# Pneumonia Prediction Project

Jessica Rodriguez

Flatiron School

Github: jesrodriguez0816

```
# Import packages and statements
from tensorflow.random import set_seed
set_seed(321)

import numpy as np
np.random.seed(123)

#import module
import pickle
```

```
!pip install lime
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     |████████████████████████████████| 275 kB 4.8 MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/di
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.7/di
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dis
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/p
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/d
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dis
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dis
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-pac
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283858 sha
  Stored in directory: /root/.cache/pip/wheels/ca/cb/e5/ac701e12d365a08917bf4c61
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

```
# Import packages and statements
import os
import glob
from google.colab import drive

import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, im
from tensorflow.keras import models, layers, optimizers, regularizers, activations
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlat
from tensorflow.keras.applications import VGG16, VGG19

from sklearn.metrics import confusion_matrix, classification_report
from sklearn.utils.class_weight import compute_class_weight

import lime
from lime import lime_base
from lime import lime_image
from skimage.segmentation import mark_boundaries
```

## Upload Data

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
# Get current directory
print(os.getcwd())

# Get contents of the directory
print(os.listdir())
```

```
     /content
     ['.config', 'drive', 'sample_data']
```

```
# Location of Zip File
drive_path = 'drive/MyDrive/Datasets/archive(6).zip'
local_path = '/content'
```

## Data Preprocessing and Exploration

- Obtain and Store the Data in a train / test / val split
- Find the total number of images
- Scale data
- Explore the data

```
# File Paths
path_train_pneumonia = 'drive/MyDrive/chest_xray/train/PNEUMONIA'
path_train_normal = 'drive/MyDrive/chest_xray/train/NORMAL'
path_test_pneumonia = 'drive/MyDrive/chest_xray/test/PNEUMONIA'
path_test_normal = 'drive/MyDrive/chest_xray/test/NORMAL'
path_val_pneumonia = 'drive/MyDrive/chest_xray/val/PNEUMONIA'
path_val_normal = 'drive/MyDrive/chest_xray/val/NORMAL'

# How many images in each set
print('Train Pneumonia', len(os.listdir(path_train_pneumonia)))
print('Train Normal', len(os.listdir(path_train_normal)))
print('Test Pneumonia', len(os.listdir(path_test_pneumonia)))
print('Test Normal', len(os.listdir(path_test_normal)))
print('Val Pneumonia', len(os.listdir(path_val_pneumonia)))
print('Val Normal', len(os.listdir(path_val_normal)))
```

```
    Train Pneumonia 3875
    Train Normal 1341
    Test Pneumonia 390
    Test Normal 234
    Val Pneumonia 8
    Val Normal 8
```

```
# Create data generators
# Ensure class_mode is binary
# Adjust batch size for training, testing, and validation sets (number of examples use
# Use a target size of 224x224 px for each image
train_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
                                                    'drive/MyDrive/chest_xray/train',
                                                     target_size=(224, 224),
                                                     batch_size=5216,
                                                     class_mode='binary',
                                                     seed=123)
test_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
                                                    'drive/MyDrive/chest_xray/test',
                                                     target_size=(224, 224),
                                                     batch_size=624,
                                                     class_mode='binary',
                                                     seed=123)
val_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
                                                    'drive/MyDrive/chest_xray/val',
                                                     target_size=(224, 224),
                                                     batch_size=16,
                                                     class_mode='binary',
                                                     seed=123)
```

```
    Found 5216 images belonging to 2 classes.
    Found 624 images belonging to 2 classes.
    Found 16 images belonging to 2 classes.
```

```
# Create the data sets with the train/test/val splits
X_train, y_train = next(train_generator)
X_test, y_test = next(test_generator)
X_val, y_val = next(val_generator)
```

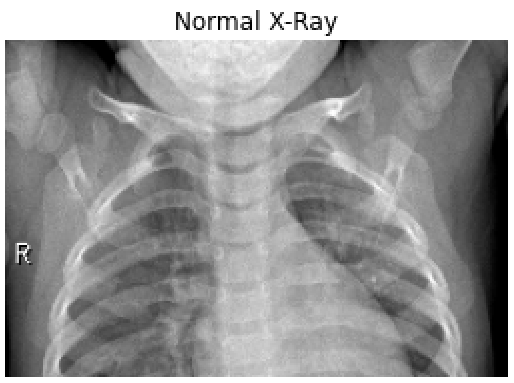
```
# Determine which classes are which (o and 1)
print(y_test[:20])
print(y_test.shape)
train_generator.class_indices
```

```
    [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1. 1. 1.]
    (624,)
    {'NORMAL': 0, 'PNEUMONIA': 1}
```

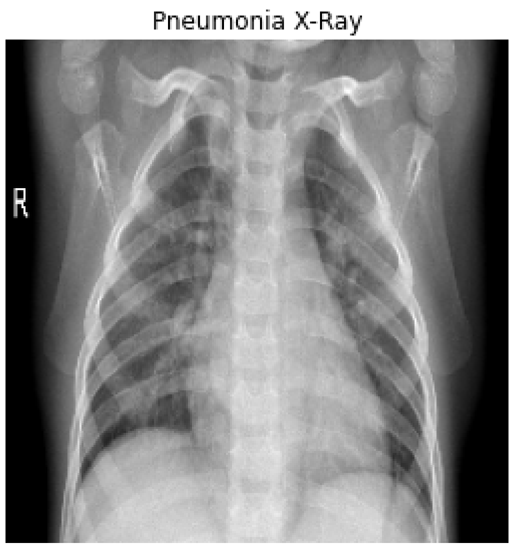
```
# Check shapes of each set
print(X_train.shape)
print(X_test.shape)
print(X_val.shape)
```

```
    (5216, 224, 224, 3)
    (624, 224, 224, 3)
    (16, 224, 224, 3)
```

```
# Look at example of normal xray
plt.imshow(np.squeeze(X_train[0]))
plt.axis('off')
plt.title('Normal X-Ray')
plt.tight_layout()
plt.show()
# Check the class label ('Normal':0)
print(y_train[0])
```

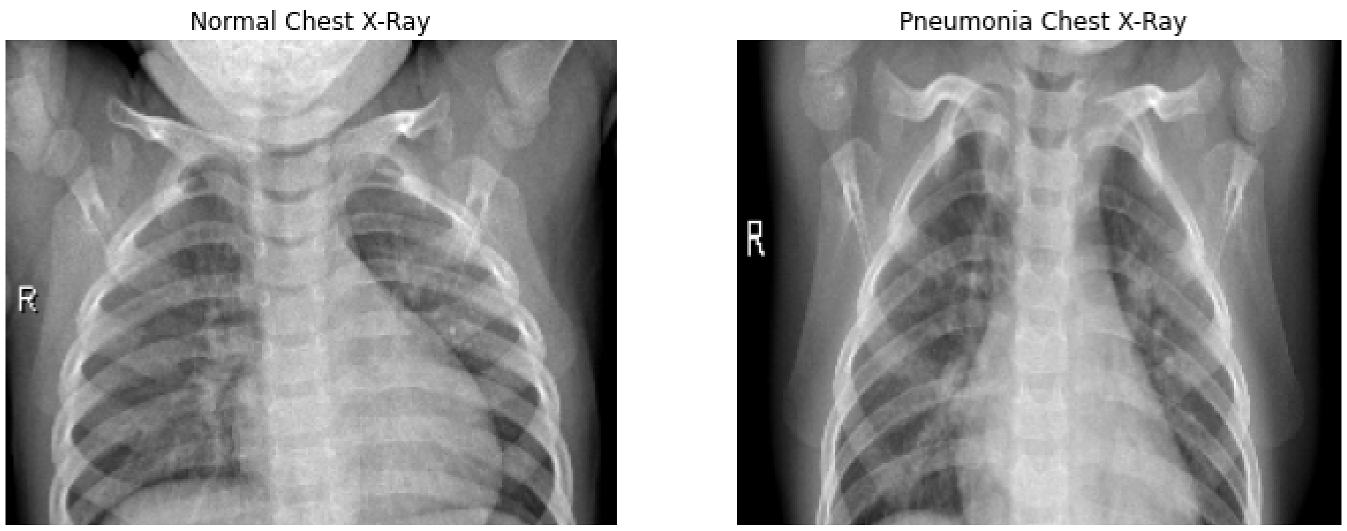


```
# Look at example of pneumonia xray
plt.imshow(np.squeeze(X_train[1]))
plt.axis('off')
plt.title('Pneumonia X-Ray')
plt.tight_layout()
plt.show()
# Check class label ('Pneumonia':1)
print(y_train[1])
```

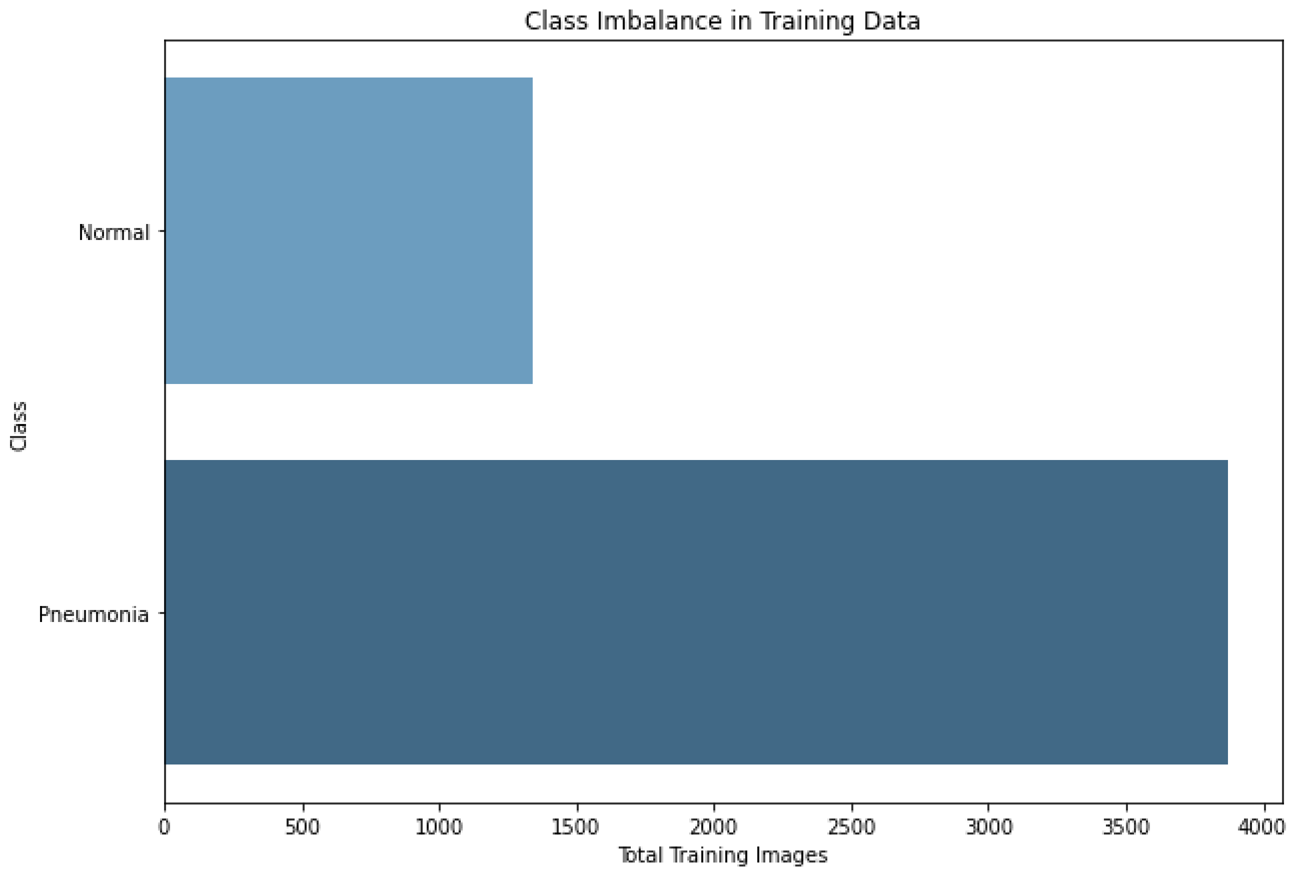


```
1.0
```

```
# Compare the two types
fig, axes = plt.subplots(1, 2, figsize=(12, 8))
axes[0].imshow(np.squeeze(X_train[0]))
axes[1].imshow(np.squeeze(X_train[1]))
axes[0].axis('off')
axes[1].axis('off')
axes[0].set_title('Normal Chest X-Ray')
axes[1].set_title('Pneumonia Chest X-Ray')
plt.show()
```



```
# Create bar plot to visualize class imbalance of training data

plt.figure(figsize=(10,7))
sns.barplot(x=[sum(y_train==0), sum(y_train==1)],
            y=['Normal', 'Pneumonia'],
            palette='Blues_d')
plt.title('Class Imbalance in Training Data')
plt.ylabel('Class')
plt.xlabel('Total Training Images')
plt.show()
```



According to the bar plot, there are far more pneumonia images than normal images.

## Baseline CNN

This model will be our reference point. It will include the following:

- 3 convolutional layers (building blocks to produce an image)
- 3 max pooling layers (to downsample previous convolutional layers)
- 1 fully connected layer

```
baseline = models.Sequential()

baseline.add(Conv2D(32, (3, 3), activation='relu',
                    input_shape=(224, 224, 3)))
baseline.add(MaxPooling2D((2, 2)))

baseline.add(Conv2D(32, (3, 3), activation='relu'))
baseline.add(MaxPooling2D(2, 2))

baseline.add(Conv2D(64, (3, 3), activation='relu'))
baseline.add(MaxPooling2D((2, 2)))

baseline.add(Flatten())
baseline.add(Dense(64, activation='relu'))
baseline.add(Dense(1, activation='sigmoid'))

baseline.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['acc'])
```

```
# Look at summary to see all layers
baseline.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 109, 109, 32)      9248      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 54, 54, 32)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 52, 52, 64)        18496     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 26, 26, 64)       0         
 2D)

 flatten (Flatten)           (None, 43264)             0

 dense (Dense)               (None, 64)                2768960

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 2,797,665
Trainable params: 2,797,665
Non-trainable params: 0
_________________________________________________________________
```

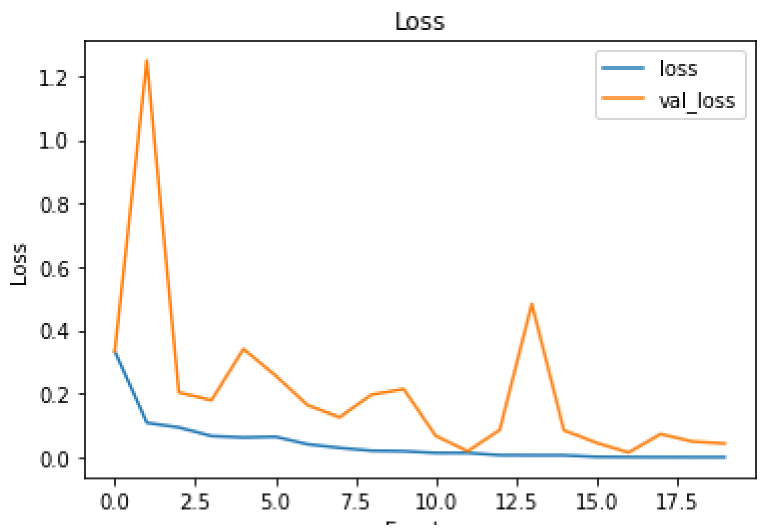
```
history = baseline.fit(X_train,
                       y_train,
                       epochs=20,
                       batch_size=50,
                       validation_data=(X_val, y_val))
```

```
Epoch 1/20
105/105 [==============================] - 13s 22ms/step - loss: 0.3330 - acc: 0
Epoch 2/20
105/105 [==============================] - 2s 19ms/step - loss: 0.1086 - acc: 0.
Epoch 3/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0939 - acc: 0.
Epoch 4/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0667 - acc: 0.
Epoch 5/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0623 - acc: 0.
Epoch 6/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0644 - acc: 0.
Epoch 7/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0410 - acc: 0.
Epoch 8/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0295 - acc: 0.
Epoch 9/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0204 - acc: 0.
Epoch 10/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0190 - acc: 0.
Epoch 11/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0135 - acc: 0.
Epoch 12/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0135 - acc: 0.
Epoch 13/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0065 - acc: 0.
Epoch 14/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0062 - acc: 0.
Epoch 15/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0061 - acc: 0.
Epoch 16/20
105/105 [==============================] - 2s 19ms/step - loss: 0.0011 - acc: 1.
Epoch 17/20
105/105 [==============================] - 2s 19ms/step - loss: 4.8557e-04 - acc
Epoch 18/20
```

```
105/105 [==============================] - 2s 19ms/step - loss: 1.6440e-04 - acc
Epoch 19/20
105/105 [==============================] - 2s 19ms/step - loss: 9.0718e-05 - acc
Epoch 20/20
105/105 [==============================] - 2s 19ms/step - loss: 7.5583e-05 - acc
```

```
# Write a function to evaliate the results of the model
def evaluate_results(results, y_test=y_test):
  """
  input results of model fitting.
  output loss and accuracy curves, and confusion matrix
  """

  history = results.history
  plt.figure()
  plt.plot(history['loss'])
  plt.plot(history['val_loss'])
  plt.legend(['loss', 'val_loss'])
  plt.title('Loss')
  plt.xlabel('Epochs')
  plt.ylabel('Loss')
  plt.show()

  plt.figure()
  plt.plot(history['acc'])
  plt.plot(history['val_acc'])
  plt.legend(['acc', 'val_acc'])
  plt.title('Accuracy')
  plt.xlabel('Epochs')
  plt.ylabel('Accuracy')
  plt.show()

  y_hat_test = results.model.predict(X_test)
  thresh = 0.5
  y_pred = (y_hat_test < thresh).astype(np.int)
  y_true = y_test.astype(np.int)
  cm = confusion_matrix(y_true, y_pred)
  sns.heatmap(cm, annot=True, cmap='Blues', fmt='0.5g')
  plt.xlabel('Predictions')
  plt.ylabel('Actuals')
  plt.title('Model Confusion Matrix')
  plt.show()

  print(classification_report(y_true, y_pred))
  print('\n')

  test_loss, test_acc = results.model.evaluate(X_test, y_test)
  print(f'Test Loss: {test_loss}')
  print(f'Test Acc: {test_acc}')
```

```
evaluate_results(history)
```



**Results of Baseline:**

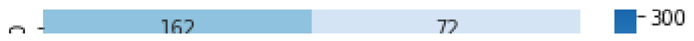
According to the confusion matrix, the model is mostly predicting the training class.

Due to a class imbalance-low validation and high training sets-this model is far from accurate. The model fits too well to the training set and is therefore "overfit".

For Iteration 2, I will implement an early stopping callback in attempt to balance the model. Essentially, this means that training will be stopped when the determined metric has stopped improving.

## CNN Iteration 2

- Address the class imbalance using class weights based on the results of the baseline model's confusion matrix, which shows the model is predicting mostly one class (training)
- Experiment with more epochs (the number of passes of the entire training dataset the machine learning algorithm has completed) and early stopping might prevent overfitting
- Reduce batch size to improve accuracy



```
from sklearn.utils import class_weight
```

```
# Set up our class weights
weights = compute_class_weight(class_weight = 'balanced', classes = np.unique(y_train)
weights_dict = dict(zip(np.unique(y_train), weights))
weights_dict

    {0.0: 1.9448173005219984, 1.0: 0.6730322580645162}
                      precision    recall  f1-score   support
```

```
# Set up early stopping and learning rate reduction
# Set patience to 5
early_stop = EarlyStopping(monitor='val_loss', mode='min', patience=5)
lr_redox = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5)
callbacks = [early_stop, lr_redox]

# Build model iteration 2 with improvements above and another conv layer
model2 = models.Sequential()

model2.add(Conv2D(32, (3, 3), activation='relu',
                  input_shape=(224, 224, 3)))
model2.add(MaxPooling2D((2, 2)))

model2.add(Conv2D(32, (3, 3), activation='relu'))
model2.add(MaxPooling2D(2, 2))

model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(MaxPooling2D((2, 2)))

model2.add(Flatten())
model2.add(Dense(64, activation='relu'))
model2.add(Dense(1, activation='sigmoid'))

model2.compile(loss='binary_crossentropy',
               optimizer='adam',
               metrics=['acc'])
```

```
model2.summary()
```

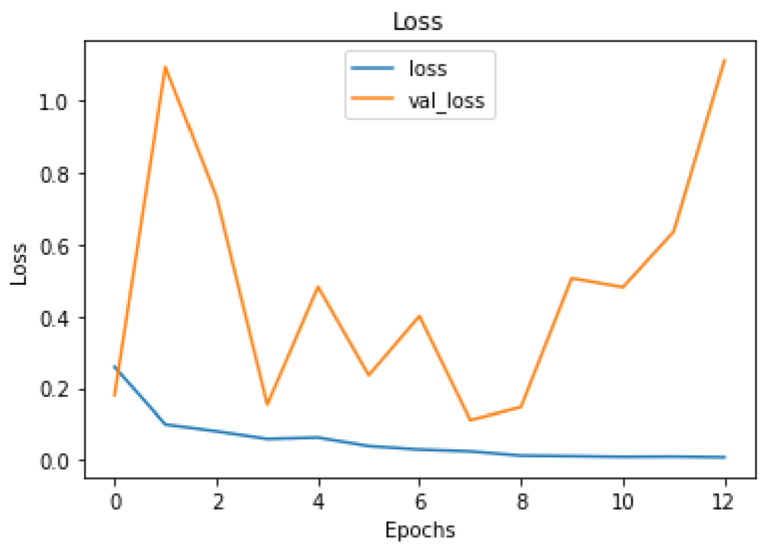
```
    Model: "sequential_1"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     conv2d_3 (Conv2D)           (None, 222, 222, 32)      896

     max_pooling2d_3 (MaxPooling  (None, 111, 111, 32)     0
     2D)

     conv2d_4 (Conv2D)           (None, 109, 109, 32)      9248

     max_pooling2d_4 (MaxPooling  (None, 54, 54, 32)       0
     2D)

     conv2d_5 (Conv2D)           (None, 52, 52, 64)        18496

     max_pooling2d_5 (MaxPooling  (None, 26, 26, 64)       0
     2D)

     flatten_1 (Flatten)         (None, 43264)             0

     dense_2 (Dense)             (None, 64)                2768960
```

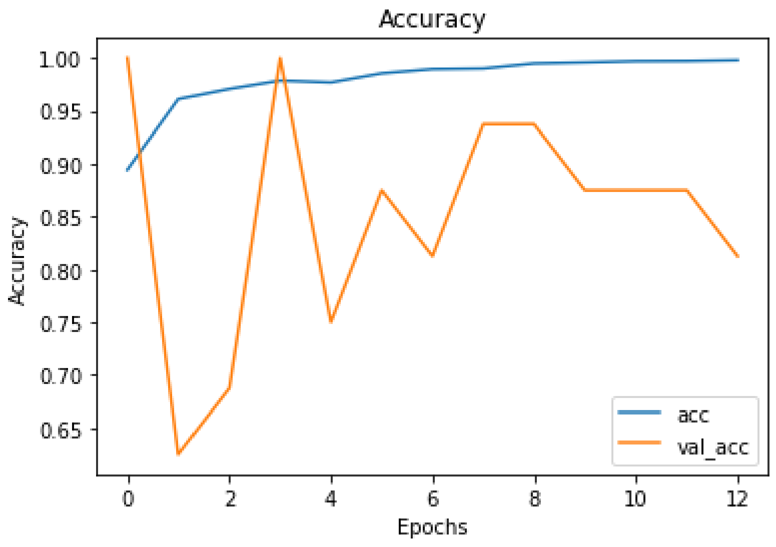
```
 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 2,797,665
Trainable params: 2,797,665
Non-trainable params: 0
_________________________________________________________________
```

```
results = model2.fit(X_train,
                   y_train,
                   epochs=50,
                   batch_size=30,
                   validation_data=(X_val, y_val),
                   callbacks=callbacks)
```

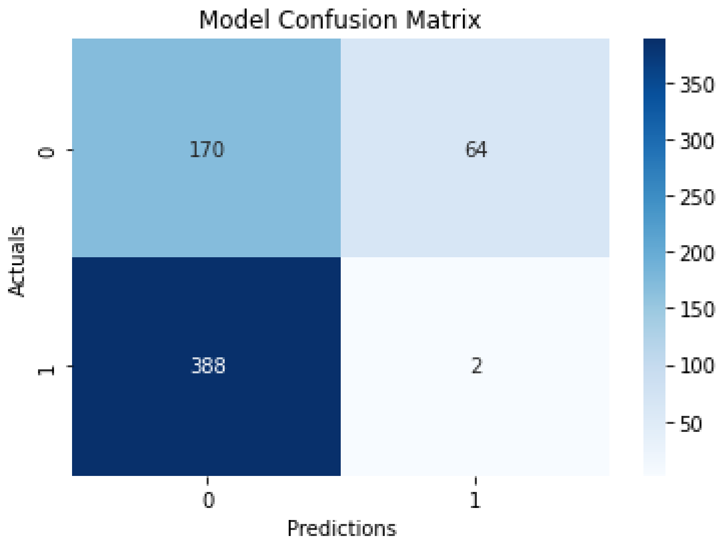
```
Epoch 1/50
174/174 [==============================] - 3s 14ms/step - loss: 0.2590 - acc: 0.
Epoch 2/50
174/174 [==============================] - 2s 13ms/step - loss: 0.0997 - acc: 0.
Epoch 3/50
174/174 [==============================] - 2s 13ms/step - loss: 0.0807 - acc: 0.
Epoch 4/50
174/174 [==============================] - 2s 13ms/step - loss: 0.0596 - acc: 0.
Epoch 5/50
174/174 [==============================] - 2s 13ms/step - loss: 0.0634 - acc: 0.
Epoch 6/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0397 - acc: 0.
Epoch 7/50
174/174 [==============================] - 2s 13ms/step - loss: 0.0300 - acc: 0.
Epoch 8/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0250 - acc: 0.
Epoch 9/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0130 - acc: 0.
Epoch 10/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0115 - acc: 0.
Epoch 11/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0097 - acc: 0.
Epoch 12/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0103 - acc: 0.
Epoch 13/50
174/174 [==============================] - 2s 12ms/step - loss: 0.0085 - acc: 0.
```

```
evaluate_results(results)
```





```
20/20 [==============================] - 0s 7ms/step
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:29: DeprecationWarn
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdo
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:30: DeprecationWarn
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdo
```



```
              precision    recall  f1-score   support

           0       0.30      0.73      0.43       234
           1       0.03      0.01      0.01       390

    accuracy                           0.28       624
   macro avg       0.17      0.37      0.22       624
```

**Results of Iteration 2:**

Our accuracy remained largely the same, though the loss improved.

The training/validation ratio is heavily skewed. In the next iteratrion I will adjust the validation training split for greater balance.

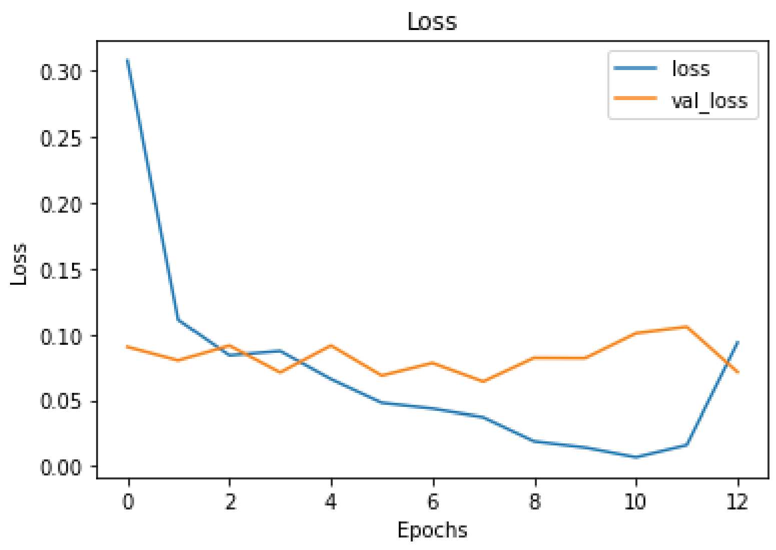
## CNN Iteration 3

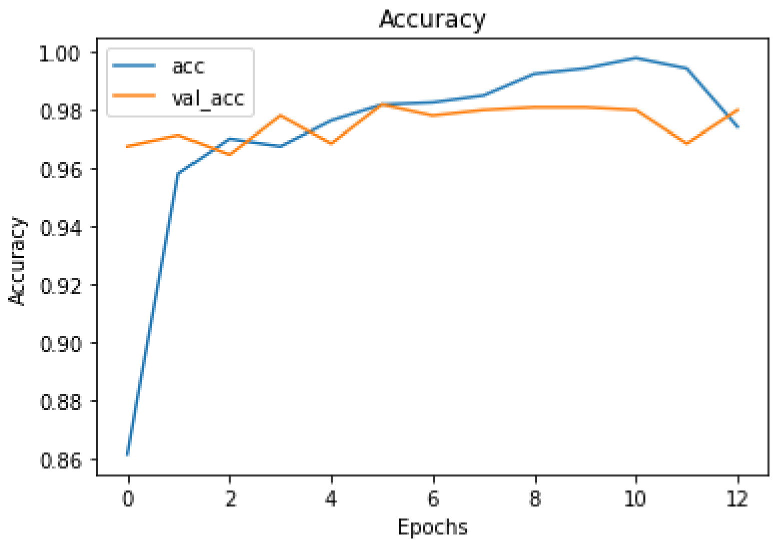
I will rework the validation size see if if this improves the loss and accuracy curves.

```
mod3 = models.Sequential()

mod3.add(Conv2D(32, (3, 3), activation='relu',
                    input_shape=(224, 224, 3)))
mod3.add(MaxPooling2D((2, 2)))

mod3.add(Conv2D(32, (3, 3), activation='relu'))
mod3.add(MaxPooling2D(2, 2))

mod3.add(Conv2D(64, (3, 3), activation='relu'))
mod3.add(MaxPooling2D((2, 2)))

mod3.add(Flatten())
mod3.add(Dense(64, activation='relu'))
mod3.add(Dense(1, activation='sigmoid'))

mod3.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['acc'])


results = mod3.fit(X_train,
                   y_train,
                   epochs=50,
                   batch_size=30,
                   validation_split=.2,
                   class_weight=weights_dict,
                   callbacks=callbacks)
```

```
    Epoch 1/50
    140/140 [==============================] - 3s 19ms/step - loss: 0.3077 - acc: 0.
    Epoch 2/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.1108 - acc: 0.
    Epoch 3/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0839 - acc: 0.
    Epoch 4/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0872 - acc: 0.
    Epoch 5/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0659 - acc: 0.
    Epoch 6/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0479 - acc: 0.
    Epoch 7/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0435 - acc: 0.
    Epoch 8/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0371 - acc: 0.
    Epoch 9/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0189 - acc: 0.
    Epoch 10/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0142 - acc: 0.
    Epoch 11/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0068 - acc: 0.
    Epoch 12/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0160 - acc: 0.
    Epoch 13/50
    140/140 [==============================] - 2s 14ms/step - loss: 0.0936 - acc: 0.
```

```
evaluate_results(results)
```





```
20/20 [==============================] - 0s 8ms/step
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:29: DeprecationWarn
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdo
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:30: DeprecationWarn
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdo
```

**Results of Iteration 3:**

The curves significantly improved. Model 3a improved a bit here. In the next iteration, I will keep this validation split instead of using the tiny validation set.

## CNN Iteration 4

In this iteration, I will add 2 convolutional layers per max pooling layer.

- I will also add convolutional layers will help detect more of the edges/nuances in the images with pneumonia
- I will also add another 2 blocks of layers (make the model deeper)

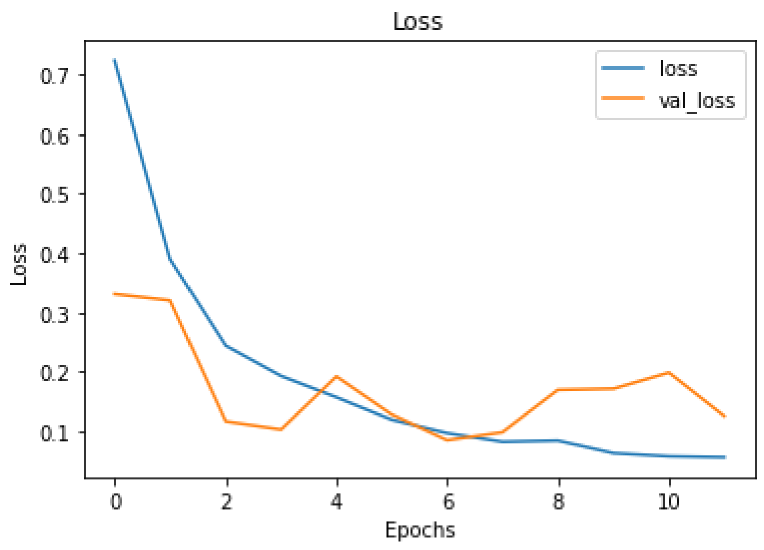
- Continue using 'RMSprop' as the optimizer

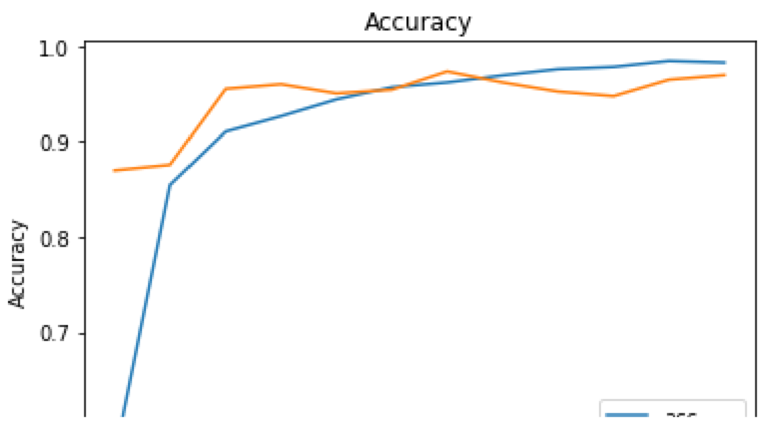
```
mod4 = models.Sequential()

mod4.add(Conv2D(32, (3, 3), activation='relu',
                    input_shape=(224, 224, 3)))
mod4.add(Conv2D(32, (3, 3), activation='relu'))
mod4.add(MaxPooling2D((2, 2)))

mod4.add(Conv2D(32, (3, 3), activation='relu'))
mod4.add(Conv2D(32, (3, 3), activation='relu'))
mod4.add(MaxPooling2D(2, 2))

mod4.add(Conv2D(64, (3, 3), activation='relu'))
mod4.add(Conv2D(64, (3, 3), activation='relu'))
mod4.add(MaxPooling2D((2, 2)))

mod4.add(Conv2D(64, (3, 3), activation='relu'))
mod4.add(Conv2D(64, (3, 3), activation='relu'))
mod4.add(MaxPooling2D((2, 2)))

mod4.add(Flatten())
mod4.add(Dense(128, activation='relu'))
mod4.add(Dense(64, activation='relu'))
mod4.add(Dense(1, activation='sigmoid'))

mod4.compile(loss='binary_crossentropy',
                optimizer='RMSprop',
                metrics=['acc'])


results = mod4.fit(X_train,
                   y_train,
                   epochs=50,
                   batch_size=32,
                   validation_split=.2,
                   class_weight=weights_dict,
                   callbacks=[early_stop])
```

```
Epoch 1/50
131/131 [==============================] - 6s 33ms/step - loss: 0.7239 - acc: 0.
Epoch 2/50
131/131 [==============================] - 3s 25ms/step - loss: 0.3898 - acc: 0.
Epoch 3/50
131/131 [==============================] - 3s 25ms/step - loss: 0.2438 - acc: 0.
Epoch 4/50
131/131 [==============================] - 3s 25ms/step - loss: 0.1926 - acc: 0.
Epoch 5/50
131/131 [==============================] - 3s 25ms/step - loss: 0.1567 - acc: 0.
Epoch 6/50
131/131 [==============================] - 3s 25ms/step - loss: 0.1187 - acc: 0.
Epoch 7/50
```

```
    131/131 [==============================] - 3s 26ms/step - loss: 0.0965 - acc: 0.
    Epoch 8/50
    131/131 [==============================] - 3s 25ms/step - loss: 0.0819 - acc: 0.
    Epoch 9/50
    131/131 [==============================] - 3s 25ms/step - loss: 0.0839 - acc: 0.
    Epoch 10/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.0628 - acc: 0.
    Epoch 11/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.0575 - acc: 0.
    Epoch 12/50
    131/131 [==============================] - 3s 25ms/step - loss: 0.0559 - acc: 0.
```

```
evaluate_results(results)
```



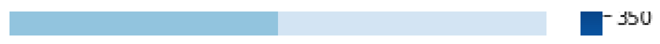


**Results of Iteration 4:**

There is significant model improvement in this iteration! According to the classification report and confusion matrix, the model is prediction normal x-rays very well.

Only a few "normal" images were classified as having pneumonia (False Positives). However, there are still many images with pneumonia misclassified as normal (False Negatives).

I will do another iteration on this model to improve accuracy.



## CNN Iteration 5

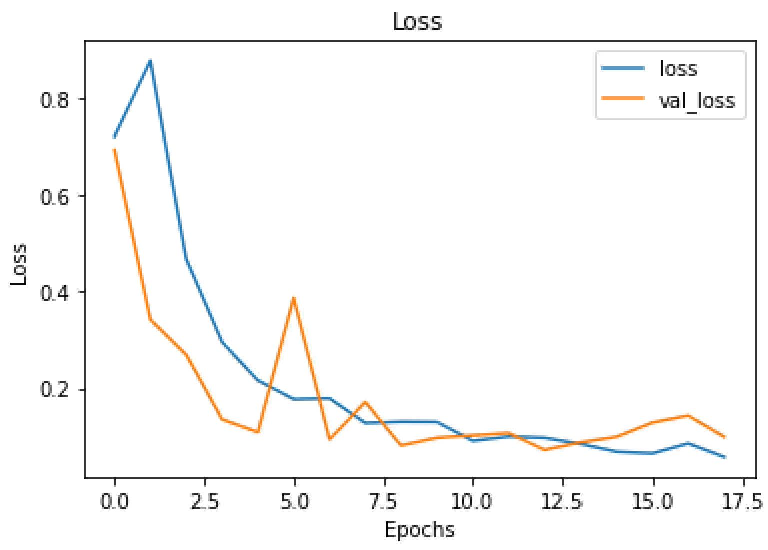
In this iteration, I attempt to pull more features out of images by deepening the neural network.

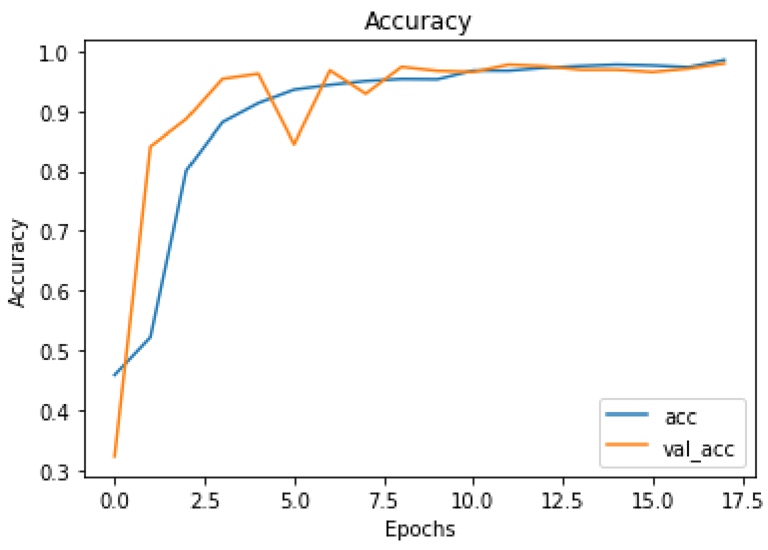
```
mod5 = models.Sequential()

mod5.add(Conv2D(32, (3, 3), activation='relu',
                    input_shape=(224, 224, 3)))
mod5.add(Conv2D(32, (3, 3), activation='relu'))
mod5.add(MaxPooling2D((2, 2)))

mod5.add(Conv2D(32, (3, 3), activation='relu'))
```

```
mod5.add(Conv2D(32, (3, 3), activation='relu'))
mod5.add(MaxPooling2D(2, 2))

mod5.add(Conv2D(64, (3, 3), activation='relu'))
mod5.add(Conv2D(64, (3, 3), activation='relu'))
mod5.add(MaxPooling2D((2, 2)))

mod5.add(Conv2D(64, (3, 3), activation='relu'))
mod5.add(Conv2D(64, (3, 3), activation='relu'))
mod5.add(MaxPooling2D((2, 2)))

mod5.add(Conv2D(128, (3, 3), activation='relu'))
mod5.add(Conv2D(128, (3, 3), activation='relu'))
mod5.add(MaxPooling2D((2, 2)))

mod5.add(Flatten())
mod5.add(Dense(128, activation='relu'))
mod5.add(Dense(64, activation='relu'))
mod5.add(Dense(1, activation='sigmoid'))

mod5.compile(loss='binary_crossentropy',
             optimizer='RMSprop',
             metrics=['acc'])


results = mod5.fit(X_train,
                   y_train,
                   epochs=50,
                   batch_size=32,
                   validation_split=.2,
                   class_weight=weights_dict,
                   callbacks=[early_stop])
```

```
    Epoch 1/50
    131/131 [==============================] - 5s 30ms/step - loss: 0.7211 - acc: 0.
    Epoch 2/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.8777 - acc: 0.
    Epoch 3/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.4675 - acc: 0.
    Epoch 4/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.2952 - acc: 0.
    Epoch 5/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.2157 - acc: 0.
    Epoch 6/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.1768 - acc: 0.
    Epoch 7/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.1785 - acc: 0.
    Epoch 8/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.1274 - acc: 0.
    Epoch 9/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.1302 - acc: 0.
    Epoch 10/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.1298 - acc: 0.
    Epoch 11/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.0905 - acc: 0.
    Epoch 12/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.0994 - acc: 0.
    Epoch 13/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.0968 - acc: 0.
    Epoch 14/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.0842 - acc: 0.
    Epoch 15/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.0680 - acc: 0.
    Epoch 16/50
    131/131 [==============================] - 3s 26ms/step - loss: 0.0645 - acc: 0.
    Epoch 17/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.0851 - acc: 0.
    Epoch 18/50
    131/131 [==============================] - 3s 27ms/step - loss: 0.0572 - acc: 0.
```

```
evaluate_results(results)
```





```
20/20 [==============================] - 0s 11ms/step
```

```
**Results of Iteration 5**

When we look at all the models, model 5 ha
False Negatives than most of the other mod

**Which model is the best fit?**

In diagnosing pneumonia, the best practice
Negatives.

**Iteration 5** has less false negatives a
of the other models.  Iteration 5 also has
be considered our final, best model.
```

**Results of Iteration 5**

When we look at all the models, model 5 has the best accuracy (75%), and less False Negatives than most of the other models.

**Which model is the best fit?**

In diagnosing pneumonia, the best practice is to reduce the number of False Negatives.

**Iteration 5** has less false negatives and less false positives than most of the other models. Iteration 5 also has an accuracy score of 75%. It will be considered our final, best model.

```
# Save Model 5 as final model
final_cnn_path = 'drive/MyDrive/Datasets/final_pre_trained_cnn.hd5'

mod5.save(final_cnn_path)
```

```
    WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit
```

```
# Reload model
final_model = models.load_model(final_cnn_path)
```

```
    Test AUC: 0.7552051205055071
```

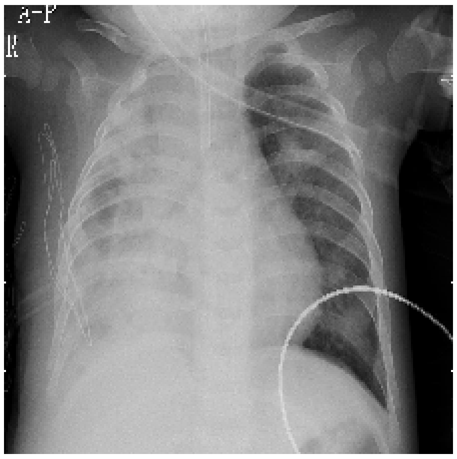
## Image Feature Exploration

Use the Lime package to explore the features that lead to each classification.

I implemented Lime with information from this article.

```
# Get random image and label
label = y_train[2]
img = X_train[2]

img = img.astype('double')

# Get model pred
pred = mod5.predict(np.array([img]))
pred_class = int(pred.round())

# Print true class, predicted class and image
print('True Class:', label)
print('Predicted Class:', pred_class)
array_to_img(img)
```

```
    1/1 [==============================] - 0s 98ms/step
    True Class: 1.0
    Predicted Class: 1
```



```
# Make an explainer
```

```
explainer = lime_image.LimeImageExplainer()
```

```
explanation = explainer.explain_instance(img, mod5.predict, top_labels=2,
                                         hide_color=None, num_samples=2000)
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
```

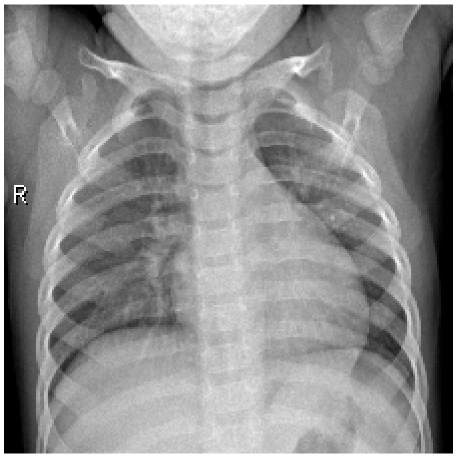
```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 35ms/step
```

```
    1/1 [==============================] - 0s 22ms/step
    1/1 [==============================] - 0s 20ms/step
```

```
  # Get random image and label
label = y_train[1]
img = X_train[1]

# Get model prediction
pred = mod5.predict(np.array([img]))
pred_class = int(pred.round())
```

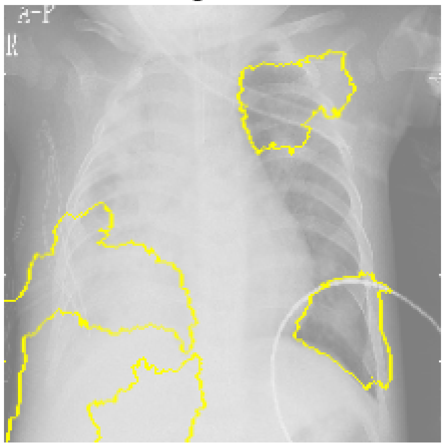
```
# Print true class, predicted class and image
print('True Class:', label)
print('Predicted Class:', pred_class)
array_to_img(img)
```

```
1/1 [==============================] - 0s 25ms/step
True Class: 0.0
Predicted Class: 0
```



```
# Use Lime to display features contributing to the pneumonia classification
temp, mask = explanation.get_image_and_mask(explanation.top_labels[0],
                                            positive_only=True,
                                            num_features=5)
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask))
plt.axis('off')
plt.title('Features Contributing to Normal Classification')
plt.show()
```

Features Contributing to Normal Classification



```
# Get a random image at index 2
label2 = y_train[2]
img2 = X_train[2]
img2 = img2.astype('double')

# Get model prediction
pred2 = mod5.predict(np.array([img2]))
```