```
import pandas as pd
import numpy as np
import itertools
import keras
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from keras.models import Sequential
from keras import optimizers
from keras.preprocessing import image
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from keras.utils.np_utils import to_categorical
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import math
import datetime
import time
```

```
Ising TensorFlow backend.
```

Loading up our image datasets

```
#Default dimensions we found online
img_width, img_height = 224, 224

#Create a bottleneck file
top_model_weights_path = 'bottleneck_fc_model.h5'

# loading up our datasets
train_data_dir = 'data/train'
validation_data_dir = 'data/validation'
test_data_dir = 'data/test'

# number of epochs to train top model
epochs = 7 #this has been changed after multiple model run
# batch size used by flow_from_directory and predict_generator
batch_size = 50
```

```
#Loading vgc16 model
vgg16 = applications.VGG16(include_top=False, weights='imagenet')
```

```
datagen = ImageDataGenerator(rescale=1. / 255)  #needed to create the bottleneck .npy files
```

Creation of weights/features with VGG16

```
#__this can take an hour and half to run so only run it once.
#once the npy files have been created, no need to run again. Convert this cell to a code cell to run

start = datetime.datetime.now()

generator = datagen.flow_from_directory(
     train_data_dir,
     target_size=(img_width, img_height),
     batch_size=batch_size,
     class_mode=None,
     shuffle=False)

nb_train_samples = len(generator.filenames)
num_classes = len(generator.class_indices)

predict_size_train = int(math.ceil(nb_train_samples / batch_size))

bottleneck_features_train = vgg16.predict_generator(generator, predict_size_train)

np.save('bottleneck_features_train.npy', bottleneck_features_train)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

```
print('-'*117)
```

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

```
#__this can take half an hour to run so only run it once. once the npy files have been created, no n

start = datetime.datetime.now()
generator = datagen.flow_from_directory(
     validation_data_dir,
     target_size=(img_width, img_height),
     batch_size=batch_size,
     class_mode=None,
     shuffle=False)

nb_validation_samples = len(generator.filenames)

predict_size_validation = int(math.ceil(nb_validation_samples / batch_size))

bottleneck_features_validation = vgg16.predict_generator(
     generator, predict_size_validation)
```

```
#training data
generator_top = datagen.flow_from_directory(
         train_data_dir,
         target_size=(img_width, img_height),
         batch_size=batch_size,
         class_mode='categorical',
         shuffle=False)

nb_train_samples = len(generator_top.filenames)
num_classes = len(generator_top.class_indices)

# load the bottleneck features saved earlier
train_data = np.load('bottleneck_features_train.npy')

# get the class lebels for the training data, in the original order
train_labels = generator_top.classes

# convert the training labels to categorical vectors
train_labels = to_categorical(train_labels, num_classes=num_classes)
```

```
Found 13412 images belonging to 6 classes.
```

```
#validation data
generator_top = datagen.flow_from_directory(
         validation_data_dir,
         target_size=(img_width, img_height),
         batch_size=batch_size,
         class_mode=None,
         shuffle=False)

nb_validation_samples = len(generator_top.filenames)

validation_data = np.load('bottleneck_features_validation.npy')


validation_labels = generator_top.classes
validation_labels = to_categorical(validation_labels, num_classes=num_classes)
```

```
Found 2549 images belonging to 6 classes.
```

```
#testing data
generator_top = datagen.flow_from_directory(
         test_data_dir,
         target_size=(img_width, img_height),
         batch_size=batch_size,
         class_mode=None,
         shuffle=False)

nb_test_samples = len(generator_top.filenames)

test_data = np.load('bottleneck_features_test.npy')


test_labels = generator_top.classes
test_labels = to_categorical(test_labels, num_classes=num_classes)
```

```
Found 1845 images belonging to 6 classes.
```

Training of model

```
#This is the best model we found. For additional models, check out I_notebook.ipynb
start = datetime.datetime.now()
model = Sequential()
model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.5))
model.add(Dense(50, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])

history = model.fit(train_data, train_labels,
      epochs=7,
      batch_size=batch_size,
      validation_data=(validation_data, validation_labels))

model.save_weights(top_model_weights_path)

(eval_loss, eval_accuracy) = model.evaluate(
 validation_data, validation_labels, batch_size=batch_size, verbose=1)

print("[INFO] accuracy: {:.2f}%".format(eval_accuracy * 100))
print("[INFO] Loss: {}".format(eval_loss))
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

```
/Users/Iffy/anaconda3/lib/python3.6/site-packages/keras/activations.py:209: UserWarning: Do not pass a
layer instance (such as LeakyReLU) as the activation argument of another layer. Instead, advanced activ
ation layers should be used just like any other layer in a model.
  identifier=identifier.__class__.__name__))
Train on 13412 samples, validate on 2549 samples
Epoch 1/7
13412/13412 [==============================] - 9s 638us/step - loss: 0.7518 - acc: 0.7349 - val_loss:
0.4372 - val_acc: 0.8666
Epoch 2/7
13412/13412 [==============================] - 6s 476us/step - loss: 0.4194 - acc: 0.8598 - val_loss:
0.3507 - val_acc: 0.8886
Epoch 3/7
13412/13412 [==============================] - 7s 523us/step - loss: 0.3327 - acc: 0.8876 - val_loss:
0.3068 - val_acc: 0.8992
Epoch 4/7
13412/13412 [==============================] - 6s 471us/step - loss: 0.2771 - acc: 0.9055 - val_loss:
0.2726 - val_acc: 0.9117
Epoch 5/7
```

```
start = datetime.datetime.now()
model = Sequential()
model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.5))
model.add(Dense(50, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])

history = model.fit(train_data, train_labels,
      epochs=7,
      batch_size=batch_size,
      validation_data=(validation_data, validation_labels))

model.save_weights(top_model_weights_path)

(eval_loss, eval_accuracy) = model.evaluate(
 validation_data, validation_labels, batch_size=batch_size, verbose=1)

print("[INFO] accuracy: {:.2f}%".format(eval_accuracy * 100))
print("[INFO] Loss: {}".format(eval_loss))
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

```
/Users/Iffy/anaconda3/lib/python3.6/site-packages/keras/activations.py:209: UserWarning: Do not pass a
layer instance (such as LeakyReLU) as the activation argument of another layer. Instead, advanced activ
ation layers should be used just like any other layer in a model.
  identifier=identifier.__class__.__name__))
Train on 13412 samples, validate on 2549 samples
Epoch 1/7
13412/13412 [==============================] - 9s 638us/step - loss: 0.7518 - acc: 0.7349 - val_loss:
0.4372 - val_acc: 0.8666
Epoch 2/7
13412/13412 [==============================] - 6s 476us/step - loss: 0.4194 - acc: 0.8598 - val_loss:
0.3507 - val_acc: 0.8886
Epoch 3/7
13412/13412 [==============================] - 7s 523us/step - loss: 0.3327 - acc: 0.8876 - val_loss:
0.3068 - val_acc: 0.8992
Epoch 4/7
13412/13412 [==============================] - 6s 471us/step - loss: 0.2771 - acc: 0.9055 - val_loss:
0.2726 - val_acc: 0.9117
Epoch 5/7
13412/13412 [==============================] - 6s 444us/step - loss: 0.2433 - acc: 0.9208 - val_loss:
0.2904 - val_acc: 0.9082
Epoch 6/7
13412/13412 [==============================] - 6s 455us/step - loss: 0.2115 - acc: 0.9294 - val_loss:
0.2852 - val_acc: 0.9109
Epoch 7/7
13412/13412 [==============================] - 6s 477us/step - loss: 0.1869 - acc: 0.9379 - val_loss:
0.2703 - val_acc: 0.9211
2549/2549 [==============================] - 0s 96us/step
[INFO] accuracy: 92.11%
[INFO] Loss: 0.2703172541517587
Time:  0:00:47.456195
```

```
#Model summary
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               2508900
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 306
=================================================================
Total params: 2,514,256
Trainable params: 2,514,256
Non-trainable params: 0
_________________________________________________________________
```

```
#Graphing our training and validation
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))
plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
```

```python
print('test data', test_data)
preds = np.round(model.predict(test_data),0)
#to fit them into classification metrics and confusion metrics, some additional modificaition
print('rounded test_labels', preds)
```

```
test data [[[[1.62910283e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    7.54943728e-01 0.00000000e+00]
   [9.73952040e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    6.17594182e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    4.81749803e-01 0.00000000e+00]
   ...
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    4.90739763e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    4.19955909e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    5.68091333e-01 0.00000000e+00]]

  [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    5.71550012e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 2.10543305e-01 ... 0.00000000e+00
    3.51710707e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 6.56101227e-01 ... 0.00000000e+00
    1.31017089e-01 0.00000000e+00]
   ...
   [0.00000000e+00 0.00000000e+00 4.20266658e-01 ... 0.00000000e+00
    2.72110671e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 6.01641834e-01 ... 0.00000000e+00
    2.36603856e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    3.80211979e-01 0.00000000e+00]]

  [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    6.11291170e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 1.34456348e+00 ... 0.00000000e+00
    2.73023814e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 2.04151082e+00 ... 0.00000000e+00
    0.00000000e+00 0.00000000e+00]
   ...
   [1.85347736e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    0.00000000e+00 0.00000000e+00]
   [3.18986595e-01 0.00000000e+00 5.86764395e-01 ... 5.43534338e-01
    2.87956595e-02 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 3.07244062e-02 ... 0.00000000e+00
    1.65340304e-01 0.00000000e+00]]

  ...

  [[0.00000000e+00 0.00000000e+00 2.26079583e-01 ... 0.00000000e+00
    7.81377614e-01 0.00000000e+00]
   [3.04938525e-01 0.00000000e+00 8.86197925e-01 ... 0.00000000e+00
    8.57958436e-01 0.00000000e+00]
   [1.00945687e+00 0.00000000e+00 1.43036103e+00 ... 0.00000000e+00
    0.00000000e+00 0.00000000e+00]
   ...
   [7.55731761e-03 0.00000000e+00 1.38314462e+00 ... 0.00000000e+00
    4.19555604e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 5.41599452e-01 ... 0.00000000e+00
    0.00000000e+00 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    4.43584591e-01 0.00000000e+00]]

  [[0.00000000e+00 0.00000000e+00 3.35014820e-01 ... 0.00000000e+00
    9.47265744e-01 0.00000000e+00]
   [2.91338444e-01 0.00000000e+00 9.04450059e-01 ... 0.00000000e+00
    1.17925084e+00 0.00000000e+00]
   [5.30939519e-01 0.00000000e+00 1.83613181e+00 ... 0.00000000e+00
    8.10585260e-01 0.00000000e+00]
   ...
   [0.00000000e+00 0.00000000e+00 1.43227601e+00 ... 0.00000000e+00
    3.92884314e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    5.36076784e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    5.76381624e-01 0.00000000e+00]]

  [[0.00000000e+00 0.00000000e+00 1.76035732e-01 ... 0.00000000e+00
    1.09022713e+00 0.00000000e+00]
   [2.85193682e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    9.74584103e-01 0.00000000e+00]
   [3.71666878e-01 0.00000000e+00 5.94100416e-01 ... 0.00000000e+00
    9.23797011e-01 0.00000000e+00]
   ...
   [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    1.29053831e+00 0.00000000e+00]
   [2.23176628e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    9.99898195e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 4.88149226e-02 ... 0.00000000e+00
    6.57890737e-01 0.00000000e+00]]]


 [[[2.73834437e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
    7.26639390e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 1.56045794e+00 ... 0.00000000e+00
    5.10039389e-01 0.00000000e+00]
   [0.00000000e+00 0.00000000e+00 1.78128552e+00 ... 0.00000000e+00
    7.29214787e-01 9.99811292e-03]
```

```
  ...
  [4.69681889e-01 0.00000000e+00 6.32544160e-02 ... 2.00737894e-01
   1.11459517e+00 0.00000000e+00]
  [4.64970171e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   7.64544129e-01 0.00000000e+00]
  [4.96487468e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.52200997e-01 0.00000000e+00]]

 [[4.33659881e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.07863486e-01 0.00000000e+00]
  [6.97760403e-01 0.00000000e+00 4.40489322e-01 ... 0.00000000e+00
   7.63945222e-01 0.00000000e+00]
  [1.00053179e+00 0.00000000e+00 3.65804732e-01 ... 0.00000000e+00
   0.00000000e+00 0.00000000e+00]
  ...
  [8.14497292e-01 0.00000000e+00 3.87326807e-01 ... 0.00000000e+00
   5.44842541e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.37416482e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.30150568e-01 0.00000000e+00]]

 ...

 [[3.54413331e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.45500720e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 3.64198267e-01 ... 0.00000000e+00
   1.18480647e+00 0.00000000e+00]
  [1.96117371e-01 0.00000000e+00 7.44675756e-01 ... 0.00000000e+00
   3.56948584e-01 0.00000000e+00]
  ...
  [1.64511085e-01 3.31927419e-01 6.59318984e-01 ... 0.00000000e+00
   1.87187105e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.27904713e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.23119366e-01 0.00000000e+00]]

 [[4.07837689e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.00833774e-01 0.00000000e+00]
  [1.52882561e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   1.26926088e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 6.77286506e-01 ... 0.00000000e+00
   1.18739462e+00 0.00000000e+00]
  ...
  [1.94572821e-01 0.00000000e+00 7.14387417e-01 ... 0.00000000e+00
   6.55267060e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.60734153e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   3.75418663e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   7.07058311e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.70728898e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 4.32805419e-02 ... 0.00000000e+00
   6.65341496e-01 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.44046378e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   3.40202093e-01 0.00000000e+00]
  [6.79454654e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.36539412e-01 0.00000000e+00]]]


...


[[[2.77962089e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   7.59022474e-01 0.00000000e+00]
  [2.10890427e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.45528197e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   3.06156039e-01 0.00000000e+00]
  ...
  [7.65709952e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.90672410e-01 0.00000000e+00]
  [1.68913573e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.91591799e-01 0.00000000e+00]
  [1.58170596e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   7.52174616e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.87323725e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.30910790e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 7.22792864e-01 ... 4.86614019e-01
   2.27496266e-01 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 2.91275799e-01 ... 1.38625443e-01
   3.95227283e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.14201224e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.94563961e-01 0.00000000e+00]]
```

```
   7.59022474e-01 0.00000000e+00]
  [2.10890427e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.45528197e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   3.06156039e-01 0.00000000e+00]
  ...
  [7.65709952e-02 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.90672410e-01 0.00000000e+00]
  [1.68913573e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.91591799e-01 0.00000000e+00]
  [1.58170596e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   7.52174616e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.87323725e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.30910790e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 7.22792864e-01 ... 4.86614019e-01
   2.27496266e-01 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 2.91275799e-01 ... 1.38625443e-01
   3.95227283e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.14201224e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.94563961e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 1.10778183e-01 ... 0.00000000e+00
   1.24503601e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.23695505e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 7.24857867e-01 ... 1.27987653e-01
   3.89741361e-01 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 4.00016963e-01 ... 1.33965030e-01
   2.96326399e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.04090369e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.15781486e-01 0.00000000e+00]]

 ...

 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   1.34632349e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 6.43011749e-01 ... 0.00000000e+00
   1.63106382e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 1.01285982e+00 ... 0.00000000e+00
   1.48830283e+00 0.00000000e+00]
  ...
  [2.53828496e-01 0.00000000e+00 2.45893747e-01 ... 0.00000000e+00
   7.90464878e-01 0.00000000e+00]
  [1.21133916e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.07814705e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   3.98586988e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 3.99015620e-02
   1.34220672e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 6.79775238e-01 ... 2.73587614e-01
   1.62597466e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 1.37078023e+00 ... 0.00000000e+00
   1.75564671e+00 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 2.93850899e-01 ... 0.00000000e+00
   1.37390518e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 1.83160990e-01 ... 0.00000000e+00
   1.51765156e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   4.97628450e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   1.73606968e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 3.80915403e-03 ... 0.00000000e+00
   1.00736690e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 6.25692248e-01 ... 0.00000000e+00
   9.75181103e-01 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 2.93635547e-01 ... 0.00000000e+00
   1.32252169e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 1.80633038e-01 ... 0.00000000e+00
   1.34793413e+00 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 2.66640127e-01 ... 1.33897662e-02
   1.01395643e+00 0.00000000e+00]]]


[[[3.19847614e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   6.40549123e-01 0.00000000e+00]
  [6.11765683e-03 0.00000000e+00 6.50527060e-01 ... 0.00000000e+00
   8.13049853e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 1.77103472e+00 ... 0.00000000e+00
   5.48596680e-01 0.00000000e+00]
  ...
  [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   1.22478902e-01 0.00000000e+00]
  [0.00000000e+00 0.00000000e+00 5.24263859e-01 ... 0.00000000e+00
   4.55555439e-01 0.00000000e+00]
  [2.09953845e-01 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
   5.04809380e-01 0.00000000e+00]]

 [[0.00000000e+00 0.00000000e+00 1.12915546e-01 ... 0.00000000e+00
```

```
 ...
 [0.00000000e+00 0.00000000e+00 6.70512557e-01 ... 0.00000000e+00
  7.56995499e-01 0.00000000e+00]
 [6.53010607e-01 0.00000000e+00 5.78189790e-02 ... 0.00000000e+00
  1.21809459e+00 0.00000000e+00]
 [1.18682408e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  1.06893003e+00 0.00000000e+00]]]]
rounded test_labels [[1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 ...
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1.]]
```

```python
animals = ['butterflies', 'chickens', 'elephants', 'horses', 'spiders', 'squirells']
classification_metrics = metrics.classification_report(test_labels, preds, target_names=animals )
print(classification_metrics)
```

```
              precision    recall  f1-score   support

 butterflies       0.98      0.88      0.93       371
    chickens       0.94      0.85      0.89       203
   elephants       0.91      0.89      0.90       152
      horses       0.97      0.95      0.96       472
     spiders       0.92      0.95      0.93       403
   squirells       0.92      0.91      0.92       244

   micro avg       0.94      0.92      0.93      1845
   macro avg       0.94      0.91      0.92      1845
weighted avg       0.94      0.92      0.93      1845
 samples avg       0.92      0.92      0.92      1845
```

```
/Users/Iffy/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1143: UndefinedM
icWarning: Precision and F-score are ill-defined and being set to 0.0 in samples with no predicted l
ls.
  'precision', 'predicted', average, warn_for)
```

Confusion Matrix

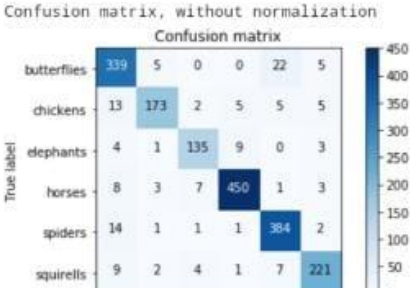
```python
#Since our data is in dummy format we put the numpy array into a dataframe and call idxmax axis=1
# label of the maximum value thus creating a categorical variable
#Basically, flipping a dummy variable back to it's categorical variable
categorical_test_labels = pd.DataFrame(test_labels).idxmax(axis=1)
categorical_preds = pd.DataFrame(preds).idxmax(axis=1)
```

```python
confusion_matrix= confusion_matrix(categorical_test_labels, categorical_preds)
```

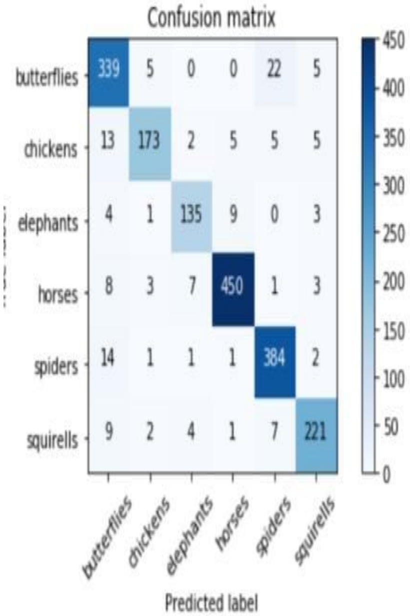
```python
#To get better visual of the confusion matrix:
def plot_confusion_matrix(cm, classes,
             normalize=False,
             title='Confusion matrix',
             cmap=plt.cm.Blues):
    #Add Normalization Option
    '''prints pretty confusion metric with normalization option '''
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

#     print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center", color="white" if cm[i,

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```python
plot_confusion_matrix(confusion_matrix, ['butterflies', 'chickens', 'elephants', 'horses', 'spider
```

```
Confusion matrix, without normalization
```



onfusion matrix, without normalization




```
#Those numbers are all over the place. Now turning normalize= True
plot_confusion_matrix(confusion_matrix,
                      ['butterflies', 'chickens', 'elephants', 'horses', 'spiders', 'squirells'],
                     normalize=True)
```

ormalized confusion matrix