EFFECTIVE WASTE MANAGEMENT BY SMART GARBAGE CLASSIFIER USING MACHINE LEARNING

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Parcha Srikanth Rao 221710309042

Penukonda Gopala Krishna Akhila 221710309045

Pranav Sundaresan Babu 221710309048

Nagineni Sai Kamal 221710309038

Under the esteemed guidance of

Mrs K Venkata Ramani

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

HYDERABAD

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "EFFECTIVE WASTE MANAGEMENT BY SMART GARBAGE CLASSIFIER USING MACHINE LEARNING" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or university for the award of any degree or diploma.

Date: 10/05/2021

221710309042 Parcha Srikanth Rao

221710309045 Penukonda Gopala Krishna Akhila

221710309048 Pranav Sundaresan Babu

221710309038 Nagineni Sai Kamal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "EFFECTIVE WASTE MANAGEMENT BY SMART GARBAGE CLASSIFIER USING MACHINE LEARNING" is a bonafide record of work carried out by **Parcha Srikanth Rao 221710309042, Penukonda Gopala Krishna Akhila 221710309045, Pranav Sundaresan Babu 221710309048, Nagineni Sai Kamal 221710309038** students submitted in partial fulfilment of the requirement for the award of the degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Head of the Department**
Mrs K Venkata Ramani	Dr S Phani Kumar
Assistant Professor	Professor

ACKNOWLEDGMENT

Our Major Project would not have been successful without the help of several people. we would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are extremely thankful to our honourable Pro-Vice Chancellor, **Prof N Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of our Major project.

We are highly indebted to **Prof N Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the Major project.

We are very much obliged to our beloved **Prof S Phani Kumar**, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this seminar and encouragement in completion of this Major project.

We hereby wish to express our deep sense of gratitude to the project coordinator **Dr S Aparna**, Assistant Professor, Department of Computer Science and Engineering, School of Technology and our guide **Mrs K Venkata Ramani,** Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the Major project.

We are also thankful to all the staff members of Computer Science and Engineering department who have cooperated in making our major project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our major project work.

Sincerely,

Parcha Srikanth Rao (221710309042)

Penukonda Gopala Krishna Akhila (221710309045)

Pranav Sundaresan Babu (221710309048)

Nagineni Sai Kamal (221710309038)

I TABLE OF CONTENTS

II LIST OF SCREENS

III LIST OF FIGURES

IV LIST OF TABLES

1 ABSTRACT

India is one of the most populated countries on the planet. Due to its sheer population, a lot of waste is produced daily. Approximately 10 million tons of waste are produced just by large metropolitan cities in India. Due to the massive amounts of waste, it is essential to manage it efficiently. The merits of managing such large amounts of waste are in the form of recycling. Recycling this waste saves both the environment and money. In India, due to socioeconomic and other issues, classification is done manually. In order to save money and time, this cumbersome process can be done through Machine Learning algorithms. Elements in the mages are classified into plastic, paper, cardboard and metals since they are the most common types of waste. Machine learning algorithms can be used to solve this cumbersome problem by inputting images and detecting the types of trash in them. This will be really helpful in identifying the types of waste in a given pile, with the help of a simple photograph. And, the dataset consists of various images of plastic, paper, cardboard and metal. A a convolutional neural network will be implemented in this project, and transfer learning will be implemented using VGG19 and Inception V3.

2 INTRODUCTION

2.1 WASTE

Waste is anything that has been discarded as it has already fulfilled its primary purpose. A waste product can turn into a joint product, a resource or a byproduct through an invention. In general, all the waste on this planet can be broadly classified into hazardous waste, municipal waste, biomedical waste and some special hazardous waste.

Billions of tonnes of waste are produced every year. It is essential to manage all of the waste in the most efficient manner. If not addressed in proper ways, humanity shall bear the costs of the consequences that will entail. Some of the costs are as follows-

Environmental costs- When waste is not managed properly, all of it can attract insects. These rodents will harbour harmful parasites which might cause deadly diseases such as worms, yellow fever, the bubonic plague, and other potentially life-threatening illnesses.

Social costs- Not just harm to the society, but the socio-environmental issue. These costs are borne by many individuals who are unfairly discriminated against, some of which include racial minorities, opposers of the reigning regime and others.

Economic costs- The burdens of dealing with waste are not low, and it is pretty high. And these prices are paid by the government and at the end by the average citizen. Money can be quite easily saved by designing better ways of mitigating pollution-reducing technology.

2.2 RECYCLING

Recycling can be defined as the process of converting a waste product or waste material into a new and usable product. The ideology of this concept is to receive the original state of the original product in some manner. Recycling is so important because it "saves" the material from being wasted. In a way, it reinvigorates it and turns it into something useful, which saves the atmosphere from a further increase in the greenhouse gas levels. Therefore, recycling essentially reduces the effects of incineration, which cause air pollution, the energy used to destroy the waste and landfilling, which causes water pollution.

Recycling helps in achieving the aim of complete environmental sustainability. Recyclable materials come in a lot of shapes and forms. Some of the most popular ones include cardboard, plastic, metal, paper, glass, rubber, tyre, batteries, electronics and other chemicals. Biodegradable waste can be broadly classified into gardening waste and food. The Source of most of this kind of waste is generally houses and industries that deal with biodegradability.

2.3 SMART AUTOMATIC GARBAGE CLASSIFIER

Due to the production of so much waste, leaving the waste to rot and not making efficient use of it is futile. Apart from it, it also causes loss of revenue and causes significant damage to the environment. Therefore, it is becoming increasingly essential to segregate waste.

The most popular kinds of recyclable wastes are cardboard, paper, plastic and metal. AS these are so widespread, it is imperative to segregate them. However, since these are so common, they are also produced in quite an abundance. Therefore, it is exceedingly painstaking to segregate them manually. Furthermore, these kinds of waste are not just by themselves. They are more often than not mixed with other types of waste that might be potentially harmful to human beings, making it impossible to segregate the waste manually.

Therefore, the project automatically segregates such waste using deep learning techniques and specifically convolutional neural networks and transfer learning. This is done by feeding the algorithm pictures of wastes segregated into their respective classes. They are subsequently provided to the convolutional neural networks to classify those images into a kind of waste.

These convolutional neural networks are much safer and faster than manually going through the waste to classify them.

3 LITERATURE REVIEW

1. "Image classification optimization and results analysis based on VGG network" [References, A, 2]

   Classification accuracy and network performance are affected by many factors while classifying Image by convolution neural network. The paper takes garbage image classification as an example, the common performance optimization methods of garbage image classification by using VGGNet model were summarised. And VGG-16 was used to test the target data set of garbage images. The classification accuracy under different conditions was measured and compared. Finally, the performance of the parameters under various optimisation schemes is analyzed. 70.36% is the accuracy achieved.

2. "Garbage Recognition and Classification System Based on Convolutional Neural Network VGG16" [References, A, 3]

   This paper is aiming to solve the problem of the identification of recyclable waste. This model in this paper is VGG 19. OpenCV is used to locate and preprocess the image. The input is an image of size 224 x 224. The image is a colour image. The model achieves an accuracy of 81.1%

3. "Automatic Image-Based Waste Classification. In: Ferrández Vicente J., Álvarez-Sánchez J., de la Paz López F., Toledo Moreo J., Adeli H. (eds) From Bioinspired Systems and Biomedical Applications to Machine Learning" [References, A, 4]

   This paper claims computer vision and deep learning can help in the automatic classification of images. This paper uses the "TrashNet" dataset, which has over 2500 images. Transfer learning is employed by this paper. The model uses VGG, Inception and ResNet models. ResNet was the best performing model with an accuracy of 86.6%.

4. "Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network, Procedia Manufacturing" [References, A, 5]

   This paper proposed an intelligent waste classification that made use of transfer learning and traditional machine learning. The authors developed a 50-layer residual CNN, known as ResNet-50. This model was used as an extractor to the Support Vector Machine (SVM), which classifies between several classes such as cardboard, paper, plastic etc. This model achieved an accuracy of 87%.

5. “Waste Classification using Convolutional Neural Network” [References, A, 6]
   This paper proposes a multiple-layered CNN, which aims to classify between cardboard, paper etc. This model is Inception V3. This model used the weights of ImageNet. The model achieved an accuracy of 92.5%.

6. “Deep Learning based Smart Garbage Classifier for Effective Waste Management” [References, A, 1]
   This paper tackles a classification problem. The authors classify among the four available classes which are cardboard, paper, plastic and metal. The authors do so by using a convolutional neural network. They have used a convolutional neural network with a single convolutional layer and achieved an accuracy of 76% at the 100th epoch.

4 PROBLEM IDENTIFICATIONS AND OBJECTIVES

4.1 PROBLEM STATEMENT

Every year millions of tonnes of waste are produced. A significant portion of that waste is recyclable. Recycling is the process of reusing waste for a new or similar purpose to what was used before. Out of all the types of waste, recyclable, cardboard, plastic, paper, and metal are the most produced and subsequently the most recyclable. Since this kind of waste is produced in abundance, manual segregation of this waste not only takes too long but is also harmful.

The project aims to solve the problem of segregation by using deep learning techniques. The project uses such techniques to segregate the images of cardboard, paper, plastic and metal into their respective classes as quickly, efficiently and as quickly as possible.

A dataset that consists of images of recyclable waste of classes cardboard, paper, plastic, and metal is fed into the convolutional neural network. This convolutional neural network then classifies the image into its respective class.

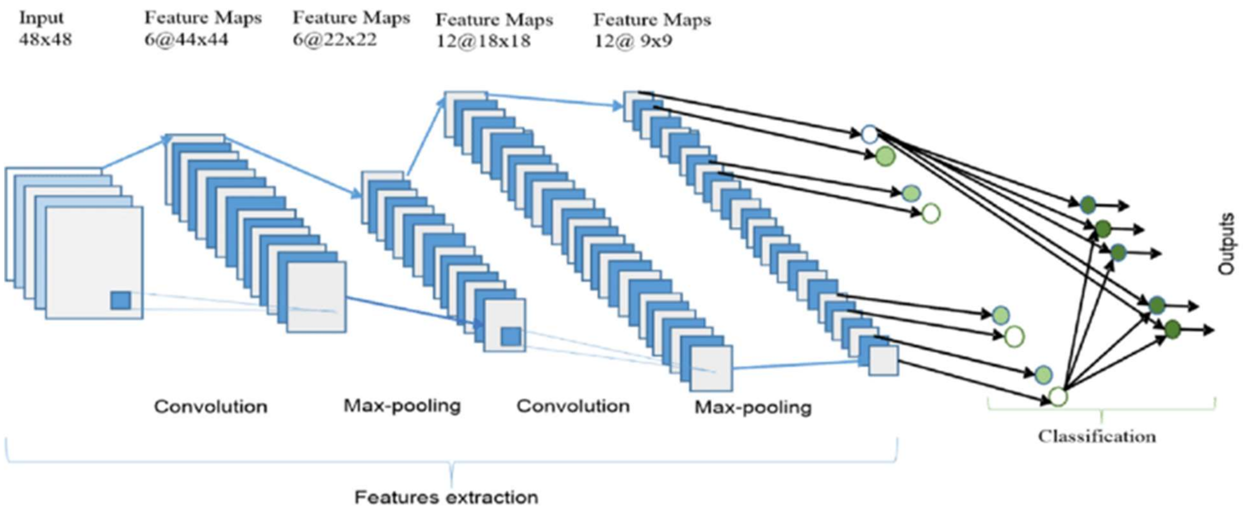
4.2 CONVOLUTIONAL NEURAL NETWORK

A convolutional neural network is a subset of deep learning. Convolutional neural networks are most generally used in computer vision problems where the deep learning model is supposed to analyse the image that has been given as an input. A Convolutional neural network almost always has what is called an input layer. The input layer's job is to take the given image as an input passed onto the hidden layer. The output layer then follows the hidden layers. Each layer in a convolutional neural network acts as an input to the next layer. In a convolutional neural network, the hidden layers perform convolutions. These convolutions involve a matrix, called a filer, to go over the image and produce a dot product and subsequently an image or a matrix. This is then generally followed by a pooling layer and an activation layer. Typically, the activation function used is ReLU.

4.2.1Convolutional layer

In a convolutional layer, an image of size height x width x channel is given. Where the height describes the height of the image, the width describes how wide the image is. Finally, channels describe the colour space the image has. For example, three here means the image has Red-Green-Blue colours in it. In this layer, the layers convolve and pass the result onto the next layer.

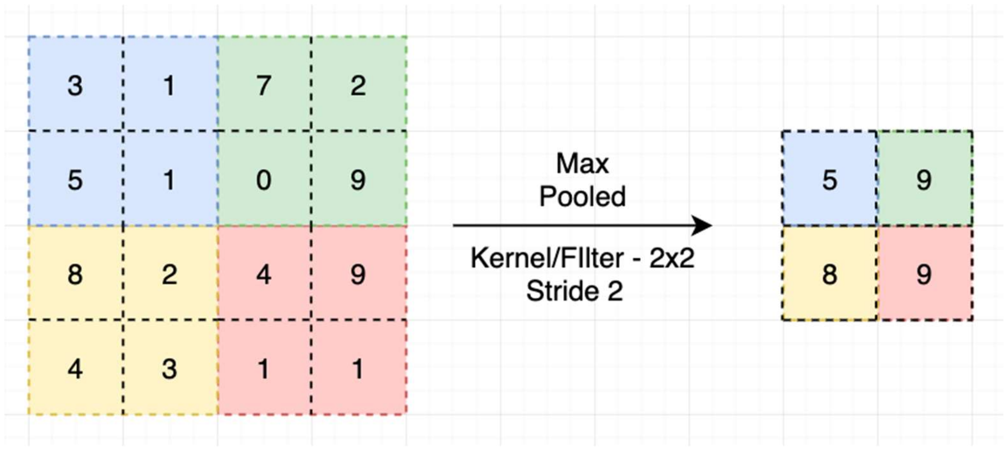
Figure 4.1 A convolutional neural network



4.2.2 Pooling layer

A convolutional neural network generally tends to have either a global or a local pooling layer. These pooling layers are there to reduce the dimensionality of the input image. Local pooling has small pooling matrices such as a 2x2 matrix which goes over the convoluted image. In the global pooling method, the entirety of the feature map is impacted upon. The two most popular kinds are max pooling and average pooling. In max pooling, the maximum value is chosen, whereas, in average pooling, the average of all of the values is given as an output.

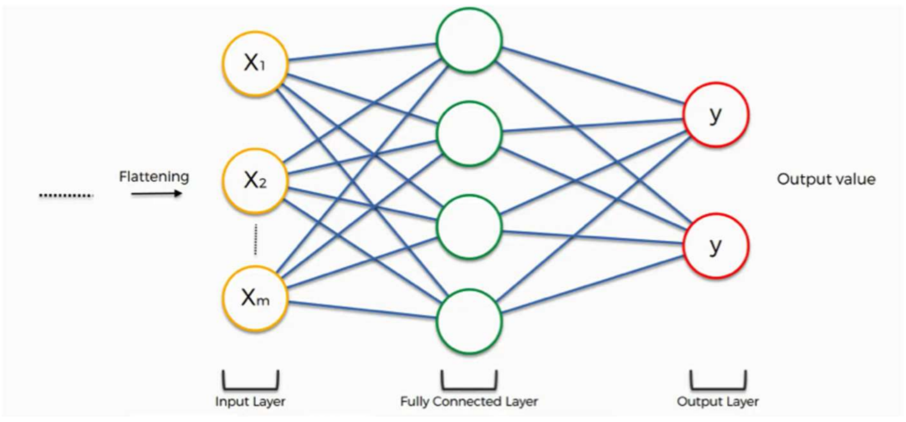
Figure 4.2 Max-Pooling



4.2.3 Fully connected layers

There are neurons in the fully connected layer, which are intertwined with every other neuron in the layer. The image that has been flattened to fit the dimension does through the layer. This layer is generally the final layer before the output layer.

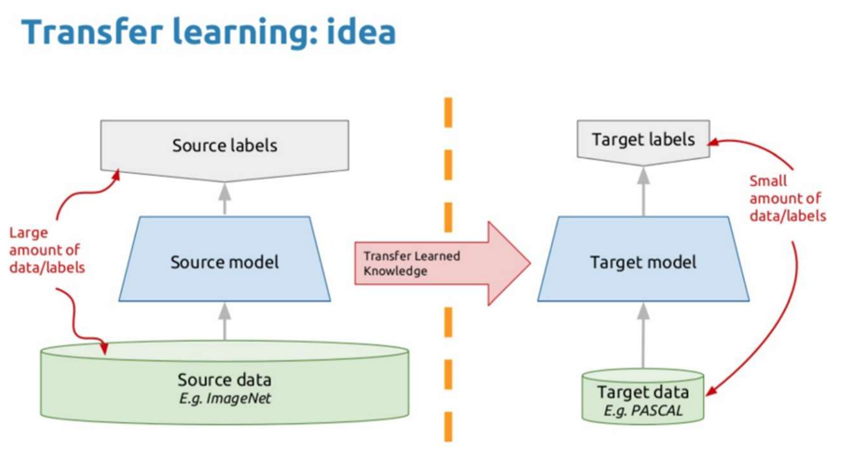
Figure 4.3 Fully connected layer



4.3 TRANSFER LEARNING

Transfer learning is a machine learning problem that is mainly concerned with research. Transfer learning primarily focuses on "knowledge" that has been gained while solving a problem, i.e. the knowledge that has been gained when solving one problem is applied to a similar but different problem, For example, the knowledge that a model gains when a convolutional neural network classifies between recyclable waste, non-recyclable waste, radioactive waste and more. The knowledge can be used to make a different convolutional neural network whose primary aim to tell if a given image is radioactive or not.

Figure 4.4 Transfer learning



4.3.1 VGG 19

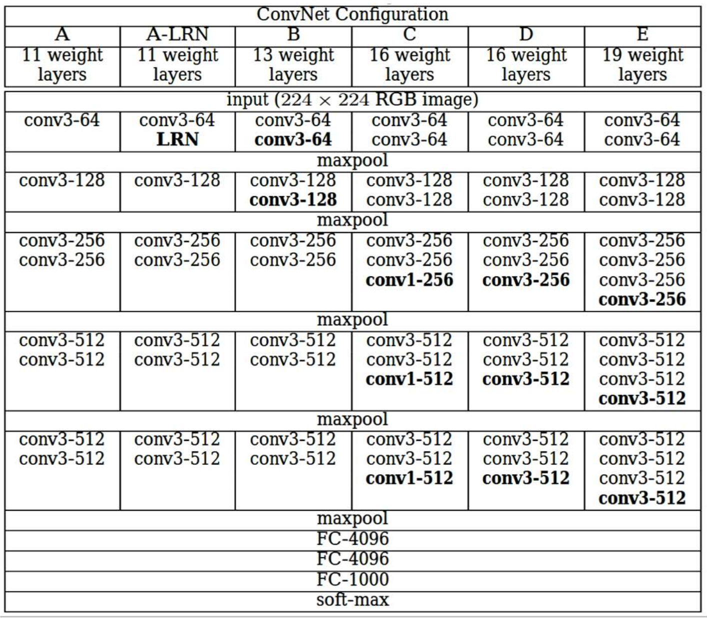
Visual geometry group, or in short, VGG, is a convolutional neural network. It is a state of an art convolutional neural network used for visual recognition. VGG is a convolutional neural network that was based on the impacts of incremental increase of the depth of the model by basically adding convolutional layers on each iteration. The convolutional neural network uses filters of size 3 x 3.

The main advantage of the model is its simplicity. The VGG 19, as the name suggests, has nineteen layers in it.

Architecture

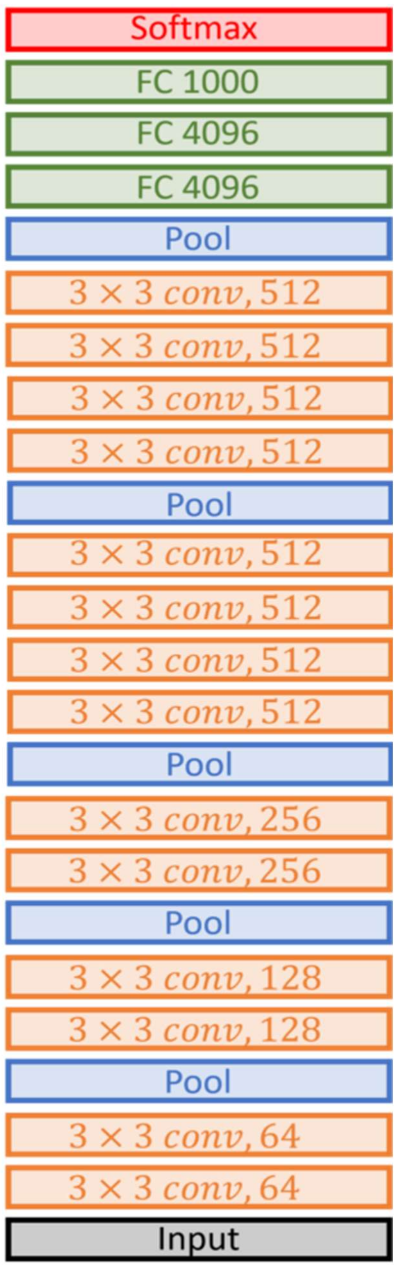
In VGG 19, the input, which is an image to the convolutional layer, is always of the size 224 x 224. Subtracting the mean value of the RGB is the only pre-processing that takes place, and that too is only applied upon the training set of each picture. Every image is passed through a convolutional layer where a filter of size 3 x 3 has a relatively small receptive field. This filter moves across the image and performs dot product. The resultant is fed forward. Some configurations in VGG 19 also make use of a convolutional filter of size 1 x 1. This causes a linear transformation of the input images. The stride of the filter is one or one pixel, where the filter moves by one row or one column each time. For every filter of size 3 x 3, padding of 1 is assigned so that a significant portion or dimension is not lost in convolution operations. Five max-pooling layers are responsible for spatial pooling. And these max-pooling layers are followed by convolutional layers. The process of max-pooling is performed with a window of size 2 x 2. The stride of this operation is 2. There are several VGG models, with each having different depths; in this case, the VGG model has 19 convolutional layers. Therefore, nineteen stacked layers of convolution are present, which are followed by three fully connected layers. Of the three fully connected layers, the first two fully connected layers have 4096 channels each. A final layer with 1000 channels follows them. The final layer has 1000 channels since VGG was trained using the ImageNet database. This database consists of 1000 classes. Finally, the last layer is the softmax layer. Evey hidden convolutional layer in VGG, regardless of its configuration, has a ReLU activation function.

Figure 4.5 Summary of VGG 19



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**conv3-64**	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4.6 VGG 19 architecture



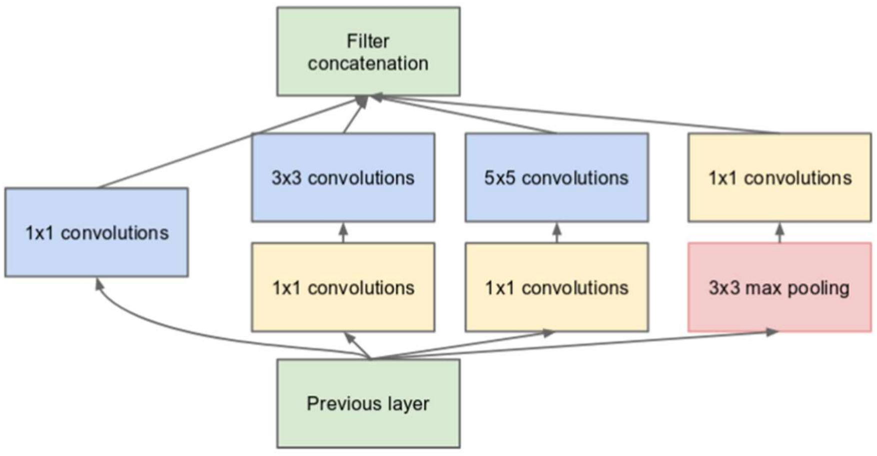
4.3.2 INCEPTION V3

Inception is a convolutional neural network. Inception is used for object detection, visual analysis. Inception V3 as the name suggests, in the third iteration of the model. Google has primarily made the inception model. It was just like visual geometric group 19 was introduced in the ImageNet recognition challenge. ImageNet is a database of images that consists of a thousand classes.

4.3.2.1 Architecture

An image can have exceedingly different variations on how much the intended subject occupies the image. Therefore, Inception has kernels of multiple sizes. Appropriate kernels are used, and their appropriateness depends upon the size of the subject in the image. A large kernel is generally preferred where the information is larger, and a smaller kernel is used where the information in an image is more local. Inception is also "wider" instead of "deeper", as the convolutional neural networks, which are the latter, tend to overfit and are also computationally expensive. The different kernel sizes are 1 x 1, 3 x 3 and 5 x 5. A max-pooling is also added to every inception module. Furthermore, the input channels of the convolutions have been limited using a preliminary 1 x 1 kernel to make the process of convolution computationally less expensive.

Figure 4.7 Inception module with dimension reduction

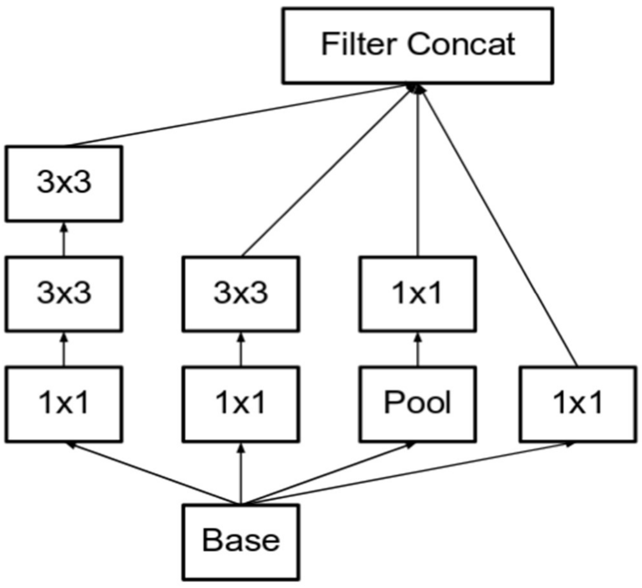


Additionally, it is pretty apparent from the architecture of inception that it is a deep model. Therefore, to prevent the middle section of the form dying out, a couple of auxiliary classifiers have been added. The loss incurred from these is added to the final loss value of the model.

In the subsequent Inception model, to alleviate the bottlenecks caused by reduced dimension, inception v2 maintained the image's dimensions. Following are premier features of inceptionv2 and inceptionv3:

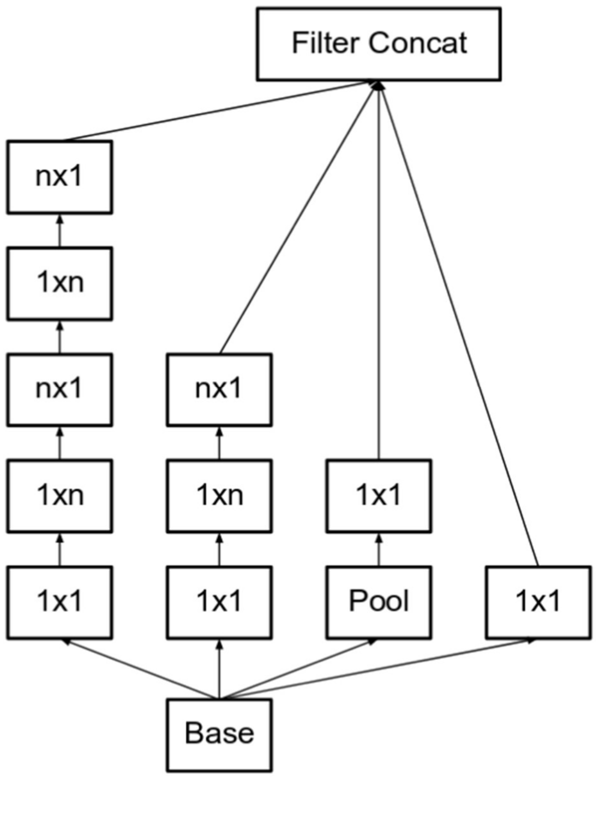
Factorisation: Factorised a 5 x 5 convolution in a couple of 3 x 3 sized convolutions. This is done because a singly 5 x 5 convolution is almost thrice as computationally expensive as a 3 x 3 convolution.

Figure 4.8 Factorisation



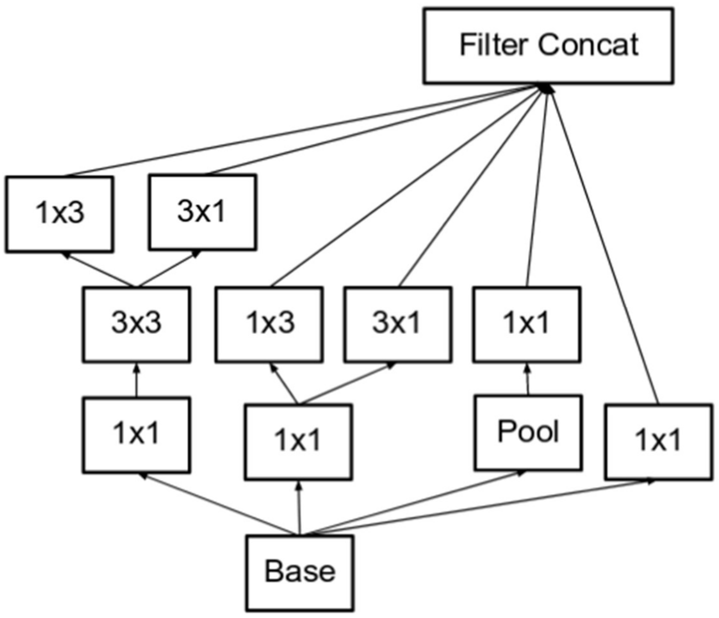
Asymmetric convolutions: A n x n convolution is replaced by 2 convolutions of size 1 x n and n x1. For example, a 3 x 3 convolution is broken up into two convolutions of size 1 x 3 and 3 x 1. This process is done so since it's approximately 33% cheaper.

Figure 4.9 Asymmetric convolution



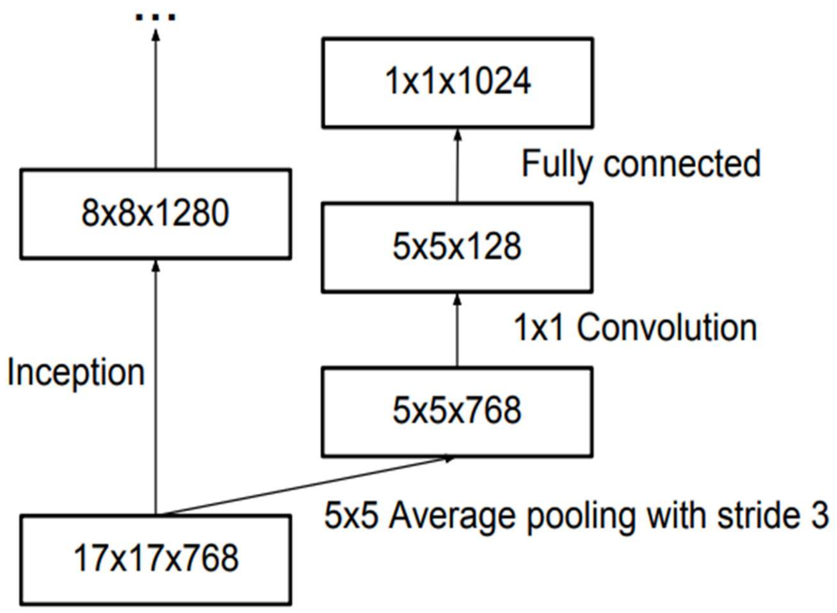
Filter banks: they have been expanded and have been changed from being deeper to being much wider. This is done to remove the representational bottlenecks.

Figure 4.10 filter bank



Auxiliary classifier: a CNN has been added with batch normalisation in the middle of inception. This is to aid loss.

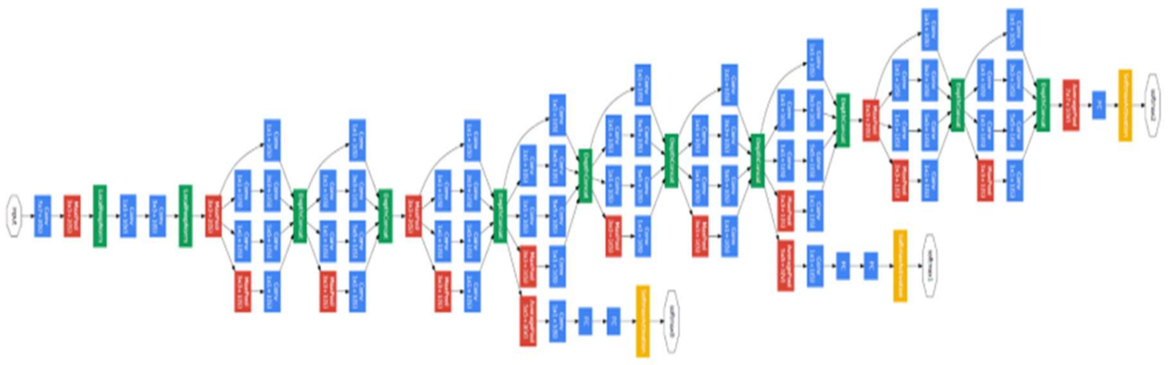
Figure 4.11 Auxiliary classifier



Some of the other features in Inception V3 include:

- Root Mean Square Propagation optimiser.
- 7x7 filters have also been factorised.
- Batch Normalisation has been applied to the Auxiliary classifiers.
- Finally, Label smoothing is applied.

Figure 4.12 Inception architecture



4.4 DATASET

A Dataset is a set of related information which constitutes different elements but can be defined as a single unit using a computer. A Dataset can be in multiple forms.

The Dataset that we are using here in this project is classified into 4 different classes and consists of the following number of images:

Table 4.1 Dataset

Class	**Number of images**
Cardboard	1172
Metal	1248
Paper	1116
Plastic	1071

The dataset is an amalgamation of several datasets, all obtained from different places. The DataSets we have gathered are from the following links provided below:

1)https://www.kaggle.com/asdasdasasdas/garbage-classificatio

2)https://www.kaggle.com/wangziang/waste-pictures

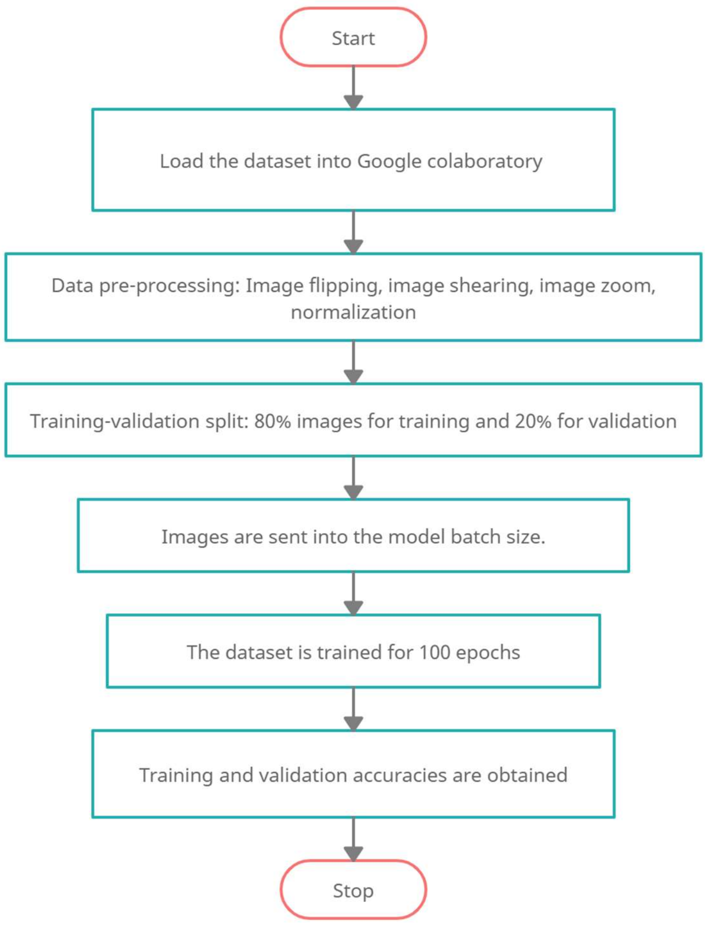
3)https://www.kaggle.com/szdxfkmgnb/waste-classification

4)Google images

5 SYSTEM METHODOLOGY

5.1 VANILLA CNN

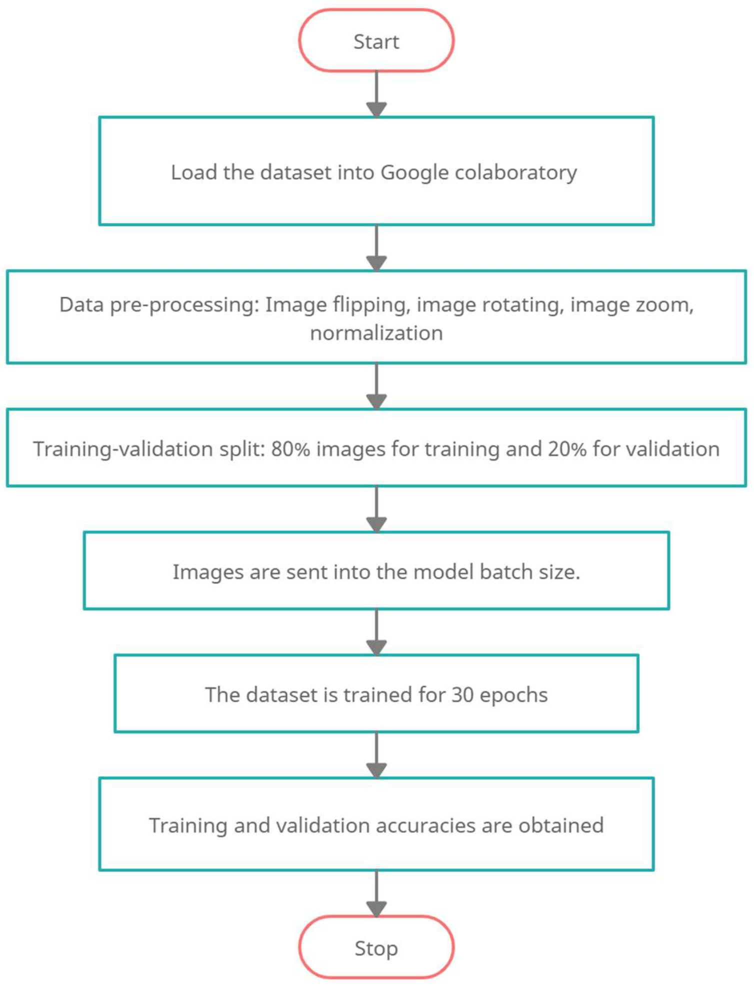
Figure 5.1 CNN workflow


Start
Load the dataset into Google colaboratory
Data pre-processing: Image flipping, image shearing, image zoom, normalization
Training-validation split: 80% images for training and 20% for validation
Images are sent into the model batch size.
The dataset is trained for 100 epochs
Training and validation accuracies are obtained
Stop

1. The dataset is loaded into the Google Colab. The dataset is stored in Google Drive.
2. The images are broken up into 128 sets. This means that the batch size is 128.
3. The entirety of the dataset is divided into two halves. These are training and validation sets. 80% of the images are used for training, whereas the other 20% is used for validation.
4. Images of size 100 x 100 x 3 are fed to the algorithm.
5. Then, every image is pre-processed before entering the model. These pre-processings are image flip, image shear, image zoom and normalisation.
6. The convolutional neural network is trained for 100 epochs.
7. Penultimately, the training and validation accuracies are obtained.
8. Finally, a graph comparing training and validation are drawn using matplotlib.

5.2 VGG 19

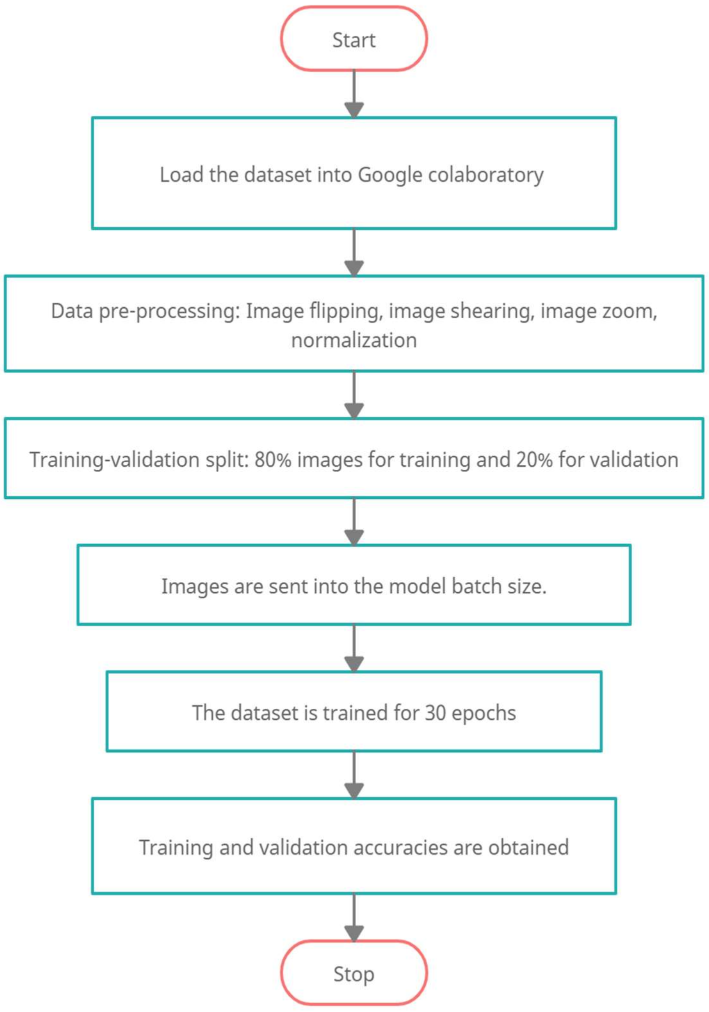
Figure 5.2 VGG 19 workflow


Start
Load the dataset into Google colaboratory
Data pre-processing: Image flipping, image rotating, image zoom, normalization
Training-validation split: 80% images for training and 20% for validation
Images are sent into the model batch size.
The dataset is trained for 30 epochs
Training and validation accuracies are obtained
Stop

1. The dataset is loaded into the Google Colab. The dataset is stored in Google Drive.
2. The images are broken up into 128 sets. This means that the batch size is 128.
3. The entirety of the dataset is divided into two halves. These are training and validation sets. 80% of the images are used for training, whereas the other 20% is used for validation.
4. Images of size 224 x 224 x 3 are fed to the algorithm.
5. Then, every image is pre-processed before entering the model. These pre-processings are image flip, image shear, image zoom and normalisation.
6. The convolutional neural network is trained for 30 epochs.
7. Penultimately, the training and validation accuracies are obtained.
8. Finally, a graph comparing training and validation are drawn using matplotlib.

5.3 INCEPTION V3

Figure 5.3 Inception V3 workflow


Start
Load the dataset into Google colaboratory
Data pre-processing: Image flipping, image shearing, image zoom, normalization
Training-validation split: 80% images for training and 20% for validation
Images are sent into the model batch size.
The dataset is trained for 30 epochs
Training and validation accuracies are obtained
Stop

1. The dataset is loaded into the Google Colab. The dataset is stored in Google Drive.
2. The images are broken up into 128 sets. This means that the batch size is 128.
3. The entirety of the dataset is divided into two halves. These are training and validation sets. 80% of the images are used for training, whereas the other 20% is used for validation.
4. Images of size 100 x 100 x 3 are fed to the algorithm.
5. Then, every image is pre-processed before entering the model. These pre-processings are image flip, image shear, image zoom and normalisation.
6. The convolutional neural network is trained for 100 epochs.
7. Penultimately, the training and validation accuracies are obtained.
8. Finally, a graph comparing training and validation are drawn using matplotlib.

6 OVERVIEW OF TECHNOLOGIES

6.1 GOOGLE COLABORATORY

Google Colaboratory, also known as Colab is an environment that runs on the cloud and stores the notebooks on Google Drive. Colab is a free to use Jupyter notebook environment. Google ‘Colab’ or ‘Colaboratory’ is a platform that enables anyone and everyone to write as well as execute any arbitrary python codes in the browser without the need of installing any other third-party software. This feature of in-browser coding makes the work hassle-free and straightforward. Google ‘Colab’ or ‘Colaboratory’ is useful especially for data analysis, machine learning and education.

6.2 GOOGLE DRIVE

Google has developed Google Drive. This service enables data storage or file storage of any format (Music, Videos, Pictures, Documents, etc.). Google Drive allows synchronisation among different devices activated by the same Google account as the Google Drive account. Google Drive is also cloud-based storage. Google Drive provides all the services “free” of cost.

6.3 TENSORFLOW

TensorFlow is an open-source machine learning software that runs from start to finish. Consider it a layer of a framework for discrete programming. It incorporates essential abilities. It is executing low-level tensor operations efficiently on CPU, GPU, or TPU.

It is calculating the gradient of every differentiable expression. Google Brain's second-generation system is nothing but TensorFlow. Its adaptable architecture enables simple computing implementation across various platforms (CPU, GPU, TPU), from desktops to clusters of servers to smartphone and edge computers.

6.4 KERAS

Keras is one of the most common advanced neural network APIs(Application Programming Interface). It's written in Python and works for a variety of back-end neural network computing engines. Keras adheres to best practices for mitigating cognitive burden by providing reliable and straightforward APIs, minimising the number of user activities needed for particular use cases, and providing direct and actionable input in the event of user error.

Neural layers, cost functions, optimisers, initialisation schemes, activation functions, and regularisation schemes are all independent components that can be combined to generate new versions. New modules, as well as new groups and features, are easy to install. Models are specified in Python code rather than in separate model configuration directories.

The main reasons to use Keras stem from its guiding principles, especially the one regarding user-friendliness. Aside from ease of practice and model building, Keras has the benefits of growing availability, support for a wide variety of production implementation options, and alignment with at least five back-end engines.

6.5 MATPLOTLIB

Matplotlib is a plotting and visualisation library, which has been developed for Python. It is considered to be one of the most popular tools for making graphs and analysing data. It is mainly used to plot snippets of code. It is used mainly for plotting complex graphs using simple lines of code. Using this, one can generate bar plots, histograms, scatterplots etc. Matplotlib has many modules which prove to be quite essential. Pyplot is one of the most popular ones of them. Pyplot offers an interface that is similar to MATLAB’s graphic module.

7 IMPLEMENTATION

7.1 IMPORTANT FILES AND FUNCTIONS

7.1.1 TensorFlow

TensorFlow is an open-source machine learning software that runs from start to finish. Consider it a layer of a framework for discrete programming. It incorporates essential abilities. It is executing low-level tensor operations efficiently on CPU, GPU, or TPU.

It is calculating the gradient of every differentiable expression. Google Brain's second-generation system is nothing but TensorFlow. Its adaptable architecture enables simple computing implementation across various platforms (CPU, GPU, TPU), from desktops to clusters of servers to smartphone and edge computers.

7.1.2 Keras

Keras is one of the most common advanced neural network APIs(Application Programming Interface). It's written in Python and works for a variety of back-end neural network computing engines. Keras adheres to best practices for mitigating cognitive burden by providing reliable and straightforward APIs, minimising the number of user activities needed for particular use cases, and providing direct and actionable input in the event of user error.

Neural layers, cost functions, optimisers, initialisation schemes, activation functions, and regularisation schemes are all independent components that can be combined to generate new versions. New modules, as well as new groups and features, are easy to install. Models are specified in Python code rather than in separate model configuration directories.

The main reasons to use Keras stem from its guiding principles, especially the one regarding user-friendliness. Aside from ease of practice and model building, Keras has the benefits of growing availability, support for a wide variety of production implementation options, and alignment with at least five back-end engines.

7.1.3 Image Data Generator

Image data generator is a pre-built function/class within Keras. Image data generator is a function that can be used for training validation split. It is also used primarily for image augmentation of image pre-processing. Some data augmentation techniques relate to brightness, zoom, flips, rotations, normalisation etc.

7.1.4 Conv2d

Keras Conv2D is a two-dimensional convolutional layer. A convolutional layer applies filters upon the image, which receives as input and subsequently produces a matrix that undergoes dot product with the kernel.

7.1.5 MaxPooling2D

MaxPooling2D is an operation that is present after a regular convolutional layer. In a model, max-pooling is a matrix that goes over the output of the convolutional layer and produces the highest valued pixel value.

7.1.6 Dropout

Dropout is an operation that is present after a regular convolutional layer. It works on a probabilistic model by removing the connections between the layers. This process is generally used to tackle overfitting in a model. Dropout forces the model to "unlearn" some of the information learned in the previous model.

7.1.7 Pyplot

It is a Matplotlib module that aims to give an interface similar to that of MATLAB. It is made to take advantage of python. Pyplot is used to create figures, creating a plotting area in a fig etc. Various decorations can also be made using Pyplot.

7.1.8 ImageNet

ImageNet is a database of images. ImageNet has millions of images in it. ImageNet is used to train several state-of-the-art CNN algorithms. Weights trained that have been achieved using ImageNet can be used as transfer learning in a different domain.

7.1.9 Adam

Adam is an algorithm that is used to optimise a machine learning algorithm. The advantage of Adam is that it uses a variable learning rate to reach global minima.

7.1.10 Categorical Cross entropy

Categorical cross-entropy is a loss function. The job of a loss function is to judge how accurately an algorithm is performing. Categorical cross-entropy is used here since it is a multi-classification problem. The loss function has to accurately punish and reward the algorithm for incorrect and correct classification problems, respectively.

$$H(p,q) = -\sum_{\forall x P} p(x)\log(q(x))$$

p(x) is the probability of the true value and q(x) is the probability of the prediction.

7.1.11 OS

This is a module that gives a way of using operating systems' functionality. It has functionalities such as open(), read() etc.

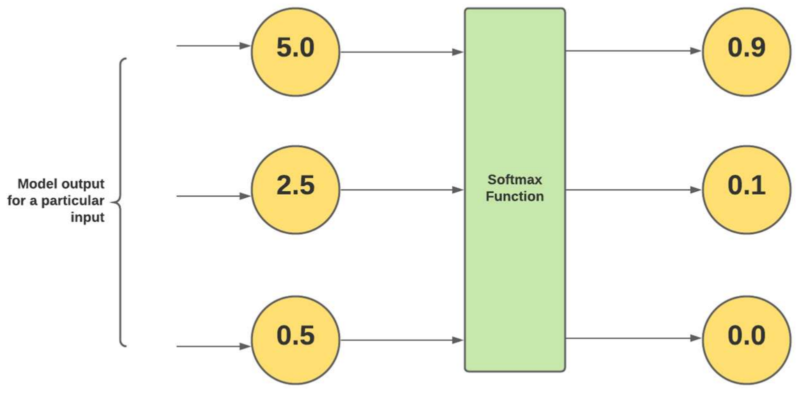
7.1.12 Numpy

Numpy is an array processing package whose main attributes are it being general purpose.Numpy array helps with bringing forth high performance with respect to arrays with multiple and single dimensions. It is also one of the most commonly used packages in python. Numpy is also a container that contains generic data of multiple dimensions.

7.1.13 Softmax

It is the most commonly used last layer in a CNN model. It changes the n-dimensional matrix and fits into a valid probabilistic distribution.

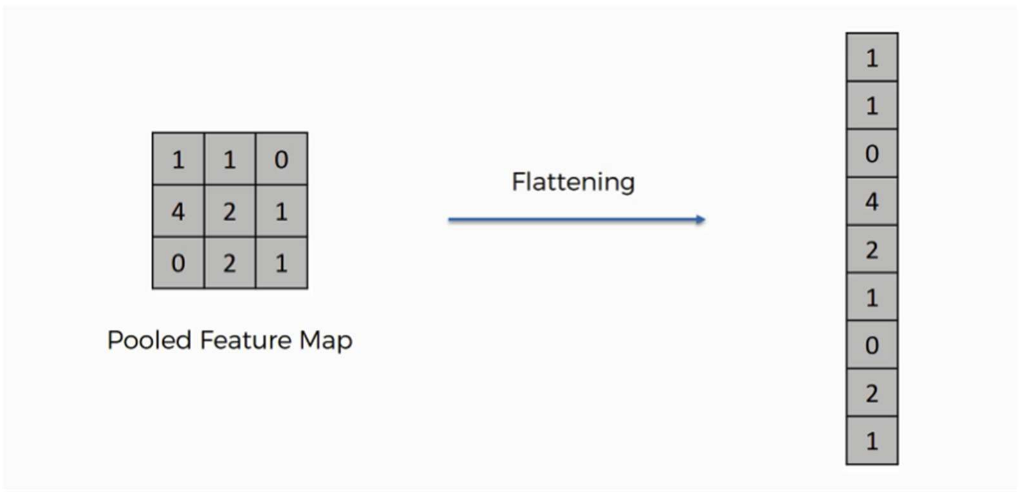
Figure 7.1 Softmax working



7.1.14 Flatten

Flatten is used between a convolutional layer and a fully connected layer. The flatten function is used because the output of a convolutional layer is in the form of an n x n matrix, and a fully connected layer is not fit to accept such an input.

Figure 7.2 Flatten

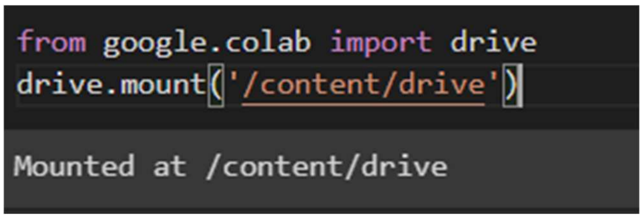


7.2 CODING

7.2.1 Importing the dataset into Google Colab

The dataset was imported into Google Colab using the following code:

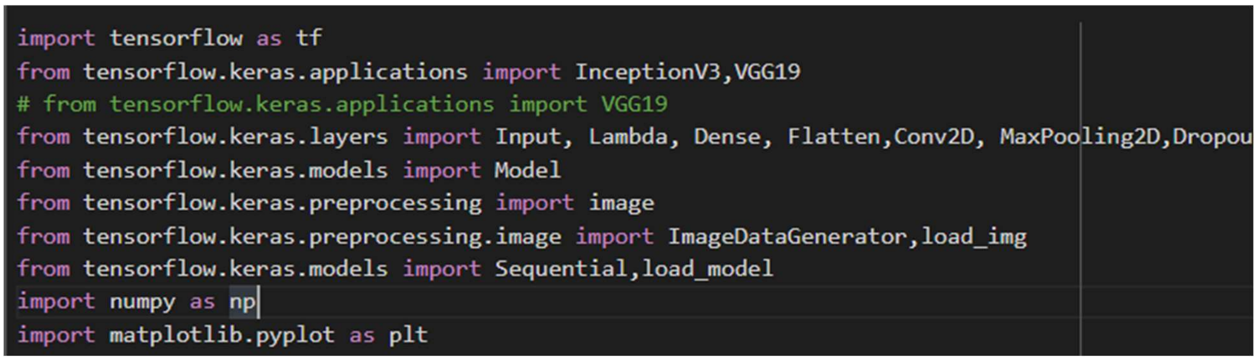
Screen 7.1 Import dataset



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

Mounted at /content/drive
```

7.2.2 Convolutional neural network implementation

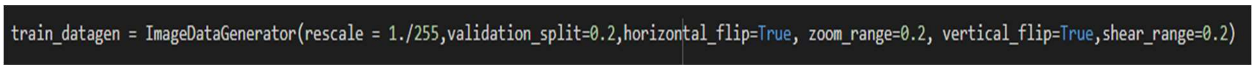
Screen 7.2 Importing libraries



```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3,VGG19
# from tensorflow.keras.applications import VGG19
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten,Conv2D, MaxPooling2D,Dropou
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential,load_model
import numpy as np
import matplotlib.pyplot as plt
```

The above-displayed code is in python language—the code snippet imports all the required libraries, models etc., for the execution of the model.

Screen 7.3 Image data generator

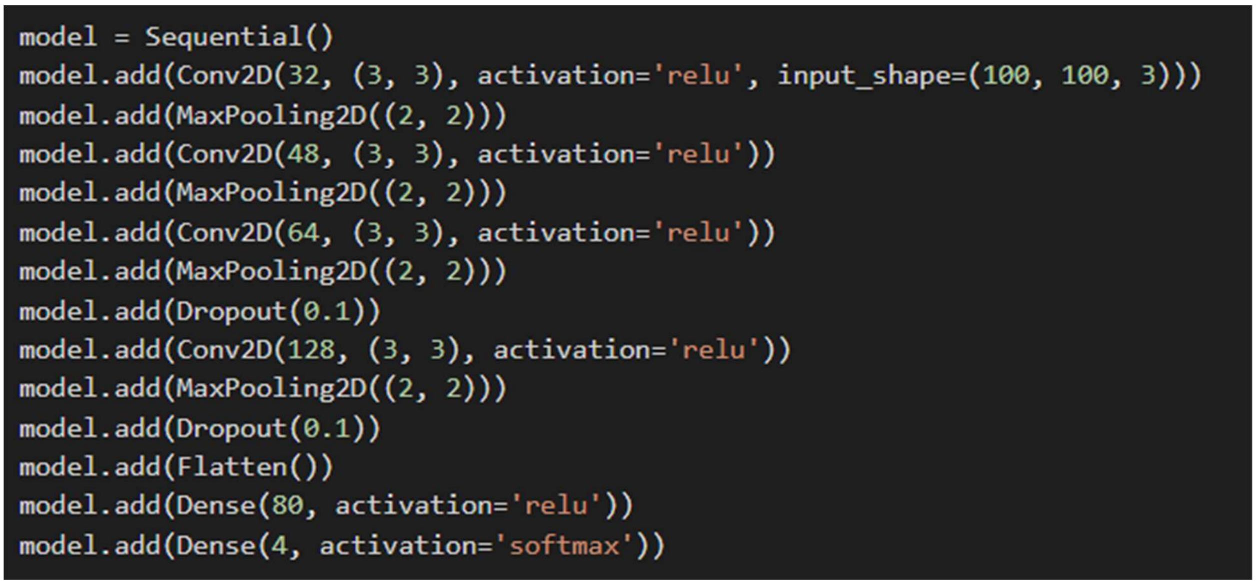


```
train_datagen = ImageDataGenerator(rescale = 1./255,validation_split=0.2,horizontal_flip=True, zoom_range=0.2, vertical_flip=True,shear_range=0.2)
```

Image data generator: Image data generator is a pre-built function/class within Keras. Image data generator is a function that can be used for training validation split. It is also used primarily for image augmentation of image pre-processing. Some data augmentation techniques relate to brightness, zoom, flips, rotations, normalisation etc. The above code snippet does the following:

- rescale = 1./255: Normalising the pixel values between 0-1.
- validation_split = 0.2: Setting the training and validation testing is 0.2. It means that 80% of images are used for training, and the other 20% of images are used for validation.
- horizontal_flip = True: Flipping the training image horizontally.
- zoom_range = 0.2: Zooming the training image by 20%.
- vertical_flip = True: Flipping the training image vertically.
- shear_range = 0.2: Shearing the image by 20%.

Screen 7.4 CNN model

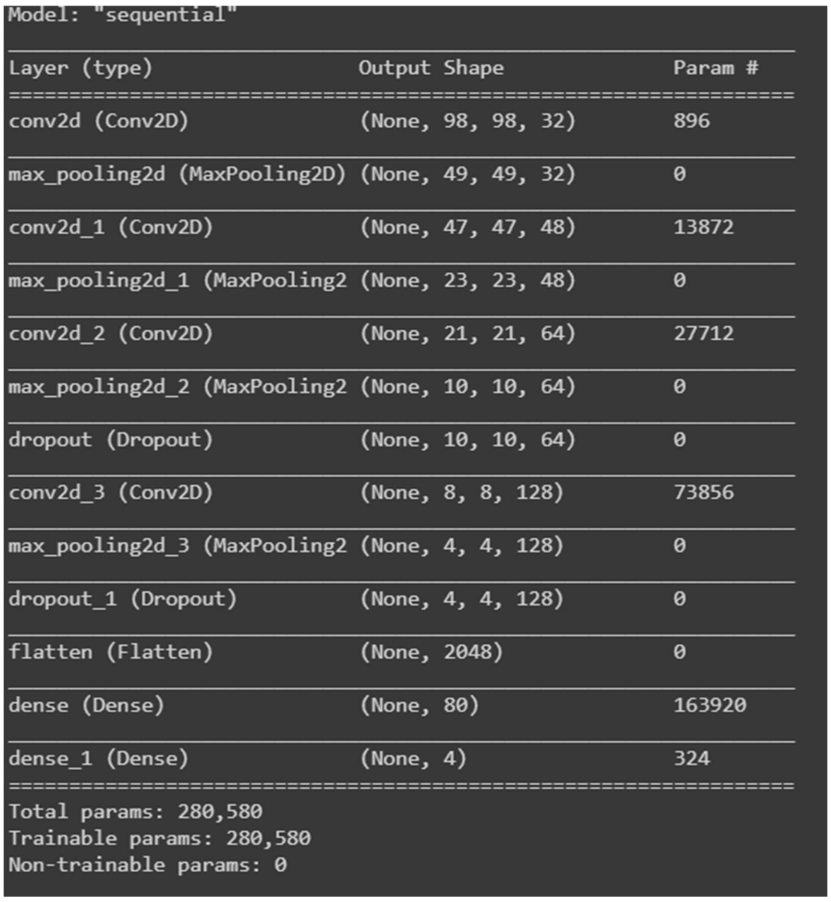


```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(48, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(80, activation='relu'))
model.add(Dense(4, activation='softmax'))
```

The above Python code snippet depicts the structure of the Convolutional neural network. This model is a sequential model. The following are the key characteristics of the model:

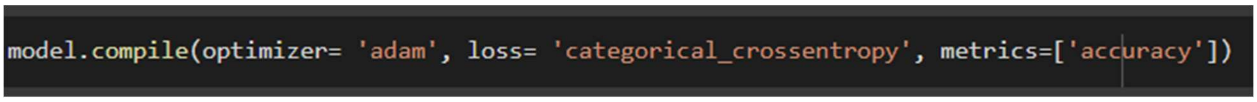
- This convolutional has four convolutional layers.
- The fully connected layer consists of 80 channels.
- The input shape is 100 x 100 x 3.
- There are 32 kernels or filters in the first convolutional layer. The second convolutional layer has 48 kernels. The third convolutional layer has 64 kernels, and finally, the fourth convolutional layer has 128 kernels.
- Each kernel is of size 3 x 3.
- The stride of the kernel is the default value used by Keras, which is 1.
- Max pooling is applied to every convolutional layer. The max-pooling is of size 2 x 2, where the stride is 2.
- The model uses Dropout layers after the third and the fourth convolutional layer.
- 10% of “deactivated”. This is done via a probabilistic function.
- The final layer is the softmax layer. It consists of 4 neurons since the model has to classify images which are divided into four classes.

Screen 7.5 CNN summary



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 48)        13872
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 48)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 64)        27712
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 80)                163920
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 324
=================================================================
Total params: 280,580
Trainable params: 280,580
Non-trainable params: 0
```

Screen 7.6 Compiling

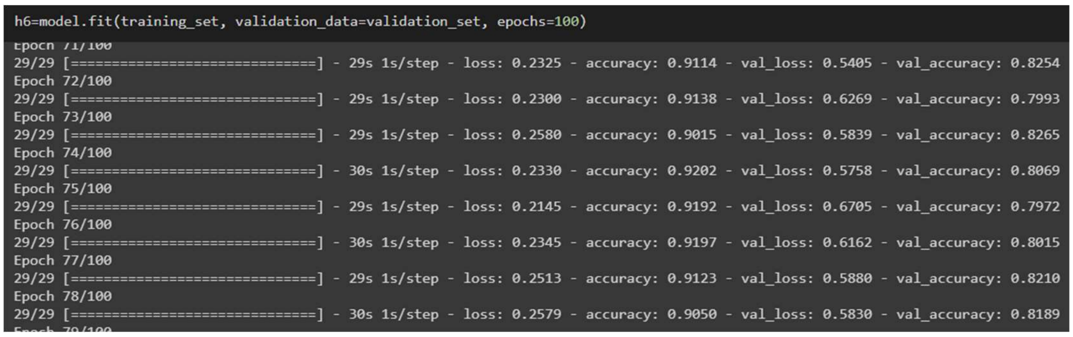


```
model.compile(optimizer= 'adam', loss= 'categorical_crossentropy', metrics=['accuracy'])
```

The above code depicts the following:

- The optimiser used by this model is Adam.
- The loss function used by this model is categorical cross-entropy.
- The metrics for evaluation of the performance of the model is its accuracy.

Screen 7.7 Execution

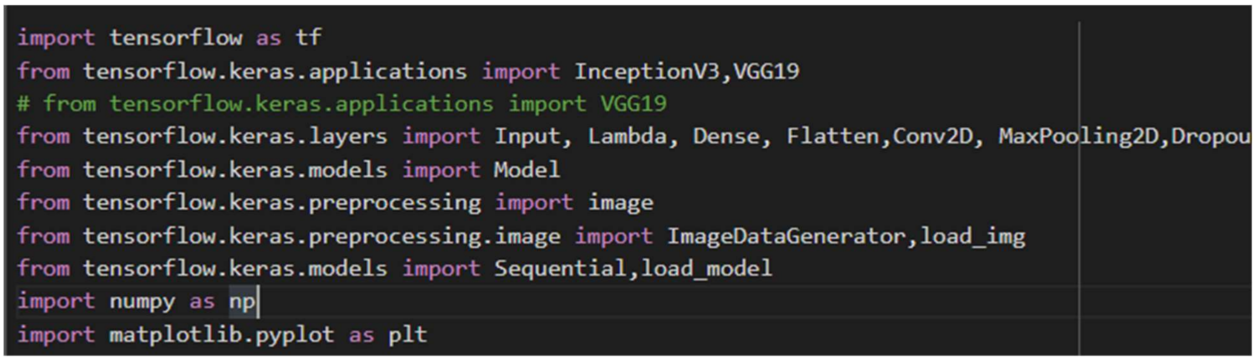


```
h6=model.fit(training_set, validation_data=validation_set, epochs=100)
Epoch 71/100
29/29 [==============================] - 29s 1s/step - loss: 0.2325 - accuracy: 0.9114 - val_loss: 0.5405 - val_accuracy: 0.8254
Epoch 72/100
29/29 [==============================] - 29s 1s/step - loss: 0.2300 - accuracy: 0.9138 - val_loss: 0.6269 - val_accuracy: 0.7993
Epoch 73/100
29/29 [==============================] - 29s 1s/step - loss: 0.2580 - accuracy: 0.9015 - val_loss: 0.5839 - val_accuracy: 0.8265
Epoch 74/100
29/29 [==============================] - 30s 1s/step - loss: 0.2330 - accuracy: 0.9202 - val_loss: 0.5758 - val_accuracy: 0.8069
Epoch 75/100
29/29 [==============================] - 29s 1s/step - loss: 0.2145 - accuracy: 0.9192 - val_loss: 0.6705 - val_accuracy: 0.7972
Epoch 76/100
29/29 [==============================] - 30s 1s/step - loss: 0.2345 - accuracy: 0.9197 - val_loss: 0.6162 - val_accuracy: 0.8015
Epoch 77/100
29/29 [==============================] - 29s 1s/step - loss: 0.2513 - accuracy: 0.9123 - val_loss: 0.5880 - val_accuracy: 0.8210
Epoch 78/100
29/29 [==============================] - 30s 1s/step - loss: 0.2579 - accuracy: 0.9050 - val_loss: 0.5830 - val_accuracy: 0.8189
```

The above python depicts the execution of the CNN model. The VGG 19 model is run for 100 epochs.

7.2.3 VGG 19

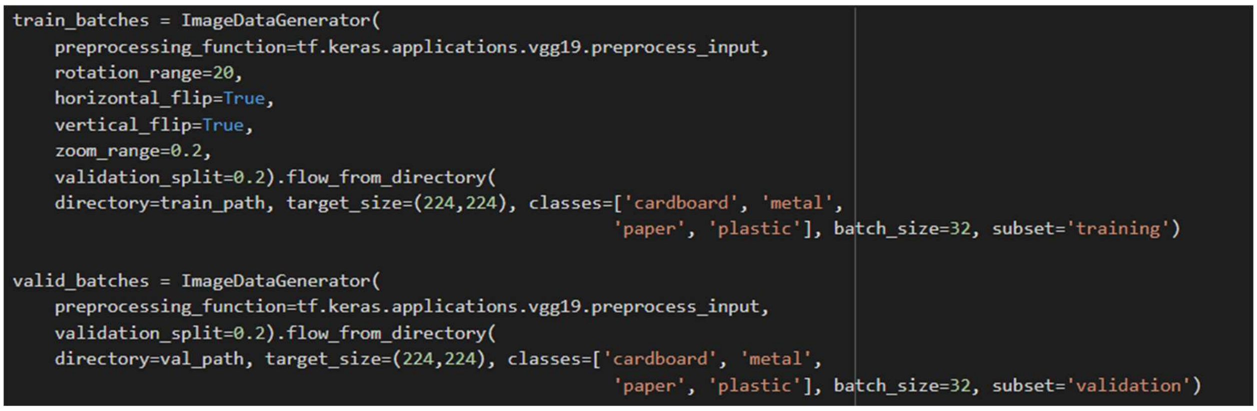
Screen 7.8 Importing libraries



```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3,VGG19
# from tensorflow.keras.applications import VGG19
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten,Conv2D, MaxPooling2D,Dropou
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential,load_model
import numpy as np
import matplotlib.pyplot as plt
```

The above-displayed code is in python language—the code snippet imports all the required libraries, models, and others to execute the model.

Screen 7.9 Image data generator

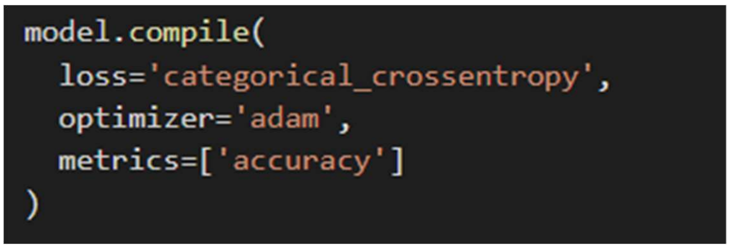


```
train_batches = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.vgg19.preprocess_input,
    rotation_range=20,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=0.2,
    validation_split=0.2).flow_from_directory(
    directory=train_path, target_size=(224,224), classes=['cardboard', 'metal',
                                                          'paper', 'plastic'], batch_size=32, subset='training')

valid_batches = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.vgg19.preprocess_input,
    validation_split=0.2).flow_from_directory(
    directory=val_path, target_size=(224,224), classes=['cardboard', 'metal',
                                                        'paper', 'plastic'], batch_size=32, subset='validation')
```

Image data generator: Image data generator is a pre-built function/class within Keras. Image data generator is a function that can be used for training validation split. It is also used primarily for image augmentation of image pre-processing. Some data augmentation techniques relate to brightness, zoom, flips, rotations, normalisation etc. The above code snippet does the following:

- rescale = 1./255: Normalising the pixel values between 0-1.
- validation_split = 0.2: Setting the training and validation testing is 0.2. It means that 80% of images are used for training, and the other 20% of images are used for validation.
- horizontal_flip = True: Flipping the training image horizontally.
- zoom_range = 0.2: Zooming the training image by 20%.
- vertical_flip = True: Flipping the training image vertically.
- rotation_range = 20: Rotating the image by 20%.

Screen 7.10 Compile

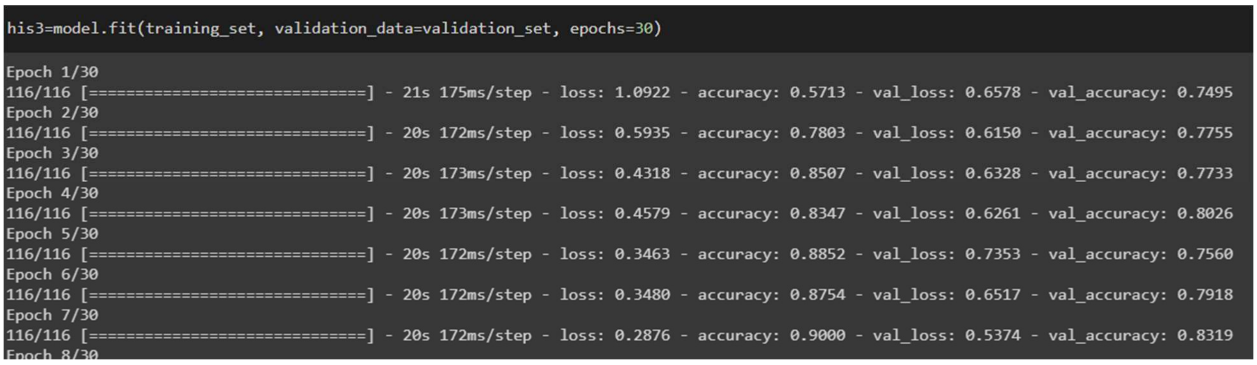


```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

The above code depicts the following:

- The optimiser used by this model is Adam.
- The loss function used by this model is categorical cross-entropy.
- The metrics for evaluation of the performance of the model is its accuracy.

Screen 7.11 Execution

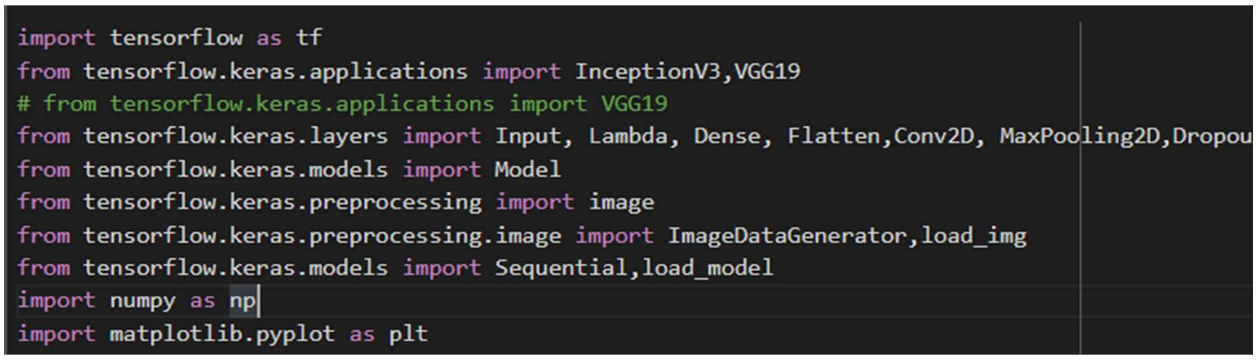


```
his3=model.fit(training_set, validation_data=validation_set, epochs=30)

Epoch 1/30
116/116 [==============================] - 21s 175ms/step - loss: 1.0922 - accuracy: 0.5713 - val_loss: 0.6578 - val_accuracy: 0.7495
Epoch 2/30
116/116 [==============================] - 20s 172ms/step - loss: 0.5935 - accuracy: 0.7803 - val_loss: 0.6150 - val_accuracy: 0.7755
Epoch 3/30
116/116 [==============================] - 20s 173ms/step - loss: 0.4318 - accuracy: 0.8507 - val_loss: 0.6328 - val_accuracy: 0.7733
Epoch 4/30
116/116 [==============================] - 20s 173ms/step - loss: 0.4579 - accuracy: 0.8347 - val_loss: 0.6261 - val_accuracy: 0.8026
Epoch 5/30
116/116 [==============================] - 20s 172ms/step - loss: 0.3463 - accuracy: 0.8852 - val_loss: 0.7353 - val_accuracy: 0.7560
Epoch 6/30
116/116 [==============================] - 20s 172ms/step - loss: 0.3480 - accuracy: 0.8754 - val_loss: 0.6517 - val_accuracy: 0.7918
Epoch 7/30
116/116 [==============================] - 20s 172ms/step - loss: 0.2876 - accuracy: 0.9000 - val_loss: 0.5374 - val_accuracy: 0.8319
Epoch 8/30
```

The above python depicts the execution of the CNN model. The VGG 19 model is run for 30 epochs.

7.2.4 Inception V3

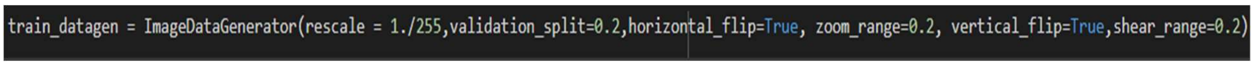
Screen 7.12 importing libraries



```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3,VGG19
# from tensorflow.keras.applications import VGG19
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten,Conv2D, MaxPooling2D,Dropou
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential,load_model
import numpy as np
import matplotlib.pyplot as plt
```

The above-displayed code is in python language—the code snippet imports all the required libraries, models etc., for the execution of the model.

Screen 7.13 Image data generator

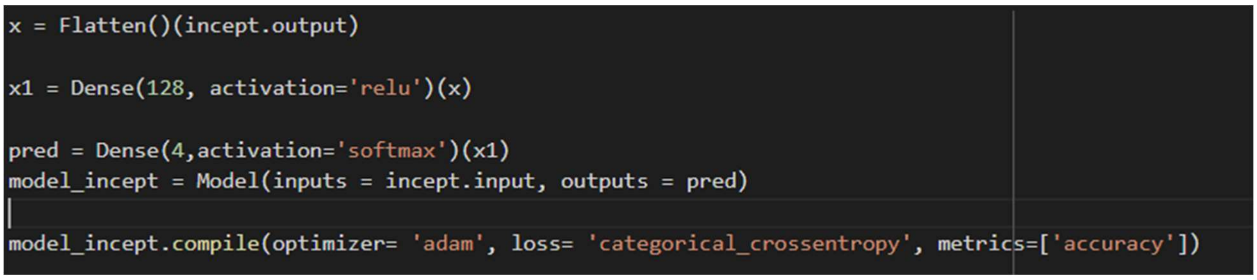


```
train_datagen = ImageDataGenerator(rescale = 1./255,validation_split=0.2,horizontal_flip=True, zoom_range=0.2, vertical_flip=True,shear_range=0.2)
```

Image data generator: Image data generator is a pre-built function/class within Keras. Image data generator is a function that can be used for training validation split. It is also used primarily for image augmentation of image pre-processing. Some data augmentation techniques relate to brightness, zoom, flips, rotations, normalisation etc. The above code snippet does the following:

- rescale = 1./255: Normalising the pixel values between 0-1.
- validation_split = 0.2: Setting the training and validation testing is 0.2. It means that 80% of images are used for training, and the other 20% of images are used for validation.
- horizontal_flip = True: Flipping the training image horizontally.
- zoom_range = 0.2: Zooming the training image by 20%.
- vertical_flip = True: Flipping the training image vertically.
- shear_range = 0.2: Shearing the image by 20%.

Screen 7.14 Compile

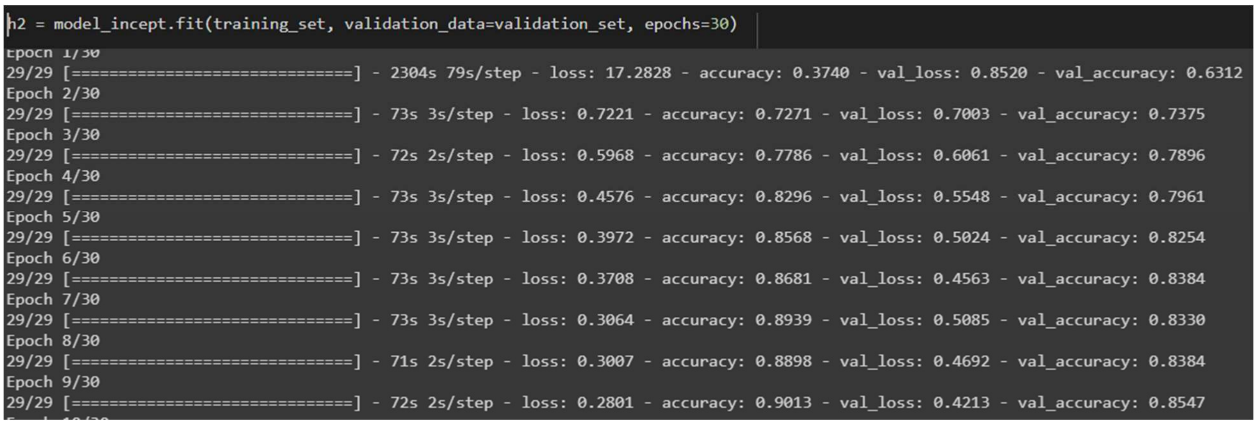


```
x = Flatten()(incept.output)

x1 = Dense(128, activation='relu')(x)

pred = Dense(4,activation='softmax')(x1)
model_incept = Model(inputs = incept.input, outputs = pred)

model_incept.compile(optimizer= 'adam', loss= 'categorical_crossentropy', metrics=['accuracy'])
```

The above code depicts the following:

- The optimiser used by this model is Adam.
- The loss function used by this model is categorical cross-entropy.
- The metrics for evaluation of the performance of the model is its accuracy.
- The activation function used in the final layer of Inception V3 is Softmax.

Screen 7.15 Execution



```
h2 = model_incept.fit(training_set, validation_data=validation_set, epochs=30)
Epoch 1/30
29/29 [==============================] - 2304s 79s/step - loss: 17.2828 - accuracy: 0.3740 - val_loss: 0.8520 - val_accuracy: 0.6312
Epoch 2/30
29/29 [==============================] - 73s 3s/step - loss: 0.7221 - accuracy: 0.7271 - val_loss: 0.7003 - val_accuracy: 0.7375
Epoch 3/30
29/29 [==============================] - 72s 2s/step - loss: 0.5968 - accuracy: 0.7786 - val_loss: 0.6061 - val_accuracy: 0.7896
Epoch 4/30
29/29 [==============================] - 73s 3s/step - loss: 0.4576 - accuracy: 0.8296 - val_loss: 0.5548 - val_accuracy: 0.7961
Epoch 5/30
29/29 [==============================] - 73s 3s/step - loss: 0.3972 - accuracy: 0.8568 - val_loss: 0.5024 - val_accuracy: 0.8254
Epoch 6/30
29/29 [==============================] - 73s 3s/step - loss: 0.3708 - accuracy: 0.8681 - val_loss: 0.4563 - val_accuracy: 0.8384
Epoch 7/30
29/29 [==============================] - 73s 3s/step - loss: 0.3064 - accuracy: 0.8939 - val_loss: 0.5085 - val_accuracy: 0.8330
Epoch 8/30
29/29 [==============================] - 71s 2s/step - loss: 0.3007 - accuracy: 0.8898 - val_loss: 0.4692 - val_accuracy: 0.8384
Epoch 9/30
29/29 [==============================] - 72s 2s/step - loss: 0.2801 - accuracy: 0.9013 - val_loss: 0.4213 - val_accuracy: 0.8547
```

The above python depicts the execution of the CNN model. The inception v3 algorithm is run for 30 epochs.

8 RESULTS AND DISCUSSIONS

8.1 OVERALL RESULTS

The algorithms developed are aimed to tackle a multi-class classification problem. In the pertinent project, the aim is to accurately classify four different classes: paper, cardboard, plastic, and metal. Three other algorithms have been implemented along with the base paper, which uses a single layer convolutional neural network. They are a convolutional neural network developed from the ground (vanilla CNN), using transfer learning- VGG 19 and Inception V3. The results are as follows:

Table 8.1 Results

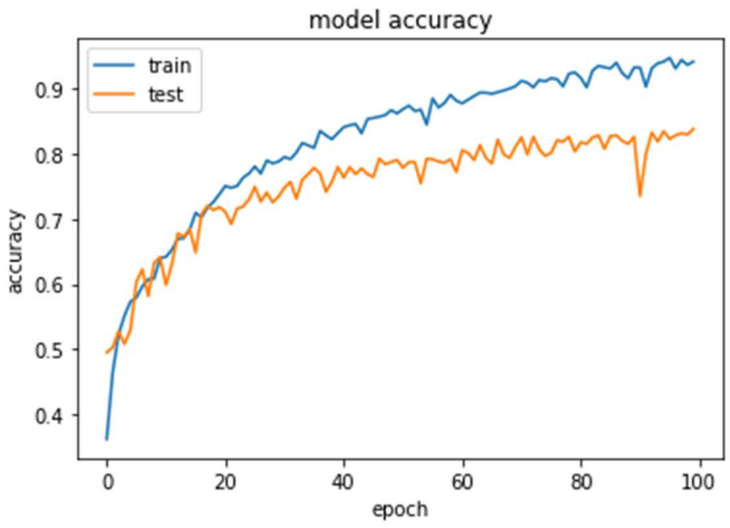
Model	Accuracy (validation)
Base paper CNN	76% (peak)
Vanilla CNN	82%
VGG 19	85%
Inception V3	87%

The results depict Inception V3 is the best performing model, whereas the vanilla CNN is not up to the mark.

8.2 VANILLA CNN

The vanilla CNN built here is a convolutional neural network with multiple hidden layers. This convolutional neural network has four convolutional layers. Each filter size is 3 x 3 with max-pooling of size 2 x 2 applied after the convolution layer. Adam is the optimiser used. Four convolutional layers are chosen in this model since, if they are lesser, the model cannot extract enough information, and any more was causing overfitting. Even the four-layer model had to be restricted by using dropout layers after the last two convolutional layers. This helped in alleviating overfitting. The model has been trained for 100 epochs. The training accuracy was 95%, and validation accuracy was 82%.

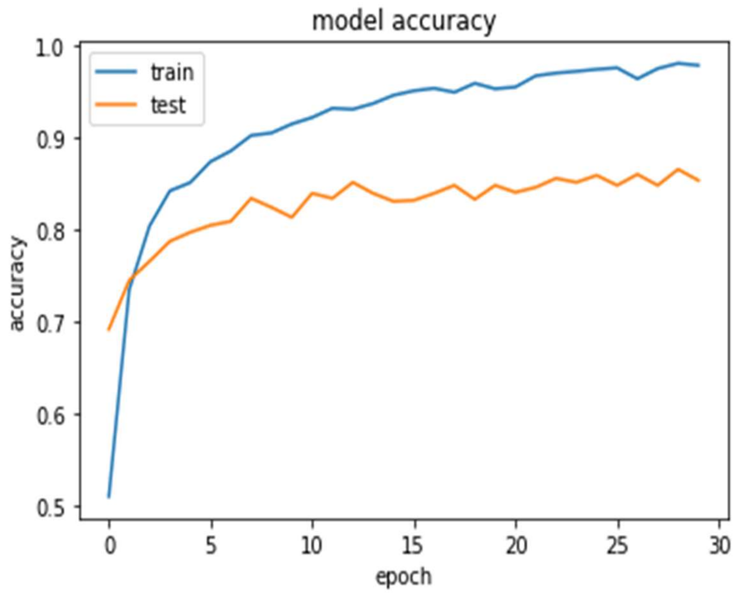
Screen 8.1 Vanilla CNN result



8.3 VGG 19

VGG 19 was trained for a total of 30 epochs. Training has been stopped at 30 epochs since there was an observable change in neither loss nor validation accuracy. Even VGG 19 suffers from overfitting. However, the extent to which the model suffers from overfitting is only marginally better than the vanilla CNN discussed above. The model produced a validation accuracy of 85% and training accuracy of 99%.

Screen 8.2 VGG 19 result

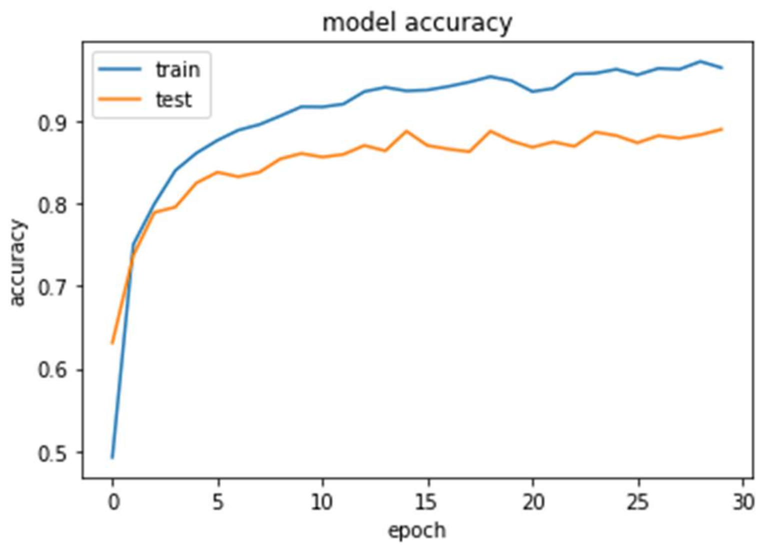


8.4 INCEPTION V3

Inception V3 has been the best performing model. Just like VGG 19, Inception V3 also uses weights obtained from ImageNet. Inception V3 does not suffer from overfitting as much as VGG 19 and

Vanilla. However, the problem still prevails, obviously to a lesser extent. Inception V3 was trained for 30 epochs. It achieved a validation accuracy of 87% and a training accuracy of 96%.

Screen 8.3 Inception V3 result



9 CONCLUSION & FUTURE SCOPE

The human race advances more and more every day. As we grow, build more, the waste we produce every day grows as well. It is not only vital to manage the waste we have, but it is also crucial to reuse it.

The algorithms implemented are a vanilla convolutional neural network, VGG 19 and Inception V3. Of these algorithms, Inception V3 performed the best since it achieved the highest accuracy of them all.

In future, the algorithms mentioned earlier can be designed to accept video as input. Video input can be from a live stream. Adding this feature can make the algorithms work in real-time, especially in waste recycling and procurement plants. Furthermore, a higher quality dataset must be devised to aid the performance of the algorithms. Higher quality dataset here refers to a greater variety of images in a class, including more classes such as glass, rubber, etc. and higher resolution images. Images from a recycling plant will prove to be of utmost importance.

10 REFERENCES

A. Journals/research papers

1. S. R., R. P., V. S., K. R. and G. M., "Deep Learning based Smart Garbage Classifier for Effective Waste Management," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1086-1089, doi: 10.1109/ICCES48766.2020.9137938.
2. J. Shi and Y. Zhao, "Image classification optimization and results analysis based on VGG network," 2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA), 2020, pp. 1090-1099, doi: 10.1109/ICIBA50161.2020.9277329.
3. H. Wang, "Garbage Recognition and Classification System Based on Convolutional Neural Network VGG16," 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), 2020, pp. 252-255, doi: 10.1109/AEMCSE50948.2020.00061.
4. Ruiz V., Sánchez Á., Vélez J.F., Raducanu B. (2019) Automatic Image-Based Waste Classification. In: Ferrández Vicente J., Álvarez-Sánchez J., de la Paz López F., Toledo Moreo J., Adeli H. (eds) From Bioinspired Systems and Biomedical Applications to Machine Learning. IWINAC 2019. Lecture Notes in Computer Science, vol 114\87. Springer, Cham. https://doi.org/10.1007/978-3-030-19651-6_41
5. Olugboja Adedeji, Zenghui Wang, Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network, Procedia Manufacturing, Volume 35, 2019, Pages 607-612, ISSN 2351-9789, https://doi.org/10.1016/j.promfg.2019.05.086.
6. Fatin Amanina Azis, Hazwani Suhaimi, and Emeroylariffion Abas. 2020. Waste Classification using Convolutional Neural Network. In Proceedings of the 2020 2nd International Conference on Information Technology and Computer Communications (ITCC 2020). Association for Computing Machinery, New York, NY, USA, 9–13. DOI:https://doi.org/10.1145/3417473.3417474

B. E-websites/downloads

1. A simple guide to the versions of Inception Network, https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
2. Kaggle, https://www.kaggle.com/asdasdasasdas/garbage-classificatio
3. Kaggle, https://www.kaggle.com/wangziang/waste-pictures
4. Kaggle, https://www.kaggle.com/szdxfkmgnb/waste-classification
5. Super data science, https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening
6. Vidya analytics, https://vitalflux.com/what-softmax-function-why-needed-machine-learning/