FINAL_PROJECT

August 12, 2020

0.1 # Final Project

0.2 ## State of the art CNN architecture over imagenet dataset

- **Duke Mervyn Martin (dukemerv@buffalo.edu)**
- **Preeti Kumari (preetiku@buffalo.edu)**
- **Kizito Nwaka (kizitonw@buffalo.edu)**

[1]:
```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, MaxPooling2D, Flatten, Conv2D, LeakyReLU,␣
↪Activation, BatchNormalization, Dropout
from keras import optimizers,regularizers
from keras.utils import plot_model
from keras.optimizers import SGD
from keras import backend
import numpy as np
import time as t
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import os
import sys
os.environ["CUDA_VISIBLE_DEVICES"]="0"
#tf.debugging.set_log_device_placement(True)
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings(action='once')
import cv2
import matplotlib.image as mpimg
from keras.utils import plot_model
```

[2]:
```
print(f"Tensor Flow Version: {tf.__version__}")
print(f"Keras Version: {keras.__version__}")
print("GPU is", "available" if tf.test.is_gpu_available() else "NOT AVAILABLE")
print("Num GPUs Available: ", len(tf.config.experimental.
↪list_physical_devices('GPU')))
```

```
Tensor Flow Version: 2.3.0
Keras Version: 2.4.0
WARNING:tensorflow:From <ipython-input-2-227a307d88ee>:3: is_gpu_available (from
tensorflow.python.framework.test_util) is deprecated and will be removed in a
future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
GPU is available
Num GPUs Available:  1
```

0.2.1 Choosing classes Randomly and Downloading 1500 Images per class

From the library ImageNet-Datasets-Downloader we use the `downloader.py` with the command given below to download all the Images respective to its classes in each folder inside the directory `imagenet/`

```
python ./downloader.py -data_root imagenet -number_of_classes 8 -images_per_class 1500
```

List of Classes

[4]:
```python
from os import listdir
classes = listdir("imagenet")
print('List of Classes ->',classes)
print("Number of classes =",len(classes))
```

```
List of Classes -> ['koala', 'ladybug', 'lichen', 'meerkat', 'Rhodesian
ridgeback', 'tamandua', 'vizsla', 'Yorkshire terrier']
Number of classes = 8
```

0.2.2 Preparing the Datasets

As all the images are of different sizes, we will be generating a square image of size **n x n**

This can be acheived by two steps - Scale down the image to n pixels by its shortest side - Crop the middle part of the image to get n x n pixels square (This method of data preparation is used by AlexNet model, VGG-16 & VGG-19)

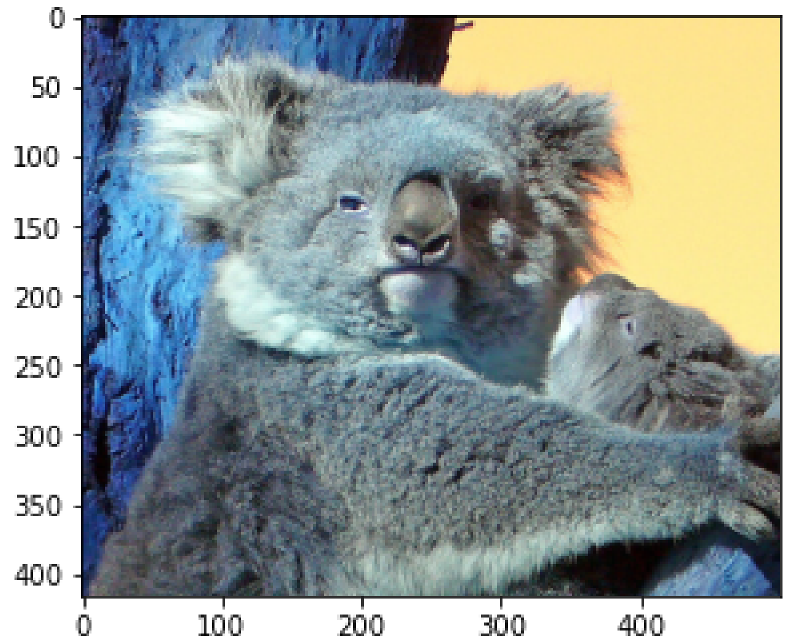
[2]:
```python
n = 128
```

0.3 Image process demo

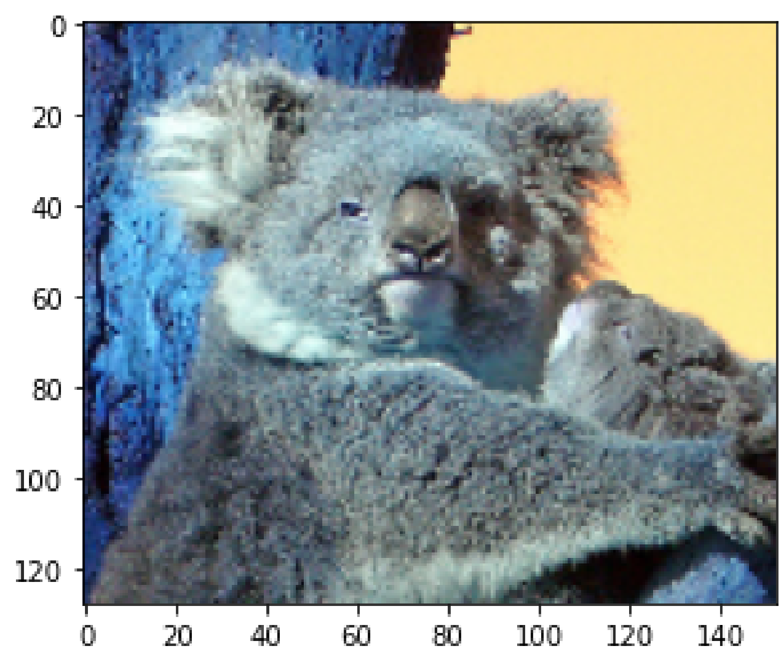
[5]:
```python
img = cv2.imread('imagenet/koala/218758752_650f6b9b5a.jpg')
print('Image Before Processing')
print('-----------------------')
plt.imshow(img)
plt.show()
print(img.shape)
print('Image After Resizing')
print('-----------------------')
if(img.shape[0]>img.shape[1]):
```

```python
    width = n
    height = int(img.shape[0] * (n/img.shape[1]))
    img = cv2.resize(img,(width,height))
    plt.imshow(img)
    plt.show()
    print(img.shape)
else:
    width = int(img.shape[1] * (n/img.shape[0]))
    height = n
    img = cv2.resize(img,(width,height))
    plt.imshow(img)
    plt.show()
    print(img.shape)
print('Image After Croping')
print('-----------------------')
if(img.shape[0]==n):
    x = int((img.shape[1]-n)/2)
    img = img[0:,x:x+n]
    plt.imshow(img)
    print(img.shape)
else:
    x = int((img.shape[0]-n)/2)
    img = img[x:x+n,0:]
    plt.imshow(img)
    plt.show()
    print(img.shape)
```

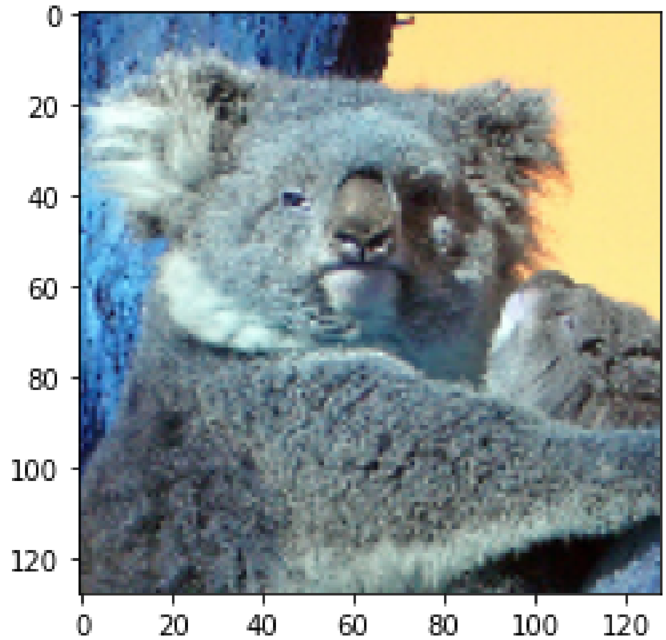
```
Image Before Processing
-----------------------
```



(416, 500, 3)
Image After Resizing



```
(128, 153, 3)
Image After Croping
-----------------------
(128, 128, 3)
```



Preprocessing Images

[4]:
```python
def img_process(image):
    img = image
    if(img.shape[0]>img.shape[1]):
        width = n
        height = int(img.shape[0] * (n/img.shape[1]))
        img = cv2.resize(img,(width,height))
    else:
        width = int(img.shape[1] * (n/img.shape[0]))
        height = n
        img = cv2.resize(img,(width,height))

    if(img.shape[0]==n):
        x = int((img.shape[1]-n)/2)
        img = img[0:,x:x+n]
    else:
        x = int((img.shape[0]-n)/2)
        img = img[x:x+n,0:]
    img = cv2.resize(img,(n,n))
    return img
```

Generate Target Labels

[7]:
```
labels= [] # Target Classes
for name in classes:
    for jpg in listdir("imagenet/"+name):
        labels.append([name])
```

Generate Colour Image Datasets after applying the preprocessing function

[8]:
```
labels=np.array(labels)
```

[9]:
```
labels.shape
```

[9]: (12000, 1)

[10]:
```
images = np.zeros((labels.shape[0],n,n,3)) #Empty Dataset
i=0
for name in classes:
    for jpg in listdir("imagenet/"+name):
        img = cv2.imread('imagenet/'+name+'/'+jpg)
        img = img_process(img)
        images[i] = img
        i+=1
print('Image data shape ->',images.shape)
```

```
Image data shape -> (12000, 128, 128, 3)
```

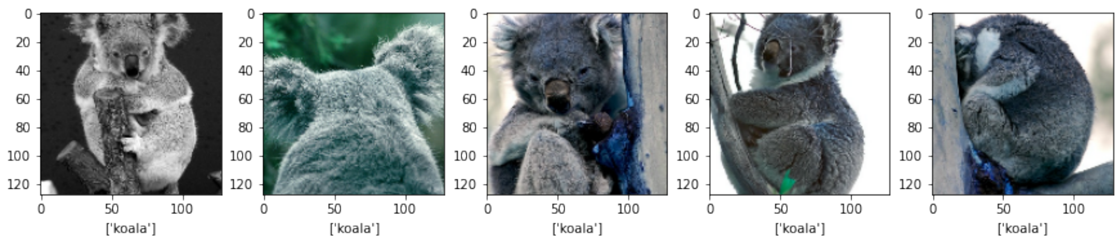
[11]:
```
# Save the Data for future use
np.save('image_data_128_color',images)
np.save('image_label_128_color',labels)
```

[4]:
```
image_data= np.load('image_data_128_color.npy')
labels= np.load('image_label_128_color.npy')
```

[5]:
```
print('Check Max Min before Normalization ->',image_data.min(),image_data.max())
image_data = image_data/255
print('Check Max Min after Normalization ->',image_data.min(),image_data.max())
```

```
Check Max Min before Normalization -> 0.0 255.0
Check Max Min after Normalization -> 0.0 1.0
```

[6]:
```
plt.figure(figsize=(15,15))
for i in range(5):
    plt.subplot(5,5,i+1)
    plt.imshow(image_data[i])
    plt.xlabel(labels[i])
plt.show()
```



0.3.1 Split Training and Testing Data

[8]:
```
from sklearn.utils import shuffle

image_data,labels = shuffle(image_data,labels)
np.save('shuffled_images_128',image_data)
np.save('shuffled_label_128',labels)
```

- We use a common shuffled data for analysing the performance of all the models, thus we reduce the effect of shuffling attributing to the accuracy of the model.

[4]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```

```
c:\users\dm97o\anaconda3\envs\tf_gpu\lib\importlib\_bootstrap.py:219:
RuntimeWarning: numpy.ufunc size changed, may indicate binary incompatibility.
Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

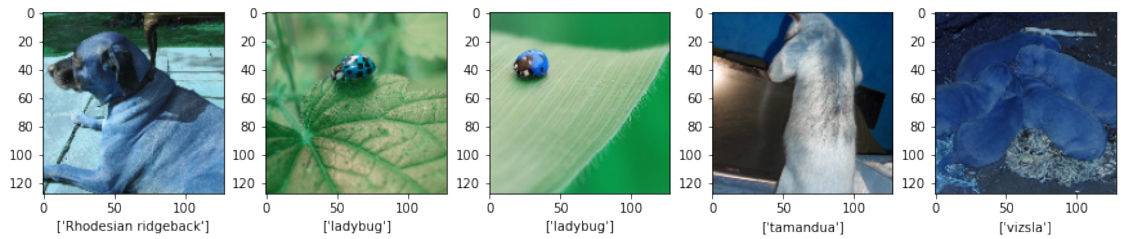
[5]:
```
images= np.load('shuffled_images_128.npy')
labels= np.load('shuffled_label_128.npy')
```

- We load the shuffled data and encode the labels using the onehot encoder.

[6]:
```
label_encoder = OneHotEncoder()
label_encoder.fit(labels)
label_encoded = label_encoder.transform(labels).toarray()
train_images, test_images = images[0:10000],images[10000:]
train_labels, test_labels = label_encoded[0:10000],label_encoded[10000:]
```

0.3.2 Some statistics about the data

[7]:
```
plt.figure(figsize=(15,15))
for i in range(5):
    plt.subplot(5,5,i+1)
    plt.imshow(train_images[i])
    plt.xlabel(labels[i])
plt.show()
```



[8]:
```
print(labels[:5])
print(label_encoded[:5])
```

```
[['Rhodesian ridgeback']
 ['ladybug']
 ['ladybug']
 ['tamandua']
 ['vizsla']]
[[1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
```

[9]:
```
print('Total Number of Entries ->',len(labels))
print('X Train Shape ->',train_images.shape)
print('Y Train Shape ->',train_labels.shape)
print('X Test Shape ->',test_images.shape)
print('Y Test Shape ->',test_labels.shape)
```

```
Total Number of Entries -> 12000
X Train Shape -> (10000, 128, 128, 3)
Y Train Shape -> (10000, 8)
X Test Shape -> (2000, 128, 128, 3)
Y Test Shape -> (2000, 8)
```

[8]:
```
 def graphs(history,score):

    plt.figure(figsize=(15,7))
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
```

```
    plt.title('Model accuracy',size=20)
    plt.ylabel('Accuracy',size=15)
    plt.xlabel('Epochs', size=15)
    plt.legend(['Training', 'Testing'], loc='best')
    plt.show()

    plt.figure(figsize=(15,7))
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss',size=20)
    plt.ylabel('Loss',size=15)
    plt.xlabel('Epochs',size=15)
    plt.legend(['Training', 'Testing'], loc='best')
    plt.show()

    print("Accuracy of the model: {:.2f}%".format(score[1]*100))
```

0.3.3 Building the network

[9]:
```
opt = SGD(lr=0.01,momentum=0.9,decay=0.01)
```

[10]:
```
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
early = EarlyStopping(monitor='val_loss', min_delta=0, patience=10,verbose=0,␣
 ↪mode='auto')
```

0.4 VGG - 16

0.5 ## VGG Archietectures

- **VGG model** was presented as an invetigation of the effect of depth on the accuracy of a CNN architecture on a large scale image recognition setting.
- The key idea is to use very small (3 x 3) filters and increasing the number of layers (depth of the network)
- We build and analyse the performance of the VGG - 16 on the MNIST digit dataset.

0.5.1 VGG - 16

- As the name says, it consits of 16 weight layers (13 convolution and 3 Fully connected layers). The original paper used 224 x 224 colored images of the ImageNet dataset (ILSVRC).
- The original model took very less epochs to converge when compared to the other proposals even with a shallow network due to the ***implicit regularisation imposed by greater depth and smaller conv filter sizes***.
- The images were randomly cropped and flipped both horizontally & verticaly to provide data augumentation.
- We implement VGG-16 on a more simpler MNIST digit dataset.

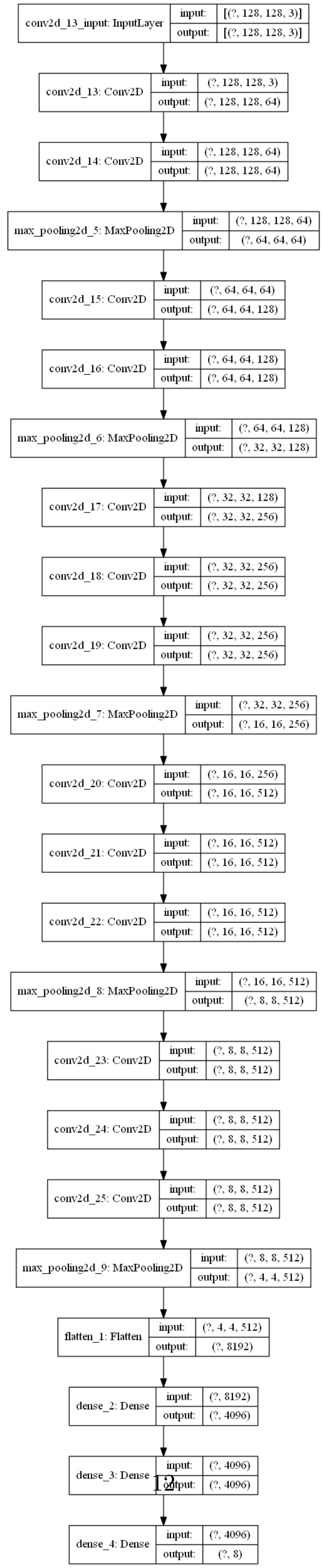
[5]:

```python
model = Sequential()
model.
 ↪add(Conv2D(input_shape=(n,n,3),filters=64,kernel_size=(3,3),padding="same",␣
 ↪activation="relu"))
model.add(Conv2D(64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Flatten())
model.add(Dense(4096,activation="relu"))
model.add(Dense(4096,activation="relu"))
model.add(Dense(units=len(classes), activation="softmax"))
model.summary()
plot_model(model, show_shapes=True, to_file='VGG_16.png')
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 128, 128, 64)      1792
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 64, 64, 128)       73856
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 32, 32, 256)       295168
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 32, 32, 256)       590080
```

```
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 16, 16, 512)       1180160
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 16, 16, 512)       2359808
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 16, 16, 512)       2359808
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              33558528
_________________________________________________________________
dense_3 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_4 (Dense)              (None, 8)                 32776
=================================================================
Total params: 65,087,304
Trainable params: 65,087,304
Non-trainable params: 0
_________________________________________________________________
```

[5]:


conv2d_13_input: InputLayer
input: [(?, 128, 128, 3)]
output: [(?, 128, 128, 3)]
conv2d_13: Conv2D
input: (?, 128, 128, 3)
output: (?, 128, 128, 64)
conv2d_14: Conv2D
input: (?, 128, 128, 64)
output: (?, 128, 128, 64)
max_pooling2d_5: MaxPooling2D
input: (?, 128, 128, 64)
output: (?, 64, 64, 64)
conv2d_15: Conv2D
input: (?, 64, 64, 64)
output: (?, 64, 64, 128)
conv2d_16: Conv2D
input: (?, 64, 64, 128)
output: (?, 64, 64, 128)
max_pooling2d_6: MaxPooling2D
input: (?, 64, 64, 128)
output: (?, 32, 32, 128)
conv2d_17: Conv2D
input: (?, 32, 32, 128)
output: (?, 32, 32, 256)
conv2d_18: Conv2D
input: (?, 32, 32, 256)
output: (?, 32, 32, 256)
conv2d_19: Conv2D
input: (?, 32, 32, 256)
output: (?, 32, 32, 256)
max_pooling2d_7: MaxPooling2D
input: (?, 32, 32, 256)
output: (?, 16, 16, 256)
conv2d_20: Conv2D
input: (?, 16, 16, 256)
output: (?, 16, 16, 512)
conv2d_21: Conv2D
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
conv2d_22: Conv2D
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
max_pooling2d_8: MaxPooling2D
input: (?, 16, 16, 512)
output: (?, 8, 8, 512)
conv2d_23: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
conv2d_24: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
conv2d_25: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
max_pooling2d_9: MaxPooling2D
input: (?, 8, 8, 512)
output: (?, 4, 4, 512)
flatten_1: Flatten
input: (?, 4, 4, 512)
output: (?, 8192)
dense_2: Dense
input: (?, 8192)
output: (?, 4096)
dense_3: Dense
input: (?, 4096)
output: (?, 4096)
dense_4: Dense
input: (?, 4096)
output: (?, 8)

[12]:
```
model.
 ↪compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
```

[13]:
```
start_time=t.time()
history = model.fit(train_images, train_labels,
          validation_data=(test_images, test_labels),
          epochs=50, batch_size=64, callbacks=[early])
print("\n Training Time: {} seconds".format(t.time()-start_time))
model.save('vgg_16_n128_colour.h5')
```

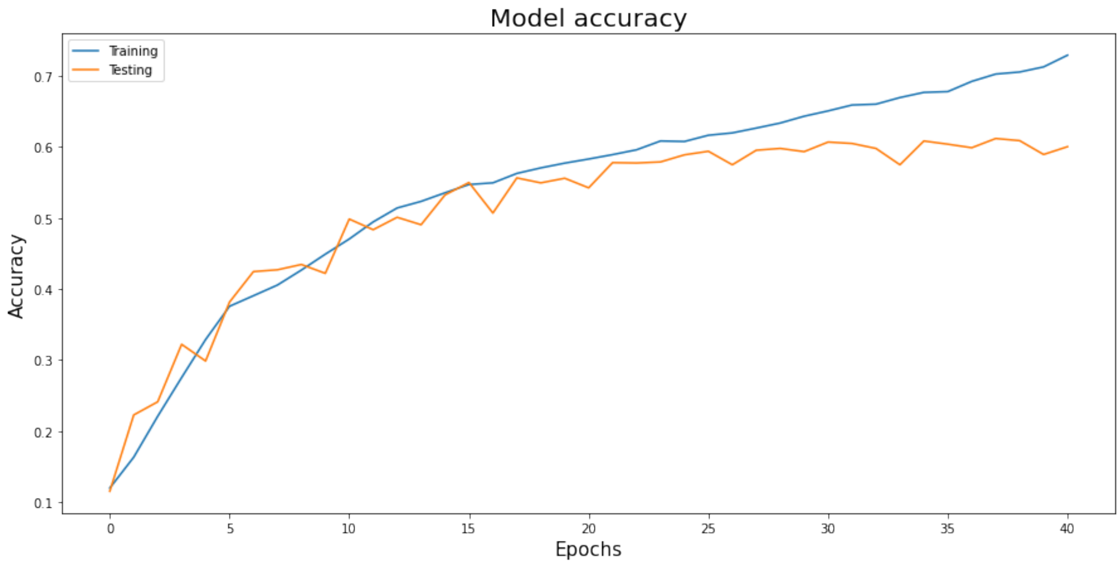
```
Epoch 1/50
  2/157 […] - ETA: 36s - loss: 2.0795 - accuracy:
0.1094WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.1237s vs `on_train_batch_end` time: 0.3431s).
Check your callbacks.
157/157 [==============================] - 79s 505ms/step - loss: 2.0793 -
accuracy: 0.1195 - val_loss: 2.0783 - val_accuracy: 0.1150
Epoch 2/50
157/157 [==============================] - 78s 498ms/step - loss: 2.0742 -
accuracy: 0.1630 - val_loss: 2.0662 - val_accuracy: 0.2225
Epoch 3/50
157/157 [==============================] - 79s 501ms/step - loss: 2.0235 -
accuracy: 0.2206 - val_loss: 1.9504 - val_accuracy: 0.2410
Epoch 4/50
157/157 [==============================] - 79s 504ms/step - loss: 1.8996 -
accuracy: 0.2752 - val_loss: 1.8147 - val_accuracy: 0.3220
Epoch 5/50
157/157 [==============================] - 79s 505ms/step - loss: 1.7712 -
accuracy: 0.3287 - val_loss: 1.8372 - val_accuracy: 0.2985
Epoch 6/50
157/157 [==============================] - 79s 505ms/step - loss: 1.6641 -
accuracy: 0.3756 - val_loss: 1.6350 - val_accuracy: 0.3815
Epoch 7/50
157/157 [==============================] - 79s 506ms/step - loss: 1.6170 -
accuracy: 0.3905 - val_loss: 1.5708 - val_accuracy: 0.4245
Epoch 8/50
157/157 [==============================] - 80s 507ms/step - loss: 1.5750 -
accuracy: 0.4055 - val_loss: 1.5524 - val_accuracy: 0.4270
Epoch 9/50
157/157 [==============================] - 80s 509ms/step - loss: 1.5326 -
accuracy: 0.4266 - val_loss: 1.5331 - val_accuracy: 0.4345
Epoch 10/50
157/157 [==============================] - 80s 510ms/step - loss: 1.4652 -
accuracy: 0.4490 - val_loss: 1.6354 - val_accuracy: 0.4220
Epoch 11/50
```

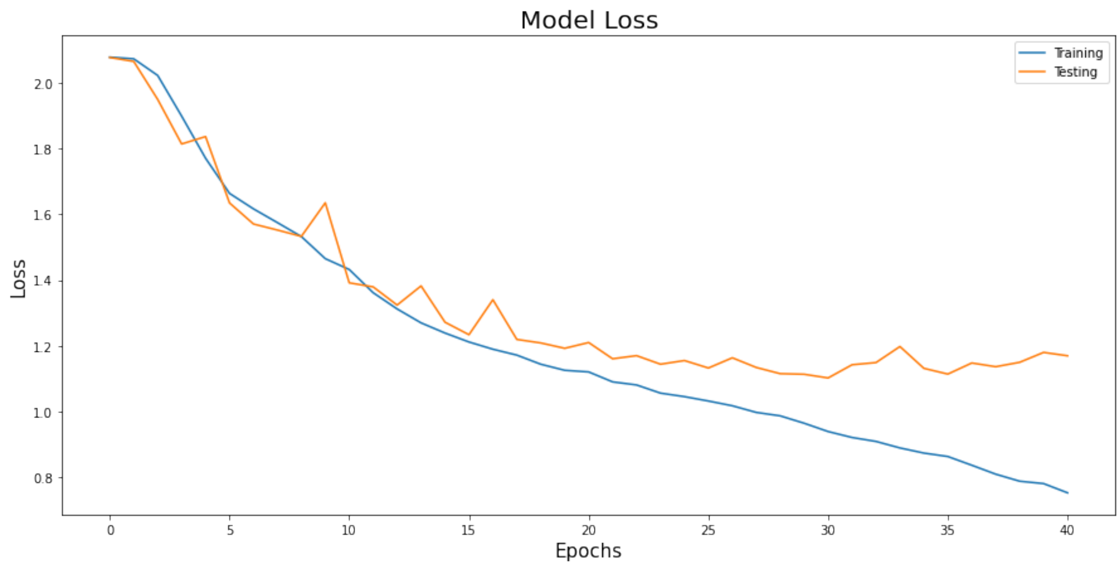
```
157/157 [==============================] - 80s 508ms/step - loss: 1.4322 -
accuracy: 0.4704 - val_loss: 1.3916 - val_accuracy: 0.4985
Epoch 12/50
157/157 [==============================] - 80s 507ms/step - loss: 1.3614 -
accuracy: 0.4945 - val_loss: 1.3791 - val_accuracy: 0.4835
Epoch 13/50
157/157 [==============================] - 80s 508ms/step - loss: 1.3120 -
accuracy: 0.5141 - val_loss: 1.3238 - val_accuracy: 0.5010
Epoch 14/50
157/157 [==============================] - 80s 508ms/step - loss: 1.2695 -
accuracy: 0.5234 - val_loss: 1.3821 - val_accuracy: 0.4905
Epoch 15/50
157/157 [==============================] - 79s 506ms/step - loss: 1.2385 -
accuracy: 0.5355 - val_loss: 1.2715 - val_accuracy: 0.5325
Epoch 16/50
157/157 [==============================] - 79s 504ms/step - loss: 1.2113 -
accuracy: 0.5472 - val_loss: 1.2334 - val_accuracy: 0.5500
Epoch 17/50
157/157 [==============================] - 79s 502ms/step - loss: 1.1892 -
accuracy: 0.5495 - val_loss: 1.3401 - val_accuracy: 0.5070
Epoch 18/50
157/157 [==============================] - 79s 502ms/step - loss: 1.1713 -
accuracy: 0.5628 - val_loss: 1.2190 - val_accuracy: 0.5565
Epoch 19/50
157/157 [==============================] - 79s 501ms/step - loss: 1.1435 -
accuracy: 0.5706 - val_loss: 1.2083 - val_accuracy: 0.5495
Epoch 20/50
157/157 [==============================] - 79s 502ms/step - loss: 1.1250 -
accuracy: 0.5773 - val_loss: 1.1920 - val_accuracy: 0.5560
Epoch 21/50
157/157 [==============================] - 79s 502ms/step - loss: 1.1200 -
accuracy: 0.5831 - val_loss: 1.2096 - val_accuracy: 0.5425
Epoch 22/50
157/157 [==============================] - 79s 502ms/step - loss: 1.0897 -
accuracy: 0.5893 - val_loss: 1.1601 - val_accuracy: 0.5780
Epoch 23/50
157/157 [==============================] - 79s 502ms/step - loss: 1.0803 -
accuracy: 0.5961 - val_loss: 1.1696 - val_accuracy: 0.5775
Epoch 24/50
157/157 [==============================] - 79s 502ms/step - loss: 1.0554 -
accuracy: 0.6085 - val_loss: 1.1435 - val_accuracy: 0.5790
Epoch 25/50
157/157 [==============================] - 79s 502ms/step - loss: 1.0447 -
accuracy: 0.6078 - val_loss: 1.1547 - val_accuracy: 0.5890
Epoch 26/50
157/157 [==============================] - 79s 502ms/step - loss: 1.0314 -
accuracy: 0.6166 - val_loss: 1.1319 - val_accuracy: 0.5940
Epoch 27/50
```

```
157/157 [==============================] - 79s 502ms/step - loss: 1.0169 -
accuracy: 0.6199 - val_loss: 1.1631 - val_accuracy: 0.5750
Epoch 28/50
157/157 [==============================] - 79s 502ms/step - loss: 0.9966 -
accuracy: 0.6267 - val_loss: 1.1336 - val_accuracy: 0.5955
Epoch 29/50
157/157 [==============================] - 79s 502ms/step - loss: 0.9861 -
accuracy: 0.6338 - val_loss: 1.1147 - val_accuracy: 0.5980
Epoch 30/50
157/157 [==============================] - 79s 502ms/step - loss: 0.9637 -
accuracy: 0.6434 - val_loss: 1.1129 - val_accuracy: 0.5935
Epoch 31/50
157/157 [==============================] - 79s 501ms/step - loss: 0.9383 -
accuracy: 0.6509 - val_loss: 1.1016 - val_accuracy: 0.6070
Epoch 32/50
157/157 [==============================] - 79s 501ms/step - loss: 0.9202 -
accuracy: 0.6592 - val_loss: 1.1419 - val_accuracy: 0.6050
Epoch 33/50
157/157 [==============================] - 79s 501ms/step - loss: 0.9082 -
accuracy: 0.6604 - val_loss: 1.1485 - val_accuracy: 0.5980
Epoch 34/50
157/157 [==============================] - 79s 502ms/step - loss: 0.8884 -
accuracy: 0.6697 - val_loss: 1.1974 - val_accuracy: 0.5750
Epoch 35/50
157/157 [==============================] - 79s 501ms/step - loss: 0.8727 -
accuracy: 0.6770 - val_loss: 1.1309 - val_accuracy: 0.6085
Epoch 36/50
157/157 [==============================] - 79s 501ms/step - loss: 0.8622 -
accuracy: 0.6781 - val_loss: 1.1134 - val_accuracy: 0.6040
Epoch 37/50
157/157 [==============================] - 79s 502ms/step - loss: 0.8357 -
accuracy: 0.6925 - val_loss: 1.1472 - val_accuracy: 0.5990
Epoch 38/50
157/157 [==============================] - 79s 502ms/step - loss: 0.8084 -
accuracy: 0.7028 - val_loss: 1.1359 - val_accuracy: 0.6120
Epoch 39/50
157/157 [==============================] - 79s 502ms/step - loss: 0.7870 -
accuracy: 0.7057 - val_loss: 1.1493 - val_accuracy: 0.6090
Epoch 40/50
157/157 [==============================] - 79s 502ms/step - loss: 0.7797 -
accuracy: 0.7129 - val_loss: 1.1795 - val_accuracy: 0.5895
Epoch 41/50
157/157 [==============================] - 79s 501ms/step - loss: 0.7516 -
accuracy: 0.7294 - val_loss: 1.1692 - val_accuracy: 0.6005

 Training Time: 3267.4202382564545 seconds
```

[14]:
```
score = model.evaluate(test_images,test_labels,verbose=0)
print('Test Loss - ',score[0])
print('Test Accuracy - ',score[1])
graphs(history,score)
```

```
Test Loss -  1.1692144870758057
Test Accuracy -  0.6004999876022339
```






```
Accuracy of the model: 60.05%
```

0.6 ## VGG - 19

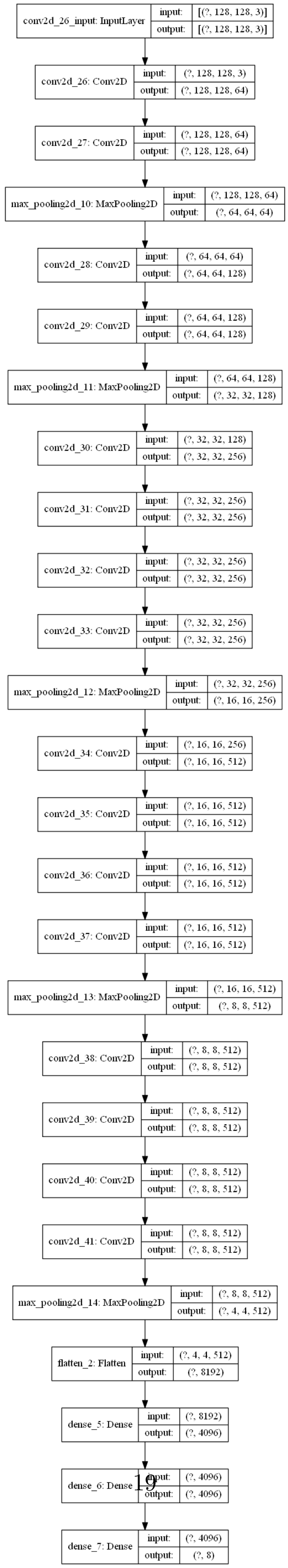
- It is a variation of the VGG-19, with 19 weight layers, 16 from convolution and 3 from the FC

[6]:
```
#vgg-19
model2 = Sequential()
model2.
 ↪add(Conv2D(input_shape=(n,n,3),filters=64,kernel_size=(3,3),padding="same",␣
 ↪activation="relu"))
model2.add(Conv2D(64,kernel_size=(3,3),padding="same", activation="relu"))
model2.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model2.add(Conv2D(128, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(128, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model2.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(256, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(Conv2D(512, kernel_size=(3,3), padding="same", activation="relu"))
model2.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model2.add(Flatten())
model2.add(Dense(4096,activation="relu"))
model2.add(Dense(4096,activation="relu"))
model2.add(Dense(units=len(classes), activation="softmax"))
model2.summary()
plot_model(model2, show_shapes=True, to_file='VGG_19.png')
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_26 (Conv2D)           (None, 128, 128, 64)      1792
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 64, 64, 128)       73856
```

```
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 32, 32, 256)       295168
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 16, 16, 512)       1180160
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 16, 16, 512)       2359808
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 16, 16, 512)       2359808
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 16, 16, 512)       2359808
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 4, 4, 512)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_5 (Dense)              (None, 4096)              33558528
_________________________________________________________________
dense_6 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_7 (Dense)              (None, 8)                 32776
=================================================================
Total params: 70,397,000
Trainable params: 70,397,000
Non-trainable params: 0
_________________________________________________________________
```

[6]:


conv2d_26_input: InputLayer
input: [(?, 128, 128, 3)]
output: [(?, 128, 128, 3)]
conv2d_26: Conv2D
input: (?, 128, 128, 3)
output: (?, 128, 128, 64)
conv2d_27: Conv2D
input: (?, 128, 128, 64)
output: (?, 128, 128, 64)
max_pooling2d_10: MaxPooling2D
input: (?, 128, 128, 64)
output: (?, 64, 64, 64)
conv2d_28: Conv2D
input: (?, 64, 64, 64)
output: (?, 64, 64, 128)
conv2d_29: Conv2D
input: (?, 64, 64, 128)
output: (?, 64, 64, 128)
max_pooling2d_11: MaxPooling2D
input: (?, 64, 64, 128)
output: (?, 32, 32, 128)
conv2d_30: Conv2D
input: (?, 32, 32, 128)
output: (?, 32, 32, 256)
conv2d_31: Conv2D
input: (?, 32, 32, 256)
output: (?, 32, 32, 256)
conv2d_32: Conv2D
input: (?, 32, 32, 256)
output: (?, 32, 32, 256)
conv2d_33: Conv2D
input: (?, 32, 32, 256)
output: (?, 32, 32, 256)
max_pooling2d_12: MaxPooling2D
input: (?, 32, 32, 256)
output: (?, 16, 16, 256)
conv2d_34: Conv2D
input: (?, 16, 16, 256)
output: (?, 16, 16, 512)
conv2d_35: Conv2D
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
conv2d_36: Conv2D
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
conv2d_37: Conv2D
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
max_pooling2d_13: MaxPooling2D
input: (?, 16, 16, 512)
output: (?, 8, 8, 512)
conv2d_38: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
conv2d_39: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
conv2d_40: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
conv2d_41: Conv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
max_pooling2d_14: MaxPooling2D
input: (?, 8, 8, 512)
output: (?, 4, 4, 512)
flatten_2: Flatten
input: (?, 4, 4, 512)
output: (?, 8192)
dense_5: Dense
input: (?, 8192)
output: (?, 4096)
dense_6: Dense
input: (?, 4096)
output: (?, 4096)
dense_7: Dense
input: (?, 4096)
output: (?, 8)

[15]:
```
model2.
 ↪compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
```

[16]:
```
start_time=t.time()
history2 = model2.fit(train_images, train_labels,
          validation_data=(test_images, test_labels),
          epochs=100, batch_size=64,callbacks=[early])
print("\n Training Time: {} seconds".format(t.time()-start_time))
model2.save('vgg_19_n128_colour.h5')
```

```
Epoch 1/100
  2/157 […] - ETA: 42s - loss: 2.0794 - accuracy:
0.1719WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.1492s vs `on_train_batch_end` time: 0.4012s).
Check your callbacks.
157/157 [==============================] - 93s 593ms/step - loss: 2.0799 -
accuracy: 0.1212 - val_loss: 2.0801 - val_accuracy: 0.1150
Epoch 2/100
157/157 [==============================] - 91s 583ms/step - loss: 2.0795 -
accuracy: 0.1250 - val_loss: 2.0800 - val_accuracy: 0.1150
Epoch 3/100
157/157 [==============================] - 92s 583ms/step - loss: 2.0792 -
accuracy: 0.1252 - val_loss: 2.0796 - val_accuracy: 0.1150
Epoch 4/100
157/157 [==============================] - 92s 583ms/step - loss: 2.0788 -
accuracy: 0.1253 - val_loss: 2.0792 - val_accuracy: 0.1150
Epoch 5/100
157/157 [==============================] - 92s 584ms/step - loss: 2.0781 -
accuracy: 0.1378 - val_loss: 2.0783 - val_accuracy: 0.1430
Epoch 6/100
157/157 [==============================] - 92s 584ms/step - loss: 2.0765 -
accuracy: 0.1804 - val_loss: 2.0760 - val_accuracy: 0.1695
Epoch 7/100
157/157 [==============================] - 92s 583ms/step - loss: 2.0733 -
accuracy: 0.2111 - val_loss: 2.0715 - val_accuracy: 0.2265
Epoch 8/100
157/157 [==============================] - 92s 584ms/step - loss: 2.0672 -
accuracy: 0.2286 - val_loss: 2.0633 - val_accuracy: 0.2275
Epoch 9/100
157/157 [==============================] - 92s 584ms/step - loss: 2.0567 -
accuracy: 0.2237 - val_loss: 2.0496 - val_accuracy: 0.2185
Epoch 10/100
157/157 [==============================] - 92s 584ms/step - loss: 2.0334 -
accuracy: 0.2161 - val_loss: 2.0054 - val_accuracy: 0.2190
Epoch 11/100
```

```
157/157 [==============================] - 92s 584ms/step - loss: 1.9654 -
accuracy: 0.2113 - val_loss: 1.9493 - val_accuracy: 0.2145
Epoch 12/100
157/157 [==============================] - 92s 584ms/step - loss: 1.9196 -
accuracy: 0.2216 - val_loss: 1.9099 - val_accuracy: 0.2320
Epoch 13/100
157/157 [==============================] - 92s 584ms/step - loss: 1.8858 -
accuracy: 0.2329 - val_loss: 1.8826 - val_accuracy: 0.2645
Epoch 14/100
157/157 [==============================] - 92s 583ms/step - loss: 1.8578 -
accuracy: 0.2500 - val_loss: 1.8792 - val_accuracy: 0.2585
Epoch 15/100
157/157 [==============================] - 92s 583ms/step - loss: 1.8396 -
accuracy: 0.2637 - val_loss: 1.8417 - val_accuracy: 0.2335
Epoch 16/100
157/157 [==============================] - 92s 583ms/step - loss: 1.7945 -
accuracy: 0.2843 - val_loss: 1.7679 - val_accuracy: 0.3005
Epoch 17/100
157/157 [==============================] - 92s 583ms/step - loss: 1.7651 -
accuracy: 0.3024 - val_loss: 1.7336 - val_accuracy: 0.3135
Epoch 18/100
157/157 [==============================] - 91s 583ms/step - loss: 1.7634 -
accuracy: 0.3059 - val_loss: 1.7508 - val_accuracy: 0.3005
Epoch 19/100
157/157 [==============================] - 91s 582ms/step - loss: 1.7313 -
accuracy: 0.3252 - val_loss: 1.7216 - val_accuracy: 0.3285
Epoch 20/100
157/157 [==============================] - 92s 583ms/step - loss: 1.6920 -
accuracy: 0.3571 - val_loss: 1.6626 - val_accuracy: 0.3705
Epoch 21/100
157/157 [==============================] - 91s 583ms/step - loss: 1.6443 -
accuracy: 0.3806 - val_loss: 1.6287 - val_accuracy: 0.4015
Epoch 22/100
157/157 [==============================] - 91s 583ms/step - loss: 1.6009 -
accuracy: 0.3885 - val_loss: 1.6152 - val_accuracy: 0.4050
Epoch 23/100
157/157 [==============================] - 92s 583ms/step - loss: 1.5741 -
accuracy: 0.4068 - val_loss: 1.5335 - val_accuracy: 0.4585
Epoch 24/100
157/157 [==============================] - 92s 584ms/step - loss: 1.5687 -
accuracy: 0.4103 - val_loss: 1.6175 - val_accuracy: 0.3980
Epoch 25/100
157/157 [==============================] - 92s 584ms/step - loss: 1.5411 -
accuracy: 0.4253 - val_loss: 1.5061 - val_accuracy: 0.4530
Epoch 26/100
157/157 [==============================] - 92s 584ms/step - loss: 1.5144 -
accuracy: 0.4333 - val_loss: 1.5174 - val_accuracy: 0.4345
Epoch 27/100
```

```
157/157 [==============================] - 92s 584ms/step - loss: 1.4827 -
accuracy: 0.4424 - val_loss: 1.4914 - val_accuracy: 0.4400
Epoch 28/100
157/157 [==============================] - 92s 584ms/step - loss: 1.4807 -
accuracy: 0.4412 - val_loss: 1.4906 - val_accuracy: 0.4545
Epoch 29/100
157/157 [==============================] - 92s 584ms/step - loss: 1.4696 -
accuracy: 0.4435 - val_loss: 1.4602 - val_accuracy: 0.4490
Epoch 30/100
157/157 [==============================] - 92s 584ms/step - loss: 1.4568 -
accuracy: 0.4506 - val_loss: 1.4696 - val_accuracy: 0.4545
Epoch 31/100
157/157 [==============================] - 92s 584ms/step - loss: 1.4229 -
accuracy: 0.4599 - val_loss: 1.4777 - val_accuracy: 0.4465
Epoch 32/100
157/157 [==============================] - 92s 584ms/step - loss: 1.4187 -
accuracy: 0.4634 - val_loss: 1.3999 - val_accuracy: 0.4795
Epoch 33/100
157/157 [==============================] - 92s 584ms/step - loss: 1.3875 -
accuracy: 0.4725 - val_loss: 1.5598 - val_accuracy: 0.4110
Epoch 34/100
157/157 [==============================] - 92s 585ms/step - loss: 1.3933 -
accuracy: 0.4722 - val_loss: 1.4840 - val_accuracy: 0.4695
Epoch 35/100
157/157 [==============================] - 92s 585ms/step - loss: 1.3894 -
accuracy: 0.4736 - val_loss: 1.4072 - val_accuracy: 0.4750
Epoch 36/100
157/157 [==============================] - 92s 585ms/step - loss: 1.3527 -
accuracy: 0.4898 - val_loss: 1.3661 - val_accuracy: 0.4940
Epoch 37/100
157/157 [==============================] - 92s 584ms/step - loss: 1.3298 -
accuracy: 0.4947 - val_loss: 1.5163 - val_accuracy: 0.4385
Epoch 38/100
157/157 [==============================] - 92s 584ms/step - loss: 1.3294 -
accuracy: 0.4948 - val_loss: 1.3582 - val_accuracy: 0.5070
Epoch 39/100
157/157 [==============================] - 92s 584ms/step - loss: 1.3097 -
accuracy: 0.5100 - val_loss: 1.3224 - val_accuracy: 0.5140
Epoch 40/100
157/157 [==============================] - 92s 584ms/step - loss: 1.2796 -
accuracy: 0.5188 - val_loss: 1.3038 - val_accuracy: 0.5240
Epoch 41/100
157/157 [==============================] - 92s 584ms/step - loss: 1.2630 -
accuracy: 0.5206 - val_loss: 1.3179 - val_accuracy: 0.5125
Epoch 42/100
157/157 [==============================] - 92s 584ms/step - loss: 1.2755 -
accuracy: 0.5162 - val_loss: 1.3314 - val_accuracy: 0.5070
Epoch 43/100
```

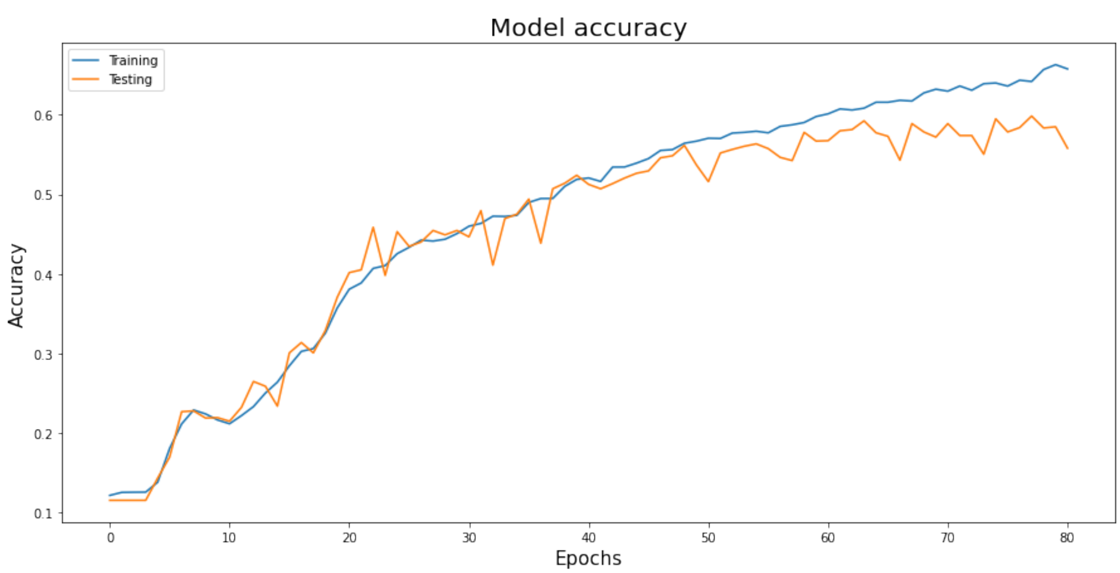
```
157/157 [==============================] - 92s 584ms/step - loss: 1.2420 -
accuracy: 0.5343 - val_loss: 1.3027 - val_accuracy: 0.5135
Epoch 44/100
157/157 [==============================] - 92s 585ms/step - loss: 1.2390 -
accuracy: 0.5344 - val_loss: 1.3027 - val_accuracy: 0.5205
Epoch 45/100
157/157 [==============================] - 92s 586ms/step - loss: 1.2264 -
accuracy: 0.5393 - val_loss: 1.2874 - val_accuracy: 0.5265
Epoch 46/100
157/157 [==============================] - 92s 585ms/step - loss: 1.2034 -
accuracy: 0.5450 - val_loss: 1.2806 - val_accuracy: 0.5295
Epoch 47/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1931 -
accuracy: 0.5552 - val_loss: 1.2502 - val_accuracy: 0.5460
Epoch 48/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1922 -
accuracy: 0.5563 - val_loss: 1.2473 - val_accuracy: 0.5485
Epoch 49/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1754 -
accuracy: 0.5643 - val_loss: 1.2344 - val_accuracy: 0.5615
Epoch 50/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1658 -
accuracy: 0.5669 - val_loss: 1.2634 - val_accuracy: 0.5370
Epoch 51/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1494 -
accuracy: 0.5706 - val_loss: 1.3133 - val_accuracy: 0.5160
Epoch 52/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1530 -
accuracy: 0.5703 - val_loss: 1.2313 - val_accuracy: 0.5520
Epoch 53/100
157/157 [==============================] - 92s 585ms/step - loss: 1.1325 -
accuracy: 0.5771 - val_loss: 1.2257 - val_accuracy: 0.5565
Epoch 54/100
157/157 [==============================] - 92s 584ms/step - loss: 1.1324 -
accuracy: 0.5780 - val_loss: 1.2254 - val_accuracy: 0.5605
Epoch 55/100
157/157 [==============================] - 92s 584ms/step - loss: 1.1285 -
accuracy: 0.5794 - val_loss: 1.2001 - val_accuracy: 0.5635
Epoch 56/100
157/157 [==============================] - 92s 584ms/step - loss: 1.1303 -
accuracy: 0.5774 - val_loss: 1.2132 - val_accuracy: 0.5575
Epoch 57/100
157/157 [==============================] - 92s 584ms/step - loss: 1.1058 -
accuracy: 0.5855 - val_loss: 1.2355 - val_accuracy: 0.5465
Epoch 58/100
157/157 [==============================] - 92s 584ms/step - loss: 1.1100 -
accuracy: 0.5875 - val_loss: 1.2463 - val_accuracy: 0.5425
Epoch 59/100
```

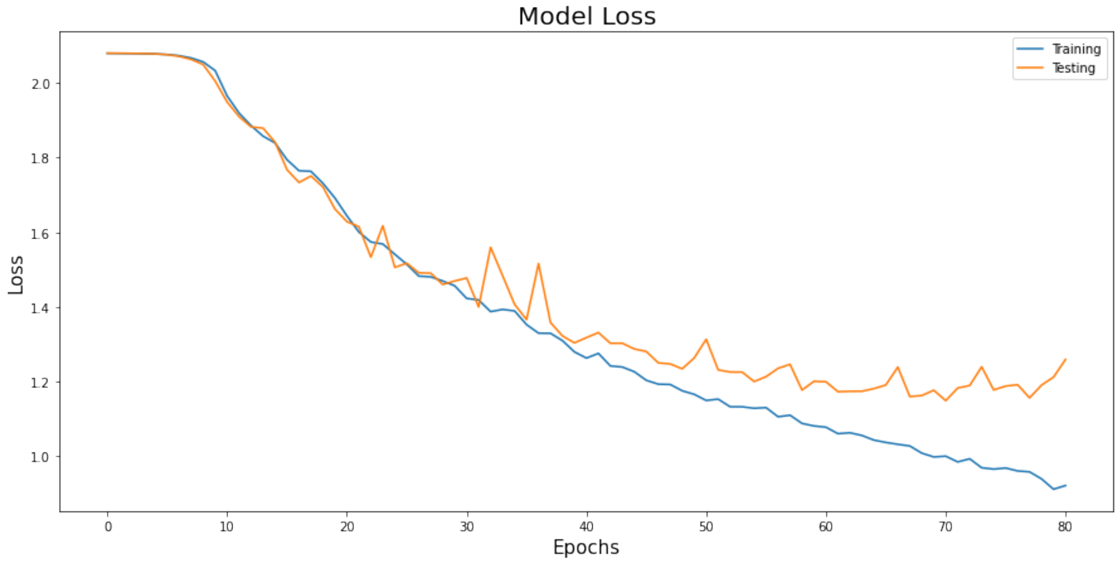
```
157/157 [==============================] - 92s 583ms/step - loss: 1.0879 -
accuracy: 0.5904 - val_loss: 1.1773 - val_accuracy: 0.5780
Epoch 60/100
157/157 [==============================] - 92s 583ms/step - loss: 1.0811 -
accuracy: 0.5979 - val_loss: 1.2008 - val_accuracy: 0.5670
Epoch 61/100
157/157 [==============================] - 92s 583ms/step - loss: 1.0778 -
accuracy: 0.6013 - val_loss: 1.1995 - val_accuracy: 0.5675
Epoch 62/100
157/157 [==============================] - 92s 583ms/step - loss: 1.0604 -
accuracy: 0.6075 - val_loss: 1.1730 - val_accuracy: 0.5800
Epoch 63/100
157/157 [==============================] - 92s 584ms/step - loss: 1.0628 -
accuracy: 0.6061 - val_loss: 1.1739 - val_accuracy: 0.5815
Epoch 64/100
157/157 [==============================] - 92s 583ms/step - loss: 1.0557 -
accuracy: 0.6084 - val_loss: 1.1742 - val_accuracy: 0.5925
Epoch 65/100
157/157 [==============================] - 91s 582ms/step - loss: 1.0433 -
accuracy: 0.6159 - val_loss: 1.1811 - val_accuracy: 0.5775
Epoch 66/100
157/157 [==============================] - 91s 582ms/step - loss: 1.0370 -
accuracy: 0.6159 - val_loss: 1.1909 - val_accuracy: 0.5730
Epoch 67/100
157/157 [==============================] - 91s 582ms/step - loss: 1.0319 -
accuracy: 0.6183 - val_loss: 1.2389 - val_accuracy: 0.5430
Epoch 68/100
157/157 [==============================] - 91s 582ms/step - loss: 1.0275 -
accuracy: 0.6174 - val_loss: 1.1599 - val_accuracy: 0.5890
Epoch 69/100
157/157 [==============================] - 92s 583ms/step - loss: 1.0083 -
accuracy: 0.6276 - val_loss: 1.1626 - val_accuracy: 0.5785
Epoch 70/100
157/157 [==============================] - 92s 583ms/step - loss: 0.9981 -
accuracy: 0.6323 - val_loss: 1.1769 - val_accuracy: 0.5720
Epoch 71/100
157/157 [==============================] - 92s 584ms/step - loss: 1.0001 -
accuracy: 0.6298 - val_loss: 1.1490 - val_accuracy: 0.5890
Epoch 72/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9848 -
accuracy: 0.6363 - val_loss: 1.1828 - val_accuracy: 0.5740
Epoch 73/100
157/157 [==============================] - 92s 584ms/step - loss: 0.9931 -
accuracy: 0.6310 - val_loss: 1.1895 - val_accuracy: 0.5740
Epoch 74/100
157/157 [==============================] - 92s 584ms/step - loss: 0.9691 -
accuracy: 0.6392 - val_loss: 1.2396 - val_accuracy: 0.5505
Epoch 75/100
```

```
157/157 [==============================] - 92s 585ms/step - loss: 0.9655 -
accuracy: 0.6401 - val_loss: 1.1777 - val_accuracy: 0.5950
Epoch 76/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9684 -
accuracy: 0.6362 - val_loss: 1.1879 - val_accuracy: 0.5785
Epoch 77/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9606 -
accuracy: 0.6436 - val_loss: 1.1916 - val_accuracy: 0.5840
Epoch 78/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9581 -
accuracy: 0.6420 - val_loss: 1.1564 - val_accuracy: 0.5985
Epoch 79/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9392 -
accuracy: 0.6570 - val_loss: 1.1907 - val_accuracy: 0.5835
Epoch 80/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9115 -
accuracy: 0.6632 - val_loss: 1.2120 - val_accuracy: 0.5850
Epoch 81/100
157/157 [==============================] - 92s 585ms/step - loss: 0.9215 -
accuracy: 0.6578 - val_loss: 1.2593 - val_accuracy: 0.5580

 Training Time: 7479.614980459213 seconds
```

[17]:
```python
score2 = model2.evaluate(test_images,test_labels,verbose=0)
print('Test Loss - ',score2[0])
print('Test Accuracy - ',score2[1])
graphs(history2,score2)
```

```
Test Loss -  1.259333610534668
Test Accuracy -  0.5580000281333923
```






```
Accuracy of the model: 55.80%
```

0.7 ## AlexNet

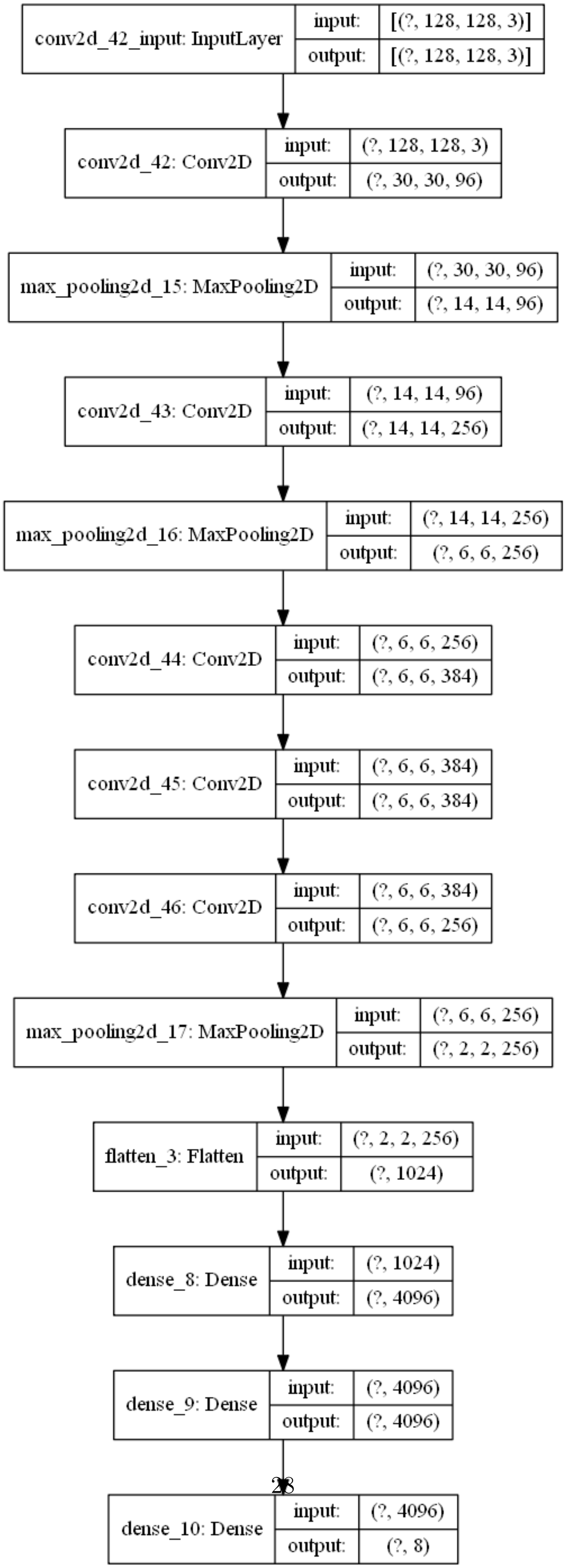
- It is comparetively a very small network with only 5 convolution layers and varying filter sizes.

[7]:
```
#alexnet
model3 = Sequential()
model3.
 ↪add(Conv2D(input_shape=(n,n,3),filters=96,kernel_size=(11,11),strides=(4,4),padding="valid"
 ↪activation="relu"))
model3.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
model3.add(Conv2D(256, kernel_size=(5,5), padding="same",strides=(1,1),␣
 ↪activation="relu"))
model3.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
model3.add(Conv2D(384, kernel_size=(3,3), padding="same", strides=(1,1),␣
 ↪activation="relu"))
model3.add(Conv2D(384, kernel_size=(3,3), padding="same", strides=(1,1),␣
 ↪activation="relu"))
model3.add(Conv2D(256, kernel_size=(3,3), padding="same", strides=(1,1),␣
 ↪activation="relu"))
model3.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
model3.add(Flatten())
model3.add(Dense(4096,activation="relu"))
model3.add(Dense(4096,activation="relu"))
model3.add(Dense(units=len(classes), activation="softmax"))
```

```
model3.summary()
plot_model(model3, show_shapes=True, to_file='ALEX_NET.png')
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_42 (Conv2D)           (None, 30, 30, 96)        34944
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 14, 14, 96)        0
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 14, 14, 256)       614656
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 6, 6, 384)         885120
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 6, 6, 384)         1327488
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 6, 6, 256)         884992
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 2, 2, 256)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 4096)              4198400
_________________________________________________________________
dense_9 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_10 (Dense)             (None, 8)                 32776
=================================================================
Total params: 24,759,688
Trainable params: 24,759,688
Non-trainable params: 0
_________________________________________________________________
```

[7]:


conv2d_42_input: InputLayer
input: [(?, 128, 128, 3)]
output: [(?, 128, 128, 3)]
conv2d_42: Conv2D
input: (?, 128, 128, 3)
output: (?, 30, 30, 96)
max_pooling2d_15: MaxPooling2D
input: (?, 30, 30, 96)
output: (?, 14, 14, 96)
conv2d_43: Conv2D
input: (?, 14, 14, 96)
output: (?, 14, 14, 256)
max_pooling2d_16: MaxPooling2D
input: (?, 14, 14, 256)
output: (?, 6, 6, 256)
conv2d_44: Conv2D
input: (?, 6, 6, 256)
output: (?, 6, 6, 384)
conv2d_45: Conv2D
input: (?, 6, 6, 384)
output: (?, 6, 6, 384)
conv2d_46: Conv2D
input: (?, 6, 6, 384)
output: (?, 6, 6, 256)
max_pooling2d_17: MaxPooling2D
input: (?, 6, 6, 256)
output: (?, 2, 2, 256)
flatten_3: Flatten
input: (?, 2, 2, 256)
output: (?, 1024)
dense_8: Dense
input: (?, 1024)
output: (?, 4096)
dense_9: Dense
input: (?, 4096)
output: (?, 4096)
dense_10: Dense
input: (?, 4096)
output: (?, 8)

[12]:
```
model3.
 ↪compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
```

[13]:
```
start_time=t.time()
history3 = model3.fit(train_images, train_labels,
          validation_data=(test_images, test_labels),
          epochs=100, batch_size=64,callbacks=[early])
print("\n Training Time: {} seconds".format(t.time()-start_time))
model3.save('alexnet_n128_colour.h5')
```

```
Epoch 1/100
157/157 [==============================] - 8s 54ms/step - loss: 1.9990 -
accuracy: 0.2024 - val_loss: 1.8580 - val_accuracy: 0.3075
Epoch 2/100
157/157 [==============================] - 8s 49ms/step - loss: 1.7447 -
accuracy: 0.3254 - val_loss: 1.9200 - val_accuracy: 0.2960
Epoch 3/100
157/157 [==============================] - 8s 49ms/step - loss: 1.5547 -
accuracy: 0.4045 - val_loss: 1.4474 - val_accuracy: 0.4690
Epoch 4/100
157/157 [==============================] - 8s 49ms/step - loss: 1.3725 -
accuracy: 0.4887 - val_loss: 1.3616 - val_accuracy: 0.4940
Epoch 5/100
157/157 [==============================] - 8s 49ms/step - loss: 1.2997 -
accuracy: 0.5190 - val_loss: 1.4937 - val_accuracy: 0.4405
Epoch 6/100
157/157 [==============================] - 8s 49ms/step - loss: 1.2072 -
accuracy: 0.5506 - val_loss: 1.2207 - val_accuracy: 0.5485
Epoch 7/100
157/157 [==============================] - 8s 49ms/step - loss: 1.1483 -
accuracy: 0.5744 - val_loss: 1.2489 - val_accuracy: 0.5485
Epoch 8/100
157/157 [==============================] - 8s 49ms/step - loss: 1.1211 -
accuracy: 0.5842 - val_loss: 1.2298 - val_accuracy: 0.5440
Epoch 9/100
157/157 [==============================] - 8s 49ms/step - loss: 1.0981 -
accuracy: 0.5937 - val_loss: 1.1190 - val_accuracy: 0.5950
Epoch 10/100
157/157 [==============================] - 8s 49ms/step - loss: 1.0569 -
accuracy: 0.6077 - val_loss: 1.0918 - val_accuracy: 0.5980
Epoch 11/100
157/157 [==============================] - 8s 49ms/step - loss: 1.0337 -
accuracy: 0.6184 - val_loss: 1.0918 - val_accuracy: 0.6025
Epoch 12/100
157/157 [==============================] - 8s 49ms/step - loss: 1.0029 -
```

```
accuracy: 0.6268 - val_loss: 1.0568 - val_accuracy: 0.6110
Epoch 13/100
157/157 [==============================] - 8s 49ms/step - loss: 0.9875 -
accuracy: 0.6340 - val_loss: 1.1050 - val_accuracy: 0.5945
Epoch 14/100
157/157 [==============================] - 8s 50ms/step - loss: 0.9688 -
accuracy: 0.6386 - val_loss: 1.0247 - val_accuracy: 0.6300
Epoch 15/100
157/157 [==============================] - 8s 50ms/step - loss: 0.9435 -
accuracy: 0.6514 - val_loss: 1.0384 - val_accuracy: 0.6170
Epoch 16/100
157/157 [==============================] - 8s 50ms/step - loss: 0.9284 -
accuracy: 0.6562 - val_loss: 1.0282 - val_accuracy: 0.6310
Epoch 17/100
157/157 [==============================] - 8s 50ms/step - loss: 0.9095 -
accuracy: 0.6686 - val_loss: 1.0407 - val_accuracy: 0.6135
Epoch 18/100
157/157 [==============================] - 8s 50ms/step - loss: 0.9018 -
accuracy: 0.6666 - val_loss: 0.9957 - val_accuracy: 0.6380
Epoch 19/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8751 -
accuracy: 0.6814 - val_loss: 1.0291 - val_accuracy: 0.6225
Epoch 20/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8671 -
accuracy: 0.6882 - val_loss: 1.0147 - val_accuracy: 0.6305
Epoch 21/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8510 -
accuracy: 0.6868 - val_loss: 0.9729 - val_accuracy: 0.6535
Epoch 22/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8365 -
accuracy: 0.6929 - val_loss: 0.9775 - val_accuracy: 0.6420
Epoch 23/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8265 -
accuracy: 0.6949 - val_loss: 0.9778 - val_accuracy: 0.6490
Epoch 24/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8043 -
accuracy: 0.7074 - val_loss: 0.9656 - val_accuracy: 0.6450
Epoch 25/100
157/157 [==============================] - 8s 50ms/step - loss: 0.8037 -
accuracy: 0.7039 - val_loss: 0.9902 - val_accuracy: 0.6350
Epoch 26/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7811 -
accuracy: 0.7172 - val_loss: 0.9436 - val_accuracy: 0.6605
Epoch 27/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7828 -
accuracy: 0.7127 - val_loss: 0.9693 - val_accuracy: 0.6560
Epoch 28/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7736 -
```

```
accuracy: 0.7191 - val_loss: 0.9208 - val_accuracy: 0.6695
Epoch 29/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7562 -
accuracy: 0.7235 - val_loss: 0.9628 - val_accuracy: 0.6490
Epoch 30/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7540 -
accuracy: 0.7248 - val_loss: 0.9729 - val_accuracy: 0.6535
Epoch 31/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7318 -
accuracy: 0.7329 - val_loss: 0.9215 - val_accuracy: 0.6710
Epoch 32/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7240 -
accuracy: 0.7362 - val_loss: 0.9316 - val_accuracy: 0.6540
Epoch 33/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7168 -
accuracy: 0.7379 - val_loss: 0.9241 - val_accuracy: 0.6630
Epoch 34/100
157/157 [==============================] - 8s 50ms/step - loss: 0.7003 -
accuracy: 0.7461 - val_loss: 0.9128 - val_accuracy: 0.6760
Epoch 35/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6994 -
accuracy: 0.7480 - val_loss: 0.9275 - val_accuracy: 0.6625
Epoch 36/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6839 -
accuracy: 0.7498 - val_loss: 0.9375 - val_accuracy: 0.6605
Epoch 37/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6914 -
accuracy: 0.7466 - val_loss: 0.8952 - val_accuracy: 0.6745
Epoch 38/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6652 -
accuracy: 0.7586 - val_loss: 0.9353 - val_accuracy: 0.6680
Epoch 39/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6599 -
accuracy: 0.7596 - val_loss: 0.8892 - val_accuracy: 0.6780
Epoch 40/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6473 -
accuracy: 0.7672 - val_loss: 0.9025 - val_accuracy: 0.6755
Epoch 41/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6494 -
accuracy: 0.7653 - val_loss: 0.9057 - val_accuracy: 0.6815
Epoch 42/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6291 -
accuracy: 0.7718 - val_loss: 0.9044 - val_accuracy: 0.6760
Epoch 43/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6266 -
accuracy: 0.7715 - val_loss: 0.8987 - val_accuracy: 0.6815
Epoch 44/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6158 -
```

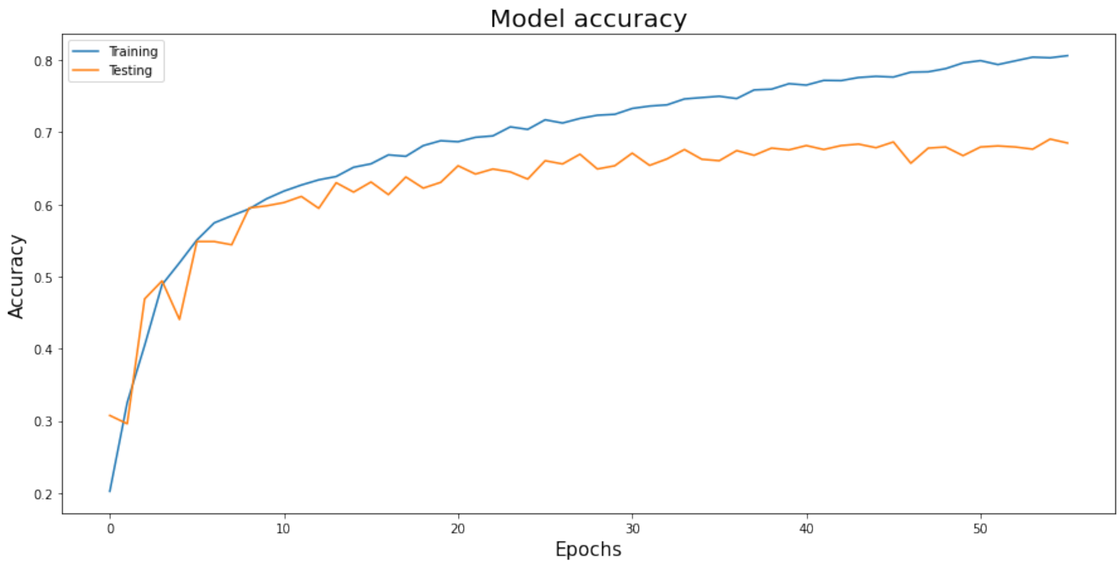
```
accuracy: 0.7758 - val_loss: 0.8964 - val_accuracy: 0.6835
Epoch 45/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6119 -
accuracy: 0.7774 - val_loss: 0.9166 - val_accuracy: 0.6785
Epoch 46/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6160 -
accuracy: 0.7764 - val_loss: 0.8826 - val_accuracy: 0.6865
Epoch 47/100
157/157 [==============================] - 8s 50ms/step - loss: 0.6048 -
accuracy: 0.7832 - val_loss: 0.9630 - val_accuracy: 0.6570
Epoch 48/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5962 -
accuracy: 0.7837 - val_loss: 0.9109 - val_accuracy: 0.6780
Epoch 49/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5893 -
accuracy: 0.7881 - val_loss: 0.8983 - val_accuracy: 0.6795
Epoch 50/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5733 -
accuracy: 0.7960 - val_loss: 0.9185 - val_accuracy: 0.6675
Epoch 51/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5628 -
accuracy: 0.7992 - val_loss: 0.9101 - val_accuracy: 0.6795
Epoch 52/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5677 -
accuracy: 0.7937 - val_loss: 0.9100 - val_accuracy: 0.6810
Epoch 53/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5553 -
accuracy: 0.7989 - val_loss: 0.9205 - val_accuracy: 0.6795
Epoch 54/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5495 -
accuracy: 0.8040 - val_loss: 0.9032 - val_accuracy: 0.6765
Epoch 55/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5504 -
accuracy: 0.8031 - val_loss: 0.8839 - val_accuracy: 0.6905- loss: 0.5526 - accu
Epoch 56/100
157/157 [==============================] - 8s 50ms/step - loss: 0.5406 -
accuracy: 0.8061 - val_loss: 0.8885 - val_accuracy: 0.6850

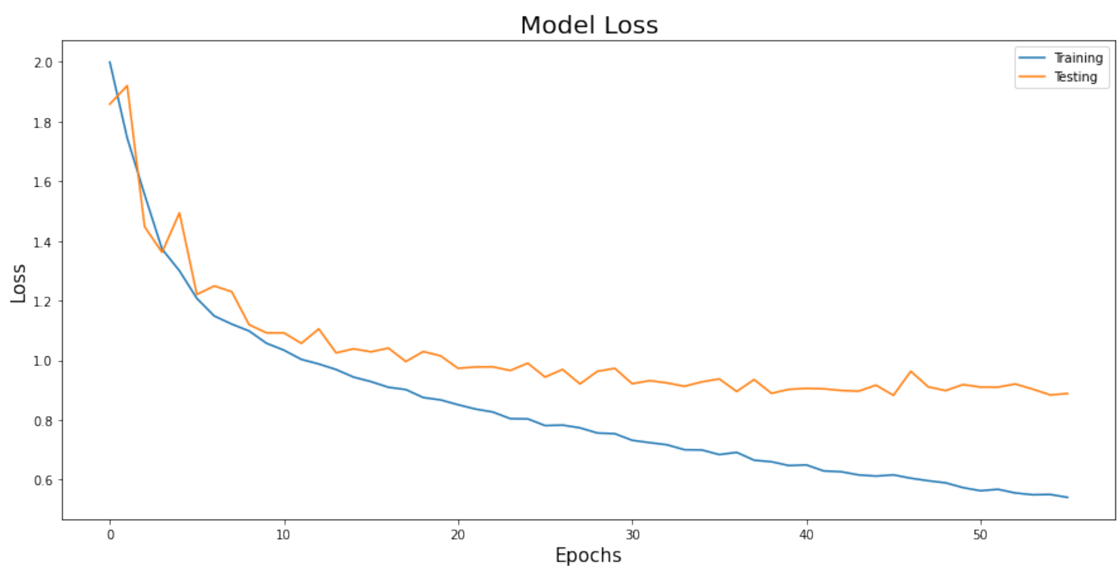
 Training Time: 447.03455662727356 seconds
```

[14]:

```
score3 = model3.evaluate(test_images,test_labels,verbose=0)
print('Test Loss - ',score3[0])
print('Test Accuracy - ',score3[1])
graphs(history3,score3)
```

```
Test Loss -  0.888465404510498
Test Accuracy -  0.6850000023841858
```






```
Accuracy of the model: 68.50%
```

0.8 Inception Module

- ```
  Some ideas on the inception module implementation from: [1]
  F. SHAIKH, ``Inception Network | Implementation Of GoogleNet
  In Keras'', Analytics Vidhya, 2020. [Online]. Available:
  https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/
  [Accessed: 12- Aug- 2020]
  ```
  ```

[8]:

```python
# function for creating a projected inception module (performance improved)
def inception_module(layer_in, f1, f2_in, f2_out, f3_in, f3_out, f4_out):
    conv1 = Conv2D(f1, (1,1), padding='same', activation='relu')(layer_in)
    conv3 = Conv2D(f2_in, (1,1), padding='same', activation='relu')(layer_in)
    conv3 = Conv2D(f2_out, (3,3), padding='same', activation='relu')(conv3)
    conv5 = Conv2D(f3_in, (1,1), padding='same', activation='relu')(layer_in)
    conv5 = Conv2D(f3_out, (5,5), padding='same', activation='relu')(conv5)
    pool = MaxPooling2D((3,3), strides=(1,1), padding='same')(layer_in)
    pool = Conv2D(f4_out, (1,1), padding='same', activation='relu')(pool)
    # concatenate filters, assumes filters/channels last
    layer_out = concatenate([conv1, conv3, conv5, pool], axis=-1)
    return layer_out
```

0.9 Let's test our function

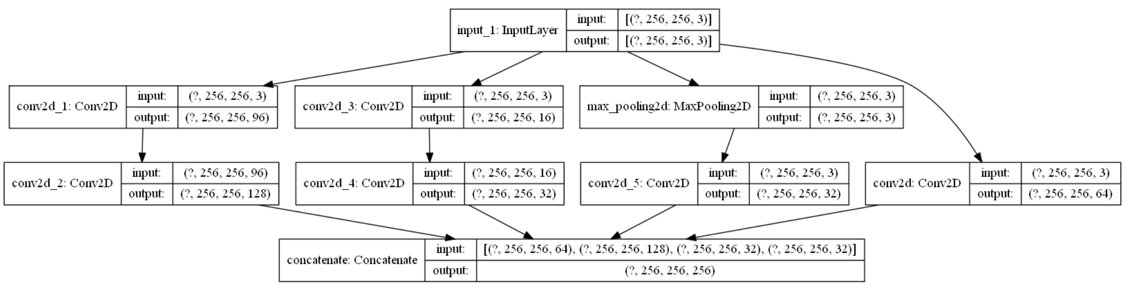
[9]:

```python
visible = Input(shape=(256, 256, 3))
# add inception block 1
layer = inception_module(visible, 64, 96, 128, 16, 32, 32)
# create model
model = Model(inputs=visible, outputs=layer)
# summarize model
model.summary()
# plot model architecture
plot_model(model, show_shapes=True)
```

```
Model: "functional_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
________________________________________________________________________________
__________________
conv2d_1 (Conv2D)               (None, 256, 256, 96) 384         input_1[0][0]
________________________________________________________________________________
__________________
conv2d_3 (Conv2D)               (None, 256, 256, 16) 64          input_1[0][0]
________________________________________________________________________________
__________________
max_pooling2d (MaxPooling2D)    (None, 256, 256, 3)  0           input_1[0][0]
________________________________________________________________________________
__________________
conv2d (Conv2D)                 (None, 256, 256, 64) 256         input_1[0][0]
________________________________________________________________________________
__________________
conv2d_2 (Conv2D)               (None, 256, 256, 128 110720      conv2d_1[0][0]
```

```
------------------------------------------------------------------------------------------------
conv2d_4 (Conv2D)               (None, 256, 256, 32) 12832       conv2d_3[0][0]
------------------------------------------------------------------------------------------------
conv2d_5 (Conv2D)               (None, 256, 256, 32) 128
max_pooling2d[0][0]
------------------------------------------------------------------------------------------------
concatenate (Concatenate)       (None, 256, 256, 256 0           conv2d[0][0]
                                                                 conv2d_2[0][0]
                                                                 conv2d_4[0][0]
                                                                 conv2d_5[0][0]
================================================================================================
Total params: 124,384
Trainable params: 124,384
Non-trainable params: 0
------------------------------------------------------------------------------------------------
```

[9]:



0.10 GoogleNet using inception module

0.10.1 Relevant sources

[1] C. Szegedy et al., ``Going deeper with convolutions,'' 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[10]:
```
kernel_init = keras.initializers.glorot_uniform()
bias_init = keras.initializers.Constant(value=0.2)
```

[12]:
```
# define model input
input_layer = Input(shape=(128, 128, 3))
```

```
x = Conv2D(64, (7, 7), padding='same', strides=(2, 2),␣
↪activation='relu',kernel_initializer=kernel_init,␣
↪bias_initializer=bias_init)(input_layer)
x = MaxPool2D((3, 3), padding='same', strides=(2, 2))(x)
x = Conv2D(64, (1, 1), padding='same', strides=(1, 1), activation='relu')(x)
x = Conv2D(192, (3, 3), padding='same', strides=(1, 1), activation='relu')(x)
x = MaxPool2D((3, 3), padding='same', strides=(2, 2))(x)

x = inception_module(x, 64, 96, 128, 16, 32, 32)#3a
x = inception_module(x, 128, 128, 192, 32, 96, 64)#3b
x = MaxPool2D((3, 3), padding='same', strides=(2, 2))(x)

x = inception_module(x, 192, 96, 208, 16, 48, 64) #4a

x1 = AveragePooling2D((5, 5), strides=3)(x)
x1 = Conv2D(128, (1, 1), padding='same', activation='relu')(x1)
x1 = Flatten()(x1)
x1 = Dense(1024, activation='relu')(x1)
x1 = Dropout(0.7)(x1)
x1 = Dense(len(classes), activation='softmax', name="Softmax_1")(x1)

x = inception_module(x, 160, 112, 224, 24, 64, 64) #4b
x = inception_module(x, 128, 128, 256, 24, 64, 64) #4c
x = inception_module(x, 112, 144, 288, 32, 64, 64) #4d

x2 = AveragePooling2D((5, 5), strides=3)(x)
x2 = Conv2D(128, (1, 1), padding='same', activation='relu')(x2)
x2 = Flatten()(x2)
x2 = Dense(1024, activation='relu')(x2)
x2 = Dropout(0.7)(x2)
x2 = Dense(len(classes), activation='softmax',name="Softmax_2")(x2)

x = inception_module(x, 256, 160, 320, 32, 128, 128) #4e

x = MaxPool2D((3, 3), padding='same', strides=(2, 2))(x)

x = inception_module(x, 256, 160, 320, 32, 128, 128) #5a
x = inception_module(x, 384, 192, 384, 48, 128, 128 ) #5b

x = GlobalAveragePooling2D()(x)
x = Dropout(0.4)(x)
x = Dense(len(classes), activation='softmax',name="OUTPUT")(x)

model = Model(input_layer, [x, x1, x2], name='inception_v1')
model.summary()
plot_model(model)
```

```
Model: "inception_v1"
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
input_2 (InputLayer)            [(None, 128, 128, 3) 0
__________________________________________________________________________________
__________________
conv2d_6 (Conv2D)               (None, 64, 64, 64)   9472        input_2[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_1 (MaxPooling2D)  (None, 32, 32, 64)   0           conv2d_6[0][0]
__________________________________________________________________________________
__________________
conv2d_7 (Conv2D)               (None, 32, 32, 64)   4160
max_pooling2d_1[0][0]
__________________________________________________________________________________
__________________
conv2d_8 (Conv2D)               (None, 32, 32, 192)  110784      conv2d_7[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 192)  0           conv2d_8[0][0]
__________________________________________________________________________________
__________________
conv2d_10 (Conv2D)              (None, 16, 16, 96)   18528
max_pooling2d_2[0][0]
__________________________________________________________________________________
__________________
conv2d_12 (Conv2D)              (None, 16, 16, 16)   3088
max_pooling2d_2[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 192)  0
max_pooling2d_2[0][0]
__________________________________________________________________________________
__________________
conv2d_9 (Conv2D)               (None, 16, 16, 64)   12352
max_pooling2d_2[0][0]
__________________________________________________________________________________
__________________
conv2d_11 (Conv2D)              (None, 16, 16, 128)  110720      conv2d_10[0][0]
__________________________________________________________________________________
__________________
conv2d_13 (Conv2D)              (None, 16, 16, 32)   12832       conv2d_12[0][0]
__________________________________________________________________________________
__________________
conv2d_14 (Conv2D)              (None, 16, 16, 32)   6176
```

```
max_pooling2d_3[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 16, 16, 256)  0           conv2d_9[0][0]
                                                                 conv2d_11[0][0]
                                                                 conv2d_13[0][0]
                                                                 conv2d_14[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 16, 16, 128)  32896
concatenate_1[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 16, 16, 32)   8224
concatenate_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 16, 16, 256)  0
concatenate_1[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 16, 16, 128)  32896
concatenate_1[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 16, 16, 192)  221376      conv2d_16[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 16, 16, 96)   76896       conv2d_18[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 16, 16, 64)   16448
max_pooling2d_4[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 16, 16, 480)  0           conv2d_15[0][0]
                                                                 conv2d_17[0][0]
                                                                 conv2d_19[0][0]
                                                                 conv2d_20[0][0]
__________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 8, 8, 480)    0
concatenate_2[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 8, 8, 96)     46176
max_pooling2d_5[0][0]
__________________________________________________________________________________________________
```

```
__________________
conv2d_24 (Conv2D)             (None, 8, 8, 16)     7696
max_pooling2d_5[0][0]
________________________________________________________________________________
__________________
max_pooling2d_6 (MaxPooling2D)  (None, 8, 8, 480)    0
max_pooling2d_5[0][0]
________________________________________________________________________________
__________________
conv2d_21 (Conv2D)             (None, 8, 8, 192)    92352
max_pooling2d_5[0][0]
________________________________________________________________________________
__________________
conv2d_23 (Conv2D)             (None, 8, 8, 208)    179920       conv2d_22[0][0]
________________________________________________________________________________
__________________
conv2d_25 (Conv2D)             (None, 8, 8, 48)     19248        conv2d_24[0][0]
________________________________________________________________________________
__________________
conv2d_26 (Conv2D)             (None, 8, 8, 64)     30784
max_pooling2d_6[0][0]
________________________________________________________________________________
__________________
concatenate_3 (Concatenate)     (None, 8, 8, 512)    0            conv2d_21[0][0]
                                                                  conv2d_23[0][0]
                                                                  conv2d_25[0][0]
                                                                  conv2d_26[0][0]
________________________________________________________________________________
__________________
conv2d_29 (Conv2D)             (None, 8, 8, 112)    57456
concatenate_3[0][0]
________________________________________________________________________________
__________________
conv2d_31 (Conv2D)             (None, 8, 8, 24)     12312
concatenate_3[0][0]
________________________________________________________________________________
__________________
max_pooling2d_7 (MaxPooling2D)  (None, 8, 8, 512)    0
concatenate_3[0][0]
________________________________________________________________________________
__________________
conv2d_28 (Conv2D)             (None, 8, 8, 160)    82080
concatenate_3[0][0]
________________________________________________________________________________
__________________
conv2d_30 (Conv2D)             (None, 8, 8, 224)    226016       conv2d_29[0][0]
________________________________________________________________________________
__________________
```

```
conv2d_32 (Conv2D)              (None, 8, 8, 64)     38464       conv2d_31[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_33 (Conv2D)              (None, 8, 8, 64)     32832
max_pooling2d_7[0][0]
--------------------------------------------------------------------------------
------------------
concatenate_4 (Concatenate)     (None, 8, 8, 512)    0           conv2d_28[0][0]
                                                                 conv2d_30[0][0]
                                                                 conv2d_32[0][0]
                                                                 conv2d_33[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_35 (Conv2D)              (None, 8, 8, 128)    65664
concatenate_4[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_37 (Conv2D)              (None, 8, 8, 24)     12312
concatenate_4[0][0]
--------------------------------------------------------------------------------
------------------
max_pooling2d_8 (MaxPooling2D)  (None, 8, 8, 512)    0
concatenate_4[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_34 (Conv2D)              (None, 8, 8, 128)    65664
concatenate_4[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_36 (Conv2D)              (None, 8, 8, 256)    295168      conv2d_35[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_38 (Conv2D)              (None, 8, 8, 64)     38464       conv2d_37[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_39 (Conv2D)              (None, 8, 8, 64)     32832
max_pooling2d_8[0][0]
--------------------------------------------------------------------------------
------------------
concatenate_5 (Concatenate)     (None, 8, 8, 512)    0           conv2d_34[0][0]
                                                                 conv2d_36[0][0]
                                                                 conv2d_38[0][0]
                                                                 conv2d_39[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_41 (Conv2D)              (None, 8, 8, 144)    73872
concatenate_5[0][0]
--------------------------------------------------------------------------------
```

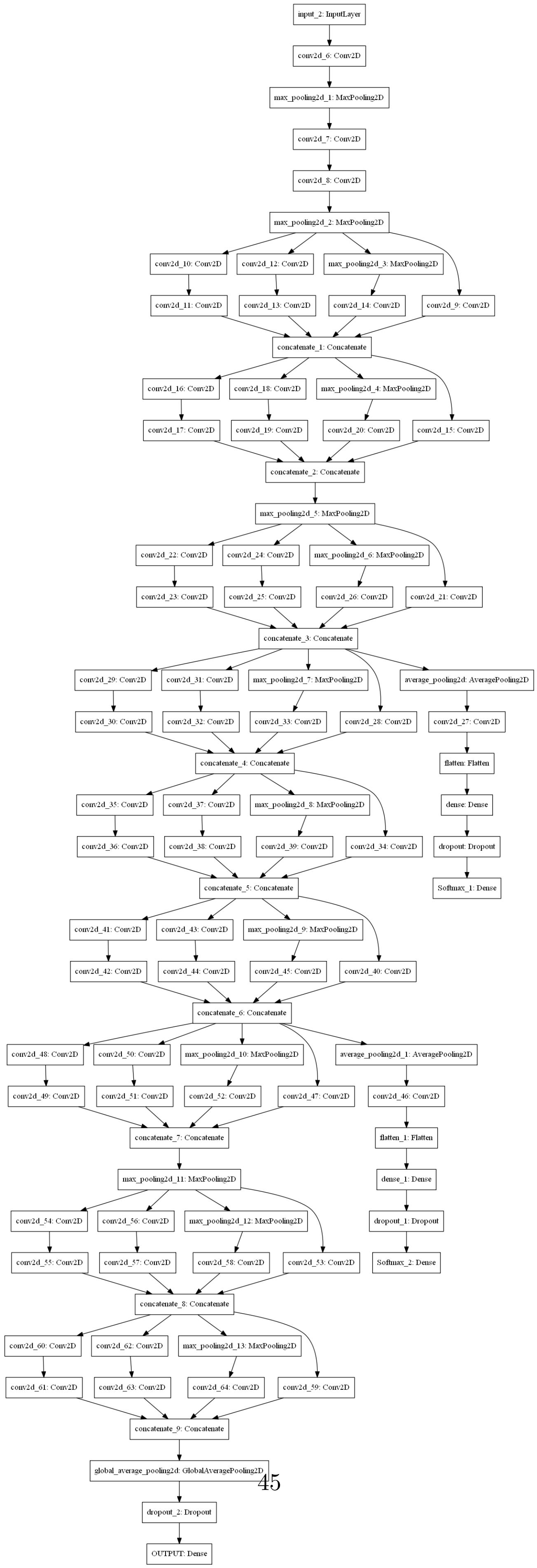
```
__________________
conv2d_43 (Conv2D)              (None, 8, 8, 32)     16416
concatenate_5[0][0]
________________________________________________________________________________
__________________
max_pooling2d_9 (MaxPooling2D)  (None, 8, 8, 512)    0
concatenate_5[0][0]
________________________________________________________________________________
__________________
conv2d_40 (Conv2D)              (None, 8, 8, 112)    57456
concatenate_5[0][0]
________________________________________________________________________________
__________________
conv2d_42 (Conv2D)              (None, 8, 8, 288)    373536      conv2d_41[0][0]
________________________________________________________________________________
__________________
conv2d_44 (Conv2D)              (None, 8, 8, 64)     51264       conv2d_43[0][0]
________________________________________________________________________________
__________________
conv2d_45 (Conv2D)              (None, 8, 8, 64)     32832
max_pooling2d_9[0][0]
________________________________________________________________________________
__________________
concatenate_6 (Concatenate)     (None, 8, 8, 528)    0           conv2d_40[0][0]
                                                                 conv2d_42[0][0]
                                                                 conv2d_44[0][0]
                                                                 conv2d_45[0][0]
________________________________________________________________________________
__________________
conv2d_48 (Conv2D)              (None, 8, 8, 160)    84640
concatenate_6[0][0]
________________________________________________________________________________
__________________
conv2d_50 (Conv2D)              (None, 8, 8, 32)     16928
concatenate_6[0][0]
________________________________________________________________________________
__________________
max_pooling2d_10 (MaxPooling2D) (None, 8, 8, 528)    0
concatenate_6[0][0]
________________________________________________________________________________
__________________
conv2d_47 (Conv2D)              (None, 8, 8, 256)    135424
concatenate_6[0][0]
________________________________________________________________________________
__________________
conv2d_49 (Conv2D)              (None, 8, 8, 320)    461120      conv2d_48[0][0]
________________________________________________________________________________
__________________
```

```
conv2d_51 (Conv2D)             (None, 8, 8, 128)    102528      conv2d_50[0][0]
__________________________________________________________________________________
__________________
conv2d_52 (Conv2D)             (None, 8, 8, 128)    67712
max_pooling2d_10[0][0]
__________________________________________________________________________________
__________________
concatenate_7 (Concatenate)    (None, 8, 8, 832)    0           conv2d_47[0][0]
                                                                conv2d_49[0][0]
                                                                conv2d_51[0][0]
                                                                conv2d_52[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_11 (MaxPooling2D) (None, 4, 4, 832)   0
concatenate_7[0][0]
__________________________________________________________________________________
__________________
conv2d_54 (Conv2D)             (None, 4, 4, 160)    133280
max_pooling2d_11[0][0]
__________________________________________________________________________________
__________________
conv2d_56 (Conv2D)             (None, 4, 4, 32)     26656
max_pooling2d_11[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_12 (MaxPooling2D) (None, 4, 4, 832)   0
max_pooling2d_11[0][0]
__________________________________________________________________________________
__________________
conv2d_53 (Conv2D)             (None, 4, 4, 256)    213248
max_pooling2d_11[0][0]
__________________________________________________________________________________
__________________
conv2d_55 (Conv2D)             (None, 4, 4, 320)    461120      conv2d_54[0][0]
__________________________________________________________________________________
__________________
conv2d_57 (Conv2D)             (None, 4, 4, 128)    102528      conv2d_56[0][0]
__________________________________________________________________________________
__________________
conv2d_58 (Conv2D)             (None, 4, 4, 128)    106624
max_pooling2d_12[0][0]
__________________________________________________________________________________
__________________
concatenate_8 (Concatenate)    (None, 4, 4, 832)    0           conv2d_53[0][0]
                                                                conv2d_55[0][0]
                                                                conv2d_57[0][0]
                                                                conv2d_58[0][0]
__________________________________________________________________________________
```

```
__________________
conv2d_60 (Conv2D)              (None, 4, 4, 192)    159936
concatenate_8[0][0]
________________________________________________________________________________
__________________
conv2d_62 (Conv2D)              (None, 4, 4, 48)     39984
concatenate_8[0][0]
________________________________________________________________________________
__________________
max_pooling2d_13 (MaxPooling2D) (None, 4, 4, 832)    0
concatenate_8[0][0]
________________________________________________________________________________
__________________
average_pooling2d (AveragePooli (None, 2, 2, 512)    0
concatenate_3[0][0]
________________________________________________________________________________
__________________
average_pooling2d_1 (AveragePoo (None, 2, 2, 528)    0
concatenate_6[0][0]
________________________________________________________________________________
__________________
conv2d_59 (Conv2D)              (None, 4, 4, 384)    319872
concatenate_8[0][0]
________________________________________________________________________________
__________________
conv2d_61 (Conv2D)              (None, 4, 4, 384)    663936      conv2d_60[0][0]
________________________________________________________________________________
__________________
conv2d_63 (Conv2D)              (None, 4, 4, 128)    153728      conv2d_62[0][0]
________________________________________________________________________________
__________________
conv2d_64 (Conv2D)              (None, 4, 4, 128)    106624
max_pooling2d_13[0][0]
________________________________________________________________________________
__________________
conv2d_27 (Conv2D)              (None, 2, 2, 128)    65664
average_pooling2d[0][0]
________________________________________________________________________________
__________________
conv2d_46 (Conv2D)              (None, 2, 2, 128)    67712
average_pooling2d_1[0][0]
________________________________________________________________________________
__________________
concatenate_9 (Concatenate)     (None, 4, 4, 1024)   0           conv2d_59[0][0]
                                                                 conv2d_61[0][0]
                                                                 conv2d_63[0][0]
                                                                 conv2d_64[0][0]
________________________________________________________________________________
```

```
__________________
flatten (Flatten)               (None, 512)          0           conv2d_27[0][0]
________________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 512)          0           conv2d_46[0][0]
________________________________________________________________________________
__________________
global_average_pooling2d (Globa (None, 1024)         0
concatenate_9[0][0]
________________________________________________________________________________
__________________
dense (Dense)                   (None, 1024)         525312      flatten[0][0]
________________________________________________________________________________
__________________
dense_1 (Dense)                 (None, 1024)         525312      flatten_1[0][0]
________________________________________________________________________________
__________________
dropout_2 (Dropout)             (None, 1024)         0
global_average_pooling2d[0][0]
________________________________________________________________________________
__________________
dropout (Dropout)               (None, 1024)         0           dense[0][0]
________________________________________________________________________________
__________________
dropout_1 (Dropout)             (None, 1024)         0           dense_1[0][0]
________________________________________________________________________________
__________________
OUTPUT (Dense)                  (None, 8)            8200        dropout_2[0][0]
________________________________________________________________________________
__________________
Softmax_1 (Dense)               (None, 8)            8200        dropout[0][0]
________________________________________________________________________________
__________________
Softmax_2 (Dense)               (None, 8)            8200        dropout_1[0][0]
================================================================================
==================
Total params: 7,182,152
Trainable params: 7,182,152
Non-trainable params: 0
________________________________________________________________________________
__________________
```

[12]:


input_2: InputLayer
conv2d_6: Conv2D
max_pooling2d_1: MaxPooling2D
conv2d_7: Conv2D
conv2d_8: Conv2D
max_pooling2d_2: MaxPooling2D
conv2d_10: Conv2D
conv2d_12: Conv2D
max_pooling2d_3: MaxPooling2D
conv2d_11: Conv2D
conv2d_13: Conv2D
conv2d_14: Conv2D
conv2d_9: Conv2D
concatenate_1: Concatenate
conv2d_16: Conv2D
conv2d_18: Conv2D
max_pooling2d_4: MaxPooling2D
conv2d_17: Conv2D
conv2d_19: Conv2D
conv2d_20: Conv2D
conv2d_15: Conv2D
concatenate_2: Concatenate
max_pooling2d_5: MaxPooling2D
conv2d_22: Conv2D
conv2d_24: Conv2D
max_pooling2d_6: MaxPooling2D
conv2d_23: Conv2D
conv2d_25: Conv2D
conv2d_26: Conv2D
conv2d_21: Conv2D
concatenate_3: Concatenate
conv2d_29: Conv2D
conv2d_31: Conv2D
max_pooling2d_7: MaxPooling2D
average_pooling2d: AveragePooling2D
conv2d_30: Conv2D
conv2d_32: Conv2D
conv2d_33: Conv2D
conv2d_28: Conv2D
conv2d_27: Conv2D
concatenate_4: Concatenate
flatten: Flatten
conv2d_35: Conv2D
conv2d_37: Conv2D
max_pooling2d_8: MaxPooling2D
dense: Dense
conv2d_36: Conv2D
conv2d_38: Conv2D
conv2d_39: Conv2D
conv2d_34: Conv2D
dropout: Dropout
concatenate_5: Concatenate
Softmax_1: Dense
conv2d_41: Conv2D
conv2d_43: Conv2D
max_pooling2d_9: MaxPooling2D
conv2d_42: Conv2D
conv2d_44: Conv2D
conv2d_45: Conv2D
conv2d_40: Conv2D
concatenate_6: Concatenate
conv2d_48: Conv2D
conv2d_50: Conv2D
max_pooling2d_10: MaxPooling2D
average_pooling2d_1: AveragePooling2D
conv2d_49: Conv2D
conv2d_51: Conv2D
conv2d_52: Conv2D
conv2d_47: Conv2D
conv2d_46: Conv2D
concatenate_7: Concatenate
flatten_1: Flatten
max_pooling2d_11: MaxPooling2D
dense_1: Dense
conv2d_54: Conv2D
conv2d_56: Conv2D
max_pooling2d_12: MaxPooling2D
dropout_1: Dropout
conv2d_55: Conv2D
conv2d_57: Conv2D
conv2d_58: Conv2D
conv2d_53: Conv2D
Softmax_2: Dense
concatenate_8: Concatenate
conv2d_60: Conv2D
conv2d_62: Conv2D
max_pooling2d_13: MaxPooling2D
conv2d_61: Conv2D
conv2d_63: Conv2D
conv2d_64: Conv2D
conv2d_59: Conv2D
concatenate_9: Concatenate
global_average_pooling2d: GlobalAveragePooling2D
dropout_2: Dropout
OUTPUT: Dense

[13]:
```
epochs = 50
initial_lrate = 0.01

def decay(epoch, steps=100):
    initial_lrate = 0.01
    drop = 0.96
    epochs_drop = 8
    lrate = initial_lrate * math.pow(drop, math.floor((1+epoch)/epochs_drop))
    return lrate

sgd = SGD(lr=initial_lrate, momentum=0.9, nesterov=False)

lr_sc = LearningRateScheduler(decay, verbose=1)

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

[14]:
```
history = model.fit(train_images, [train_labels,train_labels,train_labels],␣
 ↪validation_data=(test_images, [test_labels,test_labels,test_labels]),␣
 ↪epochs=epochs, batch_size=256, callbacks=[lr_sc])
```

```
Epoch 00001: LearningRateScheduler reducing learning rate to 0.01.
Epoch 1/50
 2/40 [>…] - ETA: 28s - loss: 6.2378 - OUTPUT_loss:
2.0805 - Softmax_1_loss: 2.0780 - Softmax_2_loss: 2.0793 - OUTPUT_accuracy:
0.1113 - Softmax_1_accuracy: 0.1309 - Softmax_2_accuracy:
0.1016WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.5235s vs `on_train_batch_end` time: 0.9779s).
Check your callbacks.
40/40 [==============================] - 76s 2s/step - loss: 6.2381 -
OUTPUT_loss: 2.0800 - Softmax_1_loss: 2.0787 - Softmax_2_loss: 2.0795 -
OUTPUT_accuracy: 0.1202 - Softmax_1_accuracy: 0.1354 - Softmax_2_accuracy:
0.1252 - val_loss: 6.2335 - val_OUTPUT_loss: 2.0794 - val_Softmax_1_loss: 2.0748
- val_Softmax_2_loss: 2.0793 - val_OUTPUT_accuracy: 0.1195 -
val_Softmax_1_accuracy: 0.1925 - val_Softmax_2_accuracy: 0.1195

Epoch 00002: LearningRateScheduler reducing learning rate to 0.01.
Epoch 2/50
40/40 [==============================] - 58s 1s/step - loss: 6.2260 -
OUTPUT_loss: 2.0785 - Softmax_1_loss: 2.0700 - Softmax_2_loss: 2.0775 -
OUTPUT_accuracy: 0.1307 - Softmax_1_accuracy: 0.1569 - Softmax_2_accuracy:
0.1378 - val_loss: 6.2009 - val_OUTPUT_loss: 2.0758 - val_Softmax_1_loss: 2.0516
- val_Softmax_2_loss: 2.0734 - val_OUTPUT_accuracy: 0.2115 -
```

```
val_Softmax_1_accuracy: 0.2090 - val_Softmax_2_accuracy: 0.1990

Epoch 00003: LearningRateScheduler reducing learning rate to 0.01.
Epoch 3/50
40/40 [==============================] - 60s 2s/step - loss: 6.1570 -
OUTPUT_loss: 2.0654 - Softmax_1_loss: 2.0314 - Softmax_2_loss: 2.0602 -
OUTPUT_accuracy: 0.1744 - Softmax_1_accuracy: 0.1802 - Softmax_2_accuracy:
0.1612 - val_loss: 6.0465 - val_OUTPUT_loss: 2.0305 - val_Softmax_1_loss: 2.0004
- val_Softmax_2_loss: 2.0156 - val_OUTPUT_accuracy: 0.2305 -
val_Softmax_1_accuracy: 0.2185 - val_Softmax_2_accuracy: 0.2175

Epoch 00004: LearningRateScheduler reducing learning rate to 0.01.
Epoch 4/50
40/40 [==============================] - 61s 2s/step - loss: 5.8924 -
OUTPUT_loss: 1.9629 - Softmax_1_loss: 1.9611 - Softmax_2_loss: 1.9683 -
OUTPUT_accuracy: 0.2097 - Softmax_1_accuracy: 0.2078 - Softmax_2_accuracy:
0.2101 - val_loss: 5.7264 - val_OUTPUT_loss: 1.9135 - val_Softmax_1_loss: 1.9060
- val_Softmax_2_loss: 1.9068 - val_OUTPUT_accuracy: 0.2400 -
val_Softmax_1_accuracy: 0.2655 - val_Softmax_2_accuracy: 0.2410

Epoch 00005: LearningRateScheduler reducing learning rate to 0.01.
Epoch 5/50
40/40 [==============================] - 59s 1s/step - loss: 5.7030 -
OUTPUT_loss: 1.8967 - Softmax_1_loss: 1.8983 - Softmax_2_loss: 1.9080 -
OUTPUT_accuracy: 0.2458 - Softmax_1_accuracy: 0.2504 - Softmax_2_accuracy:
0.2338 - val_loss: 5.8739 - val_OUTPUT_loss: 1.9272 - val_Softmax_1_loss: 2.0244
- val_Softmax_2_loss: 1.9224 - val_OUTPUT_accuracy: 0.2025 -
val_Softmax_1_accuracy: 0.1935 - val_Softmax_2_accuracy: 0.1980

Epoch 00006: LearningRateScheduler reducing learning rate to 0.01.
Epoch 6/50
40/40 [==============================] - 60s 2s/step - loss: 5.6529 -
OUTPUT_loss: 1.8677 - Softmax_1_loss: 1.9008 - Softmax_2_loss: 1.8844 -
OUTPUT_accuracy: 0.2480 - Softmax_1_accuracy: 0.2390 - Softmax_2_accuracy:
0.2472 - val_loss: 5.3931 - val_OUTPUT_loss: 1.7901 - val_Softmax_1_loss: 1.8033
- val_Softmax_2_loss: 1.7997 - val_OUTPUT_accuracy: 0.2950 -
val_Softmax_1_accuracy: 0.3110 - val_Softmax_2_accuracy: 0.2820

Epoch 00007: LearningRateScheduler reducing learning rate to 0.01.
Epoch 7/50
40/40 [==============================] - 32s 807ms/step - loss: 5.4414 -
OUTPUT_loss: 1.8105 - Softmax_1_loss: 1.8012 - Softmax_2_loss: 1.8297 -
OUTPUT_accuracy: 0.2816 - Softmax_1_accuracy: 0.2970 - Softmax_2_accuracy:
0.2775 - val_loss: 5.6986 - val_OUTPUT_loss: 1.9106 - val_Softmax_1_loss: 1.8835
- val_Softmax_2_loss: 1.9044 - val_OUTPUT_accuracy: 0.2430 -
val_Softmax_1_accuracy: 0.2460 - val_Softmax_2_accuracy: 0.2455

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0096.
```

```
Epoch 8/50
40/40 [==============================] - 22s 560ms/step - loss: 5.4563 -
OUTPUT_loss: 1.8184 - Softmax_1_loss: 1.8176 - Softmax_2_loss: 1.8203 -
OUTPUT_accuracy: 0.2891 - Softmax_1_accuracy: 0.2973 - Softmax_2_accuracy:
0.2905 - val_loss: 5.2625 - val_OUTPUT_loss: 1.7581 - val_Softmax_1_loss: 1.7520
- val_Softmax_2_loss: 1.7525 - val_OUTPUT_accuracy: 0.3075 -
val_Softmax_1_accuracy: 0.3220 - val_Softmax_2_accuracy: 0.3195

Epoch 00009: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 9/50
40/40 [==============================] - 23s 563ms/step - loss: 5.0586 -
OUTPUT_loss: 1.6841 - Softmax_1_loss: 1.6824 - Softmax_2_loss: 1.6921 -
OUTPUT_accuracy: 0.3366 - Softmax_1_accuracy: 0.3406 - Softmax_2_accuracy:
0.3318 - val_loss: 5.6729 - val_OUTPUT_loss: 1.9650 - val_Softmax_1_loss: 1.7989
- val_Softmax_2_loss: 1.9090 - val_OUTPUT_accuracy: 0.2625 -
val_Softmax_1_accuracy: 0.3205 - val_Softmax_2_accuracy: 0.2845

Epoch 00010: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 10/50
40/40 [==============================] - 23s 563ms/step - loss: 4.9203 -
OUTPUT_loss: 1.6403 - Softmax_1_loss: 1.6307 - Softmax_2_loss: 1.6493 -
OUTPUT_accuracy: 0.3594 - Softmax_1_accuracy: 0.3631 - Softmax_2_accuracy:
0.3564 - val_loss: 5.1822 - val_OUTPUT_loss: 1.7407 - val_Softmax_1_loss: 1.7411
- val_Softmax_2_loss: 1.7004 - val_OUTPUT_accuracy: 0.3240 -
val_Softmax_1_accuracy: 0.3525 - val_Softmax_2_accuracy: 0.3425

Epoch 00011: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 11/50
40/40 [==============================] - 23s 565ms/step - loss: 4.7377 -
OUTPUT_loss: 1.5797 - Softmax_1_loss: 1.5774 - Softmax_2_loss: 1.5805 -
OUTPUT_accuracy: 0.3862 - Softmax_1_accuracy: 0.3932 - Softmax_2_accuracy:
0.3872 - val_loss: 5.1441 - val_OUTPUT_loss: 1.7678 - val_Softmax_1_loss: 1.6014
- val_Softmax_2_loss: 1.7749 - val_OUTPUT_accuracy: 0.3085 -
val_Softmax_1_accuracy: 0.3730 - val_Softmax_2_accuracy: 0.3130

Epoch 00012: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 12/50
40/40 [==============================] - 23s 564ms/step - loss: 4.4626 -
OUTPUT_loss: 1.4812 - Softmax_1_loss: 1.4931 - Softmax_2_loss: 1.4882 -
OUTPUT_accuracy: 0.4222 - Softmax_1_accuracy: 0.4213 - Softmax_2_accuracy:
0.4202 - val_loss: 5.8929 - val_OUTPUT_loss: 2.0213 - val_Softmax_1_loss: 1.8636
- val_Softmax_2_loss: 2.0080 - val_OUTPUT_accuracy: 0.2185 -
val_Softmax_1_accuracy: 0.2680 - val_Softmax_2_accuracy: 0.2275

Epoch 00013: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 13/50
40/40 [==============================] - 23s 564ms/step - loss: 5.2081 -
OUTPUT_loss: 1.7512 - Softmax_1_loss: 1.7089 - Softmax_2_loss: 1.7479 -
```

```
OUTPUT_accuracy: 0.3144 - Softmax_1_accuracy: 0.3289 - Softmax_2_accuracy:
0.3141 - val_loss: 5.0090 - val_OUTPUT_loss: 1.7137 - val_Softmax_1_loss: 1.6147
- val_Softmax_2_loss: 1.6806 - val_OUTPUT_accuracy: 0.3580 -
val_Softmax_1_accuracy: 0.3925 - val_Softmax_2_accuracy: 0.3740

Epoch 00014: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 14/50
40/40 [==============================] - 23s 566ms/step - loss: 4.4792 -
OUTPUT_loss: 1.4936 - Softmax_1_loss: 1.4854 - Softmax_2_loss: 1.5002 -
OUTPUT_accuracy: 0.4258 - Softmax_1_accuracy: 0.4297 - Softmax_2_accuracy:
0.4236 - val_loss: 7.5746 - val_OUTPUT_loss: 2.5663 - val_Softmax_1_loss: 2.6947
- val_Softmax_2_loss: 2.3136 - val_OUTPUT_accuracy: 0.1910 -
val_Softmax_1_accuracy: 0.1925 - val_Softmax_2_accuracy: 0.2095

Epoch 00015: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 15/50
40/40 [==============================] - 23s 564ms/step - loss: 4.9839 -
OUTPUT_loss: 1.6537 - Softmax_1_loss: 1.6829 - Softmax_2_loss: 1.6473 -
OUTPUT_accuracy: 0.3685 - Softmax_1_accuracy: 0.3582 - Softmax_2_accuracy:
0.3707 - val_loss: 4.5518 - val_OUTPUT_loss: 1.5021 - val_Softmax_1_loss: 1.5368
- val_Softmax_2_loss: 1.5129 - val_OUTPUT_accuracy: 0.4095 -
val_Softmax_1_accuracy: 0.4160 - val_Softmax_2_accuracy: 0.4090

Epoch 00016: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 16/50
40/40 [==============================] - 23s 566ms/step - loss: 4.3151 -
OUTPUT_loss: 1.4291 - Softmax_1_loss: 1.4450 - Softmax_2_loss: 1.4410 -
OUTPUT_accuracy: 0.4376 - Softmax_1_accuracy: 0.4355 - Softmax_2_accuracy:
0.4338 - val_loss: 5.2748 - val_OUTPUT_loss: 1.8029 - val_Softmax_1_loss: 1.7123
- val_Softmax_2_loss: 1.7596 - val_OUTPUT_accuracy: 0.3380 -
val_Softmax_1_accuracy: 0.3580 - val_Softmax_2_accuracy: 0.3545

Epoch 00017: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 17/50
40/40 [==============================] - 23s 566ms/step - loss: 4.2401 -
OUTPUT_loss: 1.4129 - Softmax_1_loss: 1.4116 - Softmax_2_loss: 1.4155 -
OUTPUT_accuracy: 0.4531 - Softmax_1_accuracy: 0.4570 - Softmax_2_accuracy:
0.4531 - val_loss: 4.2604 - val_OUTPUT_loss: 1.3928 - val_Softmax_1_loss: 1.4321
- val_Softmax_2_loss: 1.4355 - val_OUTPUT_accuracy: 0.4565 -
val_Softmax_1_accuracy: 0.4630 - val_Softmax_2_accuracy: 0.4545

Epoch 00018: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 18/50
40/40 [==============================] - 23s 565ms/step - loss: 3.9672 -
OUTPUT_loss: 1.3237 - Softmax_1_loss: 1.3192 - Softmax_2_loss: 1.3244 -
OUTPUT_accuracy: 0.4833 - Softmax_1_accuracy: 0.4916 - Softmax_2_accuracy:
0.4825 - val_loss: 4.3107 - val_OUTPUT_loss: 1.4450 - val_Softmax_1_loss: 1.4249
- val_Softmax_2_loss: 1.4409 - val_OUTPUT_accuracy: 0.4270 -
```

```
val_Softmax_1_accuracy: 0.4245 - val_Softmax_2_accuracy: 0.4275

Epoch 00019: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 19/50
40/40 [==============================] - 23s 565ms/step - loss: 3.8298 -
OUTPUT_loss: 1.2729 - Softmax_1_loss: 1.2791 - Softmax_2_loss: 1.2778 -
OUTPUT_accuracy: 0.5065 - Softmax_1_accuracy: 0.5143 - Softmax_2_accuracy:
0.5077 - val_loss: 4.2529 - val_OUTPUT_loss: 1.4734 - val_Softmax_1_loss: 1.3554
- val_Softmax_2_loss: 1.4242 - val_OUTPUT_accuracy: 0.4435 -
val_Softmax_1_accuracy: 0.4835 - val_Softmax_2_accuracy: 0.4705

Epoch 00020: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 20/50
40/40 [==============================] - 23s 568ms/step - loss: 3.8400 -
OUTPUT_loss: 1.2779 - Softmax_1_loss: 1.2700 - Softmax_2_loss: 1.2920 -
OUTPUT_accuracy: 0.5160 - Softmax_1_accuracy: 0.5203 - Softmax_2_accuracy:
0.5107 - val_loss: 3.8997 - val_OUTPUT_loss: 1.3221 - val_Softmax_1_loss: 1.2833
- val_Softmax_2_loss: 1.2943 - val_OUTPUT_accuracy: 0.5000 -
val_Softmax_1_accuracy: 0.5130 - val_Softmax_2_accuracy: 0.5100

Epoch 00021: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 21/50
40/40 [==============================] - 23s 569ms/step - loss: 3.6066 -
OUTPUT_loss: 1.2023 - Softmax_1_loss: 1.1995 - Softmax_2_loss: 1.2049 -
OUTPUT_accuracy: 0.5367 - Softmax_1_accuracy: 0.5434 - Softmax_2_accuracy:
0.5371 - val_loss: 3.7307 - val_OUTPUT_loss: 1.2403 - val_Softmax_1_loss: 1.2428
- val_Softmax_2_loss: 1.2476 - val_OUTPUT_accuracy: 0.5355 -
val_Softmax_1_accuracy: 0.5375 - val_Softmax_2_accuracy: 0.5410

Epoch 00022: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 22/50
40/40 [==============================] - 23s 567ms/step - loss: 3.4835 -
OUTPUT_loss: 1.1582 - Softmax_1_loss: 1.1588 - Softmax_2_loss: 1.1665 -
OUTPUT_accuracy: 0.5573 - Softmax_1_accuracy: 0.5576 - Softmax_2_accuracy:
0.5543 - val_loss: 4.0985 - val_OUTPUT_loss: 1.4268 - val_Softmax_1_loss: 1.2844
- val_Softmax_2_loss: 1.3874 - val_OUTPUT_accuracy: 0.4570 -
val_Softmax_1_accuracy: 0.4930 - val_Softmax_2_accuracy: 0.4695

Epoch 00023: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 23/50
40/40 [==============================] - 23s 568ms/step - loss: 3.4658 -
OUTPUT_loss: 1.1523 - Softmax_1_loss: 1.1570 - Softmax_2_loss: 1.1565 -
OUTPUT_accuracy: 0.5618 - Softmax_1_accuracy: 0.5581 - Softmax_2_accuracy:
0.5597 - val_loss: 3.4725 - val_OUTPUT_loss: 1.1674 - val_Softmax_1_loss: 1.1540
- val_Softmax_2_loss: 1.1511 - val_OUTPUT_accuracy: 0.5660 -
val_Softmax_1_accuracy: 0.5785 - val_Softmax_2_accuracy: 0.5825

Epoch 00024: LearningRateScheduler reducing learning rate to
```

```
0.008847359999999999.
Epoch 24/50
40/40 [==============================] - 23s 566ms/step - loss: 3.2603 -
OUTPUT_loss: 1.0851 - Softmax_1_loss: 1.0856 - Softmax_2_loss: 1.0897 -
OUTPUT_accuracy: 0.5924 - Softmax_1_accuracy: 0.5889 - Softmax_2_accuracy:
0.5876 - val_loss: 4.4006 - val_OUTPUT_loss: 1.5657 - val_Softmax_1_loss: 1.3881
- val_Softmax_2_loss: 1.4468 - val_OUTPUT_accuracy: 0.4600 -
val_Softmax_1_accuracy: 0.4945 - val_Softmax_2_accuracy: 0.4820

Epoch 00025: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 25/50
40/40 [==============================] - 23s 567ms/step - loss: 3.9236 -
OUTPUT_loss: 1.3001 - Softmax_1_loss: 1.3077 - Softmax_2_loss: 1.3157 -
OUTPUT_accuracy: 0.5050 - Softmax_1_accuracy: 0.5059 - Softmax_2_accuracy:
0.5027 - val_loss: 3.4034 - val_OUTPUT_loss: 1.1397 - val_Softmax_1_loss: 1.1338
- val_Softmax_2_loss: 1.1299 - val_OUTPUT_accuracy: 0.5745 -
val_Softmax_1_accuracy: 0.5790 - val_Softmax_2_accuracy: 0.5715

Epoch 00026: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 26/50
40/40 [==============================] - 23s 567ms/step - loss: 3.2459 -
OUTPUT_loss: 1.0798 - Softmax_1_loss: 1.0802 - Softmax_2_loss: 1.0859 -
OUTPUT_accuracy: 0.5888 - Softmax_1_accuracy: 0.5891 - Softmax_2_accuracy:
0.5856 - val_loss: 3.4042 - val_OUTPUT_loss: 1.1540 - val_Softmax_1_loss: 1.1075
- val_Softmax_2_loss: 1.1426 - val_OUTPUT_accuracy: 0.5575 -
val_Softmax_1_accuracy: 0.5870 - val_Softmax_2_accuracy: 0.5645

Epoch 00027: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 27/50
40/40 [==============================] - 23s 568ms/step - loss: 3.0722 -
OUTPUT_loss: 1.0155 - Softmax_1_loss: 1.0257 - Softmax_2_loss: 1.0310 -
OUTPUT_accuracy: 0.6114 - Softmax_1_accuracy: 0.6141 - Softmax_2_accuracy:
0.6105 - val_loss: 3.2974 - val_OUTPUT_loss: 1.0850 - val_Softmax_1_loss: 1.1174
- val_Softmax_2_loss: 1.0949 - val_OUTPUT_accuracy: 0.5865 -
val_Softmax_1_accuracy: 0.5755 - val_Softmax_2_accuracy: 0.5890

Epoch 00028: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 28/50
40/40 [==============================] - 23s 569ms/step - loss: 2.9450 -
OUTPUT_loss: 0.9732 - Softmax_1_loss: 0.9897 - Softmax_2_loss: 0.9821 -
OUTPUT_accuracy: 0.6333 - Softmax_1_accuracy: 0.6281 - Softmax_2_accuracy:
0.6334 - val_loss: 4.0322 - val_OUTPUT_loss: 1.3557 - val_Softmax_1_loss: 1.3315
- val_Softmax_2_loss: 1.3450 - val_OUTPUT_accuracy: 0.5175 -
val_Softmax_1_accuracy: 0.5300 - val_Softmax_2_accuracy: 0.5315
```

```
Epoch 00029: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 29/50
40/40 [==============================] - 23s 567ms/step - loss: 3.0890 -
OUTPUT_loss: 1.0268 - Softmax_1_loss: 1.0290 - Softmax_2_loss: 1.0333 -
OUTPUT_accuracy: 0.6241 - Softmax_1_accuracy: 0.6233 - Softmax_2_accuracy:
0.6194 - val_loss: 3.7559 - val_OUTPUT_loss: 1.2649 - val_Softmax_1_loss: 1.2274
- val_Softmax_2_loss: 1.2637 - val_OUTPUT_accuracy: 0.5310 -
val_Softmax_1_accuracy: 0.5525 - val_Softmax_2_accuracy: 0.5330

Epoch 00030: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 30/50
40/40 [==============================] - 23s 567ms/step - loss: 2.8281 -
OUTPUT_loss: 0.9301 - Softmax_1_loss: 0.9531 - Softmax_2_loss: 0.9450 -
OUTPUT_accuracy: 0.6531 - Softmax_1_accuracy: 0.6467 - Softmax_2_accuracy:
0.6504 - val_loss: 2.8133 - val_OUTPUT_loss: 0.9415 - val_Softmax_1_loss: 0.9289
- val_Softmax_2_loss: 0.9428 - val_OUTPUT_accuracy: 0.6470 -
val_Softmax_1_accuracy: 0.6615 - val_Softmax_2_accuracy: 0.6460

Epoch 00031: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 31/50
40/40 [==============================] - 23s 567ms/step - loss: 2.7152 -
OUTPUT_loss: 0.8959 - Softmax_1_loss: 0.9133 - Softmax_2_loss: 0.9060 -
OUTPUT_accuracy: 0.6670 - Softmax_1_accuracy: 0.6640 - Softmax_2_accuracy:
0.6608 - val_loss: 3.5656 - val_OUTPUT_loss: 1.1958 - val_Softmax_1_loss: 1.1756
- val_Softmax_2_loss: 1.1943 - val_OUTPUT_accuracy: 0.5580 -
val_Softmax_1_accuracy: 0.5680 - val_Softmax_2_accuracy: 0.5580

Epoch 00032: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 32/50
40/40 [==============================] - 23s 568ms/step - loss: 2.7815 -
OUTPUT_loss: 0.9190 - Softmax_1_loss: 0.9298 - Softmax_2_loss: 0.9328 -
OUTPUT_accuracy: 0.6575 - Softmax_1_accuracy: 0.6515 - Softmax_2_accuracy:
0.6549 - val_loss: 3.3008 - val_OUTPUT_loss: 1.1112 - val_Softmax_1_loss: 1.1035
- val_Softmax_2_loss: 1.0860 - val_OUTPUT_accuracy: 0.6080 -
val_Softmax_1_accuracy: 0.6020 - val_Softmax_2_accuracy: 0.6140

Epoch 00033: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 33/50
40/40 [==============================] - 23s 568ms/step - loss: 2.6493 -
OUTPUT_loss: 0.8755 - Softmax_1_loss: 0.8880 - Softmax_2_loss: 0.8858 -
OUTPUT_accuracy: 0.6739 - Softmax_1_accuracy: 0.6692 - Softmax_2_accuracy:
0.6676 - val_loss: 6.0949 - val_OUTPUT_loss: 2.1007 - val_Softmax_1_loss: 1.8723
```

```
- val_Softmax_2_loss: 2.1218 - val_OUTPUT_accuracy: 0.3770 -
val_Softmax_1_accuracy: 0.4125 - val_Softmax_2_accuracy: 0.3840

Epoch 00034: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 34/50
40/40 [==============================] - 23s 566ms/step - loss: 3.4261 -
OUTPUT_loss: 1.1471 - Softmax_1_loss: 1.1279 - Softmax_2_loss: 1.1511 -
OUTPUT_accuracy: 0.5782 - Softmax_1_accuracy: 0.5892 - Softmax_2_accuracy:
0.5754 - val_loss: 3.4782 - val_OUTPUT_loss: 1.1641 - val_Softmax_1_loss: 1.1742
- val_Softmax_2_loss: 1.1398 - val_OUTPUT_accuracy: 0.5715 -
val_Softmax_1_accuracy: 0.5760 - val_Softmax_2_accuracy: 0.5820

Epoch 00035: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 35/50
40/40 [==============================] - 23s 567ms/step - loss: 2.6128 -
OUTPUT_loss: 0.8586 - Softmax_1_loss: 0.8836 - Softmax_2_loss: 0.8707 -
OUTPUT_accuracy: 0.6829 - Softmax_1_accuracy: 0.6766 - Softmax_2_accuracy:
0.6794 - val_loss: 3.2507 - val_OUTPUT_loss: 1.0843 - val_Softmax_1_loss: 1.0747
- val_Softmax_2_loss: 1.0918 - val_OUTPUT_accuracy: 0.6065 -
val_Softmax_1_accuracy: 0.6170 - val_Softmax_2_accuracy: 0.5990

Epoch 00036: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 36/50
40/40 [==============================] - 23s 568ms/step - loss: 2.5633 -
OUTPUT_loss: 0.8426 - Softmax_1_loss: 0.8609 - Softmax_2_loss: 0.8598 -
OUTPUT_accuracy: 0.6843 - Softmax_1_accuracy: 0.6792 - Softmax_2_accuracy:
0.6829 - val_loss: 2.6859 - val_OUTPUT_loss: 0.8918 - val_Softmax_1_loss: 0.9061
- val_Softmax_2_loss: 0.8880 - val_OUTPUT_accuracy: 0.6695 -
val_Softmax_1_accuracy: 0.6775 - val_Softmax_2_accuracy: 0.6720

Epoch 00037: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 37/50
40/40 [==============================] - 23s 567ms/step - loss: 2.4281 -
OUTPUT_loss: 0.7974 - Softmax_1_loss: 0.8189 - Softmax_2_loss: 0.8117 -
OUTPUT_accuracy: 0.7030 - Softmax_1_accuracy: 0.6980 - Softmax_2_accuracy:
0.6998 - val_loss: 3.6172 - val_OUTPUT_loss: 1.2336 - val_Softmax_1_loss: 1.1798
- val_Softmax_2_loss: 1.2038 - val_OUTPUT_accuracy: 0.5990 -
val_Softmax_1_accuracy: 0.5940 - val_Softmax_2_accuracy: 0.5995

Epoch 00038: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 38/50
40/40 [==============================] - 23s 567ms/step - loss: 2.4884 -
OUTPUT_loss: 0.8209 - Softmax_1_loss: 0.8345 - Softmax_2_loss: 0.8330 -
```

```
OUTPUT_accuracy: 0.7002 - Softmax_1_accuracy: 0.6952 - Softmax_2_accuracy:
0.6924 - val_loss: 3.1439 - val_OUTPUT_loss: 1.0982 - val_Softmax_1_loss: 0.9850
- val_Softmax_2_loss: 1.0608 - val_OUTPUT_accuracy: 0.5930 -
val_Softmax_1_accuracy: 0.6320 - val_Softmax_2_accuracy: 0.6000

Epoch 00039: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 39/50
40/40 [==============================] - 23s 568ms/step - loss: 2.3307 -
OUTPUT_loss: 0.7709 - Softmax_1_loss: 0.7849 - Softmax_2_loss: 0.7749 -
OUTPUT_accuracy: 0.7084 - Softmax_1_accuracy: 0.7051 - Softmax_2_accuracy:
0.7086 - val_loss: 2.9164 - val_OUTPUT_loss: 0.9848 - val_Softmax_1_loss: 0.9352
- val_Softmax_2_loss: 0.9964 - val_OUTPUT_accuracy: 0.6435 -
val_Softmax_1_accuracy: 0.6600 - val_Softmax_2_accuracy: 0.6375

Epoch 00040: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 40/50
40/40 [==============================] - 23s 572ms/step - loss: 2.2766 -
OUTPUT_loss: 0.7438 - Softmax_1_loss: 0.7699 - Softmax_2_loss: 0.7629 -
OUTPUT_accuracy: 0.7220 - Softmax_1_accuracy: 0.7161 - Softmax_2_accuracy:
0.7177 - val_loss: 2.4460 - val_OUTPUT_loss: 0.8220 - val_Softmax_1_loss: 0.8140
- val_Softmax_2_loss: 0.8101 - val_OUTPUT_accuracy: 0.6985 -
val_Softmax_1_accuracy: 0.6980 - val_Softmax_2_accuracy: 0.6995

Epoch 00041: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 41/50
40/40 [==============================] - 23s 574ms/step - loss: 2.0873 -
OUTPUT_loss: 0.6809 - Softmax_1_loss: 0.7163 - Softmax_2_loss: 0.6901 -
OUTPUT_accuracy: 0.7494 - Softmax_1_accuracy: 0.7368 - Softmax_2_accuracy:
0.7433 - val_loss: 2.8081 - val_OUTPUT_loss: 0.9376 - val_Softmax_1_loss: 0.9417
- val_Softmax_2_loss: 0.9288 - val_OUTPUT_accuracy: 0.6610 -
val_Softmax_1_accuracy: 0.6730 - val_Softmax_2_accuracy: 0.6640

Epoch 00042: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 42/50
40/40 [==============================] - 23s 572ms/step - loss: 2.2156 -
OUTPUT_loss: 0.7238 - Softmax_1_loss: 0.7553 - Softmax_2_loss: 0.7365 -
OUTPUT_accuracy: 0.7334 - Softmax_1_accuracy: 0.7235 - Softmax_2_accuracy:
0.7288 - val_loss: 2.5242 - val_OUTPUT_loss: 0.8492 - val_Softmax_1_loss: 0.8333
- val_Softmax_2_loss: 0.8417 - val_OUTPUT_accuracy: 0.6875 -
val_Softmax_1_accuracy: 0.7015 - val_Softmax_2_accuracy: 0.6890

Epoch 00043: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 43/50
40/40 [==============================] - 23s 567ms/step - loss: 1.9847 -
OUTPUT_loss: 0.6456 - Softmax_1_loss: 0.6779 - Softmax_2_loss: 0.6611 -
OUTPUT_accuracy: 0.7564 - Softmax_1_accuracy: 0.7515 - Softmax_2_accuracy:
0.7564 - val_loss: 2.5666 - val_OUTPUT_loss: 0.8688 - val_Softmax_1_loss: 0.8352
```

```
- val_Softmax_2_loss: 0.8626 - val_OUTPUT_accuracy: 0.6985 -
val_Softmax_1_accuracy: 0.6950 - val_Softmax_2_accuracy: 0.7000

Epoch 00044: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 44/50
40/40 [==============================] - 23s 567ms/step - loss: 1.8612 -
OUTPUT_loss: 0.6009 - Softmax_1_loss: 0.6404 - Softmax_2_loss: 0.6199 -
OUTPUT_accuracy: 0.7732 - Softmax_1_accuracy: 0.7625 - Softmax_2_accuracy:
0.7673 - val_loss: 2.5650 - val_OUTPUT_loss: 0.8445 - val_Softmax_1_loss: 0.8687
- val_Softmax_2_loss: 0.8518 - val_OUTPUT_accuracy: 0.7030 -
val_Softmax_1_accuracy: 0.6865 - val_Softmax_2_accuracy: 0.6900

Epoch 00045: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 45/50
40/40 [==============================] - 23s 567ms/step - loss: 2.1314 -
OUTPUT_loss: 0.6967 - Softmax_1_loss: 0.7288 - Softmax_2_loss: 0.7059 -
OUTPUT_accuracy: 0.7433 - Softmax_1_accuracy: 0.7344 - Softmax_2_accuracy:
0.7378 - val_loss: 3.1333 - val_OUTPUT_loss: 1.0662 - val_Softmax_1_loss: 1.0009
- val_Softmax_2_loss: 1.0662 - val_OUTPUT_accuracy: 0.6400 -
val_Softmax_1_accuracy: 0.6555 - val_Softmax_2_accuracy: 0.6425

Epoch 00046: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 46/50
40/40 [==============================] - 23s 567ms/step - loss: 2.0166 -
OUTPUT_loss: 0.6545 - Softmax_1_loss: 0.6932 - Softmax_2_loss: 0.6689 -
OUTPUT_accuracy: 0.7584 - Softmax_1_accuracy: 0.7482 - Softmax_2_accuracy:
0.7562 - val_loss: 2.8666 - val_OUTPUT_loss: 0.9801 - val_Softmax_1_loss: 0.9215
- val_Softmax_2_loss: 0.9650 - val_OUTPUT_accuracy: 0.6630 -
val_Softmax_1_accuracy: 0.6695 - val_Softmax_2_accuracy: 0.6640

Epoch 00047: LearningRateScheduler reducing learning rate to 0.008153726976.
Epoch 47/50
40/40 [==============================] - 23s 570ms/step - loss: 1.7691 -
OUTPUT_loss: 0.5698 - Softmax_1_loss: 0.6162 - Softmax_2_loss: 0.5832 -
OUTPUT_accuracy: 0.7842 - Softmax_1_accuracy: 0.7698 - Softmax_2_accuracy:
0.7829 - val_loss: 2.6169 - val_OUTPUT_loss: 0.8805 - val_Softmax_1_loss: 0.8646
- val_Softmax_2_loss: 0.8717 - val_OUTPUT_accuracy: 0.7000 -
val_Softmax_1_accuracy: 0.6895 - val_Softmax_2_accuracy: 0.6980

Epoch 00048: LearningRateScheduler reducing learning rate to
0.007827577896959998.
Epoch 48/50
40/40 [==============================] - 23s 568ms/step - loss: 1.7475 -
OUTPUT_loss: 0.5602 - Softmax_1_loss: 0.6086 - Softmax_2_loss: 0.5787 -
OUTPUT_accuracy: 0.7921 - Softmax_1_accuracy: 0.7776 - Softmax_2_accuracy:
0.7872 - val_loss: 2.8409 - val_OUTPUT_loss: 0.9324 - val_Softmax_1_loss: 0.9569
- val_Softmax_2_loss: 0.9517 - val_OUTPUT_accuracy: 0.6890 -
val_Softmax_1_accuracy: 0.6690 - val_Softmax_2_accuracy: 0.6820
```

```
Epoch 00049: LearningRateScheduler reducing learning rate to
0.007827577896959998.
Epoch 49/50
40/40 [==============================] - 23s 567ms/step - loss: 1.9356 -
OUTPUT_loss: 0.6241 - Softmax_1_loss: 0.6667 - Softmax_2_loss: 0.6447 -
OUTPUT_accuracy: 0.7663 - Softmax_1_accuracy: 0.7556 - Softmax_2_accuracy:
0.7609 - val_loss: 3.0808 - val_OUTPUT_loss: 1.0472 - val_Softmax_1_loss: 0.9758
- val_Softmax_2_loss: 1.0578 - val_OUTPUT_accuracy: 0.6675 -
val_Softmax_1_accuracy: 0.6690 - val_Softmax_2_accuracy: 0.6595

Epoch 00050: LearningRateScheduler reducing learning rate to
0.007827577896959998.
Epoch 50/50
40/40 [==============================] - 23s 567ms/step - loss: 1.8910 -
OUTPUT_loss: 0.6181 - Softmax_1_loss: 0.6414 - Softmax_2_loss: 0.6315 -
OUTPUT_accuracy: 0.7703 - Softmax_1_accuracy: 0.7609 - Softmax_2_accuracy:
0.7657 - val_loss: 2.6695 - val_OUTPUT_loss: 0.8794 - val_Softmax_1_loss: 0.8799
- val_Softmax_2_loss: 0.9102 - val_OUTPUT_accuracy: 0.6925 -
val_Softmax_1_accuracy: 0.6975 - val_Softmax_2_accuracy: 0.6885
```

[17]:
```
model.save('googlenet_n128_colour.h5')
```

[20]:
```
from keras.models import load_model
model=load_model('googlenet_n128_colour.h5')
```

[24]:
```
score = model.evaluate(test_images,test_labels,verbose=0)
print('Test Loss - ',score[3])
print('Test Accuracy - ',score[5])
```

```
Test Loss -  0.9102452397346497
Test Accuracy -  0.6974999904632568
```

0.11 ## Residual Module

- Residual Model solves the problem of vanishing gradient. The idea is to add a shortcut layer for the flow of information from one layer to another which improves the performance of the model.
- Let H(x) be the underlying mapping to be fit by a deep network, where x is the input to the first layer and the output will be F(x).The residual function is H(x)-x and after approximation we achieve F(x)+x. So we add the input `x' to the output function before feeding it to the activation layer.
- Ref: https://machinelearningmastery.com/how-to-implement-major-architecture-innovations-f

[11]:
```
#Residual Block :- 2 Conv layer, 64 : 3x3 filters, Relu activation function
def residual_module(layer_in, n_filters):
    merge_input = layer_in
```

```
    # check if the number of filters needs to be increase, assumes channels␣
↪last format
    if layer_in.shape[-1] != n_filters:
        merge_input = Conv2D(n_filters, (1,1), padding='same',␣
↪activation='relu', kernel_initializer='he_normal')(layer_in)
    # conv1
    conv1 = Conv2D(n_filters, (3,3), padding='same', activation='relu',␣
↪kernel_initializer='he_normal')(layer_in)
    # conv2
    conv2 = Conv2D(n_filters, (3,3), padding='same', activation='linear',␣
↪kernel_initializer='he_normal')(conv1)

    # add filters, assumes filters/channels last
    layer_out = add([conv2, merge_input])
    # activation function
    layer_out = Activation('relu')(layer_out)
    return layer_out
```

0.12 ResNet using residual module

[12]:
```
# define model input
visible = Input(shape=(128,128,3))

layer = Conv2D(32, (3,3), padding='same', activation='relu',␣
↪kernel_initializer='he_normal')(visible) #reduced the filtersize from 32 to 8

#Stacks of 4 residual model
layer = residual_module(visible, 64) #First stack
layer = residual_module(layer, 128) #Second stack
layer = residual_module(layer, 256) #Third stack
layer = residual_module(layer, 512) #Fourth stack

layer = AveragePooling2D((5,5),strides=5)(layer) #Average Pooling layer

layer = Flatten()(layer)

layer = Dense(200, activation='relu')(layer) #FC layer with 200
layer = Dense(len(classes), activation='softmax', name='OUTPUT')(layer) #FC␣
↪layer with 8 classes

# create model
model = Model(visible,layer)

# summarize model
model.summary()
```

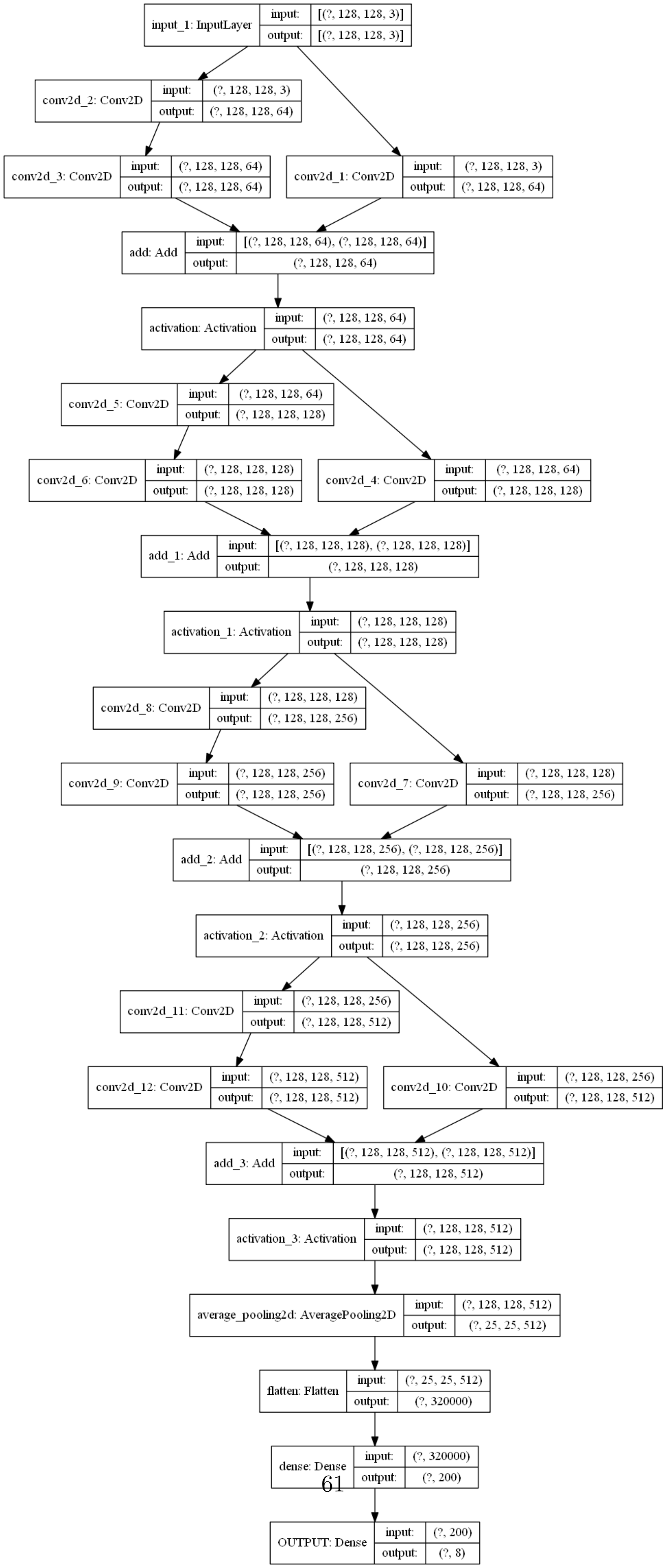
```python
# plot model architecture
plot_model(model, show_shapes=True, to_file='residual_module.png')
```

```
Model: "functional_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 128, 128, 3) 0
________________________________________________________________________________
__________________
conv2d_2 (Conv2D)               (None, 128, 128, 64) 1792        input_1[0][0]
________________________________________________________________________________
__________________
conv2d_3 (Conv2D)               (None, 128, 128, 64) 36928       conv2d_2[0][0]
________________________________________________________________________________
__________________
conv2d_1 (Conv2D)               (None, 128, 128, 64) 256         input_1[0][0]
________________________________________________________________________________
__________________
add (Add)                       (None, 128, 128, 64) 0           conv2d_3[0][0]
                                                                 conv2d_1[0][0]
________________________________________________________________________________
__________________
activation (Activation)         (None, 128, 128, 64) 0           add[0][0]
________________________________________________________________________________
__________________
conv2d_5 (Conv2D)               (None, 128, 128, 128 73856
activation[0][0]
________________________________________________________________________________
__________________
conv2d_6 (Conv2D)               (None, 128, 128, 128 147584      conv2d_5[0][0]
________________________________________________________________________________
__________________
conv2d_4 (Conv2D)               (None, 128, 128, 128 8320
activation[0][0]
________________________________________________________________________________
__________________
add_1 (Add)                     (None, 128, 128, 128 0           conv2d_6[0][0]
                                                                 conv2d_4[0][0]
________________________________________________________________________________
__________________
activation_1 (Activation)       (None, 128, 128, 128 0           add_1[0][0]
________________________________________________________________________________
__________________
conv2d_8 (Conv2D)               (None, 128, 128, 256 295168
```

```
activation_1[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 128, 128, 256 590080      conv2d_8[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 128, 128, 256 33024
activation_1[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 128, 128, 256 0           conv2d_9[0][0]
                                                                 conv2d_7[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 128, 128, 256 0           add_2[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 128, 128, 512 1180160
activation_2[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 128, 128, 512 2359808     conv2d_11[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 128, 128, 512 131584
activation_2[0][0]
__________________________________________________________________________________________________
add_3 (Add)                     (None, 128, 128, 512 0           conv2d_12[0][0]
                                                                 conv2d_10[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 128, 128, 512 0           add_3[0][0]
__________________________________________________________________________________________________
average_pooling2d (AveragePooli (None, 25, 25, 512)  0
activation_3[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 320000)       0
average_pooling2d[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 200)          64000200    flatten[0][0]
__________________________________________________________________________________________________
OUTPUT (Dense)                  (None, 8)            1608        dense[0][0]
==================================================================================================
```

```
==================
Total params: 68,860,368
Trainable params: 68,860,368
Non-trainable params: 0
--------------------------------------------------------------------------------
------------------
```

[12]:


input_1: InputLayer
input: [(?, 128, 128, 3)]
output: [(?, 128, 128, 3)]
conv2d_2: Conv2D
input: (?, 128, 128, 3)
output: (?, 128, 128, 64)
conv2d_3: Conv2D
input: (?, 128, 128, 64)
output: (?, 128, 128, 64)
conv2d_1: Conv2D
input: (?, 128, 128, 3)
output: (?, 128, 128, 64)
add: Add
input: [(?, 128, 128, 64), (?, 128, 128, 64)]
output: (?, 128, 128, 64)
activation: Activation
input: (?, 128, 128, 64)
output: (?, 128, 128, 64)
conv2d_5: Conv2D
input: (?, 128, 128, 64)
output: (?, 128, 128, 128)
conv2d_6: Conv2D
input: (?, 128, 128, 128)
output: (?, 128, 128, 128)
conv2d_4: Conv2D
input: (?, 128, 128, 64)
output: (?, 128, 128, 128)
add_1: Add
input: [(?, 128, 128, 128), (?, 128, 128, 128)]
output: (?, 128, 128, 128)
activation_1: Activation
input: (?, 128, 128, 128)
output: (?, 128, 128, 128)
conv2d_8: Conv2D
input: (?, 128, 128, 128)
output: (?, 128, 128, 256)
conv2d_9: Conv2D
input: (?, 128, 128, 256)
output: (?, 128, 128, 256)
conv2d_7: Conv2D
input: (?, 128, 128, 128)
output: (?, 128, 128, 256)
add_2: Add
input: [(?, 128, 128, 256), (?, 128, 128, 256)]
output: (?, 128, 128, 256)
activation_2: Activation
input: (?, 128, 128, 256)
output: (?, 128, 128, 256)
conv2d_11: Conv2D
input: (?, 128, 128, 256)
output: (?, 128, 128, 512)
conv2d_12: Conv2D
input: (?, 128, 128, 512)
output: (?, 128, 128, 512)
conv2d_10: Conv2D
input: (?, 128, 128, 256)
output: (?, 128, 128, 512)
add_3: Add
input: [(?, 128, 128, 512), (?, 128, 128, 512)]
output: (?, 128, 128, 512)
activation_3: Activation
input: (?, 128, 128, 512)
output: (?, 128, 128, 512)
average_pooling2d: AveragePooling2D
input: (?, 128, 128, 512)
output: (?, 25, 25, 512)
flatten: Flatten
input: (?, 25, 25, 512)
output: (?, 320000)
dense: Dense
input: (?, 320000)
output: (?, 200)
OUTPUT: Dense
input: (?, 200)
output: (?, 8)

0.12.1 Model Summary

- We see that we have 68,860,368 trainable parameters. The number of parameters depend on the size of the filter and also the total number of filters used. As this uses a deep dense layer we see such a huge number of parameters.

[ ]:
```
#Stochastic Gradient Descent optimizer
opt = SGD(lr=0.01,momentum=0.9,decay=0.01)
from keras.callbacks import EarlyStopping

#Early stopping with a patience of 4 on validation loss
early = EarlyStopping(monitor='val_loss', min_delta=0, patience=4,verbose=0,␣
 ↪mode='auto')

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

[14]:
```
#Training our 8 class 1500 images per class training dataset on ResNet
start_time=t.time()
history = model.fit(train_images, train_labels,
          validation_data=(test_images, test_labels),
          epochs=30, batch_size=4, callbacks=[early])
print("\n Training Time: {} seconds".format(t.time()-start_time))
```

```
Epoch 1/30
   2/2500 […] - ETA: 5:35 - loss: 14.6347 -
accuracy: 0.1250WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow
compared to the batch time (batch time: 0.0030s vs `on_train_batch_end` time:
0.2643s). Check your callbacks.
2500/2500 [==============================] - ETA: 0s - loss: 2.3877 - accuracy:
0.1193WARNING:tensorflow:Callbacks method `on_test_batch_end` is slow compared
to the batch time (batch time: 0.0020s vs `on_test_batch_end` time: 0.1023s).
Check your callbacks.
2500/2500 [==============================] - 717s 287ms/step - loss: 2.3877 -
accuracy: 0.1193 - val_loss: 2.0801 - val_accuracy: 0.1195
Epoch 2/30
2500/2500 [==============================] - 717s 287ms/step - loss: 2.0796 -
accuracy: 0.1219 - val_loss: 2.0801 - val_accuracy: 0.1150
Epoch 3/30
2500/2500 [==============================] - 717s 287ms/step - loss: 2.0794 -
accuracy: 0.1278 - val_loss: 2.0797 - val_accuracy: 0.1155
Epoch 4/30
2500/2500 [==============================] - 717s 287ms/step - loss: 2.0789 -
accuracy: 0.1293 - val_loss: 2.0787 - val_accuracy: 0.1245
```

```
Epoch 5/30
2500/2500 [==============================] - 717s 287ms/step - loss: 2.0752 -
accuracy: 0.1377 - val_loss: 2.0536 - val_accuracy: 0.1830
Epoch 6/30
2500/2500 [==============================] - 718s 287ms/step - loss: 2.0437 -
accuracy: 0.1833 - val_loss: 1.9830 - val_accuracy: 0.2025
Epoch 7/30
2500/2500 [==============================] - 718s 287ms/step - loss: 1.9782 -
accuracy: 0.2276 - val_loss: 1.9755 - val_accuracy: 0.2270
Epoch 8/30
2500/2500 [==============================] - 718s 287ms/step - loss: 1.9353 -
accuracy: 0.2563 - val_loss: 1.9223 - val_accuracy: 0.2710
Epoch 9/30
2500/2500 [==============================] - 718s 287ms/step - loss: 1.9047 -
accuracy: 0.2667 - val_loss: 1.9208 - val_accuracy: 0.2605
Epoch 10/30
2500/2500 [==============================] - 718s 287ms/step - loss: 1.8714 -
accuracy: 0.2975 - val_loss: 1.9086 - val_accuracy: 0.2480
Epoch 11/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.8472 -
accuracy: 0.3047 - val_loss: 1.8980 - val_accuracy: 0.2680
Epoch 12/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.8156 -
accuracy: 0.3185 - val_loss: 1.8935 - val_accuracy: 0.2610
Epoch 13/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.7931 -
accuracy: 0.3342 - val_loss: 1.8693 - val_accuracy: 0.2950
Epoch 14/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.7706 -
accuracy: 0.3450 - val_loss: 1.8786 - val_accuracy: 0.2915
Epoch 15/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.7488 -
accuracy: 0.3565 - val_loss: 1.8801 - val_accuracy: 0.2750
Epoch 16/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.7208 -
accuracy: 0.3655 - val_loss: 2.0175 - val_accuracy: 0.2675
Epoch 17/30
2500/2500 [==============================] - 717s 287ms/step - loss: 1.7037 -
accuracy: 0.3749 - val_loss: 1.8811 - val_accuracy: 0.2950
```

```
 Training Time: 12207.317234992981 seconds
```

[17]:
```python
#Saving the model
model.save('resnet_n128_colour.h5')
```

0.13 ## Combined Model 1

- Inspired from AlexNet and Inception module

[36]:

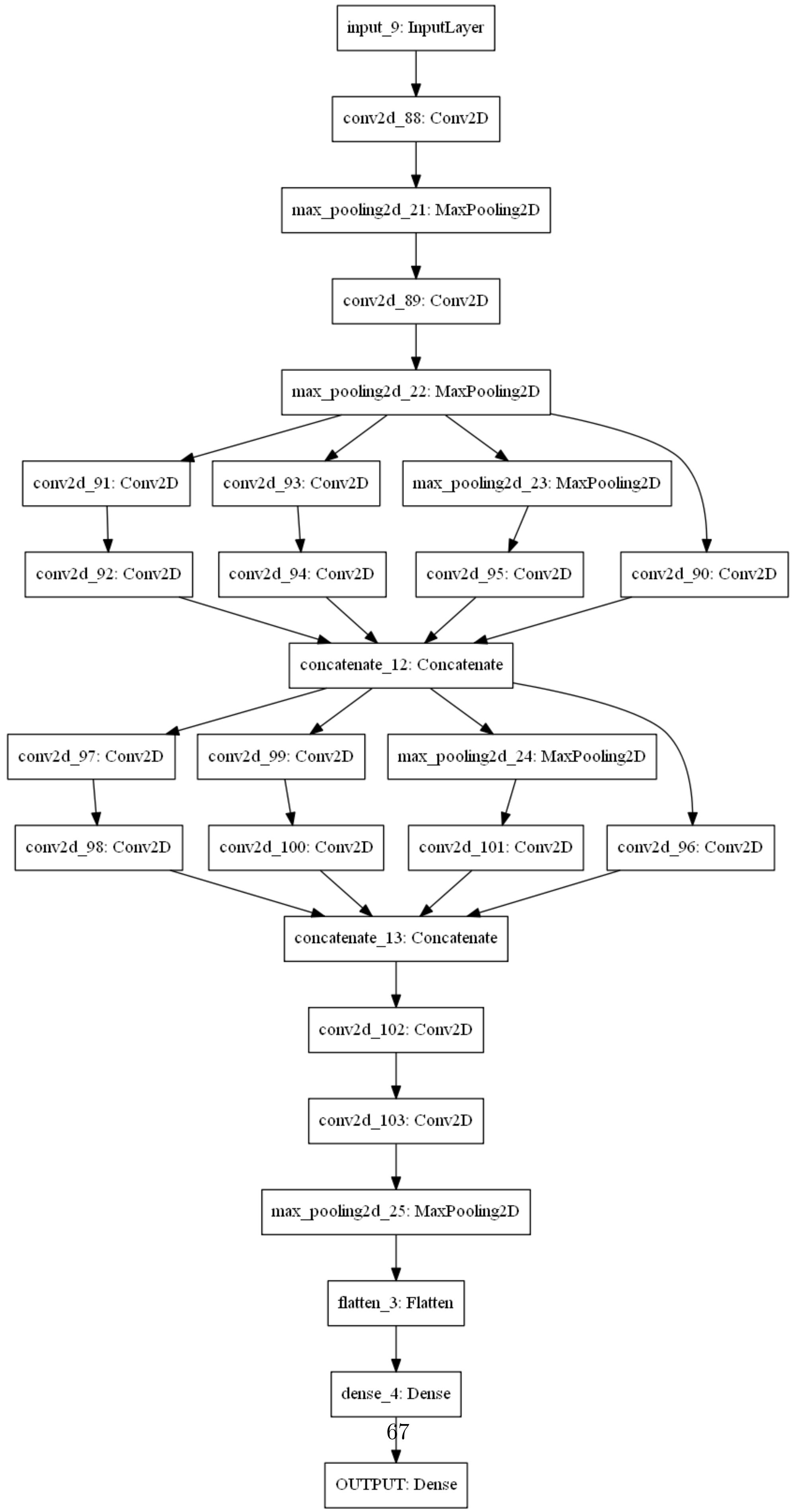
```
input_layer = Input(shape=(n, n, 3))
m = Conv2D(96,kernel_size=(11,11),strides=(4,4),padding="valid",␣
↪activation="relu")(input_layer)
m = MaxPooling2D(pool_size=(3,3),strides=(2,2))(m)
m = Conv2D(256, kernel_size=(5,5), padding="same",strides=(1,1),␣
↪activation="relu")(m)
m = MaxPooling2D(pool_size=(3,3),strides=(2,2))(m)
m = inception_module(m, 160, 112, 224, 24, 64, 64)
m = inception_module(m, 128, 128, 256, 24, 64, 64)
m = Conv2D(384, kernel_size=(3,3), padding="same", strides=(1,1),␣
↪activation="relu")(m)
m = Conv2D(256, kernel_size=(3,3), padding="same", strides=(1,1),␣
↪activation="relu")(m)
m = MaxPooling2D(pool_size=(3,3),strides=(2,2))(m)
m = Flatten()(m)
m = Dense(4096,activation="relu")(m)
m = Dense(units=len(classes), activation="softmax", name = "OUTPUT")(m)

model4 = Model(input_layer, m, name='Combination_1')

model4.summary()
plot_model(model4)
```

```
Model: "Combination_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_9 (InputLayer)            [(None, 128, 128, 3) 0
________________________________________________________________________________
__________________
conv2d_88 (Conv2D)              (None, 30, 30, 96)   34944       input_9[0][0]
________________________________________________________________________________
__________________
max_pooling2d_21 (MaxPooling2D) (None, 14, 14, 96)   0           conv2d_88[0][0]
________________________________________________________________________________
__________________
conv2d_89 (Conv2D)              (None, 14, 14, 256)  614656
max_pooling2d_21[0][0]
________________________________________________________________________________
__________________
max_pooling2d_22 (MaxPooling2D) (None, 6, 6, 256)    0           conv2d_89[0][0]
________________________________________________________________________________
__________________
conv2d_91 (Conv2D)              (None, 6, 6, 112)    28784
max_pooling2d_22[0][0]
```

```
__________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 6, 6, 24)     6168
max_pooling2d_22[0][0]
__________________________________________________________________________________
max_pooling2d_23 (MaxPooling2D) (None, 6, 6, 256)    0
max_pooling2d_22[0][0]
__________________________________________________________________________________
conv2d_90 (Conv2D)              (None, 6, 6, 160)    41120
max_pooling2d_22[0][0]
__________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 6, 6, 224)    226016      conv2d_91[0][0]
__________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 6, 6, 64)     38464       conv2d_93[0][0]
__________________________________________________________________________________
conv2d_95 (Conv2D)              (None, 6, 6, 64)     16448
max_pooling2d_23[0][0]
__________________________________________________________________________________
concatenate_12 (Concatenate)    (None, 6, 6, 512)    0           conv2d_90[0][0]
                                                                 conv2d_92[0][0]
                                                                 conv2d_94[0][0]
                                                                 conv2d_95[0][0]
__________________________________________________________________________________
conv2d_97 (Conv2D)              (None, 6, 6, 128)    65664
concatenate_12[0][0]
__________________________________________________________________________________
conv2d_99 (Conv2D)              (None, 6, 6, 24)     12312
concatenate_12[0][0]
__________________________________________________________________________________
max_pooling2d_24 (MaxPooling2D) (None, 6, 6, 512)    0
concatenate_12[0][0]
__________________________________________________________________________________
conv2d_96 (Conv2D)              (None, 6, 6, 128)    65664
concatenate_12[0][0]
__________________________________________________________________________________
conv2d_98 (Conv2D)              (None, 6, 6, 256)    295168      conv2d_97[0][0]
__________________________________________________________________________________
```

```
------------------
conv2d_100 (Conv2D)            (None, 6, 6, 64)     38464       conv2d_99[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_101 (Conv2D)            (None, 6, 6, 64)     32832
max_pooling2d_24[0][0]

--------------------------------------------------------------------------------
------------------
concatenate_13 (Concatenate)   (None, 6, 6, 512)    0           conv2d_96[0][0]
                                                                conv2d_98[0][0]
conv2d_100[0][0]
conv2d_101[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_102 (Conv2D)            (None, 6, 6, 384)    1769856
concatenate_13[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_103 (Conv2D)            (None, 6, 6, 256)    884992
conv2d_102[0][0]

--------------------------------------------------------------------------------
------------------
max_pooling2d_25 (MaxPooling2D) (None, 2, 2, 256)   0
conv2d_103[0][0]

--------------------------------------------------------------------------------
------------------
flatten_3 (Flatten)            (None, 1024)         0
max_pooling2d_25[0][0]

--------------------------------------------------------------------------------
------------------
dense_4 (Dense)                (None, 4096)         4198400     flatten_3[0][0]

--------------------------------------------------------------------------------
------------------
OUTPUT (Dense)                 (None, 8)            32776       dense_4[0][0]
================================================================================
==================
Total params: 8,402,728
Trainable params: 8,402,728
Non-trainable params: 0

--------------------------------------------------------------------------------
------------------
```

[36]:


input_9: InputLayer
conv2d_88: Conv2D
max_pooling2d_21: MaxPooling2D
conv2d_89: Conv2D
max_pooling2d_22: MaxPooling2D
conv2d_91: Conv2D
conv2d_93: Conv2D
max_pooling2d_23: MaxPooling2D
conv2d_92: Conv2D
conv2d_94: Conv2D
conv2d_95: Conv2D
conv2d_90: Conv2D
concatenate_12: Concatenate
conv2d_97: Conv2D
conv2d_99: Conv2D
max_pooling2d_24: MaxPooling2D
conv2d_98: Conv2D
conv2d_100: Conv2D
conv2d_101: Conv2D
conv2d_96: Conv2D
concatenate_13: Concatenate
conv2d_102: Conv2D
conv2d_103: Conv2D
max_pooling2d_25: MaxPooling2D
flatten_3: Flatten
dense_4: Dense
OUTPUT: Dense

[11]:
```
opt = SGD(lr=0.01,momentum=0.9,decay=0.01)
```

[14]:
```
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
early = EarlyStopping(monitor='val_loss', min_delta=0, patience=10,verbose=0,␣
 ↪mode='auto')
```

[40]:
```
model4.
 ↪compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
```

[43]:
```
start_time=t.time()
history4 = model4.fit(train_images, train_labels,
          validation_data=(test_images, test_labels),
          epochs=100, batch_size=64,callbacks=[early])
print("\n Training Time: {} seconds".format(t.time()-start_time))
```

```
Epoch 1/100
157/157 [==============================] - 14s 91ms/step - loss: 2.0319 -
accuracy: 0.1849 - val_loss: 1.9150 - val_accuracy: 0.2515
Epoch 2/100
157/157 [==============================] - 10s 65ms/step - loss: 1.8166 -
accuracy: 0.2834 - val_loss: 2.1114 - val_accuracy: 0.2200
Epoch 3/100
157/157 [==============================] - 11s 69ms/step - loss: 1.6566 -
accuracy: 0.3701 - val_loss: 1.4968 - val_accuracy: 0.4470
Epoch 4/100
157/157 [==============================] - 10s 65ms/step - loss: 1.4466 -
accuracy: 0.4558 - val_loss: 1.3831 - val_accuracy: 0.4925
Epoch 5/100
157/157 [==============================] - 11s 68ms/step - loss: 1.3376 -
accuracy: 0.4998 - val_loss: 1.2881 - val_accuracy: 0.5420
Epoch 6/100
157/157 [==============================] - 11s 68ms/step - loss: 1.2407 -
accuracy: 0.5417 - val_loss: 1.2111 - val_accuracy: 0.5590
Epoch 7/100
157/157 [==============================] - 11s 68ms/step - loss: 1.1865 -
accuracy: 0.5605 - val_loss: 1.1884 - val_accuracy: 0.5660
Epoch 8/100
157/157 [==============================] - 11s 68ms/step - loss: 1.1564 -
accuracy: 0.5701 - val_loss: 1.1737 - val_accuracy: 0.5690
Epoch 9/100
157/157 [==============================] - 11s 71ms/step - loss: 1.1136 -
accuracy: 0.5865 - val_loss: 1.2325 - val_accuracy: 0.5505
Epoch 10/100
157/157 [==============================] - 11s 69ms/step - loss: 1.1058 -
accuracy: 0.5896 - val_loss: 1.1485 - val_accuracy: 0.5940
```

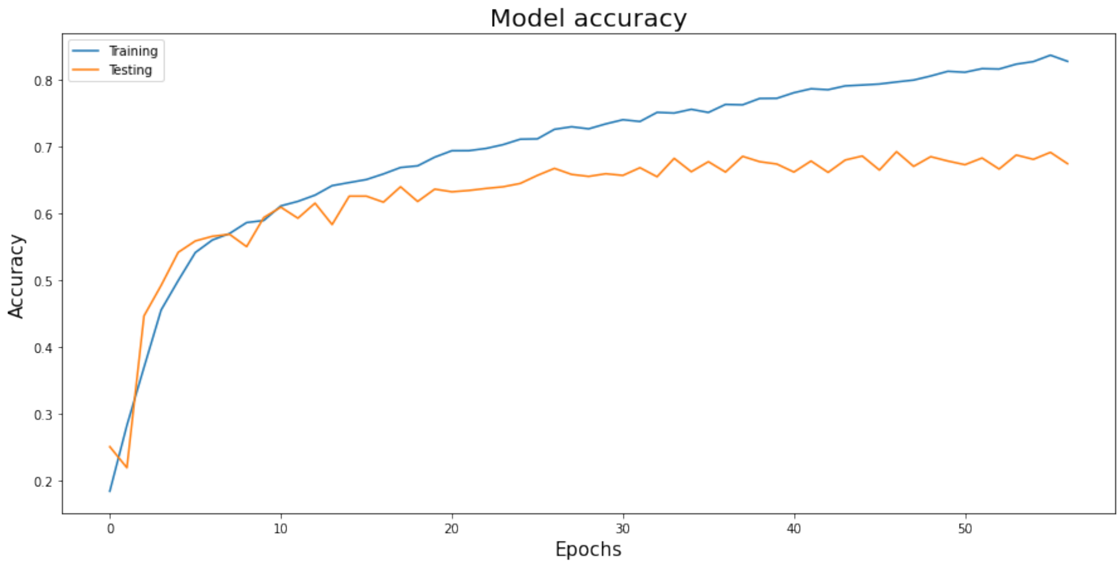
```
Epoch 11/100
157/157 [==============================] - 10s 66ms/step - loss: 1.0583 -
accuracy: 0.6114 - val_loss: 1.0973 - val_accuracy: 0.6095
Epoch 12/100
157/157 [==============================] - 10s 67ms/step - loss: 1.0229 -
accuracy: 0.6182 - val_loss: 1.1073 - val_accuracy: 0.5930
Epoch 13/100
157/157 [==============================] - 11s 71ms/step - loss: 1.0037 -
accuracy: 0.6275 - val_loss: 1.0658 - val_accuracy: 0.6155
Epoch 14/100
157/157 [==============================] - 11s 70ms/step - loss: 0.9649 -
accuracy: 0.6417 - val_loss: 1.1621 - val_accuracy: 0.5835
Epoch 15/100
157/157 [==============================] - 10s 66ms/step - loss: 0.9540 -
accuracy: 0.6463 - val_loss: 1.0625 - val_accuracy: 0.6260
Epoch 16/100
157/157 [==============================] - 11s 69ms/step - loss: 0.9422 -
accuracy: 0.6508 - val_loss: 1.0335 - val_accuracy: 0.6260
Epoch 17/100
157/157 [==============================] - 11s 69ms/step - loss: 0.9237 -
accuracy: 0.6592 - val_loss: 1.0348 - val_accuracy: 0.6170
Epoch 18/100
157/157 [==============================] - 11s 71ms/step - loss: 0.8915 -
accuracy: 0.6688 - val_loss: 1.0017 - val_accuracy: 0.6400
Epoch 19/100
157/157 [==============================] - 10s 66ms/step - loss: 0.8826 -
accuracy: 0.6713 - val_loss: 1.0509 - val_accuracy: 0.6180
Epoch 20/100
157/157 [==============================] - 12s 74ms/step - loss: 0.8593 -
accuracy: 0.6843 - val_loss: 0.9898 - val_accuracy: 0.6365
Epoch 21/100
157/157 [==============================] - 12s 74ms/step - loss: 0.8292 -
accuracy: 0.6939 - val_loss: 1.0013 - val_accuracy: 0.6325
Epoch 22/100
157/157 [==============================] - 11s 70ms/step - loss: 0.8298 -
accuracy: 0.6940 - val_loss: 1.0019 - val_accuracy: 0.6345
Epoch 23/100
157/157 [==============================] - 11s 70ms/step - loss: 0.8115 -
accuracy: 0.6973 - val_loss: 0.9976 - val_accuracy: 0.6375
Epoch 24/100
157/157 [==============================] - 11s 68ms/step - loss: 0.8031 -
accuracy: 0.7030 - val_loss: 0.9980 - val_accuracy: 0.6400
Epoch 25/100
157/157 [==============================] - 11s 72ms/step - loss: 0.7813 -
accuracy: 0.7111 - val_loss: 0.9746 - val_accuracy: 0.6450
Epoch 26/100
157/157 [==============================] - 11s 68ms/step - loss: 0.7662 -
accuracy: 0.7116 - val_loss: 0.9414 - val_accuracy: 0.6570
```

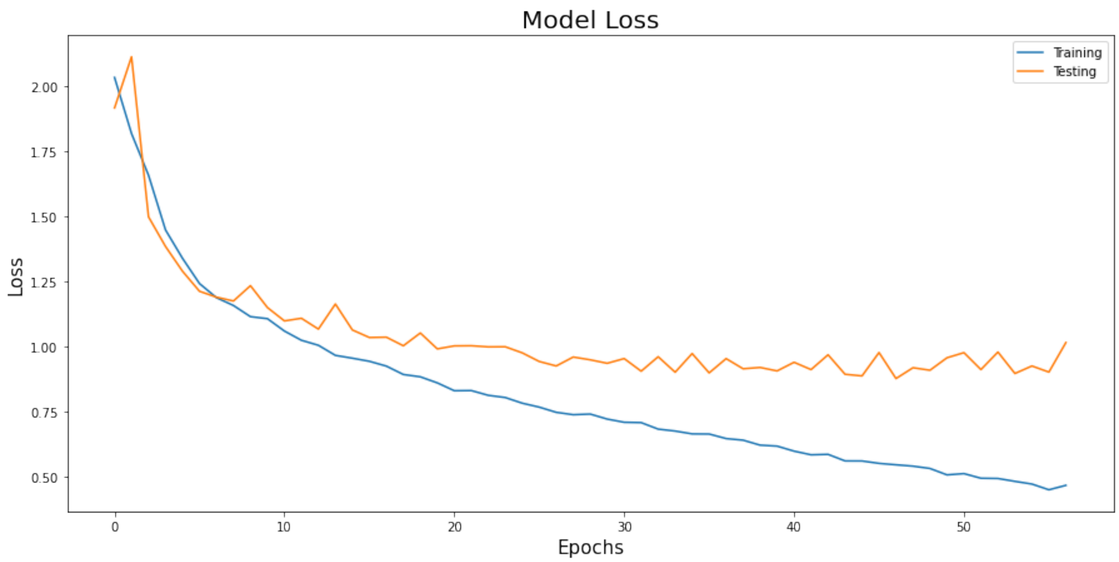
```
Epoch 27/100
157/157 [==============================] - 11s 70ms/step - loss: 0.7462 -
accuracy: 0.7259 - val_loss: 0.9242 - val_accuracy: 0.6675
Epoch 28/100
157/157 [==============================] - 11s 72ms/step - loss: 0.7371 -
accuracy: 0.7297 - val_loss: 0.9585 - val_accuracy: 0.6585
Epoch 29/100
157/157 [==============================] - 12s 75ms/step - loss: 0.7396 -
accuracy: 0.7266 - val_loss: 0.9479 - val_accuracy: 0.6555
Epoch 30/100
157/157 [==============================] - 11s 69ms/step - loss: 0.7203 -
accuracy: 0.7340 - val_loss: 0.9345 - val_accuracy: 0.6595
Epoch 31/100
157/157 [==============================] - 11s 72ms/step - loss: 0.7080 -
accuracy: 0.7402 - val_loss: 0.9526 - val_accuracy: 0.6570
Epoch 32/100
157/157 [==============================] - 12s 74ms/step - loss: 0.7067 -
accuracy: 0.7376 - val_loss: 0.9042 - val_accuracy: 0.6685
Epoch 33/100
157/157 [==============================] - 11s 71ms/step - loss: 0.6818 -
accuracy: 0.7512 - val_loss: 0.9598 - val_accuracy: 0.6550
Epoch 34/100
157/157 [==============================] - 11s 69ms/step - loss: 0.6743 -
accuracy: 0.7502 - val_loss: 0.9002 - val_accuracy: 0.6825
Epoch 35/100
157/157 [==============================] - 11s 71ms/step - loss: 0.6635 -
accuracy: 0.7558 - val_loss: 0.9719 - val_accuracy: 0.6625
Epoch 36/100
157/157 [==============================] - 11s 72ms/step - loss: 0.6629 -
accuracy: 0.7511 - val_loss: 0.8976 - val_accuracy: 0.6775
Epoch 37/100
157/157 [==============================] - 12s 75ms/step - loss: 0.6455 -
accuracy: 0.7631 - val_loss: 0.9525 - val_accuracy: 0.6620
Epoch 38/100
157/157 [==============================] - 11s 73ms/step - loss: 0.6393 -
accuracy: 0.7625 - val_loss: 0.9135 - val_accuracy: 0.6855
Epoch 39/100
157/157 [==============================] - 11s 68ms/step - loss: 0.6202 -
accuracy: 0.7720 - val_loss: 0.9185 - val_accuracy: 0.6775
Epoch 40/100
157/157 [==============================] - 11s 72ms/step - loss: 0.6165 -
accuracy: 0.7722 - val_loss: 0.9052 - val_accuracy: 0.6740
Epoch 41/100
157/157 [==============================] - 12s 74ms/step - loss: 0.5972 -
accuracy: 0.7806 - val_loss: 0.9384 - val_accuracy: 0.6620
Epoch 42/100
157/157 [==============================] - 10s 67ms/step - loss: 0.5832 -
accuracy: 0.7865 - val_loss: 0.9103 - val_accuracy: 0.6785
```

```
Epoch 43/100
157/157 [==============================] - 11s 73ms/step - loss: 0.5849 -
accuracy: 0.7851 - val_loss: 0.9672 - val_accuracy: 0.6615
Epoch 44/100
157/157 [==============================] - 12s 74ms/step - loss: 0.5596 -
accuracy: 0.7908 - val_loss: 0.8924 - val_accuracy: 0.6800
Epoch 45/100
157/157 [==============================] - 11s 71ms/step - loss: 0.5594 -
accuracy: 0.7921 - val_loss: 0.8860 - val_accuracy: 0.6860
Epoch 46/100
157/157 [==============================] - 11s 71ms/step - loss: 0.5502 -
accuracy: 0.7936 - val_loss: 0.9758 - val_accuracy: 0.6650
Epoch 47/100
157/157 [==============================] - 11s 68ms/step - loss: 0.5448 -
accuracy: 0.7967 - val_loss: 0.8759 - val_accuracy: 0.6925
Epoch 48/100
157/157 [==============================] - 11s 68ms/step - loss: 0.5396 -
accuracy: 0.7995 - val_loss: 0.9174 - val_accuracy: 0.6705
Epoch 49/100
157/157 [==============================] - 11s 68ms/step - loss: 0.5306 -
accuracy: 0.8056 - val_loss: 0.9079 - val_accuracy: 0.6850
Epoch 50/100
157/157 [==============================] - 12s 75ms/step - loss: 0.5061 -
accuracy: 0.8126 - val_loss: 0.9557 - val_accuracy: 0.6785
Epoch 51/100
157/157 [==============================] - 12s 74ms/step - loss: 0.5110 -
accuracy: 0.8112 - val_loss: 0.9754 - val_accuracy: 0.6730
Epoch 52/100
157/157 [==============================] - 11s 69ms/step - loss: 0.4931 -
accuracy: 0.8166 - val_loss: 0.9103 - val_accuracy: 0.6830
Epoch 53/100
157/157 [==============================] - 11s 71ms/step - loss: 0.4922 -
accuracy: 0.8160 - val_loss: 0.9778 - val_accuracy: 0.6665
Epoch 54/100
157/157 [==============================] - 11s 71ms/step - loss: 0.4812 -
accuracy: 0.8233 - val_loss: 0.8953 - val_accuracy: 0.6875
Epoch 55/100
157/157 [==============================] - 11s 69ms/step - loss: 0.4709 -
accuracy: 0.8271 - val_loss: 0.9242 - val_accuracy: 0.6810
Epoch 56/100
157/157 [==============================] - 11s 70ms/step - loss: 0.4488 -
accuracy: 0.8367 - val_loss: 0.9007 - val_accuracy: 0.6915
Epoch 57/100
157/157 [==============================] - 11s 69ms/step - loss: 0.4659 -
accuracy: 0.8275 - val_loss: 1.0141 - val_accuracy: 0.6745

 Training Time: 651.438455581665 seconds
```

```
[45]: score4 = model4.evaluate(test_images,test_labels,verbose=0)
print('Test Loss - ',score4[0])
print('Test Accuracy - ',score4[1])
graphs(history4,score4)
```

```
Test Loss -  1.0140718221664429
Test Accuracy -  0.6744999885559082
```






```
Accuracy of the model: 67.45%
```

0.14 ## Combination model 2

- Reduced the overfitting in model 1
- Added another inception model

[15]:
```
#combination 2
input_layer = Input(shape=(n, n, 3))
m = Conv2D(96,kernel_size=(11,11),strides=(4,4),kernel_initializer=kernel_init,␣
↪bias_initializer=bias_init,padding="valid", activation="relu")(input_layer)
m = MaxPooling2D(pool_size=(3,3),strides=(2,2))(m)
m = Conv2D(256, kernel_size=(5,5), padding="same",strides=(1,1),␣
↪activation="relu")(m)
m = MaxPooling2D(pool_size=(3,3),strides=(2,2))(m)
m = inception_module(m, 160, 112, 224, 24, 64, 64)
m = Dropout(0.3)(m)
m = Conv2D(384, kernel_size=(3,3), padding="same", strides=(1,1),␣
↪activation="relu")(m)
m = BatchNormalization()(m)
m = Conv2D(256, kernel_size=(3,3), padding="same", strides=(1,1),␣
↪activation="relu")(m)
m = inception_module(m, 128, 128, 256, 24, 64, 64)
m = Dropout(0.5)(m)
m = inception_module(m, 128, 128, 256, 24, 64, 64)
m = Conv2D(256, kernel_size=(3,3), padding="same", strides=(1,1),␣
↪activation="relu")(m)
m = MaxPooling2D(pool_size=(3,3),strides=(2,2))(m)
m = Flatten()(m)
m = Dense(4096,activation="relu")(m)
m = Dense(4096,activation="relu")(m)
m = Dense(units=len(classes), activation="softmax", name = "OUTPUT")(m)

model5 = Model(input_layer, m, name='Combination_1')

model5.summary()
plot_model(model5)
```

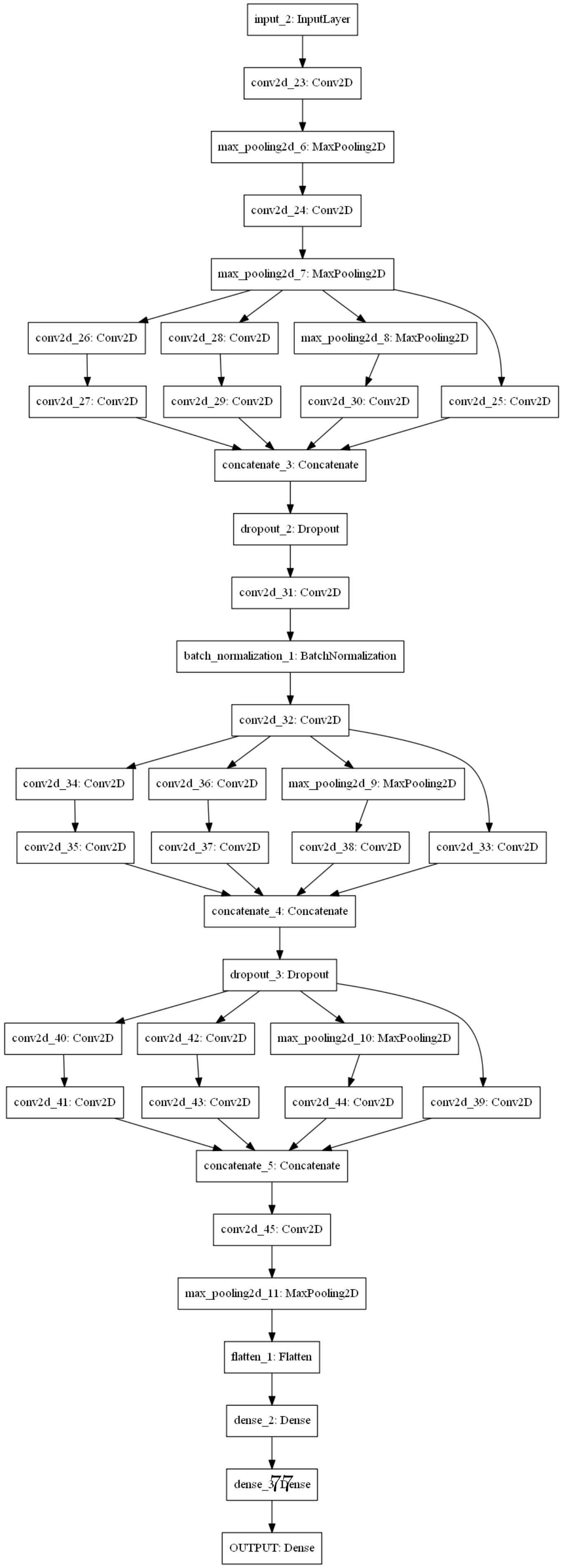
```
Model: "Combination_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_2 (InputLayer)            [(None, 128, 128, 3) 0
________________________________________________________________________________
__________________
conv2d_23 (Conv2D)              (None, 30, 30, 96)   34944       input_2[0][0]
________________________________________________________________________________
__________________
```

```
max_pooling2d_6 (MaxPooling2D)  (None, 14, 14, 96)   0           conv2d_23[0][0]
________________________________________________________________________________
__________________
conv2d_24 (Conv2D)              (None, 14, 14, 256)  614656
max_pooling2d_6[0][0]
________________________________________________________________________________
__________________
max_pooling2d_7 (MaxPooling2D)  (None, 6, 6, 256)    0           conv2d_24[0][0]
________________________________________________________________________________
__________________
conv2d_26 (Conv2D)              (None, 6, 6, 112)    28784
max_pooling2d_7[0][0]
________________________________________________________________________________
__________________
conv2d_28 (Conv2D)              (None, 6, 6, 24)     6168
max_pooling2d_7[0][0]
________________________________________________________________________________
__________________
max_pooling2d_8 (MaxPooling2D)  (None, 6, 6, 256)    0
max_pooling2d_7[0][0]
________________________________________________________________________________
__________________
conv2d_25 (Conv2D)              (None, 6, 6, 160)    41120
max_pooling2d_7[0][0]
________________________________________________________________________________
__________________
conv2d_27 (Conv2D)              (None, 6, 6, 224)    226016      conv2d_26[0][0]
________________________________________________________________________________
__________________
conv2d_29 (Conv2D)              (None, 6, 6, 64)     38464       conv2d_28[0][0]
________________________________________________________________________________
__________________
conv2d_30 (Conv2D)              (None, 6, 6, 64)     16448
max_pooling2d_8[0][0]
________________________________________________________________________________
__________________
concatenate_3 (Concatenate)     (None, 6, 6, 512)    0           conv2d_25[0][0]
                                                                 conv2d_27[0][0]
                                                                 conv2d_29[0][0]
                                                                 conv2d_30[0][0]
________________________________________________________________________________
__________________
dropout_2 (Dropout)             (None, 6, 6, 512)    0
concatenate_3[0][0]
________________________________________________________________________________
__________________
conv2d_31 (Conv2D)              (None, 6, 6, 384)    1769856     dropout_2[0][0]
________________________________________________________________________________
```

```
__________________
batch_normalization_1 (BatchNor (None, 6, 6, 384)    1536        conv2d_31[0][0]
________________________________________________________________________________
__________________
conv2d_32 (Conv2D)              (None, 6, 6, 256)    884992
batch_normalization_1[0][0]
________________________________________________________________________________
__________________
conv2d_34 (Conv2D)              (None, 6, 6, 128)    32896       conv2d_32[0][0]
________________________________________________________________________________
__________________
conv2d_36 (Conv2D)              (None, 6, 6, 24)     6168        conv2d_32[0][0]
________________________________________________________________________________
__________________
max_pooling2d_9 (MaxPooling2D)  (None, 6, 6, 256)    0           conv2d_32[0][0]
________________________________________________________________________________
__________________
conv2d_33 (Conv2D)              (None, 6, 6, 128)    32896       conv2d_32[0][0]
________________________________________________________________________________
__________________
conv2d_35 (Conv2D)              (None, 6, 6, 256)    295168      conv2d_34[0][0]
________________________________________________________________________________
__________________
conv2d_37 (Conv2D)              (None, 6, 6, 64)     38464       conv2d_36[0][0]
________________________________________________________________________________
__________________
conv2d_38 (Conv2D)              (None, 6, 6, 64)     16448
max_pooling2d_9[0][0]
________________________________________________________________________________
__________________
concatenate_4 (Concatenate)     (None, 6, 6, 512)    0           conv2d_33[0][0]
                                                                 conv2d_35[0][0]
                                                                 conv2d_37[0][0]
                                                                 conv2d_38[0][0]
________________________________________________________________________________
__________________
dropout_3 (Dropout)             (None, 6, 6, 512)    0
concatenate_4[0][0]
________________________________________________________________________________
__________________
conv2d_40 (Conv2D)              (None, 6, 6, 128)    65664       dropout_3[0][0]
________________________________________________________________________________
__________________
conv2d_42 (Conv2D)              (None, 6, 6, 24)     12312       dropout_3[0][0]
________________________________________________________________________________
__________________
max_pooling2d_10 (MaxPooling2D) (None, 6, 6, 512)    0           dropout_3[0][0]
________________________________________________________________________________
```

```
------------------
conv2d_39 (Conv2D)              (None, 6, 6, 128)    65664        dropout_3[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_41 (Conv2D)              (None, 6, 6, 256)    295168       conv2d_40[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_43 (Conv2D)              (None, 6, 6, 64)     38464        conv2d_42[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_44 (Conv2D)              (None, 6, 6, 64)     32832
max_pooling2d_10[0][0]

--------------------------------------------------------------------------------
------------------
concatenate_5 (Concatenate)     (None, 6, 6, 512)    0            conv2d_39[0][0]
                                                                  conv2d_41[0][0]
                                                                  conv2d_43[0][0]
                                                                  conv2d_44[0][0]

--------------------------------------------------------------------------------
------------------
conv2d_45 (Conv2D)              (None, 6, 6, 256)    1179904
concatenate_5[0][0]

--------------------------------------------------------------------------------
------------------
max_pooling2d_11 (MaxPooling2D) (None, 2, 2, 256)    0            conv2d_45[0][0]

--------------------------------------------------------------------------------
------------------
flatten_1 (Flatten)             (None, 1024)         0
max_pooling2d_11[0][0]

--------------------------------------------------------------------------------
------------------
dense_2 (Dense)                 (None, 4096)         4198400      flatten_1[0][0]

--------------------------------------------------------------------------------
------------------
dense_3 (Dense)                 (None, 4096)         16781312     dense_2[0][0]

--------------------------------------------------------------------------------
------------------
OUTPUT (Dense)                  (None, 8)            32776        dense_3[0][0]
================================================================================
==================
Total params: 26,787,520
Trainable params: 26,786,752
Non-trainable params: 768

--------------------------------------------------------------------------------
------------------
```

[15]:


input_2: InputLayer
conv2d_23: Conv2D
max_pooling2d_6: MaxPooling2D
conv2d_24: Conv2D
max_pooling2d_7: MaxPooling2D
conv2d_26: Conv2D
conv2d_28: Conv2D
max_pooling2d_8: MaxPooling2D
conv2d_27: Conv2D
conv2d_29: Conv2D
conv2d_30: Conv2D
conv2d_25: Conv2D
concatenate_3: Concatenate
dropout_2: Dropout
conv2d_31: Conv2D
batch_normalization_1: BatchNormalization
conv2d_32: Conv2D
conv2d_34: Conv2D
conv2d_36: Conv2D
max_pooling2d_9: MaxPooling2D
conv2d_35: Conv2D
conv2d_37: Conv2D
conv2d_38: Conv2D
conv2d_33: Conv2D
concatenate_4: Concatenate
dropout_3: Dropout
conv2d_40: Conv2D
conv2d_42: Conv2D
max_pooling2d_10: MaxPooling2D
conv2d_41: Conv2D
conv2d_43: Conv2D
conv2d_44: Conv2D
conv2d_39: Conv2D
concatenate_5: Concatenate
conv2d_45: Conv2D
max_pooling2d_11: MaxPooling2D
flatten_1: Flatten
dense_2: Dense
dense_3: Dense
OUTPUT: Dense

[16]:
```
initial_lrate = 0.01
def decay(epoch, steps=100):
    initial_lrate = 0.01
    drop = 0.96
    epochs_drop = 8
    lrate = initial_lrate * math.pow(drop, math.floor((1+epoch)/epochs_drop))
    return lrate

sgd = SGD(lr=initial_lrate, momentum=0.9, nesterov=False)

lr_sc = LearningRateScheduler(decay, verbose=1)

model5.compile(loss='categorical_crossentropy', optimizer=sgd,␣
 ↪metrics=['accuracy'])
```

[18]:
```
start_time=t.time()
history5 = model5.fit(train_images, train_labels,
          validation_data=(test_images, test_labels),
          epochs=100, batch_size=64,callbacks=[early,lr_sc])
print("\n Training Time: {} seconds".format(t.time()-start_time))
```

```
Epoch 00001: LearningRateScheduler reducing learning rate to 0.01.
Epoch 1/100
  2/157 […] - ETA: 5s - loss: 2.2022 - accuracy:
0.1562WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0270s vs `on_train_batch_end` time: 0.0409s).
Check your callbacks.
157/157 [==============================] - 12s 79ms/step - loss: 2.0067 -
accuracy: 0.1898 - val_loss: 2.0169 - val_accuracy: 0.2345

Epoch 00002: LearningRateScheduler reducing learning rate to 0.01.
Epoch 2/100
157/157 [==============================] - 11s 70ms/step - loss: 1.7394 -
accuracy: 0.3131 - val_loss: 1.7675 - val_accuracy: 0.2940

Epoch 00003: LearningRateScheduler reducing learning rate to 0.01.
Epoch 3/100
157/157 [==============================] - 11s 70ms/step - loss: 1.4978 -
accuracy: 0.4201 - val_loss: 1.6466 - val_accuracy: 0.3695

Epoch 00004: LearningRateScheduler reducing learning rate to 0.01.
Epoch 4/100
157/157 [==============================] - 11s 70ms/step - loss: 1.3442 -
accuracy: 0.4932 - val_loss: 1.4556 - val_accuracy: 0.4570
```

```
Epoch 00005: LearningRateScheduler reducing learning rate to 0.01.
Epoch 5/100
157/157 [==============================] - 11s 71ms/step - loss: 1.2883 -
accuracy: 0.5114 - val_loss: 1.5681 - val_accuracy: 0.4350

Epoch 00006: LearningRateScheduler reducing learning rate to 0.01.
Epoch 6/100
157/157 [==============================] - 11s 71ms/step - loss: 1.1929 -
accuracy: 0.5514 - val_loss: 1.3907 - val_accuracy: 0.4630

Epoch 00007: LearningRateScheduler reducing learning rate to 0.01.
Epoch 7/100
157/157 [==============================] - 11s 71ms/step - loss: 1.1139 -
accuracy: 0.5863 - val_loss: 2.2350 - val_accuracy: 0.2765

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 8/100
157/157 [==============================] - 11s 71ms/step - loss: 1.0404 -
accuracy: 0.6119 - val_loss: 1.5471 - val_accuracy: 0.4285

Epoch 00009: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 9/100
157/157 [==============================] - 11s 71ms/step - loss: 0.9790 -
accuracy: 0.6355 - val_loss: 1.2087 - val_accuracy: 0.5595

Epoch 00010: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 10/100
157/157 [==============================] - 11s 71ms/step - loss: 0.9430 -
accuracy: 0.6522 - val_loss: 1.6276 - val_accuracy: 0.4505

Epoch 00011: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 11/100
157/157 [==============================] - 11s 71ms/step - loss: 0.8896 -
accuracy: 0.6676 - val_loss: 1.0411 - val_accuracy: 0.6015

Epoch 00012: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 12/100
157/157 [==============================] - 11s 71ms/step - loss: 0.8635 -
accuracy: 0.6810 - val_loss: 1.2039 - val_accuracy: 0.5565

Epoch 00013: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 13/100
157/157 [==============================] - 11s 71ms/step - loss: 0.8047 -
accuracy: 0.6974 - val_loss: 1.6535 - val_accuracy: 0.4650

Epoch 00014: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 14/100
```

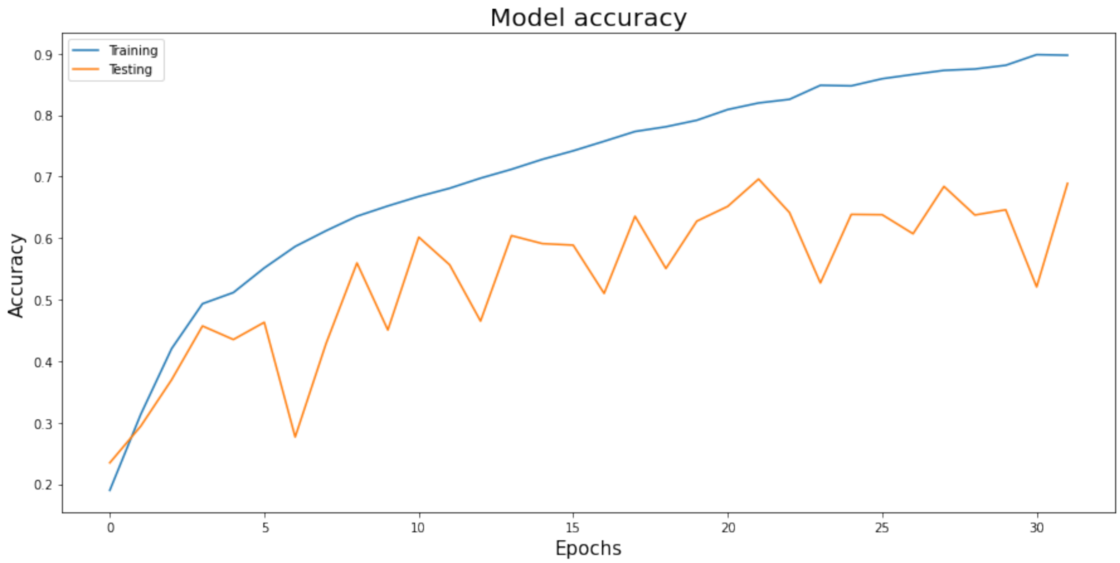
```
157/157 [==============================] - 11s 71ms/step - loss: 0.7526 -
accuracy: 0.7118 - val_loss: 1.1366 - val_accuracy: 0.6040

Epoch 00015: LearningRateScheduler reducing learning rate to 0.0096.
Epoch 15/100
157/157 [==============================] - 11s 71ms/step - loss: 0.7164 -
accuracy: 0.7280 - val_loss: 1.1746 - val_accuracy: 0.5910

Epoch 00016: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 16/100
157/157 [==============================] - 11s 71ms/step - loss: 0.6975 -
accuracy: 0.7419 - val_loss: 1.1037 - val_accuracy: 0.5885

Epoch 00017: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 17/100
157/157 [==============================] - 11s 71ms/step - loss: 0.6417 -
accuracy: 0.7574 - val_loss: 1.3874 - val_accuracy: 0.5100

Epoch 00018: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 18/100
157/157 [==============================] - 11s 71ms/step - loss: 0.6073 -
accuracy: 0.7733 - val_loss: 1.0211 - val_accuracy: 0.6355

Epoch 00019: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 19/100
157/157 [==============================] - 11s 71ms/step - loss: 0.5751 -
accuracy: 0.7810 - val_loss: 1.4679 - val_accuracy: 0.5505

Epoch 00020: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 20/100
157/157 [==============================] - 11s 71ms/step - loss: 0.5441 -
accuracy: 0.7916 - val_loss: 1.1338 - val_accuracy: 0.6275

Epoch 00021: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 21/100
157/157 [==============================] - 11s 71ms/step - loss: 0.5009 -
accuracy: 0.8090 - val_loss: 0.9677 - val_accuracy: 0.6515

Epoch 00022: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 22/100
157/157 [==============================] - 11s 71ms/step - loss: 0.4808 -
accuracy: 0.8197 - val_loss: 0.8817 - val_accuracy: 0.6960

Epoch 00023: LearningRateScheduler reducing learning rate to 0.009216.
Epoch 23/100
157/157 [==============================] - 11s 71ms/step - loss: 0.4576 -
accuracy: 0.8256 - val_loss: 1.1543 - val_accuracy: 0.6415
```

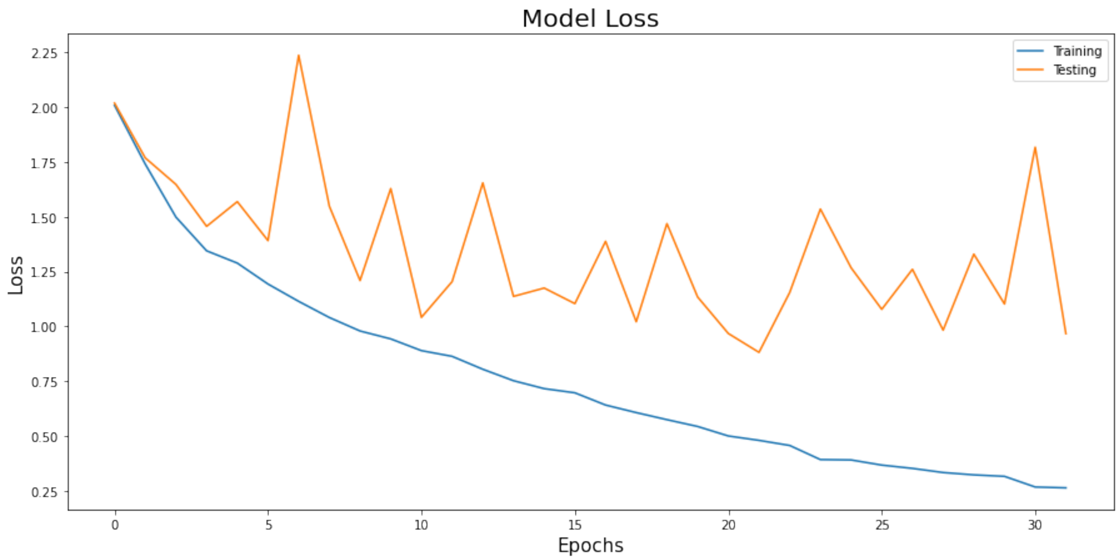
```
Epoch 00024: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 24/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3933 -
accuracy: 0.8484 - val_loss: 1.5344 - val_accuracy: 0.5270

Epoch 00025: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 25/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3917 -
accuracy: 0.8475 - val_loss: 1.2672 - val_accuracy: 0.6385

Epoch 00026: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 26/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3682 -
accuracy: 0.8591 - val_loss: 1.0776 - val_accuracy: 0.6380

Epoch 00027: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 27/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3532 -
accuracy: 0.8661 - val_loss: 1.2599 - val_accuracy: 0.6070

Epoch 00028: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 28/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3343 -
accuracy: 0.8727 - val_loss: 0.9829 - val_accuracy: 0.6840

Epoch 00029: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 29/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3237 -
accuracy: 0.8749 - val_loss: 1.3291 - val_accuracy: 0.6375

Epoch 00030: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 30/100
157/157 [==============================] - 11s 71ms/step - loss: 0.3169 -
accuracy: 0.8810 - val_loss: 1.1018 - val_accuracy: 0.6460

Epoch 00031: LearningRateScheduler reducing learning rate to
0.008847359999999999.
Epoch 31/100
157/157 [==============================] - 11s 71ms/step - loss: 0.2684 -
accuracy: 0.8983 - val_loss: 1.8160 - val_accuracy: 0.5205
```

```
Epoch 00032: LearningRateScheduler reducing learning rate to
0.008493465599999998.
Epoch 32/100
157/157 [==============================] - 11s 71ms/step - loss: 0.2646 -
accuracy: 0.8974 - val_loss: 0.9670 - val_accuracy: 0.6890

 Training Time: 366.57504773139954 seconds
```

[19]:
```
score5 = model5.evaluate(test_images,test_labels,verbose=0)
print('Test Loss - ',score5[0])
print('Test Accuracy - ',score5[1])
graphs(history5,score5)
```

```
Test Loss -  0.9669646620750427
Test Accuracy -  0.6890000104904175
```





Accuracy of the model: 68.90%

0.15 Results Discussion

Our Imagenet dataset conists of 8 classes with 1500 colored images in each class. For this dataset with the input size chosen as 128 x 128 with 3 channels the following results are obtained when trained until convergence.

Observations

- **VGG-16** - It is a complex deep CNN archietecture with 16 weight layers and 3 fully connected layers. The model has a total of 64.5 million trainable parameters. The model produces an accuracy of **60.50 %** over the test data.

- **VGG-19** - It is a variant of model 1 with 19 weight layers and 3 fully connected layers. It has a total of 70 million parameters. The model produces an accuracy of **59.64 %** over the test data.
- **AlexNet** - This model is comparatively shallow than the other 2 models with only 5 weight layers and 3 fully connected layers. The models consists of 24.8 million trainable parameters and produces an accuracy of **68.50 %** over the test data.
- **GoogleNet** - Used inception module, it provided an accuracy of **69.74 %** on the test data. This is the highest accuracy obtained on the test data.
- **ResNet** - Used residual module, it provided an accuracy of **58.85 %** on the test data.
- **Combined Model 1** - It is a combination of AlexNet and Inception module. The model is shallow with two inception module. The test accuracy was **67.55 %** the model was slightly overfitting, which was observed from the loss.
- **Combined Model 2** - It is a variation of combined model 1. Few regularizing parameters were added along with one more inception module, thus making it 3

inception mudules. The test accuracy was **69.55 %**, which was the second best accuracy observed on the test data.

Key Inference

- We can see that the more shallow **AlexNet model** has performed very well over the dataset, there are many reasons that attribute to this. It can be due to the fact that the images are considered at a **lowered pixel ratio**, in which some of the valuable features may be lost.
- We observe **GoogleNet** performed very well on the data. This is mainly **attributed to the inception modules.**
- The **Combined model using the inception model**, seemingly provided better results which can clearly supports the above stated claim.
- The **relatively less performance of the VGG-19 model** when compared with VGG-16 can be **attributed to the depth of the CNN archieture.** The input parameters are very less when considered with the actual depth of the model, as increased number of layers with less features to catch can reduce the activation of neurons with certain features and due to the implicit regularizatons because of the depth of the model.
- Since the data set has only 1500 images for consideration and some of these images were seen to having **obstructions such as human interventions**, the model may be restricted to learn only certain features of the data, if trained on more data which is clean, the performance of the model can be considerably increased.
- Another potential factor that could be affecting the for model's performance is **cropping the image**. Although we crop the image from the lowest side in order to capture the maximum features out of it, in some cases the part of the image that is discarded may contain much more feature information than expected, which can be addresed by increasing the dataset size. By increasing the size of the data, these missed features may not affect the models performance as the probability that all the cropped images miss the same features is very less. #### Room for Improvements
- The current model only uses 1500 images per class to train. Increasing the datasets size would increase the accuracy to a very good extent.
- Increasing the input image size. Current Model uses 128 x 128 images, increasing the size would give more additional information to extract, thus increasing the performance.
- Using data augumentaion, can also improve accuracy of the model.