Import Libraries ¶

In [1]:

```
import pandas as pd
import numpy as np
import numpy.random as nr
from numpy.random import seed
import matplotlib.pyplot as plt
from glob import glob
from pathlib import Path
import cv2
import keras
import tensorflow as tf
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Conv2D,Dense,Flatten,Dropout,MaxPooling2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
%matplotlib inline
```

```
Using TensorFlow backend.
```

Load Dataset

In [2]:

```
train_path = Path('train_mod')
test_path = Path('test_mod')
valid_path = Path('valid_mod')
```

In [3]:

```
train_set = ImageDataGenerator().flow_from_directory(train_path, target_size
                                                     classes = ['NORMAL', 'P
                                                     batch_size=10)

test_set = ImageDataGenerator().flow_from_directory(test_path, target_size=(
                                                    classes = ['NORMAL', 'PN
                                                    batch_size=1)

valid_set = ImageDataGenerator().flow_from_directory(valid_path, target_size
                                                     classes = ['NORMAL', 'P
                                                     batch_size=10)
```

```
Found 4100 images belonging to 2 classes.
Found 878 images belonging to 2 classes.
Found 878 images belonging to 2 classes.
```

Train on VGG16 Model

In [4]:

```
vgg16_model = VGG16()
#vgg16_model.summary()
```

```
WARNING:tensorflow:From /home/students/student5_14a/anaconda3/envs/Keras36/
lib/python3.6/site-packages/tensorflow_core/python/ops/resource_variable_op
s.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.op
s.resource_variable_ops) with constraint is deprecated and will be removed
in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

In [5]:

```
last_layer = vgg16_model.get_layer('fc2').output
out = Dense(2, activation='softmax', name='output_layer')(last_layer)
custom_vgg16_model = Model(inputs=vgg16_model.input, outputs=out)
custom_vgg16_model.summary()
```

```
Model: "model"
_________________________________________________________________
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 (Flatten)            (None, 25088)             0         
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544 
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312  
_________________________________________________________________
output_layer (Dense)         (None, 2)                 8194      
=================================================================
Total params: 134,268,738
```

```
Trainable params: 134,268,738
Non-trainable params: 0
_________________________________________________________________
```

In [6]:

```
for layer in custom_vgg16_model.layers[:-1]:
    layer.trainable = False
custom_vgg16_model.summary()
```

```
Model: "model"
_________________________________________________________________
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 (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
output_layer (Dense)         (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 8,194
```

```
Non-trainable params: 134,260,544
_________________________________________________________________
```

In [7]:

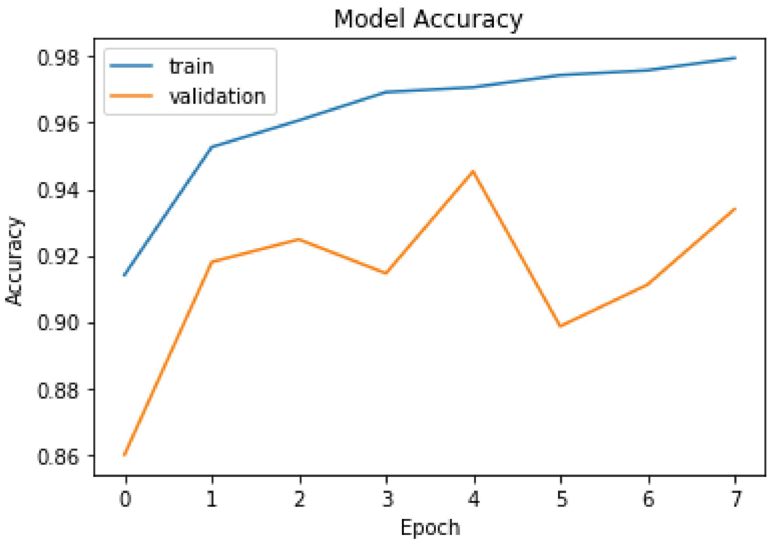
```
#compile model
custom_vgg16_model.compile(Adam(learning_rate=0.0001),
                           loss='categorical_crossentropy',
                           metrics=['accuracy'])

## Define the callback list
filepath = 'best_model.hdf5' # define where the model is saved
callbacks_list = [
    tf.keras.callbacks.EarlyStopping(
        monitor = 'val_loss', # Use loss to monitor the model
        verbose=1,
        patience = 3 # Stop after one step with lower accuracy
    ),
    tf.keras.callbacks.ModelCheckpoint(
        filepath = filepath, # file where the checkpoint is saved
        monitor = 'val_loss', # Don't overwrite the saved model unless val_l
        verbose=1,
        save_best_only = True # Only save model if it is the best
    )
]

nr.seed(1234)
tf.set_random_seed(4321)
history = custom_vgg16_model.fit_generator(train_set, steps_per_epoch=410,
                                           validation_data=valid_set,
                                           validation_steps=88,
                                           epochs=10, verbose=1,
                                           callbacks = callbacks_list)
```

```
Epoch 1/10
409/410 [============================>.] - ETA: 1s - loss: 0.2109 - acc: 0.
9142Epoch 1/10
 88/410 [=====>........................] - ETA: 6:34 - loss: 0.2947 - acc:
0.8599
Epoch 00001: val_loss improved from inf to 0.29473, saving model to best_mo
del.hdf5
410/410 [==============================] - 618s 2s/step - loss: 0.2106 - ac
c: 0.9141 - val_loss: 0.2947 - val_acc: 0.8599
Epoch 2/10
409/410 [============================>.] - ETA: 1s - loss: 0.1209 - acc: 0.
9526Epoch 1/10
 88/410 [=====>........................] - ETA: 6:42 - loss: 0.1898 - acc:
0.9180
Epoch 00002: val_loss improved from 0.29473 to 0.18978, saving model to bes
t_model.hdf5
410/410 [==============================] - 621s 2s/step - loss: 0.1207 - ac
c: 0.9527 - val_loss: 0.1898 - val_acc: 0.9180
Epoch 3/10
409/410 [============================>.] - ETA: 1s - loss: 0.1030 - acc: 0.
9606Epoch 1/10
 88/410 [=====>........................] - ETA: 6:29 - loss: 0.1751 - acc:
0.9248
Epoch 00003: val_loss improved from 0.18978 to 0.17510, saving model to bes
t_model.hdf5
410/410 [==============================] - 619s 2s/step - loss: 0.1028 - ac
c: 0.9607 - val_loss: 0.1751 - val_acc: 0.9248
```

```
Epoch 4/10
409/410 [============================>.] - ETA: 1s - loss: 0.0888 - acc: 0.
9692Epoch 1/10
 88/410 [=====>........................] - ETA: 6:40 - loss: 0.1911 - acc:
0.9146
Epoch 00004: val_loss did not improve from 0.17510
410/410 [==============================] - 616s 2s/step - loss: 0.0886 - ac
c: 0.9693 - val_loss: 0.1911 - val_acc: 0.9146
Epoch 5/10
409/410 [============================>.] - ETA: 1s - loss: 0.0825 - acc: 0.
9707Epoch 1/10
 88/410 [=====>........................] - ETA: 9:04 - loss: 0.1389 - acc:
0.9453
Epoch 00005: val_loss improved from 0.17510 to 0.13892, saving model to bes
t_model.hdf5
410/410 [==============================] - 686s 2s/step - loss: 0.0824 - ac
c: 0.9707 - val_loss: 0.1389 - val_acc: 0.9453
Epoch 6/10
409/410 [============================>.] - ETA: 1s - loss: 0.0733 - acc: 0.
9743Epoch 1/10
 88/410 [=====>........................] - ETA: 9:09 - loss: 0.2168 - acc:
0.8986
Epoch 00006: val_loss did not improve from 0.13892
410/410 [==============================] - 847s 2s/step - loss: 0.0732 - ac
c: 0.9744 - val_loss: 0.2168 - val_acc: 0.8986
Epoch 7/10
409/410 [============================>.] - ETA: 1s - loss: 0.0668 - acc: 0.
9758Epoch 1/10
 88/410 [=====>........................] - ETA: 8:07 - loss: 0.1991 - acc:
0.9112
Epoch 00007: val_loss did not improve from 0.13892
410/410 [==============================] - 828s 2s/step - loss: 0.0667 - ac
c: 0.9759 - val_loss: 0.1991 - val_acc: 0.9112
Epoch 8/10
409/410 [============================>.] - ETA: 0s - loss: 0.0610 - acc: 0.
9795Epoch 1/10
 88/410 [=====>........................] - ETA: 3:22 - loss: 0.1581 - acc:
0.9339
Epoch 00008: val_loss did not improve from 0.13892
410/410 [==============================] - 464s 1s/step - loss: 0.0608 - ac
c: 0.9795 - val_loss: 0.1581 - val_acc: 0.9339
Epoch 00008: early stopping
```

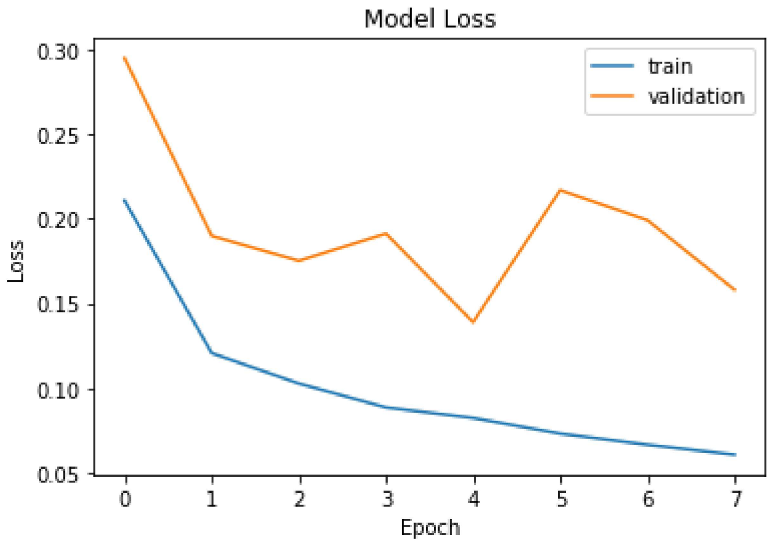
In [8]:

```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'validation'], loc='best')
plt.show()
```



In [9]:

```
# 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', 'validation'], loc='best')
plt.show()
```



Predict

In [10]:

```
train_loss, train_acc = custom_vgg16_model.evaluate_generator(train_set,
                                                              steps=train_se
                                                              verbose=1)
test_loss, test_acc = custom_vgg16_model.evaluate_generator(test_set,
                                                            steps=test_set.s
                                                            verbose=1)
```

```
4100/4100 [==============================] - 2468s 602ms/step - loss: 0.053
5 - acc: 0.9827
878/878 [==============================] - 126s 143ms/step - loss: 0.5025 -
acc: 0.8383
```

In [12]:

```
print("Train Accuracy: {:.2f}".format(train_acc))
print("Train Loss: {:.2f}".format(train_loss))
print("Test Accuracy: {:.2f}".format(test_acc))
print("Test Loss: {:.2f}".format(test_loss))
```

```
Train Accuracy: 0.98
Train Loss: 0.05
Test Accuracy: 0.84
Test Loss: 0.50
```

In [13]:

```
predictions = custom_vgg16_model.predict_generator(test_set,
                                                   steps=test_set.samples,
                                                   verbose=1)
y_pred = np.argmax(predictions, axis=1)
y_test = test_set.classes
print(predictions.shape)
```

```
878/878 [==============================] - 123s 140ms/step
(878, 2)
```

In [14]:

```
report = classification_report(y_test, y_pred,
                               target_names=['NORMAL', 'PNEUMONIA'])
print(report)
```

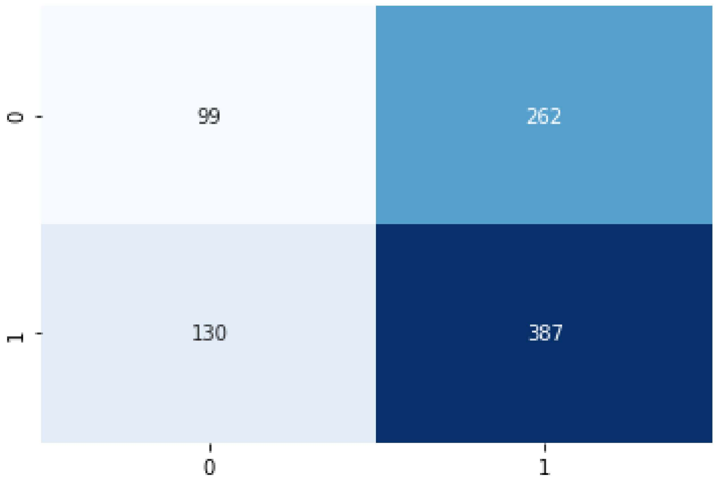
```
              precision    recall  f1-score   support

      NORMAL       0.43      0.27      0.34       361
   PNEUMONIA       0.60      0.75      0.66       517

    accuracy                           0.55       878
   macro avg       0.51      0.51      0.50       878
weighted avg       0.53      0.55      0.53       878
```

In [16]:

```
#plot confusion matrix
sns.heatmap(
    confusion_matrix(y_test, y_pred),
    annot=True,
    fmt="d",
    cbar = False,
    cmap = plt.cm.Blues
)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb51eb3ada0>
```


99
262
130
387
0
1
0
1