Predicting whether a patient is normal, has bacterial or has viral pneumonia

Transfer learninig - VGG16

In [1]:
```
#Importing packages

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import image
import keras
from keras.models import Sequential
from keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense, Dropout, Activation
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.optimizers import SGD
from keras.losses import categorical_crossentropy
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import os, os.path
from shutil import move
```

```
Using TensorFlow backend.
```

Splitting the pneumonia into viral and bacteria foldersfolders

In [2]:
```
#Splitting the pneumonia into viral and bacteria foldersfolders
train_location="chest_xray/train"
test_location="chest_xray/test"

orig_train_pneumonia="chest_xray/train/PNEUMONIA"
orig_test_pneumonia="chest_xray/test/PNEUMONIA"
```

In [3]:
```
#Definition to move viral and bacteria pneumonia images into differenct directory and remove the parent directo
#Purpose is to be able to use ImageDataGenerator.flow_from_direcotry()
def reorganize_files(pneumonia_directory, parent_directory):
    bac_dir=parent_directory + "/Bacterial"
    vir_dir=parent_directory + "/Viral"

    os.mkdir(bac_dir)
    os.mkdir(vir_dir)

    for filename in os.listdir(pneumonia_directory):
        if (filename.lower().find("bacteria") == -1): #Did not contain bacteria in name, must be viral.
            move(pneumonia_directory+"/"+filename,vir_dir)
        else: #This is bacterial pneumonia.
            move(pneumonia_directory+"/"+filename,bac_dir)

    os.rmdir(pneumonia_directory)
```

In [4]:
```
#Check if the folder were indeed been organized., if not, move it.

if(os.path.exists(orig_train_pneumonia)):
    reorganize_files(orig_train_pneumonia, train_location)

if(os.path.exists(orig_test_pneumonia)):
    reorganize_files(orig_test_pneumonia, test_location)
```

Preview the images

In [5]:
```
#Load a test image and see everything loaded

location = "chest_xray/train/NORMAL"
location_b = "chest_xray/train/Bacterial"
location_v ="chest_xray/train/Viral"

train_normal_example = plt.imread(location + "/IM-0311-0001.jpeg")
train_bacterial_example = plt.imread(location_b + "/person1000_bacteria_2931.jpeg")
train_Viral_example = plt.imread(location_v + "/person1000_virus_1681.jpeg")
#As this is black and white image, it only has one dimension.

print(train_normal_example.shape)
print(train_bacterial_example.shape)
print(train_Viral_example.shape)
```

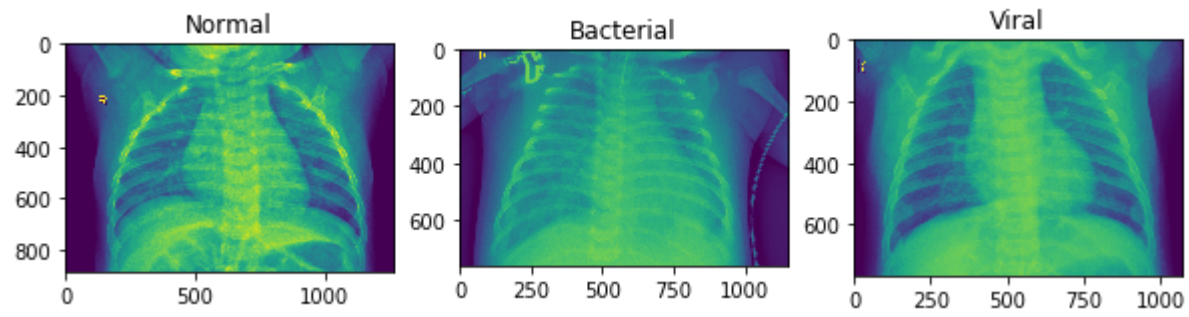
```
(885, 1268)
(760, 1152)
(768, 1072)
```

In [6]:
```
f = plt.figure(figsize= (10,6))
a0 = f.add_subplot(1, 3, 1)
img_plot = plt.imshow(train_normal_example)
a0.set_title('Normal')

a1 = f.add_subplot(1,3,2)
img_plot = plt.imshow(train_bacterial_example)
a1.set_title('Bacterial')

a2 = f.add_subplot(1, 3, 3)
img_plot = plt.imshow(train_Viral_example)
a2.set_title('Viral')
```

Out[6]: Text(0.5, 1.0, 'Viral')



Explore dataset - imbalanced data

1. There are more images in class bacterial.
2. There are only 16 images in validation dataset(8 normal, 8 bacterial). It does not contain any viral pneuomial image

In [7]:
```
#training dataset
DIR1= location
DIR2 = location_b
DIR3 = location_v
print ("Training data, normal:", len([name for name in os.listdir(DIR1) if os.path.isfile(os.path.join(DIR1, nar
print ("Training data, bacterial:", len([name for name in os.listdir(DIR2) if os.path.isfile(os.path.join(DIR2,
print ("Training data, viral:", len([name for name in os.listdir(DIR3) if os.path.isfile(os.path.join(DIR3, name
```

```
Training data, normal: 1341
Training data, bacterial: 2530
Training data, viral: 1345
```

In [8]:
```
# test dataset
DIR4= "chest_xray/test/NORMAL"
DIR5 = "chest_xray/test/Bacterial"
DIR6 = "chest_xray/test/Viral"
print ("Test data, normal:", len([name for name in os.listdir(DIR4) if os.path.isfile(os.path.join(DIR4, name))]
print ("Test data, bacterial:", len([name for name in os.listdir(DIR5) if os.path.isfile(os.path.join(DIR5, name
print ("Test data, viral:", len([name for name in os.listdir(DIR6) if os.path.isfile(os.path.join(DIR6, name))]
```

```
Test data, normal: 234
Test data, bacterial: 242
Test data, viral: 148
```

In [9]:
```
# validation dataset
DIR6= "chest_xray/val/NORMAL"
DIR7 = "chest_xray/val/PNEUMONIA"
print ("validation data, normal:", len([name for name in os.listdir(DIR6) if os.path.isfile(os.path.join(DIR6, n
print ("validation data, bacterial:", len([name for name in os.listdir(DIR7) if os.path.isfile(os.path.join(DIR7
```

```
validation data, normal: 8
validation data, bacterial: 8
```

Load dataset

In [10]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,
                                 shear_range = 0.2,
                                 zoom_range = 0.2,
                                 horizontal_flip = True)

test_datagen=ImageDataGenerator(rescale=1./255)

classes=["Bacterial", "NORMAL", "Viral"]


training_set = train_datagen.flow_from_directory(train_location,
                                                 target_size = (224, 224), #For transfer Learning.  Make sure to
                                                 batch_size = 32,
                                                 classes=classes,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(test_location,
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            classes=classes,
                                            class_mode = 'categorical')
```

```
Found 5216 images belonging to 3 classes.
Found 624 images belonging to 3 classes.
```

transfer learning - VGG 16

In [11]:
```
from keras.applications.vgg16 import VGG16
from keras.layers import Flatten
from keras.layers import Dense
from keras.models import Model

input_shape=224,224,3
VGG16_model = VGG16(weights='imagenet',input_shape=input_shape)
#VGG16_model.summary()
output = VGG16_model.get_layer("block5_pool").output

x1=Flatten()(output)
x2 = Dense(1000, activation='relu')(x1)
x3 = Dense(700, activation='relu')(x2)
x4 = Dense(300, activation='relu')(x3)
x5 = Dense(100, activation='relu')(x4)
x6 = Dense(50, activation='relu')(x5)
x7 = Dense(15, activation='relu')(x6)
my_preds = Dense(3, activation='softmax')(x7)

my_model2=Model(VGG16_model.input, my_preds)
my_model2.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              25089000
_________________________________________________________________
dense_2 (Dense)              (None, 700)               700700
_________________________________________________________________
dense_3 (Dense)              (None, 300)               210300
_________________________________________________________________
dense_4 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_5 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_6 (Dense)              (None, 15)                765
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 48
=================================================================
Total params: 40,750,651
Trainable params: 40,750,651
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
#compile
from keras.optimizers import adam
import keras
my_model2.compile(loss=keras.losses.categorical_crossentropy,
              optimizer='adam',
              metrics=['accuracy'])
```

In [18]:
```
#fit
hist2=my_model2.fit_generator(training_set,
                         steps_per_epoch=10,
                         epochs=30,
                         validation_data=test_set,
                         validation_steps=5)
```

```
Epoch 1/30
10/10 [==============================] - 7s 706ms/step - loss: 0.6905 - accuracy: 0.6781 - val_loss: 0.7473 -
val_accuracy: 0.7250
Epoch 2/30
10/10 [==============================] - 6s 637ms/step - loss: 0.6555 - accuracy: 0.7094 - val_loss: 0.9024 -
val_accuracy: 0.6938
Epoch 3/30
10/10 [==============================] - 6s 632ms/step - loss: 0.6621 - accuracy: 0.7031 - val_loss: 0.5514 -
val_accuracy: 0.7500
Epoch 4/30
10/10 [==============================] - 7s 661ms/step - loss: 0.6523 - accuracy: 0.6938 - val_loss: 1.0484 -
val_accuracy: 0.6875
Epoch 5/30
10/10 [==============================] - 7s 675ms/step - loss: 0.6488 - accuracy: 0.7219 - val_loss: 0.7472 -
val_accuracy: 0.7188
Epoch 6/30
10/10 [==============================] - 7s 686ms/step - loss: 0.6704 - accuracy: 0.7156 - val_loss: 0.3930 -
val_accuracy: 0.7312
Epoch 7/30
10/10 [==============================] - 7s 657ms/step - loss: 0.6447 - accuracy: 0.7375 - val_loss: 0.6556 -
val_accuracy: 0.7688
Epoch 8/30
10/10 [==============================] - 6s 634ms/step - loss: 0.6367 - accuracy: 0.7156 - val_loss: 0.5601 -
val_accuracy: 0.6944
Epoch 9/30
10/10 [==============================] - 7s 659ms/step - loss: 0.6534 - accuracy: 0.7188 - val_loss: 0.6539 -
val_accuracy: 0.7437
Epoch 10/30
10/10 [==============================] - 6s 638ms/step - loss: 0.7241 - accuracy: 0.6625 - val_loss: 0.8619 -
val_accuracy: 0.7312
Epoch 11/30
10/10 [==============================] - 7s 673ms/step - loss: 0.6807 - accuracy: 0.7281 - val_loss: 0.7340 -
val_accuracy: 0.7312
Epoch 12/30
10/10 [==============================] - 6s 647ms/step - loss: 0.6415 - accuracy: 0.7281 - val_loss: 0.8184 -
val_accuracy: 0.7222
Epoch 13/30
10/10 [==============================] - 7s 658ms/step - loss: 0.5845 - accuracy: 0.7250 - val_loss: 0.7818 -
val_accuracy: 0.7188
Epoch 14/30
10/10 [==============================] - 7s 664ms/step - loss: 0.6578 - accuracy: 0.7156 - val_loss: 0.7039 -
val_accuracy: 0.6875
Epoch 15/30
10/10 [==============================] - 7s 676ms/step - loss: 0.6879 - accuracy: 0.6906 - val_loss: 0.6252 -
val_accuracy: 0.6687
Epoch 16/30
10/10 [==============================] - 7s 658ms/step - loss: 0.6410 - accuracy: 0.6844 - val_loss: 0.4230 -
val_accuracy: 0.7361
Epoch 17/30
10/10 [==============================] - 7s 709ms/step - loss: 0.6489 - accuracy: 0.6906 - val_loss: 0.6890 -
val_accuracy: 0.7563
Epoch 18/30
10/10 [==============================] - 7s 650ms/step - loss: 0.7114 - accuracy: 0.6406 - val_loss: 2.0584 -
val_accuracy: 0.4938
Epoch 19/30
10/10 [==============================] - 7s 684ms/step - loss: 0.6994 - accuracy: 0.7063 - val_loss: 0.6891 -
val_accuracy: 0.6313
Epoch 20/30
10/10 [==============================] - 6s 641ms/step - loss: 0.5995 - accuracy: 0.7156 - val_loss: 0.4614 -
val_accuracy: 0.7569
Epoch 21/30
10/10 [==============================] - 7s 691ms/step - loss: 0.6973 - accuracy: 0.7156 - val_loss: 0.7049 -
val_accuracy: 0.7063
Epoch 22/30
10/10 [==============================] - 6s 626ms/step - loss: 0.6422 - accuracy: 0.7125 - val_loss: 0.6628 -
val_accuracy: 0.7375
Epoch 23/30
10/10 [==============================] - 7s 669ms/step - loss: 0.6170 - accuracy: 0.7437 - val_loss: 0.6685 -
val_accuracy: 0.7812
Epoch 24/30
10/10 [==============================] - 6s 644ms/step - loss: 0.6576 - accuracy: 0.7219 - val_loss: 0.8216 -
val_accuracy: 0.7500
Epoch 25/30
10/10 [==============================] - 7s 664ms/step - loss: 0.7416 - accuracy: 0.6812 - val_loss: 1.1180 -
val_accuracy: 0.7250
Epoch 26/30
10/10 [==============================] - 6s 642ms/step - loss: 0.6177 - accuracy: 0.7406 - val_loss: 0.5067 -
val_accuracy: 0.7250
Epoch 27/30
10/10 [==============================] - 7s 661ms/step - loss: 0.6036 - accuracy: 0.7406 - val_loss: 0.6551 -
val_accuracy: 0.7688
```

```
Epoch 28/30
10/10 [==============================] - 6s 637ms/step - loss: 0.6299 - accuracy: 0.7281 - val_loss: 0.6270 -
val_accuracy: 0.6111
Epoch 29/30
10/10 [==============================] - 7s 654ms/step - loss: 0.7060 - accuracy: 0.6750 - val_loss: 0.8576 -
val_accuracy: 0.7125
Epoch 30/30
10/10 [==============================] - 7s 667ms/step - loss: 0.6906 - accuracy: 0.7094 - val_loss: 0.7301 -
val_accuracy: 0.7875
```

Confusion Matrix

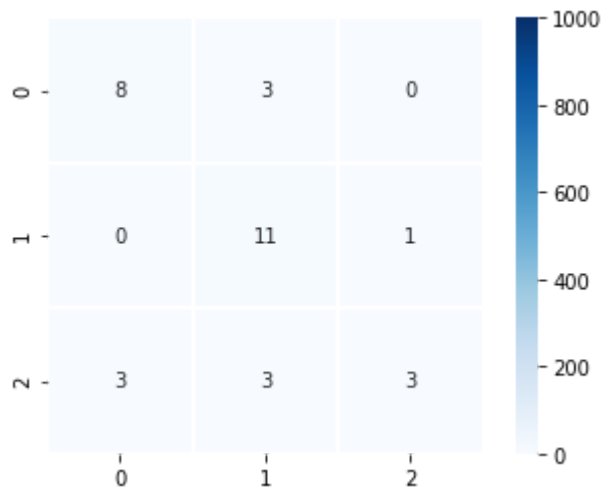
In [19]:
```
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

test_image, test_label_original=next(test_set)
predicted_label_original=my_model2.predict(test_image, verbose=1)
predicted_label=predicted_label_original.argmax(axis=1)
test_label=test_label_original.argmax(axis=1)

confusion_matrix=confusion_matrix(test_label, predicted_label)
print(confusion_matrix)

matrix=sns.heatmap(confusion_matrix,linewidths=1,vmax=1000,
square=True, cmap="Blues",annot=True)
accuracy_score=accuracy_score(test_label, predicted_label)
print('test dataset accuracy:',accuracy_score)
```

```
32/32 [==============================] - 0s 2ms/step
[[ 8  3  0]
 [ 0 11  1]
 [ 3  3  3]]
test dataset accuracy: 0.6875
```



In [20]:
```
from sklearn.metrics import classification_report
target_names = ['Normal', 'Bacterial', 'Viral']
print(classification_report(test_label, predicted_label, target_names=target_names))
```

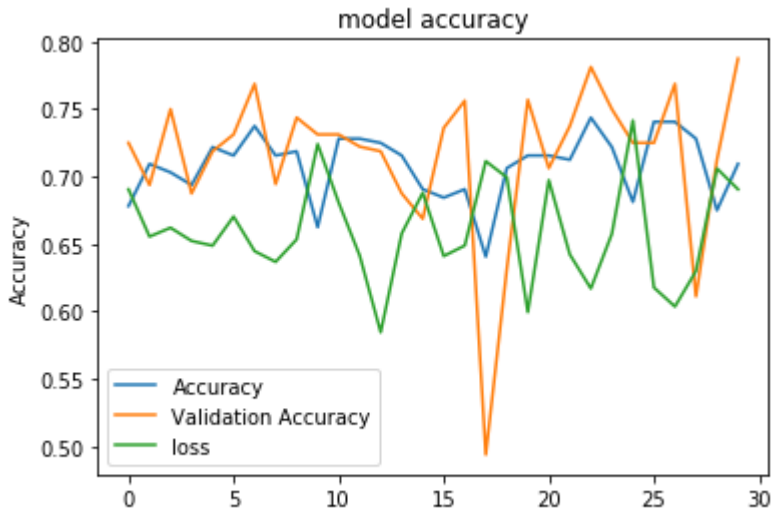
```
              precision    recall  f1-score   support

      Normal       0.73      0.73      0.73        11
   Bacterial       0.65      0.92      0.76        12
       Viral       0.75      0.33      0.46         9

    accuracy                           0.69        32
   macro avg       0.71      0.66      0.65        32
weighted avg       0.70      0.69      0.66        32
```

In [22]:
```
import matplotlib.pyplot as plt
from keras.callbacks import History

plt.plot(hist2.history["accuracy"])
plt.plot(hist2.history['val_accuracy'])
plt.plot(hist2.history['loss'])
#plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation Loss"])
plt.show()
```



ROC

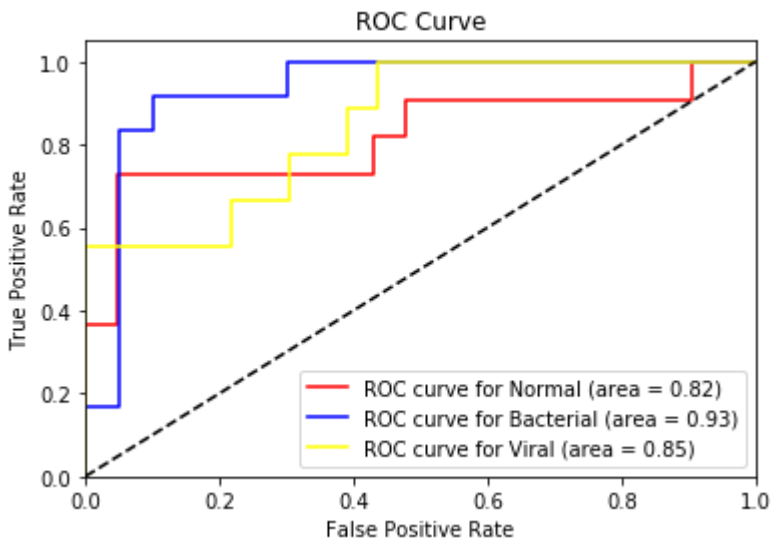
In [25]:
```
from sklearn.metrics import roc_curve, auc, roc_auc_score
import matplotlib.pyplot as plt

fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(test_label_original[:, i], predicted_label_original[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])


colors=["red","blue","yellow"]
plt.figure()
for i in range(3):
    plt.plot(fpr[i], tpr[i],color=colors[i], label='ROC curve for %s (area = %0.2f)' % (target_names[i],roc_auc[
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")

plt.show()
```



In [ ]: