Configuration Manual

MSc Research Project
Data Analytics

Vrushali Surve
Student ID: x19212712

School of Computing
National College of Ireland

Supervisor: Paul Stynes, Pramod Pathak, Rejwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vrushali Atul Surve		
Student ID:	X19212712		
Programme:	Data Analytics	**Year:**	2021-22
Module:	MSc Research Project		
Lecturer:	Paul Stynes, Pramod Pathak, Rejwanul Haque		
Submission Due Date:	16/12/2021		
Project Title:	An Image-based Transfer Learning Framework for Classification of E-Commerce Products.		
Word Count:	821	**Page Count:**	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vrushali Atul Surve

Date: 16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vrushali Surve
Student ID: x19212712

1 Hardware/Software Requirements

This configuration manual describes the procedures taken while running the scripts adopted in this research. These instructions will help you run the code successfully. This manual also includes information on the machine's hardware configuration in which the code was executed. The minimal setup required for the system is also described. The process flow of a Convolutional Neural Network (CNN) is covered in this manual, as well as VGG19, InceptionV3, MobileNet, and ResNet50 are image classification algorithms that use transfer learning.

2 System Specification

2.1 Hardware Requirements

The hardware specifications for the system on which the research project is executed are as follows:

- Processor: AMD Ryzen 5 3500U
- Storage: 100 GB.
- RAM: 8GB.

2.2 Software Requirements

The following software is necessary to carry out the experiments:

- Windows Edition: Windows 10 Home.
- Integrated Development Environment: Google Colab
- Scripting Language: Python 3.7
- Cloud Storage: Google Drive.
- Microsoft Tools: Microsoft Excel
- Other Tool: Jupyter Notebook, Anaconda Navigator (64 bit), Notepad ++

3 Setting up environment

3.1 Anaconda Setup

Anaconda Navigator 64-bit configuration is required. First, open anaconda cmd to create an environment for this research using the below statement.

```
(base) C:\Users\vrushali>conda create -n ecommerce python=3.7
```

Then type anaconda navigator into the start menu. Launch Anaconda Navigator by searching for it. The next screen you will see is as fig 1. select e-commerce environment. On the Anaconda platform, there are a variety of navigations and applications. Jupyter Notebook will be utilized for this project. The Jupyter notebook will open in Chrome on port 8888 by default.

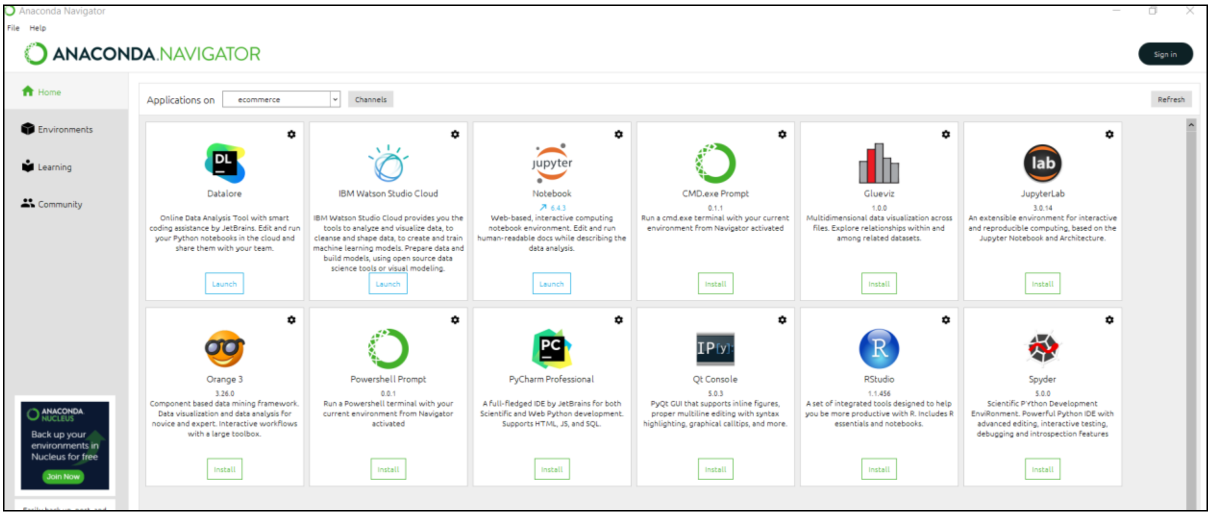

Figure 1 Anaconda Navigator

3.2 Colab Setup

Google Colaboratory environment settled. Fig 2 below can help grasp that concept better.

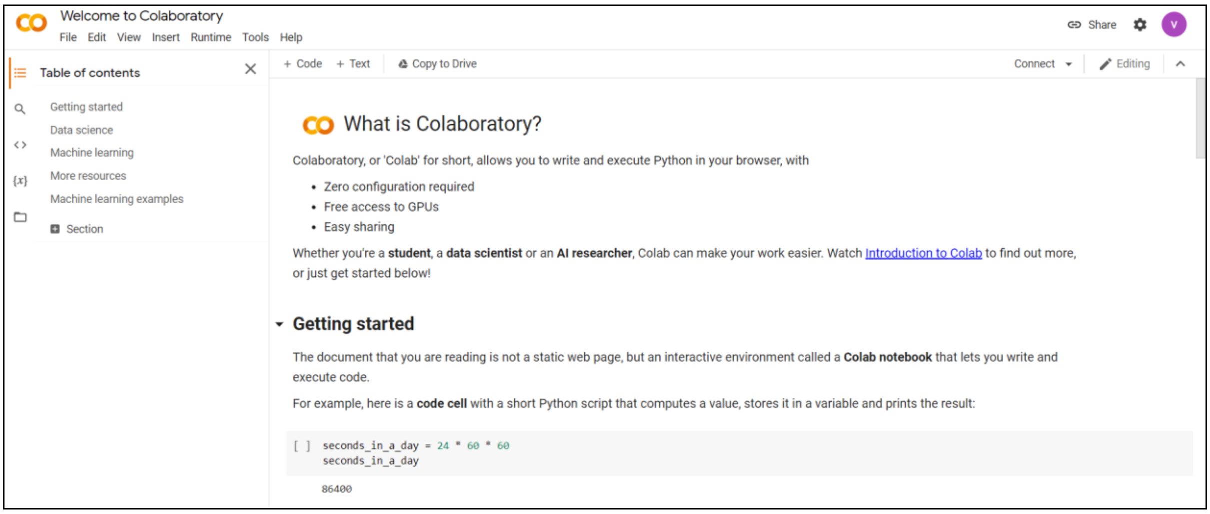

Figure 2 Google Colaboratory

4 Data Selection

The data collection can be found on Kaggle's dataset repository[1] This experiment uses the 'Flipkart Products' dataset shown in below figure 3. This dataset repository has the raw data CSV file and has around unique 20000*15 records.

[1] PromptCloud (2017) Flipkart Products. Available at: https://kaggle.com/PromptCloudHQ/flipkart-products (Accessed: 9 December 2021).

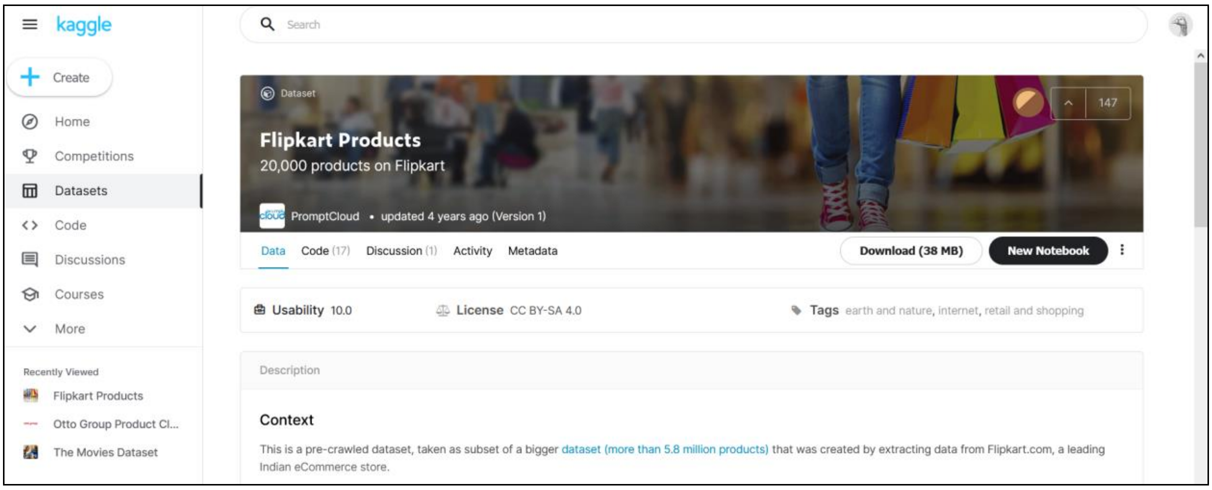

Figure 3 Kaggle Flipkart Dataset

5 Database Creation

Images are imported from the CSV downloaded from Kaggle using the escaping method. BeautifulSoup tool was used for this. First, to create a dataset, we must activate the e-commerce environment, as shown in fig 4. Then scape.py file needs to be run to import the image. Filescrap code is as illustrated in fig 5.

```
(base) C:\Users\vrushali>conda activate ecommerce

(ecommerce) C:\Users\vrushali>d:

(ecommerce) D:\>cd D:\thesis-project\code\flipscrap

(ecommerce) D:\thesis-project\code\flipscrap>python scrape.py
inside
0 http://img5a.flixcart.com/image/short/u/4/a/altht-3p-21-alisha-38-original-imaeh2d5vm5zbtgg.jpeg
1  http://img5a.flixcart.com/image/short/p/j/z/altght4p-26-alisha-38-original-imaeh2d5kbufss6n.jpeg
2  http://img5a.flixcart.com/image/short/p/j/z/altght4p-26-alisha-38-original-imaeh2d5npdybzyt.jpeg
3  http://img5a.flixcart.com/image/short/z/j/7/altght-7-alisha-38-original-imaeh2d5jsz2ghd6.jpeg
inside
0 http://img6a.flixcart.com/image/sofa-bed/j/f/y/fhd112-double-foam-fabhomedecor-leatherette-black-leatherette-1100x11
-imaeh3gemjjcg9ta.jpeg
1  http://img6a.flixcart.com/image/sofa-bed/j/f/y/fhd112-double-foam-fabhomedecor-leatherette-black-leatherette-origin
-imaeh3gemjjcg9ta.jpeg
2  http://img6a.flixcart.com/image/sofa-bed/j/f/y/fhd112-double-foam-fabhomedecor-leatherette-black-leatherette-origin
-imaeh3genfxkqvuv.jpeg
3  http://img5a.flixcart.com/image/sofa-bed/j/f/y/fhd112-double-foam-fabhomedecor-leatherette-black-leatherette-origin
-imaeh3ge2sfeczef.jpeg
4  http://img5a.flixcart.com/image/sofa-bed/j/f/y/fhd112-double-foam-fabhomedecor-leatherette-black-leatherette-origin
-imaeh3geuy7d74g2.jpeg
5  http://img5a.flixcart.com/image/sofa-bed/j/f/y/fhd112-double-foam-fabhomedecor-leatherette-black-leatherette-origin
-imaeh3gerfhdxzwj.jpeg
inside
0 http://img5a.flixcart.com/image/shoe/7/z/z/red-as-454-aw-11-original-imaeebfwsdf6jdf6.jpeg
1  http://img6a.flixcart.com/image/shoe/7/z/z/red-as-454-aw-11-original-imaeebfwsdf6jdf6.jpeg
```

Figure 4 Image Scrap

```python
import os, sys, requests
import pandas as pd
import time
from bs4 import BeautifulSoup
from random import randint
import re, logger

df = pd.read_csv("flipkart_com-ecommerce_sample.csv")
df = df[['product', 'imageurls']]

def download(url, filename):
    try:
        with requests.get(url, stream=True, timeout=10) as r:
            r.raise_for_status()
            with open(filename, 'wb') as f:
                for chunk in r.iter_content(chunk_size=1024):
                    f.write(chunk)
    except Exception as e:
        logger.saveLog(e)

for idx,data in enumerate(df['product']):
    prods = str(data.replace('["',"").replace('"]',"")).split(" >> ")
    if len(str(data.replace('["',"").replace('"]',"")).split(" >> "))>=4:
        # if prods[0] == "Clothing":
        folderName = prods[0] + os.sep + prods[1]
        if not os.path.exists(folderName):
            os.makedirs(folderName)
        iterator = df['imageurls'][idx].replace('[',"").replace(']',"").replace('"','').split(",")
        for imgidx, imgurl  in enumerate(iterator):
            filename = folderName +os.sep+ imgurl.split("/")[-1]
            download(imgurl, filename)
            print(imgidx, imgurl)
```

Figure 5 Web scrapping Code

The crawlers gathered data from the column image URL content in this CSV and imported the picture path. That image was saved in the correct folder for the category. It produced a log folder with a log file containing failure warnings while downloading images by using the logger function.

```python
import logging
from os import path, mkdir
if not path.exists("logs"):
    mkdir("logs")

logging.basicConfig(filename="./logs/log.log",
                    format='%(asctime)s %(message)s',
                    filemode='a', level=logging.INFO)

logger = logging.getLogger()


def saveLog(msg):
    logger.info(msg)
```

Figure 6 Log Code

After that count, check for better understanding using the below fig 7 command.

```
Anaconda Prompt (Anaconda3)

(base) C:\Users\vrushali>conda activate ecommerce

(ecommerce) C:\Users\vrushali>d:

(ecommerce) D:\>cd D:\thesis-project\dataset\scrap-dataset

(ecommerce) D:\thesis-project\dataset\scrap-dataset> *.* /s >> filecount.txt
'*.*' is not recognized as an internal or external command,
operable program or batch file.

(ecommerce) D:\thesis-project\dataset\scrap-dataset>dir  *.* /s >> filecount.txt

(ecommerce) D:\thesis-project\dataset\scrap-dataset>cd D:\thesis-project\dataset\cleaned-dataset

(ecommerce) D:\thesis-project\dataset\cleaned-dataset>dir  *.* /s >> filecount.txt

(ecommerce) D:\thesis-project\dataset\cleaned-dataset>cd D:\thesis-project\dataset\aug-dataset

(ecommerce) D:\thesis-project\dataset\aug-dataset>dir  *.* /s >> filecount.txt

(ecommerce) D:\thesis-project\dataset\aug-dataset>dir  *.* /s >> filecount.txt

(ecommerce) D:\thesis-project\dataset\aug-dataset>
```

Figure 7 File Count To Check

There was 27 category after this. Categories were manually split for more apparent appreciation since there were multiple levels of categorization. So after cleaning, there are 15 categories with fewer images, so using the augmentation process, images are imported. Image augmentation is shown in 8. Count of cleaned categories and after augmentation process is given in snip no 9. Then these image data are saved, zipped and uploaded on google drive.

```
In [2]: # Import Libraries
        import cv2 as cv
        import glob
        import numpy as np
        from scipy import misc
        import imageio
        import os
        import sys
        import imgaug as ia
        from imgaug import augmenters as iaa

In [3]: # Specify folder images to be augmented
        images = []
        files = glob.glob("D://thesis-project//dataset//cleaned-dataset//wrist-watches//*.jpeg")

In [4]: # Reading Images from the folder
        for myFile in files:
            image = cv.imread (myFile,1)
            image_rgb = cv.cvtColor(image, cv.COLOR_BGR2RGB) #change from BRG to RGB
            images.append(image_rgb)

In [5]: # Augementing Images
        sometimes = lambda aug: iaa.Sometimes(0.5, aug)

        seq = iaa.Sequential(
            [
                iaa.Fliplr(0.4), # horizontally flip 50% of all images
                iaa.Flipud(0.1), # vertically flip 20% of all images
                sometimes(iaa.Affine(
                    rotate=(-10, 10),
                )),
                        iaa.OneOf([
                            iaa.GaussianBlur((0, 3.0)),
                            iaa.AverageBlur(k=(2, 7)),
                            iaa.MedianBlur(k=(3, 11)),
                        ])],

                random_order=True)

In [6]: # Folder Path were images to be augmented
        write_to_dir = "D://thesis-project//dataset//aug-dataset//wrist-watches//"

        # Specifying Number of images per image to be augmented
        n_augs_per_image = 20

In [7]: # Saving Augmented Images
        for i, image in enumerate(images):
            image_augs = seq.augment_images([image] * n_augs_per_image)
            for j, image_aug in enumerate(image_augs):
                imageio.imwrite(os.path.join(write_to_dir, "%03d_%02d.jpg" % (i, j)), image_aug)
                print(os.path.join(write_to_dir, "%03d_%02d.jpg" % (i, j)))

        D://thesis-project//dataset//aug-dataset//wrist-watches//000_00.jpg
        D://thesis-project//dataset//aug-dataset//wrist-watches//000_01.jpg
        D://thesis-project//dataset//aug-dataset//wrist-watches//000_02.jpg
        D://thesis-project//dataset//aug-dataset//wrist-watches//000_03.jpg
        D://thesis-project//dataset//aug-dataset//wrist-watches//000_04.jpg
        D://thesis-project//dataset//aug-dataset//wrist-watches//000_05.jpg
```

Figure 8 Augmentation Code

Category	Cleaned Count	Augmented Count
bags	189	1134
cookware	160	1120
fragrances	589	1178
infant-wear	279	1116
jewellery	336	1008
kids-clothing	493	986
kids-footwear	173	1038
mens-clothing	1239	1239
mens-footwear	225	1125
network-components	557	1114
school-supplies	692	1384
showpieces	445	890
women-clothing	1230	1230
women-footwear	1097	1097
wrist-watches	51	1020

Figure 9 Count

6 Implementation

The libraries in fig 10 need to develop in this project are below.

Import Required Libraries

```python
#Importing required libraries for all modals
import tensorflow as tf
graph = tf.compat.v1.reset_default_graph() #Clears the default graph stack and resets the global default graph.

import os #used module that is handy when processing files from different places in the system
import time #which allows us to handle various operations regarding time, its conversions and representations
import cv2 #used to load an image from the specified file.
import keras #provides a Python interface for artificial neural networks
import glob #string of literal and/or wildcard characters used to match filepaths
import random #includes the module into the namespace under the name 'random'
import numpy as np #work with array with image data
import pandas as pd #mathematical calculation
import seaborn as sns #for data visualization
from keras import models #represents the actual neural network model
from keras import optimizers #improve training speed and performance
import matplotlib.pyplot as plt #a collection of command style functions that make matplotlib work like MATLAB.
from keras.preprocessing import image #Provides utilities for working with image data, text data, and sequence data
from tensorflow.keras import utils #utilities
from sklearn.metrics import classification_report
from tensorflow.keras.models import load_model #used to measure the quality of predictions from a classification algorithm.
from keras.applications import ResNet50 #to train ultra deep neural networks and for model,good performance
from tensorflow.keras.layers import Input #to instantiate a Keras tensor
from tensorflow.keras.models import Model #simple, user-friendly way to define a neural network, which will then be built by TensorFlow
from tensorflow.keras.layers import Dense #regular deeply connected neural network layer
from tensorflow.keras.models import Sequential #helps to form a cluster of a layer that is linearly stacked into tf. keras. Model.
from sklearn.model_selection import train_test_split #sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay #for Plotting Confusion Matrix
from tensorflow.keras import datasets, layers, models, callbacks #commands you can define that execute in response to a specific modeling action, such as opening a model or stopping a simulation.
from keras.layers.core import Dense, Dropout, Activation, Flatten #dropout -Dropout technique works by randomly reducing the number of interconnecting neurons within a neural network ,actication- used to implement Softmax activation
from keras.layers.convolutional import Convolution2D, MaxPooling2D #a convolution kernel that is convolved with the layer input to produce a tensor of outputs.,maxpooling- Max pooling operation for 2D spatial data
from keras.applications import VGG19, InceptionV3, Xception, MobileNetV2, ResNet50, MobileNet #Preprocesses a tensor or Numpy array encoding a batch of images
from keras.callbacks import EarlyStopping, ModelCheckpoint #EarlyStopping-Training will stop when the chosen performance measure stops improving.,modelcheckppint- fault tolerance technique for long running processes.
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten, Activation, BatchNormalization

import warnings
warnings.filterwarnings("ignore")
```

Figure 10 Library List

6.1 Importing Data

As seen in figure 11, mounting the google drive to the colab notebook. Select the Gmail account from the URL and enter the authentication code.

E-Commerce Product Classification

```python
[ ]  # Mounting Google Drive locally
     from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Figure 11 Google Drive Mount

Figure 12, Keras & TensorFlow library imported.

```
#deep learning API written in Python-version matching with 2.3.0
!pip install keras==2.4.3

Collecting keras==2.4.3
  Downloading Keras-2.4.3-py2.py3-none-any.whl (36 kB)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages (from keras==2.4.3) (1.19.5)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.7/dist-packages (from keras==2.4.3) (1.4.1)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages (from keras==2.4.3) (3.13)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from keras==2.4.3) (3.1.0)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py->keras==2.4.3) (1.5.2)
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: keras 2.7.0
    Uninstalling keras-2.7.0:
      Successfully uninstalled keras-2.7.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of th
tensorflow 2.7.0 requires keras<2.8,>=2.7.0rc0, but you have keras 2.4.3 which is incompatible.
Successfully installed keras-2.4.3

#an open source software library for high performance numerical computation -its matching for keras version 2.4.3-for model calculation
!pip install tensorflow==2.3.0

Collecting tensorflow==2.3.0
  Downloading tensorflow-2.3.0-cp37-cp37m-manylinux2010_x86_64.whl (320.4 MB)
     |████████████████████████████████| 320.4 MB 9.9 kB/s
Collecting gast==0.3.3
  Downloading gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Collecting numpy<1.19.0,>=1.16.0
  Downloading numpy-1.18.5-cp37-cp37m-manylinux1_x86_64.whl (20.1 MB)
     |████████████████████████████████| 20.1 MB 1.3 MB/s
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.0) (1.15.0)
```

Figure 12 Keras-Tensorflow import

Test directory, train directory, image size and classes are set in figure 13. then a training dataset was created.

```
# Reading Train Directory, Test Directory and Number of Classes
directory = '/content/drive/MyDrive/aug-dataset/'
test_directory = '/content/drive/MyDrive/test-dataset/'
classes = ['bags', 'cookware', 'fragrances', 'infant-wear', 'jewellery', 'kids-clothing', 'kids-footwear', 'mens-clothing', 'mens-footwear', 'network-components', 'school-supplies', 'showpieces', 'women-cl

# Defining a shape to be used for our models.
img_size = 224

# See Total Number of Categories
print("Total Number of Categories -", len(classes))

Total Number of Categories - 15

# Creating Training Dataset
training_data = []
i = 0


def train_data():
    for category in classes:
        path = os.path.join(directory,category)
        class_num = classes.index(category)

        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_COLOR)
            RGB_img = cv2.cvtColor(img_array, cv2.COLOR_BGR2RGB)
            new_img = cv2.resize(RGB_img,(img_size,img_size))
            training_data.append([new_img,class_num])
            print('Image Creating for Training -', len(training_data))

# Calling Function to Create Training Data
train_data()
```

Figure 13 Dataset create

Dataset is split into training and testing ration 70:30, as shown in figure 14.

```
[ ]  print('Total number images for training -', len(training_data))

     # Randomly Shuffling Data
     random.shuffle(training_data)

     x = []
     y = []

     # Assign Features and Labels
     for features, label in training_data:
         x.append(features)
         y.append(label)

     # Reshapping Image Arrays
     x = np.array(x).reshape(-1,img_size,img_size,3)
     x[0].shape

     # Split Dataset Ratio (70:30)
     x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=96)

     # Deleting Variables
     del x,y

     # Converting Variables to Categories
     Y_train = utils.to_categorical(y_train,num_classes=15)
     Y_test = utils.to_categorical(y_test,num_classes=15)

     Total number images for training - 16679
```

Figure 14 Split DS for Model Training

There were three experiments conducted in this research to archive the aim.

6.2 Experiment 1: Effectivity of CNN Model

The CNN model is shown in figures 15 a and 15b.

```python
class MyModel():

    def __init__(self, baseModel, classes, D):
        self.baseModel = baseModel
        self.classes = classes
        self.D = D


    def build(self):
        """
            It takes basemodel , number of classes, and number of hidden units as a input.
            And return a new CNN architecure.

        """
        headModel = self.baseModel.output
        headModel = Flatten(name="Flatten")(headModel)
        headModel = Dense(self.D, activation='relu')(headModel)
        headModel = Dropout(0.5)(headModel)
        headModel = Dense(self.classes, activation='softmax')(headModel)

        return headModel

class MyCNN():

    def __init__(self, classes):
        self.classes = classes


    def build(self):

        model = Sequential()
        model.add(Convolution2D(32, (3, 3) , input_shape=(224,224,3)))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.5))

        model.add(Convolution2D(64, ( 3, 3)))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.5))

        model.add(Convolution2D(128, (3, 3)))
        model.add(MaxPooling2D(pool_size=(8, 8)))#stride is not given default to the pool size

        model.add(Dropout(0.5))

        model.add(Flatten())
        model.add(Dense(self.classes))
        model.add(Activation('softmax'))

        return model
```

```python
# Number of Classes
n_classes = 15

# Early Stop Patience
early_stop_patience = 10

# Batch Size
batch_size = 100

# Number of Epochs
n_epochs = 20
```

Figure 15. a CNN

```python
# Number of Classes
n_classes = 15

# Early Stop Patience
early_stop_patience = 10

# Batch Size
batch_size = 100

# Number of Epochs
n_epochs = 20
```

```python
# initialize head of network
my_model = MyCNN(classes=n_classes)
my_model = my_model.build()
```

```python
my_model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
print(my_model.summary())

weight_path = '/content/drive/MyDrive/weights/mycnn'

cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')


results = my_model.fit(x_train,Y_train,
                    batch_size=batch_size,epochs=n_epochs,
                    verbose=1,
                    validation_data=(x_test,Y_test),
                    callbacks=[cb_checkpointer, cb_early_stopper])

print(results.history.keys())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 15)                69135
_________________________________________________________________
activation (Activation)      (None, 15)                0
=================================================================
Total params: 162,383
Trainable params: 162,383
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/20
117/117 [==============================] - ETA: 0s - loss: 65.9987 - accuracy: 0.1398WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/training/tracking/tracking.py:111: Model.state_updates (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
```

Figure 15. b CNN

6.3 Experiment 2: Effectiveness of transfer learning models like VGG19 and InceptionV3.

In this experiment, both models use pre-trained ImageNet weights shown in VGG19 in fig 16 and InceptionV3 in fig 17.

```
# VGG19
# load pretrained model:
baseModel = VGG19(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

# initialize head of network
my_model = MyModel(baseModel, classes=n_classes, D=64)
headModel = my_model.build()

# replace the FC with headModel:
my_model = Model(inputs=baseModel.input, outputs=headModel)

# unfreezing some of conv layers:
for layer in baseModel.layers[:]:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 2s 0us/step
```

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

weight_path = '/content/drive/MyDrive/weights/vgg19/'

cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')

results = my_model.fit(x_train,Y_train,
                    batch_size=batch_size,epochs=n_epochs,
                    verbose=1,
                    validation_data=(x_test,Y_test),
                    callbacks=[cb_checkpointer, cb_early_stopper])


print(results.history.keys())
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
```

Figure 16 VGG19

```
[ ] #InceptionV3
    # load pretrained model:
    baseModel = InceptionV3(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

    # initialize head of network
    my_model = MyModel(baseModel, classes=n_classes, D=256)
    headModel = my_model.build()

    # replace the FC with headModel:
    my_model = Model(inputs=baseModel.input, outputs=headModel)

    # unfreezing some of conv layers:
    for layer in baseModel.layers[15:]:
        layer.trainable = True

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
    87916544/87910968 [==============================] - 1s 0us/step

[ ] batch_size = 100
    n_epochs = 10

    my_model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
    print(my_model.summary())

    weight_path = '/content/drive/MyDrive/weights/inceptiov3/'

    cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
    cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')

    results = my_model.fit(x_train,Y_train,
                     batch_size=batch_size,epochs=n_epochs,
                     verbose=1,
                     validation_data=(x_test,Y_test),
                     callbacks=[cb_checkpointer, cb_early_stopper])


    print(results.history.keys())

    Model: "functional_3"
    __________________________________________________________________________________________________
    Layer (type)                    Output Shape         Param #     Connected to
    ==================================================================================================
    input_2 (InputLayer)            [(None, 224, 224, 3) 0
    __________________________________________________________________________________________________
    conv2d_3 (Conv2D)               (None, 111, 111, 32) 864         input_2[0][0]
    __________________________________________________________________________________________________
    batch_normalization (BatchNorma (None, 111, 111, 32) 96          conv2d_3[0][0]
    __________________________________________________________________________________________________
    activation_1 (Activation)       (None, 111, 111, 32) 0           batch_normalization[0][0]
```

Figure 17 InceptionV3

6.4 Experiment 3: Effectivity of transfer learning models like ResNet50 and MobileNet model.

Transfer learning models are better as they save time. Below fig 18 and 19 shows an implementation of model ResNet50 and MobileNet, respectively.

```
#MobileNet
# load pretrained model:
baseModel = MobileNet(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

# initialize head of network
my_model = MyModel(baseModel, classes=n_classes, D=256)
headModel = my_model.build()

# replace the FC with headModel:
my_model = Model(inputs=baseModel.input, outputs=headModel)

# unfreezing some of conv layers:
for layer in baseModel.layers[15:]:
    layer.trainable = True
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5
17227776/17225924 [==============================] - 0s 0us/step
```

```
batch_size = 50
n_epochs = 10

my_model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
print(my_model.summary())

weight_path = '/content/drive/MyDrive/weights/mobilenet/'

cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')

results = my_model.fit(x_train,Y_train,
                    batch_size=batch_size,epochs=n_epochs,
                    verbose=1,
                    validation_data=(x_test,Y_test),
                    callbacks=[cb_checkpointer, cb_early_stopper])


print(results.history.keys())
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 225, 225, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
```

Figure 18 MobileNet

```
#ResNet50
# load pretrained model:
baseModel = ResNet50(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

# initialize head of network
my_model = MyModel(baseModel, classes=n_classes, D=256)
headModel = my_model.build()

# replace the FC with headModel:
my_model = Model(inputs=baseModel.input, outputs=headModel)

# unfreezing some of conv layers:
for layer in baseModel.layers[15:]:
    layer.trainable = True
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 1s 0us/step
```

```
batch_size = 50
n_epochs = 10

my_model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
print(my_model.summary())

weight_path = '/content/drive/MyDrive/weights/ResNet50/'

cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')

results = my_model.fit(x_train,Y_train,
                    batch_size=batch_size,epochs=n_epochs,
                    verbose=1,
                    validation_data=(x_test,Y_test),
                    callbacks=[cb_checkpointer, cb_early_stopper])


print(results.history.keys())
```

```
Model: "functional_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_3[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
```

Figure 19 ResNet50

7 Evaluation & Result

In these steps, graphs are created using performance measures like Confusion matrix, classification Report and loss/accuracy versus epochs graph. Code for these graphs is shown in Figures 20,21 and 22.

```
# Plot training & validation accuracy values
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='lower right')
plt.show()

# Plot training & validation loss values
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper right')
plt.show()
```

Figure 20 Accuracy\Loss verses Epoch

```
# Predicting on Validation Set
predict_x = my_model.predict(x_test)
pred = np.argmax(predict_x,axis=1)

# Plotting Confusion Matrix
cm = confusion_matrix(y_test, pred)
plt.figure(figsize=(15, 12))
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);  #annot=True to annotate cells, ftm='g' to disable scientific notation

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(classes,rotation=90); ax.yaxis.set_ticklabels(classes,rotation=0);
```

Figure 21 Confusion Matrix

```
# Classification Report
print(classification_report(y_test, pred))
```

Figure 22 Classification Report

Inferencing on testing data is done to check model flow.

```
# Inferencing on Testing Data
images = []
titles = []

for img in glob.glob(test_directory + "*.jpeg"):
    start_time = time.time()
    img_name = os.path.split(img)[-1]
    im = cv2.imread(img)
    im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    img_og = image.load_img(img,target_size=(224,224))
    img = np.asarray(img_og)
    img = np.expand_dims(img, axis=0)
    prediction = my_model.predict(img)
    MaxPosition=np.argmax(prediction)
    prediction_label=classes[MaxPosition]
    print(prediction_label)
    org = (50, 50)
    fontScale = 1
    color = (255, 0, 0)
    thickness = 2
    cv2.putText(im, prediction_label, org, cv2.FONT_HERSHEY_SIMPLEX,
                fontScale, color, thickness, cv2.LINE_AA)
    end_time = time.time()
    infer_timne = end_time - start_time
    print('Time Requirted: ',{infer_timne})
    plt.imshow(im)
    plt.show()
cv2.destroyAllWindows()
```

Figure 23 Inferencing Code