Recycl-EZ

Jeremy Ng, Tay Kiat Hong, Daniel Teo, Yuri Kim

50.038 Computatioanl Data Science

Information Systems Technology and Design, Singapore University of Technology and Design

11 December 2020

***Abstract*—"There is no such thing as 'away'. When we throw anything away it must go somewhere." This is an apt quote by Annie Leonard, which is the premise of our project. The project explores the methods in which waste sorting is being conducted for further recycling efforts. Currently, many approaches require human intervention and additional efforts for the sorting, however, it might not be the most effective. This is where automating the waste sorting process would come in handy, making the task more efficient and seamless. Using Deep Learning Network Architectures for waste classification, we experimented with quite a few models, with a dataset pooled from multiple sources, web scraped images and our own image inputs. The Convolutional Neural Network (CNN) was used for the task of image classification. The intended outcome would be to have a camera placed in trash bins to detect and segregate the trash out accordingly without the need for added human labour. Based on the image input from the camera, the corresponding chutes will be activated. The introduction of this automation would hopefully serve to save time and resources, lessening the burden on governments, industrialists and the society as a whole. Given further fine tuning of our best model, the InceptionV3, with certain parameters, we managed to achieve the best performance of 94.4% accuracy.**

I. Introduction

As the world wide population grows exponentially, there has been a corresponding rise in the solid waste production that plagues mankind. Thus, to better manage such a problem, proper waste management becomes more imperative than ever before, otherwise there would be great adverse effects on public health and the environment at large. With the trash generated from developed cities piling towards the sky, it is important that we recognise this growing problem and seek to ameliorate this tricky situation. If done well, many benefits could be reaped like conserving natural resources, saving energy, preventing more environmental pollution and so much more. Currently in developed countries, mechanical and chemical sorting methods are being deployed as sorting measures whereas in developing countries, it is not uncommon to find it done via manual labour by collectors with the intention of trading it for profit.

Utilizing deep learning, which is a type of machine learning algorithm that taps into multiple layers of feature extractions and data representation, specifically the Convolutional Neural Network, to analyze the image, we are able to achieve object recognition, predict its classification and sort them into the respective categories.

II. Dataset Collection

In order to build our model which can correctly classify recyclable waste, we have to first feed our model in with our dataset comprising recyclable waste images. Our final dataset which is used to train, validate and test our model consists of 4608 images spanning over 5 specific recyclable waste categories: cardboard, glass, metal, paper, plastic. In our data collection process, we merged several datasets from various open source collections like github and kaggle. We also scrapped additional images from google and naver search engines to further add on the numbers to the dataset.

Figure 1: images of labelled dataset

These are the following sources which we referenced from and combine to form our own dataset:

- https://www.kaggle.com/asdasdasasdas/garbage-classification
- https://github.com/openrecycle/dataset
- https://www.google.com/imghp?hl=EN
- https://search.naver.com/search.naver

III. Data Preprocessing

Data preprocessing is a technique that cleans and transform data into a compatible format to allow the model to interpret and understand it best. During the data collection phase, there were several irrelevant photos which do not belong to any of our specific categories and hence were not required within the scope of our project. Furthermore, there was also a great imbalance in the number of images for each respective category (cardboard, glass, metal, paper, plastic) for our dataset. Therefore, by collecting additional images by scrapping from different sources such as google and naver, and manually removing some images that were visually deemed to be too poor, we managed to reduce the noise in our dataset and also produce a much more evenly distributed dataset overall across the five categories. Lastly for data preprocessing, we created a simple python script to randomly allocate 80%, 10%, 10% of the dataset into training, validation and testing sets respectively. Training data is used to train the model, Validation data is used to validate the model as it is being trained and Testing data is used to test on the trained model to determine its performance. The final dataset will first be split up into training, validation, testing sets followed splitting again by category. As we are using keras to train our models, images can be resized easily using their ImageDataGenerator library and hence we will not be doing it here.

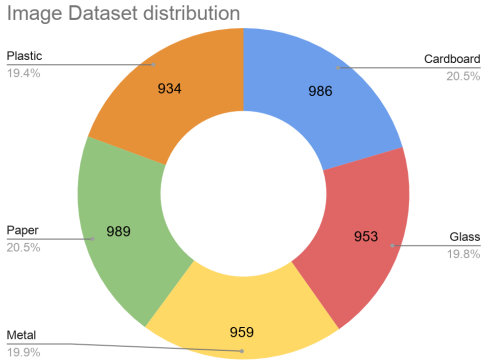

Figure 2: Dataset distribution of different recyclable waste categories

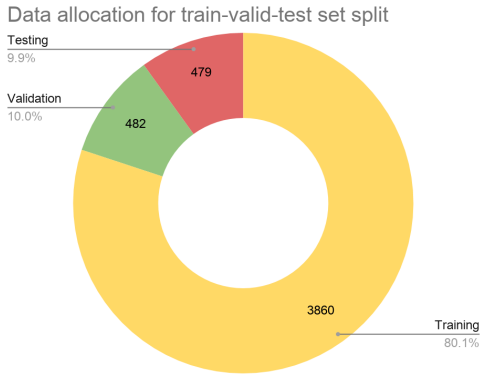

Figure 3: Data allocation for training, validation, testing set split

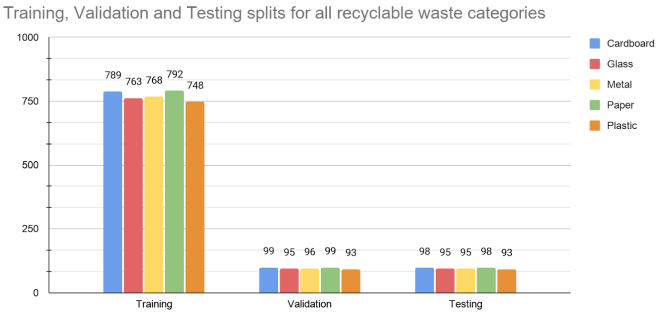

Figure 4: Training, Validation and Testing splits for all recyclable waste categories

IV. Problem and Algorithm/Mode

A. Problem Statement

Recycling is the process of collecting and processing materials that would otherwise be thrown away as trash and turning them into new products. This process, when done collectively by everyone, can provide many environmental and economical benefits. However, it is a behaviour where people engage in it less than they should. The reward for recycling and the repercussions for infrequently recycling are not immediate, hence this makes it difficult for people to want to change out of their usual behaviour of not recycling. People generally find it inconvenient or time consuming to dispose of their waste in recycling bins. Recycling bins are generally not widely available and people have to momentarily pause to think before deciding which category to throw their waste into. This hinders them from wanting to recycle their waste in recycling bins and rather throw them in general waste. We aim to make recycling easy by taking a step ahead to sort the waste for people when they dispose of their waste in the recycling bins. This creates a much more seamless recycling process, and hence further promoting recycling.

B. Solution

Our solution to this problem will be Recycl-EZ. It is a recycling bin with a unique feature which can help people classify their recyclable waste. This feature is a waste sorting mechanism to classify the waste before categorising and then segmenting it. Upon discarding the recyclable waste into the bin, a camera will capture the image of it and then classify using the sorting mechanism. The mechanism consists of a machine learning model where it is specifically trained to classify recyclable waste on certain specific categories. On detection and classification by the camera and the model respectively, the bin will then sort the waste according to its category type. This greatly reduces human error and time involved when it comes to sorting waste in bins, as well as greatly reducing the re-sorting efforts at centralised recycling plants. Siting of related works:

- Olugboja Adedeji, Zenghui Wang, Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network, Procedia Manufacturing, Volume 35, 2019, Pages 607-612, ISSN 2351-9789, https://doi.org/10.1016/j.promfg.2019.05.086. (http://www.sciencedirect.com/science/article/pii/S2351978919307231) [1]
- Md. Wahidur Rahman, Rahabul Islam, Arafat Hasan, Nasima Islam Bithi, Md. Mahmodul Hasan, Mohammad Motiur Rahman, Intelligent waste management system using deep learning with IoT, Journal of King Saud University - Computer and Information Sciences, 2020, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2020.08.016. (http://www.sciencedirect.com/science/article/pii/S1319157820304547) [7]

V. Convolutional Neural Network (CNN)

A. Motivation

We have decided to build our recyclable waste sorting mechanism for Recycl-EZ with Convolutional Neural network (CNN) deep learning models. One of the defining features for CNN for image classification is its convolutional layers that are present within the model. As we are dealing primarily with image classification, where the RGB pixels and dimensions of the images matters, CNN will be the best[1]. CNN is widely used for image data and classification prediction problems, and given that the CNN input is traditionally 2 dimensional, a field or a matrix, but is versatile enough to be changed to be one dimensional, it provides better internal representation.

B. Simple CNN Architecture

We implemented our very own simple CNN architecture as part of our naive approach at the start of our project where it consists of 5 layers taking input images of 224 by 224 in dimension. Using the keras library, we used a sequential type model as it allows us to build the model by implementing it layer by layer. To begin we have Conv 2D layers. These are convolutional layers that will deal with our input images, that are essentially 2D images. There are 16 nodes in the first layer and 32 nodes in the second layer and 64 in the third layer, all of them with a kernel size of 3, implying that we have a 3x3 filter matrix. For activation function, we used ReLU (Rectified Linear Activation) that is proven to work well in neural networks. [11] Towards the end of the layers, we have a flatten layer, which serves as a connection between the layers. Dense is the standard layer type we will utilize for our output layer, a common find in many cases for neural networks. Our number of nodes in our output layer will be 5, corresponding to the different image classification, one for each possible outcome. The activation is softmax which makes the output sum up to 1, and the model's prediction will be based on the category that has the highest probability.

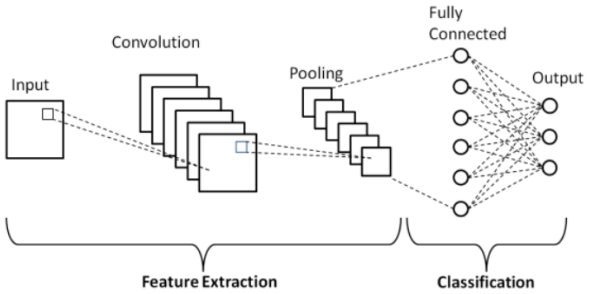

Figure 5: Layers Convolutional Neural Network

C. Pretrained Model Architectures with Transfer Learning

Apart from crafting our own CNN architecture from scratch where the layers are usually shallow with very simple layouts, we also looked into building our model upon other CNN architectures which are well-known for classifying images relatively accurately. Such architectures boast more convolutional layers, adding complexity and more filters to an input to create a feature map that summarises the presence of detected features in the input. Such model architectures are also already pre-trained on a large benchmark dataset such as Imagenet. Imagenet is a large visual database with over 15 million labelled high resolution images belonging to more than 20000 categories where it is used to provide a resource to promote the research and development of improved methods for computer vision. As such by having such pre-trained architectures into our model built, we are able to utilise Transfer Learning where the knowledge from pre-training is transferable to the problem that we want to solve.[2] [6] Instead of starting the learning process from scratch for our problem during training, it starts from the patterns that have been learned when solving a different problem. This leverages on previous learnings and avoids starting from scratch. This saves us time on training on these complex model architectures while yielding much better results. For the scope of the project, we have looked into and applied these two CNN architectures into our model: VGG-16 and Inception-V3. The VGG-16 is a relatively straightforward architecture whereas the Inception-V3 is much more complex one consisting of Inception modules.

D. VGG-16

The VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition".

[1] https://www.cv-foundation.org/openaccess/content$_cvpr_2016/html/Wang_CNN-RNN_AUnified_CVPR_2016_paper.html$

[2] https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

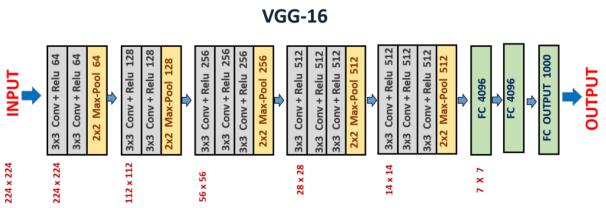

Figure 6: Layers of VGG 16 convolutional neural network model architecture

VGG-16 consists of a total of 16 layers of trainable parameters, 13 convolutional layers and 3 fully connected layers and takes in an input image size of 224 x 224. The architecture is relatively straightforward. It has 2 contiguous blocks of 2 convolution layers[3] [5] followed by a max-pooling, then it has 3 contiguous blocks of 3 convolution layers followed by max-pooling, and at last, we have 3 fully connected dense layers. The last 3 convolution layers have different depths in different architectures. After each max-pooling the size of the input images gets halved and eventually becomes 7x7.

E. Inception-V3

The Inception-V3 CNN architecture mainly focuses on consuming less computational power by modifying the previous Inception layers. This idea was proposed in the paper "Rethinking the Inception Architecture for Computer Vision", published in 2015. It was co-authored by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens.

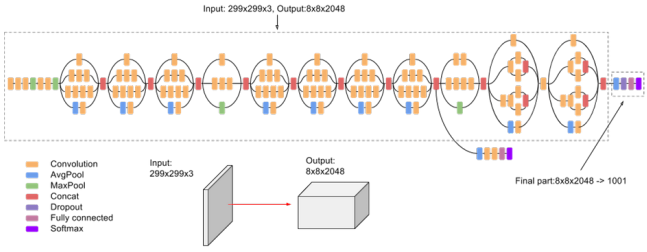

Figure 7: Layers of Inception-V3 convolutional neural network model

Inception-V3 consists of a total of 42 layers of trainable parameters with 11 inception modules stacked after 5 convolutional layers where it takes in an input image size of 299 x 299.[4] [10] Each Inception module consists of pooling layers and convolutional filters with rectified linear units as activation functions and are applied continuously instead of sequentially.[5]

[3] These modules are incorporated to allow for more efficient computation and deeper Networks through dimensionality reduction. This solves the problem of computational expense and overfitting due to its complex network.

VI. Evaluation Methodology

The performance of the models are evaluated using the accuracy metrics as our recyclable waste dataset class distribution is similar with almost an equal number of images in each category. In order to finalise and select the best CNN architecture to be implemented for our model used in the sorting mechanism, the comparison of performance between several models have to be clearly observed. The several models are constructed across 3 different architectures and tuning certain parameters.

CNN Architectures used:

1) Simple CNN
2) VGG-16
3) Inception-V3

Parameters tuned for comparison within each architecture:

1) Comparing best performance between applying Flatten, GlobalMaxPooling or GlobalAveragePooling before the fully connected dense layer.
2) (Using the best performance in 1, use it to make further comparison) Comparing between addition of Dropout before the last fully connected dense layer and without Dropout.
3) Comparing between further retraining of the last few layers of pretrained architecture (applies to only VGG-16 and Inception V-3) and no retraining.
4) Comparing between addition of Data Augmentation of training images fed into the model and no Data Augmentation at all.

(fixed values of the parameters will be stated in the table below)

There are infinite approaches to tuning the hyperparameters for CNN deep learning models which affects the result in its performance. However, we will only be doing our comparison of model performances with respect to the following parameters as mentioned above.

We chose to tune between Flatten, GlobalMaxPooling (GMP) and GlobalAverage Pooling (GAP)[6] [9] as there were articles mentioning the use of Pooling over Flatten to prevent overfitting when training the data as it further reduces the amount of training parameters used in the model. Flatten is a relatively straightforward process where it just converts a three-dimensional tensor hxwxd to one-dimensional by simply lining up all the values into a single dimension 1xhwd. Pooling is used to reduce the spatial dimensions of a three-dimensional tensor by dimensionality reduction where a tensor

[3]https://medium.com/towards-artificial-intelligence/the-architecture-and-implementation-of-vgg-16-b050e5a5920b

[4]https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

[5]https://deepai.org/machine-learning-glossary-and-terms/inception-module: :text=Inception%20Modules%20are%20used%20in,as%20overfitting%2C%20among%20other%20issues.

[6]https://datascience.stackexchange.com/questions/28120/globalaveragepooling2d-in-inception-v3-example/28155

with dimensions h×w×d is reduced in size to have dimensions 1×1×d. GMP reduces each h×w feature map to a single number by simply taking the max of hw values while GAP[7] [8] takes the average of it.

We added the choice of having Dropout as a tuned hyper-parameter to prevent overfitting of the model during training as there is a possibility that there are irrelevant nodes which might be too specific to the training dataset and might not be generalised enough for other images.

We added the choice of having the last few layers of the pretrained architectures[8] [4] to be retrained again while freezing the weights of the other layers as there is a possibility of yielding better performance. When having the weights in the last few layers to be re-trained again allows the model to focus on learning dataset-specific features and be much more relatable in solving our problem for our own dataset.

The last tuned parameter that we have chosen to include will be the application of Data Augmentation[9] [2] to the training images when training the model. Image Data Augmentation is a technique that can be used to artificially increase the size of the training dataset by constructing modified versions of the training images. This can result in the ability to generate more skillful models as the augmentation techniques create variations of the image that can improve the models ability to generalise and work on other images better. This technique is applied when the size of training data is too small and unable to train accurately and having this technique applied properly in balance will also reduce the problem of overfitting.

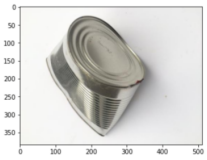

Figure 8.1. Original image

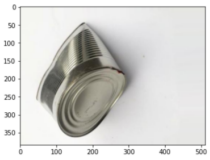

Figure 8.2. Augmented image 1

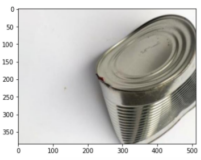

Figure. 8.3. Augmented image 2

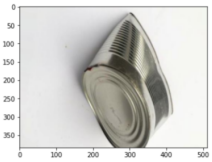

Figure 8.4. Augmented image 3

[7]https://www.quora.com/Why-was-global-average-pooling-used-instead-of-a-fully-connected-layer-in-GoogLeNet-and-how-was-it-different

[8]https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

[9]https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

Standardised parameter values and choices	
Optimizer	ADAM
Learning Rate	0.0001
Epoch	30
Loss	Categorical Cross Entropy
Layer applied to final Fully Connected Dense layer (select one)	Flatten GlobalMaxPooling2D GlobalAveragePooling2D
Dropout, if applied	0.2
Retrain layer of Pre-trained Model Architectures, if applied (VGG_16, Inception_V3)	Last 3 layers [:-3]
Data Augmentation, if applied	shear_range = 0.2, zoom_range = 0.2, width_shift_range=0.2, height_shift_range=0.2, rotation_range=20, horizontal_flip = True, vertical_flip = True

Figure 9: Display of fixed parameters that are applied across all models

For our optimiser, we used the Adam optimiser which is a replacement optimisation algorithm for stochastic gradient descent for training deep learning models specific to our use case. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

As for learning rate and Epochs, we set it to 0.0001 and 30 respectively after experimentation, as it yielded the best results. They are coupled with the categorical cross entropy loss, otherwise known as softmax loss, that serves to train a CNN to output a probability over the 5 classes for each image, thus being applicable for our use case of multi-class classification.

VII. Results Discussion

Among the three different architectures and having the parameters tuned as mentioned above, our model performs the best at 94.36% when utilising the Inception-V3 architecture and performs the worst at 53.44% when utilising the simple CNN architecture which we constructed on our own via the naive approach.

Accuracies of Model on Test Dataset	Simple CNN
Flatten/Pooling before FC Dense Layer	60.33% (GAP)
Addition of Dropout	61.38% (GAP)
Addition of Dropout + Data Augmentation	53.44% (GMP)

Figure 10: Result of simple CNN

Accuracies of Model on Test Dataset	VGG-16	Inception-V3
Flatten/Pooling before FC Dense Layer	80.90% (Flatten)	91.65% (GAP)
Addition of Dropout	82.05%	92.69%
Addition of Dropout + Re-training of Layers	86.64%	90.81%
Addition of Dropout + Data Augmentation	84.34%	93.95%
Addition of Dropout + Re-training of Layers + Data Augmentation	**90.1%**	**94.36%**

Figure 11: Result of VGG-16 and Inception-V3

Entire documentation of our validation/testing results of all our models can be viewed here: https://docs.google.com/document/d/1CxpNfiXHuDJdfo8SAcN-DUkofP126gddliqh53KkPyc/edit?usp=sharing

Here are the findings and inferences we have gathered from the test results from all our models across the different architectures and tuned parameters.

A. Performance of model with Simple CNN implementation:

The Simple CNN implementation was expectedly yielding much poor results with the range of accuracies on the test set reflecting around the 50% - 60% range. In general the models with the simple implementation does not seem to overfit nor underfit based on the accuracy and loss graphs and there is a possibility that the performance might improve with further epochs. Evidently when compared to the 2 other pretrained model architectures. This implementation was used as our initial naive model approach to our problem solution.

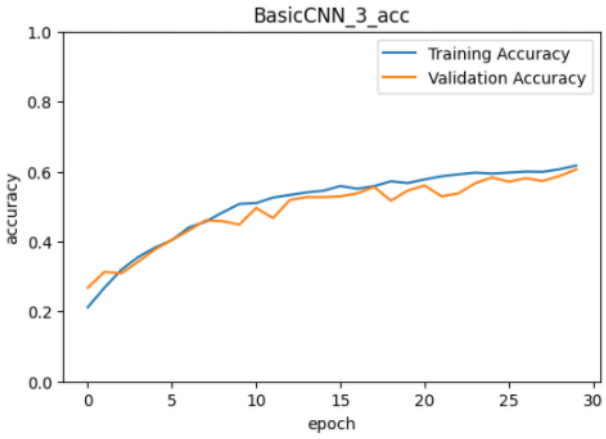

Figure 12: Training and Validation Accuracy for the best performing Naive CNN model

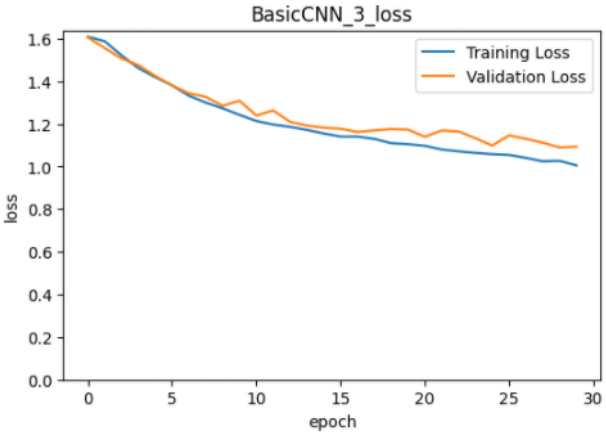

Figure 13: Training and Validation Loss for the best performing Naive CNN model

	precision	recall	f1-score	support
cardboard	0.75	0.85	0.79	98
glass	0.55	0.56	0.55	95
metal	0.54	0.53	0.53	95
paper	0.55	0.65	0.6	98
plastic	0.69	0.47	0.56	93
accuracy			0.61	479
macro avg	0.62	0.61	0.61	479
weighted avg	0.62	0.61	0.61	479

Figure 14: Classification report for best performing Naive CNN model

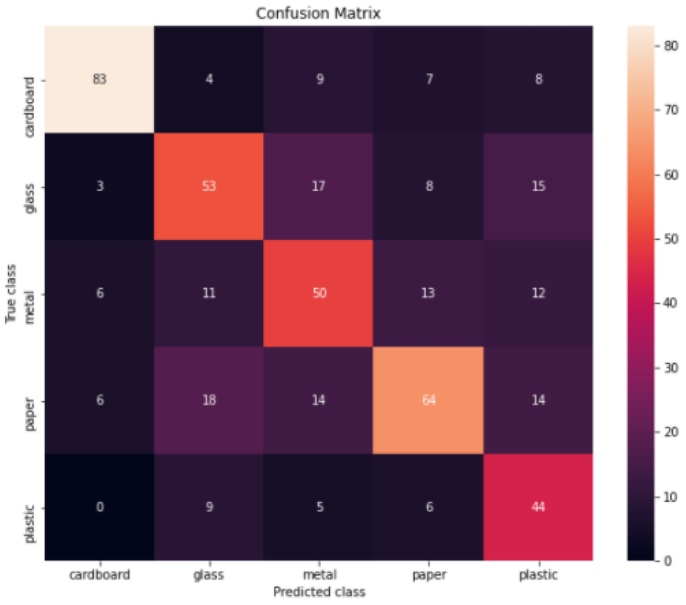

Figure 15: Confusion Matrix for the best performing Naive CNN model

B. Performance of mode with VGG-16 implementation:

The VGG-16 implementation shows a significant improvement in performance in the test accuracies as compared to the simple CNN implementation having the model accuracies ranging above 80%. It works the best when having the data flattened right after the pre-trained model and right before the fully connected dense layer. Model improves when dropout of 0.2 is applied and improves much significantly when the last 3 layers of the pre-trained VGG-architecture is retrained. The best VGG-16 implementation is a result of adding dropout, having the layers re-trained and data augmentation. However, the model seems to be slightly overfitted based on the loss graph as despite training loss decreasing consistently per epoch, validation did not improve. From the classification report and confusion matrix, classification is done relatively

well on the test set except that a noticeable amount of glass and metal is recognised as plastic and glass respectively.

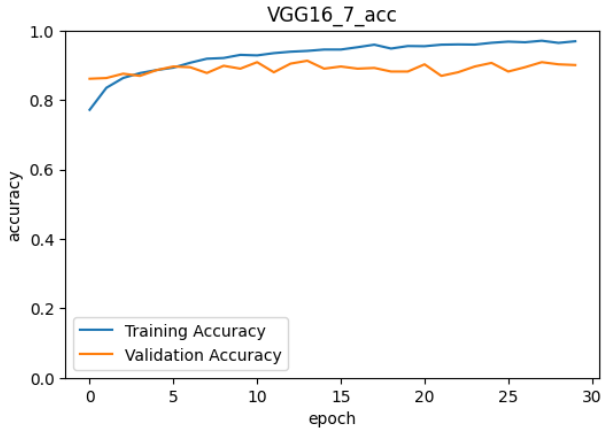

Figure 16: Training and Validation Accuracy for the best performing VGG-16 model

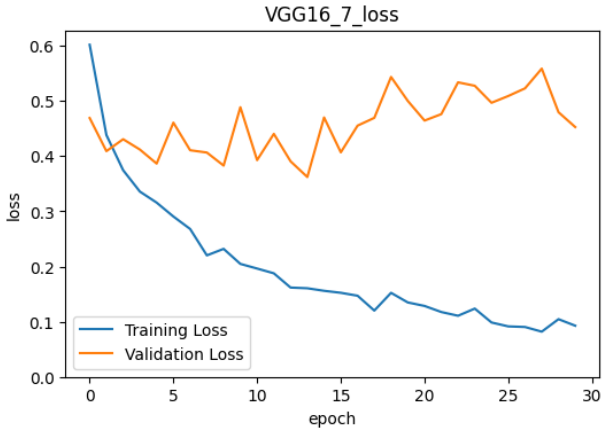

Figure 17: Training and Validation Loss for the best performing VGG-16 model

	precision	recall	f1-score	support
cardboard	0.98	0.96	0.97	98
glass	0.87	0.85	0.86	95
metal	0.87	0.93	0.9	95
paper	0.89	0.93	0.91	98
plastic	0.9	0.84	0.87	93
accuracy			0.9	479
macro avg	0.9	0.9	0.9	479
weighted avg	0.9	0.9	0.9	479

Figure 18: Classification report for best performing VGG-16 model

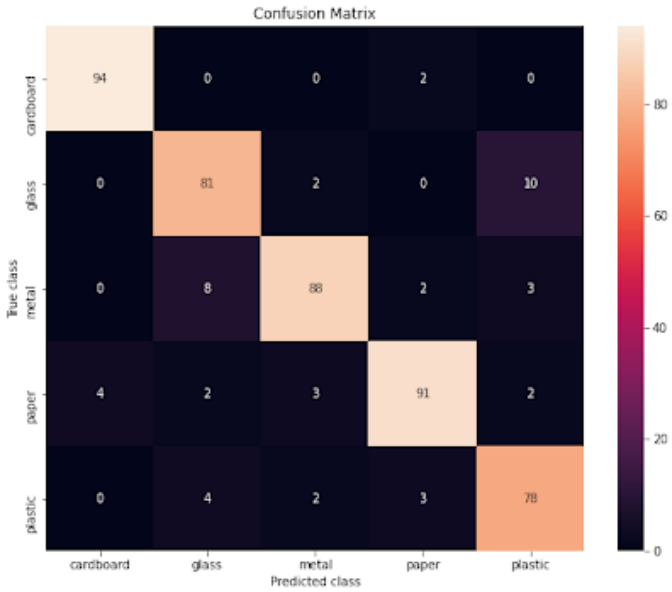

Figure 19: Confusion Matrix for the best performing VGG-16 model

C. Performance of model with Inception-V3 implementation:

The Inception-V3 implementation is very accurate and performs even better than the VGG-16 implementation, having the model accuracies ranging above 90%. It works the best when having the data undergo average pooling right after the pre-trained model and right before the fully connected dense layer. Model shows improvement with the addition of dropout or when having the data augmented however it doesn't work well when the last 3 layers are retrained. However, the best Inception-V3 implementation parameters are similar to VGG-16's where best performance results yield when adding dropout, having the layers re-trained and data augmentation. The best performing model results yield at 94.4%. There is a possibility that the model might improve further with more epochs given that both training and validation accuracies and losses in the graphs are increasing and decreasing together respectively per epoch. From the classification report and confusion matrix, classification of the test set shows a slight improvement from the VGG-16 implementation, however having the same minor issue that some amount of glass is still recognised as plastic.

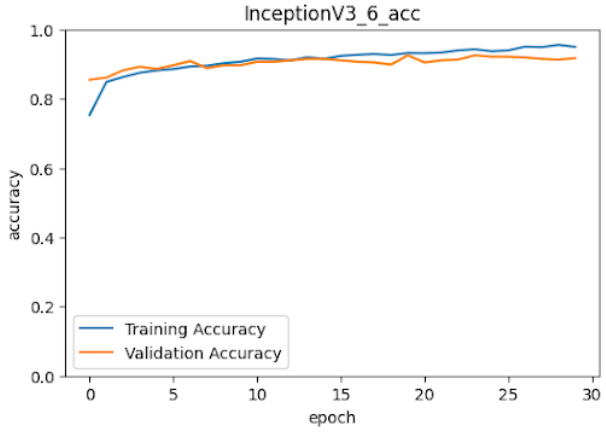

Figure 20: Training and Validation Accuracy for the best performing Inception-V3 model

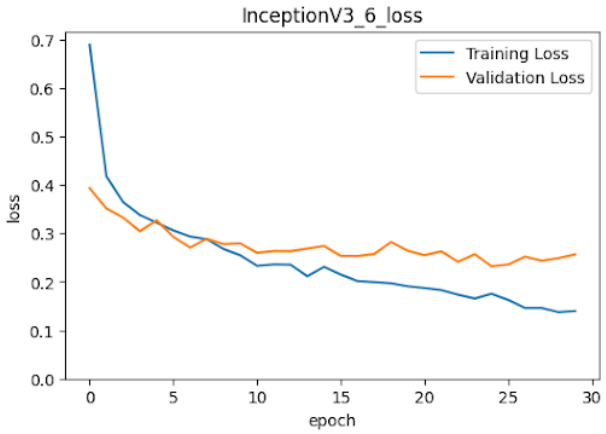

Figure 21: Training and Validation Loss for the best performing Inception-V3 model

	precision	recall	f1-score	support
cardboard	0.97	1	0.98	98
glass	0.93	0.93	0.93	95
metal	0.91	0.98	0.94	95
paper	0.96	0.97	0.96	98
plastic	0.95	0.84	0.89	93
accuracy			0.94	479
macro avg	0.94	0.94	0.94	479
weighted avg	0.94	0.94	0.94	479

Figure 22: Classification report for best performing Inception-V3 model

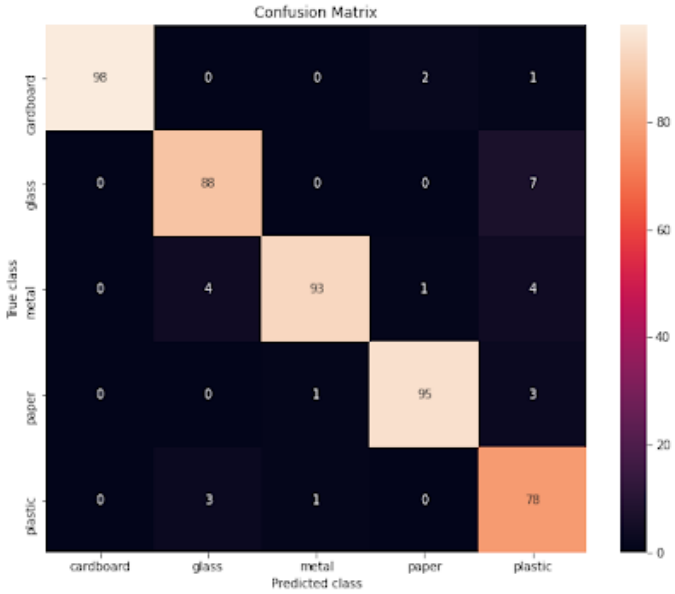

Figure 23: Confusion Matrix for the best performing Inception-V3 model

D. *Conclusion:*

After experimenting with different models, fine tuning our parameters and using special techniques like data augmentation, we finally managed to produce a reasonably successful image classification model. Our research project to classify recyclables ends with our best performing model being the Inception-V3 implementation at an accuracy of 94.4%. This is a vast improvement as compared to when we first started off with the naive approach of implementing our own simple CNN architecture at an accuracy of 61.38%. From this project we realised that there were several limitations. Firstly, our classifier is only able to classify to a limit of 5 different categories and we should implement an additional category to identify non-recyclable waste to allow users to dispose of non-recyclables as well if they choose to. Secondly, our classifier is also limited to identifying a single object when classifying and we could improve our classifier by identifying multiple objects in the same image. Lastly, there was not much experimentation on how we augment our images during training and we believe it is something to look into as certain data augmentations can definitely improve and better generalise our model better.

VIII. Implementation of Model into our Design Solution: (ReactJs + Flask)

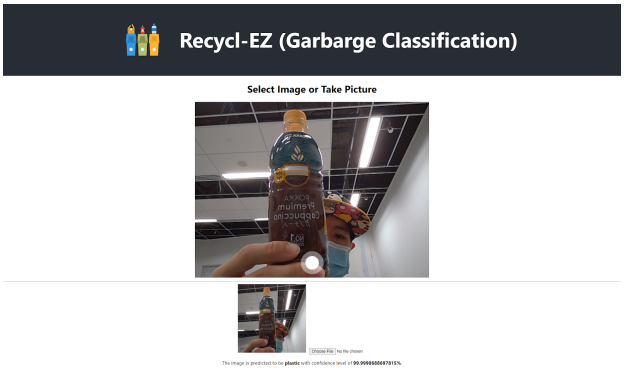

Figure 24: Frontend User Interface implementation for recyclable waste classification

For demonstration of Recyl-EZ's sorting mechanism of recyclable waste classification, we deployed a webpage using Reactjs as our frontend and our trained model is deployed in a flask server. When the user enters the page, they can choose to either upload an image through their local file system or take a picture using their device's camera. The image will then be sent to the flask server and the server will reply with the label type and confidence level to the user. [1]

The following links are the documentation of the entire project and the Github repository for the frontend UI:
https://drive.google.com/drive/folders/1CQJvljtZtGnTVJrAm8R8MA4yVc-x26nn?usp=sharing
https://github.com/EclipseEsp/50.038$_C DS$

References

[1] Olugboja Adedeji and Zenghui Wang. Intelligent waste classification system using deep learning convolutional neural network. *Procedia Manufacturing*, 35:607–612, 2019.

[2] Jason Brownlee. How to configure image data augmentation in keras. *Machine Learning Mastery [en línea]. Disponible en: https://machinelearningmastery. com/how-to-configure-image-dataaugmentation-when-training-deep-learning-neural-networks*, 2019.

[3] DeepAI. Inception module, 2019, May 17.

[4] Dishashree GuptaDishashree. Transfer learning: Pretrained models in deep learning, 2017, June 1.

[5] V. Khandelwal. The architecture and implementation of vgg-16, 2020, August 18.

[6] P Marcelino. Transfer learning from pre-trained models, towards data science, 2018, October 23.

[7] Md Wahidur Rahman, Rahabul Islam, Arafat Hasan, Nasima Islam Bithi, Md Mahmodul Hasan, and Mohammad Motiur Rahman. Intelligent waste management system using deep learning with iot. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[8] Akarsh Rajvanshi. Why was global average pooling used instead of a fully connected layer in googlenet, and how was it different?, 2019, May 20.

[9] Tophat. Globalaveragepooling2d in inception v3 example.

[10] S. Tsang. Review: Inception-v3 - 1st runner up (image classification) in ilsvrc 2015, 2019, March 23.

[11] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.