1 X-Ray Classification using Convolutional Neural Networks (CNN)

Author: Eric Wehmueller

1.1 Overview

This project is the fourth project for Flatiron School's bootcamp program in Data Science. We are being placed into a hypothetical situation as a Data Scientist and hoping to provide value to our business for the scenario we are given.

1.2 Business Problem

A concern in the years 2020 and 2021 has been "flattening the curve" for as to not overwhelm the health care system in the United States, and in other countries as well. Using image analysis, classification, and convolutional neural networks (CNNs), the goal is to be able to correctly identify x-rays with a pneumonia diagnosis. There may only be so many radiologists available at any given time- the hope is that something as complicated as xrays, to the untrained human eye, can be correctly read and analyzed by a model generated from a trained CNN.

1.3 Setup and Preprocessing

```
import numpy as np
import matplotlib.pyplot as plt
import os
from PIL import Image

# tensorflow/keras libraries
import keras
import tensorflow as tf
from sklearn import metrics
from keras import optimizers
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confu
```

executed in 5.38s, finished 02:29:40 2021-05-24

```
#setup GPU support for tensor
print("Num GPUs: ", len(tf.config.experimental.list_phys
```

executed in 448ms, finished 02:29:40 2021-05-24

```
Num GPUs:  0
```

Let's quickly take a look at our file structure, and set up file paths for future steps.

```
images_home = "../xray-classification/chest_xray_images/
train_files = images_home+"train/"
test_files = images_home+"test/"
val_files = images_home+"val/"
```

executed in 12ms, finished 02:29:40 2021-05-24

```
print(os.listdir(train_files))
```

executed in 14ms, finished 02:29:40 2021-05-24

```
['NORMAL', 'PNEUMONIA']
```

```
train_norm = train_files+"NORMAL/"
train_sick = train_files+"PNEUMONIA/"
```

executed in 14ms, finished 02:29:40 2021-05-24

```
print(len(os.listdir(train_norm)))
print(len(os.listdir(train_sick)))
```

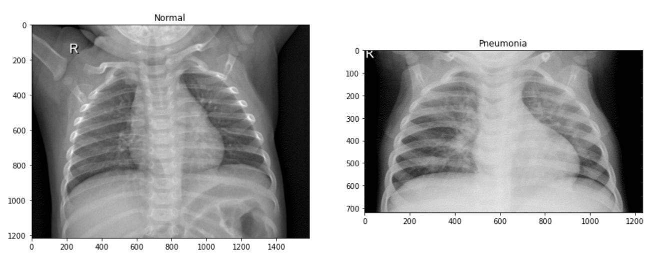
executed in 15ms, finished 02:29:40 2021-05-24

```
1042
3576
```

With ~1050 "normal" images and ~3600 pneumonia images, we may have a class imbalance to account for beyond our initial model.

```
norm_pic_file = os.listdir(train_norm)[40]
sick_pic_file = os.listdir(train_sick)[40]

norm_pic_full_filname = train_norm + norm_pic_file
sick_pic_full_filname = train_sick + sick_pic_file

pic_norm = Image.open(norm_pic_full_filname).convert('1'
pic_sick = Image.open(sick_pic_full_filname).convert('1'

f = plt.figure(figsize=(15,15))
a_norm = f.add_subplot(1,2,1)
img_plot = plt.imshow(pic_norm)
a_norm.set_title("Normal")

a_sick = f.add_subplot(1,2,2)
img_plot = plt.imshow(pic_sick)
a_sick.set_title("Pneumonia");
```

executed in 633ms, finished 02:29:41 2021-05-24



As we can see, there's going to be a size imbalance between all the picture sizes. We will standardize this soon, for our model to succeed.

1.4 Modeling

```
model = Sequential()

model.add(
    Conv2D(
        32,
        (3, 3),
        activation='relu',
        input_shape=(64,64,3),
        padding='same')
    )
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(activation = 'relu', units = 128))
model.add(Dense(activation='sigmoid', units = 1)) #2 res

model.summary()
```

executed in 121ms, finished 02:29:41 2021-05-24

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Par
am #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 30, 30, 64)        184
96
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 14400)             0
_________________________________________________________________
dense (Dense)                (None, 128)               184
3328
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
```

```
=================================================================
=======
Total params: 1,862,849
Trainable params: 1,862,849
Non-trainable params: 0
_________________________________________________________________
_______
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy',
               tf.keras.metrics.Precision(name='precisio
               tf.keras.metrics.Recall(name='recall')])

#binary_crossentroy due to the binary results
```

executed in 30ms, finished 02:29:41 2021-05-24

1.4.1 Image Data Generation

This will be used to add shear and zoom to bolter our training data, adding more images for our model to train on and learn from.

```
#model.fit(train_images, train_labels, epochs=2, batch_s
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = Tru

train_generator = train_datagen.flow_from_directory(trai
                                  target_size = (64,64),
                                  batch_size = 32,
                                  class_mode = 'binary')


test_datagen = ImageDataGenerator(rescale = 1./255)
validation_generator = test_datagen.flow_from_directory(
                                  target_size = (64,64),
                                  batch_size = 32,
                                  class_mode = 'binary')

test_generator = test_datagen.flow_from_directory(test_f
                                  target_size = (64,64),
                                  batch_size = 32,
                                  class_mode = 'binary')
```

executed in 293ms, finished 02:29:41 2021-05-24

```
Found 4616 images belonging to 2 classes.
Found 616 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

To note, the distribution between train/val/test is roughly 70-15-15% which is acceptable.

```
cnn_model = model.fit(train_generator,
                            steps_per_epoch = 80,
                            epochs = 10,
                            validation_data = validation_ge
                            validation_steps = 16)
```

executed in 5m 51s, finished 02:35:32 2021-05-24

```
Epoch 1/10
80/80 [==============================] - 48s 598ms/step -
loss: 0.4210 - accuracy: 0.8226 - precision: 0.8322 - reca
ll: 0.9648 - val_loss: 0.2421 - val_accuracy: 0.9082 - val
_precision: 0.9383 - val_recall: 0.8769
Epoch 2/10
80/80 [==============================] - 38s 480ms/step -
loss: 0.2379 - accuracy: 0.9012 - precision: 0.9279 - reca
ll: 0.9469 - val_loss: 0.1996 - val_accuracy: 0.9297 - val
_precision: 0.9026 - val_recall: 0.9602
Epoch 3/10
80/80 [==============================] - 35s 432ms/step -
loss: 0.2221 - accuracy: 0.9069 - precision: 0.9364 - reca
ll: 0.9440 - val_loss: 0.1579 - val_accuracy: 0.9531 - val
_precision: 0.9876 - val_recall: 0.9192
Epoch 4/10
80/80 [==============================] - 33s 408ms/step -
loss: 0.2176 - accuracy: 0.9121 - precision: 0.9347 - reca
ll: 0.9540 - val_loss: 0.1734 - val_accuracy: 0.9434 - val
_precision: 0.9712 - val_recall: 0.9147
Epoch 5/10
80/80 [==============================] - 33s 407ms/step -
loss: 0.2118 - accuracy: 0.9109 - precision: 0.9351 - reca
ll: 0.9504 - val_loss: 0.1389 - val_accuracy: 0.9609 - val
_precision: 0.9755 - val_recall: 0.9447
Epoch 6/10
80/80 [==============================] - 32s 399ms/step -
loss: 0.1847 - accuracy: 0.9234 - precision: 0.9493 - reca
ll: 0.9522 - val_loss: 0.1797 - val_accuracy: 0.9141 - val
_precision: 0.8750 - val_recall: 0.9646
Epoch 7/10
80/80 [==============================] - 32s 399ms/step -
loss: 0.1880 - accuracy: 0.9215 - precision: 0.9453 - reca
ll: 0.9535 - val_loss: 0.1222 - val_accuracy: 0.9570 - val
_precision: 0.9643 - val_recall: 0.9492
Epoch 8/10
80/80 [==============================] - 32s 405ms/step -
loss: 0.1488 - accuracy: 0.9422 - precision: 0.9592 - reca
ll: 0.9660 - val_loss: 0.1748 - val_accuracy: 0.9258 - val
_precision: 0.9094 - val_recall: 0.9508
Epoch 9/10
80/80 [==============================] - 32s 395ms/step -
loss: 0.1661 - accuracy: 0.9274 - precision: 0.9520 - reca
ll: 0.9558 - val_loss: 0.2400 - val_accuracy: 0.8984 - val
_precision: 0.8362 - val_recall: 0.9839
Epoch 10/10
80/80 [==============================] - 32s 396ms/step -
loss: 0.1470 - accuracy: 0.9452 - precision: 0.9613 - reca
```

```
ll: 0.9677 - val_loss: 0.1778 - val_accuracy: 0.9238 - val
_precision: 0.8877 - val_recall: 0.9731
```

```
test_acc = model.evaluate(test_generator, steps= 20)
```

executed in 8.71s, finished 02:35:41 2021-05-24

```
20/20 [==============================] - 8s 395ms/step - l
oss: 0.5944 - accuracy: 0.7724 - precision: 0.7340 - recal
l: 0.9974
```

An evaluation on our test_gen indicates that we might be slightly overfitting, due to the reduced accuracy here. However, we see a recall score of 99%, indicating we are doing well when correctly identifying actual pneumonia.

```
print('The testing accuracy is :',test_acc[1]*100, '%')
```

executed in 14ms, finished 02:35:41 2021-05-24

```
The testing accuracy is : 77.24359035491943 %
```

```
cnn_model.history
```

executed in 15ms, finished 02:35:41 2021-05-24

```
{'loss': [0.42098575830459595,
  0.23786011338233948,
  0.22213956713676453,
  0.21758008003234863,
  0.21182234585285187,
  0.18468818068504333,
  0.1879621297121048,
  0.14882703125476837,
  0.16606035828590393,
  0.14699338376522064],
 'accuracy': [0.8225551843643188,
  0.901171863079071,
  0.9069400429725647,
  0.9120662212371826,
  0.9108833074569702,
  0.9234374761581421,
  0.9215299487113953,
  0.942187488079071,
  0.9274448156356812,
  0.9451892971992493],
 'precision': [0.8321585655212402,
  0.9279058575630188,
  0.9363957643508911,
  0.9346534609794617,
  0.9350780248641968,
  0.9493226408958435,
  0.9452887773513794,
  0.9592145085334778,
  0.952023983001709,
  0.9612837433815002],
 'recall': [0.9647599458694458,
  0.946946918964386,
  0.9440203309059143,
  0.954017162322998,
  0.9503836035728455,
  0.9521892070770264,
  0.9535002708435059,
  0.9660243391990662,
  0.9558454751968384,
  0.9676923155784607],
 'val_loss': [0.24205166101455688,
  0.19961197674274445,
  0.15787260234355927,
  0.17344331741333008,
  0.13887637853622437,
  0.17971713840961456,
  0.12224026769399643,
  0.1748303323984146,
  0.2400045245885849,
  0.17784236371517181],
 'val_accuracy': [0.908203125,
  0.9296875,
  0.953125,
```

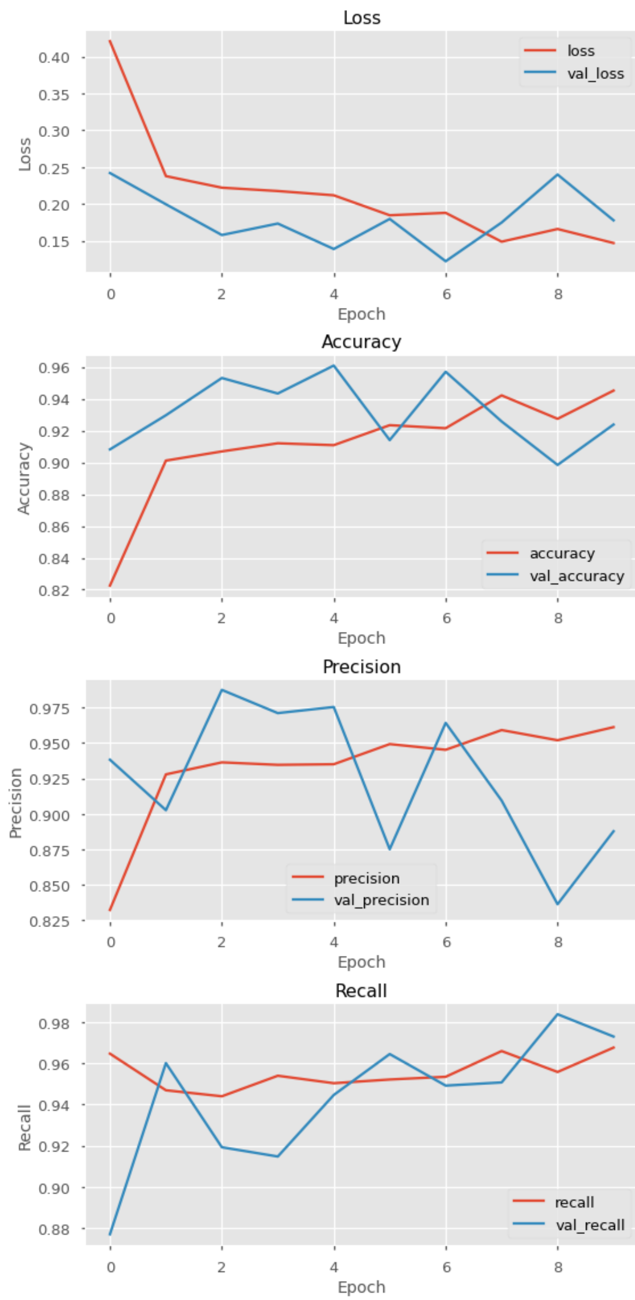
```
 0.943359375,
 0.9609375,
 0.9140625,
 0.95703125,
 0.92578125,
 0.8984375,
 0.923828125],
'val_precision': [0.9382715821266174,
 0.9026217460632324,
 0.9876033067703247,
 0.9711934328079224,
 0.9755101799964905,
 0.875,
 0.9642857313156128,
 0.9094203114509583,
 0.8361774682998657,
 0.8877192735671997],
'val_recall': [0.8769230842590332,
 0.9601593613624573,
 0.9192307591438293,
 0.9147287011146545,
 0.9446640610694885,
 0.9645669460296631,
 0.94921875,
 0.9507575631141663,
 0.9839357137680054,
 0.9730769395828247]}
```

```
def plot_history(history, style=['ggplot', 'seaborn-talk
    # We can pass in a model history object or a diction
    if not isinstance(history, dict): # We prefer this t
        history = history.history

    metrics_lst = [m for m in history.keys() if not m.st
    N = len(metrics_lst)
    with plt.style.context(style):
        fig, ax_lst = plt.subplots(nrows=N, figsize=(8,
        ax_lst = [ax_lst] if N == 1 else ax_lst.flatten(
        for metric, ax in zip(metrics_lst, ax_lst):
            val_m = f'val_{metric}'
            ax.plot(history[metric], label=metric)
            ax.plot(history[val_m], label=val_m)
            ax.set(title=metric.title(), xlabel='Epoch',
            ax.legend()
        fig.tight_layout()
        plt.show()
```

executed in 14ms, finished 02:35:41 2021-05-24

```
plot_history(cnn_model)
```

executed in 633ms, finished 02:35:42 2021-05-24



```
print(train_generator.class_indices)
test_images, test_labels = next(test_generator)
print(len(test_images), len(test_labels))
```

executed in 262ms, finished 02:35:42 2021-05-24

```
{'NORMAL': 0, 'PNEUMONIA': 1}
32 32
```

```
# y_hat = np.concatenate(model.predict(test_generator).r
# print(len(y_hat))
# report = metrics.classification_report(test_labels, y_
# print(report)
```

executed in 15ms, finished 02:35:42 2021-05-24

1.5 Model Iteration

Here I'm going to add more convulutional layers, a dropout, and a swish activation on our dense layer.

```
model2 = Sequential()

model2.add(
    Conv2D(
        32,
        (3, 3),
        activation='relu',
        input_shape=(64,64,3),
        padding='same')
    )
model2.add(MaxPooling2D((2, 2)))
model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(MaxPooling2D((2, 2)))

model2.add(Conv2D(96, (3, 3), dilation_rate=(2, 2), acti
model2.add(Conv2D(96, (3, 3), padding="valid", activatio
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Conv2D(128, (3, 3), dilation_rate=(2, 2), act
model2.add(Conv2D(128, (3, 3), padding="valid", activati
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Flatten())

model2.add(Dense(activation = 'swish', units = 128))
model2.add(keras.layers.Dropout(0.4))
model2.add(Dense(activation='sigmoid', units = 1)) #2 re

model2.summary()
```

executed in 108ms, finished 02:35:42 2021-05-24

```
Model: "sequential_1"
_________________________________________________________________
_______
Layer (type)                 Output Shape              Par
am #
=================================================================
=======
conv2d_2 (Conv2D)            (None, 64, 64, 32)        896
_________________________________________________________________
_______
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
_______
conv2d_3 (Conv2D)            (None, 30, 30, 64)        184
96
_________________________________________________________________
_______
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 64)        0
```

```
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 96)        553
92
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 13, 13, 96)        830
40
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 96)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 6, 6, 128)         110
720
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 128)         147
584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               656
64
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 481,921
Trainable params: 481,921
Non-trainable params: 0
_________________________________________________________________
```

```
model2.compile(optimizer='adam',
               loss='binary_crossentropy',
               metrics=['accuracy',
                tf.keras.metrics.Precision(name='precisio
                tf.keras.metrics.Recall(name='recall')])
```

executed in 30ms, finished 02:35:42 2021-05-24

```
cnn_model2 = model2.fit(train_generator,
                        steps_per_epoch = 50,
                        epochs = 15,
                        validation_data = validation_ge
                        validation_steps = 16)
```

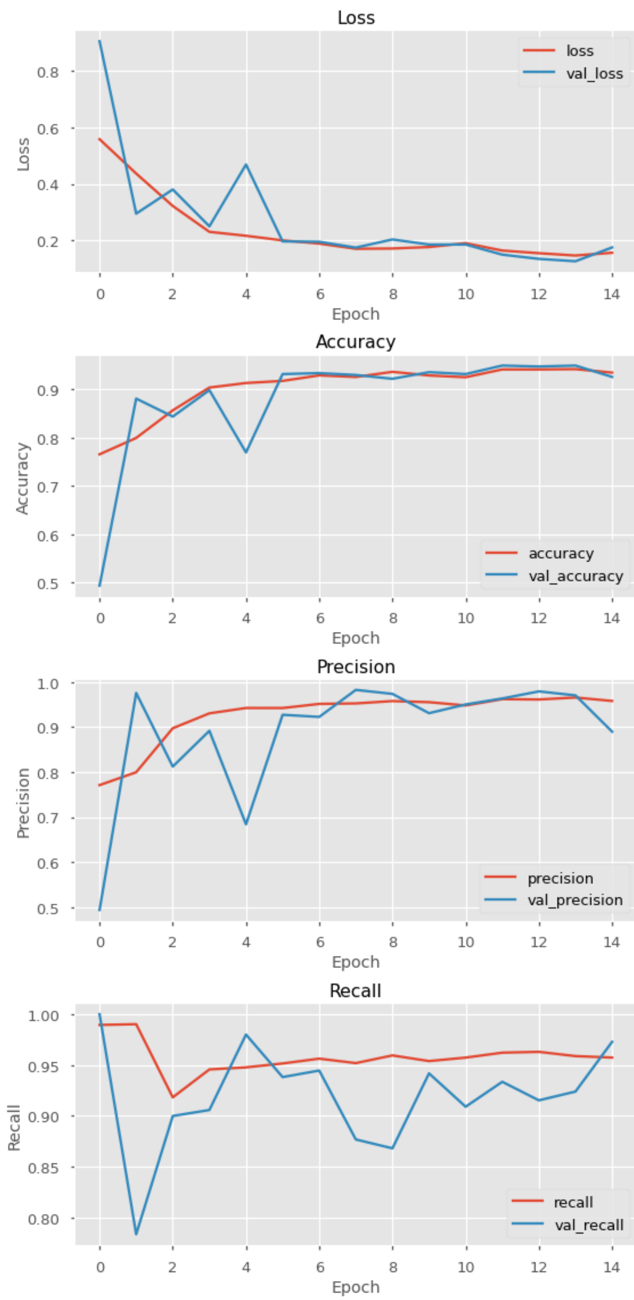
executed in 5m 47s, finished 02:41:30 2021-05-24

```
Epoch 1/15
50/50 [==============================] - 24s 482ms/step -
loss: 0.5582 - accuracy: 0.7656 - precision: 0.7713 - reca
ll: 0.9895 - val_loss: 0.9051 - val_accuracy: 0.4941 - val
_precision: 0.4941 - val_recall: 1.0000
Epoch 2/15
50/50 [==============================] - 23s 460ms/step -
loss: 0.4366 - accuracy: 0.7995 - precision: 0.7999 - reca
ll: 0.9902 - val_loss: 0.2946 - val_accuracy: 0.8809 - val
_precision: 0.9760 - val_recall: 0.7838
Epoch 3/15
50/50 [==============================] - 23s 465ms/step -
loss: 0.3218 - accuracy: 0.8566 - precision: 0.8976 - reca
ll: 0.9183 - val_loss: 0.3801 - val_accuracy: 0.8438 - val
_precision: 0.8125 - val_recall: 0.9000
Epoch 4/15
50/50 [==============================] - 23s 470ms/step -
loss: 0.2302 - accuracy: 0.9038 - precision: 0.9307 - reca
ll: 0.9457 - val_loss: 0.2497 - val_accuracy: 0.8984 - val
_precision: 0.8919 - val_recall: 0.9059
Epoch 5/15
50/50 [==============================] - 23s 466ms/step -
loss: 0.2162 - accuracy: 0.9131 - precision: 0.9426 - reca
ll: 0.9478 - val_loss: 0.4686 - val_accuracy: 0.7695 - val
_precision: 0.6844 - val_recall: 0.9800
Epoch 6/15
50/50 [==============================] - 23s 450ms/step -
loss: 0.1998 - accuracy: 0.9175 - precision: 0.9425 - reca
ll: 0.9516 - val_loss: 0.1963 - val_accuracy: 0.9316 - val
_precision: 0.9275 - val_recall: 0.9382
Epoch 7/15
50/50 [==============================] - 22s 445ms/step -
loss: 0.1885 - accuracy: 0.9289 - precision: 0.9516 - reca
ll: 0.9563 - val_loss: 0.1950 - val_accuracy: 0.9336 - val
_precision: 0.9228 - val_recall: 0.9447
Epoch 8/15
50/50 [==============================] - 22s 444ms/step -
loss: 0.1702 - accuracy: 0.9256 - precision: 0.9527 - reca
ll: 0.9520 - val_loss: 0.1743 - val_accuracy: 0.9297 - val
_precision: 0.9828 - val_recall: 0.8769
Epoch 9/15
50/50 [==============================] - 22s 446ms/step -
loss: 0.1711 - accuracy: 0.9362 - precision: 0.9580 - reca
ll: 0.9595 - val_loss: 0.2032 - val_accuracy: 0.9219 - val
_precision: 0.9739 - val_recall: 0.8682
Epoch 10/15
50/50 [==============================] - 22s 449ms/step -
loss: 0.1763 - accuracy: 0.9287 - precision: 0.9555 - reca
```

```
ll: 0.9540 - val_loss: 0.1851 - val_accuracy: 0.9355 - val
_precision: 0.9310 - val_recall: 0.9419
Epoch 11/15
50/50 [==============================] - 22s 436ms/step -
loss: 0.1899 - accuracy: 0.9251 - precision: 0.9482 - reca
ll: 0.9574 - val_loss: 0.1852 - val_accuracy: 0.9316 - val
_precision: 0.9504 - val_recall: 0.9091
Epoch 12/15
50/50 [==============================] - 22s 448ms/step -
loss: 0.1638 - accuracy: 0.9413 - precision: 0.9622 - reca
ll: 0.9622 - val_loss: 0.1494 - val_accuracy: 0.9492 - val
_precision: 0.9637 - val_recall: 0.9336
Epoch 13/15
50/50 [==============================] - 23s 451ms/step -
loss: 0.1545 - accuracy: 0.9413 - precision: 0.9615 - reca
ll: 0.9630 - val_loss: 0.1344 - val_accuracy: 0.9473 - val
_precision: 0.9794 - val_recall: 0.9154
Epoch 14/15
50/50 [==============================] - 22s 448ms/step -
loss: 0.1463 - accuracy: 0.9419 - precision: 0.9659 - reca
ll: 0.9588 - val_loss: 0.1259 - val_accuracy: 0.9492 - val
_precision: 0.9706 - val_recall: 0.9240
Epoch 15/15
50/50 [==============================] - 22s 442ms/step -
loss: 0.1561 - accuracy: 0.9346 - precision: 0.9582 - reca
ll: 0.9574 - val_loss: 0.1748 - val_accuracy: 0.9258 - val
_precision: 0.8901 - val_recall: 0.9729
```

```
plot_history(cnn_model2)
```

executed in 556ms, finished 02:41:30 2021-05-24



```
test_acc2 = model2.evaluate(test_generator, steps= 20)
```

executed in 5.41s, finished 02:41:36 2021-05-24

```
20/20 [==============================] - 5s 245ms/step - l
oss: 0.4880 - accuracy: 0.8285 - precision: 0.7918 - recal
l: 0.9846
```

These graphs look like a solid improvement from our initial model. Our loss is continually decreasing towards the end of the epochs, and our precision reaches 94% by the 11th epoch. Recall doesn't seem to improve all that much but sits at a respectable 96% by the 15th epoch. Overfitting may still be an issue, due to the lower values across the board for accuracy, precision, and recall on our model's evaluation on the test data set.

```
confidence_array = model2.predict(test_generator)
```

executed in 5.59s, finished 02:41:41 2021-05-24

```python
print(confidence_array)
```

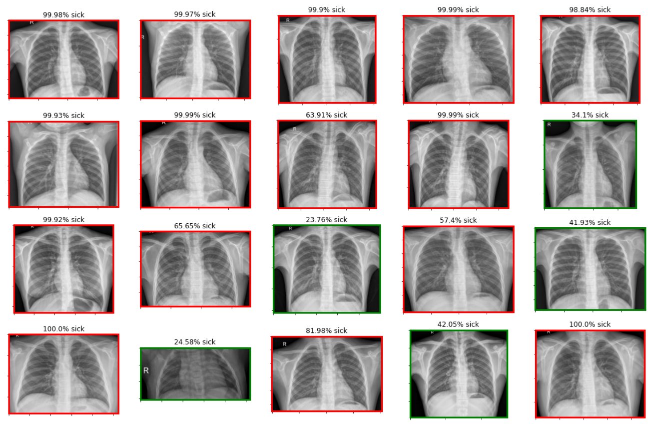
executed in 15ms, finished 02:41:41 2021-05-24

```
[0.6565139 ]
[0.23760834]
[0.5739843 ]
[0.4192769 ]
[0.9999622 ]
[0.24582711]
[0.81980044]
[0.42054802]
[0.9999591 ]
[0.9964088 ]
[0.98785746]
[0.9694909 ]
[0.92280984]
[0.296581  ]
[0.9995308 ]
[0.43862346]
[0.4114322 ]
[0.9999864 ]
[0.9999311 ]
[0.999901  ]
[0.13386405]
[0.99920285]
```

```
#f = plt.figure(figsize=(15,15))

w = 10
h = 10
fig = plt.figure(figsize=(20, 13))
columns = 5
rows = 4

# prep (x,y) for extra plotting
xs = np.linspace(0, 2*np.pi, 60)  # from 0 to 2pi
ys = np.abs(np.sin(xs))           # absolute of sine

# ax enables access to manipulate each of subplots
ax = []

index = 0
for filename in test_generator.filepaths[:20]:
    img = Image.open(filename).convert('1')
    # create subplot and append to ax
    ax.append( fig.add_subplot(rows, columns, index+

    sick_pct = round(confidence_array[index][0]*100,
    string_title = ""
    if (sick_pct > 50):
        string_title = str(sick_pct)+"% sick"
        ax[-1].spines['bottom'].set_color('red')
        ax[-1].spines['top'].set_color('red')
        ax[-1].spines['right'].set_color('red')
        ax[-1].spines['left'].set_color('red')
    else:
        string_title = str(sick_pct)+"% sick"
        ax[-1].spines['bottom'].set_color('green')
        ax[-1].spines['top'].set_color('green')
        ax[-1].spines['right'].set_color('green')
        ax[-1].spines['left'].set_color('green')

    for axis in ['top','bottom','left','right']:
      ax[-1].spines[axis].set_linewidth(3)

    ax[-1].set_title(string_title)  # set title
    ax[-1].set_yticklabels([])
    ax[-1].set_xticklabels([])

    plt.imshow(img)
    index= index+1
```

```python
    plt.show()  # finally, render the plot
```

executed in 6.93s, finished 04:06:39 2021-05-24




```
Let's iterate on our model one last time to look for
more improvements.
```

Let's iterate on our model one last time to look for more improvements.

```
model3 = Sequential()

model3.add(
    Conv2D(
        32,
        (3, 3),
        activation='relu',
        input_shape=(64,64,3),
        padding='same')
    )
model3.add(MaxPooling2D((2, 2)))
model3.add(Conv2D(64, (3, 3), activation='relu'))
model3.add(MaxPooling2D((2, 2)))

model3.add(Conv2D(128, (3, 3), dilation_rate=(2, 2), act
model3.add(Conv2D(128, (3, 3), padding="valid", activati
model3.add(MaxPooling2D(pool_size=(2, 2)))

model3.add(Flatten())

model2.add(Dense(activation = 'swish', units = 128))
model3.add(keras.layers.Dropout(0.4))
model3.add(Dense(activation='sigmoid', units = 1)) # 2 r

model3.summary()
```

executed in 97ms, finished 03:49:28 2021-05-24

```
Model: "sequential_3"
_________________________________________________________________
_______
Layer (type)                 Output Shape              Par
am #
=================================================================
=======
conv2d_12 (Conv2D)           (None, 64, 64, 32)        896
_________________________________________________________________
_______
max_pooling2d_9 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
_______
conv2d_13 (Conv2D)           (None, 30, 30, 64)        184
96
_________________________________________________________________
_______
max_pooling2d_10 (MaxPooling (None, 15, 15, 64)        0
_________________________________________________________________
_______
conv2d_14 (Conv2D)           (None, 15, 15, 128)       738
56
_________________________________________________________________
```

```
_______
conv2d_15 (Conv2D)           (None, 13, 13, 128)       147
584
_________________________________________________________
_______
max_pooling2d_11 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________
_______
flatten_3 (Flatten)          (None, 4608)              0
_________________________________________________________
_______
dropout_2 (Dropout)          (None, 4608)              0
_________________________________________________________
_______
dense_7 (Dense)              (None, 1)                 460
9
=========================================================
=======
Total params: 245,441
Trainable params: 245,441
Non-trainable params: 0
_________________________________________________________
_______
```

```python
model3.compile(optimizer='adam',
               loss='binary_crossentropy',
               metrics=['accuracy',
                tf.keras.metrics.Precision(name='precisio
                tf.keras.metrics.Recall(name='recall')])
```

executed in 17ms, finished 03:49:29 2021-05-24

```python
train_datagen_final = ImageDataGenerator(rescale = 1./25
                                        rotation_range=4
                                        width_shift_rang
                                        height_shift_ran
                                        shear_range=0.2,
                                        zoom_range=0.2,
                                        horizontal_flip=
                                        fill_mode='neare

train_generator_final = train_datagen_final.flow_from_di
                                    target_size = (64,64),
                                    batch_size = 32,
                                    class_mode = 'binary')
```

executed in 229ms, finished 03:49:31 2021-05-24

```
Found 4616 images belonging to 2 classes.
```

```
cnn_model3 = model3.fit(train_generator_final,
                        steps_per_epoch = 50,
                        epochs = 15,
                        validation_data = validation_ge
                        validation_steps = 16)
```

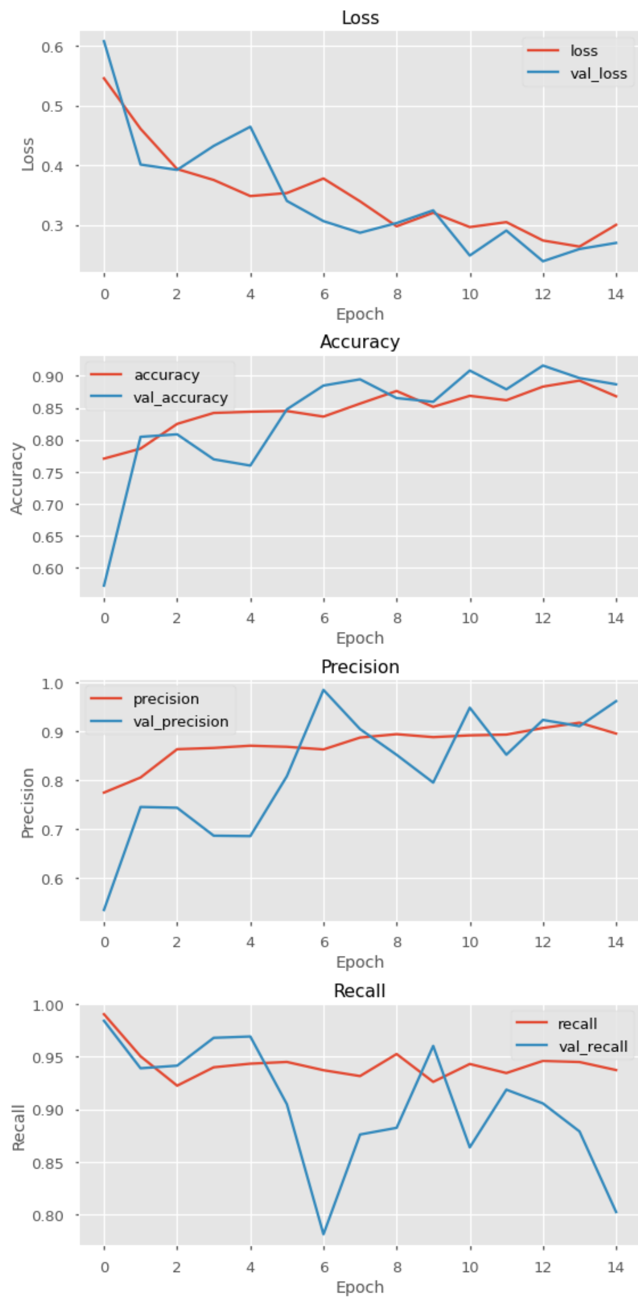
executed in 5m 45s, finished 03:55:17 2021-05-24

```
Epoch 1/15
50/50 [==============================] - 23s 464ms/step -
loss: 0.5457 - accuracy: 0.7706 - precision: 0.7749 - reca
ll: 0.9903 - val_loss: 0.6078 - val_accuracy: 0.5723 - val
_precision: 0.5346 - val_recall: 0.9841
Epoch 2/15
50/50 [==============================] - 23s 456ms/step -
loss: 0.4611 - accuracy: 0.7862 - precision: 0.8058 - reca
ll: 0.9503 - val_loss: 0.4012 - val_accuracy: 0.8047 - val
_precision: 0.7455 - val_recall: 0.9389
Epoch 3/15
50/50 [==============================] - 23s 451ms/step -
loss: 0.3936 - accuracy: 0.8249 - precision: 0.8637 - reca
ll: 0.9224 - val_loss: 0.3923 - val_accuracy: 0.8086 - val
_precision: 0.7438 - val_recall: 0.9414
Epoch 4/15
50/50 [==============================] - 22s 443ms/step -
loss: 0.3754 - accuracy: 0.8420 - precision: 0.8664 - reca
ll: 0.9399 - val_loss: 0.4325 - val_accuracy: 0.7695 - val
_precision: 0.6866 - val_recall: 0.9679
Epoch 5/15
50/50 [==============================] - 22s 444ms/step -
loss: 0.3484 - accuracy: 0.8439 - precision: 0.8709 - reca
ll: 0.9433 - val_loss: 0.4645 - val_accuracy: 0.7598 - val
_precision: 0.6858 - val_recall: 0.9691
Epoch 6/15
50/50 [==============================] - 23s 454ms/step -
loss: 0.3534 - accuracy: 0.8450 - precision: 0.8687 - reca
ll: 0.9449 - val_loss: 0.3403 - val_accuracy: 0.8477 - val
_precision: 0.8085 - val_recall: 0.9048
Epoch 7/15
50/50 [==============================] - 23s 454ms/step -
loss: 0.3778 - accuracy: 0.8363 - precision: 0.8633 - reca
ll: 0.9371 - val_loss: 0.3064 - val_accuracy: 0.8848 - val
_precision: 0.9852 - val_recall: 0.7812
Epoch 8/15
50/50 [==============================] - 22s 447ms/step -
loss: 0.3396 - accuracy: 0.8566 - precision: 0.8876 - reca
ll: 0.9315 - val_loss: 0.2869 - val_accuracy: 0.8945 - val
_precision: 0.9050 - val_recall: 0.8760
Epoch 9/15
50/50 [==============================] - 22s 449ms/step -
loss: 0.2977 - accuracy: 0.8763 - precision: 0.8945 - reca
ll: 0.9525 - val_loss: 0.3032 - val_accuracy: 0.8652 - val
_precision: 0.8523 - val_recall: 0.8824
Epoch 10/15
50/50 [==============================] - 22s 449ms/step -
```

```
loss: 0.3204 - accuracy: 0.8515 - precision: 0.8885 - reca
ll: 0.9260 - val_loss: 0.3244 - val_accuracy: 0.8594 - val
_precision: 0.7954 - val_recall: 0.9602
Epoch 11/15
50/50 [==============================] - 23s 451ms/step -
loss: 0.2964 - accuracy: 0.8687 - precision: 0.8921 - reca
ll: 0.9430 - val_loss: 0.2488 - val_accuracy: 0.9082 - val
_precision: 0.9487 - val_recall: 0.8638
Epoch 12/15
50/50 [==============================] - 22s 438ms/step -
loss: 0.3047 - accuracy: 0.8619 - precision: 0.8936 - reca
ll: 0.9344 - val_loss: 0.2906 - val_accuracy: 0.8789 - val
_precision: 0.8525 - val_recall: 0.9186
Epoch 13/15
50/50 [==============================] - 23s 455ms/step -
loss: 0.2740 - accuracy: 0.8831 - precision: 0.9072 - reca
ll: 0.9460 - val_loss: 0.2392 - val_accuracy: 0.9160 - val
_precision: 0.9237 - val_recall: 0.9055
Epoch 14/15
50/50 [==============================] - 23s 455ms/step -
loss: 0.2638 - accuracy: 0.8925 - precision: 0.9180 - reca
ll: 0.9448 - val_loss: 0.2597 - val_accuracy: 0.8965 - val
_precision: 0.9109 - val_recall: 0.8789
Epoch 15/15
50/50 [==============================] - 23s 454ms/step -
loss: 0.3002 - accuracy: 0.8680 - precision: 0.8958 - reca
ll: 0.9372 - val_loss: 0.2699 - val_accuracy: 0.8867 - val
_precision: 0.9621 - val_recall: 0.8024
```

```
plot_history(cnn_model3)
```

executed in 599ms, finished 03:56:13 2021-05-24



For this model, we drastically changed the training image generator to give our model many more varieties of pictures to learn on. Additionally, we removed a redundant layer of back to back convolutional layers seen in our previous model. This appears to have lowered the accuracy, but letting this model train for many more epochs may potentially solve this issue. Still, a 88% accuracy model is still respectable. Additionally, the recall numbers approach 95%, which is arguably the more important stat to take away. We are looking to correctly identify cases where pneumonia is detected in patients who will need help as soon as possible.

The curves on the history graphs are not nearly as smooth as model 2, but as stated above, this could be potentially remedied by letting the model run for many more epochs. This model is just learning slower than more previous iterations but that does not necessarily make it poor. The image datagen it was given to train on is likely the main contributing factor to the accuracy drops, but the model would likely be able to extrapolate to new x-rays more effectively.

1.6 Conclusion

Although our model has some overfitting issues due to the lower scores on the test data, we have a solid result. Essentially, overfitting can occur when a model becomes TOO good at getting solid accuracy numbers and other favorable statistics ONLY for the train/test images. This could hinder it's ability to extrapolate what it has learned to new x-ray images and provide incorrect results outside of this data set. Unfortunately, our precision was not climbing very much at all; but it appears that our recall was tending upwards. This means that as the model was training for more epochs, it was getting better at correctly identifying patients with pneumonia but not becoming significantly better at identifying "normal" cases correctly.

Our recall was consistently near 97% the end of the model's runtime. This is probably the most relevant in the context of our project- this is the percentage of actual positives that were correctly classified. In a situation like this, we want to find all patients who actually do have pneumonia so that they are placed into necessary care as soon as possible. This makes it the standout statistic of the bunch in our outputs.

1.7 Future Work

Good options moving forward would be to experiment with dense layers, adding more to have a more stable upwards curve while letting our model run for more epochs. Additionally, this dataset did also have an issue with class distribution, as we had many more available xrays of patients WITH pneumonia than without. Accounting for this in a future model would be ideal.