

An Automatic Garbage Classification System Using RESNET-34

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and
Engineering

by

ADITYA SIRANGI (Roll. No. 17SCS0008)

GUNTUKU VIVEK (Roll. No. 17SCS8128)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI

SALAI, CHENNAI – 600119

APRIL - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **ADITYA SIRANGI (Roll. No. 17SCS0008)** and **GUNTUKU VIVEK (Roll. No. 17SCS8128)** who carried out the project entitled "**AN AUTOMATIC GARBAGE CLASSIFICATION SYSTEM USING RESNET-34**" under my supervision from **November 2020** to **April 2021**.

Internal Guide

Dr. L. Mary Gladence, M.E., Ph.D.,

Head of the Department

Dr. S. Vigneshwari, M.E., Ph.D.,

Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

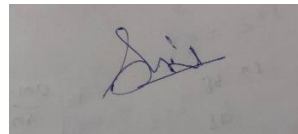
External Examiner

DECLARATION

I **ADITYA SIRANGI** hereby declare that the Project Report entitled “**AN AUTOMATIC GARBAGE CLASSIFICATION SYSTEM USING RESNET-34**” is done by me and my partner **G VIVEK** under the guidance of **Dr. L. MARY GLADENCE, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'Sirangi'.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L. Mary Gladence ME, Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Trash arrangement has consistently been a significant issue in ecological security, As the economy is developing rapidly and with the improvement of people's living standards, the amount of garbage is also increasing rapidly. To tackle the provincial living trash handling issues, we freely built up a one of a kind innovation of perfect and far reaching use of trash assets. Based on modern biodegradation and automatic sorting, we built a complete industrial chain with mixed domestic waste as mineral raw material. The natural discharge would be siphoned to anaerobic maturation framework, while the buildup would be another puncturing through broken drum to fundamental gathering framework. By our advanced gadgetry, we secure inorganic material as sand and glass, mash and sundries, for example, bamboo materials, and unadulterated plastic finally. The items would be extra preparing to mechanical materials as expected of market. The sand and rock glass and other inorganic materials can deliver empty squares, clearing block and wipe block. At last, the prevalence of the proposed grouping calculation is checked with the built trash information. The order exactness of the proposed calculation is upgraded by 1.01%. The trial results show that the grouping precision is pretty much as high as 99%, the characterization pattern of the framework is pretty much as brisk as 0.95 s.

TABLE OF CONTENTS

Contents	Page.No
1.INTRODUCTION	4
1.1 What is Deep Learning?	4
1.1.1 How Deep Learning works?	4
1.1.2 Deep Learning vs Machine Learning	4
1.1.3 Applications of Deep Learning	5
1.1.4 Advantages of Deep Learning	6
1.1.5 Disadvantages of Deep Learning	7
1.2 Motivation	7
1.3 Existing System	8
1.3.1 Limitations of existing system	8
1.4 Objectives	8
1.5 Applications	8
2.LITERATURE SURVEY	9
2.1 Introduction	9
2.2 Algorithm Design	11
3.PROBLEM ANALYSIS	15
3.1 Existing Approach	15
3.1.1 Drawbacks	15
3.2 Proposed System	15
3.2.1 Advantages	15
3.3 Software and Hardware requirements	15

3.4 About Dataset	17
3.6 Modules	18
4. DESIGN	19
4.1 Convolutional Neural Network(CNN)	19
4.2 Numpy	23
4.3 Matplotlib	24
4.4 Keras	24
4.5 Residual Neural Network(ResNET)	25
5. IMPLEMENTATION	29
5.1 Flowchart	29
5.2 Code	29
6. TESTING AND VALIDATION	36
6.1 Software Testing	36
6.1.1 Types of Testing	36
6.2 Execution Environment	38
6.2.1 Steps to start Google Colab	39
6.2.2 Pros of Google Colab	41
6.2.3 Cons of Google Colab	42
7. RESULT ANALYSIS	43
8. CONCLUSION	52
9. REFERENCES	53

LIST OF FIGURES

Fig.No	Figure Name	PageNo
4.1	A CNN Sequence	19
4.2	Convoluting a 5x5x5 image with a 3x3x3 Kernel	20
4.3	Convolution operation on a MxNx3 image Matrix with 3x3x3 kernel	21
4.4	3x3 pooling over 5x5 convoluted feature	21
4.5	Types of Pooling	22
4.6	Architecture of ResNET	27
6.1	Opening Google Colab	39
6.2	Getting started with Google Colab	40
6.3	Setting up the GPU	41
7.1	Importing Packages	43
7.2	Splitting the data into Train and Test	44
7.3	Printing Labels	45
7.4	Building a Neural Network Model	46
7.5	Measuring the Accuracy	47
7.6	Output of the Neural Network Model	48
7.7	ResNET-50 Model Building	49
7.8	ResNET-50 Prediction	49
7.9	ResNET-34 Model Building	50
7.10	ResNET-34 Prediction	50
7.11	Plotting loss and accuracy curves	51

CHAPTER 1

1.INTRODUCTIO

N

1.1 What Is Deep Learning?

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

1.1.1 How Deep Learning Works

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

1.1.2 Deep Learning vs. Machine Learning

One of the most common AI techniques used for processing big data is machine

learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data.

If a digital payments company wanted to detect the occurrence or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set, and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

1.1.3 Applications of Deep Learning

- **Virtual Assistants:** The core functionality that requires translating the speech and language of the human's speech, is deep learning. The common examples of virtual assistants are Cortana, Siri, and Alexa.
- **Vision for Driverless, Autonomous Cars:** In order to navigate an autonomous car, say, a Tesla, one needs a human-like experience and expertise. To understand the scenarios of roads, the working of signals, pedestrians, significances of different signs, speed limits and many more situations like these, a large amount of real data is required. With the large data, the efficiency of the algorithms will be improved which will subsequently increase the decision-making flow.
- **Service and Chat Bots:** The continuous interaction of chatbots with human beings for providing customer services requires strong responses.

To respond in a helpful manner with all the tricky questions and apt response, deep learning is required for training algorithms.

- **Translations:** Translating the speech automatically in multiple languages requires deep learning supervision. This is a helpful mechanism for tourists, travelers, and government officials.
- **Facial Recognition:** Facial recognition has many features from being used in the security to the tagging mechanism/feature used on Facebook. Along with the importance, it has its fair share of issues as well. For example, to recognize the same person with weight gain, weight loss, beard, without a beard, new hairstyles, etc.
- **Shopping and Entertainment:** All the shopping applications like Amazon and Myntra and entertainment applications like Amazon Prime and Netflix store your data and buying habits to show the suggestions for future buying and watching. It always comes as a title “You may like to watch/buy”. The more data is inputted in the Deep learning algorithm, the more efficient it becomes in decision making.
- **Pharmaceuticals:** Customizing medicines based on the particular genome and diseases. Deep learning has widened the scope of such applications and has gained the attention of the largest pharmaceutical companies.

1.1.4 Advantages of Deep Learning

- Ability to generate new features from the limited available training datasets.

- Its ability to work on unsupervised learning techniques helps in generating actionable and reliable task outcomes.
- It reduces the time required for feature engineering, one of the tasks that requires major time in practicing machine learning.
- With continuous training, its architecture has become adaptive to change and is able to work on diverse problems.

1.1.5 Disadvantages of Deep Learning

With the increasing popularity, deep learning also has a handful of threats that needs to be addressed:

- The complete training process relies on the continuous flow of the data, which decreases the scope for improvement in the training process.
- The cost of computational training significantly increases with an increase in the number of datasets.
- Lack of transparency in fault revision. No intermediate steps to provide the arguments for a certain fault. In order to resolve the issue, a complete algorithm gets revised.
- Need for expensive resources, high-speed processing units and powerful GPU's for training to the data sets.

1.2 MOTIVATION

With the rapid development of economy and the improvement of people's living standards, the amount of garbage is increasing rapidly. According to the latest report of International Lianhe Zaobao, the global garbage volume will increase by 70% by 2050, and the task of garbage classification will be even more arduous.

Scholars at home and abroad have done a lot of researches on garbage classification, but most of the proposed schemes are innovations of terminal recycling method. In 2019, China started to require residential garbage classification, in which case the front-end collection is highly dependent on people's awareness.

1.3 Existing System

For the commercialization of intelligent garbage bin, there are some attempts made by foreign companies, such as NASTAR in the United States, Zeton, and the Japanese somatosensory intelligent garbage bin. There are a lot of researches on the intelligence garbage bin in academia.

T. Kano et al. proposed a group-control robot system, which provides a decentralized control scheme for indoor garbage collection. But it depends on the cooperation between robots and takes a long time. P. Wesley et al. proposed a low-cost intelligence garbage bin.

1.3.1 Limitations of existing system

Data acquisition, highly error-prone and time consuming.

1.4 Objectives

The main objective is improve the efficiency of front-end garbage collection, an automatic garbage classification system is proposed based on deep learning.

1.5 Applications

This application can use in automatic garbage collection classification.

2.

LITERATURE SURVEY

2.1 INTRODUCTION

With the rapid development of economy and the improvement of people's living standards, the amount of garbage is increasing rapidly. According to the latest report of International Lianhe Zaobao, the global garbage volume will increase by 70% by 2050, and the task of garbage classification will be even more arduous. Scholars at home and abroad have done a lot of researches on garbage classification, but most of the proposed schemes are innovations of terminal recycling method. In 2019, China started to require residential garbage classification, in which case the front-end collection is highly dependent on people's awareness. Therefore, the efficiency of the garbage classification still needs to be improved.

It is of great academic value and practical significance to study an effective automatic garbage classification method. For the commercialization of intelligent garbage bin, there are some attempts made by foreign companies, such as NASTAR in the United States, Zeton, and the Japanese somatosensory intelligent garbage bin. There are a lot of researches on the intelligence garbage bin in academia. Kano et al. proposed a group-control robot system, which provides a The associate editor coordinating the review of this manuscript and approving it for publication was Chao Shen . decentralized control scheme for indoor garbage collection. But it depends on the cooperation between robots and takes a long time. Wesley et al. proposed a low-cost intelligence garbage bin. It optimized the treatment of waste by taking the capacitance and spectrum of waste as the

classification standard. However, this method is highly dependent on the surrounding environment, its classification still has limitations.

The research of garbage classification system is mature, but the accuracy and speed of garbage classification still need to be improved. And there are few researches on garbage classification based on deep learning. At present, deep learning technology is widely used in image classification, and has some remarkable achievements. ResNet proposed by He et al. achieved an accuracy of 93.03% and 95.51% on cifar-10 and Imagenet respectively. Zhong et al. combined residual and Inception blocks, achieved the accuracy of 99.66%, 98.04%, and 95.32% on MNIST, SVHN, and cifar-10, respectively. In the public data set, using deep learning algorithm can get high classification accuracy, which still can be improved. It also provides a new idea for the research of garbage classification. With the rapid development of artificial intelligence,

image classification algorithms are gradually diversified. The mainstream algorithms are AlexNet, VGG, Inception and ResNet. Scholars implemented experiments using these methods and made attempts to modify in order to gain better results. These are helpful for the improvement of network structure in this paper. Thanks to the nonlinear characteristics of activation function, neural network with improved activation function has shown good results. In order to learn the distribution characteristics of the nonlinear data better, some improvements of the network are mainly focused on the network's depth. From AlexNet, VGG, GoogleNet to ResNet-152, the problem of gradient dispersion becomes more and more prominent.

Among most popular algorithms used to solve this problem, the effect of improving the activation function is outstanding. Reference defined the activation function as a mapping which can be derived almost everywhere. The ReLU (corrected linear unit) proposed by Nair has the characteristics of fast convergence speed, simple calculation and strong sparseness. Later, Leaky ReLU, ELU, and PReLU are emerging one after another, and all have achieved good results. RReLU is particular effective when trained with small data set. The SLU is proposed, which

effectively solved the problem of migration and vanishment of gradient. The elu-softsign is proposed in, which effectively

alleviated the invalidity of negative distributed sample and improved the classification accuracy. These provide ideas for improving the activation function of image classification network.

2.2 ALGORITHM

DESIGN BASIC MODEL

SELECTION

To achieve an accurate classification of garbage types for the collected pictures, excellent classification algorithm is needed. Therefore, we have carefully compared and selected the basic model. Current popular image classification algorithms are Inception, KNN, SVM, AlexNet, VGG, Migration Learning, and CNN, among which AlexNet, VGG, Inception V3, and ResNet are the typical models. To select an appropriate image classification model, we use the existing cifar-10 data set to compare LeNet, AlexNet, VGG, Inception V3, and ResNet models.

ResNet series is selected as the basic model. ResNet-34 and ResNet-152 show good performance in both accuracy and loss respectively. However, because the deeper the model, the slower the calculation speed. Therefore, ResNet-34 is chosen as the basic model.

ResNet-34 is proposed by He Kaiming, which originally has 34 layers. In addition to a $7 * 7$ convolution layer and a FC layer, it arranges $3 * 3$ convolution in the order of [3], [4], [6], [3], and performs a dimension reduction at each group junction. This network has a good effect in alleviating the gradient vanishing.

EXPERIMENTAL ENVIRONMENT

In this paper, ResNet-34 is selected for the experiment, which is verified on cifar-10 and MNIST data set. The computer configuration of the experiment

platform is i5-6500 CPU, 8gram, 64 bits Windows 7, and GPU which includes two Tesla-P100.

DATASET

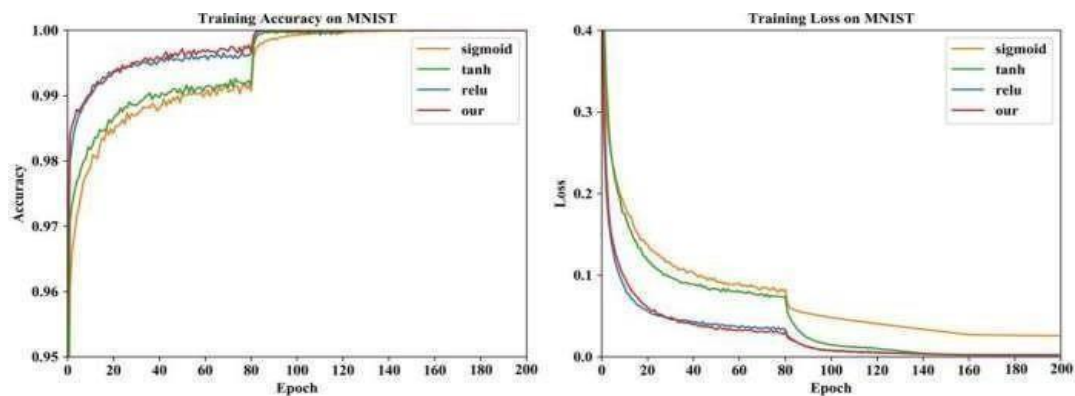
We scrapped the web for images on organic and recyclable items We trained the model with 22564 images of which 12565 images were of organic items and 9999 images were of recyclable items. For the test set we had 1401 images for organic items and 1112 images of recyclable items making a total of 2513 items. classifications:-

- Glass
- Trash
- Paper
- Cardboard
- Plastic
- Metal

We take 501 images for glass classification, 137 images for trash classification, 482 images for paper classification, 403 images for cardboard classification, 482 images for plastic classification, 410 images for metal classification.

EXPERIMENT IN MNIST DATA SET

MNIST contains 10 kinds of handwritten Arabic numerals, including 60 000 training data and 10 000 testing data. The image gray level is 8 and the resolution is 28*28.



CONCLUSION AND FUTURE SCOPE

Aiming at the problem of garbage classification, this paper proposes an improved algorithm based on ResNet-34. Firstly, experiments on common data sets have been implemented to determine the basic model and ResNet-34 shows best performance and is chosen. The model is tested on the garbage data set with 6 types of garbage. The accuracy of a normal convolutional neural network is around 0.9293 for training data and 0.7331 for the validation data. The

accuracy of ResNet-34-A using multi feature fusion is increased to 0.994 1.

Finally, an automatic garbage classification system is integrated with the proposed algorithm.

There are still some problems worthy of further exploration. Firstly, the classification of small target is not well and can be improved in future. Secondly, the classification criteria can be further extended, such as kitchen waste.

3.

PROBLEM ANALYSIS

3.1 EXISTING APPROACH:

For the commercialization of intelligent garbage bin, there are some attempts made by foreign companies, such as NASTAR in the United States, Zeton, and the Japanese somatosensory intelligent garbage bin. There are a lot of researches on the intelligence garbage bin in academia. T. Kano et al. proposed a group-control robot system, which provides a decentralized control scheme for indoor garbage collection. But it depends on the cooperation between robots and takes a long time. P. Wesley et al. proposed a low-cost intelligence garbage bin.

3.1.1 Drawbacks

Data acquisition, highly error-prone and time consuming.

3.2 Proposed System

an automatic garbage classification system is proposed based on deep learning. The overall system of the garbage bin is designed based on ResNet-34 algorithm, and its network structure is further optimized by several aspects to increase the accuracy to a higher level and to obtain an efficient system.

3.2.1 Advantages

The proposed system is capable of producing results with higher accuracy as compared to that of an already existing system.

3.3 Software And Hardware

Requirements SOFTWARE

REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- Python idel 3.7 version (or)
- Anaconda 3.7 (or)
- Jupyter (or)
- Google colab

HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Operating system : windows, linux

- Processor : minimum intel i3

- Ram : minimum 4 gb

- Hard disk : minimum 250gb

3.4 About Dataset

We scrapped the web for images on organic and recyclable items We trained the model with 22564 images of which 12565 images were of organic items and 9999 images were of recyclable items. For the test set we had 1401 images for organic items and 1112 images of recyclable items making a total of 2513 items. classifications:-

- Glass
- Trash
- Paper
- Cardboard
- Plastic
- Metal

We take 501 images for glass classification, 137 images for trash classification, 482 images for paper classification, 403 images for cardboard classification, 482 images for plastic classification, 410 images for metal classification.

3.5 Algorithms Used

- **Numpy** : it is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
- **Matplotlib** : Matplotlib is an amazing visualization library in Python for 2D plots of arrays.

- **Keras :** KERAS is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use.
- **ResNet :** A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers.

3.6 Modules

1. **Data Collection :** Collect sufficient data samples and legitimate software samples.
2. **Data Preprocessing :** Data Augmented techniques will be used for better performance.
3. **Train and Test Modeling:** Split the data into train and test data Train will be used for training the model and Test data to check the performance.

4.

DESIGN

4.1 Convolutional Neural Network

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Example of CNN :

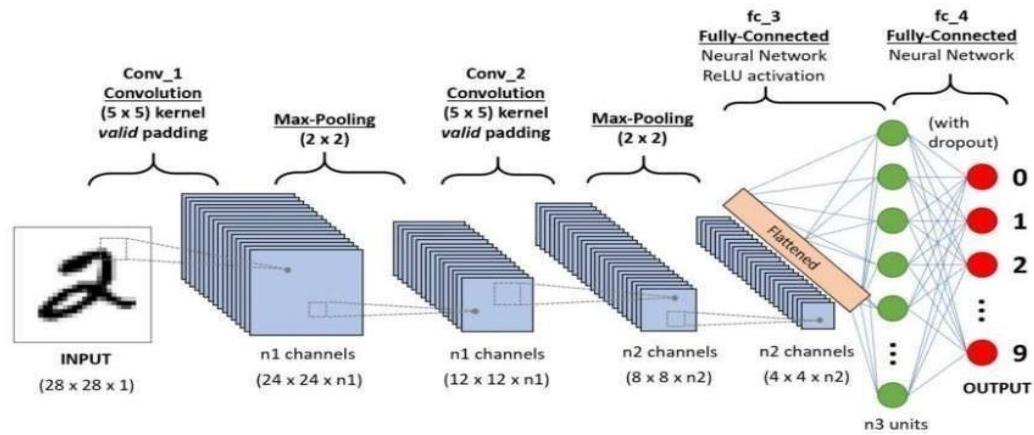


Figure 4.1 A CNN sequence

Convolution Layer — The Kernel

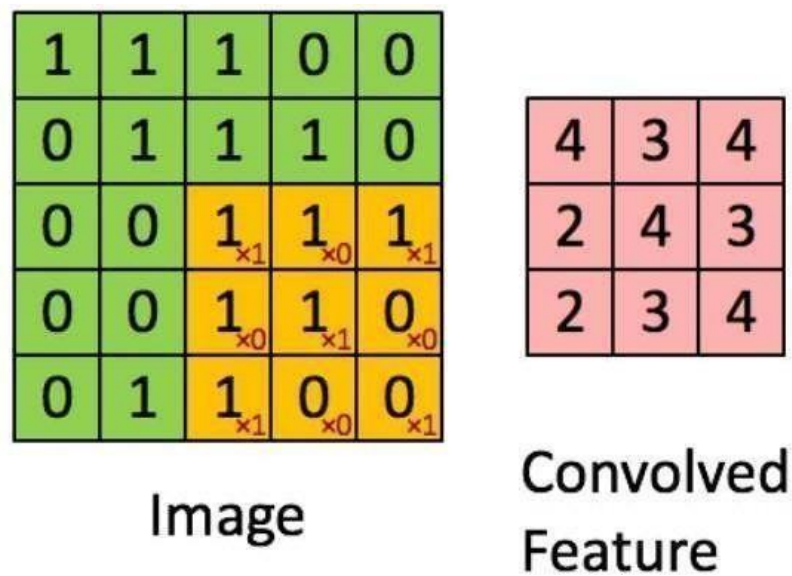


Figure 4.2 Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix.

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

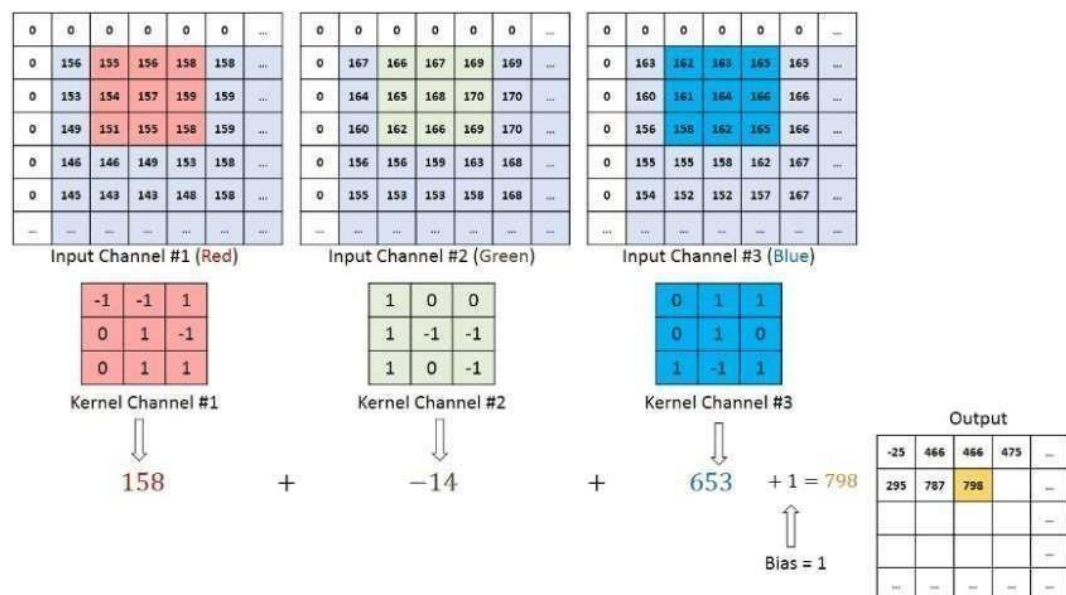


Figure 4.3 Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

Pooling Layer

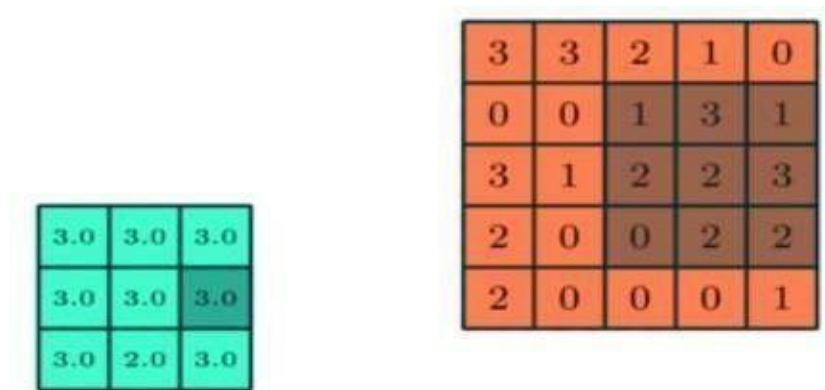


Figure 4.4 3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

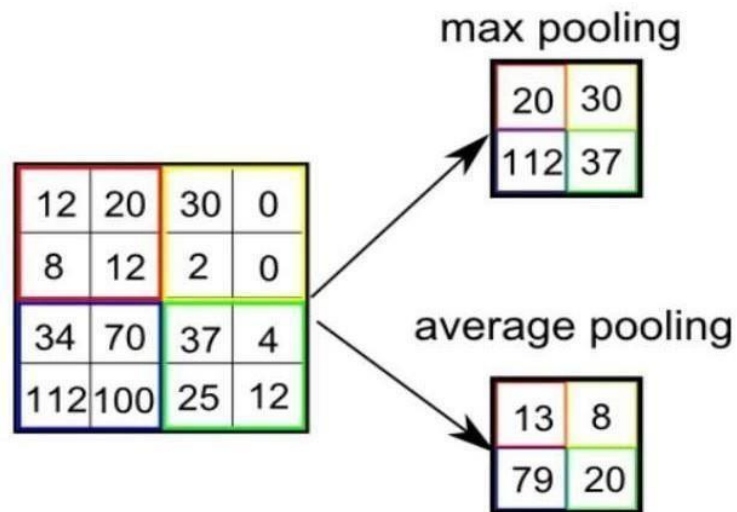


Figure 4.5 Types of Pooling

Fully Connected Layer (FC Layer)

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

1. LeNet
2. AlexNet
3. VGGNet
4. GoogLeNet

5. ResNet

6. ZFNet

4.2 Numpy

NumPy is a general-purpose array-processing package. It provides a high- performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Installation

Mac and Linux users can install NumPy via pip command:

Pip install numpy

4.3 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of

arrays. Matplotlib is a multi-platform data visualization library built on NumPy

arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Installation

Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package :

```
python - mpip install -U matplotlib
```

4.4 Keras

KERAS is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer.

Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

Advantages of Keras

- Fast Deployment and Easy to understand
- Large Community Support
- Have multiple Backends
- Cross-Platform and Easy Model Deployment

4.5 A Residual Neural Network (ResNet)

ResNet is a short name for a residual network, but what's residual learning?

Deep convolutional neural networks have achieved the human level image classification result. Deep networks extract low, middle and high-level features and classifiers in an end-to-end multi-layer fashion, and the number of stacked layers can enrich the “levels” of features.

When the deeper network starts to converge, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Such degradation is not caused by overfitting or by adding more layers to a deep network leads to higher training error. The deterioration of training accuracy shows that not all systems are easy to optimize.

To overcome this problem, Microsoft introduced a deep residual learning framework. Instead of hoping every few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping. The formulation of $F(x)+x$ can be realized by feedforward neural networks with shortcut connections. Shortcut connections are those skipping one or more layers shown in Figure 1. The shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. By using the residual network, there are many problems which can be solved such as:

ResNets are easy to optimize, but the “plain” networks (that simply stack layers) shows higher training error when the depth increases. ResNets can easily gain accuracy from greatly increased depth, producing results which are better than previous networks.

4.2 Architecture of ResNet

1. For the same output feature map, the layers have the same number of filters;
2. If the size of the features map is halved, the number of filters is doubled to preserve the time complexity of each layer.

It is worth noticing that the ResNet model has fewer filters and lower complexity than VGG nets.

Residual Network: Based on the above plain network, a shortcut connection is inserted which turn the network into its counterpart residual version. The identity shortcuts $F(x\{W\}+x)$ can be directly used when the input and output are of the same dimensions . When the dimensions increase , it considers two options:

1. The shortcut performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no additional parameter.
2. The projection shortcut in $F(x\{W\}+x)$ is used to match dimensions (done by 1×1 convolutions).

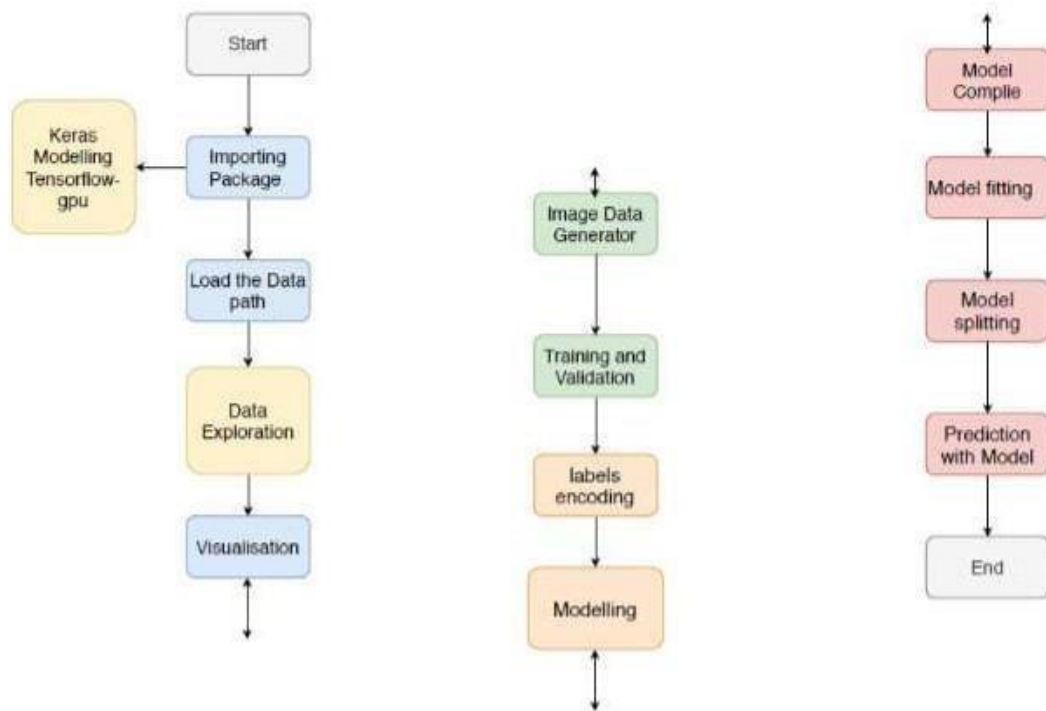
Each ResNet block is either two layers deep (used in small networks like ResNet 18, 34) or 3 layers deep (ResNet 50, 101, 152).

50-layer ResNet: Each 2-layer block is replaced in the 34-layer net with this 3- layer bottleneck block, resulting in a 50-layer ResNet (see above table). They use option 2 for increasing dimensions. This model has 3.8 billion FLOPs.

101-layer and 152-layer ResNets: they construct 101-layer and 152-layer ResNets by using more 3-layer blocks (above table). Even after the depth is increased, the 152-layer ResNet (11.3 billion FLOPs) has lower complexity than VGG-16/19 nets (15.3/19.6 billion FLOPs)

5. IMPLEMENTATION

5.1 Flow Chart



5.2 Code

- import sys
 sys.path.append('/content/drive/My Drive/An Automatic Garbage Classification System Based on Deep Learning')
- ***Importing the package for analysis & classifying the garbage types***
- import numpy as np
 import matplotlib.pyplot as plt
 from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img
 from keras.layers import Conv2D, Flatten, MaxPooling2D, Dense
 from keras.models import Sequential
 import glob, os, random
- import tensorflow as tf

```
from resnet import resnet_18, resnet_34, resnet_50, resnet_101, resnet_152
```

- ***Dataset folder path***

- `base_path = '/content/drive/My Drive/An Automatic Garbage Classification System Based on Deep Learning/Garbage classification'`

```
img_list = glob.glob(os.path.join(base_path, '**/*.jpg'))  
print(len(img_list))
```

- ***Visualize the first 6 image from the path***

- ```
for i, img_path in enumerate(random.sample(img_list,
6)): img = load_img(img_path)
img = img_to_array(img, dtype=np.uint8)
plt.subplot(2, 3, i+1)
plt.imshow(img.squeeze())
```

- ***Training the images based on the labels with splitting the data to train and test which result***

***Found 2276 images belonging to 6 classes. Found 251 images belonging to 6 classes.***

- ```
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    shear_range=0.1,  
    zoom_range=0.1,  
    width_shift_range=0.1,  
    height_shift_range=0.1,  
    horizontal_flip=True,  
    vertical_flip=True,
```

validation_split=0.1

)

```
test_datagen = ImageDataGenerator(  
    rescale=1./255,  
    validation_split=0.1  
)
```

- ```
train_generator =
 train_datagen.flow_from_directory(base_path,
 target_size=(300, 300),
 batch_size=16,
 class_mode='categorical'
 , subset='training',
 seed=0
)
```

```
validation_generator = test_datagen.flow_from_directory(
 base_path,
 target_size=(300, 300),
 batch_size=16,
 class_mode='categorical'
 , subset='validation',
 seed=0
)
```

- ```
labels = (train_generator.class_indices)  
labels = dict((v,k) for k,v in labels.items())
```

```
print(labels)
```

- **Neural Network Model**

- ```
model = Sequential([
 Conv2D(filters=32, kernel_size=3, padding='same', activation='relu',
inp ut_shape=(300, 300, 3)),
 MaxPooling2D(pool_size=2),

 Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'),
 MaxPooling2D(pool_size=2),

 Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
 MaxPooling2D(pool_size=2),

 Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
 MaxPooling2D(pool_size=2),

 Flatten(),

 Dense(64, activation='relu'),

 Dense(6, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics
=['acc'])

model.summary()
```

- **Predicting based on NN**

- ```
model.fit_generator(train_generator, epochs=100, validation_data=validati
on_generator)
```

- `test_x, test_y = validation_generator._getitem____(1)`

```
preds = model.predict(test_x)
```

```
plt.figure(figsize=(16, 16))
```

```
for i in range(16):
```

```
    plt.subplot(4, 4, i+1)
```

```
    plt.title('pred:%s / truth:%s' % (labels[np.argmax(preds[i])], labels[np.argmax(test_y[i])]))
```

```
    plt.imshow(test_x[i])
```

Modelling with ResNET-50

ResNet-50

- `model1 = resnet_50()`
- `image_height = 224`
`image_width = 224`
`channels = 3`
- `model1.build(input_shape=(None, image_height, image_width, channels))`
- `model1.summary()`
- `predictions = model1(test_x, training=True)`

- `plt.figure(figsize=(16, 16))`

```

for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.title('pred:%s / truth:%s' % (labels[np.argmax(preds[i])],
labels[ np.argmax(test_y[i])]))
    plt.imshow(test_x[i])

```

Modelling with ResNET-34

ResNet-34

- `model2 = resnet_34()`
- `model2.build(input_shape=(None, image_height, image_width, channels))`
`model2.summary()`
- `predictions = model2(test_x, training=True)`
`plt.figure(figsize=(16, 16))`
for i in range(16):
 `plt.subplot(4, 4, i+1)`
 `plt.title('pred:%s / truth:%s' % (labels[np.argmax(preds[i])], labels[np.argmax(test_y[i])]))`
 `plt.imshow(test_x[i])`
- *Plot the loss and accuracy curves for training and validation*
- `fig, ax = plt.subplots(2,1)`
 `ax[0].plot(hist.history['loss'], color='b', label="Training loss")`

```
ax[0].plot(hist.history['val_loss'], color='r', label="validation loss", axes
=ax[0])
```

```
legend = ax[0].legend(loc='best', shadow=True)
```

```
ax[1].plot(hist.history['acc'], color='b', label="Training accuracy")
```

```
ax[1].plot(hist.history['val_acc'], color='r', label="Validation accuracy")
```

```
legend = ax[1].legend(loc='best', shadow=True)
```

6.

TESTING AND VALIDATION

6.1 SOFTWARE TESTING

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

6.1.1 Types of Testing

1. White Box Testing
2. Black Box Testing
3. Unit testing
4. Integration Testing
5. Alpha Testing
6. Beta Testing
7. Performance Testing and so on

White Box Testing

Testing technique based on knowledge of the internal logic of an Application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers.

Black Box Testing

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

Unit Testing

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

Integration Testing

The phase in software testing in which individual software modules are combined

and tested as a group. It is usually conducted by testing teams.

Alpha Testing

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

Beta Testing

Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

Performance Testing

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

6.2 Execution Environment

Google Colab

It is a free jupyter notebook environment provided by Google

Google Colab is a free cloud service and now it supports free GPU. You can improve your Python programming language coding skills. Develop deep learning applications using popular libraries such as Keras, TensorFlow, Py Torch, and Open CV.

If you want to create a machine learning model but say you don't have a computer that can take the workload, Google Colab is the platform for you. Even if you have a GPU or a good computer creating a local environment with anaconda and installing packages and resolving installation issues are a hassle.

The main purpose of using this is it is free to use and the training examples are

directly tested by itself and they are executed with high speed compared to our

personal systems. And they provide high security, not only safety but also our system does not crash due to load. It is a good move by Google for providing free services.

We can start our colab account by using our Google account and the direct link is <https://colab.research.google.com/notebooks/welcome.ipynb#recent=true>

We follow some guidelines to create a project or to execute basic programs using Google Colab.

6.2.1 Steps to start the Google Colab

STEP 1 :

Open the google colab link, when we see the below image click on the NEW NOTEBOOK which can be seen at the bottom of the screen.

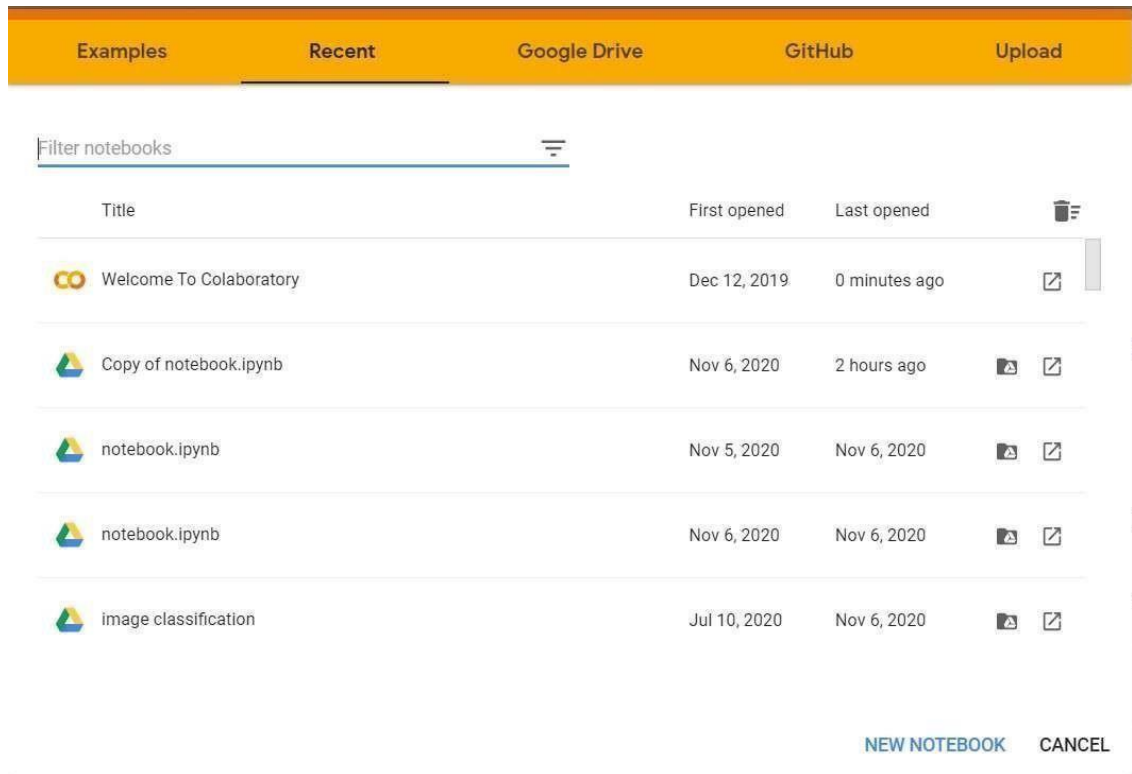


Figure 6.1 Opening Google Colab

STEP 2 :

You see the new window with the command line where you can execute the code and also change the runtime type to GPU (graphics process unit). Because then it makes the code run much faster.

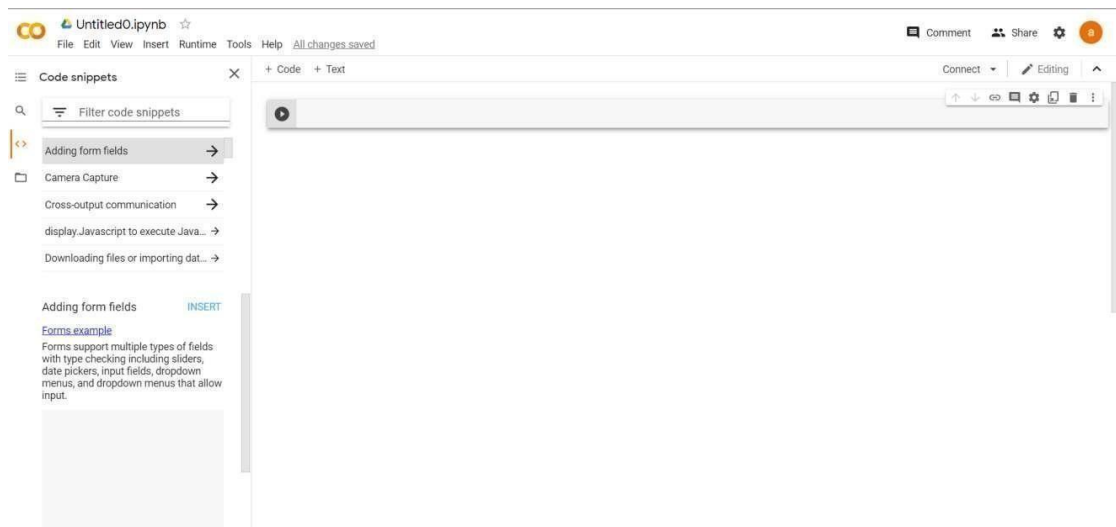


Figure 6.2 Getting Started with Google Colab

STEP 3 :

Creating a Project : Go to the home page of google colab then you can see the create project option then click on it and follow the procedure to complete the project. Select any of the three options and then enter the commands on the command line then execute. It then activates the API key for you which is useful at the time of execution of machine learning program.

STEP 4 :

Setting up the GPU: Want to use GPU? It's as simple as going to the “runtime” dropdown menu, selecting “change runtime type” and selecting GPU in the hardware accelerator drop-down menu.

Notebook settings

Runtime type
Python 3 ▼

Hardware accelerator
GPU ▼

☐ Omit code cell output when saving this notebook

CANCEL SAVE

Figure 6.3 Setting up the GPU

6.2.2 Pros of Google Colab:

- Pre-built with lots of python library
- Quick Start of python learning
- No infra setup required
- No charges for GPU usage
- Can run your code for 24 hrs without interruptions but not more than that
- Your notebooks is saved in google drive only

6.2.3 Cons of Google Colab:

- Need to install all specific libraries which does not come with standard python (Need to repeat this with every session)
- Google Drive is your source and target for Storage, there are other like local (which eats your bandwidth if dataset is big)
- Google provided the code to connect and use with google drive but that

will not work with lots of other data format

- Google Storage is used with current session, so if you have downloaded some file and want to use it later, better save it before closing the session.
- Difficult to work with BIGGER datasets as you have to download and store them in Google drive (15 GB Free space with gmail id, additional required a payment towards google).

7.

RESULT ANALYSIS

Importing the package for analysis & classifying the garbage types

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] import sys
sys.path.append('/content/drive/My Drive/An Automatic Garbage Classification System Based on Deep Learning')

[ ] import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img
from keras.layers import Conv2D, Flatten, MaxPooling2D, Dense
from keras.models import Sequential

import glob, os, random

[ ] import tensorflow as tf
from resnet import resnet_18, resnet_34, resnet_50, resnet_101, resnet_152

[ ] base_path = '/content/drive/My Drive/An Automatic Garbage Classification System Based on Deep Learning/Garbage classification'

img_list = glob.glob(os.path.join(base_path, '*/*.jpg'))

print(len(img_list))

2527
```

Figure 7.1 Importing Packages

Visualize the first 6 image from the path

```
[ ] for i, img_path in enumerate(random.sample(img_list, 6)):
    img = load_img(img_path)
    img = img_to_array(img, dtype=np.uint8)

    plt.subplot(2, 3, i+1)
    plt.imshow(img.squeeze())
```



Training the images based on the labels with splitting the data to train and test which result

Found 2276 images belonging to 6 classes.

Found 251 images belonging to 6 classes.

```
[ ] train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    shear_range=0.1,  
    zoom_range=0.1,  
    width_shift_range=0.1,  
    height_shift_range=0.1,  
    horizontal_flip=True,  
    vertical_flip=True,  
    validation_split=0.1  
)  
  
test_datagen = ImageDataGenerator(  
    rescale=1./255,  
    validation_split=0.1  
)
```

Figure 7.2 Splitting the data into train and test

```
[ ] train_generator = train_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    seed=0
)

validation_generator = test_datagen.flow_from_directory(
    base_path,
    target_size=(300, 300),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    seed=0
)

Found 2276 images belonging to 6 classes.
Found 251 images belonging to 6 classes.
```

```
[ ] labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())

print(labels)

{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

Figure 7.3 Printing labels

Neural Network Model

```
[ ] model = Sequential([
    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=(300, 300, 3)),
    MaxPooling2D(pool_size=2),

    Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=2),

    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=2),

    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
    MaxPooling2D(pool_size=2),

    Flatten(),

    Dense(64, activation='relu'),

    Dense(6, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 300, 300, 32)	896
max_pooling2d (MaxPooling2D)	(None, 150, 150, 32)	0
conv2d_1 (Conv2D)	(None, 150, 150, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 75, 75, 64)	0
conv2d_2 (Conv2D)	(None, 75, 75, 32)	18464
max_pooling2d_2 (MaxPooling2D)	(None, 37, 37, 32)	0
conv2d_3 (Conv2D)	(None, 37, 37, 32)	9248
max_pooling2d_3 (MaxPooling2D)	(None, 18, 18, 32)	0
flatten (Flatten)	(None, 10368)	0
dense (Dense)	(None, 64)	663616
dense_1 (Dense)	(None, 6)	390
Total params: 711,110		
Trainable params: 711,110		
Non-trainable params: 0		

Figure 7.4 Building a Neural Network Model

```
[ ] model.fit_generator(train_generator, epochs=100, validation_data=validation_generator)
#model.fit_generator(train_generator, epochs=20, validation_data=validation_generator)

Epoch 1/100
143/143 [=====] - 1305s 9s/step - loss: 1.5983 - acc: 0.3238 - val_loss: 1.4569 - val_acc: 0.4143
Epoch 2/100
143/143 [=====] - 55s 385ms/step - loss: 1.4124 - acc: 0.4183 - val_loss: 1.4899 - val_acc: 0.3546
Epoch 3/100
143/143 [=====] - 55s 384ms/step - loss: 1.3219 - acc: 0.4583 - val_loss: 1.3132 - val_acc: 0.4582
Epoch 4/100
143/143 [=====] - 55s 382ms/step - loss: 1.2710 - acc: 0.4930 - val_loss: 1.3161 - val_acc: 0.4502
Epoch 5/100
143/143 [=====] - 55s 382ms/step - loss: 1.1935 - acc: 0.5233 - val_loss: 1.2412 - val_acc: 0.4940
Epoch 6/100
143/143 [=====] - 55s 385ms/step - loss: 1.1143 - acc: 0.5545 - val_loss: 1.1688 - val_acc: 0.5538
Epoch 7/100
143/143 [=====] - 55s 383ms/step - loss: 1.0661 - acc: 0.5874 - val_loss: 1.1411 - val_acc: 0.5697
Epoch 8/100
143/143 [=====] - 55s 383ms/step - loss: 1.0409 - acc: 0.6041 - val_loss: 1.1275 - val_acc: 0.5378
Epoch 9/100
143/143 [=====] - 55s 382ms/step - loss: 1.0348 - acc: 0.5949 - val_loss: 1.1882 - val_acc: 0.5179
Epoch 10/100
143/143 [=====] - 55s 382ms/step - loss: 0.9920 - acc: 0.6147 - val_loss: 1.1310 - val_acc: 0.5538
Epoch 11/100
143/143 [=====] - 54s 379ms/step - loss: 0.9571 - acc: 0.6476 - val_loss: 1.0551 - val_acc: 0.5896
Epoch 12/100
143/143 [=====] - 55s 383ms/step - loss: 0.9404 - acc: 0.6344 - val_loss: 1.0060 - val_acc: 0.5896
Epoch 13/100
143/143 [=====] - 55s 382ms/step - loss: 0.9216 - acc: 0.6520 - val_loss: 0.9746 - val_acc: 0.5817
Epoch 14/100
143/143 [=====] - 54s 375ms/step - loss: 0.8764 - acc: 0.6714 - val_loss: 1.0187 - val_acc: 0.6255
Epoch 15/100
143/143 [=====] - 54s 376ms/step - loss: 0.8332 - acc: 0.6920 - val_loss: 1.1333 - val_acc: 0.6056
Epoch 16/100
143/143 [=====] - 54s 377ms/step - loss: 0.8206 - acc: 0.6938 - val_loss: 0.9629 - val_acc: 0.6414
Epoch 17/100
143/143 [=====] - 54s 378ms/step - loss: 0.7805 - acc: 0.7118 - val_loss: 1.0759 - val_acc: 0.6375
Epoch 18/100
```

```
Epoch 83/100
143/143 [=====] - 53s 370ms/step - loss: 0.2604 - acc: 0.9051 - val_loss: 1.0443 - val_acc: 0.7092
Epoch 84/100
143/143 [=====] - 53s 371ms/step - loss: 0.2118 - acc: 0.9209 - val_loss: 1.0266 - val_acc: 0.7291
Epoch 85/100
143/143 [=====] - 53s 370ms/step - loss: 0.2875 - acc: 0.9011 - val_loss: 0.9602 - val_acc: 0.7490
Epoch 86/100
143/143 [=====] - 53s 371ms/step - loss: 0.2503 - acc: 0.9073 - val_loss: 1.3562 - val_acc: 0.7131
Epoch 87/100
143/143 [=====] - 53s 372ms/step - loss: 0.2167 - acc: 0.9266 - val_loss: 0.9278 - val_acc: 0.7371
Epoch 88/100
143/143 [=====] - 53s 369ms/step - loss: 0.2167 - acc: 0.9244 - val_loss: 1.1287 - val_acc: 0.7052
Epoch 89/100
143/143 [=====] - 53s 370ms/step - loss: 0.2479 - acc: 0.9227 - val_loss: 1.3198 - val_acc: 0.7410
Epoch 90/100
143/143 [=====] - 53s 372ms/step - loss: 0.2564 - acc: 0.9091 - val_loss: 0.9948 - val_acc: 0.7649
Epoch 91/100
143/143 [=====] - 53s 370ms/step - loss: 0.2199 - acc: 0.9240 - val_loss: 0.7754 - val_acc: 0.7649
Epoch 92/100
143/143 [=====] - 53s 370ms/step - loss: 0.2410 - acc: 0.9178 - val_loss: 0.9903 - val_acc: 0.7530
Epoch 93/100
143/143 [=====] - 54s 374ms/step - loss: 0.2268 - acc: 0.9196 - val_loss: 0.9105 - val_acc: 0.7570
Epoch 94/100
143/143 [=====] - 53s 367ms/step - loss: 0.2058 - acc: 0.9249 - val_loss: 1.0567 - val_acc: 0.7689
Epoch 95/100
143/143 [=====] - 53s 370ms/step - loss: 0.2409 - acc: 0.9117 - val_loss: 1.0028 - val_acc: 0.7490
Epoch 96/100
143/143 [=====] - 53s 368ms/step - loss: 0.2141 - acc: 0.9244 - val_loss: 1.1661 - val_acc: 0.7530
Epoch 97/100
143/143 [=====] - 53s 371ms/step - loss: 0.2151 - acc: 0.9257 - val_loss: 0.9824 - val_acc: 0.7371
Epoch 98/100
143/143 [=====] - 54s 376ms/step - loss: 0.1883 - acc: 0.9350 - val_loss: 1.0890 - val_acc: 0.7490
Epoch 99/100
143/143 [=====] - 53s 371ms/step - loss: 0.1890 - acc: 0.9341 - val_loss: 0.9010 - val_acc: 0.7570
Epoch 100/100
143/143 [=====] - 53s 369ms/step - loss: 0.2027 - acc: 0.9293 - val_loss: 1.1387 - val_acc: 0.7331
<tensorflow.python.keras.callbacks.History at 0x7f5740038cf8>
```

Figure 7.5 Measuring the accuracy

Predicting based on NN

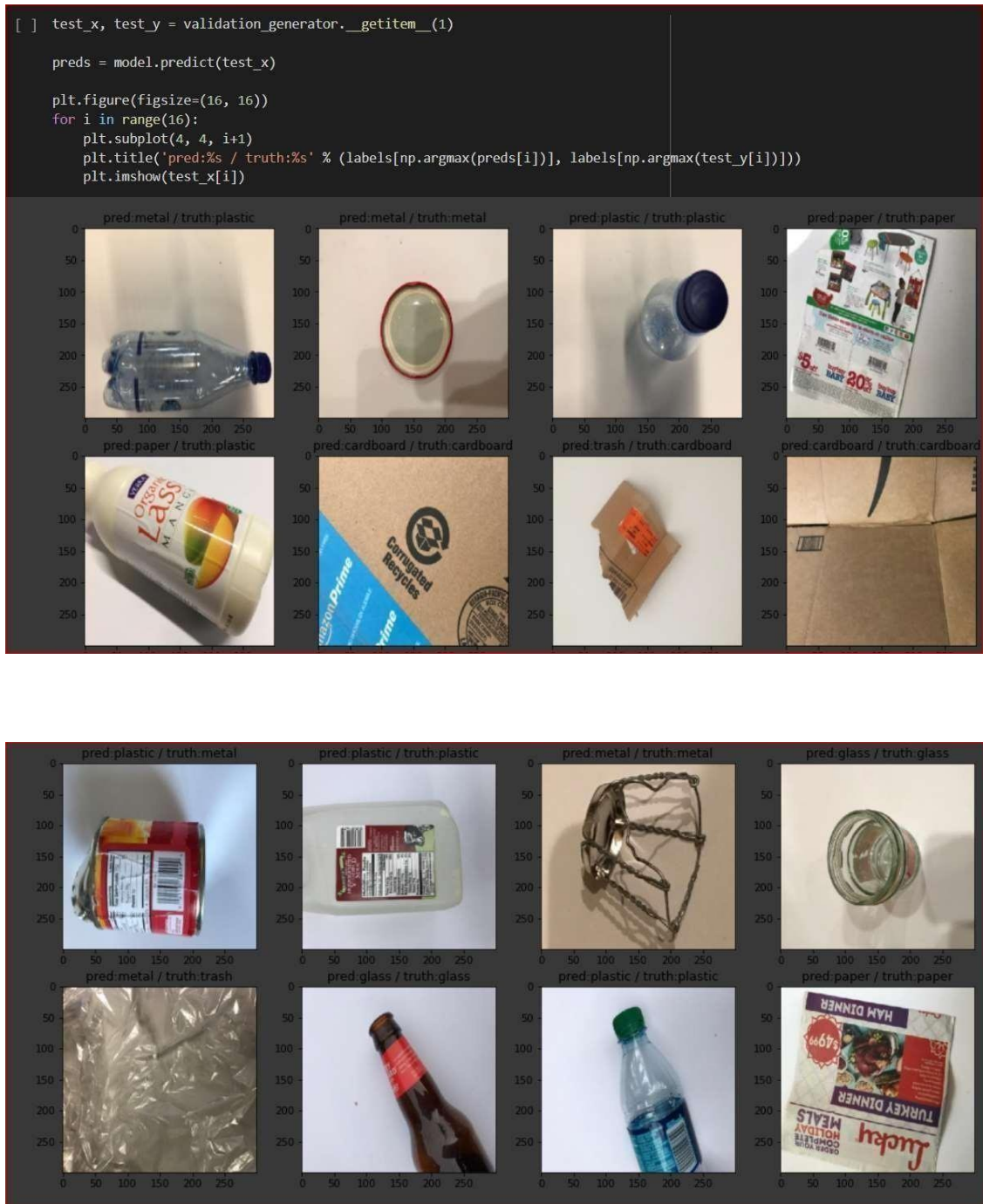


Figure 7.6 Output of Neural Network Model

Modelling with ResNET-50

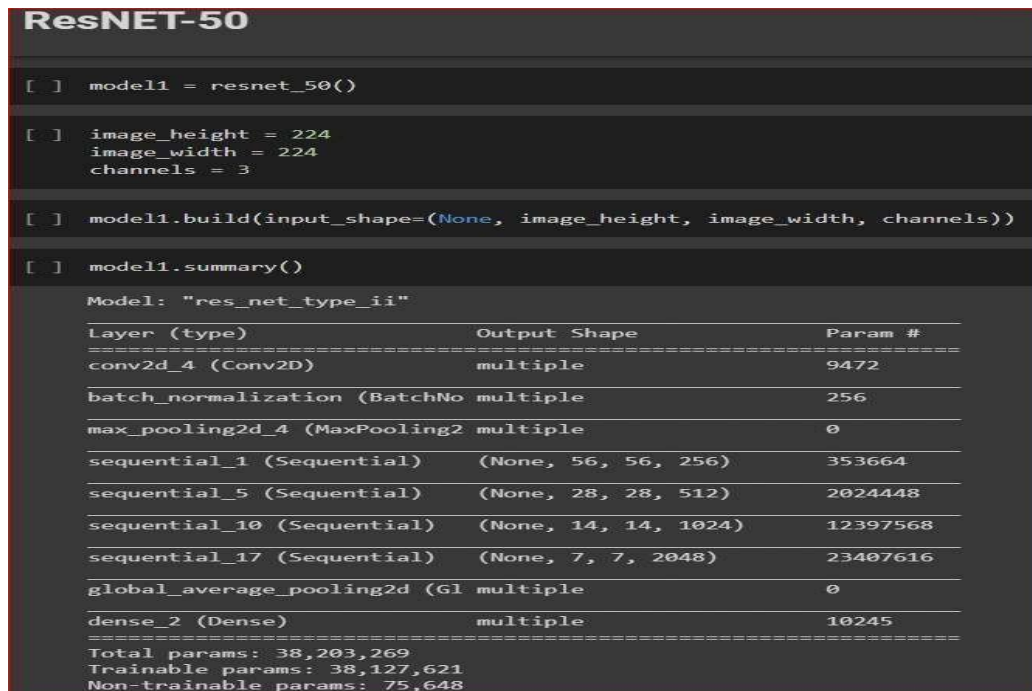


Figure 7.7 ResNet 50 Model Building

Prediction based on ResNet-50

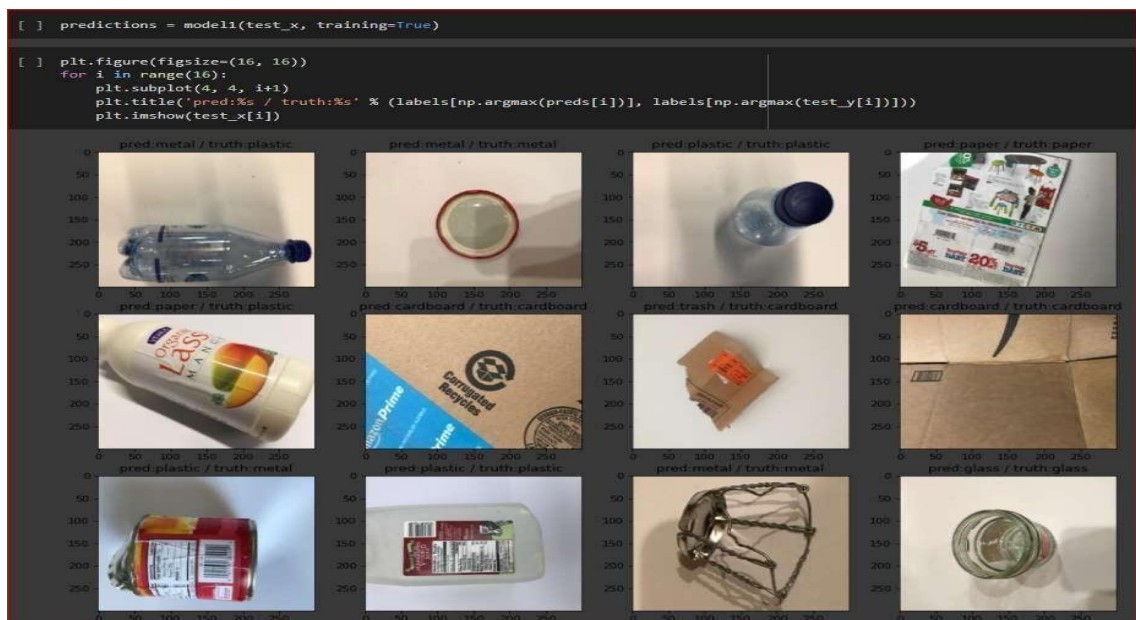


Figure 7.8 ResNet 50 Prediction

Modelling with ResNET-34

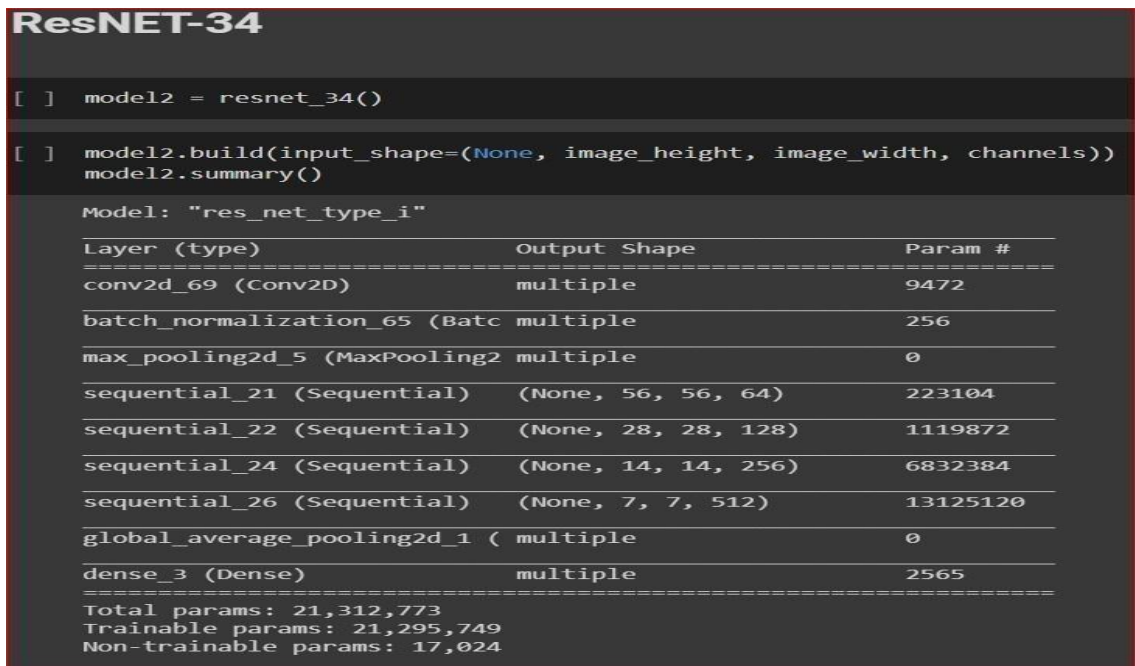


Figure 7.9 ResNet 34 Model Building

Prediction based on ResNet-34

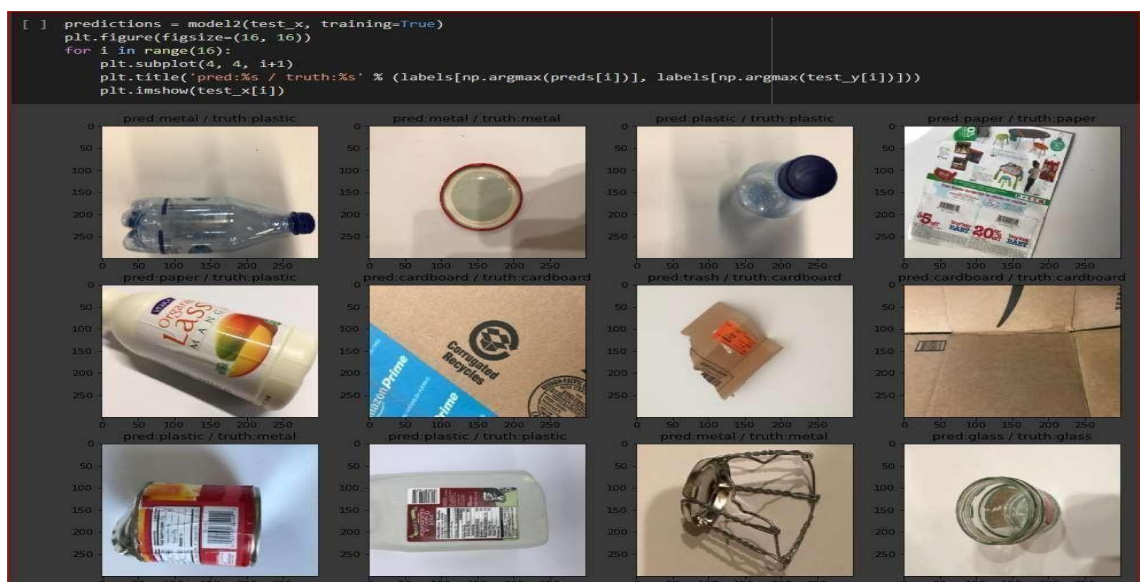


Figure 7.10 ResNet 34 Prediction



Figure 7.11 Plotting loss and accuracy curves

Aiming at the problem of garbage classification, this paper proposes an improved algorithm based on ResNet-34. Firstly, experiments on common data sets have been implemented to determine the basic model and ResNet-34 shows best performance and is chosen. The model is tested on the garbage data set with 6 types of garbage. The accuracy of a normal convolutional neural network is around 0.9293 for training data and 0.7331 for the validation data. The accuracy of ResNet-34-A using multi feature fusion has been increased. Finally, an automatic garbage classification system is integrated with the proposed algorithm.

There are still some problems worthy of further exploration. Firstly, the classification of small target is not well and can be improved in future. Secondly, the classification criteria can be further extended, such as kitchen waste.

J.-J. Wang, N.-N. Zhao, and J.-H. Li, "Current situation of marine microplastics pollution and prevention proposals in China," *China Environ. Sci.*, vol. 39, no. 7, pp. 3056_3063, Jul. 2019, doi: [10.19674/j.cnki.issn1000-6923.2019.0360](https://doi.org/10.19674/j.cnki.issn1000-6923.2019.0360).

W.-B. Li, G. Ma, E.-Q. Yang, Y.-M. Cai, Z. Chen, R.-F. Gao, J.-H. Yan, X.-F. Cao, and E.-J. Pan, "Study on characteristics of electric dust removal fly ash and bag fly ash in circulating fluidized bed waste incineration system," *Proc. CSEE*, vol. 39, no. 5, pp. 1397_1405, Mar. 2019, doi: [10.13334/j.0258-8013.pcsee.181110](https://doi.org/10.13334/j.0258-8013.pcsee.181110).

D. Porshnov, V. Ozols, and M. Klavins, "Thermogravimetric analysis as express tool for quality assessment of refuse derived fuels used for pyro-gasification," *Environ. Technol.*, vol. 41, no. 1, pp. 35_39, Mar. 2020.

P. Kellow, R. J. P. C. Joel, D. Ousmane, D. A. Kumar, D.-A. C. V. Hugo, and K. A. Sergei, "A smart waste management solution geared towards citizens," *Sensors*, vol. 20, no. 8, pp. 1_15, Apr. 2020, doi: [10.3390/s20082380](https://doi.org/10.3390/s20082380).

Y.-G. Cheng, N. Chen, and H. Zhang, "Coal fly ash as an inducer to study its application in the production of methane gas from domestic waste," *Fresenius Environ. Bull.*, vol. 29, no. 2, pp. 1082_1089, 2020.

Y. Ren, J.-J. Yang, H.-J. Cao, Q.-Y. Zhang, and Q. Liu, "All components resourcing system of rural garbage based on post-gather automatic sorting and disposal technology and its application," *Trans. Chin. Soc. Agricult. Eng.*, vol. 35, no. 4, pp. 248_254, Feb. 2019, doi: [10.11975/j.issn.1002-6819.2019.04.031](https://doi.org/10.11975/j.issn.1002-6819.2019.04.031).

D.-E. Zhao, R. Wu, B.-G. Zhao, and Y.-Y. Chen, "Research on garbage classification and recognition on hyperspectral imaging technology," *Spectrosc. Spectral Anal.*, vol. 39, no. 3, pp. 921_926, 2019.

T. Kano, E. Naito, and T. Aoshima, ``Decentralized control for swarm

robots that can effectively execute spatially distributed tasks," *Artif. Life*, vol. 26, pp. 243_260, Apr. 2020, doi: [10.1162/artl_a_00317](https://doi.org/10.1162/artl_a_00317).

W. Pereira, S. Parulekar, S. Phaltankar, and V. Kamble, "Smart bin (waste segregation and optimisation)," in *Proc. Amity Int. Conf. Artif. Intell. (AICAI)*, Dubai, United Arab Emirates, Feb. 2019, pp. 274_279.

K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Dec. 2015, pp. 770_778. [Online]. Available: <http://arxiv.org/abs/1512.03385>

<https://www.investopedia.com/terms/d/deeplearning.asp#:~:text=Deep%20learning%20is%20an%20AI,is%20both%20unstructured%20and%20unlabeled>.

<https://www.analyticssteps.com/blogs/deep-learning-overview-practical-examples-popular-algorithms>

<https://neurohive.io/en/popular-networks/resnet/>

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

<https://www.geeksforgeeks.org/python-introduction-matplotlib/>

<https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/>