Precision:

	Model 1		Model 2		Model 3	
	0	1	0	1	0	1
Recall	50%	86%	65%	98%	79%	95%
Precision	74%	68%	96%	78%	92%	85%
Accuracy	70%		83%		88%	

6. CONCLUSION AND ROADMAP

Finally we decided to go with **Model 3** which is Inception Resnet V2 for building the IoT and deep learning based smart trash bin for government agency to classify the waste into Organic and Recyclable as this model was having highest recall, precision and accuracy.

Also this model will get better if we try to train it on more larger datasets with more epoches to extract features and increase accuracy to predict True- Positive and False- Negative.

We will try to contact the government agency and other waste management agency to provide us support and fund to take it more further and even to create larger dataset to make it more efficient.

After getting success on this work will decrease the landfills and water-air pollution.

Also further roadmap is that we going to play around the hyperparameters to fine tune the model and going to launch a demo website via which by uploading a image, it will show the prediction of that image. This can be used to scan the waste and classify them as organic or recyclable.

Here is the demo of the website we created using “Flask” to deploy our train model for web service application to predict.

Figure: Snapshot of Website we created using Flask to deploy model