In [1]:

```
import os
import glob
import h5py
import shutil
import imgaug as aug
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mimg
import imgaug.augmenters as iaa
from os import listdir, makedirs, getcwd, remove
from os.path import isfile, join, abspath, exists, isdir, expanduser
from PIL import Image
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from keras.models import Sequential, Model
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.preprocessing.image import ImageDataGenerator,load_img, img_to_array
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Input, Flatten, SeparableConv2D
from keras.layers import GlobalMaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.merge import Concatenate
from keras.models import Model
from keras.optimizers import Adam, SGD, RMSprop
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
import cv2
import tensorflow as tf
from keras import backend as K
color = sns.color_palette()
%matplotlib inline
```

```
Using TensorFlow backend.
```

In [2]:

```
import tensorflow as tf

# Set the seed for hash based operations in python
os.environ['PYTHONHASHSEED'] = '0'

# Set the numpy seed
np.random.seed(111)

# Disable multi-threading in tensorflow ops
session_conf = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=1, inter_op_parallelism_threads=1)

# Set the random seed in tensorflow at graph level
tf.random.set_seed(111)



# Make the augmentation sequence deterministic
aug.seed(111)
```

In [3]:

```
# Define path to the data directory
data_dir = Path('chest_xray')

# Path to train directory (Fancy pathlib...no more os.path!!)
train_dir = data_dir / 'train'

# Path to test directory
test_dir = data_dir / 'test'
```

In [4]:

```
# Get the path to the normal and pneumonia sub-directories
normal_cases_dir = train_dir / 'NORMAL'
pneumonia_cases_dir = train_dir / 'PNEUMONIA'

# Get the list of all the images
normal_cases = normal_cases_dir.glob('*.*g')
#normal_cases.extend('*.png')
#normal_cases.extend('*.jpg')
pneumonia_cases = pneumonia_cases_dir.glob('*.*g')
#pneumonia_cases = pneumonia_cases_dir.glob('*.jpg')
#pneumonia_cases = pneumonia_cases_dir.glob('*.png')

print(normal_cases)
# An empty list. We will insert the data into this list in (img_path, label) format
train_data = []

# Go through all the normal cases. The label for these cases will be 0
for img in normal_cases:
    train_data.append((img,0))

# Go through all the pneumonia cases. The label for these cases will be 1
for img in pneumonia_cases:
    train_data.append((img, 1))

# Get a pandas dataframe from the data we have in our list
train_data = pd.DataFrame(train_data, columns=['image', 'label'],index=None)

# Shuffle the data
train_data = train_data.sample(frac=1.).reset_index(drop=True)
# How the dataframe looks like?
train_data.head()
```

```
<generator object Path.glob at 0x7efd9af2dad0>
```

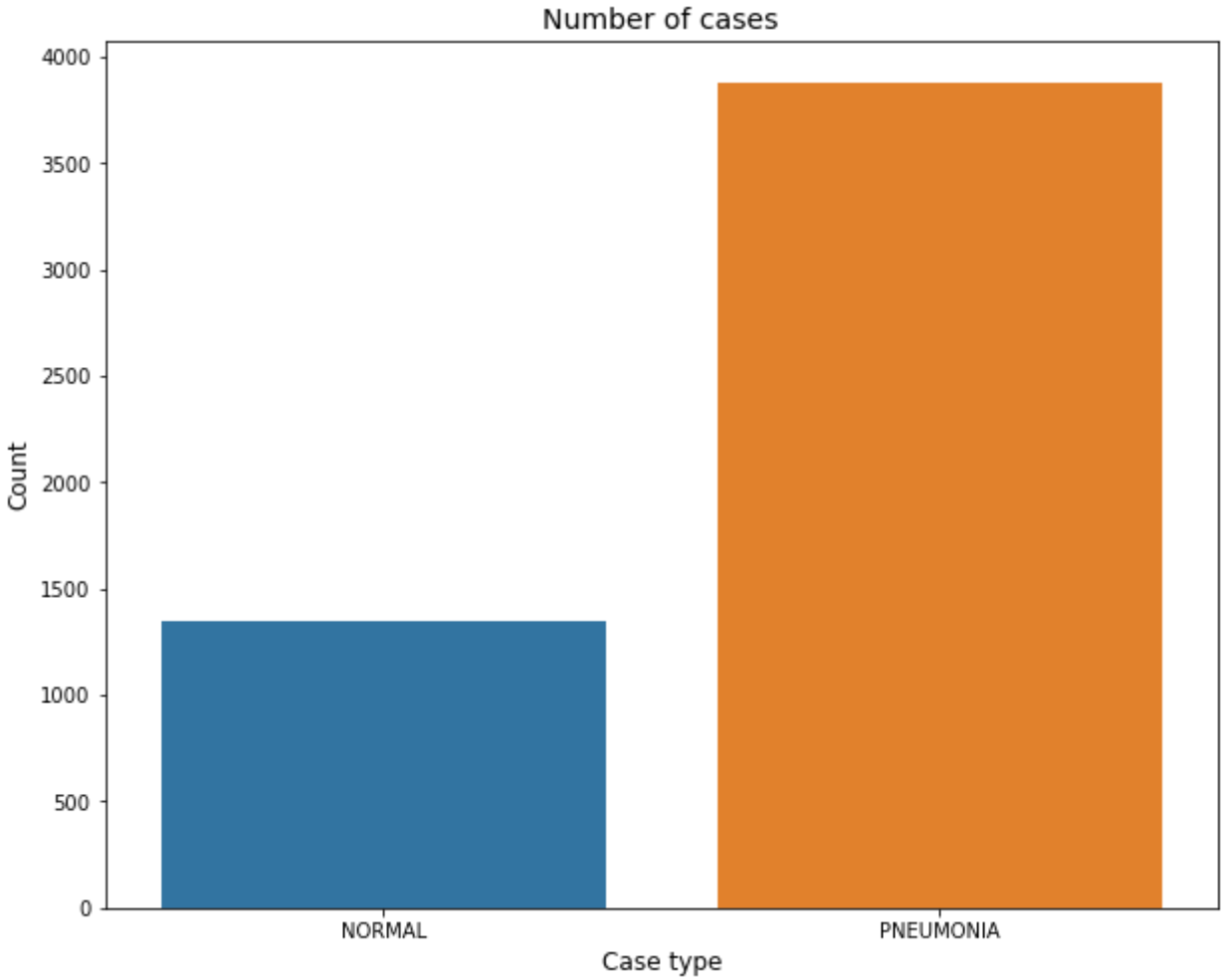
Out[4]:

	image	label
0	chest_xray/train/PNEUMONIA/BACTERIA-3134196-00...	1
1	chest_xray/train/PNEUMONIA/BACTERIA-1083680-00...	1
2	chest_xray/train/PNEUMONIA/BACTERIA-1950119-00...	1
3	chest_xray/train/PNEUMONIA/VIRUS-8028911-0002....	1
4	chest_xray/train/PNEUMONIA/VIRUS-5331068-0002....	1

In [5]:

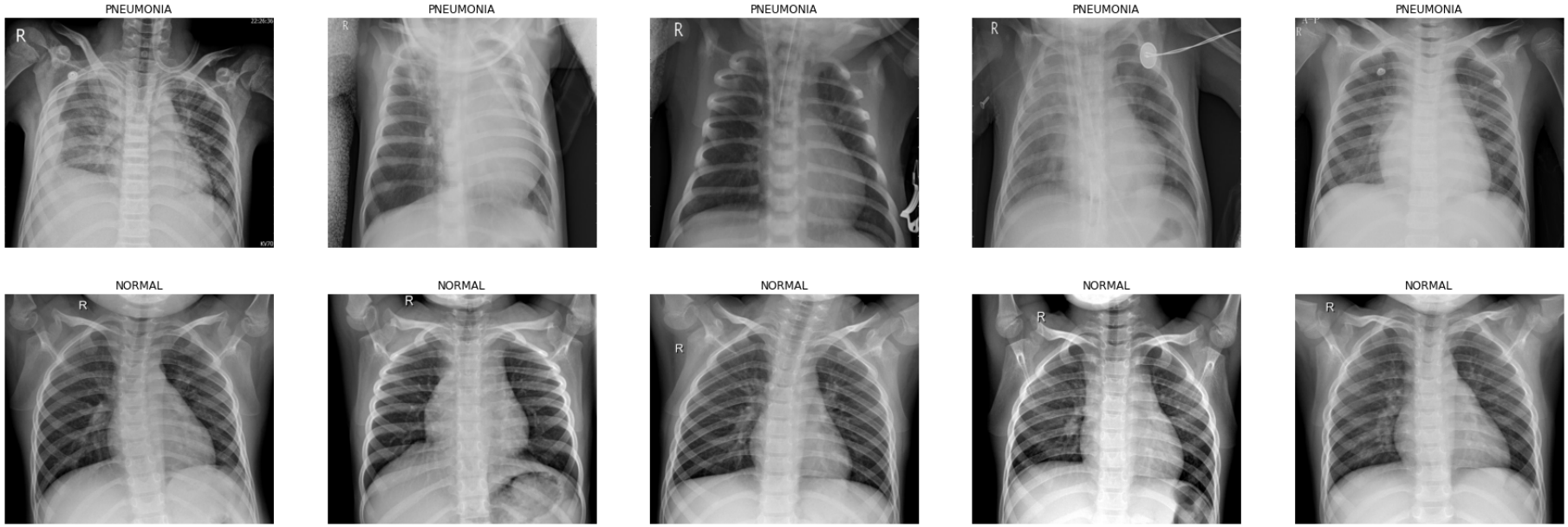
```
# Get the counts for each class
cases_count = train_data['label'].value_counts()
print(cases_count)

# Plot the results
plt.figure(figsize=(10,8))
sns.barplot(x=cases_count.index, y= cases_count.values)
plt.title('Number of cases', fontsize=14)
plt.xlabel('Case type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(range(len(cases_count.index)), ['NORMAL', 'PNEUMONIA'])
plt.show()
```

```
1    3883
0    1349
Name: label, dtype: int64
```


Number of cases
Count
4000
3500
3000
2500
2000
1500
1000
500
0
NORMAL
PNEUMONIA
Case type

In [6]:

```
pneumonia_samples = (train_data[train_data['label']==1]['image'].iloc[:5]).tolist()
normal_samples = (train_data[train_data['label']==0]['image'].iloc[:5]).tolist()

# Concat the data in a single list and del the above two list
samples = pneumonia_samples + normal_samples
del pneumonia_samples, normal_samples

# Plot the data
f, ax = plt.subplots(2,5, figsize=(30,10))
for i in range(10):
    img = imread(samples[i])
    ax[i//5, i%5].imshow(img, cmap='gray')
    if i<5:
        ax[i//5, i%5].set_title("PNEUMONIA")
    else:
        ax[i//5, i%5].set_title("NORMAL")
    ax[i//5, i%5].axis('off')
    ax[i//5, i%5].set_aspect('auto')
plt.show()
```



In [ ]:

```

```

In [7]:

```
# Get the path to the sub-directories
normal_cases_dir = test_dir / 'NORMAL'
pneumonia_cases_dir = test_dir / 'PNEUMONIA'

# Get the list of all the images
normal_cases = normal_cases_dir.glob('*.*g')
pneumonia_cases = pneumonia_cases_dir.glob('*.*g')

# List that are going to contain validation images data and the corresponding labels
valid_data = []
valid_labels = []


# Some images are in grayscale while majority of them contains 3 channels. So, if the image is grayscale, we will convert into a i
mage with 3 channels.
# We will normalize the pixel values and resizing all the images to 224x224

# Normal cases
for img in normal_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (224,224))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = to_categorical(0, num_classes=2)
    valid_data.append(img)
    valid_labels.append(label)

# Pneumonia cases
for img in pneumonia_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (224,224))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = to_categorical(1, num_classes=2)
    valid_data.append(img)
```

```
    valid_labels.append(label)

# Convert the list into numpy arrays
valid_data = np.array(valid_data)
valid_labels = np.array(valid_labels)

print("Total number of validation examples: ", valid_data.shape)
print("Total number of labels:", valid_labels.shape)
```

```
Total number of validation examples:  (624, 224, 224, 3)
Total number of labels: (624, 2)
```

In [8]:

```
# Augmentation sequence
seq = iaa.OneOf([
    iaa.Fliplr(), # horizontal flips
    iaa.Affine(rotate=40), # roatation
    iaa.Multiply((1.2, 1.5))]) #random brightness
```

In [9]:

```
def data_gen(data, batch_size):
    # Get total number of samples in the data
    n = len(data)
    steps = n//batch_size

    # Define two numpy arrays for containing batch data and labels
    batch_data = np.zeros((batch_size, 224, 224, 3), dtype=np.float32)
    batch_labels = np.zeros((batch_size,2), dtype=np.float32)

    # Get a numpy array of all the indices of the input data
    indices = np.arange(n)

    # Initialize a counter
    i =0
    while True:
        np.random.shuffle(indices)
        # Get the next batch
        count = 0
        next_batch = indices[(i*batch_size):(i+1)*batch_size]
        for j, idx in enumerate(next_batch):
            img_name = data.iloc[idx]['image']
            label = data.iloc[idx]['label']

            # one hot encoding
            encoded_label = to_categorical(label, num_classes=2)
            # read the image and resize
            img = cv2.imread(str(img_name))
            img = cv2.resize(img, (224,224))

            # check if it's grayscale
            if img.shape[2]==1:
                img = np.dstack([img, img, img])

            # cv2 reads in BGR mode by default
            orig_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            # normalize the image pixels
            orig_img = img.astype(np.float32)/255.

            batch_data[count] = orig_img
```

```
            batch_labels[count] = encoded_label

            # generating more samples of the undersampled class
            if label==0 and count < batch_size-2:
                aug_img1 = seq.augment_image(img)
                aug_img2 = seq.augment_image(img)
                aug_img1 = cv2.cvtColor(aug_img1, cv2.COLOR_BGR2RGB)
                aug_img2 = cv2.cvtColor(aug_img2, cv2.COLOR_BGR2RGB)
                aug_img1 = aug_img1.astype(np.float32)/255.
                aug_img2 = aug_img2.astype(np.float32)/255.

                batch_data[count+1] = aug_img1
                batch_labels[count+1] = encoded_label
                batch_data[count+2] = aug_img2
                batch_labels[count+2] = encoded_label
                count +=2

            else:
                count+=1

            if count==batch_size-1:
                break

        i+=1
        yield batch_data, batch_labels

        if i>=steps:
            i=0
```

In [10]:

```
def build_model():
    input_img = Input(shape=(224,224,3), name='ImageInput')
    x = Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_1')(input_img)
    x = Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_2')(x)
    x = MaxPooling2D((2,2), name='pool1')(x)

    x = SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_1')(x)
    x = SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_2')(x)
    x = MaxPooling2D((2,2), name='pool2')(x)

    x = SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_1')(x)
    x = BatchNormalization(name='bn1')(x)
    x = SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_2')(x)
    x = BatchNormalization(name='bn2')(x)
    x = SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_3')(x)
    x = MaxPooling2D((2,2), name='pool3')(x)

    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv4_1')(x)
    x = BatchNormalization(name='bn3')(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv4_2')(x)
    x = BatchNormalization(name='bn4')(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv4_3')(x)
    x = MaxPooling2D((2,2), name='pool4')(x)

    x = Flatten(name='flatten')(x)
    x = Dense(1024, activation='relu', name='fc1')(x)
    x = Dropout(0.3, name='dropout1')(x)
    x = Dense(512, activation='relu', name='fc2')(x)
    x = Dropout(0.2, name='dropout2')(x)
    x = Dense(2, activation='softmax', name='fc3')(x)

    model = Model(inputs=input_img, outputs=x)
    return model
```

In [11]:

```
def build_modelSOTA():
    input_img = Input(shape=(224,224,3), name='ImageInput')
    x = Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_1')(input_img)
    x = Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_2')(x)
    x = MaxPooling2D((2,2), name='pool1')(x)

    x = SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_1')(x)
    x = SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_2')(x)
    x = MaxPooling2D((2,2), name='pool2')(x)

    x = SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_1')(x)
    x = SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_2')(x)
    x = SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_3')(x)
    x = MaxPooling2D((2,2), name='pool3')(x)

    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv4_1')(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv4_2')(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv4_3')(x)
    x = MaxPooling2D((2,2), name='pool4')(x)

    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv5_1')(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv5_2')(x)
    x = SeparableConv2D(512, (3,3), activation='relu', padding='same', name='Conv5_3')(x)
    x = MaxPooling2D((2,2), name='pool5')(x)

    x = Flatten(name='flatten')(x)
    x = Dense(512, activation='relu', name='fc1')(x)
    x = BatchNormalization(name='bn1')(x)
    x = Dense(512, activation='relu', name='fc2')(x)
    x = BatchNormalization(name='bn2')(x)
    x = Dropout(0.5, name='dropout1')(x)
    x = Dense(2, activation='softmax', name='fc3')(x)

    model = Model(inputs=input_img, outputs=x)
    return model
```

In [12]:

```
model =  build_modelSOTA()
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
ImageInput (InputLayer)      (None, 224, 224, 3)       0
_________________________________________________________________
Conv1_1 (Conv2D)             (None, 224, 224, 64)      1792
_________________________________________________________________
Conv1_2 (Conv2D)             (None, 224, 224, 64)      36928
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 112, 112, 64)      0
_________________________________________________________________
Conv2_1 (SeparableConv2D)    (None, 112, 112, 128)     8896
_________________________________________________________________
Conv2_2 (SeparableConv2D)    (None, 112, 112, 128)     17664
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 56, 56, 128)       0
_________________________________________________________________
Conv3_1 (SeparableConv2D)    (None, 56, 56, 256)       34176
_________________________________________________________________
Conv3_2 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
Conv3_3 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 28, 28, 256)       0
_________________________________________________________________
Conv4_1 (SeparableConv2D)    (None, 28, 28, 512)       133888
_________________________________________________________________
Conv4_2 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
Conv4_3 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 14, 14, 512)       0
_________________________________________________________________
Conv5_1 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_2 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_3 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 7, 7, 512)         0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 512)               12845568
_________________________________________________________________
bn1 (BatchNormalization)     (None, 512)               2048
_________________________________________________________________
fc2 (Dense)                  (None, 512)               262656
_________________________________________________________________
bn2 (BatchNormalization)     (None, 512)               2048
_________________________________________________________________
dropout1 (Dropout)           (None, 512)               0
_________________________________________________________________
fc3 (Dense)                  (None, 2)                 1026
=================================================================
Total params: 14,819,202
Trainable params: 14,817,154
Non-trainable params: 2,048
_________________________________________________________________
```

In [13]:

```
model.layers
print(len(model.layers))
```

26

In [14]:

```
# Open the VGG16 weight file
f = h5py.File('vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5', 'r')

# Select the layers for which you want to set weight.

w,b = f['block1_conv1']['block1_conv1_W_1:0'], f['block1_conv1']['block1_conv1_b_1:0']
model.layers[1].set_weights = [w,b]

w,b = f['block1_conv2']['block1_conv2_W_1:0'], f['block1_conv2']['block1_conv2_b_1:0']
model.layers[2].set_weights = [w,b]

w,b = f['block2_conv1']['block2_conv1_W_1:0'], f['block2_conv1']['block2_conv1_b_1:0']
model.layers[4].set_weights = [w,b]

w,b = f['block2_conv2']['block2_conv2_W_1:0'], f['block2_conv2']['block2_conv2_b_1:0']
model.layers[5].set_weights = [w,b]

f.close()
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
ImageInput (InputLayer)      (None, 224, 224, 3)       0
_________________________________________________________________
Conv1_1 (Conv2D)             (None, 224, 224, 64)      1792
_________________________________________________________________
Conv1_2 (Conv2D)             (None, 224, 224, 64)      36928
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 112, 112, 64)      0
_________________________________________________________________
Conv2_1 (SeparableConv2D)    (None, 112, 112, 128)     8896
_________________________________________________________________
Conv2_2 (SeparableConv2D)    (None, 112, 112, 128)     17664
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 56, 56, 128)       0
_________________________________________________________________
Conv3_1 (SeparableConv2D)    (None, 56, 56, 256)       34176
_________________________________________________________________
Conv3_2 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
Conv3_3 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 28, 28, 256)       0
_________________________________________________________________
Conv4_1 (SeparableConv2D)    (None, 28, 28, 512)       133888
_________________________________________________________________
Conv4_2 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
Conv4_3 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 14, 14, 512)       0
_________________________________________________________________
Conv5_1 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_2 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_3 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 7, 7, 512)         0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 512)               12845568
_________________________________________________________________
bn1 (BatchNormalization)     (None, 512)               2048
_________________________________________________________________
fc2 (Dense)                  (None, 512)               262656
_________________________________________________________________
bn2 (BatchNormalization)     (None, 512)               2048
_________________________________________________________________
dropout1 (Dropout)           (None, 512)               0
_________________________________________________________________
fc3 (Dense)                  (None, 2)                 1026
=================================================================
Total params: 14,819,202
Trainable params: 14,817,154
Non-trainable params: 2,048
_________________________________________________________________
```

In [15]:

```
#opt = RMSprop(lr=0.0001, decay=1e-6)
opt = RMSprop(lr=1e-4, decay=0.9) # SOTA
#opt = Adam(lr=0.0001, decay=1e-5)
#opt = Adam(lr=0.0001, decay=1e-5)
es = EarlyStopping(patience=15)
chkpt = ModelCheckpoint(filepath='best_modelvgg.hdf5', save_best_only=True, save_weights_only=True)
model.compile(loss='binary_crossentropy', metrics=['accuracy'],optimizer=opt)
```

In [16]:

```
batch_size = 16
nb_epochs = 50

# Get a train data generator
train_data_gen = data_gen(data=train_data, batch_size=batch_size)

# Define the number of training steps
nb_train_steps = train_data.shape[0]//batch_size

print("Number of training and validation steps: {} and {}".format(nb_train_steps, len(valid_data)))
```

```
Number of training and validation steps: 327 and 624
```

In [17]:

```
# # Fit the model
history = model.fit_generator(train_data_gen, epochs=nb_epochs, steps_per_epoch=nb_train_steps,
                              validation_data=(valid_data, valid_labels), callbacks=[chkpt],
                              class_weight={0:1.0, 1:0.4})
```

```
Epoch 1/50
327/327 [==============================] - 85s 259ms/step - loss: 0.4597 - accuracy: 0.4424 - val_loss: 0.6954 - val_
accuracy: 0.3750
Epoch 2/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4525 - accuracy: 0.4256 - val_loss: 0.6956 - val_
accuracy: 0.3750
Epoch 3/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4585 - accuracy: 0.4407 - val_loss: 0.6957 - val_
accuracy: 0.3750
Epoch 4/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4572 - accuracy: 0.4379 - val_loss: 0.6959 - val_
accuracy: 0.3750
Epoch 5/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4585 - accuracy: 0.4411 - val_loss: 0.6961 - val_
accuracy: 0.3750
Epoch 6/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4578 - accuracy: 0.4396 - val_loss: 0.6959 - val_
accuracy: 0.3750
Epoch 7/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4543 - accuracy: 0.4310 - val_loss: 0.6959 - val_
accuracy: 0.3750
Epoch 8/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4580 - accuracy: 0.4404 - val_loss: 0.6961 - val_
accuracy: 0.3750
Epoch 9/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4531 - accuracy: 0.4283 - val_loss: 0.6961 - val_
accuracy: 0.3750
Epoch 10/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4544 - accuracy: 0.4316 - val_loss: 0.6961 - val_
accuracy: 0.3750
Epoch 11/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4626 - accuracy: 0.4522 - val_loss: 0.6961 - val_
accuracy: 0.3750
Epoch 12/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4600 - accuracy: 0.4457 - val_loss: 0.6962 - val_
accuracy: 0.3750
Epoch 13/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4592 - accuracy: 0.4438 - val_loss: 0.6963 - val_
accuracy: 0.3750
Epoch 14/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4559 - accuracy: 0.4356 - val_loss: 0.6963 - val_
```

```
accuracy: 0.3750
Epoch 15/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4574 - accuracy: 0.4394 - val_loss: 0.6962 - val_
accuracy: 0.3750
Epoch 16/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4608 - accuracy: 0.4478 - val_loss: 0.6963 - val_
accuracy: 0.3750
Epoch 17/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4535 - accuracy: 0.4297 - val_loss: 0.6962 - val_
accuracy: 0.3750
Epoch 18/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4543 - accuracy: 0.4316 - val_loss: 0.6964 - val_
accuracy: 0.3750
Epoch 19/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4588 - accuracy: 0.4430 - val_loss: 0.6964 - val_
accuracy: 0.3750
Epoch 20/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4570 - accuracy: 0.4385 - val_loss: 0.6964 - val_
accuracy: 0.3750
Epoch 21/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4508 - accuracy: 0.4230 - val_loss: 0.6964 - val_
accuracy: 0.3750
Epoch 22/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4547 - accuracy: 0.4327 - val_loss: 0.6964 - val_
accuracy: 0.3750
Epoch 23/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4579 - accuracy: 0.4407 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 24/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4546 - accuracy: 0.4325 - val_loss: 0.6964 - val_
accuracy: 0.3750
Epoch 25/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4622 - accuracy: 0.4516 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 26/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4622 - accuracy: 0.4516 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 27/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4582 - accuracy: 0.4417 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 28/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4545 - accuracy: 0.4325 - val_loss: 0.6965 - val_
```

```
accuracy: 0.3750
Epoch 29/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4573 - accuracy: 0.4394 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 30/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4562 - accuracy: 0.4367 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 31/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4585 - accuracy: 0.4425 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 32/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4567 - accuracy: 0.4381 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 33/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4553 - accuracy: 0.4346 - val_loss: 0.6965 - val_
accuracy: 0.3750
Epoch 34/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4565 - accuracy: 0.4375 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 35/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4585 - accuracy: 0.4427 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 36/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4607 - accuracy: 0.4482 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 37/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4542 - accuracy: 0.4320 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 38/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4592 - accuracy: 0.4446 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 39/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4566 - accuracy: 0.4381 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 40/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4559 - accuracy: 0.4362 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 41/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4538 - accuracy: 0.4310 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 42/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4533 - accuracy: 0.4297 - val_loss: 0.6966 - val_
```

```
accuracy: 0.3750
Epoch 43/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4574 - accuracy: 0.4402 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 44/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4550 - accuracy: 0.4341 - val_loss: 0.6967 - val_
accuracy: 0.3750
Epoch 45/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4600 - accuracy: 0.4465 - val_loss: 0.6967 - val_
accuracy: 0.3750
Epoch 46/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4623 - accuracy: 0.4524 - val_loss: 0.6967 - val_
accuracy: 0.3750
Epoch 47/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4518 - accuracy: 0.4258 - val_loss: 0.6966 - val_
accuracy: 0.3750
Epoch 48/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4586 - accuracy: 0.4430 - val_loss: 0.6967 - val_
accuracy: 0.3750
Epoch 49/50
327/327 [==============================] - 82s 250ms/step - loss: 0.4587 - accuracy: 0.4434 - val_loss: 0.6967 - val_
accuracy: 0.3750
Epoch 50/50
327/327 [==============================] - 82s 251ms/step - loss: 0.4571 - accuracy: 0.4394 - val_loss: 0.6967 - val_
accuracy: 0.3750
```

In [18]:

```
#print(history.history)
```

In [19]:

```
#print(history)
```

In [20]:

```
def showGraph(Histroy, epochs):
    # plot the training loss and accuracy
    plt.style.use("ggplot")
    plt.figure()
    plt.plot(np.arange(0, epochs), Histroy.history["loss"], label="train_loss")
    plt.plot(np.arange(0, epochs), Histroy.history["val_loss"], label="val_loss")
    plt.plot(np.arange(0, epochs), Histroy.history["accuracy"], label="train_acc")
    plt.plot(np.arange(0, epochs), Histroy.history["val_accuracy"], label="val_acc")
    plt.title("Training Loss and Accuracy")
    plt.xlabel("Epoch #")
    plt.ylabel("Loss/Accuracy")
    plt.legend()
    plt.show()
```

In [21]:

```
showGraph(history, nb_epochs)
```

In [22]:

```
# # Visualize training history
# from keras.models import Sequential
# from keras.layers import Dense
# import matplotlib.pyplot as plt
# import numpy


# # summarize history for accuracy

# plt.plot(history.history['val_accuracy'])
# plt.plot(history.history['accuracy'])
# plt.title('model accuracy')
# plt.ylabel('accuracy')
# plt.xlabel('epoch')
# plt.legend(['train', 'test'], loc='upper left')
# plt.show()
# # summarize history for loss
# plt.plot(history.history['loss'])
# plt.plot(history.history['val_loss'])
# plt.title('model loss')
# plt.ylabel('loss')
# plt.xlabel('epoch')
# plt.legend(['train', 'test'], loc='upper left')
# plt.show()


# # summarize history for loss
# plt.plot(history.history['loss'])
# plt.plot(history.history['val_loss'])
# plt.plot(history.history['val_accuracy'])
# plt.plot(history.history['accuracy'])
# plt.legend(['loss', 'val_loss','val_accuracy','accuracy' ], loc='upper left')
# plt.show()
```

In [23]:

```
# Load the model weights
model.load_weights("best_modelvgg.hdf5")
```

In [24]:

```
# Preparing test data
normal_cases_dir = test_dir / 'NORMAL'
pneumonia_cases_dir = test_dir / 'PNEUMONIA'

normal_cases = normal_cases_dir.glob('*.*g')
pneumonia_cases = pneumonia_cases_dir.glob('*.*g')

test_data = []
test_labels = []

for img in normal_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (224,224))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    else:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = to_categorical(0, num_classes=2)
    test_data.append(img)
    test_labels.append(label)

for img in pneumonia_cases:
    img = cv2.imread(str(img))
    img = cv2.resize(img, (224,224))
    if img.shape[2] ==1:
        img = np.dstack([img, img, img])
    else:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)/255.
    label = to_categorical(1, num_classes=2)
    test_data.append(img)
    test_labels.append(label)


test_data = np.array(test_data)
test_labels = np.array(test_labels)
```

```
print("Total number of test examples: ", test_data.shape)
print("Total number of labels:", test_labels.shape)
```

```
Total number of test examples:  (624, 224, 224, 3)
Total number of labels: (624, 2)
```

In [25]:

```
# Evaluation on test dataset
test_loss, test_score = model.evaluate(test_data, test_labels, batch_size=16)
print("Loss on test set: ", test_loss)
print("Accuracy on test set: ", test_score)
```

```
624/624 [==============================] - 2s 3ms/step
Loss on test set:  0.6953692069420447
Accuracy on test set:  0.375
```

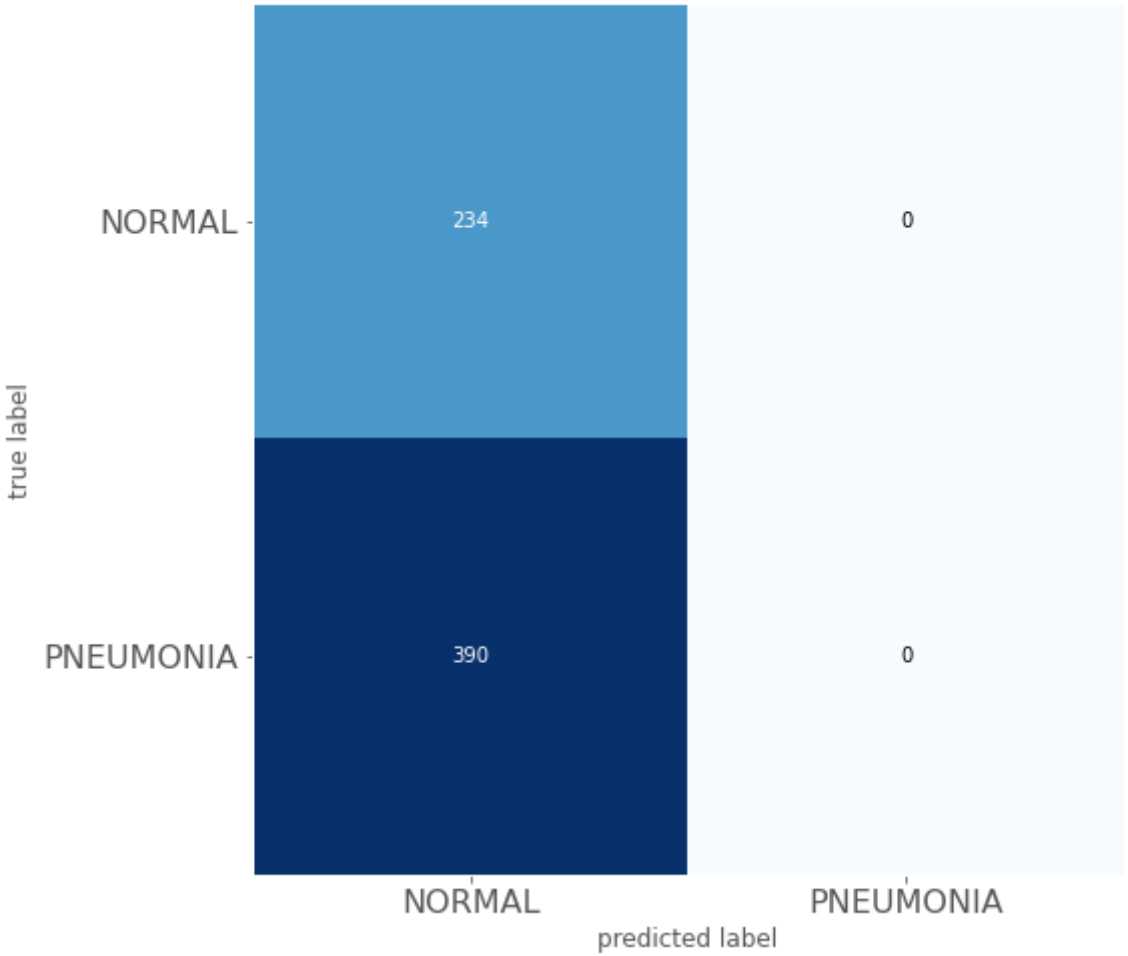
In [26]:

```
# Get the predictions on test set
preds = model.predict(test_data, batch_size=16)
preds = np.squeeze((preds > 0.5).astype('int'))
orig = test_labels.astype('int')
#print(preds)
#print(orig)

# Get predictions
preds = model.predict(test_data, batch_size=16)
preds = np.argmax(preds, axis=-1)

# Original labels
orig = np.argmax(test_labels, axis=-1)

#print(orig)
#print(preds)
```

In [27]:

```
# Get the confusion matrix
cm  = confusion_matrix(orig, preds)
plt.figure()
plot_confusion_matrix
plot_confusion_matrix(cm,figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.xticks(range(2), ['NORMAL', 'PNEUMONIA'], fontsize=16)
plt.yticks(range(2), ['NORMAL', 'PNEUMONIA'], fontsize=16)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [28]:

```
# Calculate Precision and Recall
tn, fp, fn, tp = cm.ravel()

precision = tp/(tp+fp)
recall = tp/(tp+fn)


print("Recall of the model is {:.2f}".format(recall))
print("Precision of the model is {:.2f}".format(precision))
```

```
Recall of the model is 0.00
Precision of the model is nan

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:4: RuntimeWarning: invalid value encountered in long_sca
lars
  after removing the cwd from sys.path.
```

In [29]:

```
model.save("SOTA_V_SOTA_STRUCTURE.h5")
```

Fine tuning

In [28]:

```
def showGraph(Histroy, epochs):
    # plot the training loss and accuracy
    plt.style.use("ggplot")
    plt.figure()
    plt.plot(np.arange(0, epochs), Histroy.history["loss"], label="train_loss")
    plt.plot(np.arange(0, epochs), Histroy.history["val_loss"], label="val_loss")
    plt.plot(np.arange(0, epochs), Histroy.history["accuracy"], label="train_acc")
    plt.plot(np.arange(0, epochs), Histroy.history["val_accuracy"], label="val_acc")
    plt.title("Training Loss and Accuracy")
    plt.xlabel("Epoch #")
    plt.ylabel("Loss/Accuracy")
    plt.legend()
    plt.show()
```

Step 1 & 2: # Freezing all the layers & added a new fully connected layer

In [82]:

```
model = build_model()
# Open the VGG16 weight file
f = h5py.File('vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5', 'r')

# Select the layers for which you want to set weight.

w,b = f['block1_conv1']['block1_conv1_W_1:0'], f['block1_conv1']['block1_conv1_b_1:0']
model.layers[1].set_weights = [w,b]

w,b = f['block1_conv2']['block1_conv2_W_1:0'], f['block1_conv2']['block1_conv2_b_1:0']
model.layers[2].set_weights = [w,b]

w,b = f['block2_conv1']['block2_conv1_W_1:0'], f['block2_conv1']['block2_conv1_b_1:0']
model.layers[4].set_weights = [w,b]

w,b = f['block2_conv2']['block2_conv2_W_1:0'], f['block2_conv2']['block2_conv2_b_1:0']
model.layers[5].set_weights = [w,b]

f.close()
model.summary()
model.trainable = False
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
ImageInput (InputLayer)      (None, 224, 224, 3)       0
_________________________________________________________________
Conv1_1 (Conv2D)             (None, 224, 224, 64)      1792
_________________________________________________________________
Conv1_2 (Conv2D)             (None, 224, 224, 64)      36928
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 112, 112, 64)      0
_________________________________________________________________
Conv2_1 (SeparableConv2D)    (None, 112, 112, 128)     8896
_________________________________________________________________
Conv2_2 (SeparableConv2D)    (None, 112, 112, 128)     17664
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 56, 56, 128)       0
_________________________________________________________________
Conv3_1 (SeparableConv2D)    (None, 56, 56, 256)       34176
_________________________________________________________________
Conv3_2 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
Conv3_3 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 28, 28, 256)       0
_________________________________________________________________
Conv4_1 (SeparableConv2D)    (None, 28, 28, 512)       133888
_________________________________________________________________
Conv4_2 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
Conv4_3 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 14, 14, 512)       0
_________________________________________________________________
Conv5_1 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_2 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_3 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 7, 7, 512)         0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0         
_________________________________________________________________
fc1 (Dense)                  (None, 512)               12845568  
_________________________________________________________________
bn1 (BatchNormalization)     (None, 512)               2048      
_________________________________________________________________
fc2 (Dense)                  (None, 512)               262656    
_________________________________________________________________
bn2 (BatchNormalization)     (None, 512)               2048      
_________________________________________________________________
dropout2 (Dropout)           (None, 512)               0         
_________________________________________________________________
fc3 (Dense)                  (None, 2)                 1026      
=================================================================
Total params: 14,819,202
Trainable params: 14,817,154
Non-trainable params: 2,048
_________________________________________________________________
```

Step 3: Train the weights on the new FC layer.

In [30]:

```
opt = RMSprop(lr=1e-3, decay=0.9)
#opt = Adam(lr=0.0001, decay=1e-5)
es = EarlyStopping(patience=10)
#chkpt = ModelCheckpoint(filepath='best_modelvgg.hdf5', save_best_only=True, save_weights_only=True)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'],optimizer=opt)

batch_size = 16
nb_epochs = 3

# Get a train data generator
train_data_gen = data_gen(data=train_data, batch_size=batch_size)

# Define the number of training steps
nb_train_steps = train_data.shape[0]//batch_size

print("Number of training and validation steps: {} and {}".format(nb_train_steps, len(valid_data)))


# # Fit the model
checkpoint = tf.keras.callbacks.ModelCheckpoint("PreFineTunebestVGG_StateOfTheArtData.h5", monitor="val_loss", mode="min", save_be
st_only=True, verbose=1)
history = model.fit_generator(train_data_gen, epochs=nb_epochs, steps_per_epoch=nb_train_steps,
                              validation_data=(valid_data, valid_labels),
                              callbacks=[es, checkpoint],
                              class_weight={0:1.0, 1:0.4})
```

```
Number of training and validation steps: 8 and 170
Epoch 1/3
8/8 [==============================] - 33s 4s/step - loss: 0.5742 - accuracy: 0.7422 - val_loss: 0.6949 - val_accurac
y: 0.4647

Epoch 00001: val_loss improved from inf to 0.69494, saving model to PreFineTunebestVGG_StateOfTheArtData.h5
Epoch 2/3
8/8 [==============================] - 34s 4s/step - loss: 0.5654 - accuracy: 0.7266 - val_loss: 0.6951 - val_accurac
y: 0.4647

Epoch 00002: val_loss did not improve from 0.69494
Epoch 3/3
8/8 [==============================] - 33s 4s/step - loss: 0.5289 - accuracy: 0.6328 - val_loss: 0.6952 - val_accurac
y: 0.4647

Epoch 00003: val_loss did not improve from 0.69494
```

Step 4: Unfreeze the trainable weights on some of the convolutional layers in the base network.

In [33]:

```
model.trainable = True
set_trainable = False
for layer in model.layers:
    if layer.name in ['block5_conv1']:
            set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

In [48]:

```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
ImageInput (InputLayer)      (None, 224, 224, 3)       0
_________________________________________________________________
Conv1_1 (Conv2D)             (None, 224, 224, 64)      1792
_________________________________________________________________
Conv1_2 (Conv2D)             (None, 224, 224, 64)      36928
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 112, 112, 64)      0
_________________________________________________________________
Conv2_1 (SeparableConv2D)    (None, 112, 112, 128)     8896
_________________________________________________________________
Conv2_2 (SeparableConv2D)    (None, 112, 112, 128)     17664
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 56, 56, 128)       0
_________________________________________________________________
Conv3_1 (SeparableConv2D)    (None, 56, 56, 256)       34176
_________________________________________________________________
Conv3_2 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
Conv3_3 (SeparableConv2D)    (None, 56, 56, 256)       68096
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 28, 28, 256)       0
_________________________________________________________________
Conv4_1 (SeparableConv2D)    (None, 28, 28, 512)       133888
_________________________________________________________________
Conv4_2 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
Conv4_3 (SeparableConv2D)    (None, 28, 28, 512)       267264
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 14, 14, 512)       0
_________________________________________________________________
Conv5_1 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_2 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
Conv5_3 (SeparableConv2D)    (None, 14, 14, 512)       267264
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 7, 7, 512)         0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 512)               12845568
_________________________________________________________________
bn1 (BatchNormalization)     (None, 512)               2048
_________________________________________________________________
fc2 (Dense)                  (None, 512)               262656
_________________________________________________________________
bn2 (BatchNormalization)     (None, 512)               2048
_________________________________________________________________
dropout2 (Dropout)           (None, 512)               0
_________________________________________________________________
fc3 (Dense)                  (None, 2)                 1026
=================================================================
Total params: 29,636,356
Trainable params: 14,817,154
Non-trainable params: 14,819,202
_________________________________________________________________

C:\Users\StudyEasy\anaconda3\lib\site-packages\keras\engine\training.py:297: UserWarning: Discrepancy between trainab
le weights and collected trainable weights, did you set `model.trainable` without calling `model.compile` after ?
  'Discrepancy between trainable weights and collected trainable'
```

In [63]:

```
# baseModel = VGG16(weights="imagenet", include_top=False,
# input_tensor=Input(shape=(224, 224, 3)))


# headModel = baseModel.output
# headModel = Flatten(name="flatten")(headModel)
# headModel = Dense(512, activation="relu")(headModel)
# headModel = Dropout(0.5)(headModel)
# headModel = Dense(2, activation="softmax")(headModel)


# model = Model(inputs=baseModel.input, outputs=headModel)
```

In [64]:

```
opt = RMSprop(lr=1e-4, decay=0.9)
#opt = Adam(lr=0.0001, decay=1e-5)
es = EarlyStopping(patience=10)
#chkpt = ModelCheckpoint(filepath='best_modelvgg.hdf5', save_best_only=True, save_weights_only=True)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'],optimizer=opt)

batch_size = 16
nb_epochs = 3

# Get a train data generator
train_data_gen = data_gen(data=train_data, batch_size=batch_size)

# Define the number of training steps
nb_train_steps = train_data.shape[0]//batch_size

print("Number of training and validation steps: {} and {}".format(nb_train_steps, len(valid_data)))


# # Fit the model
checkpoint = tf.keras.callbacks.ModelCheckpoint("PreFineTunebestVGG_StateOfTheArtData.h5", monitor="val_loss", mode="min", save_be
st_only=True, verbose=1)
history = model.fit_generator(train_data_gen, epochs=nb_epochs, steps_per_epoch=nb_train_steps,
                              validation_data=(valid_data, valid_labels),
                              callbacks=[es, checkpoint],
                              class_weight={0:1.0, 1:0.4})
```

```
Number of training and validation steps: 8 and 170
Epoch 1/3
8/8 [==============================] - 80s 10s/step - loss: 0.5284 - accuracy: 0.6250 - val_loss: 0.8946 - val_accura
cy: 0.4647

Epoch 00001: val_loss improved from inf to 0.89462, saving model to PreFineTunebestVGG_StateOfTheArtData.h5
Epoch 2/3
8/8 [==============================] - 83s 10s/step - loss: 0.3401 - accuracy: 0.6641 - val_loss: 0.9224 - val_accura
cy: 0.4647

Epoch 00002: val_loss did not improve from 0.89462
Epoch 3/3
8/8 [==============================] - 86s 11s/step - loss: 0.3114 - accuracy: 0.7031 - val_loss: 1.0490 - val_accura
cy: 0.4647

Epoch 00003: val_loss did not improve from 0.89462
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```