Explanation of the code:

I have used the 50 convolutional layer VGG16 model with the transfer learning component of CNN.

Made a simple image classifier with the default settings, added the features, converted and saved to the bottleneck file for training, validation and testing data. We prepare them for our convolutional neural network by loading them.

First step is to initialise the model with Sequential() method. Then, we flatten our data and add our additional three hidden layers.

We created a number of different models with various hidden layers, drop outs, and activations. The ultimate activation, however, must always be softmax because this is a labelled categorical categorization.

After building and compiling our model, we fit our training and validation data to it using the criteria we previously outlined. In order to compare the accuracy of our model training set and validation set, we add an evaluation step as the last step.

Training data – 80%

Validation data – 10%

Testing data – 10%

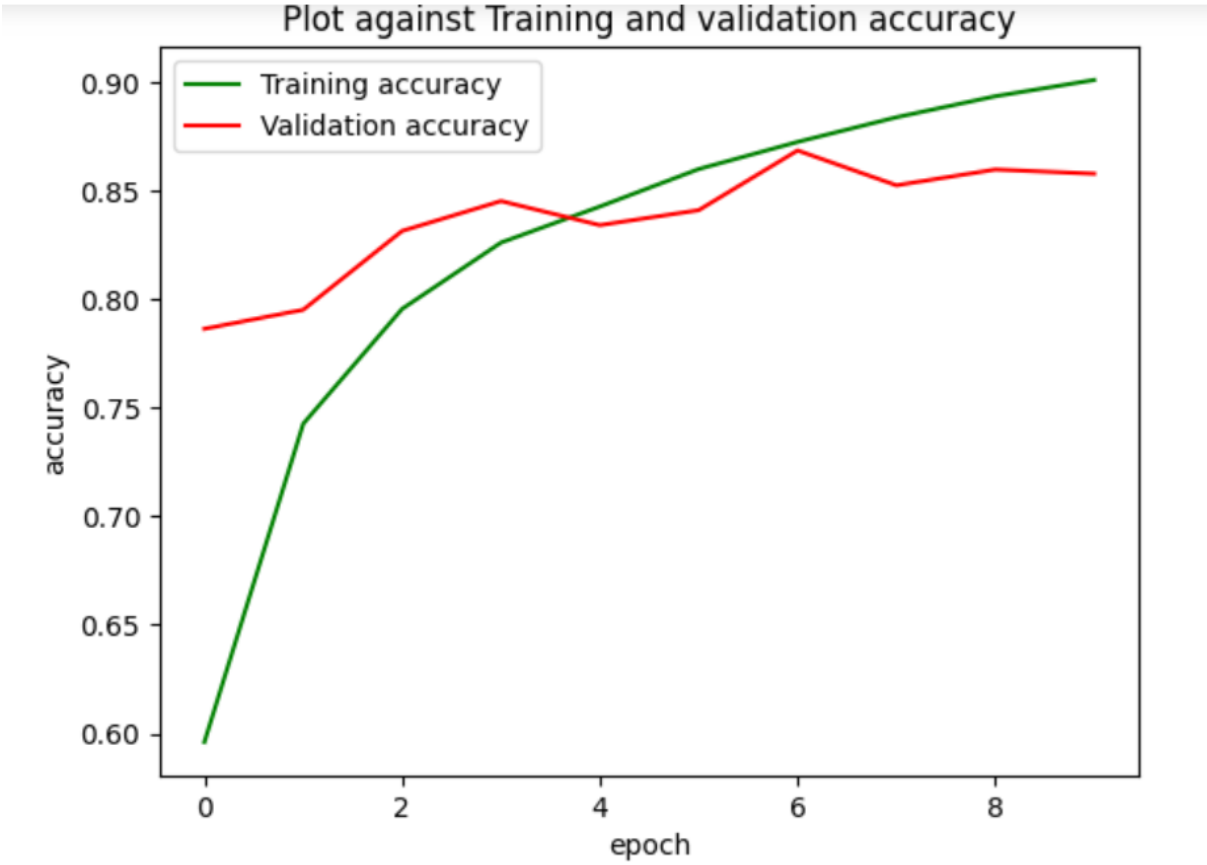
I have also validated the test data with the built model as its the unseen data. It provides us the better accuracy.

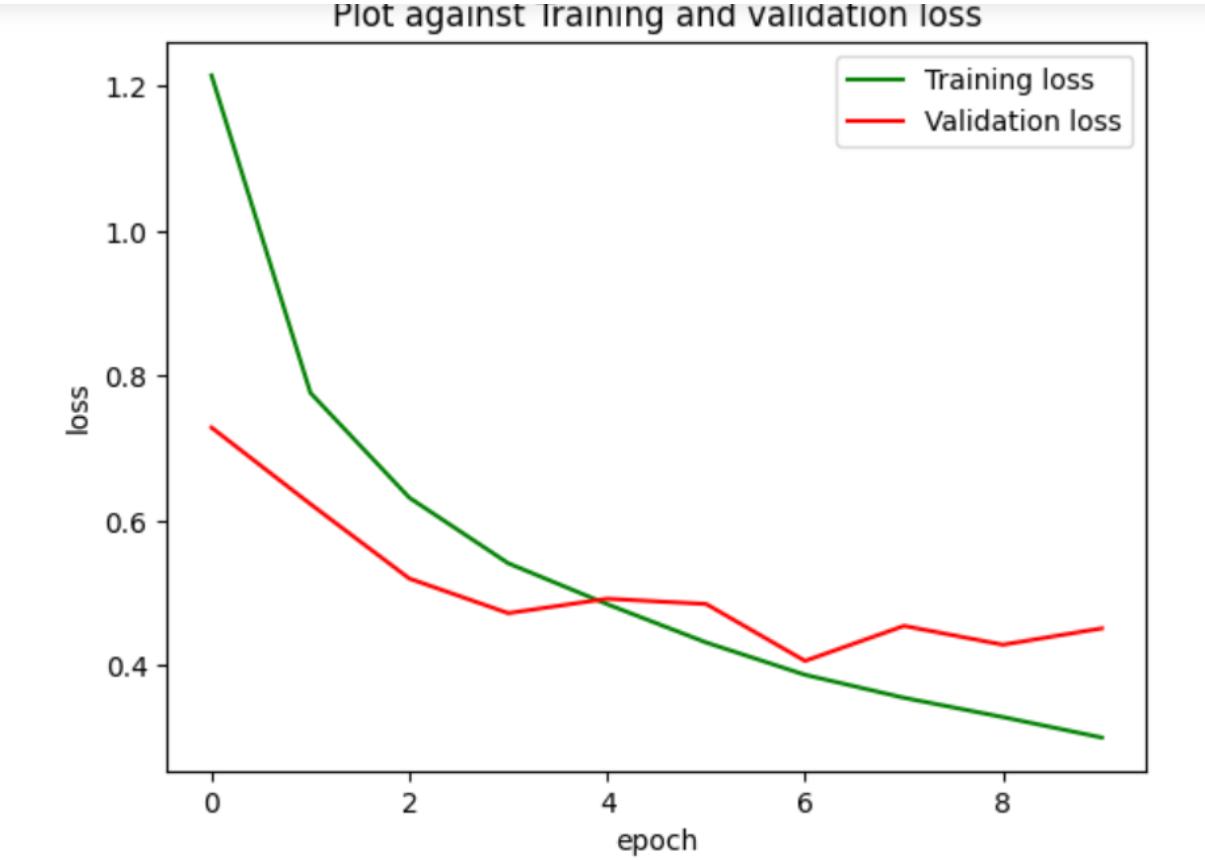
Output:

Accuracy: 85.81%

Running model for 10 epochs

```
Epoch 1/10
419/419 [==============================] - 36s 81ms/step - loss: 1.2148 - acc: 0.5960 - val_loss: 0.7286 - val_acc: 0.7865
Epoch 2/10
419/419 [==============================] - 28s 66ms/step - loss: 0.7765 - acc: 0.7428 - val_loss: 0.6230 - val_acc: 0.7953
Epoch 3/10
419/419 [==============================] - 25s 59ms/step - loss: 0.6317 - acc: 0.7957 - val_loss: 0.5201 - val_acc: 0.8317
Epoch 4/10
419/419 [==============================] - 23s 54ms/step - loss: 0.5411 - acc: 0.8263 - val_loss: 0.4723 - val_acc: 0.8454
Epoch 5/10
419/419 [==============================] - 25s 60ms/step - loss: 0.4847 - acc: 0.8429 - val_loss: 0.4922 - val_acc: 0.8344
Epoch 6/10
419/419 [==============================] - 24s 57ms/step - loss: 0.4318 - acc: 0.8602 - val_loss: 0.4849 - val_acc: 0.8412
Epoch 7/10
419/419 [==============================] - 26s 62ms/step - loss: 0.3872 - acc: 0.8727 - val_loss: 0.4064 - val_acc: 0.8688
Epoch 8/10
419/419 [==============================] - 41s 99ms/step - loss: 0.3555 - acc: 0.8841 - val_loss: 0.4547 - val_acc: 0.8527
Epoch 9/10
419/419 [==============================] - 28s 66ms/step - loss: 0.3288 - acc: 0.8938 - val_loss: 0.4287 - val_acc: 0.8600
Epoch 10/10
419/419 [==============================] - 26s 61ms/step - loss: 0.3005 - acc: 0.9013 - val_loss: 0.4513 - val_acc: 0.8581
53/53 [==============================] - 1s 15ms/step - loss: 0.4513 - acc: 0.8581
[INFO] accuracy: 85.81%
[INFO] Loss: 0.4513357877731323
Time: 0:04:44.032758
```


Plot against Training and validation accuracy
Training accuracy
Validation accuracy
accuracy
epoch
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
2
4
6
8



Evaluating test data:

```
In [47]: model.evaluate(test_data, test_labels)

         83/83 [==============================] - 2s 18ms/step - loss: 0.4417 - acc: 0.8592

Out[47]: [0.44174256920814514, 0.8591549396514893]

In [48]: preds = np.round(model.predict(test_data), 0)
         print('Result ', preds)

         83/83 [==============================] - 1s 10ms/step
         Result  [[1. 0. 0. ... 0. 0. 0.]
          [1. 0. 0. ... 0. 0. 0.]
          [1. 0. 0. ... 0. 0. 0.]
          ...
          [0. 0. 0. ... 0. 0. 1.]
          [0. 0. 0. ... 0. 0. 1.]
          [0. 0. 0. ... 0. 0. 1.]]
```

Classification Metrics with Precision, recall, f1-score and support:

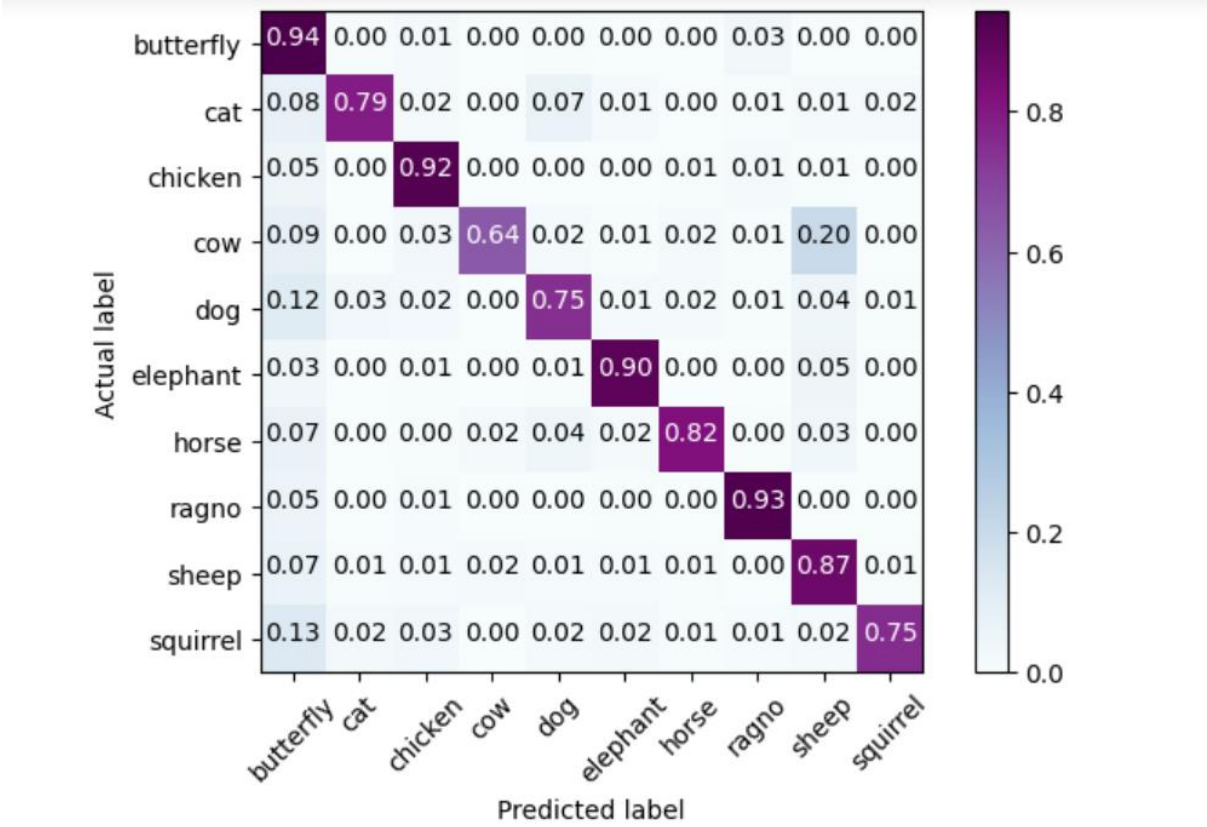
In [42]:
```
animals = ['butterfly', 'cat', 'chicken', 'cow', 'dog', 'elephant', 'horse', 'ragno', 'sheep', 'squirrel']
class_metrics = metrics.classification_report(test_labels, preds, target_names=animals)
print(class_metrics)
```

```
              precision    recall  f1-score   support

   butterfly       0.96      0.90      0.93       212
         cat       0.92      0.75      0.83       168
     chicken       0.94      0.88      0.91       311
         cow       0.93      0.60      0.72       188
         dog       0.83      0.87      0.85       487
    elephant       0.85      0.92      0.88       146
       horse       0.93      0.78      0.85       263
       ragno       0.96      0.94      0.95       483
       sheep       0.81      0.71      0.76       182
    squirrel       0.89      0.80      0.84       187

   micro avg       0.90      0.84      0.87      2627
   macro avg       0.90      0.81      0.85      2627
weighted avg       0.90      0.84      0.87      2627
 samples avg       0.84      0.84      0.84      2627
```

Confusion Matrix:



Testing the Model with single image to check the accuracy:

Placed the image of sheep in the working directory and below is the output and accuracy.

```
path = 'animaldst\sheep.PNG'
test_image(path)
```

```
[INFO] Reading Image
1/1 [==============================] - 0s 355ms/step
1/1 [==============================] - 0s 93ms/step
ID: 0, Label: butterfly 0.0%
ID: 1, Label: cat 0.0%
ID: 2, Label: chicken 0.0%
ID: 3, Label: cow 0.06%
ID: 4, Label: dog 0.0%
ID: 5, Label: elephant 0.02%
ID: 6, Label: horse 0.0%
ID: 7, Label: ragno 0.0%
ID: 8, Label: sheep 99.92%
ID: 9, Label: squirrel 0.0%
Final Result:
.
..
...
ID: 8, Label: sheep
```

Out[44]: