In [1]:

```
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
import os
```

In [2]:

```
img_width, img_height = 224, 224

top_model_weights_path = 'birds_fc_model.h5'
from google.colab import drive
drive.mount('/content/drive')
base_dir = '/content/drive/MyDrive/birds/'
train_data_dir = os.path.join(base_dir, 'train')
validation_data_dir = os.path.join(base_dir, 'validation')
test_data_dir = os.path.join(base_dir, 'test')

epoch =10
batch_size = 1
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount
("/content/drive", force_remount=True).
```

In [3]:

```
vgg16 = applications.VGG16(include_top=False, weights='imagenet')
datagen = ImageDataGenerator(rescale=1. / 255)
```

In [4]:

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

```
Found 219 images belonging to 10 classes.
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1905: U
serWarning: `Model.predict_generator` is deprecated and will be removed in a future versi
on. Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
```

```
Time:  0:00:04.383053
```

In [5]:

```
start = datetime.datetime.now()

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

predict_size_test = int(math.ceil(nb_test_samples / batch_size))

bottleneck_features_test = vgg16.predict_generator(generator, predict_size_train)

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

```
Found 29 images belonging to 10 classes.
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1905: U
serWarning: `Model.predict_generator` is deprecated and will be removed in a future versi
on. Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 219 batches). You may need to use the repeat() function when building your dataset.
Time:  0:00:00.381778
```

In [6]:

```
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)
num_classes = len(generator.class_indices)

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

bottleneck_features_validation = vgg16.predict_generator(generator, predict_size_validati
on)

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

```
Found 69 images belonging to 10 classes.
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1905: U
serWarning: `Model.predict_generator` is deprecated and will be removed in a future versi
on. Please use `Model.predict`, which supports generators.
```

```
  warnings.warn('`Model.predict_generator` is deprecated and '
```

```
Time:  0:00:00.733668
```

In [7]:

```
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)

train_data = np.load('birds_features_train.npy')

train_labels = generator_top.classes

train_labels = to_categorical(train_labels, num_classes=num_classes)
```

```
Found 219 images belonging to 10 classes.
```

In [8]:

```
generator_top = datagen.flow_from_directory(
   test_data_dir,
   target_size=(img_width, img_height),
   batch_size=batch_size,
   class_mode='categorical',
   shuffle=False)

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

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

test_labels = generator_top.classes

test_labels = to_categorical(test_labels, num_classes=num_classes)
```

```
Found 29 images belonging to 10 classes.
```

In [9]:

```
generator_top = datagen.flow_from_directory(
   validation_data_dir,
   target_size=(img_width, img_height),
   batch_size=batch_size,
   class_mode='categorical',
   shuffle=False)

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

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

validation_labels = generator_top.classes

validation_labels = to_categorical(validation_labels, num_classes=num_classes)
```

```
Found 69 images belonging to 10 classes.
```

In [10]:

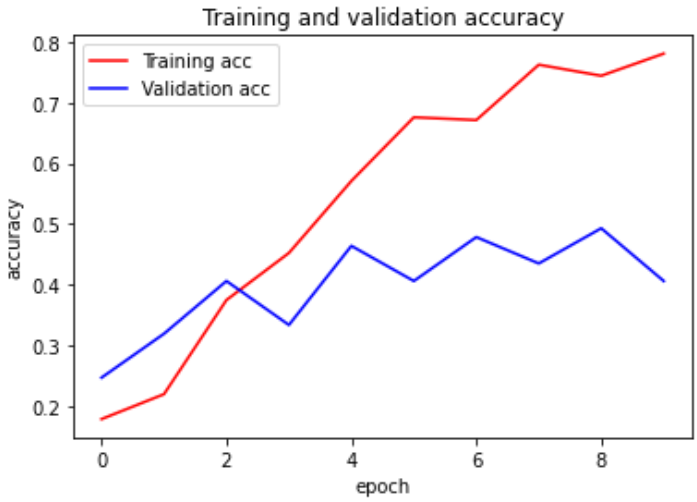
```
start = datetime.datetime.now()
model = Sequential()
model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.5))
```

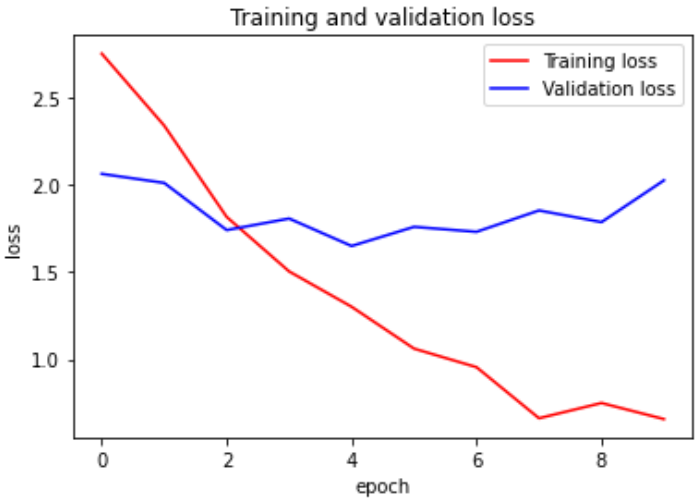
```
model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy',
   optimizer=optimizers.RMSprop(lr=1e-4),
   metrics=['acc'])
history = model.fit(train_data, train_labels,
   epochs=10,
   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)

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

```
Epoch 1/10
219/219 [==============================] - 2s 5ms/step - loss: 2.9156 - acc: 0.1286 - val
_loss: 2.0625 - val_acc: 0.2464
Epoch 2/10
219/219 [==============================] - 1s 3ms/step - loss: 2.4023 - acc: 0.2560 - val
_loss: 2.0105 - val_acc: 0.3188
Epoch 3/10
219/219 [==============================] - 1s 3ms/step - loss: 1.8834 - acc: 0.3312 - val
_loss: 1.7404 - val_acc: 0.4058
Epoch 4/10
219/219 [==============================] - 1s 4ms/step - loss: 1.5667 - acc: 0.4311 - val
_loss: 1.8064 - val_acc: 0.3333
Epoch 5/10
219/219 [==============================] - 1s 3ms/step - loss: 1.4071 - acc: 0.5711 - val
_loss: 1.6492 - val_acc: 0.4638
Epoch 6/10
219/219 [==============================] - 1s 4ms/step - loss: 1.0131 - acc: 0.7046 - val
_loss: 1.7589 - val_acc: 0.4058
Epoch 7/10
219/219 [==============================] - 1s 3ms/step - loss: 0.8614 - acc: 0.7242 - val
_loss: 1.7304 - val_acc: 0.4783
Epoch 8/10
219/219 [==============================] - 1s 3ms/step - loss: 0.7270 - acc: 0.7563 - val
_loss: 1.8529 - val_acc: 0.4348
Epoch 9/10
219/219 [==============================] - 1s 4ms/step - loss: 0.6127 - acc: 0.7517 - val
_loss: 1.7859 - val_acc: 0.4928
Epoch 10/10
219/219 [==============================] - 1s 4ms/step - loss: 0.5061 - acc: 0.8327 - val
_loss: 2.0252 - val_acc: 0.4058
69/69 [==============================] - 0s 2ms/step - loss: 2.0252 - acc: 0.4058
Time:  0:00:08.746808
```

In [11]:

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


Training and validation accuracy
Training acc
Validation acc
accuracy
epoch
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8


Training and validation loss
Training loss
Validation loss
loss
epoch
2.5
2.0
1.5
1.0
0
2
4
6
8

In [11]: