VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



**A MODEL FOR CLASSIFICATION OF WASTE BASED ON CNN AND IMAGE PROCESSING TECHNIQUES**

By

PHAM CONG TUAN

A thesis submitted to the School of Computer Science and Engineering

in partial fulfillment of the requirements for the degree of   
Bachelor of Information Technology/Computer Science/Computer Engineering

Ho Chi Minh City, Vietnam  
Year 2022

**A MODEL FOR CLASSIFICATION OF WASTE BASED ON CNN AND IMAGE PROCESSING TECHNIQUES**

APPROVED BY: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ,   
Nguyen Van A, Ph.D., Chair (*Example*)   
\*(*Type Committee names beneath lines*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
(*Typed Committee name here*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
(*Typed Committee name here*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
(*Typed Committee name here*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
(*Typed Committee name here*)

THESIS COMMITTEE   
(Whichever applies)

# ACKNOWLEGMENTS

It is with deep gratitude and appreciation that I acknowledge the professional guidance of Dr. Nguyen Van Sinh. His constant encouragement and support helped me to achieve my goal.

My gratitude goes to the other members of the laboratory, Mr. Pham Quoc Son Lam and Mr. Tran Thanh Tung. Their technical help and good humor made these years a great learning experience. I am grateful to the faculty of the School of Computer Science of the International University, especially Dr. Nguyen Van Sinh for his support since the beginning of my studies. Gratitude is also expressed to the members of my reading and examination committee.

*Note: Paper A4, Top: 2.5cm; Bottom: 2cm; Left: 3cm; Right: 2cm*

**Table of Contents**

[ACKNOWLEGMENTS 3](#_Toc124710731)

[LIST OF FIGURES 5](#_Toc124710732)

[LIST OF TABLES 6](#_Toc124710733)

[ABSTRACT 7](#_Toc124710734)

[CHAPTER 1: INTRODUCTION 8](#_Toc124710735)

[1.1. Background 8](#_Toc124710736)

[1.2. Problem Statement 8](#_Toc124710737)

[1.3. Scope and Objectives 9](#_Toc124710738)

[1.4. Assumption and Solution 10](#_Toc124710739)

[CHAPTER 2: LITURATURE REVIEW/RELATED WORK 12](#_Toc124710740)

[2.1. Classification of TrashNet Dataset Based on Deep Learning Models 12](#_Toc124710741)

[2.2. WasteNet: Waste Classification at the Edge for Smart Bins 12](#_Toc124710742)

[2.3. Exploring Features in a Bayesian Framework for Material Recognition 12](#_Toc124710743)

[2.4. SpotGarbage: Smartphone App to Detect Garbage Using Deep Learning 13](#_Toc124710744)

[CHAPTER 3: METHODOLOGY 14](#_Toc124710745)

[3.1 Overview 14](#_Toc124710746)

[3.2 User requirement analysis 17](#_Toc124710747)

[3.2.1 Data Augmentation 18](#_Toc124710748)

[3.2.2 Transfer Learning 19](#_Toc124710749)

[3.3 User Interface and Packaging 20](#_Toc124710750)

[CHAPTER 4: IMPLEMENT AND RESULTS 21](#_Toc124710751)

[4.1. Implementation 21](#_Toc124710752)

[4.1.1. Prerequisite 21](#_Toc124710753)

[4.1.2. Challenges 22](#_Toc124710754)

[4.2. Results 22](#_Toc124710755)

[CHAPTER 5: DISCUSSION AND EVALUATION 26](#_Toc124710756)

[5.1. Discussion 26](#_Toc124710757)

[5.2. Comparison 26](#_Toc124710758)

[5.3. Evaluation 27](#_Toc124710759)

[CHAPTER 6: CONCLUSION AND FUTURE WORK 28](#_Toc124710760)

[6.1. Conclusion 28](#_Toc124710761)

[6.2. Future work 28](#_Toc124710762)

[REFERENCES 29](#_Toc124710763)

[APPENDIX 33](#_Toc124710764)

# LIST OF FIGURES

1. MobileNetV2 architecture 15
2. Model workflow 16
3. Model class diagram 17
4. Model use case diagram 18
5. Sample image augmentation 19
6. Loss per epoch 23
7. Confusion matrix on test dataset of the model 24

# LIST OF TABLES

1. MobileNetV2 Bottleneck Residual Block 14
2. Model’s performance on test dataset 23
3. Comparison between the proposed model, GarbNet and WasteNet 27

# ABSTRACT

Environment-related problems have become increasingly hazardous towards human and pose a risk to the ecosystem in the 21st century, one of which is the ever-increasing volumes of waste (excessive waste) including municipal solid waste. While recycling and other basic methods to treat waste have been introduced to daily life, the conversion of waste back to its constituent parts (known as recycling) is still in its infancy due to several difficulties relating to the technology and funding.

In this prethesis, I propose a machine learning model using convolutional neural networks and image processing techniques to help classify waste through a captured photo. The work includes data gathering, preprocessing and augmentation techniques, and model training with CNN. The model achieves 96% accuracy while requiring minimal time to train by implementing transfer learning and fine-tuning on a pretrained MobileNetV2. It can be implemented together with additional components to form a system to assess the recyclability and set priorities for waste packs at waste processing plants.

# CHAPTER 1: INTRODUCTION

## Background

Ever since the first industrial revolution in England from roughly 3 centuries ago, environmental problems have become generally more and more severe, according to the warning of the scientists [[1]](#ref1). By definition, environmental issues are effects of human activity on the biophysical environment specifically, most of which are for malicious intents and purposes that cause harmful effects called environmental degradation, such as the depletion of resources, decrease of quality of air, water and soil, the destruction of ecosystems and natural habitat, the extinction of wildlife and pollution in a subtle way [[2]](#ref2). Nowadays, environmental degradation kind of is listed as one of the ten threats officially definitely cautioned by the “High-level Panel on Threats, Challenges and Change” of the United Nations (UN) in a subtle way [[3]](#ref3).

Besides some widely known difficulties such as global warming, deforestation and water pollution, there is a lesser-known environmental problem of the ever-increasing volumes of waste (excessive waste), which is mainly caused by poor waste management and over shopping as the global population and living standards rise. To minimize the consequences of this problem, I need to reduce the waste volume as well as increase our recycling capability.

In practice, our waste would be commonly divided into “dry” or “wet” category. Dry waste generally includes wood and related products, metal, plastic as well as glass [[4]](#ref4). Wet waste typically refers to organic waste, food, together with vegetable products [[4]](#ref4). Within this prethesis work’s scope, only the problem of classification of household dry waste would be concerned, which is produced by the local community.

## Problem Statement

The abundance of trash leads to several severe environmental consequences. For example, municipal solid waste contributes greatly to the loss of raw material [[5]](#ref5). Massive landfills are driving people from their homes, causing unpleasant odor and irreversible pollution. Apart from the immediately noticeable problems, municipal solid waste, especially plastic, is also the culprit for potential global issues such as the exportation of trash from rich countries to poorer nations [[6]](#ref6).

For municipal solid waste, the recyclables usually consist of glass, plastic, paper, carton paper (cardboard), metal and cloth (fabric). Cardboard and paper belong to two different categories because they have different recycling process due to their own characteristics. Distinct types of plastic also require specific procedures. Currently, the recycling efficiency is limited by both technology and funding, thus cannot keep up with the rapid increase of waste over time. Usually, the recycling process consists of three main steps: collecting/processing, manufacturing, and reselling [[7]](#ref7). The first step is the hardest of the three, with the challenge of sorting and treating each object so that I can convert it back to its smaller constituent parts. In developing countries such as Vietnam, the work is done mostly by hand. Household trash packs are loaded on a truck, which are then delivered to the city landfill. At the dump, the trash is first sorted by large filters and magnet, then the workers manually sort the pieces to collect the recyclable scrap to resell at cheap price. This poses an elevated risk to the health of the workers, as well as being a bottleneck to the process overall efficiency since the first stage of the recycling limits the two sequential steps. Compared to wealthier countries, local processing plants in poorer nations are often limited, because of the high initial investment as well as the technology difficulty, with most of the currently available plants only use processed trash as fuel to burn in order to produce energy [[8]](#ref8).

To improve the efficiency of the first step in the recycling process while overcoming the aforementioned difficulties, international organizations such as the United Nations have encouraged the act of waste sorting from its source (the most well-known one being “3 bins for different types of trash” [[9]](#ref9) and similar systems). While it may serve as a long-term solution, it seems to be not enough for the immediate drastic measure. Furthermore, the application of it seems vague, resource demanding and is possible only in big cities. Other than this work, there have been other small-scale models and campaigns to help leverage recycling knowledge, as well as recognizing and classifying objects including trash, but they all have some shortcomings such as costly training, low performance, smaller scales,... The situation calls for a systematic and more applicable method to solve the recycling challenge – which would be proposed later in this work – involves using technology advances to classify the “green” objects in the waste mass to be recycled.

## Scope and Objectives

Municipal solid waste recycling is the key to reducing excessive waste, which is a critical problem in several countries, especially developing nations and high population areas. In this prethesis, the proposed model is expected to be able to classify six categories of objects using available photos of trash, hence serve as an early stage in waste classification, with the objects being different in type, size and color only. While the scope of this work only includes the model and datasets, it has the potential to be scaled up/scaled down to suit other projects’ needs (add more categories, implement in a large-scale system, etc.)

The model utilizes multi-class classification using Convolutional Neural Network (CNN) to make it possible to recognize and classify our objects among the waste mass. CNN, also known as Shift Invariant or Space Invariant Artificial Neural Networks – SIANN [[10]](#ref10), has been used in practice for a long time. CNN is a class of artificial neural network that is commonly applied to analyze visual imagery. Most CNN models are based on the shared-weight architecture of the convolution kernels or filters that slide along the input features to provide translation-equivariant responses known as feature maps.

In this work, I implement a pretrained MobileNetV2 [[11]](#ref11) with Python programming language and utilizes PyTorch library to construct a CNN model. The model is then trained on a dataset extended from Mindy Yang’s work “Classification of Trash for Recyclability Status” [[12]](#ref12). Due to the small size of the original dataset, I need to implement data augmentation by using scaling, rotating, shearing, controlling brightness, channel shuffling, adding Gaussian blur and noise techniques to increase the diversity of the dataset. I also build a simple user interface based on PyQt5 to make the model more user-friendly. The expected outcome is a high accuracy model with the ability to classify six types of household solid waste.

## Assumption and Solution

This work only proposed a model for classifying trash only, while outside of the work’s scope, it can be improved and retrained to be used in waste processing and recycling plants, or help the workers sort the trash more efficiently. As stated, the proposed CNN model would utilize transfer learning with MobileNetV2 [[13]](#ref13) to be able to recognize and classify the objects from image inputs. To be more specific, this model classifies and recognizes six types of trash only (paper, plastic, cardboard, metal, glass, and fabric) using captured photos as the input. There are other possible scenarios that this model may help, such as assessing the recycle priority of different portions of trash through captured photos or recognizing objects for trash auto sorting system using a crane machine. The model is assumed to be applied in a processing plant, or a local trash collection area to help classify the objects and can be implemented together with other assisting technology (object detecting, mechanical robot arms…).

This work assumes that the objects to be classified are not broken and smashed into small pieces (they need to retain their shapes to some extends to be recognized successfully). For example, glass bottles are not smashed into small pieces, and papers are not shredded into tiny bits. Deformation and different lighting conditions are acceptable.

For the train dataset, through data augmentation, the images vary to reflect real life conditions. They are of different colors and shapes and under different lighting conditions. The objects are not cleaned to make their conditions similar to their counterparts in the trash bin.

# CHAPTER 2: LITURATURE REVIEW/RELATED WORK

## Classification of TrashNet Dataset Based on Deep Learning Models

“Classification of TrashNet Dataset Based on Deep Learning Models”, which is one of the most relevant works to this paper’s problem, is a publication by Rahmi Arda Aral et al at Gazi University [[14]](#ref14). The paper provides comparison on performance of different models (Densenet121, DenseNet169, InceptionResnetV2, MobileNet and Xception architectures with Adam and Adadelta as the optimizer) when it comes to tackling the challenge of recognizing recyclable garbage. The authors compare and give commentary on the efficiency of each model’s approach. In addition, the paper also provides detailed insights about each architecture when trained with TrashNet dataset. Although the dataset is not exceptionally large and diverse, it is compensated by the image augmentation preprocessing.

## WasteNet: Waste Classification at the Edge for Smart Bins

“WasteNet: Waste Classification at the Edge for Smart Bins” is a work by Gary White et al at Trinity College Dublin – Ireland [[15]](#ref15). WasteNet is a CNN waste classification that can be deployed on low power devices at the edge of the network. The authors suggest the idea of local automated waste classification, which produces fast decisions in smart bins without the need to access to the cloud.

The model utilizes machine learning and pattern recognition using a deep neural network called WasteNet, which is based on VGG-16 model. The model uses transfer learning and several other optimizations to leverage knowledge from a source task and is trained on the ImageNet dataset. The authors also apply data augmentation on the images from the dataset to improve the diversity and accuracy, as well as to prevent overfitting. The most noticeable improvement in comparison to previous works on this topic is the proposed “hybrid tuning”, which combines the benefits of freezing technique during feature extraction and fine-tuning. With the high accuracy of 97% on test dataset [[15]](#ref15), the authors are able to prove the potential of the implemented techniques.

## Exploring Features in a Bayesian Framework for Material Recognition

“Exploring Features in a Bayesian Framework for Material Recognition” [[16]](#ref16) is the result from an academic collaboration by Ce Liu, Lavanya Sharan, Edward H. Adelson and Ruth Rosenholtz. The aim of the paper is to identify different material categories such as glass, metal, fabric, plastic, or wood from a single image of a surface using computers. The paper focuses on methods for extracting specific material’s features, while proposing a variation of LDA algorithm (augmented LDA - aLDA) [[16]](#ref16). aLDA combines the features of material appearance under a Bayesian generative framework and learn an optimal combination of features to help with the material classification task. Experimental results show that this model performs material recognition function reasonably well on a challenging material dataset.

## SpotGarbage: Smartphone App to Detect Garbage Using Deep Learning

“SpotGarbage: Smartphone App to Detect Garbage Using Deep Learning” is a project by Gaurav Mittal et al at Indian Institute of Technology Ropar Rupnagar [[17]](#ref17). With the aim of engaging citizens to track and report on their neighborhoods, the team presents a convenient smartphone app called SpotGarbage, which can recognize garbage regions in a user-clicked geo-tagged image using a deep architecture of fully convolutional networks for detecting garbage in images called GarbNet. The model has been trained on a newly introduced GINI dataset [[17]](#ref17) and achieves a mean accuracy of 87.69%. The paper also proposes optimizations in the network architecture, which results in a reduction of 87.9% memory usage and 96.8% prediction time with no loss in accuracy, facilitating its usage in resource constrained smartphones. With the advantage of being light weight and easy to use, the work is a good innovation for combating the excessive production of waste.

# CHAPTER 3: METHODOLOGY

## Overview

In this prethesis work, I implement transfer learning and fine-tuning techniques on a pretrained MobileNetV2 – a CNN by Google to classify objects. MobileNetV2 is preferred over other models because it is optimized for fast training time, low computing cost and maintenance. For the dataset, I extend the self-collected dataset of Mindy Yang and Gary Thung from the report “Classification of Trash for Recyclability Status” [[12]](#ref12) by adding one class of object and implement common data augmentation techniques. In total, we have 6 classes, each class has around 900 images.

MobileNetV2 architecture consists of 2 types of blocks: stride=1 and stride=2. For each block, there is a convolutional layer of 1x1 with ReLU6, then a depth wise convolution of 3x3 and a convolution of 1x1 without non-linearity. In total, the base model has fifty-three convolution layers and 1 average pooling layer. A widely known feature of MobileNetV2 is the proposal of Inverted Residual block, also called MBConv block for efficiency reasons. By implementing Inverted Residual block, it uses an inverted structure, which is described by the number of I/O channels following a “narrow > wide > narrow” approach [[18]](#ref18).

|  |  |  |
| --- | --- | --- |
| Input | Operator | Output |
| h × w × k | 1x1 conv2d , ReLU6 | h × w × (tk) |
| h × w × tk | 3x3 dwise s=s, ReLU6 | h/s × w/s × (tk) |
| h/s × w/s × tk | linear 1x1 conv2d | h/s × w/s × k’ |

Table 1. MobileNetV2 bottleneck residual block size, with stride s and expansion factor t.

Diagram

Description automatically generated

Figure 1. MobileNetV2 Architecture

In total, MobileNetV2 consists of fifty-three layers, with the architecture described in figure 1. MobileNetV2 is originally trained on ImageNet-1k dataset to recognize one thousand types of objects (one thousand classes) [[13]](#ref13). For my model, I want to utilize the already-learned general features such as edges and shapes of the object from the pretrained weight, while customizing the last layer to adapt the model to my prethesis’s purpose (classification of six types of dry waste). Transfer learning is implemented by performing additional training on top of the already trained weight. I also freeze the first twelve layers to reduce the training time while maintaining low loss with high accuracy.

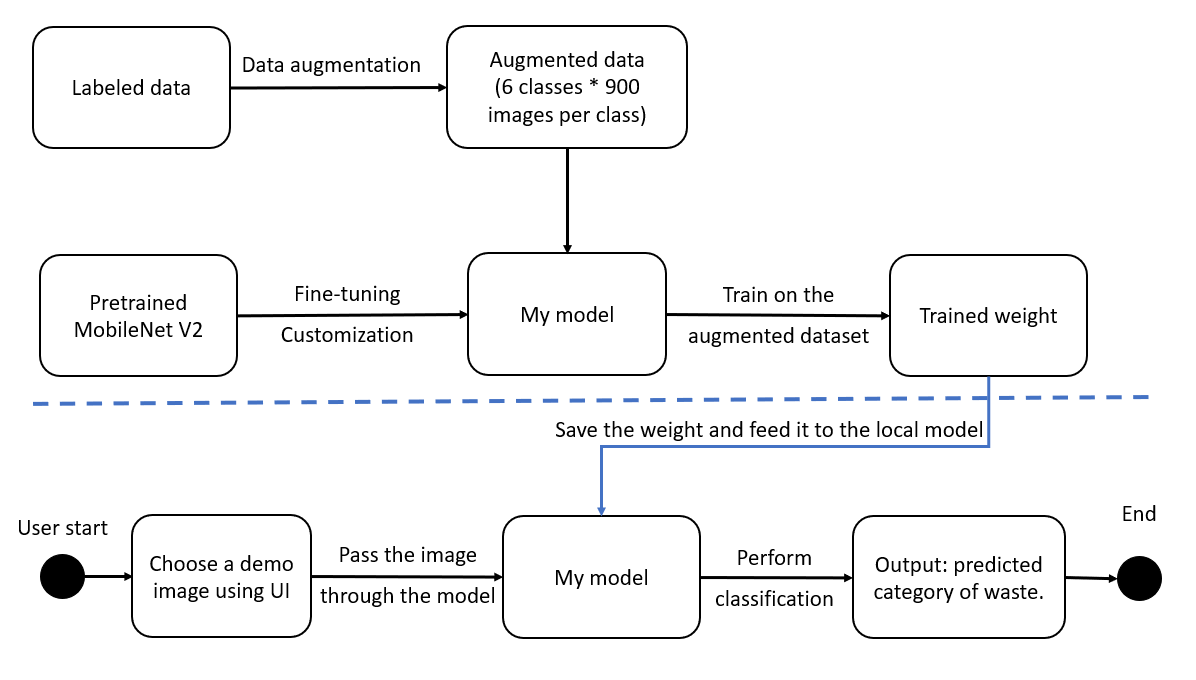


Figure 2: Model workflow

The workflow starts with collecting the data and labeling them. Then I process the data (augmentation, data cleaning…) and feed the preprocessed data to the model in batches. The model will try to extract the features of each class by converting the images into tensors and passing them through different layers to filter the features of each class. After training, the weight of the model is saved to local for later use.

On a local computer, a user can use the user interface to navigate to the demo image. The UI will pass the image address to the local model. The model will load the image, convert to tensor, and compare the demo image’s features with the features of each category. Then, by calling log\_softmax, I get an array with each element describing how suitable is the image for that class. Because log\_softmax output range is [-inf, 0], the model will find the max of the output array and return the corresponding category name. If there are multiple types of waste in the photo, the model will return the class name of the highest proportion.

## User requirement analysis

The work consists of four classes: Main, Test, Preprocessing and Train. Class Main contains code for UI, which includes PyQt objects and event listeners such as onClick [[19]](#ref19). Class Test defines methods for image classification model, which consists of model initialization, weight loading and class predicting. Class Train is used for model training purposes, which includes methods for model architecture loading, dataset loading and model training. Class Preprocessing contains methods for data agumentation using Imgaug [[25]](#ref25).

Diagram

Description automatically generatedFigure 3. Model class diagram

This work has two use cases which are classifying a piece of trash and retraining the model. The actor is the user interacting with the program. For classification of waste, there are underlying helper methods to initialize the model, load the weight and perform classification. If the model performs not as expected, the user can retrain the model.

Diagram

Description automatically generatedFigure 4. Model use case diagram

### Data Augmentation

Since the dataset is quite small, data augmentation is applied for all classes. The images are transformed using “imgaug” Python library [[20]](#ref20). It is a library dedicated to image augmentation for machine learning experiments, especially in classification tasks. There are many options to choose from, and it is relatively simple to install and implement. The operations I chose to include scaling, shearing, horizontal-vertical flipping, color channel shuffling, blur addition, low noise addition and slight rotation. The p value for transform probability ranges from 0.3 to 0.6 randomly. Combined with OpenCV [[21]](#ref21) for reading/writing images, in the end, I have a total of around nine hundred samples for each class.

Imgaug accepts both 2D and 3D input array which can be converted from the input image. For flipping operation, imgaug supports both flipping the input array vertically and horizontally and produces a flipped ndarray as the output [[22]](#ref22). For Gaussian blur, imgaug library will slide a Gausian kernel over the image to compute the transformation needed for each pixel [[23]](#ref23). I set the standard deviation of the Gaussian kernel to 0.6, which is denoted as sigma value by the library.

Both shearing, scaling, and rotating belong to the affine transformation [[24]](#ref24). To apply affine transformation, Imgaug will perform multiplication between the input array and the transformation matrix. There are other additional augmentation techniques such as channel shuffling [[25]](#ref25), which randomize the order of channels of the image.

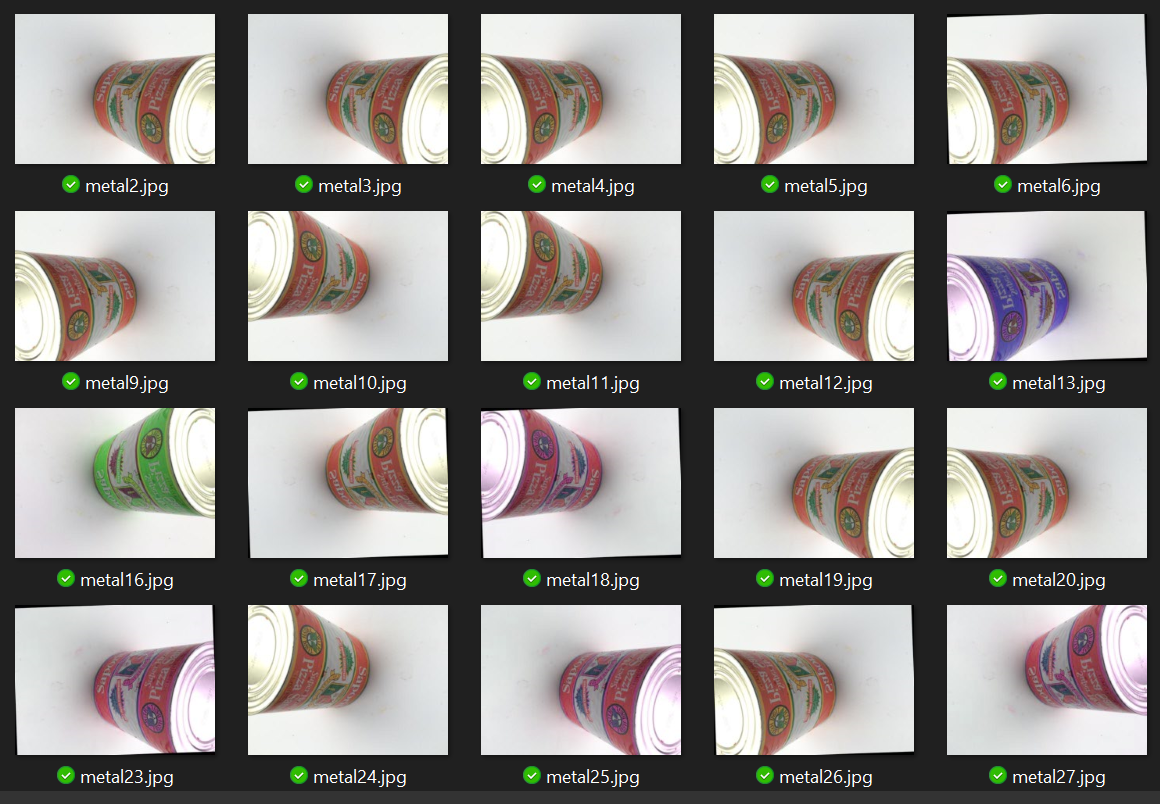


Figure 5: Sample image augmentation

### Transfer Learning

When a neural network is trained on a data, it gains knowledge of the classes, which is compiled as the weights of the network. Transfer learning is the process of extracting and transferring the weights to another neural network. By using a pre-trained neural network to solve a machine learning task that is similar to the problem the network was originally trained to solve, the time and data required for training would decrease significantly. MobileNetV2 supports transfer learning, where it is previously trained to recognize objects on a large dataset (ImageNet-1k), and then apply the knowledge to similar problems (trash classification) [[26]](#ref26).

Because the output of the original MobileNetV2 is one thousand classes of object, it does not suit the goal of my work. To make the model compatible, I change the classifier to the number of classes (which is six) and add a dropout layer to reduce overfitting. I also freeze the first twelve layers to improve the model efficiency, because the low and middle layers only pick simple features like edge, curve [[27]](#ref27). This significantly reduces the training time while maintaining high accuracy and low loss.

### User Interface and Packaging

While the model is developed, I am required to have an interface for user to interact with the model. I use PyQt5 – a Python GUI library to build the user interface. Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems, while PyQt5 is a comprehensive set of Python bindings for Qt v5. PyQt5 supports rapid development and deployment of simple custom graphical user interface [[28]](#ref28).

The UI has basic functions such as displaying the output of the model, buttons to change photos and image preview panel. I use event slots and built-in components to create the UI and import the function to run the model from helper scripts. For easier usage, I make an executable file and package the scripts into a single zip file using PyInstaller [[29]](#ref29). This would be helpful for users without existing compatible Python/PyQt version or cannot run Python.

# CHAPTER 4: IMPLEMENT AND RESULTS

## Implementation

The purpose of the training process is to get the weight for later use in the demo program. To build the model, I use Pytorch [[36]](#ref36) with Python. The model is trained on Google Colab – a free scientific processing platform for machine learning tasks using its default free plan, since training process can take an exceptionally long time, especially without a dedicated GPU. After a lot of testing, thirty-six epochs provide a good balance between performance and time cost.

First, a separated script is used to apply image augmentation on the dataset using Imgaug and OpenCV, then combine with the original dataset and upload them to Google Drive. I then run the preparation process on Google Colab, which includes fetching the pretrained MobileNetV2, defining helper functions, loading and processing the data. We change the classifier function to 6 output channels, with a linear function and a dropout layer to reduce overfitting. To load the dataset, I use PyTorch set the train : val : test ratio to 0.7 : 0.15 : 0.15. To make the process efficient, I utilize transfer learning and freeze the first 12 layers of the pretrained model. After that, the model would be ready to train with around 5400 samples of the dataset, batch size of 32, SGD optimizer with momentum of 0.9 and learning rate of 0.0002 (2e-4). In the end, the model is trained for roughly 1 hours and 30 minutes. After training, the new weight is saved for later use.

For the implementation of user interface, I use PyQt5 [[19]](#ref19) and pack the program using PyInstaller [[28]](#ref28)[[29]](#ref29). The UI has basic functions to display the image and show the trash classification result.

### Prerequisite

For package management, Anaconda [[39]](#ref39) is used to manage separated Python environments and packages. The augmented data is processed to remove outliers, then uploaded to Google Drive. Anaconda distribution of Python can be downloaded from the official website <https://www.anaconda.com/>. I then create an isolated environment and install the libraries inside the mentioned environment for easier management. With Anaconda, managing different environments, especially for scientific tasks is easy and efficient.

Other than Python, a few libraries are needed for the code to run. For plotting, I use Seaborn [[30]](#ref30) and Matplotlib [[31]](#ref31). Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Tqdm [[32]](#ref32) is used to illustrate the progress of the training. I also use the metrics from Sklearn [[33]](#ref33), as well as functions from Numpy [[34]](#ref34), Panda [[35]](#ref35) and Pytorch [[36]](#ref36).

I use Google Colab for our training platform. In Google Colab, the drive is mounted, and the dataset directory is linked to the code to allow reading the images. No decent hardware is required locally. Colab provides different user plans with multiple perks and prices. For this model, I use the free plan with no upgrade, and 15GB free Google Drive storage by default.

For packing the program, I use PyInstaller [[28]](#ref28)[[29]](#ref29) to compress all the needed code into a single executable file. PyInstaller analyzes the code to discover every imported module and library recursively for the script to execute. Then it collects the copies of all those files and pack it in a distribution. There are different export options such as outputing a folder or a self-extracting executable file.

### Challenges

Different types of the same material can have different characteristics and recyclability ratings. For example, there are seven types of plastic [[37]](#ref37) alone and only half of them are recyclable. Both nylon and film are implementations of plastic and training the model to correctly recognize them as plastic can be challenging. Besides, many different classes of trash can have similar material characteristics such as opacity and reflectivity. Another challenge is that due to the size of the dataset is small, even with data augmentation techniques, it is still limited and cannot fully reflect real life scenarios.

## Results

To determine if a model is good or not, the model is often rated by the following criteria: accuracy, loss, precision, recall and F1 score [[38]](#ref38). Accuracy and loss from the training and validating process are two popular numbers to measure a model’s performance. The rate the loss decreases can indicate if the learning rate is suitable or not, while the fluctuations between epochs can show if the model is able to learn from the dataset during the training process. Precision finds out what fraction of predicted positives is positive, while recall criteria is calculated by the ratio of true positives over all positives. It can be understood that recall measures the model’s ability to predict the positives. For F1 score, it is the harmonic mean of precision and recall.

Chart, line chart

Description automatically generated

Figure 6. Loss per epoch graph

The model achieves over 60% accuracy on the first try. After training for thirty-six epochs, the model’s accuracy increases to 96.4%. The loss goes down from 1.4 while training and fluctuates within a small margin during its decrease, then starts to stabilize from epoch #27. I stop training after 36 epochs because the loss already meets the requirements, and additional training would only increase overfitting.

|  |  |
| --- | --- |
| Trash Classification Model | |
| Accuracy | 96.42% |
| Precision | 96.43% |
| Recall | 0.964 |
| F1 score | 0.964 |

Table 2. Model’s performance on test dataset

Both accuracy and precision are over 96%, with recall criteria and F1 score being ~0.96. While the model performs very well given the criteria, the dataset is in fact relatively small, so further testing and optimizations are still possible.

Graphical user interface, application

Description automatically generated

Figure 7. Confusion matrix on test dataset of the model

Figure 7 shows the confusion matrix of the model on test dataset. I have six classes of objects on the vertical left, and what they are falsely classified as on the horizontal bottom. Among the categories, glass is most often wrongly recognized as metal and plastic. This can be explained as they share several common characteristics, including reflectivity, opacity and smoothness. For other classes, there are occasional misclassifies, though not as frequent as glass. There is also a slight imbalance between the number of test sample from each class.

# CHAPTER 5: DISCUSSION AND EVALUATION

## Discussion

MobileNetV2 is chosen as a base for extension because it is robust while being light weight. The implementation of freezing layers helps reduce the training time by a large margin while maintaining high accuracy. Despite the optimization, there are still some limitations to the current model, such as distinguishing “green” plastic from its non-recyclable counterparts, or periodic misclassification between glass and plastic. The model also requires the sample to be captured separately in order to correctly assign the category.

## Comparison

Among the models mentioned in chapter 2, MobileNetV2 stands out as a promising model. It is able to perform better than other models from section 2.1, with the accuracy competing for the top position. Other than that, the most similar models to ours are GarbNet [[17]](#ref17), proposed by Gaurav Mittal et al and WasteNet [[15]](#ref15) by Gary White et al.

Both WasteNet and my proposed model have a few mutual features. The work model’s performance is on par with WasteNet model at 96% accuracy on test dataset, and both utilize transfer learning to fine-tune the original pretrained model. Other than that, I implement more data augmentation techniques and have a different dataset split, as well as using different pretrained models while WasteNet work introduced a new augmentation method called image expansion and a different set of categories. Regarding the finetuning, the authors of WasteNet decided to unfreeze the layers gradually, while I implement fine-tuning techniques to the first fifteen layers at once. Generally, both models perform competently despite different choices during development and implementation.

Compared to GarbNet, my model is trained on different datasets with a different split ratio. The other main differences include the use of different base models and optimization techniques. GarbNet was able to achieve 87.69% of accuracy, while being noticeably lighter than our model (23MB application compared to ~600MB zip file). This can be explained as they focus more on limiting the computational resource requirements than gaining accuracy with the aim to make the model runnable on IoT devices, as well as the development of a dedicated native Android application, while I use Python libraries to create and package a Windows user interface instead of well-known optimized language like C++.

|  |  |  |  |
| --- | --- | --- | --- |
|  | My model | GarbNet | WasteNet |
| Dataset | TrashNet + custom Dataset | GINI Dataset | TrashNet |
| Pretrained model | MobileNetV2 | None | VGG-16 |
| Training time / Epochs | 1.5 hours/36 epochs | - | 1000 epochs |
| Accuracy | 96.42% | 87.7% | 97% |
| Precision | 96.43% | - | 97% |
| Recall | 96.42% | - | 97% |
| F1 score | 96.41% | - | 97% |

Table 3. Comparison between the proposed model, GarbNet and WasteNet

## Evaluation

The model sees practical use for single object classification task with high accuracy, while being flexible for retraining. Even though there are still shortcomings to overcome, the model can help with sorting waste into different categories. The model performs best when the input photo contains an object on a white background, though during my testing, any object with simple backgrounds would work.

# CHAPTER 6: CONCLUSION AND FUTURE WORK

## Conclusion

Excessive waste production is a major problem in many countries, especially with the advance of technology and the improvement of living standards. To tackle the problem, there have been efforts to increase recycling as well as educating young generations about the risk of trash. In this work, I have demonstrated an implementation of MobileNetV2 for trash classifying purpose with the hope to help make the recycling process more efficient. I use a modified TrashNet dataset with augmentation techniques from Python library Imgaug, which helps increase the diversity of the dataset. By applying transfer learning and fine-tuning the original pretrained model by freezing the first fifteen layers, the training process becomes more efficient. The model aims to be light-weight and retain the ability to be implemented on edge devices where computation power is often limited. With high accuracy and the flexibility in the model’s architecture, the model can be further adapted and retrained to suit the local needs. Despite the positive results, there are still difficulties to overcome.

## Future work

For future work, I intend to combine my model with another model for object recognition to make the system more versatile. The model would be extended to support multiple classifications per image, as well as being lighter and more efficient. To be implemented in a waste processing plant, the model would need to be able to approximate how recyclable is each portion of trash, which requires the model to recognize and pickup parts of objects from the image and try to classify the materials to measure the recyclability percentage. This aims to make the recycling process easier and more efficient.

# REFERENCES

[1] Erin Hale, “Earth's environment getting worse, not better”, <https://www.theguardian.com/environment/2012/may/15/earth-environment-wwf-rio20>, Assessed May 2012

[2] United Nations, “High-level Panel on Threats, Challenges and Change”, 2003 Conference.

[3] Giving Compass, “Environmental Issues Guide: What to Know Now”, https://givingcompass.org/article/environmental-issues

[4] Protech Group, “Wet or Dry? How to segregate and manage waste” <https://protechgroup.in/wp/wet-or-dry-how-to-segregate-and-manage-waste/>

[5] Evreka, “What is Municipal Solid Waste (MSW)?”, https://evreka.co/blog/what-is-the-meaning-of-municipal-solid-waste/

[6] Brendan Lui, “US Waste Exporting Explained”, <https://repurpose.global/blog/post/us-waste-exporting-explained>, Assessed April 2019

[7] Sara Cifani, “What Happens to Trash After It’s Tossed?”, <https://www.dumpsters.com/blog/what-happens-to-garbage>, Assessed October 2022

[8] Deltaway Energy, “Biomass explained: Waste-to-energy (Municipal Solid Waste)” from <https://www.eia.gov/energyexplained/biomass/waste-to-energy-in-depth.php>, Assessed December 2022

[9] Australian Government, “What to put in your 3 bins”, https://www.unley.sa.gov.au/Bins-pets-parking/Waste-recycling/What-to-put-in-your-3-bins

[10] Zhang, Wei, "Shift-invariant pattern recognition neural network and its optical architecture", presented at Annual Conference of the Japan Society of Applied Physics, 1988

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, presented at 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI 10.48550

[12] Mindy Yang, Gary Thung. "Classification of trash for recyclability status." CS229 report, Assessed 2016

[13] Sebastian Mathot, “Customizing MobileNetV2 through transfer learning”, from <https://pythontutorials.eu/deep-learning/transfer-learning/>.

[14] R. A. Aral, Ş. R. Keskin, M. Kaya, M. Haciömeroğlu, "Classification of TrashNet Dataset Based on Deep Learning Models", presented at 2018 IEEE International Conference on Big Data (Big Data), ISBN:978-1-5386-5035-6, DOI 10.1109, Dec 2018.

[15] Gary White, Christian Cabrera, Andrei Palade, Fan Li, Siobhan Clarke, “WasteNet: Waste Classification at the Edge for Smart Bins”, DOI 10.48550, Jun 2020.

[16] Ce Liu, Lavanya Sharan, Edward H. Adelson, Ruth Rosenholtz, “Exploring Features in a Bayesian Framework for Material Recognition”, presented at 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, ISBN:978-1-4244-6985-7, DOI 10.1109, Jun 2010.

[17] G. Mittal, K. B. Yagnik, M. Garg, N. C. Krishnan, "SpotGarbage," presented at the Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, DOI 10.1145 , Sep 2016.

[18] Paul-Louis Pröve, “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, <https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5>, Assessed April 2018

[19] Tech with Tim, “PyQt5 Buttons and Events”, <https://www.techwithtim.net/tutorials/pyqt5-tutorial/buttons-and-events/>

[20] Imgaug, official documentation from <https://imgaug.readthedocs.io/en/latest/>

[21] OpenCV, official documentation from <https://docs.opencv.org/4.x/db/deb/tutorial_display_image.html>

[22] Imgaug, official documentation from <https://imgaug.readthedocs.io/en/latest/source/api_augmenters_flip.html>

[23] Imgaug, official documentation from [https://imgaug.readthedocs.io/en/latest/source/api\_augmenters\_blur.html](https://imgaug.readthedocs.io/en/latest/source/api_augmenters_blur.html#imgaug.augmenters.blur.GaussianBlur)

[24] Imgaug, official documentation from [https://imgaug.readthedocs.io/en/latest/source/api\_augmenters\_geometric.html?highlight=affine](https://imgaug.readthedocs.io/en/latest/source/api_augmenters_geometric.html?highlight=affine#imgaug.augmenters.geometric.Affine)

[25] Imgaug, official documentation from <https://imgaug.readthedocs.io/en/latest/source/overview/meta.html?highlight=channel#channelshuffle>

[26] Tensorflow, official documentation from <https://www.tensorflow.org/tutorials/images/transfer_learning>

[27] Pytorch, official documentation from <https://pytorch.org/docs/master/notes/autograd.html>

[28] Martin Fitzpatrick, “Packaging PyQt5 applications for Windows with PyInstaller & InstallForge” from <https://www.pythonguis.com/tutorials/packaging-pyqt5-pyside2-applications-windows-pyinstaller/>. Assessed August 10, 2022.

[29] Luke Lee, “Using PyInstaller to Easily Distribute Python Applications”, from <https://realpython.com/pyinstaller-python/>.

[30] Seaborn, official documentation from <https://seaborn.pydata.org/tutorial.html>

[31] Matplotlib, official documentation from <https://matplotlib.org/stable/tutorials/index>

[32] Tqdm, official documentation from <https://tqdm.github.io/>

[33] Pedregosa et al, “Scikit-learn: Machine Learning in Python”, JMLR 12, pp. 2825-2830, 2011.

[34] Numpy, official documentation from <https://numpy.org/doc/stable/reference/index.html>

[35] Pandas, official documentation from <https://pandas.pydata.org/docs/>

[36] PyTorch, official documentation from <https://pytorch.org/docs/stable/index.html>

[37] Tod Hardin, “Plastic: It’s Not All the Same“, from <https://plasticoceans.org/7-types-of-plastic/>. Assessed February 2021.

[38] Harikrishnan N B, “Confusion Matrix, Accuracy, Precision, Recall, F1 Score”, from <https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd>. Assessed December 2019.

[39] Anaconda, official documentation from <https://docs.anaconda.com/anaconda>.

# APPENDIX

1. Appendix 1. Source code

Source code: <https://github.com/TomPham204/prethesis>

Workspace folder: <https://1drv.ms/u/s!ArPVZgo5x8MIg4ZILypkoooYYs6OUQ?e=dIdl8W>

1. Appendix 2. MobileNetV2 Architecture

|  |  |  |
| --- | --- | --- |
| Type/Stride | Filter Shape | Input Size |
| Conv/s2 | 3x3x3x32 | 224x224x3 |
| Conv dw/s1 | 3x3x32 dw | 112x112x32 |
| Conv/s1 | 1x1x32x64 | 112x112x32 |
| Conv dw/s2 | 3x3x64 dw | 112x112x64 |
| Conv/s1 | 1x1x64x128 | 56x56x64 |
| Conv dw/s1 | 3x3x128 dw | 56x56x128 |
| Conv/s1 | 1x1x128x128 | 56x56x128 |
| Conv dw/s2 | 3x3x128 dw | 56x56x128 |
| Conv/s1 | 1x1x128x256 | 56x56x128 |
| Conv dw/s1 | 3x3x256 dw | 56x56x256 |
| Conv/s1 | 1x1x256x256 | 56x56x256 |
| Conv dw/s2 | 3x3x256 dw | 56x56x256 |
| Conv/s1 | 1x1x256x512 | 14x14x256 |
| Conv dw/s1 | 3x3x256 dw | 14x14x512 |
| Conv/s1 | 1x1x512x512 | 14x14x512 |
| Conv dw/s1 | 3x3x256 dw | 14x14x512 |
| Conv/s1 | 1x1x512x512 | 14x14x512 |
| Conv dw/s1 | 3x3x256 dw | 14x14x512 |
| Conv/s1 | 1x1x512x512 | 14x14x512 |
| Conv dw/s1 | 3x3x256 dw | 14x14x512 |
| Conv/s1 | 1x1x512x512 | 14x14x512 |
| Conv dw/s1 | 3x3x256 dw | 14x14x512 |
| Conv/s1 | 1x1x512x512 | 14x14x512 |
| Conv dw/s2 | 3x3x512 dw | 14x14x512 |
| Conv/s1 | 1x1x512x1024 | 7x7x512 |
| Conv dw/s2 | 3x3x1024 dw | 7x7x1024 |
| Conv/s1 | 1x1x1024x1024 | 7x7x1024 |
| Avg pool/s1 | Pool 7x7 | 7x7x1024 |
| FC/s1 | 1024x1000 | 1x1x1024 |
| Softmax/s1 | Classifier | 1x1x1000 |

Appendix 2. Overview of MobileNetV2 architecture.