# Configuration Manual

MSc Research Project
Programme Name

Srija Venkata Sai Ravali Kothapalli
Student ID: X21227454

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

**National College of Ireland**
**MSc Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Srija Venkata Sai Ravali Kothapalli |
| **Student ID:** | X21227454 |
| **Programme:** | MSc in Data Analytics |
| **Year:** | 2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Abid Yaqoob |
| **Submission Due Date:** | 31/01/2024 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | XXX |
| **Page Count:** | 10 |

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:** | |
| **Date:** | 31st January 2024 |

**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

Srija Venkata Sai Ravali Kothapalli
Student ID: x21227454

## 1 Introduction

This manual details about the setup and various techniques that has been followed such as software and hardware specifications, and the processes used for the implementation of this research project with the proposed models which are deep learning tools such as Transfer-Learning models like EfficientNet, MobileNetV3, and InceptionV3 and the newly built Convolutional Neural Network model for "Wildfire Detection and Aerosol Identification using Satellite Imagery".

## 2 System Configuration

| | |
|---|---|
| **Programming Language** | Python Version 3 |
| **Tools** | Google Collaboratory, Word, and Excel |
| **Email** | Access to Gmail Account |

**Table 1:** Software Configuration

| | |
|---|---|
| OS | Windows 10 |
| RAM | Minimum 8GB |
| Hard Disk Space | Minimum 100 GB |

**Table 2:** Hardware Configuration

## 3 Working with Google Collaboratory

The benchmark has been run using the google collab system with Python 3 version Google Compute Engine (GPU). Here we have used T4-GPU as the hardware accelerator with the shape of High-RAM. The System RAM in the collab was 51.0GB and the Disk had 166.8 GB space. Gmail account is required to access the Google Colab and the information related to the research project such as Dataset and the Notebook files have been stored in the Google Drive and performed the analysis. Keras and the TensorFlow libraries are used with the respective proposed models to train the models.

## 4 Dataset Sources

USTS-Smoker dataset has been obtained from the Kaggle; it has all the data related to six categories of images in aerosol classes. It has 6225 RGB images with spatial resolution of 1km for each image.

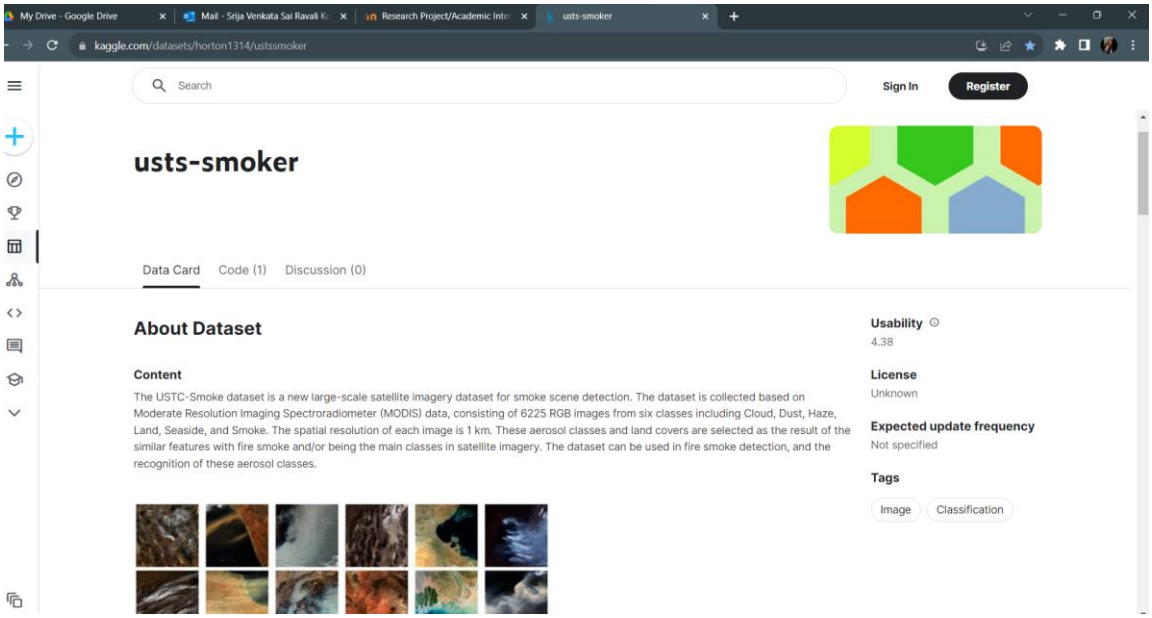



Figure 1: Dataset from Kaggle

# 5 Project Implementation

The Dataset has been obtained from the Kaggle Website and is uploaded on Google Drive, where we have to setup the Colab Environment to perform the experiments in this research.

Step 1: Uploading the Dataset on the Google Drive

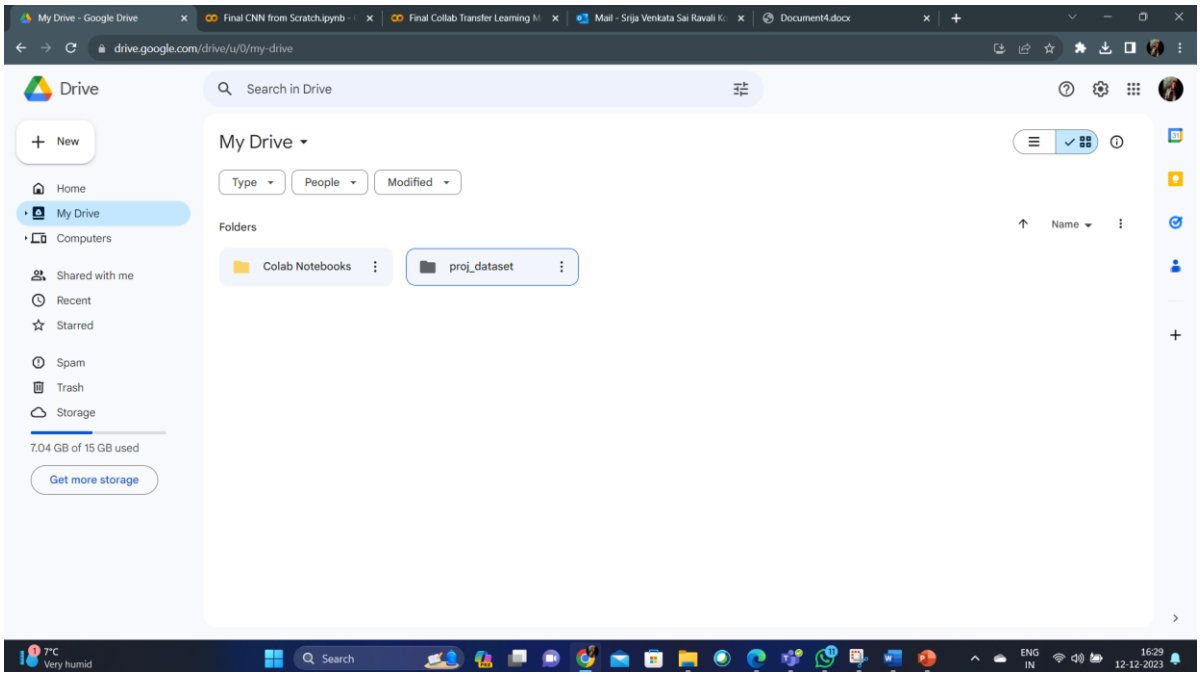



Figure 2: Dataset in Google Drive

The Dataset has been downloaded and it is stored on the Google Drive as proj_dataset. Here the drive has to be mounted in the colab where the dataset is imported to use the data for the research.

Step 2: Importing required Packages and Libraries

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization, Flatten
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
import numpy as np
import pandas as pd
import shutil
import time
from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import seaborn as sns
sns.set_style('darkgrid')
from PIL import Image
from sklearn.metrics import confusion_matrix, classification_report
from IPython.core.display import display
import kerastuner as kt
import random
import IPython
```

Figure 3: Python Libraries

Step 3: Mounting Google Drive to Colab.

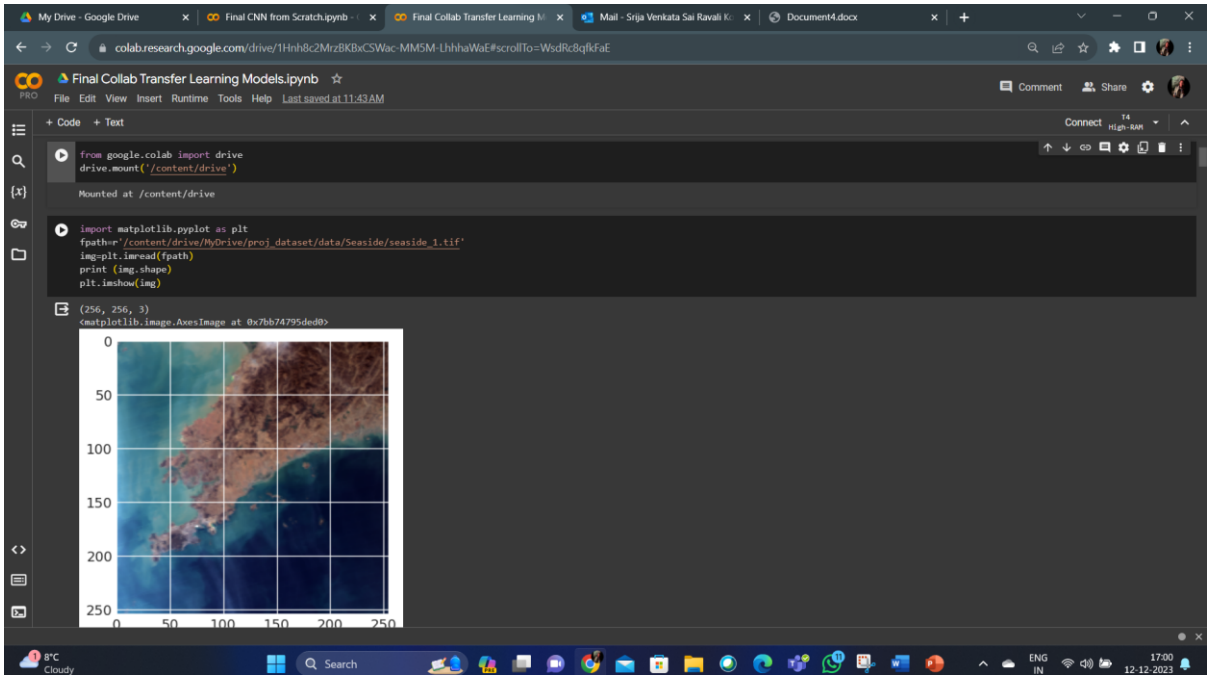



Figure 4: Mounting Google Drive

Step 4: Data Splitting and Data Balancing, where equal number of images is taken inwith respect to six categories of data related to aerosol classes in the dataset.

```
# Data Splitting
sdir=r'/content/drive/MyDrive/proj_dataset/data/'
classlist=os.listdir(sdir)
filepaths=[]
labels=[]
#classlist.remove(".DS_Store")
for klass in classlist:
    classpath=os.path.join(sdir,klass)
    flist=os.listdir(classpath)
    for f in flist:
        fpath=os.path.join(classpath,f)
        filepaths.append(fpath)
        labels.append(klass)
Fseries=pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
trsplit= .9
vsplit=.05
dsplit=vsplit/(1-trsplit)
train_df, dummy_df = train_test_split(df, train_size=trsplit, shuffle=True, random_state=123)
valid_df, test_df = train_test_split(dummy_df, train_size=dsplit, shuffle=True, random_state=123)
print('train_df length: ', len(train_df), '  test_df length: ',len(test_df), '  valid_df length: ', len(valid_df))
balance=list(train_df['labels'].value_counts())
for b in balance:
    print (b)
```

```
train_df length:  5602   test_df length:  312   valid_df length:  311
1049
923
911
911
906
902
```

Figure 5: Data Splitting

```
# Equal Number of Images in all categories (Balancing Data)
sample_list=[]
max_size= 902
min_size = 0
groups=train_df.groupby('labels')
for label in train_df['labels'].unique():
    group=groups.get_group(label)
    sample_count=len(group)
    if sample_count> max_size :
        samples=group.sample(max_size, replace=False, weights=None, random_state=123, axis=0).reset_index(drop=True)
        sample_list.append(samples)
    elif sample_count>= min_size:
        sample_list.append(group)
train_df=pd.concat(sample_list, axis=0).reset_index(drop=True)
print (len(train_df))
print (train_df['labels'].value_counts())
print(train_df)
```

```
5412
Haze       902
Seaside    902
Cloud      902
Dust       902
Smoke      902
Land       902
Name: labels, dtype: int64
                                              filepaths labels
0     /content/drive/MyDrive/proj_dataset/data/Haze/...   Haze
1     /content/drive/MyDrive/proj_dataset/data/Haze/...   Haze
2     /content/drive/MyDrive/proj_dataset/data/Haze/...   Haze
3     /content/drive/MyDrive/proj_dataset/data/Haze/...   Haze
4     /content/drive/MyDrive/proj_dataset/data/Haze/...   Haze
...                                                 ...    ...
5407  /content/drive/MyDrive/proj_dataset/data/Land/...   Land
5408  /content/drive/MyDrive/proj_dataset/data/Land/...   Land
5409  /content/drive/MyDrive/proj_dataset/data/Land/...   Land
5410  /content/drive/MyDrive/proj_dataset/data/Land/...   Land
5411  /content/drive/MyDrive/proj_dataset/data/Land/...   Land

[5412 rows x 2 columns]
```

Figure 6: Data Balancing

Step 5: Pre-Processing Steps like Image-Processing, Data Augmentation , and Data Normalization on the data.

```
# Resize, Data Normalization, Data Augmentation
height=224
width=224
channels=3
batch_size=3
img_shape=(height, width, channels)
img_size=(height, width)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/n<=80],reverse=True)[0]
test_steps=int(length/test_batch_size)
print ( 'test batch size: ' ,test_batch_size, '  test steps: ', test_steps)
def scalar(img):
    #img=img/127.5-1
    return img
trgen=ImageDataGenerator(preprocessing_function=scalar, horizontal_flip=True)
tvgen=ImageDataGenerator(preprocessing_function=scalar)
train_gen=trgen.flow_from_dataframe( train_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=True, batch_size=batch_size)
test_gen=tvgen.flow_from_dataframe( test_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=False, batch_size=test_batch_size)
valid_gen=tvgen.flow_from_dataframe( valid_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=True, batch_size=batch_size)
classes=list(train_gen.class_indices.keys())
class_count=len(classes)
train_steps=np.ceil((len(train_gen.labels)/batch_size))
```

```
test batch size:  78   test steps:  4
Found 5412 validated image filenames belonging to 6 classes.
Found 312 validated image filenames belonging to 6 classes.
Found 311 validated image filenames belonging to 6 classes.
```

Figure 6: Data Pre-Processing steps in performing Data Augumentation and Normalization

Below are the proposed models implemented each model with 25 epochs with their respective outparamters.

## Step 6: EfficientNet Model

```
# Efficientnet Model
model_name='EfficientNetB1'
base_model=tf.keras.applications.EfficientNetB1(include_top=False, weights="imagenet",input_shape=img_shape, pooling='max')
x=base_model.output
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(256, kernel_regularizer = regularizers.l2(l = 0.016),activity_regularizer=regularizers.l1(0.006),
                bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.45, seed=123)(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)
model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])
```

```
epochs =25
patience= 1 # number of epochs to wait to adjust lr if monitored value does not improve
stop_patience =3 # number of epochs to wait before stopping training if monitored value does not improve
threshold=.9 # if train accuracy is < threshhold adjust monitor accuracy, else monitor validation loss
factor=.5 # factor to reduce lr by
dwell=True # experimental, if True and monitored metric does not improve on current epoch set  modelweights back to weights of previous epoch
ask_epoch=5 # number of epochs to run before asking if you want to halt training
batches=train_steps
with tf.device("/gpu:0"):
    callbacks=[LRA(model=model,base_model= base_model,patience=patience,stop_patience=stop_patience, threshold=threshold,
                   factor=factor,dwell=dwell, batches=batches,initial_epoch=0,epochs=epochs, ask_epoch=ask_epoch )]
    history=model.fit(x=train_gen, epochs=epochs, verbose=1,callbacks=callbacks, validation_data=valid_gen,validation_steps=None,  shuffle=False,  initial_epoch=0)
```

```
 initializing callback starting train with base_model trainable

 Epoch     Loss   Accuracy  V_loss    V_acc     LR      Next LR  Monitor  Duration

Epoch 1/25
1804/1804 [==============================] - ETA: 0s - loss: 4.6131 - accuracy: 0.4913 1 /25     4.613   49.132   2.62967  38.585   0.00100  0.00100  accuracy   174.27

1804/1804 [==============================] - 174s 73ms/step - loss: 4.6131 - accuracy: 0.4913 - val_loss: 2.6297 - val_accuracy: 0.3859
Epoch 2/25
1804/1804 [==============================] - ETA: 0s - loss: 1.7512 - accuracy: 0.5767 2 /25     1.751   57.668   1.40125  69.132   0.00100  0.00100  accuracy   129.74
```

```
epochs =25
patience= 1 # number of epochs to wait to adjust lr if monitored value does not improve
stop_patience =3 # number of epochs to wait before stopping training if monitored value does not improve
threshold=.9 # if train accuracy is < threshhold adjust monitor accuracy, else monitor validation loss
factor=.5 # factor to reduce lr by
dwell=True # experimental, if True and monitored metric does not improve on current epoch set  modelweights back to weights of previous epoch
ask_epoch=5 # number of epochs to run before asking if you want to halt training
batches=train_steps
with tf.device("/gpu:0"):
    callbacks=[LRA(model=model,base_model= base_model,patience=patience,stop_patience=stop_patience, threshold=threshold,
                   factor=factor,dwell=dwell, batches=batches,initial_epoch=0,epochs=epochs, ask_epoch=ask_epoch )]
    history=model.fit(x=train_gen, epochs=epochs, verbose=1,callbacks=callbacks, validation_data=valid_gen,validation_steps=None,  shuffle=False,  initial_epoch=0)
```

```
1804/1804 [==============================] - ETA: 0s - loss: 0.9279 - accuracy: 0.791914 /25     0.928   79.194   0.90850  85.852   0.00050  0.00050  accuracy   126.93

1804/1804 [==============================] - 127s 70ms/step - loss: 0.9279 - accuracy: 0.7919 - val_loss: 0.9085 - val_accuracy: 0.8585
Epoch 15/25
1804/1804 [==============================] - ETA: 0s - loss: 0.9251 - accuracy: 0.800115 /25     0.925   80.007   0.74379  86.495   0.00050  0.00050  accuracy   126.59

1804/1804 [==============================] - 127s 70ms/step - loss: 0.9251 - accuracy: 0.8001 - val_loss: 0.7438 - val_accuracy: 0.8650
Epoch 16/25
1804/1804 [==============================] - ETA: 0s - loss: 0.8493 - accuracy: 0.817116 /25     0.849   81.707   0.79425  86.174   0.00050  0.00050  accuracy   125.55

1804/1804 [==============================] - 126s 70ms/step - loss: 0.8493 - accuracy: 0.8171 - val_loss: 0.7943 - val_accuracy: 0.8617
Epoch 17/25
1804/1804 [==============================] - ETA: 0s - loss: 0.8597 - accuracy: 0.820617 /25     0.860   82.058   0.99001  80.707   0.00050  0.00050  accuracy   125.58

1804/1804 [==============================] - 126s 70ms/step - loss: 0.8597 - accuracy: 0.8206 - val_loss: 0.9900 - val_accuracy: 0.8071
Epoch 18/25
1804/1804 [==============================] - ETA: 0s - loss: 0.8290 - accuracy: 0.819118 /25     0.829   81.911   0.89185  82.637   0.00050  0.00025  accuracy   125.07

1804/1804 [==============================] - 125s 69ms/step - loss: 0.8290 - accuracy: 0.8191 - val_loss: 0.8918 - val_accuracy: 0.8264
Epoch 19/25
1804/1804 [==============================] - ETA: 0s - loss: 0.6660 - accuracy: 0.862019 /25     0.666   86.197   0.70042  84.244   0.00025  0.00025  accuracy   125.99

1804/1804 [==============================] - 126s 70ms/step - loss: 0.6660 - accuracy: 0.8620 - val_loss: 0.7004 - val_accuracy: 0.8424
Epoch 20/25
1804/1804 [==============================] - ETA: 0s - loss: 0.6123 - accuracy: 0.875320 /25     0.612   87.528   0.62214  86.817   0.00025  0.00025  accuracy   125.48

1804/1804 [==============================] - 126s 70ms/step - loss: 0.6123 - accuracy: 0.8753 - val_loss: 0.6221 - val_accuracy: 0.8682
Epoch 21/25
1804/1804 [==============================] - ETA: 0s - loss: 0.5637 - accuracy: 0.884521 /25     0.564   88.452   0.57134  92.605   0.00025  0.00025  accuracy   126.96

1804/1804 [==============================] - 127s 70ms/step - loss: 0.5637 - accuracy: 0.8845 - val_loss: 0.5713 - val_accuracy: 0.9260
Epoch 22/25
1804/1804 [==============================] - ETA: 0s - loss: 0.5623 - accuracy: 0.884722 /25     0.562   88.470   0.65069  89.068   0.00025  0.00025  accuracy   127.62

1804/1804 [==============================] - 128s 71ms/step - loss: 0.5623 - accuracy: 0.8847 - val_loss: 0.6507 - val_accuracy: 0.8907
Epoch 23/25
1804/1804 [==============================] - ETA: 0s - loss: 0.5299 - accuracy: 0.897123 /25     0.530   89.708   0.52426  89.068   0.00025  0.00025  accuracy   126.96
```

```
1804/1804 [==============================] - 128s 71ms/step - loss: 0.5623 - accuracy: 0.8847 - val_loss: 0.6507 - val_accuracy: 0.8907
Epoch 23/25
1804/1804 [==============================] - ETA: 0s - loss: 0.5299 - accuracy: 0.897123 /25     0.530   89.708   0.52426  89.068   0.00025  0.00025  accuracy   126.96

1804/1804 [==============================] - 127s 70ms/step - loss: 0.5299 - accuracy: 0.8971 - val_loss: 0.5243 - val_accuracy: 0.8907
Epoch 24/25
1804/1804 [==============================] - ETA: 0s - loss: 0.5174 - accuracy: 0.899724 /25     0.517   89.967   0.58635  89.711   0.00025  0.00025  accuracy   127.46

1804/1804 [==============================] - 128s 71ms/step - loss: 0.5174 - accuracy: 0.8997 - val_loss: 0.5864 - val_accuracy: 0.8971
Epoch 25/25
1804/1804 [==============================] - ETA: 0s - loss: 0.4965 - accuracy: 0.905425 /25     0.497   90.540   0.67487  88.424   0.00025  0.00013  val_loss   128.36

enter H to halt ,F to fine tune model, or an integer for number of epochs to run then ask again

F
setting base_model as trainable for fine tuning of model

 Epoch     Loss   Accuracy  V_loss    V_acc     LR      Next LR  Monitor  Duration

1804/1804 [==============================] - 218s 121ms/step - loss: 0.4965 - accuracy: 0.9054 - val_loss: 0.6749 - val_accuracy: 0.8842
Training is completed - model is set with weights from epoch 24

training elapsed time was 1.0 hours,  1.0 minutes, 20.41 seconds)
```

## Step 7: MobileNet Model

```
[ ] model_name='Mobilenet'
    base_model=tf.keras.applications.MobileNetV3Large(include_top=False, weights="imagenet",input_shape=img_shape, pooling='max')
    x=base_model.output
    x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
    x = Dense(256, kernel_regularizer = regularizers.12(1 = 0.016),activity_regularizer=regularizers.11(0.006),
                    bias_regularizer=regularizers.11(0.006) ,activation='relu')(x)
    x=Dropout(rate=.45, seed=123)(x)
    output=Dense(class_count, activation='softmax')(x)
    model=Model(inputs=base_model.input, outputs=output)
    model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v3/weights_mobilenet_v3_large_224_1.0_float_no_top_v2.h5
    12683000/12683000 [==============================] - 2s 0us/step
```

```
    epochs =25
    patience= 1 # number of epochs to wait to adjust lr if monitored value does not improve
    stop_patience =3 # number of epochs to wait before stopping training if monitored value does not improve
    threshold=.9 # if train accuracy is < threshhold adjust monitor accuracy, else monitor validation loss
    factor=.5 # factor to reduce lr by
    dwell=True # experimental, if True and monitored metric does not improve on current epoch set  modelweights back to weights of previous epoch
    ask_epoch=5 # number of epochs to run before asking if you want to halt training
    batches=train_steps
    with tf.device("/gpu:0"):
        callbacks=[LRA(model=model,base_model= base_model,patience=patience,stop_patience=stop_patience, threshold=threshold,
                       factor=factor,dwell=dwell, batches=batches,initial_epoch=0,epochs=epochs, ask_epoch=ask_epoch )]
        history=model.fit(x=train_gen, epochs=epochs, verbose=1,callbacks=callbacks, validation_data=valid_gen,validation_steps=None,  shuffle=False,  initial_epoch=0)

    initializing callback starting train with base_model trainable

    Epoch     Loss   Accuracy  V_loss    V_acc     LR      Next LR  Monitor  Duration

    Epoch 1/25
    1804/1804 [==============================] - ETA: 0s - loss: 4.0526 - accuracy: 0.5244 1 /25     4.053   52.439  1430.91504 19.614   0.00100  0.00100  accuracy   97.15

    1804/1804 [==============================] - 97s 40ms/step - loss: 4.0526 - accuracy: 0.5244 - val_loss: 1430.9150 - val_accuracy: 0.1961
    Epoch 2/25
    1804/1804 [==============================] - ETA: 0s - loss: 1.5878 - accuracy: 0.6234 2 /25     1.588   62.343   6.24650  41.158   0.00100  0.00100  accuracy   68.52
```

```
    1804/1804 [==============================] - 90s 50ms/step - loss: 1.1238 - accuracy: 0.7228 - val_loss: 4.2557 - val_accuracy: 0.4469
    Epoch 6/25
     6 /25     1.115   73.060   1.35988  70.096   0.00050  0.00050  accuracy   69.07

    1804/1804 [==============================] - 69s 38ms/step - loss: 1.1147 - accuracy: 0.7306 - val_loss: 1.3599 - val_accuracy: 0.7010
    Epoch 7/25
     7 /25     1.097   73.670   1.19840  76.849   0.00050  0.00050  accuracy   68.80

    1804/1804 [==============================] - 69s 38ms/step - loss: 1.0968 - accuracy: 0.7367 - val_loss: 1.1984 - val_accuracy: 0.7685
    Epoch 8/25
     8 /25     1.012   76.755   1.23353  81.029   0.00050  0.00050  accuracy   68.07

    1804/1804 [==============================] - 68s 38ms/step - loss: 1.0119 - accuracy: 0.7676 - val_loss: 1.2335 - val_accuracy: 0.8103
    Epoch 9/25
     9 /25     1.013   76.885   1.31004  73.955   0.00050  0.00050  accuracy   67.98

    1804/1804 [==============================] - 68s 38ms/step - loss: 1.0134 - accuracy: 0.7688 - val_loss: 1.3100 - val_accuracy: 0.7395
    Epoch 10/25
    10 /25     0.969   78.677   3.39305  62.379   0.00050  0.00050  accuracy   68.32

    1804/1804 [==============================] - 68s 38ms/step - loss: 0.9693 - accuracy: 0.7868 - val_loss: 3.3930 - val_accuracy: 0.6238
    Epoch 11/25
    11 /25     0.972   78.714   1.26361  74.277   0.00050  0.00050  accuracy   68.84

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.9718 - accuracy: 0.7871 - val_loss: 1.2636 - val_accuracy: 0.7428
    Epoch 12/25
    1804/1804 [==============================] - ETA: 0s - loss: 0.9254 - accuracy: 0.792912 /25     0.925   79.287   2.98044  76.849   0.00050  0.00050  accuracy   68.42

    1804/1804 [==============================] - 68s 38ms/step - loss: 0.9254 - accuracy: 0.7929 - val_loss: 2.9804 - val_accuracy: 0.7685
    Epoch 13/25
    1804/1804 [==============================] - ETA: 0s - loss: 0.8688 - accuracy: 0.807313 /25     0.869   80.728   6.76241  52.090   0.00050  0.00050  accuracy   67.91

    1804/1804 [==============================] - 68s 38ms/step - loss: 0.8688 - accuracy: 0.8073 - val_loss: 6.7624 - val_accuracy: 0.5209
    Epoch 14/25
    1804/1804 [==============================] - ETA: 0s - loss: 0.8544 - accuracy: 0.817814 /25     0.854   81.781   1.93189  64.309   0.00050  0.00050  accuracy   68.31

    1804/1804 [==============================] - 68s 38ms/step - loss: 0.8544 - accuracy: 0.8178 - val_loss: 1.9319 - val_accuracy: 0.6431
    Epoch 15/25
    15 /25     0.830   82.409   1.28377  75.563   0.00050  0.00050  accuracy   67.68
```

```
    Epoch 16/25
    16 /25     0.844   82.373   1.42630  77.492   0.00050  0.00025  accuracy   69.28

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.8443 - accuracy: 0.8237 - val_loss: 1.4263 - val_accuracy: 0.7749
    Epoch 17/25
    17 /25     0.648   87.177   0.72405  88.746   0.00025  0.00025  accuracy   68.83

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.6480 - accuracy: 0.8718 - val_loss: 0.7240 - val_accuracy: 0.8875
    Epoch 18/25
    18 /25     0.601   87.121   0.76454  82.637   0.00025  0.00013  accuracy   70.01

    1804/1804 [==============================] - 70s 39ms/step - loss: 0.6015 - accuracy: 0.8712 - val_loss: 0.7645 - val_accuracy: 0.8264
    Epoch 19/25
    1804/1804 [==============================] - ETA: 0s - loss: 0.5279 - accuracy: 0.896919 /25     0.528   89.690   0.57119  91.318   0.00013  0.00013  accuracy   72.28

    1804/1804 [==============================] - 72s 40ms/step - loss: 0.5279 - accuracy: 0.8969 - val_loss: 0.5712 - val_accuracy: 0.9132
    Epoch 20/25
    20 /25     0.488   90.059   0.72577  88.746   0.00013  0.00006  val_loss   70.32

    1804/1804 [==============================] - 70s 39ms/step - loss: 0.4878 - accuracy: 0.9006 - val_loss: 0.7258 - val_accuracy: 0.8875
    Epoch 21/25
    1804/1804 [==============================] - ETA: 0s - loss: 0.4852 - accuracy: 0.902621 /25     0.485   90.262   0.55740  90.997   0.00006  0.00006  val_loss   68.90

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.4852 - accuracy: 0.9026 - val_loss: 0.5574 - val_accuracy: 0.9100
    Epoch 22/25
    22 /25     0.460   91.020   0.63282  89.711   0.00006  0.00003  val_loss   68.74

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.4601 - accuracy: 0.9102 - val_loss: 0.6328 - val_accuracy: 0.8971
    Epoch 23/25
    23 /25     0.458   90.983   0.55801  90.675   0.00003  0.00002  val_loss   68.89

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.4575 - accuracy: 0.9098 - val_loss: 0.5580 - val_accuracy: 0.9068
    Epoch 24/25
    1804/1804 [==============================] - ETA: 0s - loss: 0.4328 - accuracy: 0.917224 /25     0.433   91.722   0.62760  89.711   0.00002  0.00001  val_loss   68.51

    training has been halted at epoch 24 after 3 adjustments of learning rate with no improvement

    1804/1804 [==============================] - 69s 38ms/step - loss: 0.4328 - accuracy: 0.9172 - val_loss: 0.6276 - val_accuracy: 0.8971
    Training is completed - model is set with weights from epoch 21

    training elapsed time was 0.0 hours, 31.0 minutes, 51.97 seconds)
```

## Step 8: InceptionV3 Model

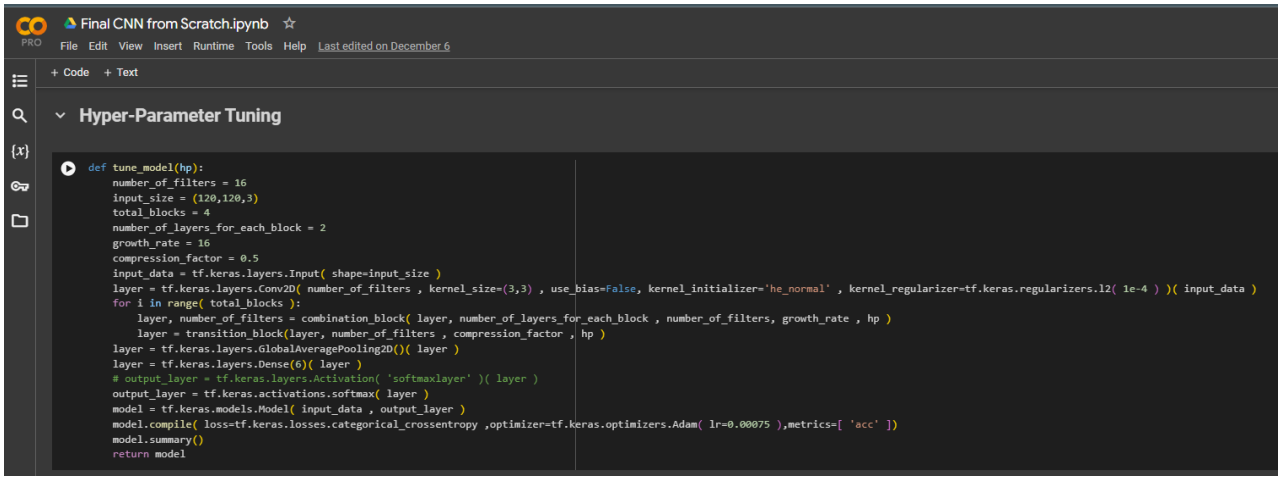
```
model_name='Incpetion'
base_model=tf.keras.applications.InceptionV3(include_top=False, weights="imagenet",input_shape=img_shape, pooling='max')
x=base_model.output
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(256, kernel_regularizer = regularizers.l2(l = 0.016),activity_regularizer=regularizers.l1(0.006),
                bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.45, seed=123)(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)
model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=['accuracy'])
```

```
epochs =25
patience= 1 # number of epochs to wait to adjust lr if monitored value does not improve
stop_patience =3 # number of epochs to wait before stopping training if monitored value does not improve
threshold=.9 # if train accuracy is < threshhold adjust monitor accuracy, else monitor validation loss
factor=.5 # factor to reduce lr by
dwell=True # experimental, if True and monitored metric does not improve on current epoch set  modelweights back to weights of previous epoch
ask_epoch=5 # number of epochs to run before asking if you want to halt training
batches=train_steps
with tf.device("/gpu:0"):
    callbacks=[LRA(model=model,base_model= base_model,patience=patience,stop_patience=stop_patience, threshold=threshold,
                   factor=factor,dwell=dwell, batches=batches,initial_epoch=0,epochs=epochs, ask_epoch=ask_epoch )]
    history=model.fit(x=train_gen, epochs=epochs, verbose=1,callbacks=callbacks, validation_data=valid_gen,validation_steps=None,  shuffle=False,  initial_epoch=0)
```

```
 initializing callback starting train with base_model trainable

 Epoch     Loss   Accuracy  V_loss    V_acc     LR      Next LR  Monitor  Duration

Epoch 1/25
1804/1804 [==============================] - ETA: 0s - loss: 4.8805 - accuracy: 0.2336 1 /25     4.880   23.356  34.59099  26.367   0.00100  0.00100  accuracy   125.20

1804/1804 [==============================] - 125s 55ms/step - loss: 4.8805 - accuracy: 0.2336 - val_loss: 34.5910 - val_accuracy: 0.2637
Epoch 2/25
1804/1804 [==============================] - ETA: 0s - loss: 1.9013 - accuracy: 0.2448 2 /25     1.901   24.483   1.79904  36.334   0.00100  0.00100  accuracy   97.67

1804/1804 [==============================] - 98s 54ms/step - loss: 1.9013 - accuracy: 0.2448 - val_loss: 1.7990 - val_accuracy: 0.3633
Epoch 3/25
1804/1804 [==============================] - ETA: 0s - loss: 1.9045 - accuracy: 0.2363 3 /25     1.904   23.633  10.78722  35.691   0.00100  0.00050  accuracy   97.61

1804/1804 [==============================] - 98s 54ms/step - loss: 1.9045 - accuracy: 0.2363 - val_loss: 10.7872 - val_accuracy: 0.3569
Epoch 4/25
1804/1804 [==============================] - ETA: 0s - loss: 1.8743 - accuracy: 0.2387 4 /25     1.874   23.873   2.93763  36.977   0.00050  0.00025  accuracy   97.21

1804/1804 [==============================] - 97s 54ms/step - loss: 1.8743 - accuracy: 0.2387 - val_loss: 2.9376 - val_accuracy: 0.3698
Epoch 5/25
1804/1804 [==============================] - ETA: 0s - loss: 1.7967 - accuracy: 0.2773 5 /25     1.797   27.735   1.80806  25.402   0.00025  0.00025  accuracy   97.58
```

```
 Epoch     Loss   Accuracy  V_loss    V_acc     LR      Next LR  Monitor  Duration

1804/1804 [==============================] - 1230s 682ms/step - loss: 1.3658 - accuracy: 0.5009 - val_loss: 1.7913 - val_accuracy: 0.5402
Epoch 16/25
1804/1804 [==============================] - ETA: 0s - loss: 1.3608 - accuracy: 0.506116 /25     1.361   50.610   1.85719  50.804   0.00013  0.00013  accuracy   99.08

1804/1804 [==============================] - 99s 55ms/step - loss: 1.3608 - accuracy: 0.5061 - val_loss: 1.8572 - val_accuracy: 0.5080
Epoch 17/25
1804/1804 [==============================] - ETA: 0s - loss: 1.3129 - accuracy: 0.535117 /25     1.313   53.511   1.38821  54.341   0.00013  0.00013  accuracy   100.17

1804/1804 [==============================] - 100s 56ms/step - loss: 1.3129 - accuracy: 0.5351 - val_loss: 1.3882 - val_accuracy: 0.5434
Epoch 18/25
1804/1804 [==============================] - ETA: 0s - loss: 1.3357 - accuracy: 0.535518 /25     1.336   53.548   1.36161  56.913   0.00013  0.00013  accuracy   100.50

1804/1804 [==============================] - 101s 56ms/step - loss: 1.3357 - accuracy: 0.5355 - val_loss: 1.3616 - val_accuracy: 0.5691
Epoch 19/25
1804/1804 [==============================] - ETA: 0s - loss: 1.2931 - accuracy: 0.544719 /25     1.293   54.472   1.64919  52.090   0.00013  0.00013  accuracy   100.22

1804/1804 [==============================] - 100s 56ms/step - loss: 1.2931 - accuracy: 0.5447 - val_loss: 1.6492 - val_accuracy: 0.5209
Epoch 20/25
1804/1804 [==============================] - ETA: 0s - loss: 1.2629 - accuracy: 0.553220 /25     1.263   55.322   1.30955  57.235   0.00013  0.00013  accuracy   99.94

1804/1804 [==============================] - 100s 55ms/step - loss: 1.2629 - accuracy: 0.5532 - val_loss: 1.3096 - val_accuracy: 0.5723
Epoch 21/25
1804/1804 [==============================] - ETA: 0s - loss: 1.2417 - accuracy: 0.573421 /25     1.242   57.336   1.32284  59.164   0.00013  0.00013  accuracy   99.79

1804/1804 [==============================] - 100s 55ms/step - loss: 1.2417 - accuracy: 0.5734 - val_loss: 1.3228 - val_accuracy: 0.5916
Epoch 22/25
1804/1804 [==============================] - ETA: 0s - loss: 1.2191 - accuracy: 0.580222 /25     1.219   58.019   1.31985  61.093   0.00013  0.00013  accuracy   99.62

1804/1804 [==============================] - 100s 55ms/step - loss: 1.2191 - accuracy: 0.5802 - val_loss: 1.3198 - val_accuracy: 0.6109
Epoch 23/25
1804/1804 [==============================] - ETA: 0s - loss: 1.1714 - accuracy: 0.607223 /25     1.171   60.717   1.32332  58.842   0.00013  0.00013  accuracy   99.67

1804/1804 [==============================] - 100s 55ms/step - loss: 1.1714 - accuracy: 0.6072 - val_loss: 1.3233 - val_accuracy: 0.5884
Epoch 24/25
1804/1804 [==============================] - ETA: 0s - loss: 1.1793 - accuracy: 0.602724 /25     1.179   60.273   1.17083  62.701   0.00013  0.00006  accuracy   99.61

1804/1804 [==============================] - 100s 55ms/step - loss: 1.1793 - accuracy: 0.6027 - val_loss: 1.1708 - val_accuracy: 0.6270
Epoch 25/25
1804/1804 [==============================] - ETA: 0s - loss: 1.1181 - accuracy: 0.626625 /25     1.118   62.657   0.99065  69.775   0.00006  0.00006  accuracy   99.48

enter H to halt ,F to fine tune model, or an integer for number of epochs to run then ask again

F
setting base_model as trainable for fine tuning of model

 Epoch     Loss   Accuracy  V_loss    V_acc     LR      Next LR  Monitor  Duration

1804/1804 [==============================] - 237s 131ms/step - loss: 1.1181 - accuracy: 0.6266 - val_loss: 0.9907 - val_accuracy: 0.6977
Training is completed - model is set with weights from epoch 25

training elapsed time was 1.0 hours,  5.0 minutes, 36.59 seconds)
```

Step 9: New CNN Model, Hyper-Parameter tuning has been performed for the model.




```
[ ]  def hyper_parameter_tuner():
         tuner = kt.Hyperband(hypermodel=tune_model,
                              objective='val_loss',
                              max_epochs=20,
                              factor=3,
                              hyperband_iterations=1,
                              seed=seed,
                              overwrite=True)

         early_stopping = tf.keras.callbacks.EarlyStopping(patience=10)

         tuner.search(train_gen, epochs=20,
                      validation_data= valid_gen,
                      verbose=1, callbacks=[early_stopping])


         return tuner
```

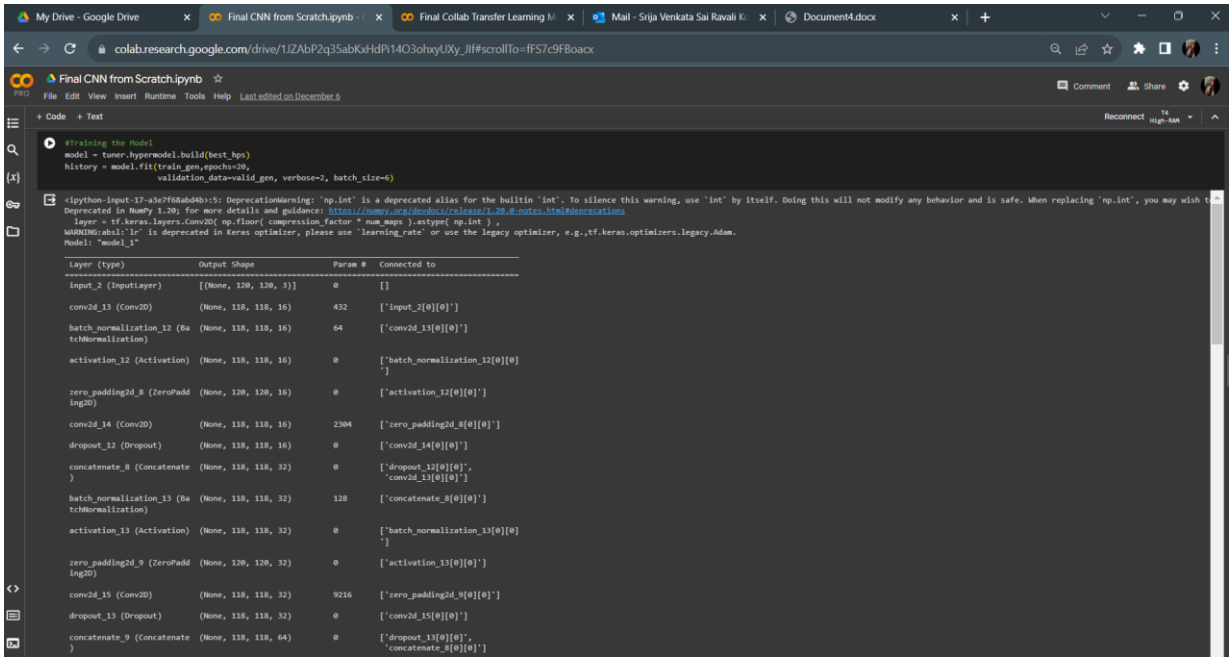
```
tuner = hyper_parameter_tuner()
```

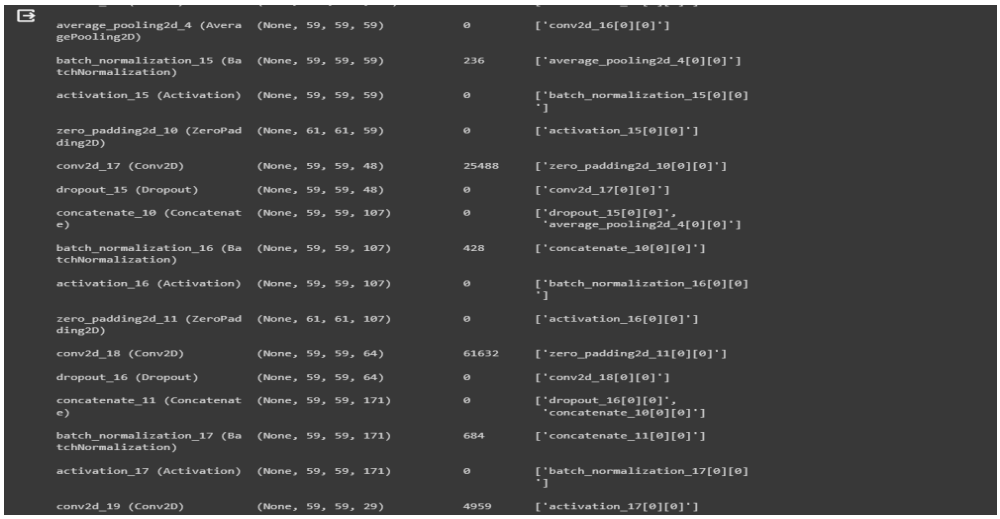
```
Trial 4 Complete [00h 02m 29s]
val_loss: 1.0628324747085571

Best val_loss So Far: 1.0387638807296753
Total elapsed time: 01h 15m 12s
```

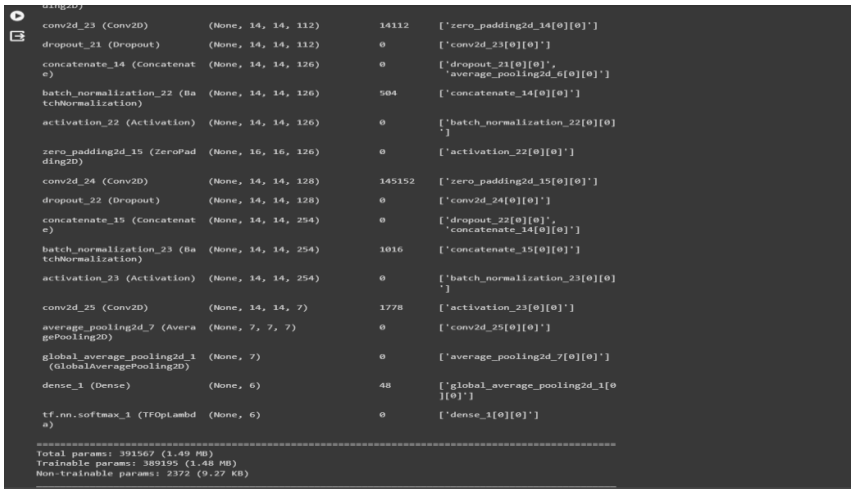
```
[ ]  #tuner = hyper_parameter_tuner()
     best_hps = tuner.get_best_hyperparameters(num_trials = 1)[0]
     print(best_hps)
     print(best_hps.get('dp_1'))
```

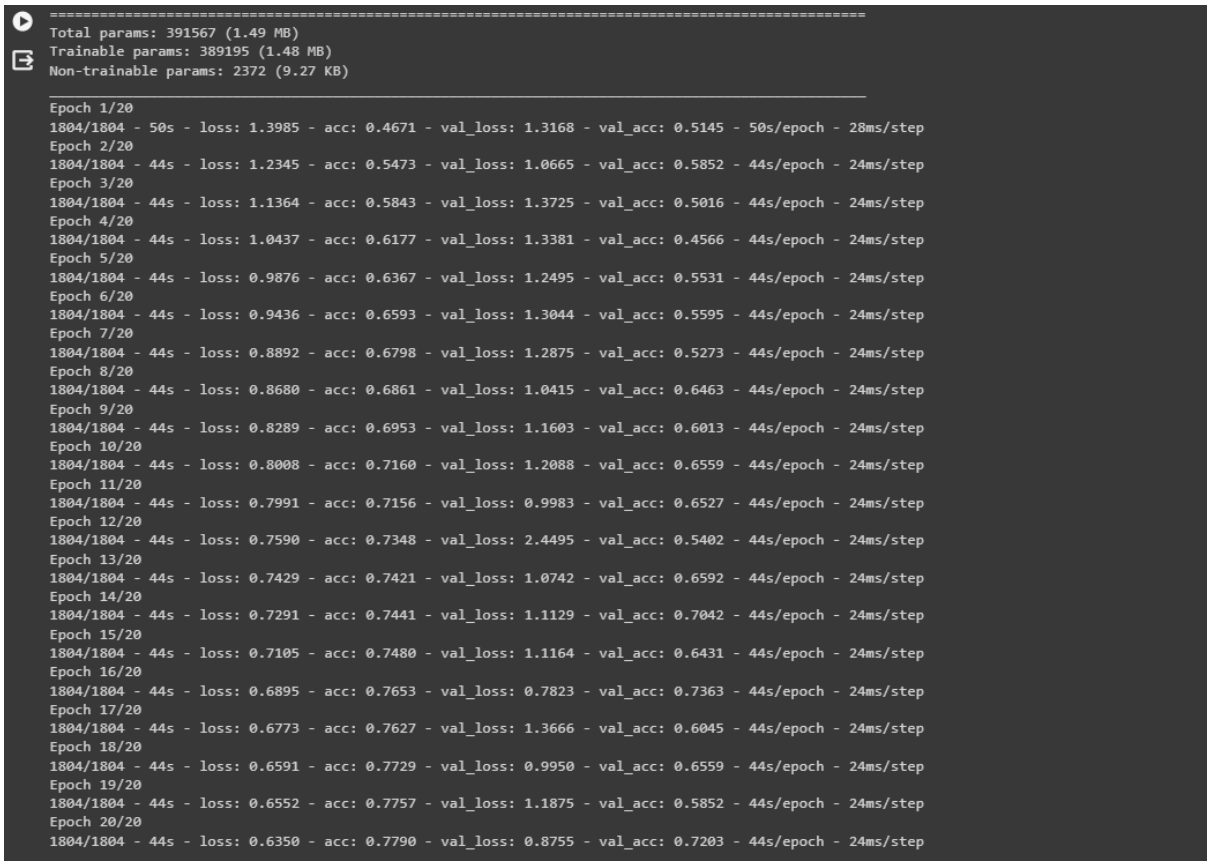
```
<keras_tuner.src.engine.hyperparameters.hyperparameters.HyperParameters object at 0x7a8514bdd630>
0.3
```

Step 10: Training the New CNN model.





```
=================================================================================
Total params: 391567 (1.49 MB)
Trainable params: 389195 (1.48 MB)
Non-trainable params: 2372 (9.27 KB)
_________________________________________________________________________________
Epoch 1/20
1804/1804 - 50s - loss: 1.3985 - acc: 0.4671 - val_loss: 1.3168 - val_acc: 0.5145 - 50s/epoch - 28ms/step
Epoch 2/20
1804/1804 - 44s - loss: 1.2345 - acc: 0.5473 - val_loss: 1.0665 - val_acc: 0.5852 - 44s/epoch - 24ms/step
Epoch 3/20
1804/1804 - 44s - loss: 1.1364 - acc: 0.5843 - val_loss: 1.3725 - val_acc: 0.5016 - 44s/epoch - 24ms/step
Epoch 4/20
1804/1804 - 44s - loss: 1.0437 - acc: 0.6177 - val_loss: 1.3381 - val_acc: 0.4566 - 44s/epoch - 24ms/step
Epoch 5/20
1804/1804 - 44s - loss: 0.9876 - acc: 0.6367 - val_loss: 1.2495 - val_acc: 0.5531 - 44s/epoch - 24ms/step
Epoch 6/20
1804/1804 - 44s - loss: 0.9436 - acc: 0.6593 - val_loss: 1.3044 - val_acc: 0.5595 - 44s/epoch - 24ms/step
Epoch 7/20
1804/1804 - 44s - loss: 0.8892 - acc: 0.6798 - val_loss: 1.2875 - val_acc: 0.5273 - 44s/epoch - 24ms/step
Epoch 8/20
1804/1804 - 44s - loss: 0.8680 - acc: 0.6861 - val_loss: 1.0415 - val_acc: 0.6463 - 44s/epoch - 24ms/step
Epoch 9/20
1804/1804 - 44s - loss: 0.8289 - acc: 0.6953 - val_loss: 1.1603 - val_acc: 0.6013 - 44s/epoch - 24ms/step
Epoch 10/20
1804/1804 - 44s - loss: 0.8008 - acc: 0.7160 - val_loss: 1.2088 - val_acc: 0.6559 - 44s/epoch - 24ms/step
Epoch 11/20
1804/1804 - 44s - loss: 0.7991 - acc: 0.7156 - val_loss: 0.9983 - val_acc: 0.6527 - 44s/epoch - 24ms/step
Epoch 12/20
1804/1804 - 44s - loss: 0.7590 - acc: 0.7348 - val_loss: 2.4495 - val_acc: 0.5402 - 44s/epoch - 24ms/step
Epoch 13/20
1804/1804 - 44s - loss: 0.7429 - acc: 0.7421 - val_loss: 1.0742 - val_acc: 0.6592 - 44s/epoch - 24ms/step
Epoch 14/20
1804/1804 - 44s - loss: 0.7291 - acc: 0.7441 - val_loss: 1.1129 - val_acc: 0.7042 - 44s/epoch - 24ms/step
Epoch 15/20
1804/1804 - 44s - loss: 0.7105 - acc: 0.7480 - val_loss: 1.1164 - val_acc: 0.6431 - 44s/epoch - 24ms/step
Epoch 16/20
1804/1804 - 44s - loss: 0.6895 - acc: 0.7653 - val_loss: 0.7823 - val_acc: 0.7363 - 44s/epoch - 24ms/step
Epoch 17/20
1804/1804 - 44s - loss: 0.6773 - acc: 0.7627 - val_loss: 1.3666 - val_acc: 0.6045 - 44s/epoch - 24ms/step
Epoch 18/20
1804/1804 - 44s - loss: 0.6591 - acc: 0.7729 - val_loss: 0.9950 - val_acc: 0.6559 - 44s/epoch - 24ms/step
Epoch 19/20
1804/1804 - 44s - loss: 0.6552 - acc: 0.7757 - val_loss: 1.1875 - val_acc: 0.5852 - 44s/epoch - 24ms/step
Epoch 20/20
1804/1804 - 44s - loss: 0.6350 - acc: 0.7790 - val_loss: 0.8755 - val_acc: 0.7203 - 44s/epoch - 24ms/step
```