Report

Team Members: Manar Mohammad Saad, Yomna Muhammad Eskander

Performance Results

CNN

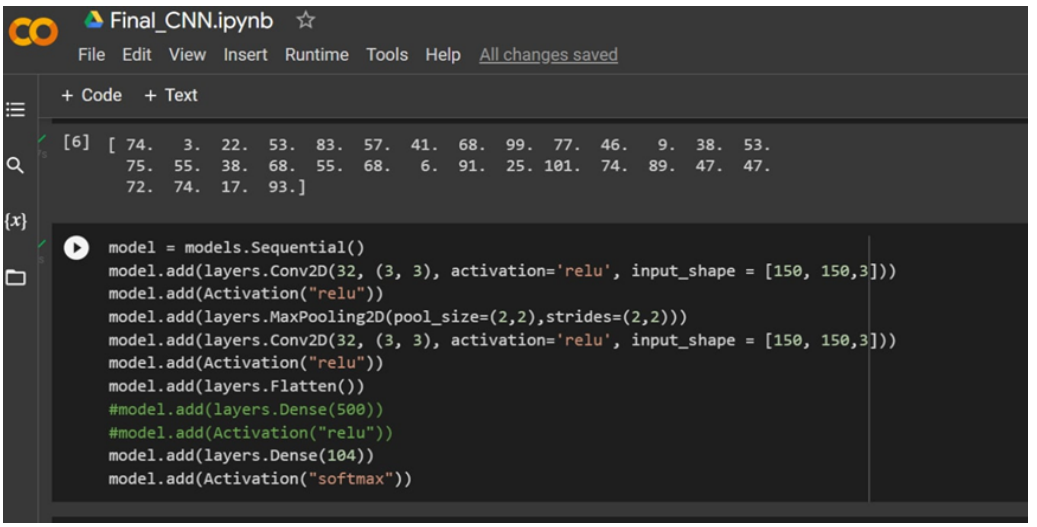
First Case: 10 Epochs with all layers with first architecture (as we have changed the architectures later on).

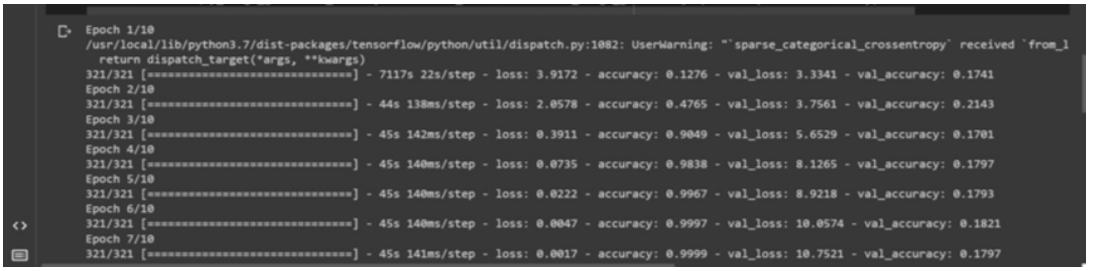
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape = [150, 150,3]))
model.add(Activation("relu"))
model.add(layers.MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape = [150, 150,3]))
model.add(Activation("relu"))
model.add(layers.Flatten())
model.add(layers.Dense(500))
model.add(Activation("relu"))
model.add(layers.Dense(104))
model.add(Activation("softmax"))
```

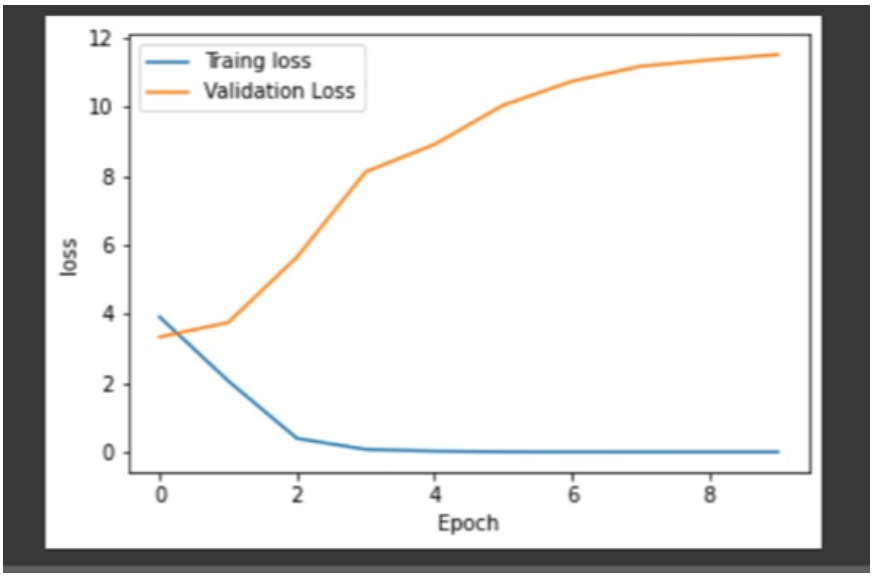
```
Epoch 1/10
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_l
  return dispatch_target(*args, **kwargs)
321/321 [==============================] - 2933s 9s/step - loss: 3.8295 - accuracy: 0.1420 - val_loss: 3.4301 - val_accuracy: 0.1606
Epoch 2/10
321/321 [==============================] - 577s 2s/step - loss: 2.8636 - accuracy: 0.2771 - val_loss: 3.0911 - val_accuracy: 0.2195
Epoch 3/10
321/321 [==============================] - 574s 2s/step - loss: 1.2870 - accuracy: 0.6593 - val_loss: 3.8746 - val_accuracy: 0.1996
Epoch 4/10
321/321 [==============================] - 610s 2s/step - loss: 0.2305 - accuracy: 0.9448 - val_loss: 4.9051 - val_accuracy: 0.2064
Epoch 5/10
321/321 [==============================] - 623s 2s/step - loss: 0.0616 - accuracy: 0.9864 - val_loss: 6.0808 - val_accuracy: 0.1876
Epoch 6/10
321/321 [==============================] - 598s 2s/step - loss: 0.0179 - accuracy: 0.9972 - val_loss: 6.9389 - val_accuracy: 0.1912
Epoch 7/10
321/321 [==============================] - 568s 2s/step - loss: 0.0266 - accuracy: 0.9938 - val_loss: 6.1256 - val_accuracy: 0.2199
Epoch 8/10
```

Results show that the training accuracy has increased rapidly but validation accuracy has decreased which suggest that the model has overfit. We then tried to change this by removing two layers because we thought the model was too much on the data.

Case Two: Removed two layers and put 10 epochs

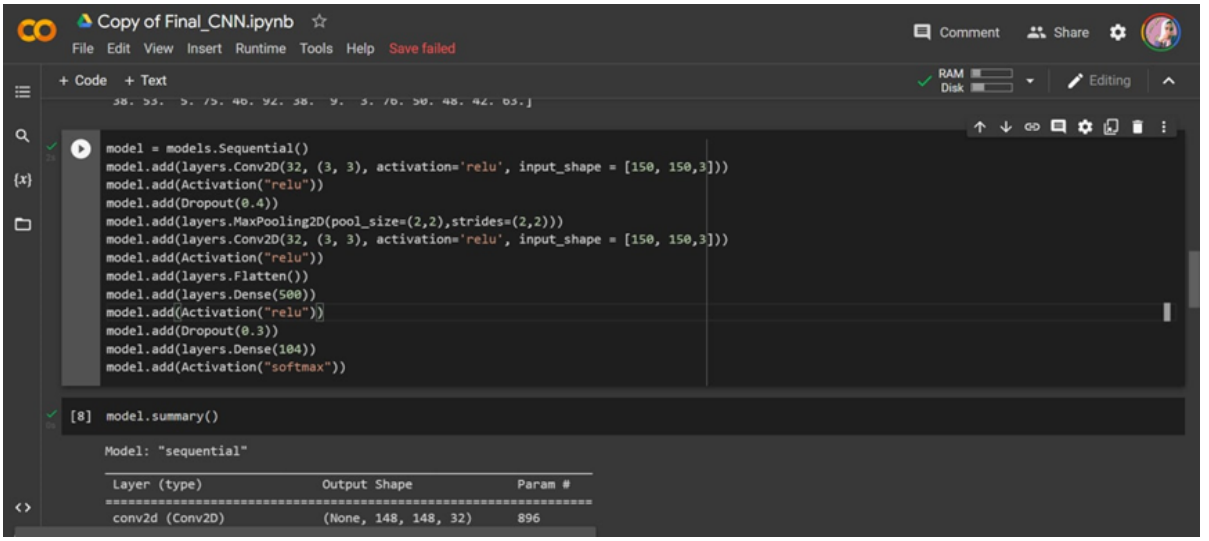


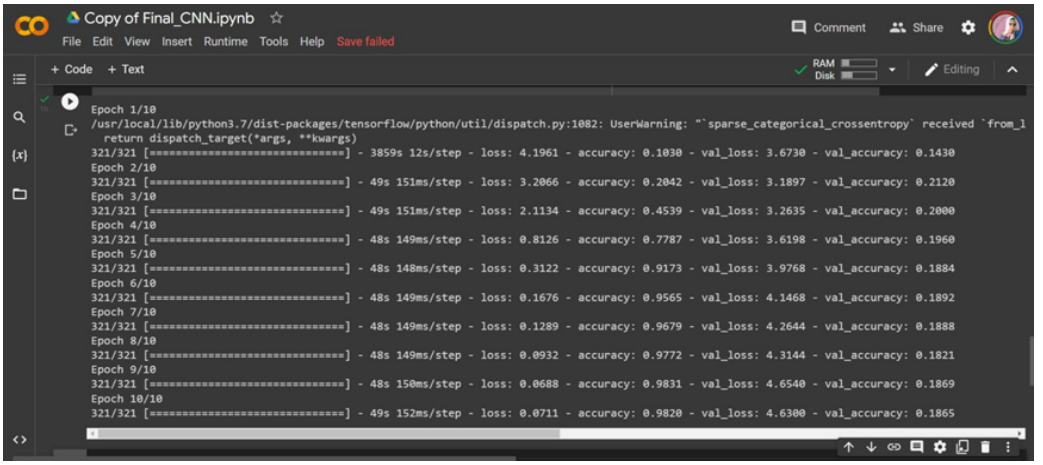




Still the results show that the model is overfitting since the validation loss is way less than the training loss. In this case, we had to change something else. Below we added drop out layers.

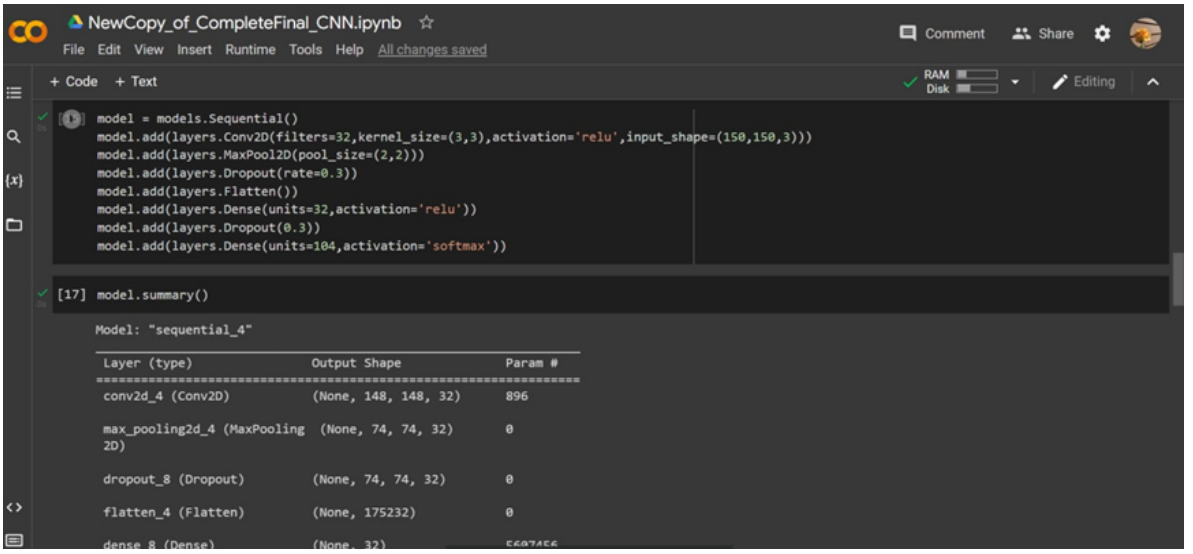
Case Three: Adding drop out layers

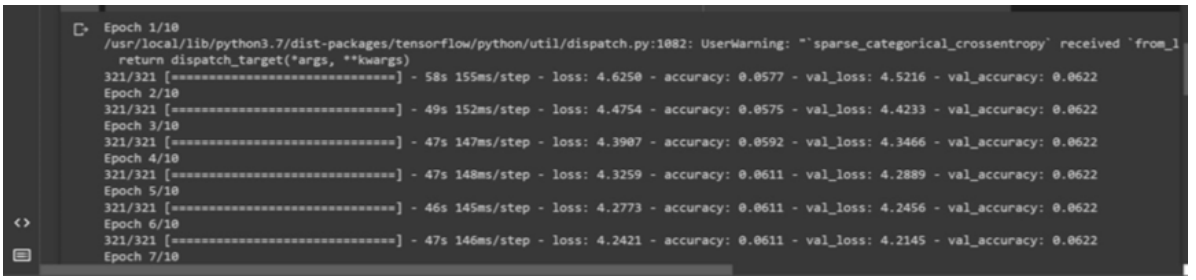


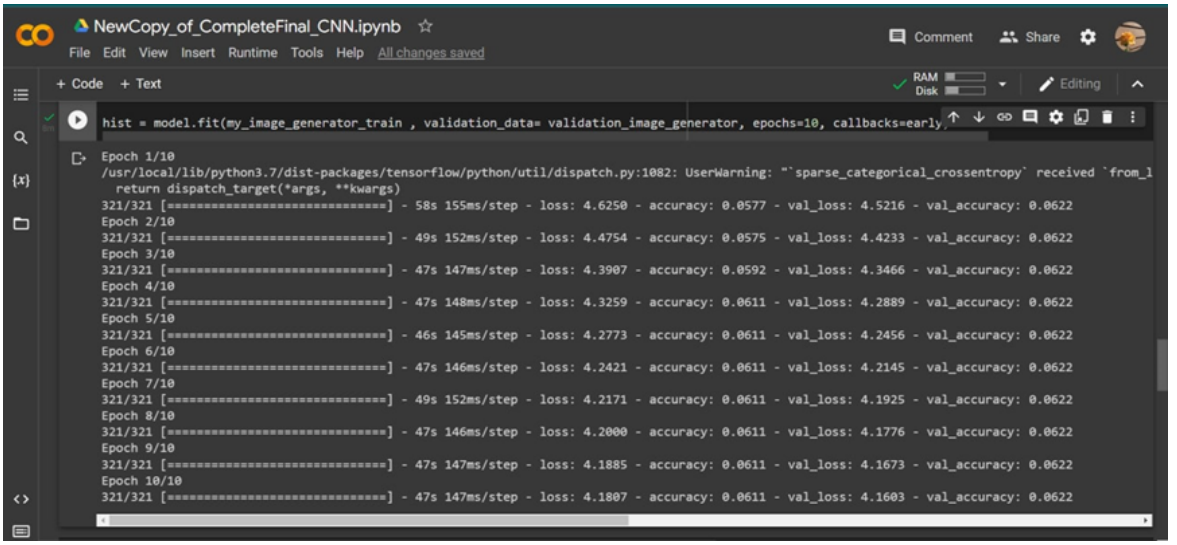


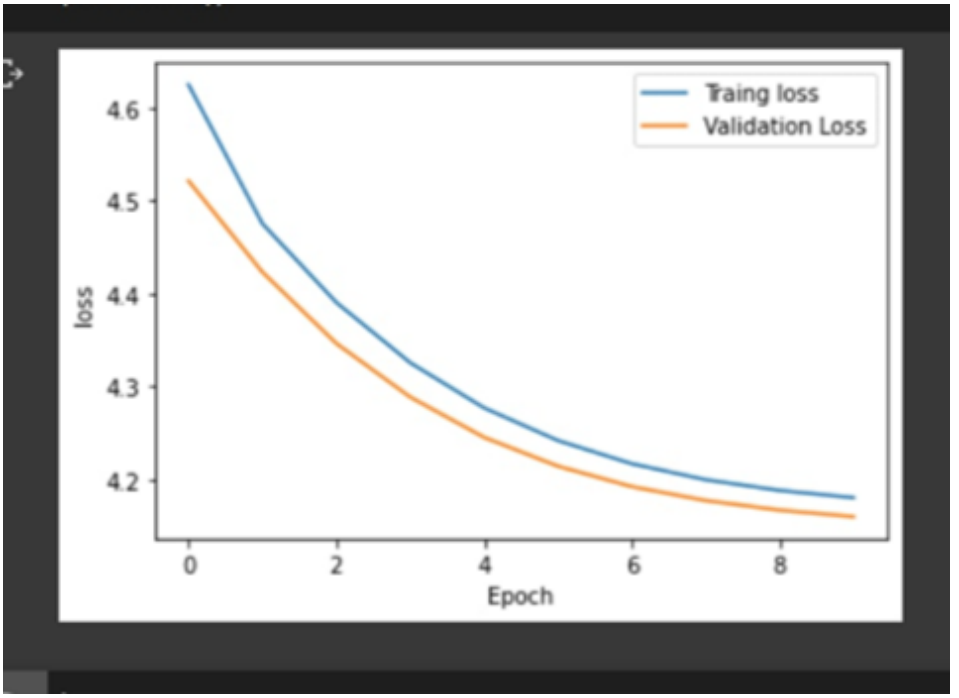
The model is still overfitting even after adding the drop out layers so we thought that the architecture was thee problem and have decided to re-do the layers.

Case Four: We have made a new architecture of CNN
the previous architecture has many layers that caused the overfitting so we have made the new one.


NewCopy_of_CompleteFinal_CNN.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
model = models.Sequential()
model.add(layers.Conv2D(filters=32,kernel_size=(3,3),activation='relu',input_shape=(150,150,3)))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Dropout(rate=0.3))
model.add(layers.Flatten())
model.add(layers.Dense(units=32,activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(units=104,activation='softmax'))
[17] model.summary()
Model: "sequential_4"
Layer (type) Output Shape Param #
conv2d_4 (Conv2D) (None, 148, 148, 32) 896
max_pooling2d_4 (MaxPooling 2D) (None, 74, 74, 32) 0
dropout_8 (Dropout) (None, 74, 74, 32) 0
flatten_4 (Flatten) (None, 175232) 0
dense_8 (Dense) (None, 32) 5607456


Epoch 1/10
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_l
return dispatch_target(*args, **kwargs)
321/321 [==========] - 58s 155ms/step - loss: 4.6250 - accuracy: 0.0577 - val_loss: 4.5216 - val_accuracy: 0.0622
Epoch 2/10
321/321 [==========] - 49s 152ms/step - loss: 4.4754 - accuracy: 0.0575 - val_loss: 4.4233 - val_accuracy: 0.0622
Epoch 3/10
321/321 [==========] - 47s 147ms/step - loss: 4.3907 - accuracy: 0.0592 - val_loss: 4.3466 - val_accuracy: 0.0622
Epoch 4/10
321/321 [==========] - 47s 148ms/step - loss: 4.3259 - accuracy: 0.0611 - val_loss: 4.2889 - val_accuracy: 0.0622
Epoch 5/10
321/321 [==========] - 46s 145ms/step - loss: 4.2773 - accuracy: 0.0611 - val_loss: 4.2456 - val_accuracy: 0.0622
Epoch 6/10
321/321 [==========] - 47s 146ms/step - loss: 4.2421 - accuracy: 0.0611 - val_loss: 4.2145 - val_accuracy: 0.0622
Epoch 7/10





These results finally showed uniform decrease in loss for both the training and the validation data. And the gap between them was getting smaller by every epoch so these were good results.

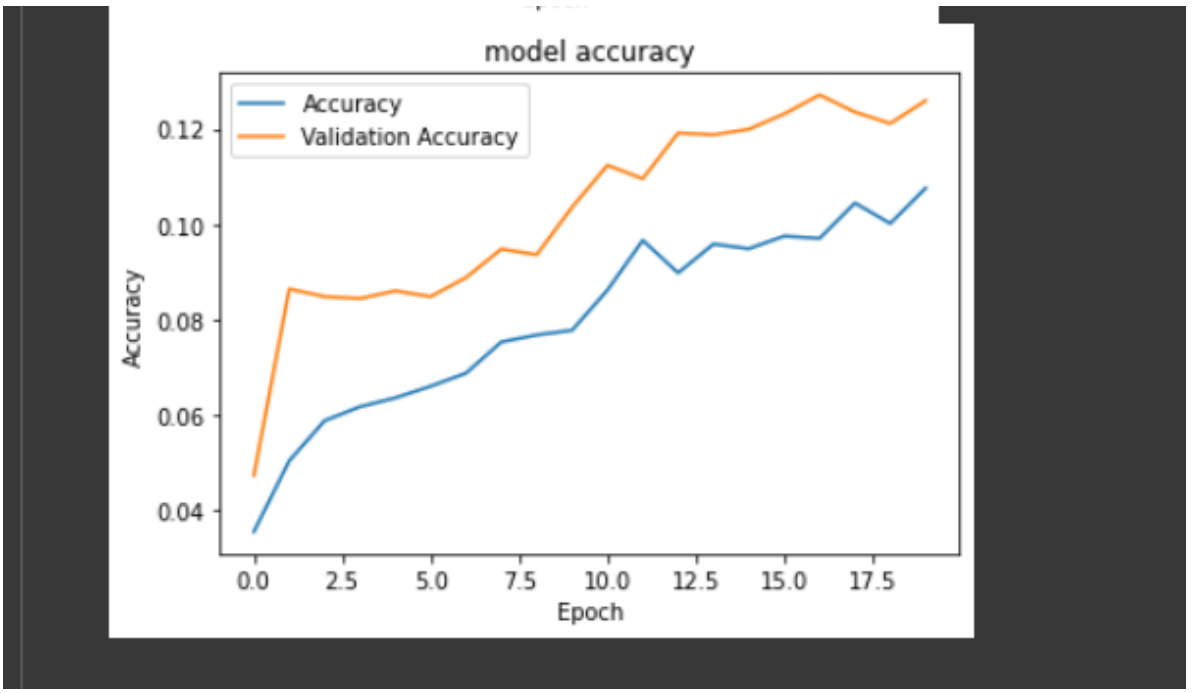
Case Five: we have increased the number of epochs to 20 so that the model can take more time to learn. We have also decreased the learning rate to 0.0001

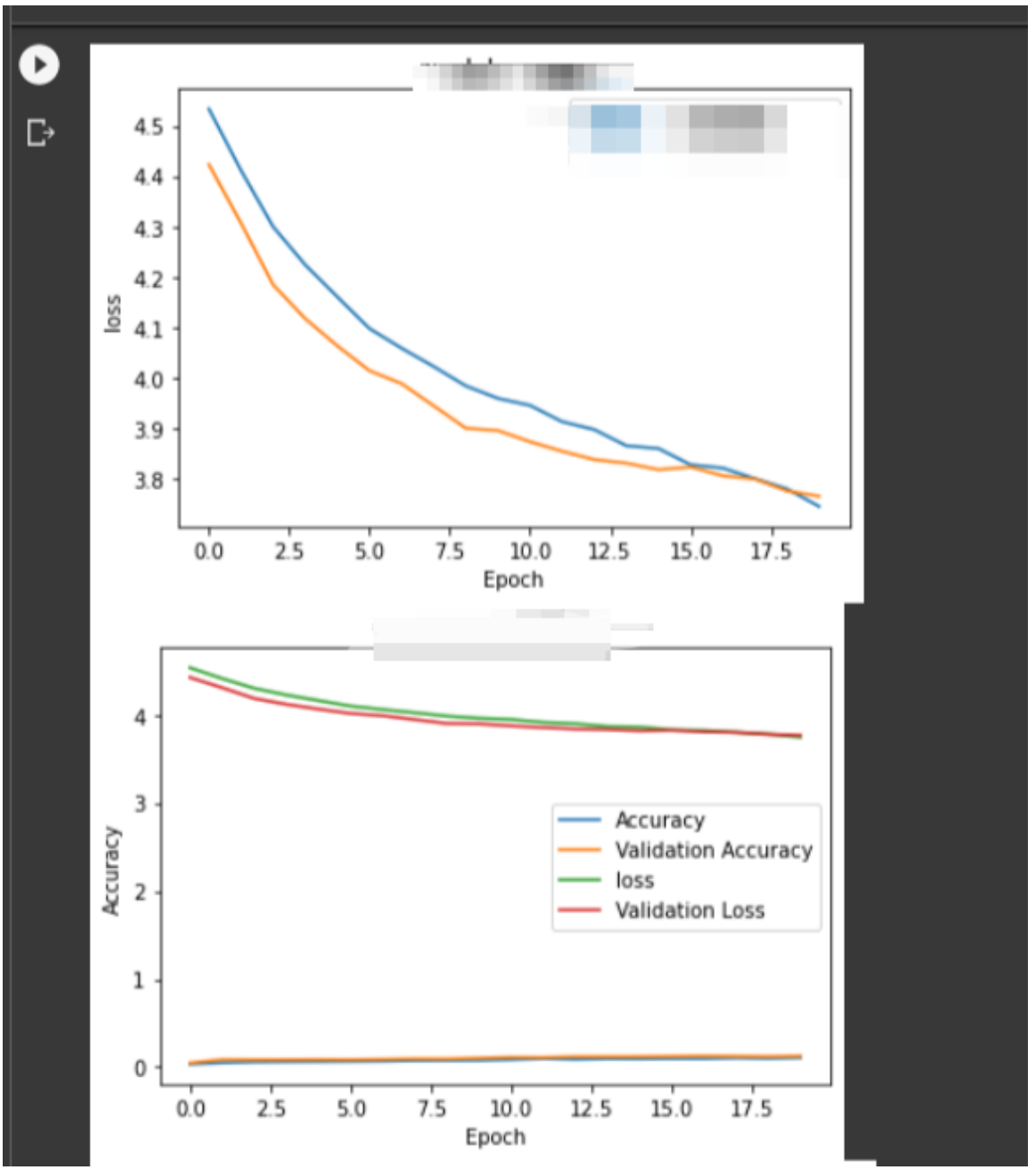
```
opti=Adam(0.0001)
model.compile(optimizer=opti,loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])

#checkpoint = ModelCheckpoint("f/content/drive/MyDrive/CNN_1.h5", monitor='loss', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period
early = EarlyStopping(monitor='loss', min_delta=0, patience=10, verbose=1, mode='auto')

hist = model.fit(my_image_generator_train , validation_data= validation_image_generator, epochs=20, callbacks=early)
```

```
Epoch 1/20
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`, b
  return dispatch_target(*args, **kwargs)
321/321 [==============================] - 1785s 6s/step - loss: 4.5331 - accuracy: 0.0356 - val_loss: 4.4229 - val_accuracy: 0.0474
Epoch 2/20
321/321 [==============================] - 47s 146ms/step - loss: 4.4111 - accuracy: 0.0505 - val_loss: 4.3068 - val_accuracy: 0.0865
Epoch 3/20
321/321 [==============================] - 46s 145ms/step - loss: 4.2996 - accuracy: 0.0589 - val_loss: 4.1847 - val_accuracy: 0.0849
Epoch 4/20
321/321 [==============================] - 46s 145ms/step - loss: 4.2241 - accuracy: 0.0618 - val_loss: 4.1175 - val_accuracy: 0.0845
Epoch 5/20
321/321 [==============================] - 45s 141ms/step - loss: 4.1608 - accuracy: 0.0637 - val_loss: 4.0636 - val_accuracy: 0.0861
Epoch 6/20
321/321 [==============================] - 47s 145ms/step - loss: 4.0979 - accuracy: 0.0661 - val_loss: 4.0144 - val_accuracy: 0.0849
Epoch 7/20
321/321 [==============================] - 46s 143ms/step - loss: 4.0590 - accuracy: 0.0688 - val_loss: 3.9888 - val_accuracy: 0.0888
Epoch 8/20
321/321 [==============================] - 46s 142ms/step - loss: 4.0227 - accuracy: 0.0754 - val_loss: 3.9449 - val_accuracy: 0.0948
Epoch 9/20
321/321 [==============================] - 46s 142ms/step - loss: 3.9843 - accuracy: 0.0768 - val_loss: 3.9005 - val_accuracy: 0.0936
Epoch 10/20
321/321 [==============================] - 45s 142ms/step - loss: 3.9596 - accuracy: 0.0778 - val_loss: 3.8959 - val_accuracy: 0.1036
Epoch 11/20
321/321 [==============================] - 45s 141ms/step - loss: 3.9457 - accuracy: 0.0862 - val_loss: 3.8738 - val_accuracy: 0.1124
Epoch 12/20
321/321 [==============================] - 44s 138ms/step - loss: 3.9134 - accuracy: 0.0967 - val_loss: 3.8548 - val_accuracy: 0.1096
Epoch 13/20
321/321 [==============================] - 45s 142ms/step - loss: 3.8978 - accuracy: 0.0898 - val_loss: 3.8383 - val_accuracy: 0.1191
Epoch 14/20
321/321 [==============================] - 45s 141ms/step - loss: 3.8658 - accuracy: 0.0959 - val_loss: 3.8312 - val_accuracy: 0.1187
Epoch 15/20
```


```
Epoch 11/20
321/321 [==============================] - 45s 141ms/step - loss: 3.9457 - accuracy: 0.0862 - val_loss: 3.8738 - val_accuracy: 0.1124
Epoch 12/20
321/321 [==============================] - 44s 138ms/step - loss: 3.9134 - accuracy: 0.0967 - val_loss: 3.8548 - val_accuracy: 0.1096
Epoch 13/20
321/321 [==============================] - 45s 142ms/step - loss: 3.8978 - accuracy: 0.0898 - val_loss: 3.8383 - val_accuracy: 0.1191
Epoch 14/20
321/321 [==============================] - 45s 141ms/step - loss: 3.8658 - accuracy: 0.0959 - val_loss: 3.8312 - val_accuracy: 0.1187
Epoch 15/20
321/321 [==============================] - 45s 140ms/step - loss: 3.8602 - accuracy: 0.0949 - val_loss: 3.8184 - val_accuracy: 0.1199
Epoch 16/20
321/321 [==============================] - 44s 138ms/step - loss: 3.8281 - accuracy: 0.0975 - val_loss: 3.8235 - val_accuracy: 0.1231
Epoch 17/20
321/321 [==============================] - 45s 142ms/step - loss: 3.8215 - accuracy: 0.0970 - val_loss: 3.8065 - val_accuracy: 0.1271
Epoch 18/20
321/321 [==============================] - 44s 138ms/step - loss: 3.8007 - accuracy: 0.1045 - val_loss: 3.7999 - val_accuracy: 0.1235
Epoch 19/20
321/321 [==============================] - 46s 144ms/step - loss: 3.7804 - accuracy: 0.1002 - val_loss: 3.7763 - val_accuracy: 0.1211
Epoch 20/20
321/321 [==============================] - 46s 142ms/step - loss: 3.7456 - accuracy: 0.1076 - val_loss: 3.7659 - val_accuracy: 0.1259
```


This has increased the accuracy a lot better than the one with less epochs as the graph shows below.


model accuracy
Accuracy
Validation Accuracy
0.12
0.10
0.08
0.06
0.04
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch



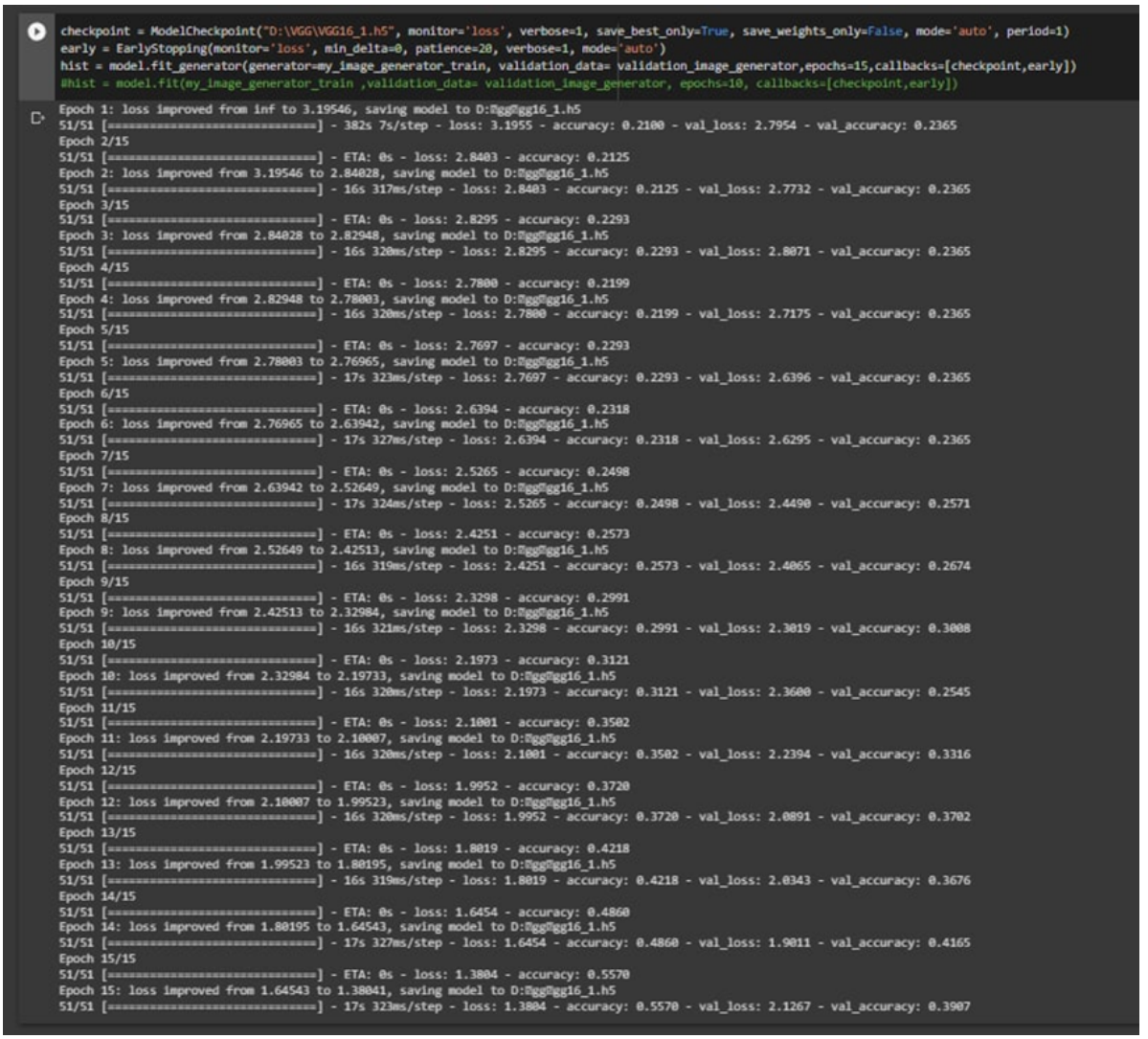
Both accuracies has increased and both losses have decreased. These results show that by more training the model was getting a lot better.

F-score of this model is 0.00462125

VGG

Second Case: When learning rate=0.001 & epochs 15

This case was successful because our model was learning and both losses were decreasing. The accuracy was increasing in both data but it was not uniform increasing as val_accuracy has decreased in epoch 15 after it was increasing.



```
checkpoint = ModelCheckpoint("D:\VGG\VGG16_1.h5", monitor='loss', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='loss', min_delta=0, patience=20, verbose=1, mode='auto')
hist = model.fit_generator(generator=my_image_generator_train, validation_data= validation_image_generator,epochs=15,callbacks=[checkpoint,early])
#hist = model.fit(my_image_generator_train ,validation_data= validation_image_generator, epochs=10, callbacks=[checkpoint,early])

Epoch 1: loss improved from inf to 3.19546, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 382s 7s/step - loss: 3.1955 - accuracy: 0.2100 - val_loss: 2.7954 - val_accuracy: 0.2365
Epoch 2/15
51/51 [==============================] - ETA: 0s - loss: 2.8403 - accuracy: 0.2125
Epoch 2: loss improved from 3.19546 to 2.84028, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 317ms/step - loss: 2.8403 - accuracy: 0.2125 - val_loss: 2.7732 - val_accuracy: 0.2365
Epoch 3/15
51/51 [==============================] - ETA: 0s - loss: 2.8295 - accuracy: 0.2293
Epoch 3: loss improved from 2.84028 to 2.82948, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 320ms/step - loss: 2.8295 - accuracy: 0.2293 - val_loss: 2.8071 - val_accuracy: 0.2365
Epoch 4/15
51/51 [==============================] - ETA: 0s - loss: 2.7800 - accuracy: 0.2199
Epoch 4: loss improved from 2.82948 to 2.78003, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 320ms/step - loss: 2.7800 - accuracy: 0.2199 - val_loss: 2.7175 - val_accuracy: 0.2365
Epoch 5/15
51/51 [==============================] - ETA: 0s - loss: 2.7697 - accuracy: 0.2293
Epoch 5: loss improved from 2.78003 to 2.76965, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 17s 323ms/step - loss: 2.7697 - accuracy: 0.2293 - val_loss: 2.6396 - val_accuracy: 0.2365
Epoch 6/15
51/51 [==============================] - ETA: 0s - loss: 2.6394 - accuracy: 0.2318
Epoch 6: loss improved from 2.76965 to 2.63942, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 17s 327ms/step - loss: 2.6394 - accuracy: 0.2318 - val_loss: 2.6295 - val_accuracy: 0.2365
Epoch 7/15
51/51 [==============================] - ETA: 0s - loss: 2.5265 - accuracy: 0.2498
Epoch 7: loss improved from 2.63942 to 2.52649, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 17s 324ms/step - loss: 2.5265 - accuracy: 0.2498 - val_loss: 2.4490 - val_accuracy: 0.2571
Epoch 8/15
51/51 [==============================] - ETA: 0s - loss: 2.4251 - accuracy: 0.2573
Epoch 8: loss improved from 2.52649 to 2.42513, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 319ms/step - loss: 2.4251 - accuracy: 0.2573 - val_loss: 2.4065 - val_accuracy: 0.2674
Epoch 9/15
51/51 [==============================] - ETA: 0s - loss: 2.3298 - accuracy: 0.2991
Epoch 9: loss improved from 2.42513 to 2.32984, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 321ms/step - loss: 2.3298 - accuracy: 0.2991 - val_loss: 2.3019 - val_accuracy: 0.3008
Epoch 10/15
51/51 [==============================] - ETA: 0s - loss: 2.1973 - accuracy: 0.3121
Epoch 10: loss improved from 2.32984 to 2.19733, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 320ms/step - loss: 2.1973 - accuracy: 0.3121 - val_loss: 2.3600 - val_accuracy: 0.2545
Epoch 11/15
51/51 [==============================] - ETA: 0s - loss: 2.1001 - accuracy: 0.3502
Epoch 11: loss improved from 2.19733 to 2.10007, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 320ms/step - loss: 2.1001 - accuracy: 0.3502 - val_loss: 2.2394 - val_accuracy: 0.3316
Epoch 12/15
51/51 [==============================] - ETA: 0s - loss: 1.9952 - accuracy: 0.3720
Epoch 12: loss improved from 2.10007 to 1.99523, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 320ms/step - loss: 1.9952 - accuracy: 0.3720 - val_loss: 2.0891 - val_accuracy: 0.3702
Epoch 13/15
51/51 [==============================] - ETA: 0s - loss: 1.8019 - accuracy: 0.4218
Epoch 13: loss improved from 1.99523 to 1.80195, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 16s 319ms/step - loss: 1.8019 - accuracy: 0.4218 - val_loss: 2.0343 - val_accuracy: 0.3676
Epoch 14/15
51/51 [==============================] - ETA: 0s - loss: 1.6454 - accuracy: 0.4860
Epoch 14: loss improved from 1.80195 to 1.64543, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 17s 327ms/step - loss: 1.6454 - accuracy: 0.4860 - val_loss: 1.9011 - val_accuracy: 0.4165
Epoch 15/15
51/51 [==============================] - ETA: 0s - loss: 1.3804 - accuracy: 0.5570
Epoch 15: loss improved from 1.64543 to 1.38041, saving model to D:\gg\gg16_1.h5
51/51 [==============================] - 17s 323ms/step - loss: 1.3804 - accuracy: 0.5570 - val_loss: 2.1267 - val_accuracy: 0.3907
```

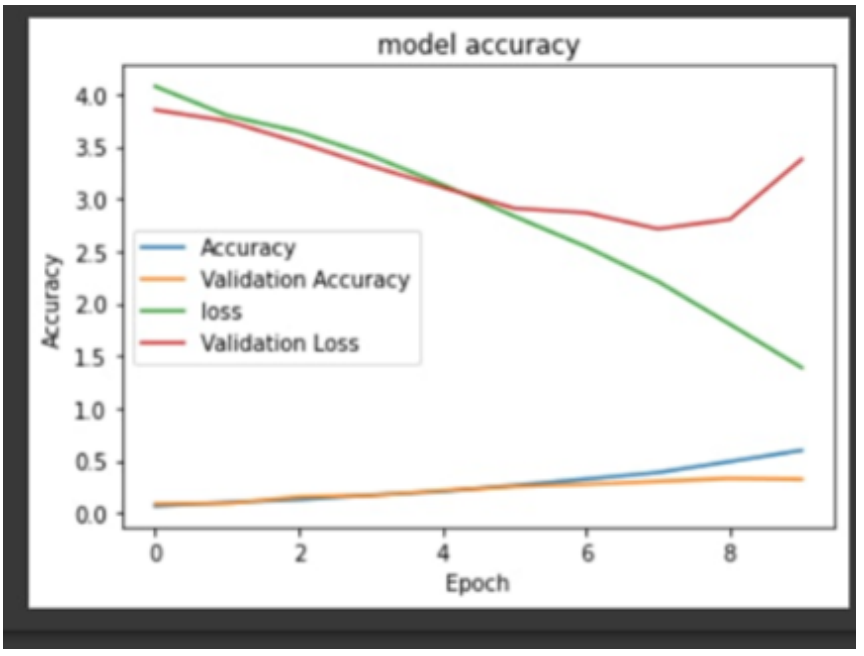
First Case: When learning rate = 0.001 & epochs = 10

The training loss and validation loss is decreasing and the validation accuracy and train accuracy is increasing but it wasn't very good so we increased the epochs in case 2

```
checkpoint = ModelCheckpoint("vgg16_1.h5", monitor='loss', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='loss', min_delta=0, patience=20, verbose=1, mode='auto')
hist = model.fit_generator(generator=my_image_generator_train, validation_data= validation_image_generator,epochs=10,callbacks=[checkpoint,early])
#hist = model.fit(my_image_generator_train ,validation_data= validation_image_generator, epochs=10, callbacks=[checkpoint,early])
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the frequency in number of batches seen.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `M
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/10
321/321 [==============================] - ETA: 0s - loss: 4.0747 - accuracy: 0.0676
Epoch 1: loss improved from inf to 4.07469, saving model to vgg16_1.h5
321/321 [==============================] - 4080s 13s/step - loss: 4.0747 - accuracy: 0.0676 - val_loss: 3.8497 - val_accuracy: 0.0876
Epoch 2/10
321/321 [==============================] - ETA: 0s - loss: 3.7971 - accuracy: 0.1026
Epoch 2: loss improved from 4.07469 to 3.79709, saving model to vgg16_1.h5
321/321 [==============================] - 87s 270ms/step - loss: 3.7971 - accuracy: 0.1026 - val_loss: 3.7438 - val_accuracy: 0.0912
Epoch 3/10
321/321 [==============================] - ETA: 0s - loss: 3.6415 - accuracy: 0.1281
Epoch 3: loss improved from 3.79709 to 3.64146, saving model to vgg16_1.h5
321/321 [==============================] - 88s 274ms/step - loss: 3.6415 - accuracy: 0.1281 - val_loss: 3.5390 - val_accuracy: 0.1546
Epoch 4/10
321/321 [==============================] - ETA: 0s - loss: 3.4144 - accuracy: 0.1726
Epoch 4: loss improved from 3.64146 to 3.41443, saving model to vgg16_1.h5
321/321 [==============================] - 89s 276ms/step - loss: 3.4144 - accuracy: 0.1726 - val_loss: 3.3155 - val_accuracy: 0.1673
Epoch 5/10
```

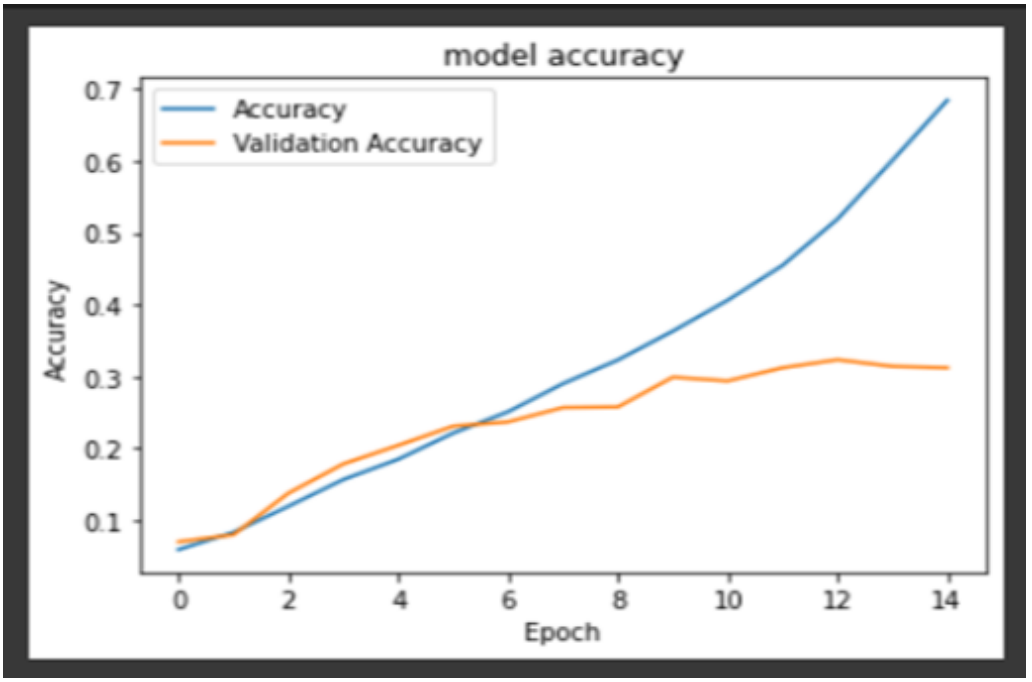
```
321/321 [==============================] - ETA: 0s - loss: 3.4144 - accuracy: 0.1726
Epoch 4: loss improved from 3.64146 to 3.41443, saving model to vgg16_1.h5
321/321 [==============================] - 89s 276ms/step - loss: 3.4144 - accuracy: 0.1726 - val_loss: 3.3155 - val_accuracy: 0.1673
Epoch 5/10
321/321 [==============================] - ETA: 0s - loss: 3.1368 - accuracy: 0.2063
Epoch 5: loss improved from 3.41443 to 3.13684, saving model to vgg16_1.h5
321/321 [==============================] - 88s 273ms/step - loss: 3.1368 - accuracy: 0.2063 - val_loss: 3.1098 - val_accuracy: 0.2155
Epoch 6/10
321/321 [==============================] - ETA: 0s - loss: 2.8332 - accuracy: 0.2655
Epoch 6: loss improved from 3.13684 to 2.83319, saving model to vgg16_1.h5
321/321 [==============================] - 87s 272ms/step - loss: 2.8332 - accuracy: 0.2655 - val_loss: 2.9099 - val_accuracy: 0.2546
Epoch 7/10
321/321 [==============================] - ETA: 0s - loss: 2.5437 - accuracy: 0.3251
Epoch 7: loss improved from 2.83319 to 2.54366, saving model to vgg16_1.h5
321/321 [==============================] - 88s 273ms/step - loss: 2.5437 - accuracy: 0.3251 - val_loss: 2.8670 - val_accuracy: 0.2757
Epoch 8/10
321/321 [==============================] - ETA: 0s - loss: 2.2111 - accuracy: 0.3880
Epoch 8: loss improved from 2.54366 to 2.21108, saving model to vgg16_1.h5
321/321 [==============================] - 88s 273ms/step - loss: 2.2111 - accuracy: 0.3880 - val_loss: 2.7134 - val_accuracy: 0.3032
Epoch 9/10
321/321 [==============================] - ETA: 0s - loss: 1.8022 - accuracy: 0.4922
Epoch 9: loss improved from 2.21108 to 1.80222, saving model to vgg16_1.h5
321/321 [==============================] - 88s 273ms/step - loss: 1.8022 - accuracy: 0.4922 - val_loss: 2.8049 - val_accuracy: 0.3319
Epoch 10/10
321/321 [==============================] - ETA: 0s - loss: 1.3876 - accuracy: 0.5995
Epoch 10: loss improved from 1.80222 to 1.38760, saving model to vgg16_1.h5
321/321 [==============================] - 88s 275ms/step - loss: 1.3876 - accuracy: 0.5995 - val_loss: 3.3801 - val_accuracy: 0.3231
```

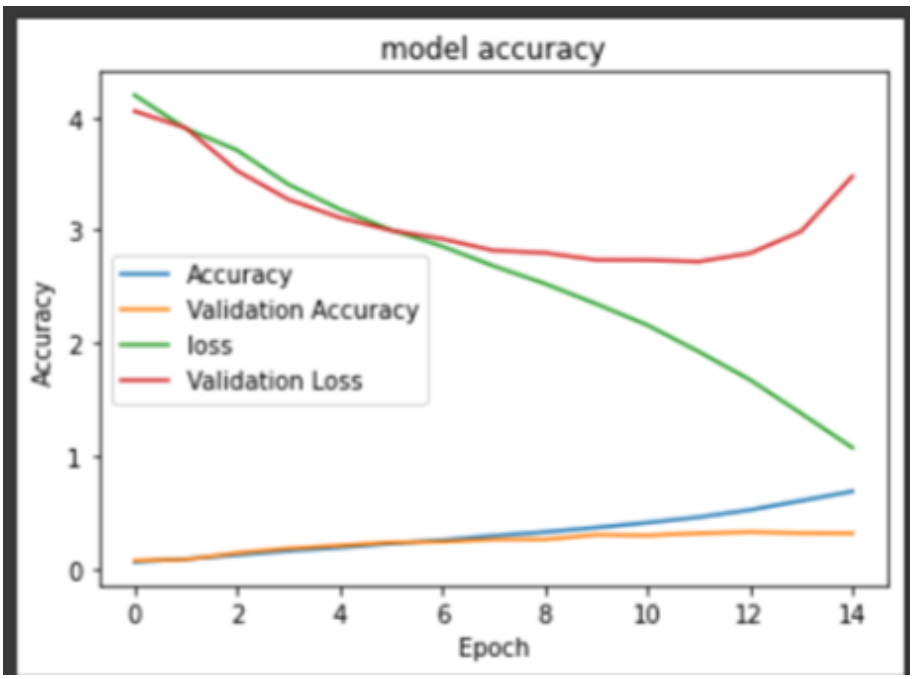


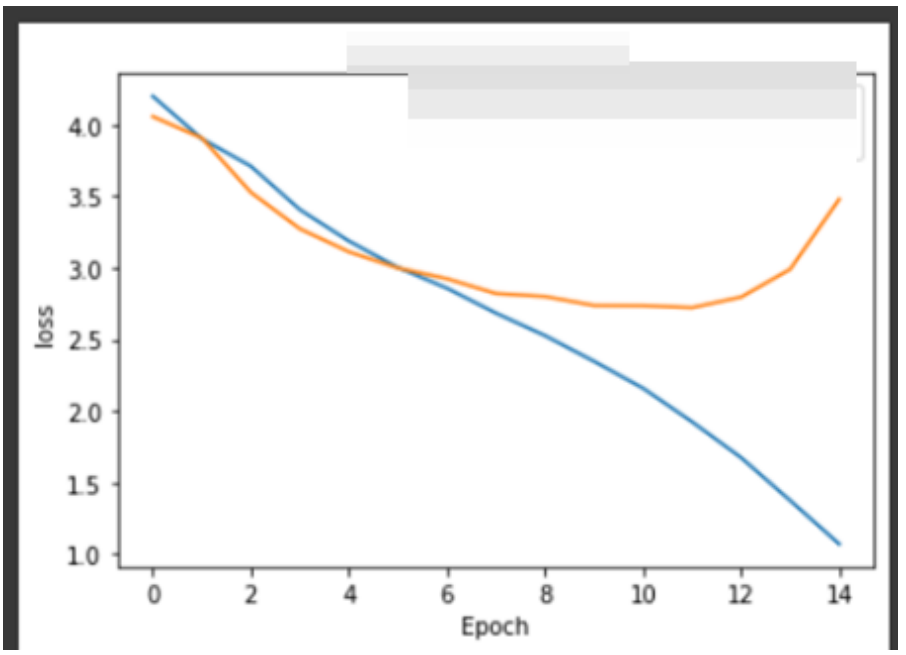
<u>Third Case:</u> When learning rate was = 0.00001 & epochs = 15

```
#checkpoint = ModelCheckpoint("f/content/drive/MyDrive/VGG16_2.h5", monitor='loss', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='loss', min_delta=0, patience=20, verbose=1, mode='auto')
hist = model.fit(my_image_generator_train ,validation_data= validation_image_generator, epochs=15, callbacks=early)
```

```
Epoch 1/15
321/321 [==============================] - 101s 270ms/step - loss: 4.1979 - accuracy: 0.0589 - val_loss: 4.0557 - val_accuracy: 0.0697
Epoch 2/15
321/321 [==============================] - 90s 280ms/step - loss: 3.9008 - accuracy: 0.0830 - val_loss: 3.9055 - val_accuracy: 0.0797
Epoch 3/15
321/321 [==============================] - 90s 281ms/step - loss: 3.7091 - accuracy: 0.1192 - val_loss: 3.5264 - val_accuracy: 0.1375
Epoch 4/15
321/321 [==============================] - 90s 280ms/step - loss: 3.4057 - accuracy: 0.1564 - val_loss: 3.2720 - val_accuracy: 0.1781
Epoch 5/15
321/321 [==============================] - 90s 281ms/step - loss: 3.1863 - accuracy: 0.1846 - val_loss: 3.1112 - val_accuracy: 0.2040
Epoch 6/15
321/321 [==============================] - 90s 281ms/step - loss: 3.0017 - accuracy: 0.2210 - val_loss: 2.9980 - val_accuracy: 0.2307
Epoch 7/15
321/321 [==============================] - 90s 280ms/step - loss: 2.8568 - accuracy: 0.2507 - val_loss: 2.9228 - val_accuracy: 0.2367
Epoch 8/15
321/321 [==============================] - 90s 281ms/step - loss: 2.6827 - accuracy: 0.2899 - val_loss: 2.8196 - val_accuracy: 0.2566
Epoch 9/15
321/321 [==============================] - 93s 289ms/step - loss: 2.5259 - accuracy: 0.3228 - val_loss: 2.7987 - val_accuracy: 0.2574
Epoch 10/15
321/321 [==============================] - 91s 282ms/step - loss: 2.3467 - accuracy: 0.3629 - val_loss: 2.7364 - val_accuracy: 0.2988
Epoch 11/15
321/321 [==============================] - 90s 281ms/step - loss: 2.1582 - accuracy: 0.4059 - val_loss: 2.7351 - val_accuracy: 0.2936
Epoch 12/15
321/321 [==============================] - 90s 280ms/step - loss: 1.9236 - accuracy: 0.4545 - val_loss: 2.7212 - val_accuracy: 0.3120
Epoch 13/15
321/321 [==============================] - 91s 282ms/step - loss: 1.6739 - accuracy: 0.5189 - val_loss: 2.7948 - val_accuracy: 0.3231
Epoch 14/15
321/321 [==============================] - 90s 280ms/step - loss: 1.3745 - accuracy: 0.6006 - val_loss: 2.9893 - val_accuracy: 0.3139
Epoch 15/15
321/321 [==============================] - 90s 279ms/step - loss: 1.0704 - accuracy: 0.6844 - val_loss: 3.4785 - val_accuracy: 0.3120
```







These results show that the model at the very end started overfitting. so we should've terminated the fitting eailer to avoid this.

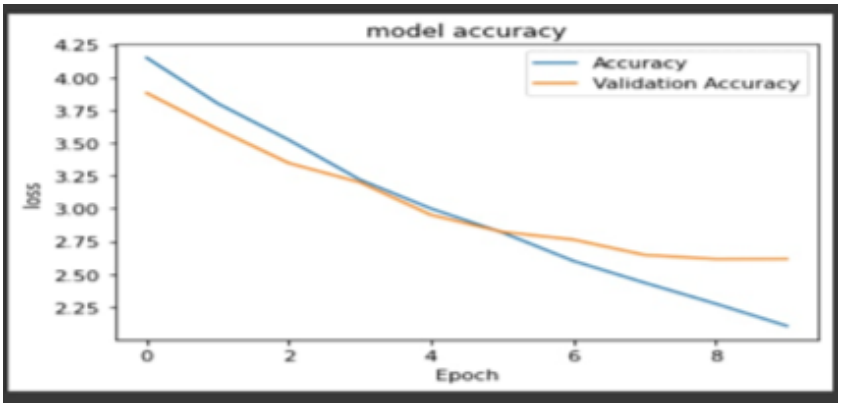
GoogleNet

Case One: lr =0.001 & epochs 10

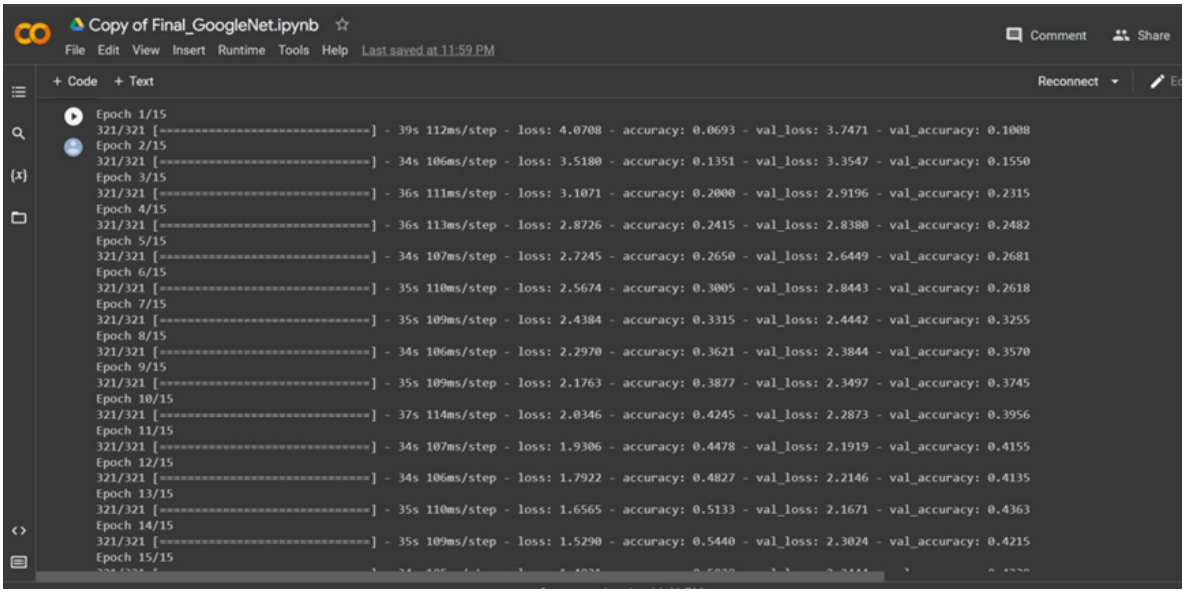
```
from keras.callbacks import ModelCheckpoint, EarlyStopping
#call_backs= ModelCheckpoint("f/content/drive/MyDrive/GoogleNet_1.h5", monitor='val_acc', verbose=1, save_best_only=True, save_weights_onl
#early = EarlyStopping(monitor='val_acc', min_delta=0, patience=20, verbose=1, mode='auto')
hist = model.fit(my_image_generator_train ,validation_data= validation_image_generator, epochs=10)
```

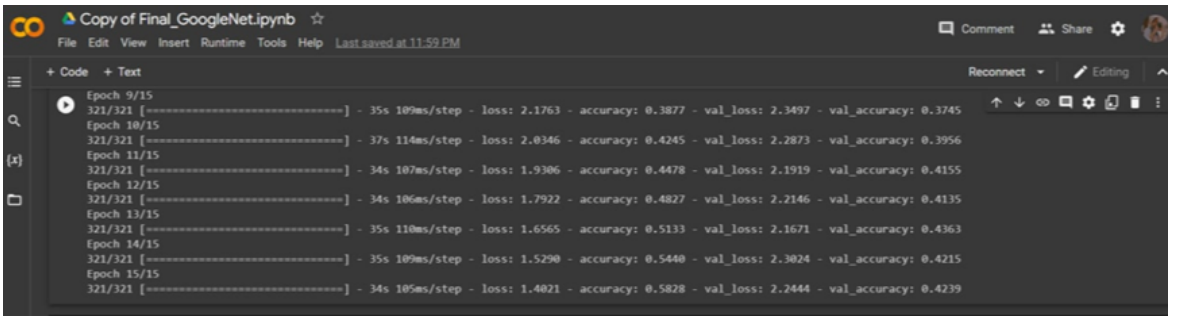
```
Epoch 1/10
321/321 [==============================] - 50s 117ms/step - loss: 4.1493 - accuracy: 0.0616 - val_loss: 3.8807 - val_accuracy: 0.0952
Epoch 2/10
321/321 [==============================] - 34s 105ms/step - loss: 3.8007 - accuracy: 0.1036 - val_loss: 3.6040 - val_accuracy: 0.1291
Epoch 3/10
321/321 [==============================] - 35s 109ms/step - loss: 3.5226 - accuracy: 0.1418 - val_loss: 3.3458 - val_accuracy: 0.1689
Epoch 4/10
321/321 [==============================] - 35s 109ms/step - loss: 3.2193 - accuracy: 0.1903 - val_loss: 3.1993 - val_accuracy: 0.1745
Epoch 5/10
321/321 [==============================] - 34s 106ms/step - loss: 2.9991 - accuracy: 0.2205 - val_loss: 2.9513 - val_accuracy: 0.2227
Epoch 6/10
321/321 [==============================] - 36s 113ms/step - loss: 2.8178 - accuracy: 0.2570 - val_loss: 2.8219 - val_accuracy: 0.2625
Epoch 7/10
321/321 [==============================] - 35s 110ms/step - loss: 2.6005 - accuracy: 0.2952 - val_loss: 2.7631 - val_accuracy: 0.2781
Epoch 8/10
321/321 [==============================] - 34s 104ms/step - loss: 2.4347 - accuracy: 0.3276 - val_loss: 2.6461 - val_accuracy: 0.2992
Epoch 9/10
321/321 [==============================] - 35s 109ms/step - loss: 2.2746 - accuracy: 0.3683 - val_loss: 2.6149 - val_accuracy: 0.3267
Epoch 10/10
321/321 [==============================] - 35s 108ms/step - loss: 2.1053 - accuracy: 0.4019 - val_loss: 2.6159 - val_accuracy: 0.3303
```

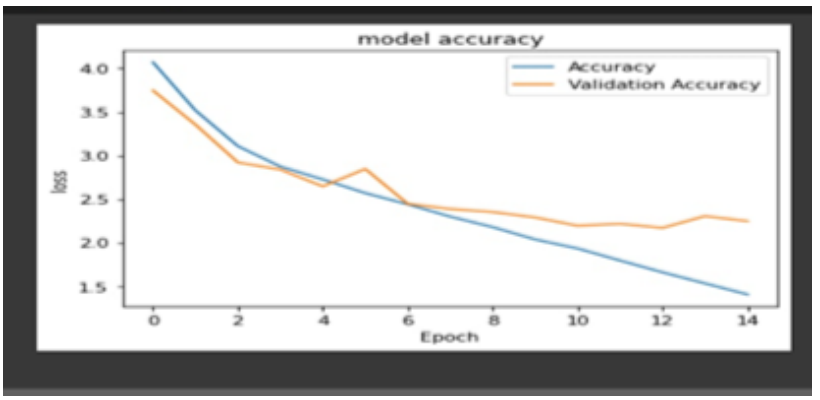
Evaluation



Second Case: lr =0.0001 & epochs 15



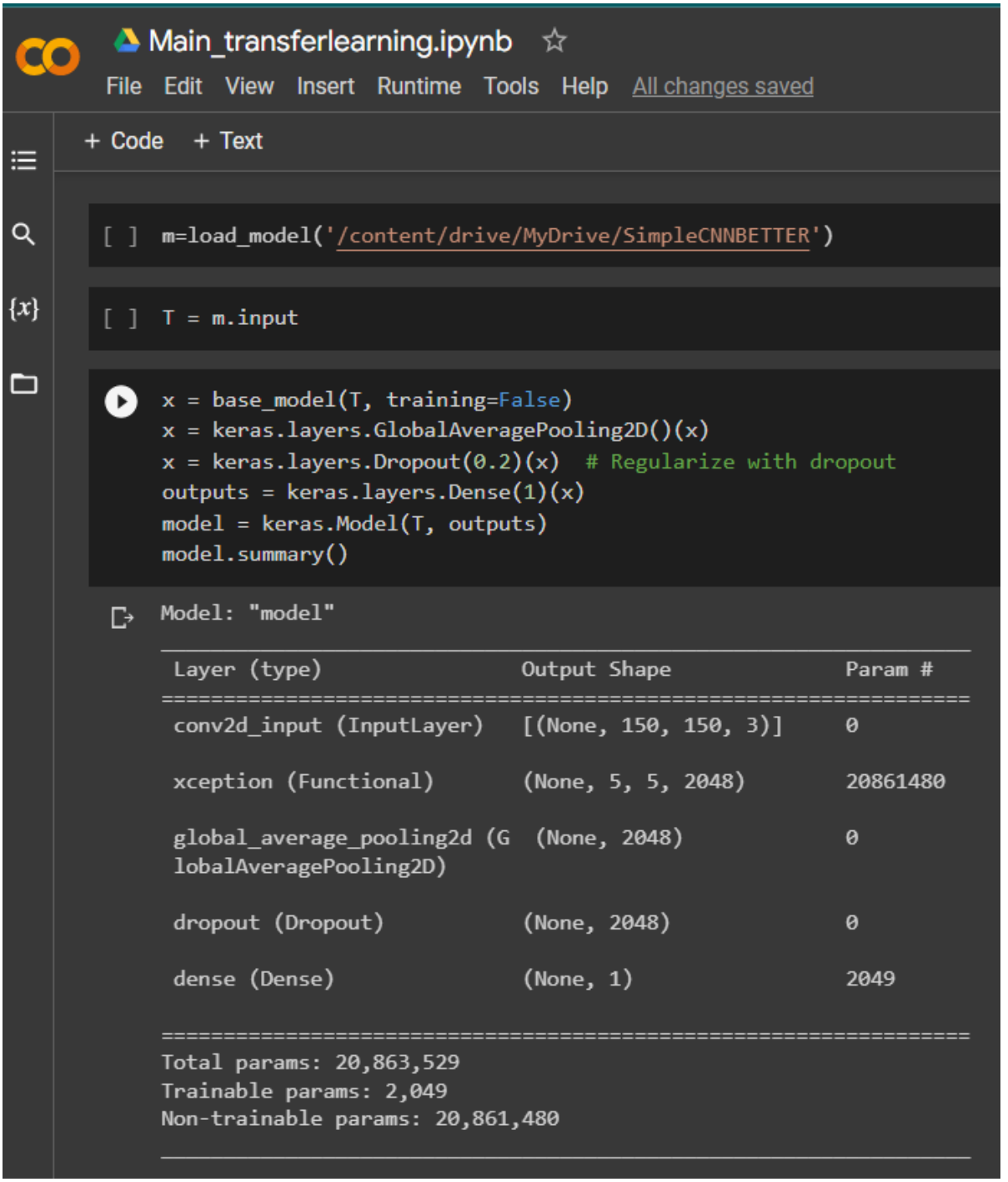




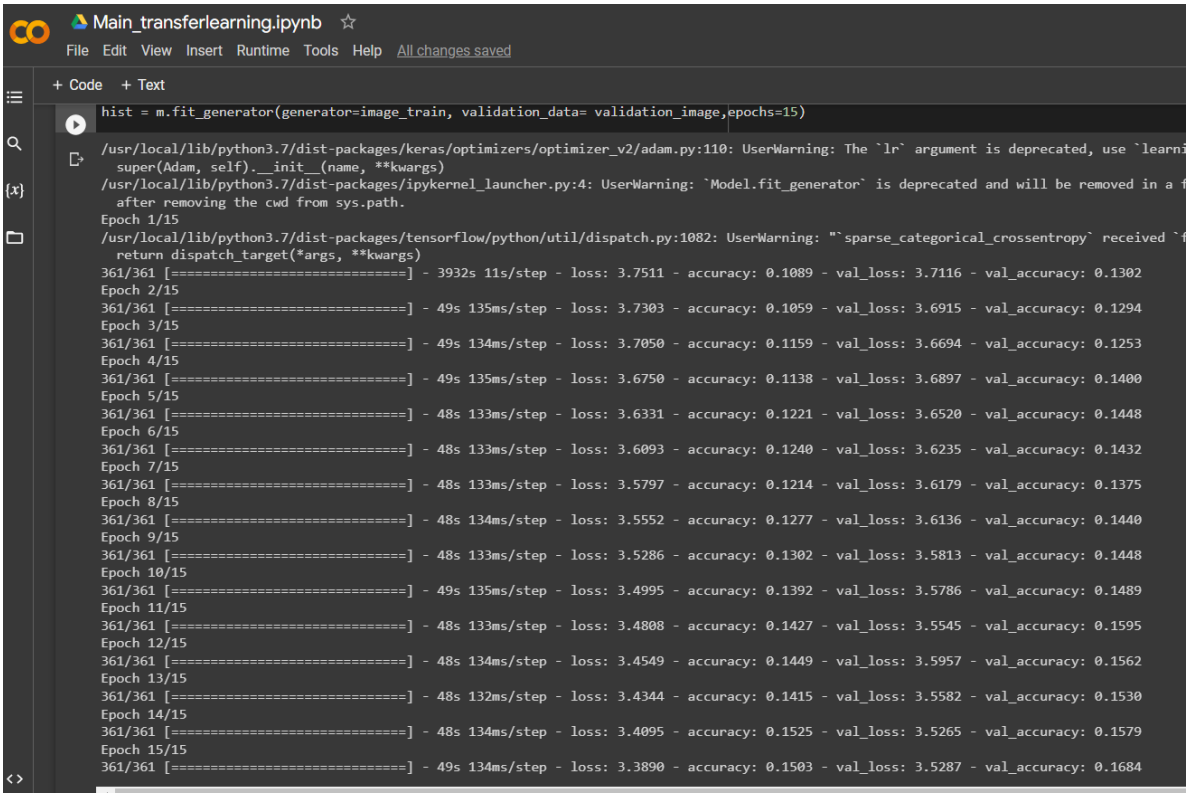
Transfer Learning

Since our data is not large, we used transfer learning to use both the Simple CNN and the Xception model from keras we made to get us better results.

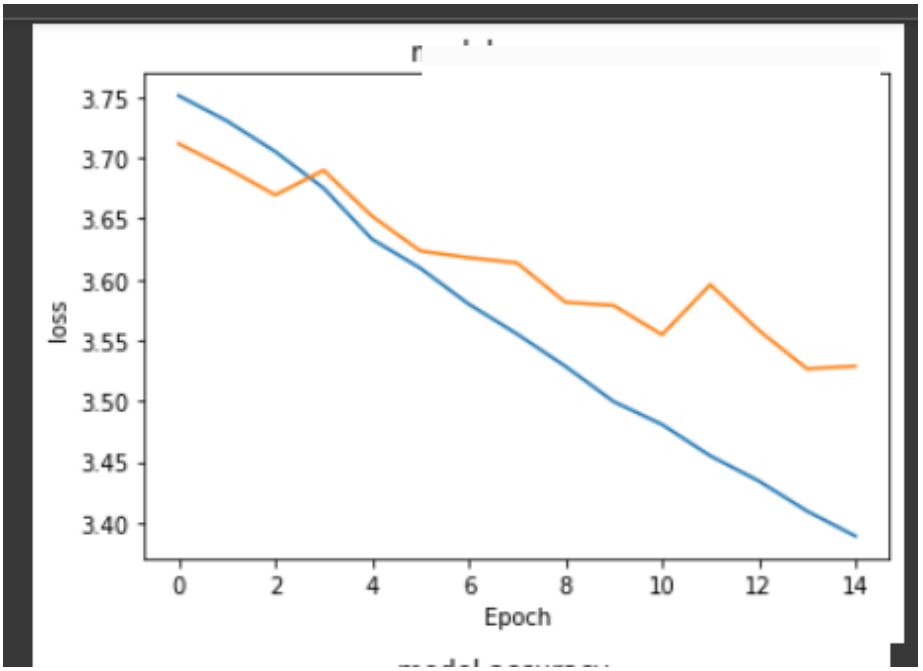
We have loaded our simple Cnn and then using the Xception model from Keras, we put the simple model on top of it.

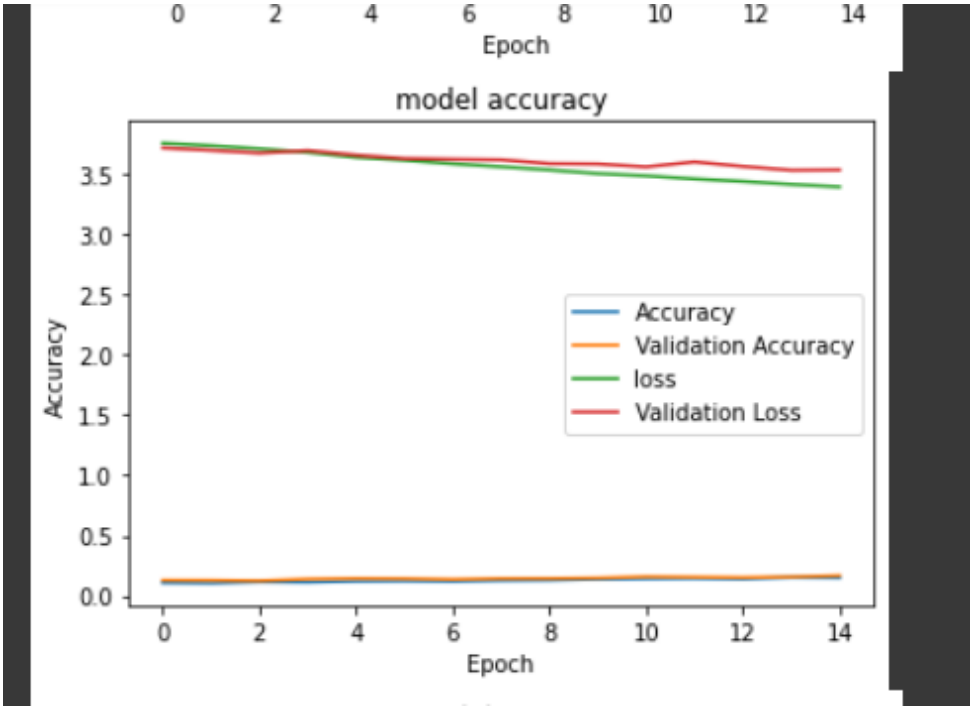


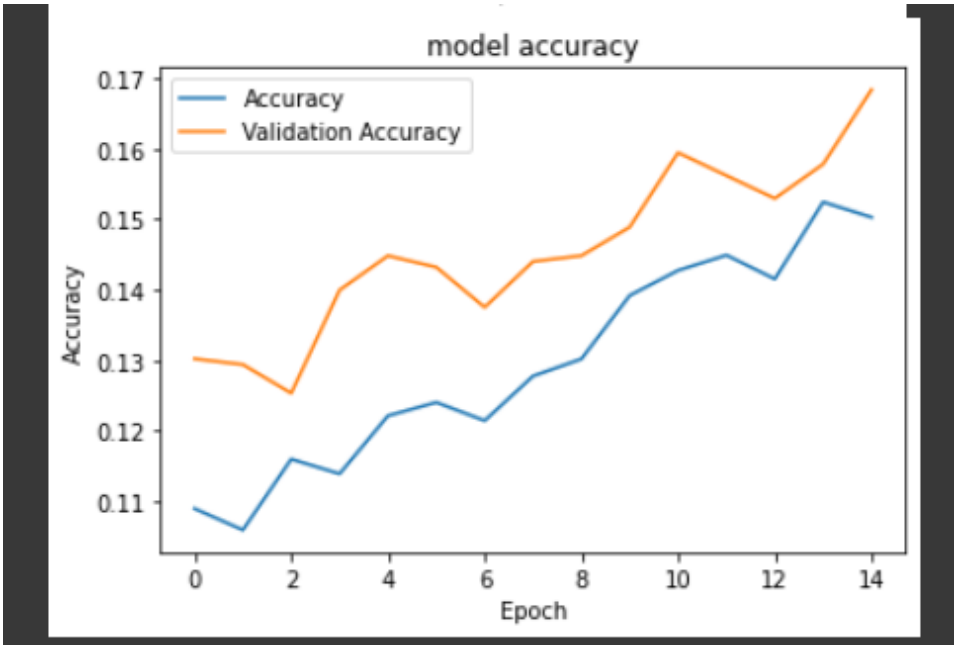
We then trained the new model



This shows us that the results are very good since the new model is now learning since both losses are decreasing and both accuracies are increasing.


loss
3.75
3.70
3.65
3.60
3.55
3.50
3.45
3.40
0
2
4
6
8
10
12
14
Epoch

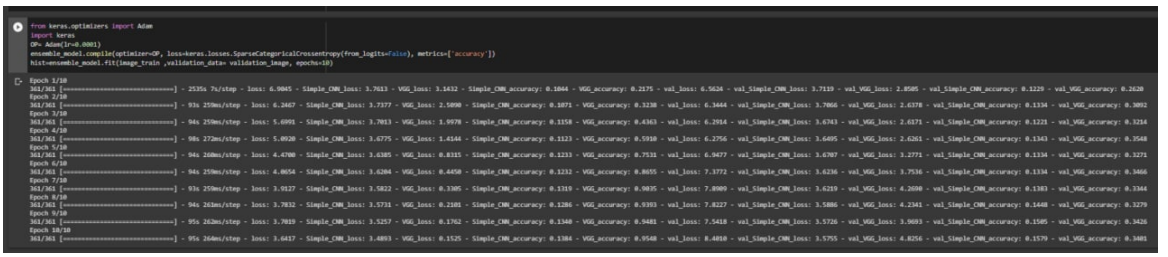

0
2
4
6
8
10
12
14
Epoch
model accuracy
Accuracy
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Accuracy
Validation Accuracy
loss
Validation Loss
0
2
4
6
8
10
12
14
Epoch



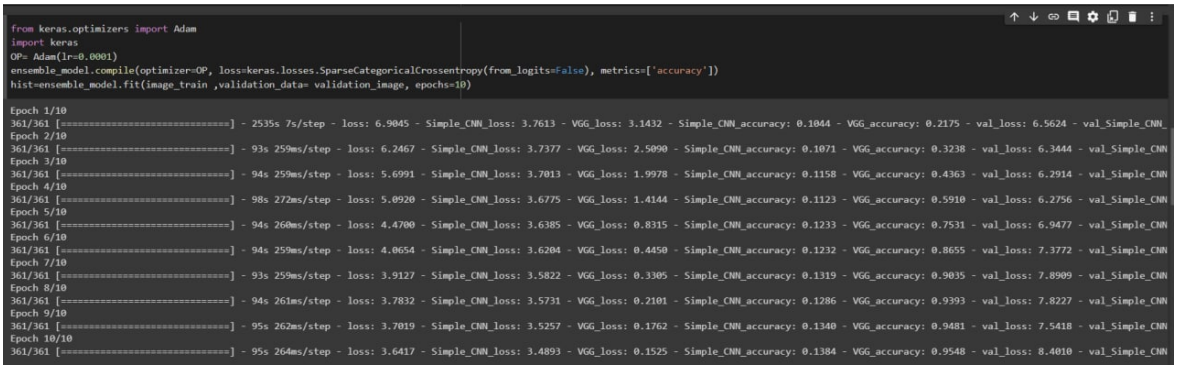
The Model got an F-score of 0.00040074670778970434

Ensemble

We can use the ensemble method were we use two of our models together to get better results, we used our simple CNN and the VGG.



```
from keras.optimizers import Adam
import keras
OP= Adam(lr=0.0001)
ensemble_model.compile(optimizer=OP, loss=keras.losses.SparseCategoricalCrossentropy(from_logits=False), metrics=['accuracy'])
hist=ensemble_model.fit(image_train ,validation_data= validation_image, epochs=10)

Epoch 1/10
361/361 [==============================] - 2535s 7s/step - loss: 6.9045 - Simple_CNN_loss: 3.7613 - VGG_loss: 3.1432 - Simple_CNN_accuracy: 0.1044 - VGG_accuracy: 0.2175 - val_loss: 6.5624 - val_Simple_CNN_loss: 3.7139 - val_VGG_loss: 2.8505 - val_Simple_CNN_accuracy: 0.1229 - val_VGG_accuracy: 0.2620
Epoch 2/10
361/361 [==============================] - 93s 259ms/step - loss: 6.2467 - Simple_CNN_loss: 3.7377 - VGG_loss: 2.5090 - Simple_CNN_accuracy: 0.1071 - VGG_accuracy: 0.3238 - val_loss: 6.3444 - val_Simple_CNN_loss: 3.7066 - val_VGG_loss: 2.6378 - val_Simple_CNN_accuracy: 0.1334 - val_VGG_accuracy: 0.3092
Epoch 3/10
361/361 [==============================] - 94s 259ms/step - loss: 5.6991 - Simple_CNN_loss: 3.7013 - VGG_loss: 1.9978 - Simple_CNN_accuracy: 0.1158 - VGG_accuracy: 0.4363 - val_loss: 6.2914 - val_Simple_CNN_loss: 3.6743 - val_VGG_loss: 2.6171 - val_Simple_CNN_accuracy: 0.1221 - val_VGG_accuracy: 0.3214
Epoch 4/10
361/361 [==============================] - 98s 272ms/step - loss: 5.0920 - Simple_CNN_loss: 3.6775 - VGG_loss: 1.4144 - Simple_CNN_accuracy: 0.1123 - VGG_accuracy: 0.5910 - val_loss: 6.2756 - val_Simple_CNN_loss: 3.6495 - val_VGG_loss: 2.6261 - val_Simple_CNN_accuracy: 0.1343 - val_VGG_accuracy: 0.3548
Epoch 5/10
361/361 [==============================] - 94s 260ms/step - loss: 4.4700 - Simple_CNN_loss: 3.6385 - VGG_loss: 0.8315 - Simple_CNN_accuracy: 0.1233 - VGG_accuracy: 0.7531 - val_loss: 6.9477 - val_Simple_CNN_loss: 3.6707 - val_VGG_loss: 3.2771 - val_Simple_CNN_accuracy: 0.1334 - val_VGG_accuracy: 0.3271
Epoch 6/10
361/361 [==============================] - 94s 259ms/step - loss: 4.0654 - Simple_CNN_loss: 3.6204 - VGG_loss: 0.4450 - Simple_CNN_accuracy: 0.1232 - VGG_accuracy: 0.8655 - val_loss: 7.3772 - val_Simple_CNN_loss: 3.6236 - val_VGG_loss: 3.7536 - val_Simple_CNN_accuracy: 0.1334 - val_VGG_accuracy: 0.3466
Epoch 7/10
361/361 [==============================] - 93s 259ms/step - loss: 3.9127 - Simple_CNN_loss: 3.5822 - VGG_loss: 0.3305 - Simple_CNN_accuracy: 0.1319 - VGG_accuracy: 0.9035 - val_loss: 7.8909 - val_Simple_CNN_loss: 3.6219 - val_VGG_loss: 4.2690 - val_Simple_CNN_accuracy: 0.1383 - val_VGG_accuracy: 0.3344
Epoch 8/10
361/361 [==============================] - 94s 263ms/step - loss: 3.7832 - Simple_CNN_loss: 3.5731 - VGG_loss: 0.2101 - Simple_CNN_accuracy: 0.1286 - VGG_accuracy: 0.9393 - val_loss: 7.8227 - val_Simple_CNN_loss: 3.5886 - val_VGG_loss: 4.2341 - val_Simple_CNN_accuracy: 0.1448 - val_VGG_accuracy: 0.3279
Epoch 9/10
361/361 [==============================] - 95s 262ms/step - loss: 3.7019 - Simple_CNN_loss: 3.5257 - VGG_loss: 0.1762 - Simple_CNN_accuracy: 0.1340 - VGG_accuracy: 0.9481 - val_loss: 7.5418 - val_Simple_CNN_loss: 3.5726 - val_VGG_loss: 3.9693 - val_Simple_CNN_accuracy: 0.1505 - val_VGG_accuracy: 0.3426
Epoch 10/10
361/361 [==============================] - 95s 264ms/step - loss: 3.6417 - Simple_CNN_loss: 3.4893 - VGG_loss: 0.1525 - Simple_CNN_accuracy: 0.1384 - VGG_accuracy: 0.9548 - val_loss: 8.4010 - val_Simple_CNN_loss: 3.5755 - val_VGG_loss: 4.8256 - val_Simple_CNN_accuracy: 0.1579 - val_VGG_accuracy: 0.3401
```



```
from keras.optimizers import Adam
import keras
OP= Adam(lr=0.0001)
ensemble_model.compile(optimizer=OP, loss=keras.losses.SparseCategoricalCrossentropy(from_logits=False), metrics=['accuracy'])
hist=ensemble_model.fit(image_train ,validation_data= validation_image, epochs=10)

Epoch 1/10
361/361 [==============================] - 2535s 7s/step - loss: 6.9045 - Simple_CNN_loss: 3.7613 - VGG_loss: 3.1432 - Simple_CNN_accuracy: 0.1044 - VGG_accuracy: 0.2175 - val_loss: 6.5624 - val_Simple_CNN_
Epoch 2/10
361/361 [==============================] - 93s 259ms/step - loss: 6.2467 - Simple_CNN_loss: 3.7377 - VGG_loss: 2.5090 - Simple_CNN_accuracy: 0.1071 - VGG_accuracy: 0.3238 - val_loss: 6.3444 - val_Simple_CNN
Epoch 3/10
361/361 [==============================] - 94s 259ms/step - loss: 5.6991 - Simple_CNN_loss: 3.7013 - VGG_loss: 1.9978 - Simple_CNN_accuracy: 0.1158 - VGG_accuracy: 0.4363 - val_loss: 6.2914 - val_Simple_CNN
Epoch 4/10
361/361 [==============================] - 98s 272ms/step - loss: 5.0920 - Simple_CNN_loss: 3.6775 - VGG_loss: 1.4144 - Simple_CNN_accuracy: 0.1123 - VGG_accuracy: 0.5910 - val_loss: 6.2756 - val_Simple_CNN
Epoch 5/10
361/361 [==============================] - 94s 260ms/step - loss: 4.4700 - Simple_CNN_loss: 3.6385 - VGG_loss: 0.8315 - Simple_CNN_accuracy: 0.1233 - VGG_accuracy: 0.7531 - val_loss: 6.9477 - val_Simple_CNN
Epoch 6/10
361/361 [==============================] - 94s 259ms/step - loss: 4.0654 - Simple_CNN_loss: 3.6204 - VGG_loss: 0.4450 - Simple_CNN_accuracy: 0.1232 - VGG_accuracy: 0.8655 - val_loss: 7.3772 - val_Simple_CNN
Epoch 7/10
361/361 [==============================] - 93s 259ms/step - loss: 3.9127 - Simple_CNN_loss: 3.5822 - VGG_loss: 0.3305 - Simple_CNN_accuracy: 0.1319 - VGG_accuracy: 0.9035 - val_loss: 7.8909 - val_Simple_CNN
Epoch 8/10
361/361 [==============================] - 94s 261ms/step - loss: 3.7832 - Simple_CNN_loss: 3.5731 - VGG_loss: 0.2101 - Simple_CNN_accuracy: 0.1286 - VGG_accuracy: 0.9393 - val_loss: 7.8227 - val_Simple_CNN
Epoch 9/10
361/361 [==============================] - 95s 262ms/step - loss: 3.7019 - Simple_CNN_loss: 3.5257 - VGG_loss: 0.1762 - Simple_CNN_accuracy: 0.1340 - VGG_accuracy: 0.9481 - val_loss: 7.5418 - val_Simple_CNN
Epoch 10/10
361/361 [==============================] - 95s 264ms/step - loss: 3.6417 - Simple_CNN_loss: 3.4893 - VGG_loss: 0.1525 - Simple_CNN_accuracy: 0.1384 - VGG_accuracy: 0.9548 - val_loss: 8.4010 - val_Simple_CNN
```

These are the results of the model.

Comparison between All Models

	VGG	CNN	GoogleNet	Ensemble	transfer
Test_loss	3.5339882373809814	3.7658066749572754	2.2606270313262940	8.5749387741088870	Nan
Test_accuracy	0.2957974076271057	0.1225754320621490	0.4245689511299133	3.6041183471679688	0.0177801717072725