
# Recyclable Garbage Classification

CLASS PROJECT

Yuelan Zhu (yz4160) | EAEE E4000 | Fall 2021

# Contents

# Introduction

Waste recycling is necessary for a sustainable society. However, the current method to sort garbage is low efficient, which in most case is employing human workers to sort by hand or using a series of large filters to separate out.  A typical garbage sorting system includes plate feeder, rotatory screening machine, winnowing separator, magnetic separation machine, and manual sorting platform, which can separate trash of different sizes, types, specific gravities and other characteristics. It costs lots of time of human workers and energy of filtration facilities. Plus, overexposure to some hazardous waste and by-products provided in sorting process may cause serious health problems to workers.

Moreover, urban garbage sorting policy is implemented in more and more cities with a growing requirement of people to hygiene issues. In many cities, domestic trash can be generally divided into three categories: recyclable waste, kitchen waste and hazardous waste. Recyclable trash can be reused and save resources. Thus, it is necessary to sort them correctly during recycle process. However, consumers may be confused about how to determine the correct way to dispose of a large variety of materials. It is vital to develop an intelligent system that can effectively detect the components of garbage.

The objective of my project is finding a method that can automatically sort the recycle waste. Based on computer vision theory, image classification is a popular approach used to detect and distinguish features of pictures. With labeled pictures, a well-trained model can define a set of target classes based on specific rules.

There are some models that I would like to apply. First are simple Convolutional neural networks (CNN) and deeper CNN and try to find the optimal hyperparameters. Then I also want to try ResNet50 and VGG16, winners of ImageNet challenge and hope they can perform better.

This method has the potential to improve processing plants with more efficiency and help reduce waste, since it is not always the case that the employees sort everything with 100% accuracy. It will not only have positive environmental effects but also beneficial economic effects.

## Data Source

The dataset that I will use is downloaded from Kaggle[1], which contains 6 classifications: cardboard (393), glass (491), metal (400), paper(584), plastic (472) and trash(127) in total of over 2000 pictures, in format of jpg. Here are example pictures for each category:

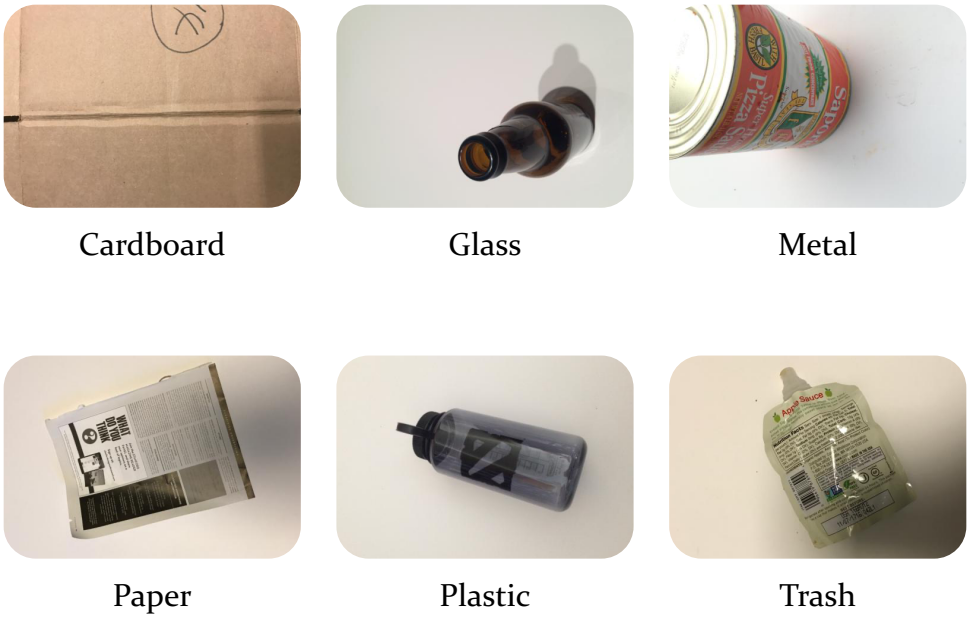



Figure 1. Example pictures for each category

Three txt files are also provided, which contain filename of pictures that have been divided into training dataset, validation dataset, and test dataset. There are 1768 images for training, 328 images for validation and 431 images for test.

# Data Preparation

Since the training, validation and test datasets are already provided, I do not need to separate the raw dataset. Before using them, it is necessary to check the distribution:

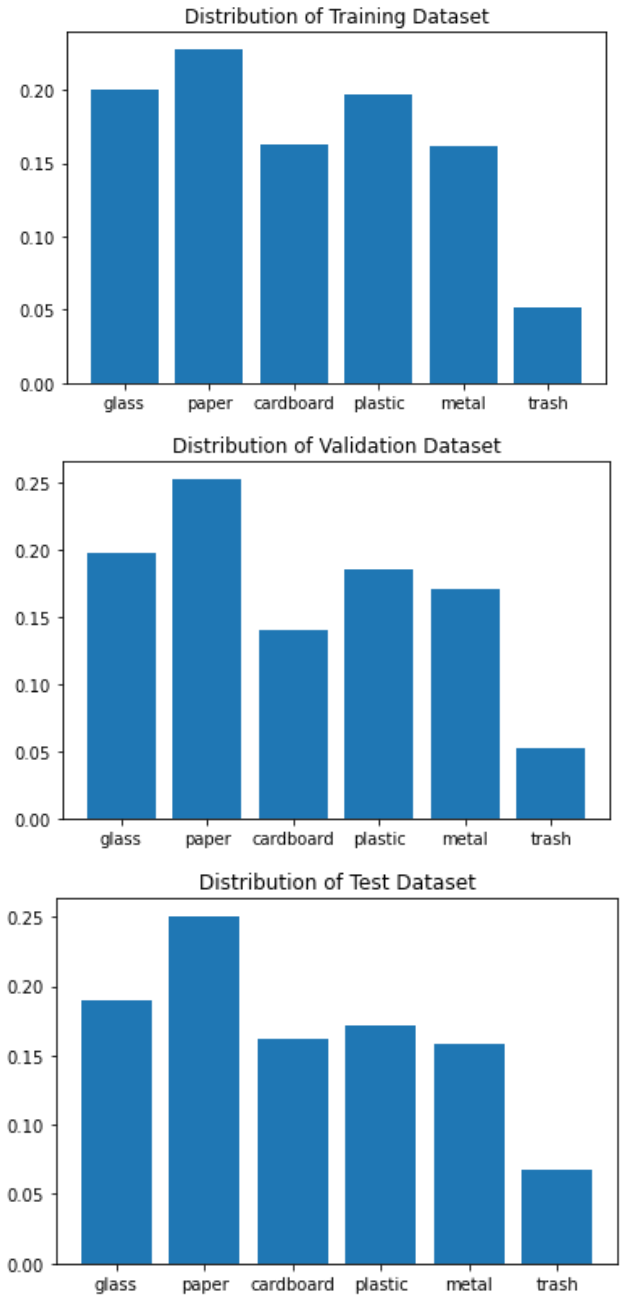



Figure 2. Distribution of training, validation and test dataset.

Distribution of classes in these three datasets are generally similar, with the most pictures labeled as “paper”, and the least pictures labeled as “trash”.

The original image size is (384, 512, 3). Rotating image data to generalize it. To satisfy the input requirement of different models, these image data also need pre-processing. For CNN, just dividing the values by 255 to normalize them. For other models, there are different pre-processing functions to make data suitable to input.

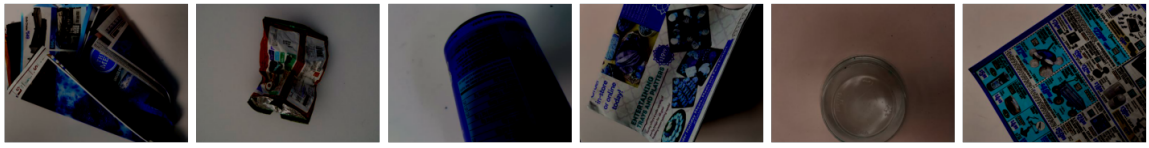


Figure 3. Examples of pictures after pre-processing.

# Methodology

## CNN

A convolutional neural network (CNN) is a type of neural network popularly dealing with image recognition and processing, which uses spatial information to reduce dimensionality and extract features of the problem. A typical CNN consists of several convolutional layers pooling layers and fully connected layers.

### Simple CNN

My first attempt is based on the MNIST notebook shown in the class. Nothing was changed of this simple CNN model except the input shape in the first layer to fit the picture size, and the number of filters in the last convolutional layer to fit the number of classes.

Table 1. Simple CNN architecture

| Layer | Filters | Kernel size | Stride | Padding | Output shape |
|---|---|---|---|---|---|
| Conv1 | 32 | 5 | 2 | Same | (None, 192, 256, 32) |

| | | | | | |
|---|---|---|---|---|---|
| Conv2 | 64 | 3 | 2 | Same | (None, 96, 128, 64) |
| Conv3 | 128 | 3 | 2 | Same | (None, 48, 64, 128) |
| Conv4 | 6 | 3 | 2 | Same | (None, 24, 32, 6) |
| Global average pooling | | | | | (None,6) |

## Deeper CNN

Since the garbage dataset is much more complex than handwritten numbers, I decided to modify the CNN model. There are three convolutional blocks followed by two fully connected layers and a soft-max on six classes. Use max pooling in first two convolutional layers and global average pooling in the last convolutional layer.

Table 2. Deeper CNN architecture

| Layer | Filters/ | Kernel size | Stride | Padding | Output shape |
|---|---|---|---|---|---|
| Conv1 | 32 | 5 | 2 | Same | (None, 192, 256, 32) |
| MaxPooling1 | | | | | (None, 96, 128, 32) |
| Conv2 | 64 | 3 | 2 | Same | (None, 48, 64, 64) |
| MaxPooling2 | | | | | (None, 24, 32, 64) |
| Conv3 | 128 | 3 | 2 | Same | (None, 12, 16, 128) |
| Global average pooling | | | | | (None, 128) |
| Fully connected layer1 | | | | | (None, 64) |

| | | | | | |
|---|---|---|---|---|---|
| Fully connected layer2 | | | | | (None, 32) |
| Classifier | | | | | (None, 6) |

This model was trained using “Adam” algorithm, with learning rate equal to 0.001, batch size equal to 32, and 200 epochs. Loss function was categorical crossentropy because the required output was 6 classes.

## Transfer Learning

Complex deep learning models have millions of parameters (weights), and training them from scratch usually requires a lot of computing resources and data. Transfer learning is a technique that simplifies most of the work by taking part of a model that has been trained on related tasks and reusing it in a new model[2].

To build an image recognizer for recycle garbage more rapidly, I would like to take advantage of a pre-trained model to detect garbage classes, which let me adapt the existing knowledge in the pre-trained model using much fewer training data than the original one required, and it is practical to train the customizing model in my own device.

Most of the original weights would not change. I removed the final layer and add a new layer that fits the classes, and trained some new layers on top of the model. The base models I chosen are ResNet50 and VGG16.

### ResNet50

Deep residual networks (ResNet) consist of many stacked “Residual Units” [3].

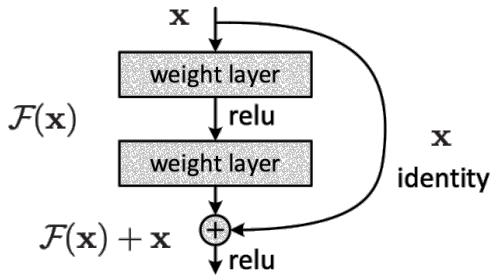



Figure 4. Residual learning: a building block

The central idea of ResNet is to learn the additive residual function F, which is realized by attaching an identity skip connection (“shortcut”). This structure can avoid gradient vanish or explosion. Thus, it is possible to create a deeper network structure by applying residual function and get a better accuracy.

I chose ResNet50 as the baseline model, and added two fully connected layers: one had 1024 units, the other is a six-classes soft-max classifier layer. Only parameters of these layers need to be trained with a total number of 2,104,326, much fewer than 25,692,038, the number of parameters of the whole ResNet50 model.

Table 3. ResNet50 architecture

| Layer | Structure |
|---|---|
| Conv1 | 7×7, 64, stride 2 |
| Conv2 | 3×3 max pool, stride 2 |
| | $\begin{bmatrix} 1\times1 & 64 \\ 3\times3 & 64 \\ 1\times1 & 256 \end{bmatrix}\times3$ |
| Conv3 | $\begin{bmatrix} 1\times1 & 128 \\ 3\times3 & 128 \\ 1\times1 & 512 \end{bmatrix}\times4$ |
| Conv4 | $\begin{bmatrix} 1\times1 & 256 \\ 3\times3 & 256 \\ 1\times1 & 1024 \end{bmatrix}\times6$ |
| Conv5 | $\begin{bmatrix} 1\times1 & 512 \\ 3\times3 & 512 \\ 1\times1 & 2048 \end{bmatrix}\times3$ |
| Fully connected layer1 | 1024 global average pool |
| Classifier | 6, softmax |

I used the same gradient decent method, learning rate and batch size as when training CNN model. The number of epochs is 30. I also used an early stop callback function to stop training when validation loss has stopped improving.

## VGG16

VGG shows a significant improvement on the prior-art configurations can be achieved by increasing the depth to 16-19 weight layers, which is substantially deeper than what has been used in the prior art. To reduce the number of parameters in such very deep networks, it uses very small 3×3 filters in all convolutional layers (the convolution stride is set to 1) [4].

I chose VGG16 as the baseline model, and added three fully connected layers: one had 512 units, one had 126 units and other is a six-classes soft-max classifier layer. Dropout is used to reduce information. Only parameters of these layers need to be trained with a total number of 330,374, much fewer than 15,046,342, the number of parameters of the whole VGG16 model.

Table 4. VGG16 architecture

| Block | Layer | Units |
|---|---|---|
| Input size (384, 512, 3) | | |
| 1 | Conv1 | 64 |
| | Conv2 | 64 |
| Max pooling | | |
| 2 | Cov1 | 128 |
| | Cov2 | 128 |
| Max pooling | | |
| 3 | Cov1 | 256 |
| | Cov2 | 256 |
| | Cov3 | 256 |
| Max pooling | | |
| 4 | Cov1 | 512 |
| | Cov2 | 512 |
| | Cov3 | 512 |
| Max pooling | | |
| 5 | Cov1 | 512 |
| | Cov2 | 512 |
| | Cov3 | 512 |
| Max pooling | | |
| Global average pooling | | |
| 6 | Fully connected layer1 | 512 |
| | Batch normalization | 512 |
| | Dropout layer, 0.2 | |
| 7 | Fully connected layer2 | 128 |
| | Batch normalization | 128 |
| | Dropout layer, 0.2 | |
| | Classifier | 6, softmax |

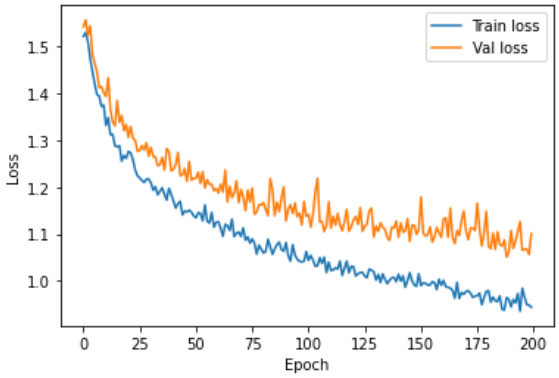
I used the same gradient decent method, learning rate and batch size as when training CNN model. The number of epochs is 30. I also used an early stop callback function to stop training when validation loss has stopped improving.

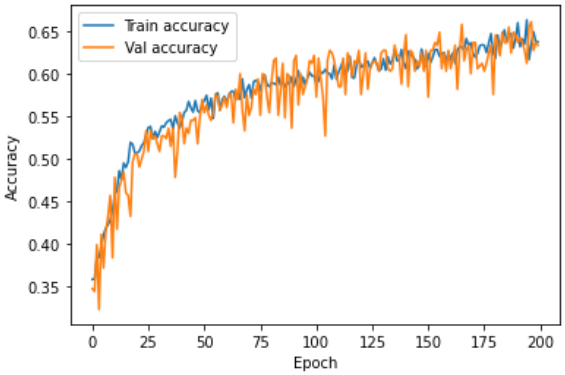
# Results

## Metrics of Training Process

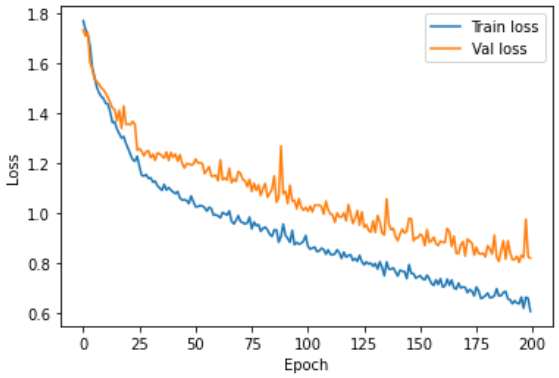
The loss and accuracy were monitored during training process.

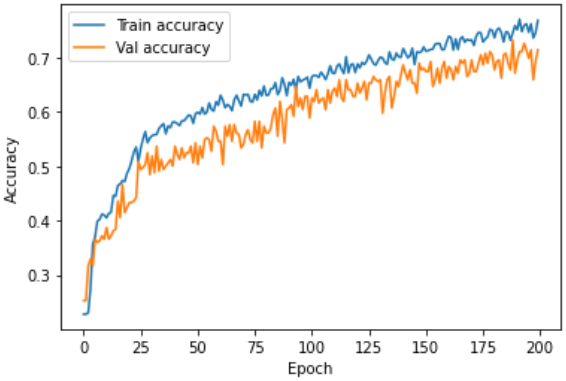
a. Simple CNN



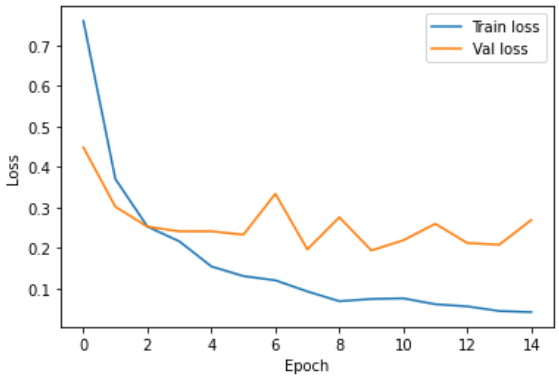



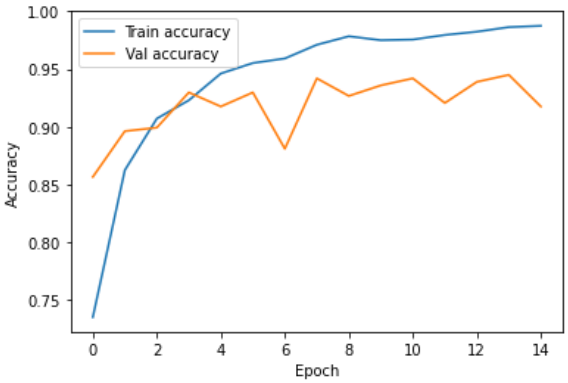

b. Deeper CNN




c. ResNet50



d. VGG16

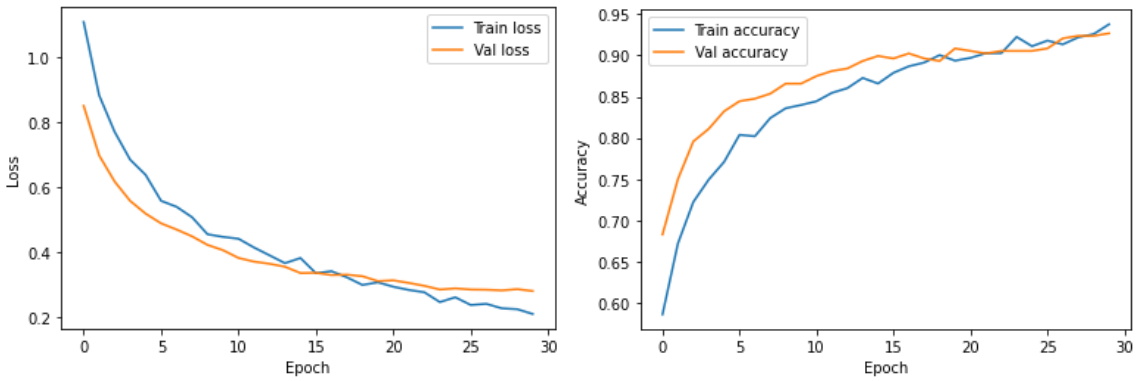



Figure 5. Loss and accuracy during training process

For classification, accuracy should be focus on. I expected validation accuracy should be as high as possible.

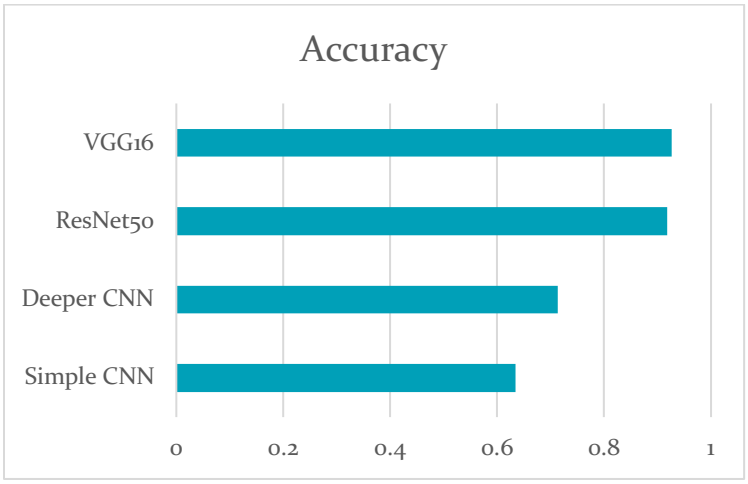



Figure 6. Accuracy of different models

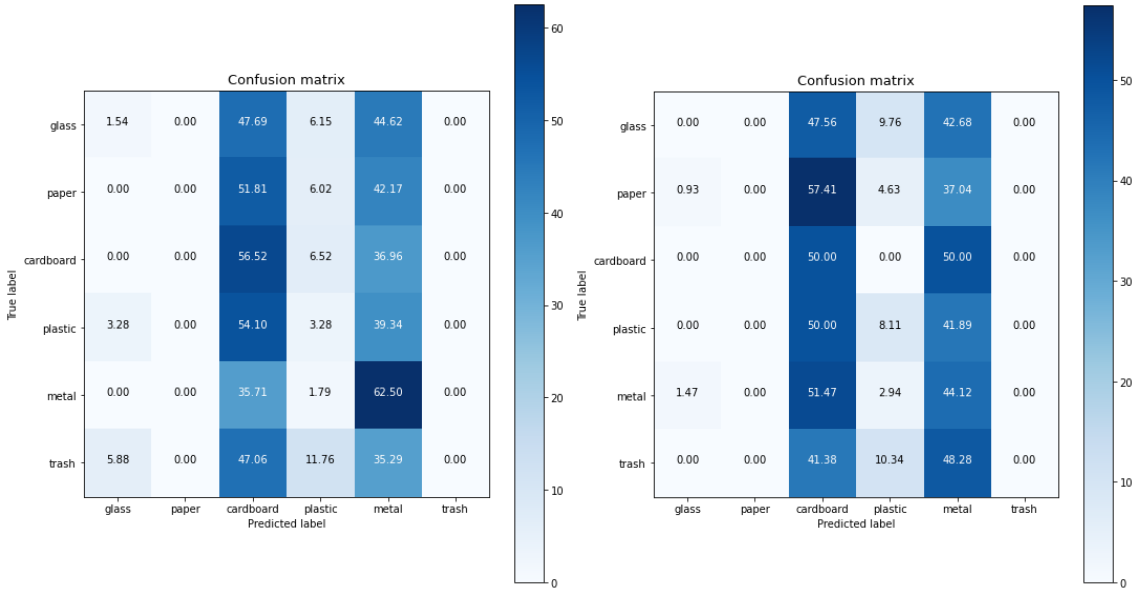
Deeper CNN perform better than the simple one, implying that a more complex approach can provide more useful information about dataset to predict. However, two CNN models have the most epochs and the lowest accuracy. Only 71% accuracy was achieved after I modified a deeper CNN model. The loss and accuracy graph indicates that they were still underfitting after 200 epochs, but it may cost lots of time to reach a relatively better performance. When I tried to add more layers to this model, the accuracy became lower, which indicated the structure of 3 convolutional layers was the most suitable one for this dataset. Also, finding good hyperparameters needs much of time even with some hyperparameter tuning libraries (like Sherpa or Optuna). This was not cost effective because there were too many parameters to train. Thus, I would like to use some pre-trained models.

The accuracy of the transfer leaning models based on both ResNet50 and VGG16 reached over 90%, superior to the former ones of CNN models, with much less epochs. For ResNet50, it even stopped training before 15$^{th}$ epoch, because the loss became low and mild decreasing according to the early stop callback function. These two models have more complex networks than traditional CNN but their performance did not degrade even I added more fully connected layers to them, which showed they are more advanced in inner structure.

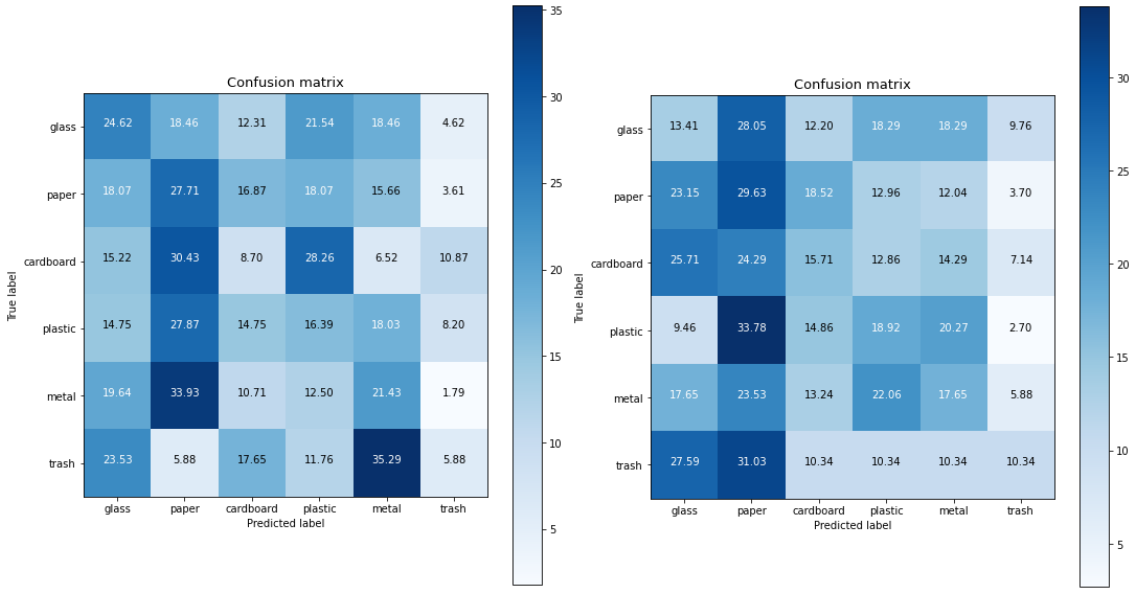
## Confusion Matrix

I plot the confusion matrices of different models to further explore their performance.

a. Deeper CNN (validation on the left, test on the right, same as below)



b. ResNet50




c. VGG16

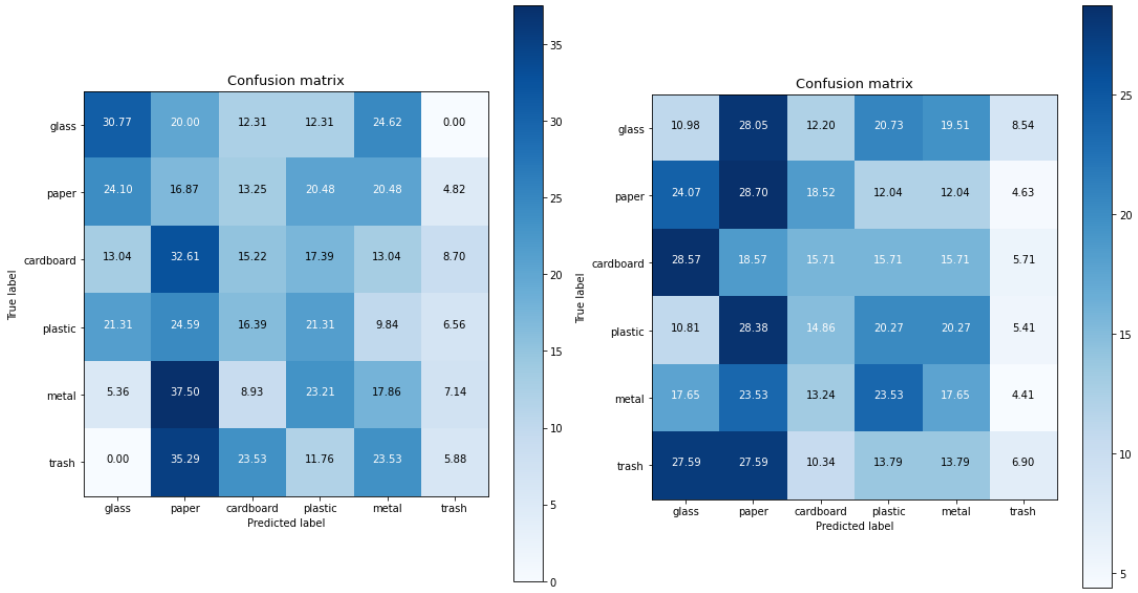



Figure 7. Confusion matrices of different models

Although the confusion matrix reflects that CNN has a relative high accuracy in predicting cardboard and metal, it is meaningless because it is always predicting a cardboard or metal class. It is weird because such case usually happens when data of one class are much more than others. However, as what is shown in distribution of three datasets, images of paper are the most. I think of a possible explanation to this result: when I rescaled the values by dividing them by 255, most of them were in a range. In

other words, after normalization, the values in a specific range had higher frequency. When they were classified during training process, the values were identified as the same class even the original ones were from different classes. As for this data source, most of them were recognized as cardboard and metal by CNN model. If that is the case, I would like to generalize images from other classes to improve this model.

ResNet50 has higher predictive accuracy of paper and metal. As for the original distribution, paper is 25% of validation dataset and test dataset, and metal is approximately 17%. The confusion matrices show that ResNet50 model can correctly identify 28% paper images, 21% metal images in validation dataset, and 30% paper images, 18% metal images in test dataset. The proportion with right labels of other classes is even lower than the original distribution.

VGG16 has higher predictive accuracy of plastic. Plastic image is 18% of validation dataset and test dataset. VGG16 can identify 21% of them in validation dataset and 20% in test dataset. The proportion with right labels of other classes is even lower than the original distribution.

Moreover, I notice that these models performed badly in predicting trash images. A manual inspection at trash images reveals that this class consists of various types of garbage, such as tissue, food packaging, toothpaste and so on, which results in difficulty in learning features. Plus, trash image has the fewest quantity, and model lacked of training to learn trash image.

# GitHub Repository

Code can be found at https://github.com/yz4160/ml-project, which is still under construction.

# Conclusions and Recommendations

F1 score is the harmonic mean of precision and recall, which is usually used to evaluate the overall performance of models. The higher F1 score is, the more robust the model is. Considering the the sample numbers of each category are not equal, I used weighted average F1 score.

Table 5. F1 score of different models

| Model | CNN | | ResNet50 | | VGG16 | |
|---|---|---|---|---|---|---|
| | validation | test | validation | test | validation | test |
| F1 score | 0.11 | 0.10 | 0.20 | 0.19 | 0.20 | 0.18 |

ResNet50 has a higher F1 score than others, which is also illustrated by confusion matrix discussed before. Thus, I recommend future modification can be based on this approach. To get a better performance, I have two suggestions:

1. Enrich data source. Larger dataset can provide more information and help model better learn features.
2. Separate trash images and others. Train a special model for trash category and combine it with the model for others.

# References


1. https://www.kaggle.com/asdasdasasdas/garbage-classification
2. https://www.tensorflow.org/js/tutorials/transfer/what_is_transfer_learning?hl=en
3. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
4. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

Cover image from: https://www.echemi.com/cms/58653.html