NOTE Run all cells in section 2, then skip to sections 5 for data loading (all preprocessing is done) and modeling helpers. Then, be sure to run the first cell in each other section to ensure the data is formatted correctly for that model type

pneumonia image

Business Problem

Pneumonia is a common infection that causes inflammation and possible fluid accumulation in the air sacs of the lungs. In China, pneumonia is one of the leading causes of death for children under 5 years old The infection is commonly caused by bactria and viruses, but can also be caused by fungal sources.

Pediatric pneumonia is initially diagnosed based on the time of the year and the results of a physical exam, paying attention the child's breathing and listening to the lungs. Physical symptoms associated with pneumonia generally include fever, rapid breathing, and increased heart rate. A further step in diagnosis would be to use chest X-rays; pneumonia is not always seen on x-rays, either because the disease is in its initial stages or it involves a part of the lung not easily seen by X-ray.. Inconclusive initial testing can result in additional blood tests to confirm or rule out the presence of infection.

As we can see, even with modern medicine pneumonia can be misdiagnosed. Not all pneumonia infections are treated in the same way; viral infections cannot be fought with the same antibiotics that would treat a bacterial infection. A fast and accurate diagnosis allows doctors to treat the infection with the appropriate care. One application of machine learning in medicine is digital diagnosis. We have been tasked with developing an identification model to determine if a chest X-ray indicates the presence of pneumonia. False negative results are to be minimized compared to false positives.

The data is sourced from Kaggle. It is already split into three folders for training, validation and testing. All the chest radiographs were screened for quality and diagnostic labeling performed by physicians. The images were collected during routine clinicial care of pediatric patients between one and five years old from Guangzhou Women and Children's Medical Center in Guangzhou, China.

Imports

In [1]:

```
import os
from os import listdir
from os.path import isfile, join
import tempfile

from PIL import Image
import math

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

import sklearn
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.utils.class_weight import compute_class_weight

import tensorflow as tf
from tensorflow import keras

from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten
```

In [2]:

```
# set figsize for matplotlib
mpl.rcParams['figure.figsize'] = (12, 10)
# set colors for plots
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
```

EDA

Helper Functions

Helper functions:

- get_ratio: return the ratio of an 2D image array width:height
- resize_and_crop: returns a scaled and cropped image

In [3]:

```
def get_ratio(array):
    return array.shape[0]/array.shape[1]

def resize_and_crop(image, new_size=256):
    """
    This function will take in an image, resize and crop it to a square.
    The default new length for a side is 256 pixels.
    The function will output a new square, centered image.

    """
    # get width and height from passed image
    width, height = image.size
    # set up variables for new width and height
    new_width = 0
    new_height = 0
    # calculate the ratio and length of the smaller dimension if the larger moves to th
    if width > height:
        ratio_wh = width / height
        new_width = int(ratio_wh * new_size)
        new_height = new_size
    else:
        ratio_hw = height / width
        new_width = new_size
        new_height = int(ratio_hw * new_size)
    # resize the image
    scaled_image = image.resize((new_width, new_height))
    # if the image is wider, crop in equally from the sides to preserve center of image
    # perform the opposite on images that are taller
    if new_width > new_height:
        # we are only cropping in from left and right, so set top crop to 0 and bottom
        top = 0
        bottom = new_size
        # set the left and right side crop values
        left = int(math.ceil((new_width - new_size) / 2))
        right = new_width - int(math.floor((new_width - new_size) / 2))
        # crop the image
        cropped_image = scaled_image.crop((left, top, right, bottom))
    else:
        # we are only cropping top and bottom, so set left crop to 0 and right crop to
        left = 0
        right = new_size
        # set the top and bottom crop values
        top = int(math.ceil((new_height - new_size) / 2))
        bottom = new_height - int(math.floor((new_height - new_size) / 2))
        # save the cropped image
        cropped_image = scaled_image.crop((left, top, right, bottom))
    # return the scaled and cropped image
    return cropped_image
```

Image size

First, we are going to build a dataframe that contains all the image information for us to look at image sizes.

In [4]:

```
folder_names = ['train', 'test', 'val']
label_names = ['NORMAL', 'PNEUMONIA']

all_images = []

# loop through the different combinations of folder name prefixes
for folder in folder_names:
    for label in label_names:
        # set up the path to each folder of images
        path = f'./chest_xray/{folder}/{label}'
        # create a list of the filenames in that directory
        filelist = list(listdir(path))
        # loop through each file in the folder
        for file_name in filelist:
            # set the filepath for the file in question
            filepath = path + r'/' + file_name
            # open the image
            image = Image.open(filepath)
            # save the image mode
            img_mode = image.mode
            # get the width and heigh of the image
            width, height = image.size
            # calculate the dimension ratio
            ratio = height / width
            # append the image infromation to the list all_images
            all_images.append((folder, label, file_name, img_mode, ratio, width, height

# set the column names
column_names = ['folder', 'label', 'file_name', 'img_mode','ratio', 'width', 'height']
# convert the list of images to a dataframe
all_images_df = pd.DataFrame(all_images, columns=column_names)
```

In [5]:

```
all_images_df.head()
```

Out[5]:

	folder	label	file_name	img_mode	ratio	width	height
0	train	NORMAL	IM-0115-0001.jpeg	L	0.888995	2090	1858
1	train	NORMAL	IM-0117-0001.jpeg	L	0.810127	1422	1152
2	train	NORMAL	IM-0119-0001.jpeg	L	0.792265	1810	1434
3	train	NORMAL	IM-0122-0001.jpeg	L	0.790482	1618	1279
4	train	NORMAL	IM-0125-0001.jpeg	L	0.703125	1600	1125

In [6]:

```
all_images_df.img_mode.value_counts()
```

Out[6]:

```
L      5573
RGB     283
Name: img_mode, dtype: int64
```

So it looks like despite our images being in black and white, we have 283 images that are coded as RGB images. We will need to convert them before making our image arrays otherwise our shapes will not match. Lets look at the sizes of our images.

In [7]:

```
all_images_df.describe()
```

Out[7]:

	ratio	width	height
count	5856.000000	5856.000000	5856.000000
mean	0.712905	1327.880806	970.689037
std	0.117312	363.500922	383.392117
min	0.295964	384.000000	127.000000
25%	0.630616	1056.000000	688.000000
50%	0.706272	1281.000000	888.000000
75%	0.792627	1560.000000	1187.000000
max	1.197044	2916.000000	2713.000000

We have a wide variety of image sizes. For best results in modeling it was advised to scale images to 256x256. We can see that there are some images whose height is below our minimum, lets look a little closer.

In [8]:

```
small_images_df = all_images_df[all_images_df.height < 256].copy()
```

In [9]:

```
small_image_count = len(small_images_df)

print(f'There are {small_image_count} images with a dimension under 256.')
print('-------------------------------')
print('Small images occur in the following data/labels:')
display(all_images_df[all_images_df.height < 256].folder.value_counts())
display(all_images_df[all_images_df.height < 256].label.value_counts())
```

```
There are 58 images with a dimension under 256.
-------------------------------
Small images occur in the following data/labels:
train    58
Name: folder, dtype: int64
PNEUMONIA    58
Name: label, dtype: int64
```

In [10]:

```
print('Target distribution in training data')
round(all_images_df[all_images_df.folder == 'train'].label.value_counts(normalize=True)
```

```
Target distribution in training data
```

Out[10]:

```
PNEUMONIA    74.29
NORMAL       25.71
Name: label, dtype: float64
```

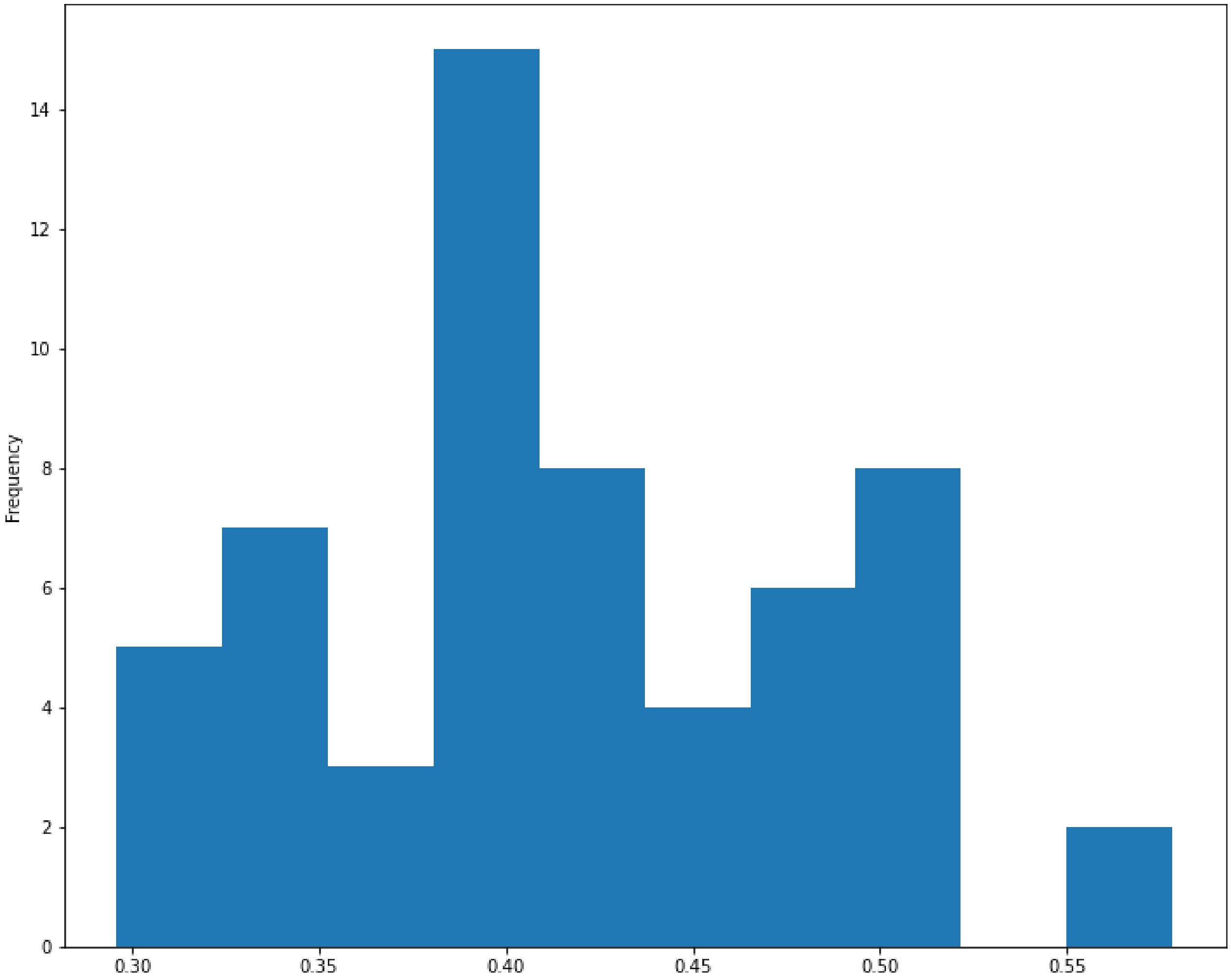
In [11]:

```
percent_small = round(small_image_count / len(all_images_df[(all_images_df.label == "PN
print(f'The {small_image_count} small images represent {percent_small}% of our training
```

```
The 58 small images represent 0.01% of our training data with the pneumonia label
```

In [12]:

```
small_images_df.ratio.plot(kind='hist');
```



We can see here that there are only 58 images out of our total 5856 which have a dimension less than our minimum. They are all from training images positive with pneumonia, where we have a large imbalance of that data. Additionally, the ratios of these images are small, meaning that they are significantly wider than they are tall. This would also pose a problem as our images are chest x-rays, and those wider images would likely get cut off when we resize them to square.

We will not consider these images in modeling so we will drop them from the dataframe.

In [13]:

```
images_df = all_images_df.drop(small_images_df.index)
images_df.reset_index(drop=True, inplace=True)
```

Since we discovered that the smaller images had very low ratios, let's consider the ratios of the all the rest of the images.

In [14]:

```
images_df.ratio.describe()
```

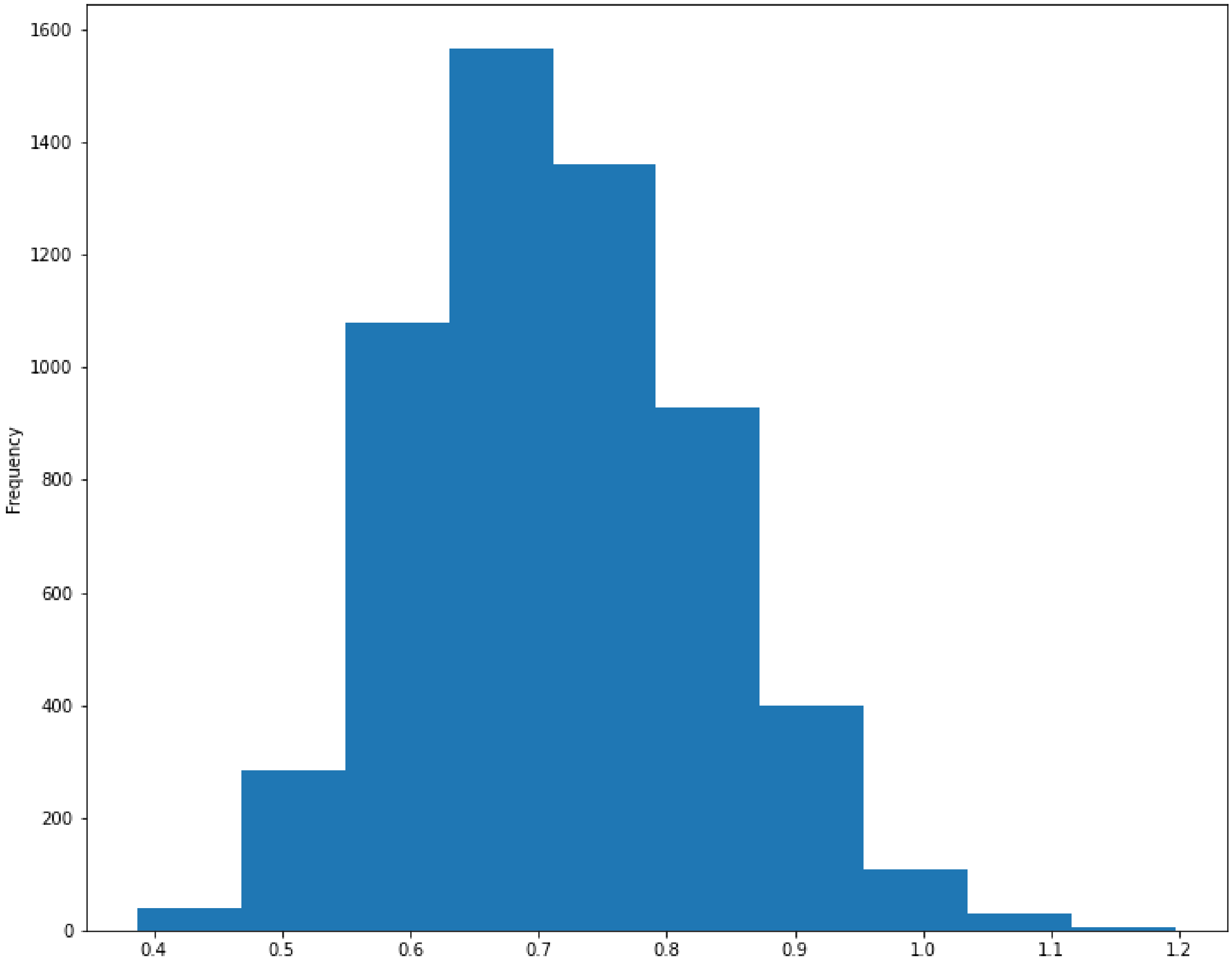
Out[14]:

```
count    5798.000000
mean        0.715844
std         0.113936
min         0.387387
25%         0.632822
50%         0.707828
```

```
75%         0.793645
max         1.197044
Name: ratio, dtype: float64
```

In [15]:

```
images_df.ratio.plot(kind='hist');
```



Most of our images (75%) have a ratio above 0.63, but the minimum is very low at 0.38. It's suspected that cropping the images that we have with low ratio (which indicates very wide images) may be problematic, so it may be something we have to return to and consider dropping some images with a ratio below a certain threshold. We can do some quick investigation into that.

Cropping

In [16]:

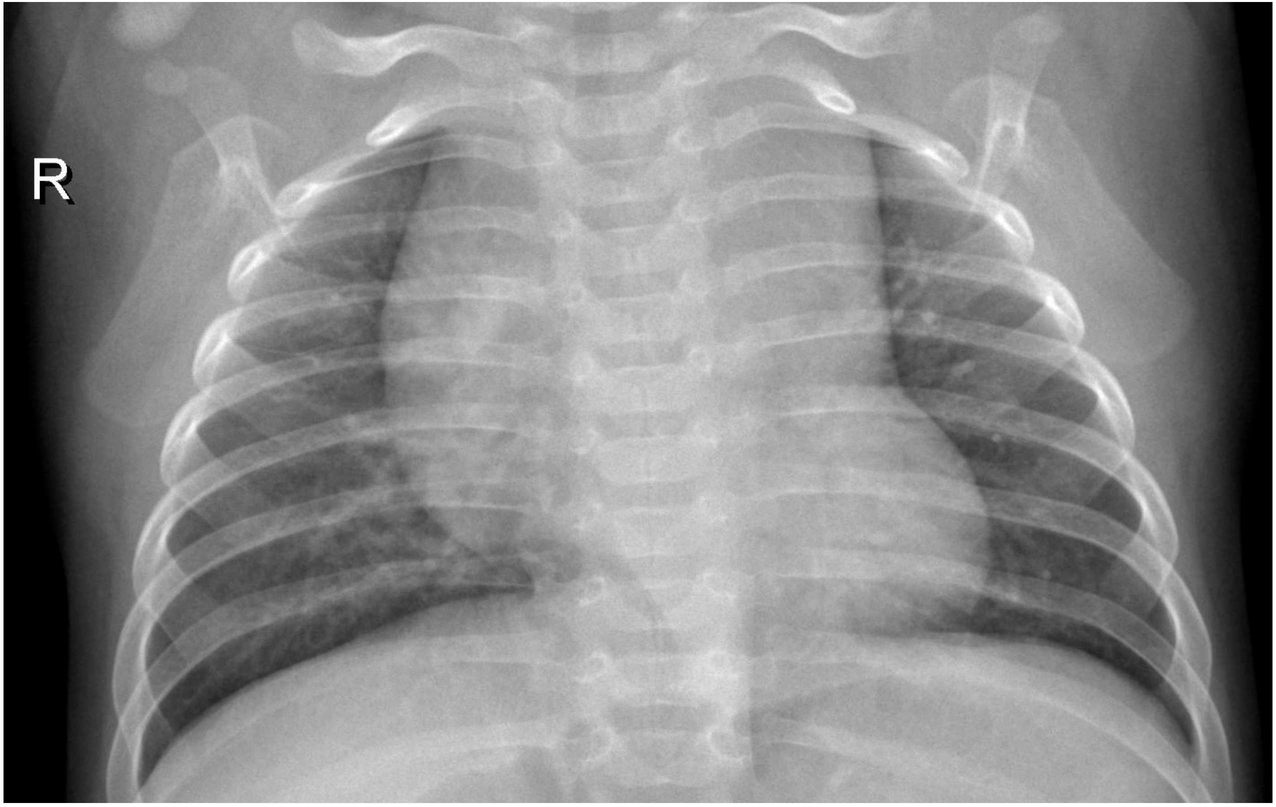
```
def compare_cropping(index_value, df=images_df):
    """
    This helper function will take in the index value of an image in the default
    dataframe: all_images_df. It will display the width, height, and ratio of the
    original with the original image. It will then perform the cropping and display
    the new imaged data along with the new cropped image for comparison.
    """
    # Get the folder and img_type value from the dataframe to contsruct the folder root
    folder = df.iloc[index_value]['folder']
    label = df.iloc[index_value]['label']
    # constrcut the folder root for the file in question
    root=f"./chest_xray/{folder}/{label}/"
    # set the image path and open the image
    image_path = root + df.iloc[index_value]['file_name']
    image = Image.open(image_path)
    # print the dimensions, ratio, and original image
    print(f"width: {image.width}\theight: {image.height}\tratio: {image.height/image.wi
    display(image)
    # crop the image
    cropped_image = resize_and_crop(image)
    # print the new image (will be 256x256 with ratio of 1.0)
    display(cropped_image)
```

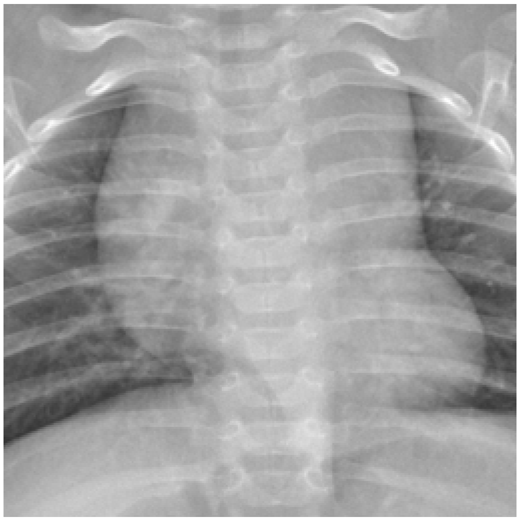
Let's look at an image with our 25th percentile value for ratio: 0.63

In [17]:

```
compare_cropping(images_df[images_df.ratio == 0.63].index[0])
```

```
width: 1400     height: 882     ratio: 0.63
```





Here we can see we are losing some of our image on the sides of the chest cavity. We know that data is expensive, so we want to use the most of our images that we can. For arguments' sake, if we were willing to lose the worst 5% of our images based on ratio, let's examine an image with the ratio of the 5% quantile.

In [18]:

```
images_df.quantile(q=0.05)
```

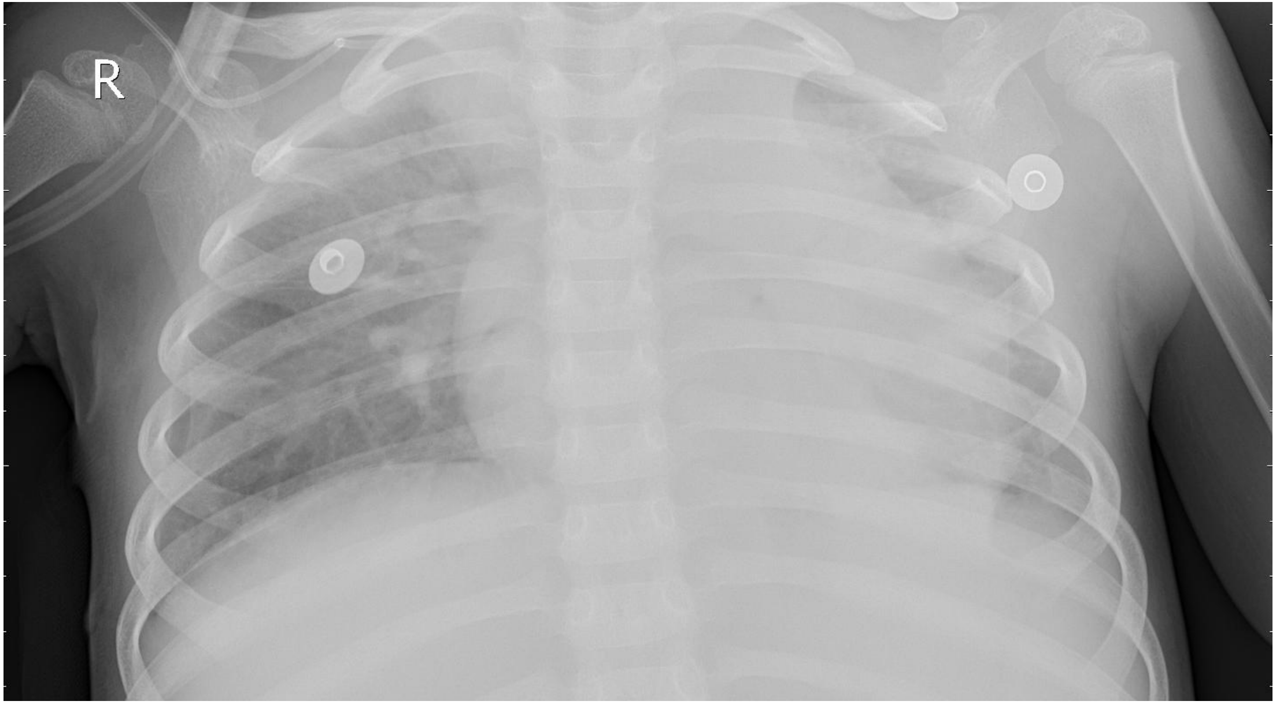
Out[18]:

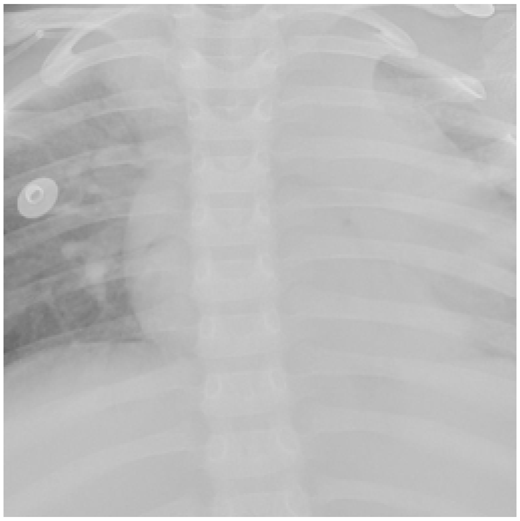
```
ratio       0.54482
width     860.85000
height    528.00000
Name: 0.05, dtype: float64
```

In [19]:

```
compare_cropping(images_df[images_df.ratio == 0.55].index[0])
```

```
width: 1440     height: 792     ratio: 0.55
```





We are definitely losing data by cropping images with ratios this low. Out of curiosity, let's see where most of these images are.

In [20]:

```
images_df[images_df.ratio <= 0.55].folder.value_counts()
```

Out[20]:

```
train    283
test      44
Name: folder, dtype: int64
```

In [21]:

```
images_df[images_df.ratio <= 0.55].label.value_counts()
```

Out[21]:

```
PNEUMONIA    321
NORMAL         6
Name: label, dtype: int64
```

If we see impact during modeling, we may consider dropping images with ratios less than 0.55 from consideration. Most of the images are from our training set of positive cases (which we have a large majority of).

Image Preprocessing

Now that we have a dataframe of the raw images data, we need to resize, shrink and convert the images to arrays for modeling. We will do that with a helper function applied to the dataframe and input the arrays into a new feature called 'image_array'.

In [22]:

```
def create_image_array(df):
    # set path for image
    path = path = f'./chest_xray/{df.folder}/{df.label}/{df.file_name}'
    # open image
    image = Image.open(path)
    # convert to greyscale
    if image.mode == 'RGB':
        image = image.convert('L')
    # resize and crop the image
    image_modified = resize_and_crop(image)
    # return the image as a numpy array
    return np.asarray(image_modified)

images_df['image_array'] = images_df.apply(create_image_array, axis=1)
```

We also need to convert the label to a binary value for our target.

In [23]:

```
def create_target(df):
    if df.label == 'PNEUMONIA':
        return 1
    else:
        return 0

images_df['target'] = images_df.apply(create_target, axis=1)
```

Now we can set a dataframe for each of our three datasets (train, test and validation)

In [24]:

```
train_df = images_df[images_df.folder == 'train'][['image_array', 'target']]
test_df = images_df[images_df.folder == 'test'][['image_array', 'target']]
val_df = images_df[images_df.folder == 'val'][['image_array', 'target']]
```

Then we split the image_array and target apart, save each datasets' X and y, shuffle the data, and check the shape of each array.

In [25]:

```
X_train = np.array(train_df.image_array.values.tolist())
y_train = np.asarray(train_df.target)

X_test = np.array(test_df.image_array.values.tolist())
y_test = np.asarray(test_df.target)

X_val = np.array(val_df.image_array.values.tolist())
y_val = np.asarray(val_df.target)

# helper function to shuffle data/target in the same way
def shuffle_arrays(data, target):
    assert len(data) == len(target)
    p = np.random.permutation(len(data))
    return data[p], target[p]
# shuffle all data
X_train, y_train = shuffle_arrays(X_train, y_train)
X_val, y_val = shuffle_arrays(X_val, y_val)
X_test, y_test = shuffle_arrays(X_test, y_test)

# verify shapes are accurate
print(f"Train data/target shapes: {X_train.shape}, {y_train.shape}")
print(f"Test data/target shapes: {X_test.shape}, {y_test.shape}")
print(f"Val data/target shapes: {X_val.shape}, {y_val.shape}")
```

```
Train data/target shapes: (5158, 256, 256), (5158,)
Test data/target shapes: (624, 256, 256), (624,)
Val data/target shapes: (16, 256, 256), (16,)
```

In [ ]:

```
# set the data path
data_path = f"./data/"
# list of variables we are saving
datasets = ['X_train', 'y_train', 'X_test', 'y_test', 'X_val', 'y_val']

for dataset in datasets:
    np.save(file=f'{data_path}{dataset}.npy', arr=globals()[dataset])
```

Data Loading

In [26]:

```
# set the data path and get the list of files
data_path = f"./data/"
data_files = list(listdir(data_path))

for file in data_files:
    # split off the components of the file name
    dataset = file[:-4]
    # set a global variable for each file with that files name and the data from the fi
    globals()[f"{dataset}"] = np.load(data_path + file)

# set the data types to floa
X_train_orig = X_train.astype('float32')
X_val_orig = X_val.astype('float32')
X_test_orig = X_test.astype('float32')
```

```
# normalize the data
X_train_orig /= 255.
X_val_orig /= 255.
X_test_orig /= 255.

# verify shapes are accurate
print(f"Train data/target shapes: {X_train_orig.shape}, {y_train.shape}")
print(f"Val data/target shapes: {X_val_orig.shape}, {y_val.shape}")
print(f"Test data/target shapes: {X_test_orig.shape}, {y_test.shape}")
```

```
Train data/target shapes: (5158, 256, 256), (5158,)
Val data/target shapes: (16, 256, 256), (16,)
Test data/target shapes: (624, 256, 256), (624,)
```

Modeling helpers

Global Variables

In [27]:

```
# setting up training data class imbalance for eventual weighting
neg, pos = np.bincount(y_train)
total = neg+pos

percent_neg = round(neg/total*100, 2)
percent_pos = round(pos/total*100, 2)

print(f'Training data contains:\nTotal: {total}\nNegative: {neg} ({percent_neg}%)\nPosi
```

```
Training data contains:
Total: 5158
Negative: 1341 (26.0%)
Positive: 3817 (74.0%)
```

In [28]:

```
# setting up metrics for model comparison
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'), # using binary accuracy
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]
```

In [29]:

```
# set up the class weights using y_train
classes = list(np.unique(y_train))
weights = list(compute_class_weight(class_weight='balanced', classes=np.unique(y_train)
class_weights = dict(zip(classes, weights))
class_weights
```

Out[29]:

```
{0: 1.9231916480238629, 1: 0.675661514278229}
```

Helper Functions

In [30]:

```
def set_x_y():
    # get user input for testing dataset
    print("Choose which testing dataset to use for reporting")
    print("Input S for training data\nInput T for test data\nInput V for validation dat
    test_choice = input()
    # set the correct global datasets to the local X_test and y_test for reporting
    if test_choice.lower() == 's':
        X_test, y_test = globals()['X_train'], globals()['y_train']
    elif test_choice.lower() == 't':
        X_test, y_test = globals()['X_test'], globals()['y_test']
    else:
        X_test, y_test = globals()['X_val'], globals()['y_val']
    # return the test target and label
    return X_test, y_test

def converted_ypred(model, X_test, threshold=0.5):
    # generate predictions
    y_pred = model.predict(X_test)
    # convert the predictions based on the threshold
    # default is: 0.5
    converted_ypred = []
    for i in range(len(y_pred)):
        if y_pred[i][0] > threshold:
            converted_ypred.append(1)
        else:
            converted_ypred.append(0)
    # return the converted y_preds as an array
    return np.asarray(converted_ypred)

def model_report(y_test, y_pred):
    # set up labels for classifcation report and confution matrix
    report_labels = ['normal', 'pneumonia']
    column_labels = ['predicted normal', 'predicted pneumonia']
    index_labels = ['actual normal', 'actual pneumonia']
    # generate the confusion matrix
    cmatrix = confusion_matrix(y_test, y_pred)
    # convert into dataframe
    cmatrix_df = pd.DataFrame(cmatrix, columns=column_labels, index=index_labels)
    print('---------------------------------------------------------------\n')
    # print the classification report
    print(classification_report(y_test, y_pred, zero_division=0, target_names=report_la
    # show report as heatmap
    fig, ax = plt.subplots(figsize=(8,6))
    ax = sns.heatmap(data=cmatrix_df, annot=True, cmap='Blues', fmt='g')
    locs, labels = plt.xticks()
    plt.setp(labels, rotation=0)
    plt.show()

def plot_metrics(results):
    # set the history
    history = results.history
    # set up metrics to plot
    metrics = ['loss', 'accuracy', 'precision', 'recall']
    # set up x-axis (number of epochs)
    epochs = list(range(1, len(history['loss'])+1))
    # generate plot for each of the 4 metrics in a 2x2 plot
    plt.figure(figsize=(15,15))
    for n, metric in enumerate(metrics):
        name = metric.replace("_"," ").capitalize()
        plt.subplot(2,2,n+1)
        plt.plot(epochs, history[metric], color=colors[0], label='Train')
        plt.plot(epochs, history['val_'+metric], color=colors[1], linestyle="--", label
        plt.xlabel('Epoch')
        plt.ylabel(name)
        plt.ylim([0,1.1])
    plt.legend();

def full_report(model, results=None):
    # set the X_train and y_train
    X_train, y_train = globals()['X_train'], globals()['y_train']
    # set the testing X and y based on input
    X_test, y_test = set_x_y()
    # create the y_pred
    y_pred = converted_ypred(model, X_test)
    # print the model report and confusion matrix
    model_report(y_test, y_pred)
    # display the 2x2 plot of metrics
    if results != None:
        plot_metrics(results)
```

Baseline model

Correcting data shape

In [31]:

```
# set the lengths of each set of data
train_len = X_train_orig.shape[0]
test_len = X_test_orig.shape[0]
val_len = X_val_orig.shape[0]

# set the new reshape size for the images
img_size = X_train_orig.shape[1] ** 2

# reshape all three data groups
X_train = X_train_orig.reshape(train_len, img_size).astype('float32')
X_test = X_test_orig.reshape(test_len, img_size).astype('float32')
X_val = X_val_orig.reshape(val_len, img_size).astype('float32')

# verify shapes are accurate
print(f"Train data/target shapes: {X_train.shape}, {y_train.shape}")
print(f"Test data/target shapes: {X_test.shape}, {y_test.shape}")
print(f"Val data/target shapes: {X_val.shape}, {y_val.shape}")
```

```
Train data/target shapes: (5158, 65536), (5158,)
Test data/target shapes: (624, 65536), (624,)
Val data/target shapes: (16, 65536), (16,)
```

MLP model

Inspiration from TensorFlow Tutorial to enact some changes to process and account for imbalanced data. Source

In [32]:

```
def make_baseline_model():
    # set up model
    model = keras.Sequential()
    model.add(Dense(128, activation='relu', input_shape = (X_train.shape[-1],)))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    # compile model
    model.compile(loss='binary_crossentropy',
                  optimizer='Adam',
                  metrics=METRICS)

    return model
```

In [33]:

```
baseline_model = make_baseline_model()
baseline_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               8388736   
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dropout_1 (Dropout)         (None, 64)                0         
                                                                 
 dense_2 (Dense)             (None, 32)                2080      
                                                                 
 dropout_2 (Dropout)         (None, 32)                0         
                                                                 
 dense_3 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 8,399,105
Trainable params: 8,399,105
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
EPOCHS = 5
BATCH_SIZE = 30

baseline_results = baseline_model.fit(X_train, y_train,
                                      batch_size=BATCH_SIZE,
                                      epochs=EPOCHS,
                                      class_weight=dict(enumerate(class_weights)),
                                      validation_data=(X_val, y_val))
```

```
Epoch 1/5
172/172 [==============================] - 8s 39ms/step - loss: 0.7830 - tp: 3650.0000 -
fp: 1300.0000 - tn: 41.0000 - fn: 167.0000 - accuracy: 0.7156 - precision: 0.7374 - reca
ll: 0.9562 - auc: 0.4954 - prc: 0.7382 - val_loss: 174.4564 - val_tp: 8.0000 - val_fp:
```

```
8.0000 - val_tn: 0.0000e+00 - val_fn: 0.0000e+00 - val_accuracy: 0.5000 - val_precision:
0.5000 - val_recall: 1.0000 - val_auc: 0.5000 - val_prc: 0.5000
Epoch 2/5
172/172 [==============================] - 6s 34ms/step - loss: 0.0826 - tp: 3801.0000 -
fp: 1336.0000 - tn: 5.0000 - fn: 16.0000 - accuracy: 0.7379 - precision: 0.7399 - recal
l: 0.9958 - auc: 0.4990 - prc: 0.7396 - val_loss: 191.2362 - val_tp: 8.0000 - val_fp: 8.
0000 - val_tn: 0.0000e+00 - val_fn: 0.0000e+00 - val_accuracy: 0.5000 - val_precision:
0.5000 - val_recall: 1.0000 - val_auc: 0.5000 - val_prc: 0.5000
Epoch 3/5
172/172 [==============================] - 6s 34ms/step - loss: 0.0116 - tp: 3810.0000 -
fp: 1339.0000 - tn: 2.0000 - fn: 7.0000 - accuracy: 0.7390 - precision: 0.7399 - recall:
0.9982 - auc: 0.4999 - prc: 0.7400 - val_loss: 211.7686 - val_tp: 8.0000 - val_fp: 8.000
0 - val_tn: 0.0000e+00 - val_fn: 0.0000e+00 - val_accuracy: 0.5000 - val_precision: 0.50
00 - val_recall: 1.0000 - val_auc: 0.5000 - val_prc: 0.5000
Epoch 4/5
172/172 [==============================] - 6s 34ms/step - loss: 0.0848 - tp: 3806.0000 -
fp: 1337.0000 - tn: 4.0000 - fn: 11.0000 - accuracy: 0.7387 - precision: 0.7400 - recal
l: 0.9971 - auc: 0.4999 - prc: 0.7400 - val_loss: 251.9705 - val_tp: 8.0000 - val_fp: 8.
0000 - val_tn: 0.0000e+00 - val_fn: 0.0000e+00 - val_accuracy: 0.5000 - val_precision:
0.5000 - val_recall: 1.0000 - val_auc: 0.5000 - val_prc: 0.5000
Epoch 5/5
172/172 [==============================] - 6s 34ms/step - loss: 0.0233 - tp: 3814.0000 -
fp: 1339.0000 - tn: 2.0000 - fn: 3.0000 - accuracy: 0.7398 - precision: 0.7402 - recall:
0.9992 - auc: 0.5000 - prc: 0.7400 - val_loss: 247.2593 - val_tp: 8.0000 - val_fp: 8.000
0 - val_tn: 0.0000e+00 - val_fn: 0.0000e+00 - val_accuracy: 0.5000 - val_precision: 0.50
00 - val_recall: 1.0000 - val_auc: 0.5000 - val_prc: 0.5000
```

In [35]:

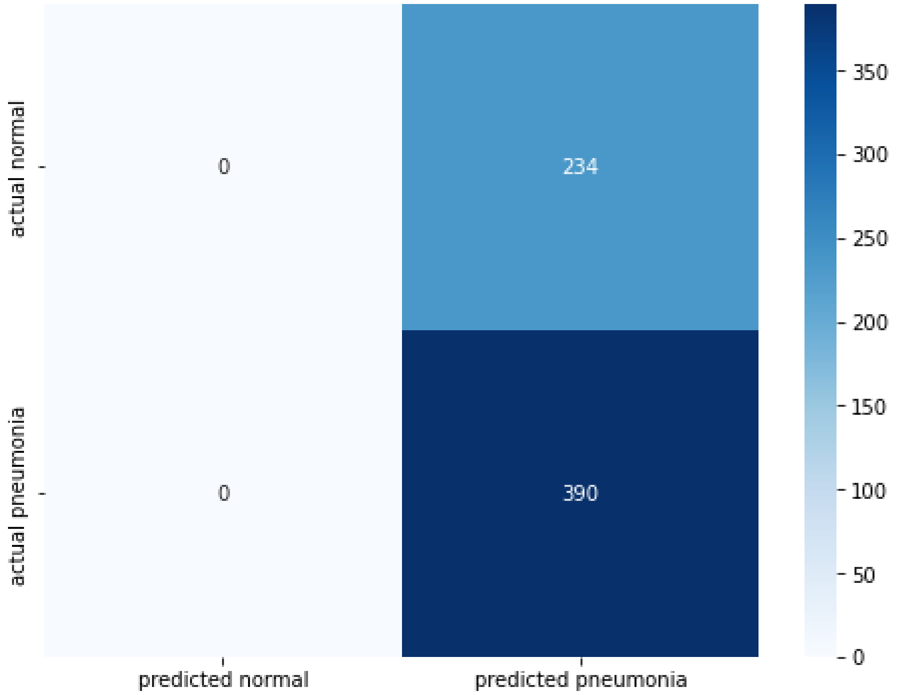
```python
full_report(baseline_model, baseline_results)
```

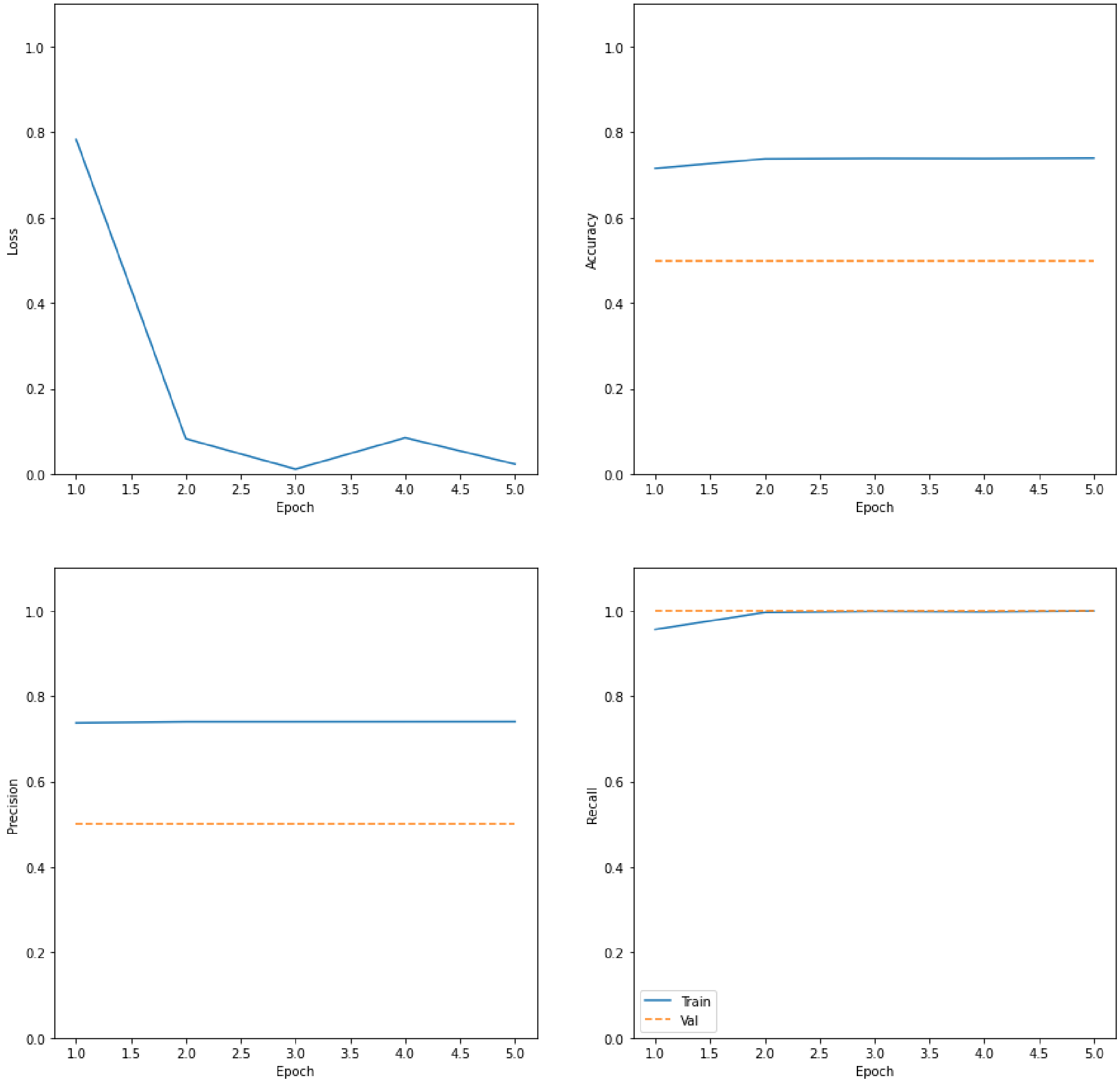
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.00      0.00      0.00       234
   pneumonia       0.62      1.00      0.77       390

    accuracy                           0.62       624
   macro avg       0.31      0.50      0.38       624
weighted avg       0.39      0.62      0.48       624
```


actual normal
actual pneumonia
predicted normal
predicted pneumonia
0
234
0
390
350
300
250
200
150
100
50
0


Loss
Epoch
Accuracy
Epoch
Precision
Epoch
Recall
Epoch
Train
Val
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

This doesn't look great. Our model accuracy is at 74% and steady, and the validation is at 50%. Unsurprisingly, these are the percentages of our data that are labeled with pneumonia (74% for training, 50% for validation). It seems that the model is predicting the majority class for each image, definitely not good.

CNN models

Correcting data shape

In [36]:

```
# set up the X train/test/val for CNNs
X_train = X_train_orig.reshape(len(X_train_orig), 256, 256, 1)
X_val = X_val_orig.reshape(len(X_val_orig), 256, 256, 1)
X_test = X_test_orig.reshape(len(X_test_orig), 256, 256, 1)

# verify shapes are accurate
print(f"Train data/target shapes: {X_train.shape}, {y_train.shape}")
print(f"Test data/target shapes: {X_test.shape}, {y_test.shape}")
print(f"Val data/target shapes: {X_val.shape}, {y_val.shape}")
```

```
Train data/target shapes: (5158, 256, 256, 1), (5158,)
Test data/target shapes: (624, 256, 256, 1), (624,)
Val data/target shapes: (16, 256, 256, 1), (16,)
```

Callbacks

In [37]:

```
# we are going to employ a patience of 2 for early stopping
CALLBACKS = [
    EarlyStopping(patience=2),

    ModelCheckpoint(filepath='model.epoch{epoch:02d}-loss{val_loss:.2f}.hdf5',
                               monitor='val_loss',
                               verbose=1,
                               save_best_only=True,
                               mode='min'
                               )
]
```

Hyperparameter Defaults

To ensure we are only changing particular hyperparameters at a time, we will set defaults to be used in model building.

- The input shape is set correct for our images size (256, 256, 1).
- Default epochs will be 25, using early stopping this should be fine.
- Default batch size will be 128.
- Default layer activation will use 'relu'
- Default output activation will use 'sigmoid'
- Default loss will use binary crossentropy
- Default optimizer will use 'adam'.
- The list of metrics was defined in section 6.1

In [38]:

```
# set the default input shape to (256, 256, 1)
INPUT_SHAPE = (256, 256, 1)

# set the default epoch length to 10
EPOCHS = 25

# set the default batch_size to 128
BATCH_SIZE = 128

# set the default layer activation to 'relu'
ACTIVATION = 'relu'

# set the default output activation to 'sigmoid'
OUTPUT = 'sigmoid'

# set the default loss to 'binary crossentropy'
LOSS = 'binary_crossentropy'

# set the default optimizer to 'adam'
OPTIMIZER = 'adam'
```

CNN model v1

For the first CNN model, we will use 3 layers of convolution, starting with 32 filters and doubling each layer.

In [39]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v1 = make_cnn_model()
cnn_model_v1.summary()
```

```
Model: "sequential_1"
```

```
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 115200)            0         
                                                                 
 dense_4 (Dense)             (None, 256)               29491456  
                                                                 
 dense_5 (Dense)             (None, 1)                 257       
                                                                 
=================================================================
Total params: 29,584,385
Trainable params: 29,584,385
Non-trainable params: 0
_________________________________________________________________
```

In [40]:

```
cnn_model_v1_results = cnn_model_v1.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.4141 - tp: 3403.0000 - fp: 60
1.0000 - tn: 748.0000 - fn: 422.0000 - accuracy: 0.8023 - precision: 0.8499 - recall: 0.
8897 - auc: 0.8461 - prc: 0.9200
Epoch 1: val_loss improved from inf to 0.53612, saving model to model.epoch01-loss0.54.h
df5
41/41 [==============================] - 242s 6s/step - loss: 0.4141 - tp: 3403.0000 - f
p: 601.0000 - tn: 748.0000 - fn: 422.0000 - accuracy: 0.8023 - precision: 0.8499 - recal
l: 0.8897 - auc: 0.8461 - prc: 0.9200 - val_loss: 0.5361 - val_tp: 8.0000 - val_fp: 5.00
00 - val_tn: 3.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6875 - val_precision: 0.6154
- val_recall: 1.0000 - val_auc: 0.8906 - val_prc: 0.8977
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1591 - tp: 3683.0000 - fp: 17
3.0000 - tn: 1168.0000 - fn: 134.0000 - accuracy: 0.9405 - precision: 0.9551 - recall:
0.9649 - auc: 0.9794 - prc: 0.9924
Epoch 2: val_loss did not improve from 0.53612
41/41 [==============================] - 258s 6s/step - loss: 0.1591 - tp: 3683.0000 - f
p: 173.0000 - tn: 1168.0000 - fn: 134.0000 - accuracy: 0.9405 - precision: 0.9551 - reca
ll: 0.9649 - auc: 0.9794 - prc: 0.9924 - val_loss: 0.5868 - val_tp: 7.0000 - val_fp: 2.0
000 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - v
al_recall: 0.8750 - val_auc: 0.8203 - val_prc: 0.6953
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1239 - tp: 3690.0000 - fp: 12
4.0000 - tn: 1217.0000 - fn: 127.0000 - accuracy: 0.9513 - precision: 0.9675 - recall:
```

```
0.9667 - auc: 0.9876 - prc: 0.9953
Epoch 3: val_loss did not improve from 0.53612
41/41 [==============================] - 240s 6s/step - loss: 0.1239 - tp: 3690.0000 - f
p: 124.0000 - tn: 1217.0000 - fn: 127.0000 - accuracy: 0.9513 - precision: 0.9675 - reca
ll: 0.9667 - auc: 0.9876 - prc: 0.9953 - val_loss: 0.6202 - val_tp: 8.0000 - val_fp: 5.0
000 - val_tn: 3.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6875 - val_precision: 0.6154
- val_recall: 1.0000 - val_auc: 0.9531 - val_prc: 0.9493
```

In [41]:

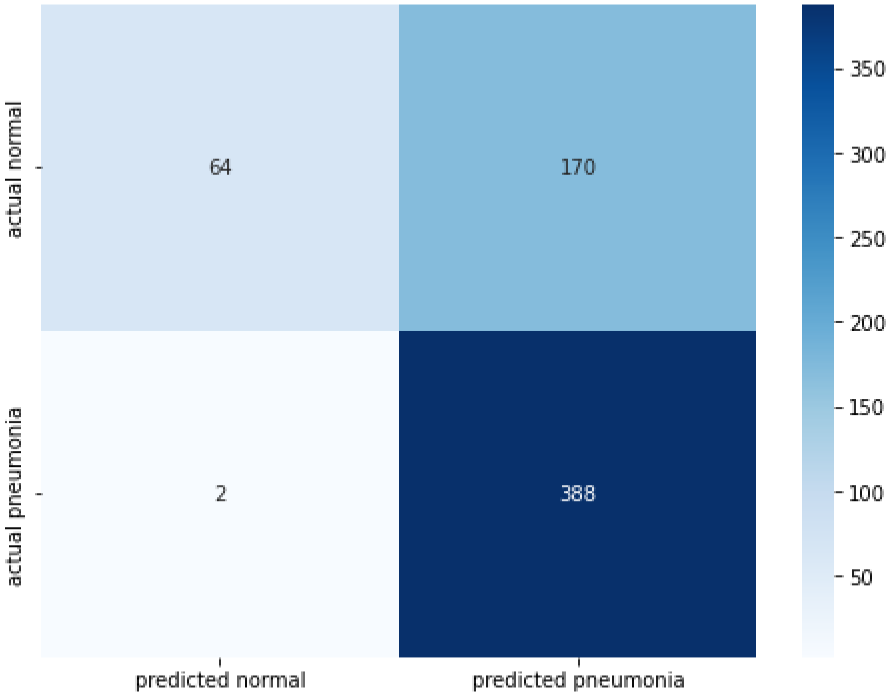
```
full_report(cnn_model_v1, cnn_model_v1_results)
```

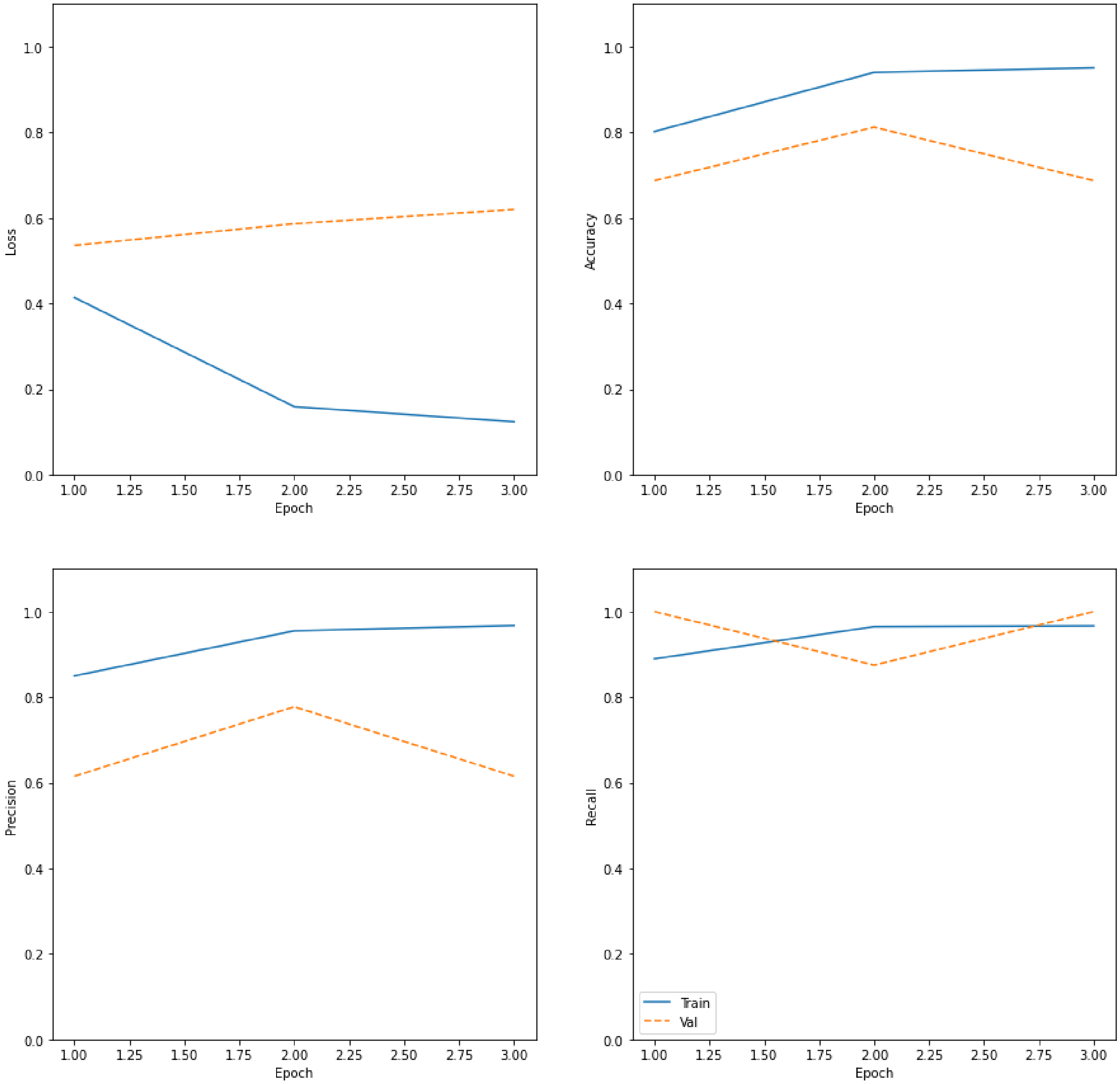
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.97      0.27      0.43       234
   pneumonia       0.70      0.99      0.82       390

    accuracy                           0.72       624
   macro avg       0.83      0.63      0.62       624
weighted avg       0.80      0.72      0.67       624
```





CNN model v2

Adding another convolution layer with 64 filters

In [42]:
```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))
    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v2 = make_cnn_model()
cnn_model_v2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_3 (Conv2D)           (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 127, 127, 32)     0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 125, 125, 64)      18496     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 62, 62, 64)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 60, 60, 64)        36928     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 30, 30, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 28, 28, 128)       73856     
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 14, 14, 128)      0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 25088)             0         
                                                                 
 dense_6 (Dense)             (None, 256)               6422784   
                                                                 
 dense_7 (Dense)             (None, 1)                 257       
                                                                 
=================================================================
Total params: 6,552,641
Trainable params: 6,552,641
Non-trainable params: 0
_________________________________________________________________
```

In [43]:

```
cnn_model_v2_results = cnn_model_v2.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.4241 - tp: 3645.0000 - fp: 78
```

```
6.0000 - tn: 563.0000 - fn: 180.0000 - accuracy: 0.8133 - precision: 0.8226 - recall: 0.
9529 - auc: 0.8159 - prc: 0.9242
Epoch 1: val_loss did not improve from 0.53612
41/41 [==============================] - 225s 5s/step - loss: 0.4241 - tp: 3645.0000 - f
p: 786.0000 - tn: 563.0000 - fn: 180.0000 - accuracy: 0.8133 - precision: 0.8226 - recal
l: 0.9529 - auc: 0.8159 - prc: 0.9242 - val_loss: 0.6760 - val_tp: 7.0000 - val_fp: 5.00
00 - val_tn: 3.0000 - val_fn: 1.0000 - val_accuracy: 0.6250 - val_precision: 0.5833 - va
l_recall: 0.8750 - val_auc: 0.9219 - val_prc: 0.9500
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1524 - tp: 3666.0000 - fp: 15
7.0000 - tn: 1184.0000 - fn: 151.0000 - accuracy: 0.9403 - precision: 0.9589 - recall:
0.9604 - auc: 0.9805 - prc: 0.9926
Epoch 2: val_loss did not improve from 0.53612
41/41 [==============================] - 245s 6s/step - loss: 0.1524 - tp: 3666.0000 - f
p: 157.0000 - tn: 1184.0000 - fn: 151.0000 - accuracy: 0.9403 - precision: 0.9589 - reca
ll: 0.9604 - auc: 0.9805 - prc: 0.9926 - val_loss: 0.8186 - val_tp: 8.0000 - val_fp: 6.0
000 - val_tn: 2.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6250 - val_precision: 0.5714
- val_recall: 1.0000 - val_auc: 0.9141 - val_prc: 0.9101
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1174 - tp: 3706.0000 - fp: 11
4.0000 - tn: 1227.0000 - fn: 111.0000 - accuracy: 0.9564 - precision: 0.9702 - recall:
0.9709 - auc: 0.9885 - prc: 0.9957
Epoch 3: val_loss did not improve from 0.53612
41/41 [==============================] - 259s 6s/step - loss: 0.1174 - tp: 3706.0000 - f
p: 114.0000 - tn: 1227.0000 - fn: 111.0000 - accuracy: 0.9564 - precision: 0.9702 - reca
ll: 0.9709 - auc: 0.9885 - prc: 0.9957 - val_loss: 0.5425 - val_tp: 7.0000 - val_fp: 2.0
000 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - v
al_recall: 0.8750 - val_auc: 0.8672 - val_prc: 0.8577
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 0.1073 - tp: 3724.0000 - fp: 10
9.0000 - tn: 1232.0000 - fn: 93.0000 - accuracy: 0.9608 - precision: 0.9716 - recall: 0.
9756 - auc: 0.9903 - prc: 0.9964
Epoch 4: val_loss did not improve from 0.53612
41/41 [==============================] - 238s 6s/step - loss: 0.1073 - tp: 3724.0000 - f
p: 109.0000 - tn: 1232.0000 - fn: 93.0000 - accuracy: 0.9608 - precision: 0.9716 - recal
l: 0.9756 - auc: 0.9903 - prc: 0.9964 - val_loss: 0.5925 - val_tp: 8.0000 - val_fp: 6.00
00 - val_tn: 2.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6250 - val_precision: 0.5714
- val_recall: 1.0000 - val_auc: 0.9219 - val_prc: 0.9283
Epoch 5/25
41/41 [==============================] - ETA: 0s - loss: 0.0850 - tp: 3733.0000 - fp: 8
2.0000 - tn: 1259.0000 - fn: 84.0000 - accuracy: 0.9678 - precision: 0.9785 - recall: 0.
9780 - auc: 0.9938 - prc: 0.9977
Epoch 5: val_loss did not improve from 0.53612
41/41 [==============================] - 242s 6s/step - loss: 0.0850 - tp: 3733.0000 - f
p: 82.0000 - tn: 1259.0000 - fn: 84.0000 - accuracy: 0.9678 - precision: 0.9785 - recal
l: 0.9780 - auc: 0.9938 - prc: 0.9977 - val_loss: 1.0964 - val_tp: 8.0000 - val_fp: 6.00
00 - val_tn: 2.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6250 - val_precision: 0.5714
- val_recall: 1.0000 - val_auc: 0.8125 - val_prc: 0.8634
```

In [44]:

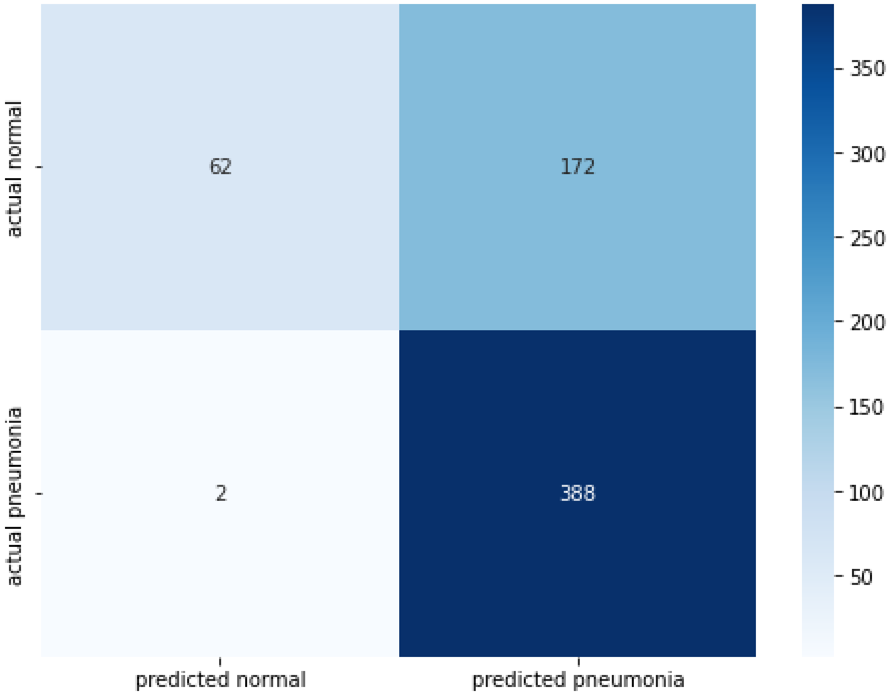
```
full_report(cnn_model_v2, cnn_model_v2_results)
```

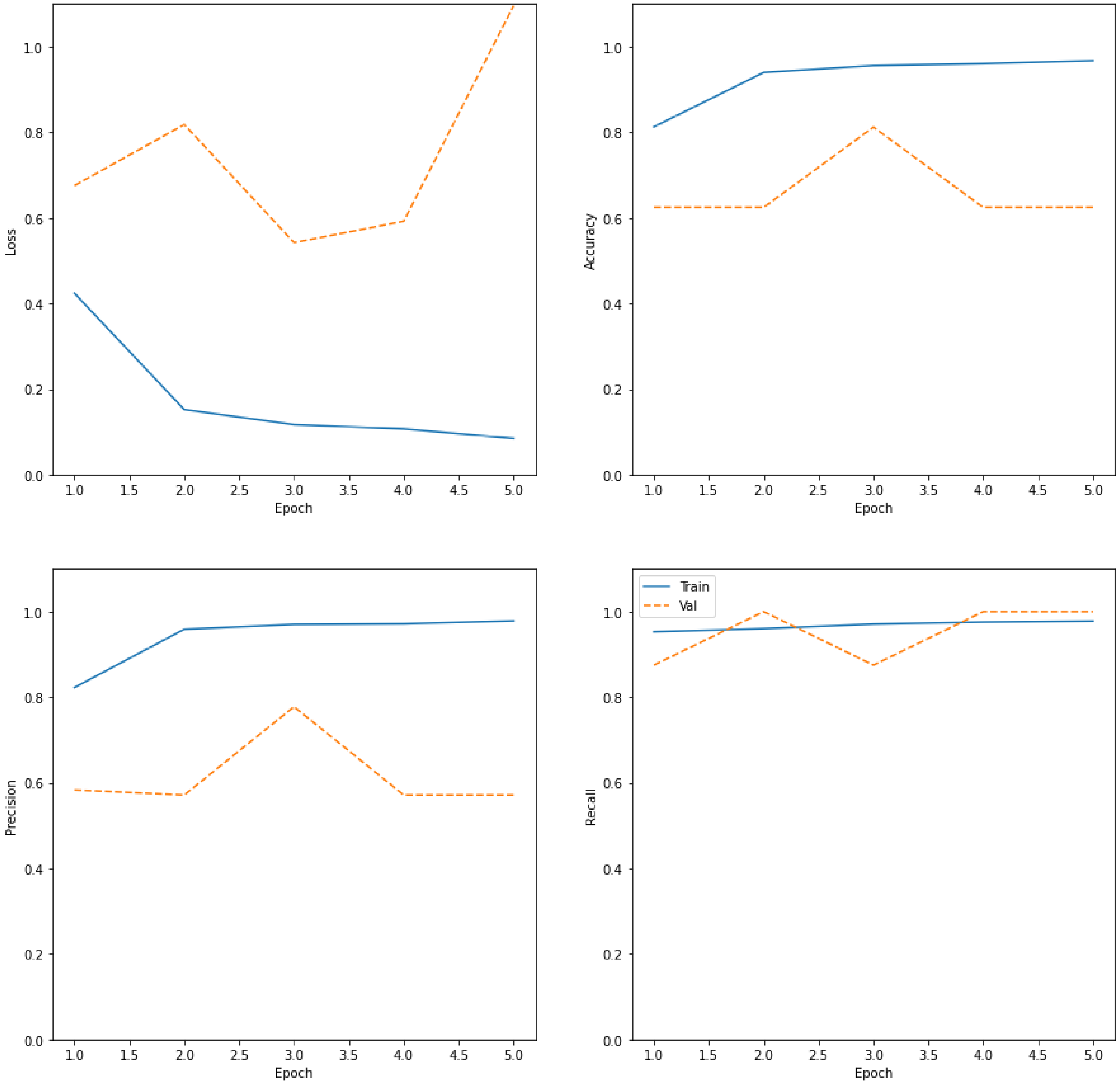
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support
```

```
      normal       0.97      0.26      0.42       234
   pneumonia       0.69      0.99      0.82       390

    accuracy                           0.72       624
   macro avg       0.83      0.63      0.62       624
weighted avg       0.80      0.72      0.67       624
```





CNN model v3

Adding in class weighting (calculated in 6.1)

In [45]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))
    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v3 = make_cnn_model()
cnn_model_v3.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_7 (Conv2D)           (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 127, 127, 32)     0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 125, 125, 64)      18496     
                                                                 
 max_pooling2d_8 (MaxPooling  (None, 62, 62, 64)       0         
 2D)                                                             
                                                                 
 conv2d_9 (Conv2D)           (None, 60, 60, 64)        36928     
                                                                 
 max_pooling2d_9 (MaxPooling  (None, 30, 30, 64)       0         
 2D)                                                             
                                                                 
 conv2d_10 (Conv2D)          (None, 28, 28, 128)       73856     
                                                                 
 max_pooling2d_10 (MaxPoolin  (None, 14, 14, 128)      0         
 g2D)                                                            
                                                                 
 flatten_2 (Flatten)         (None, 25088)             0         
                                                                 
 dense_8 (Dense)             (None, 256)               6422784   
                                                                 
 dense_9 (Dense)             (None, 1)                 257       
                                                                 
=================================================================
Total params: 6,552,641
Trainable params: 6,552,641
Non-trainable params: 0
_________________________________________________________________
```

In [46]:

```
cnn_model_v3_results = cnn_model_v3.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
```

```
41/41 [==============================] - ETA: 0s - loss: 0.4961 - tp: 2844.0000 - fp: 32
9.0000 - tn: 1020.0000 - fn: 981.0000 - accuracy: 0.7468 - precision: 0.8963 - recall:
0.7435 - auc: 0.8358 - prc: 0.9314
Epoch 1: val_loss improved from 0.53612 to 0.29326, saving model to model.epoch01-loss0.
29.hdf5
41/41 [==============================] - 184s 4s/step - loss: 0.4961 - tp: 2844.0000 - f
p: 329.0000 - tn: 1020.0000 - fn: 981.0000 - accuracy: 0.7468 - precision: 0.8963 - reca
ll: 0.7435 - auc: 0.8358 - prc: 0.9314 - val_loss: 0.2933 - val_tp: 8.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.9375 - val_precision: 0.8889
- val_recall: 1.0000 - val_auc: 0.9844 - val_prc: 0.9853
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1611 - tp: 3564.0000 - fp: 8
9.0000 - tn: 1252.0000 - fn: 253.0000 - accuracy: 0.9337 - precision: 0.9756 - recall:
0.9337 - auc: 0.9840 - prc: 0.9939
Epoch 2: val_loss did not improve from 0.29326
41/41 [==============================] - 184s 4s/step - loss: 0.1611 - tp: 3564.0000 - f
p: 89.0000 - tn: 1252.0000 - fn: 253.0000 - accuracy: 0.9337 - precision: 0.9756 - recal
l: 0.9337 - auc: 0.9840 - prc: 0.9939 - val_loss: 0.4653 - val_tp: 8.0000 - val_fp: 3.00
00 - val_tn: 5.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8125 - val_precision: 0.7273
- val_recall: 1.0000 - val_auc: 0.9219 - val_prc: 0.9283
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1292 - tp: 3635.0000 - fp: 6
1.0000 - tn: 1280.0000 - fn: 182.0000 - accuracy: 0.9529 - precision: 0.9835 - recall:
0.9523 - auc: 0.9889 - prc: 0.9959
Epoch 3: val_loss did not improve from 0.29326
41/41 [==============================] - 182s 4s/step - loss: 0.1292 - tp: 3635.0000 - f
p: 61.0000 - tn: 1280.0000 - fn: 182.0000 - accuracy: 0.9529 - precision: 0.9835 - recal
l: 0.9523 - auc: 0.9889 - prc: 0.9959 - val_loss: 0.3937 - val_tp: 6.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 2.0000 - val_accuracy: 0.7500 - val_precision: 0.7500 - va
l_recall: 0.7500 - val_auc: 0.9062 - val_prc: 0.9108
```

In [47]:

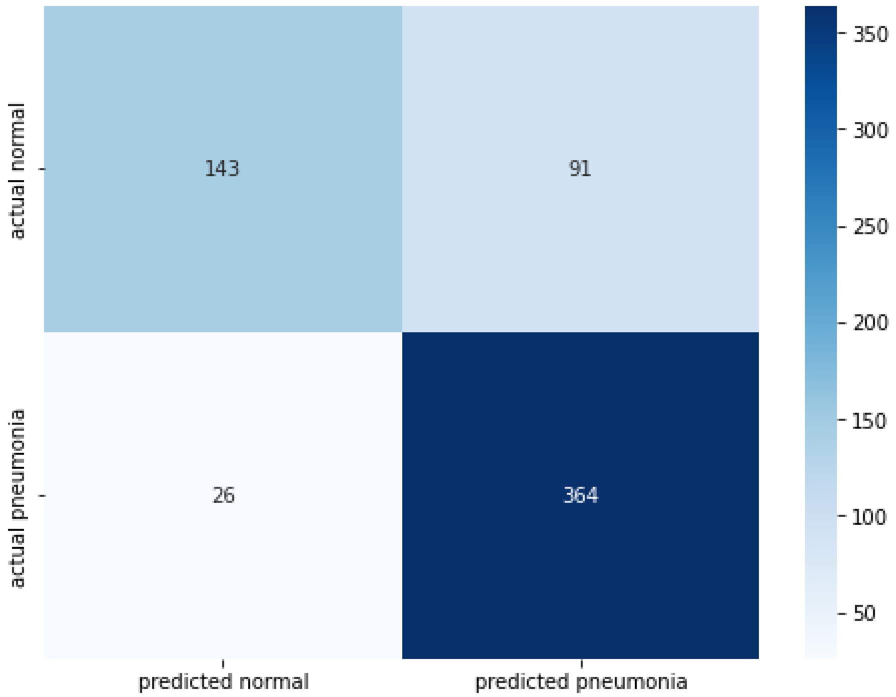
```
full_report(cnn_model_v3, cnn_model_v3_results)
```

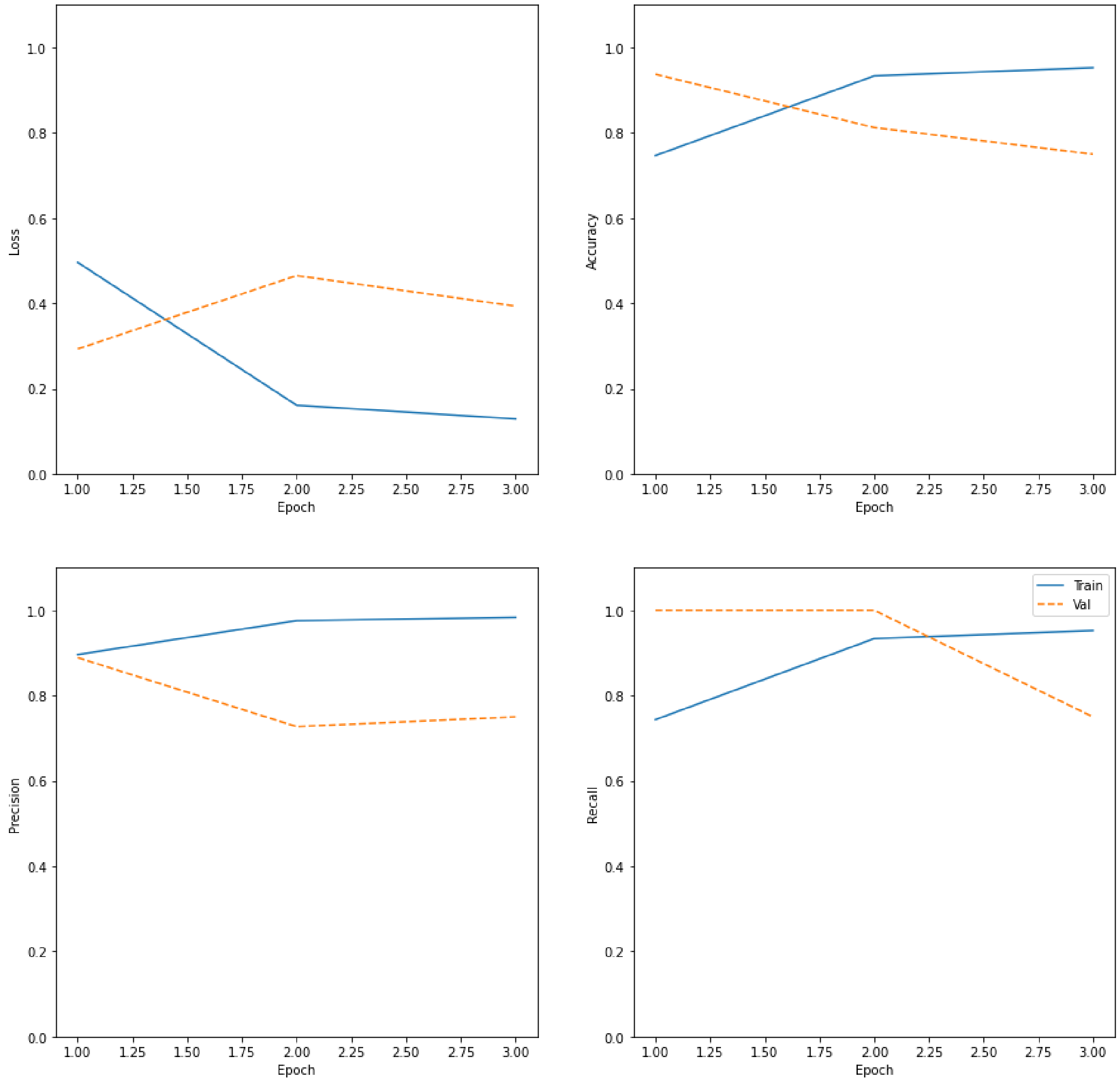
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.85      0.61      0.71       234
   pneumonia       0.80      0.93      0.86       390

    accuracy                           0.81       624
   macro avg       0.82      0.77      0.79       624
weighted avg       0.82      0.81      0.80       624
```


actual normal
actual pneumonia
predicted normal
predicted pneumonia
143
91
26
364
350
300
250
200
150
100
50


Loss
Accuracy
Precision
Recall
Epoch
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
0.0
0.2
0.4
0.6
0.8
1.0
Train
Val

CNN Model v4

Change the optimizer to RMSProp with a learning rate of 0.01

In [48]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=keras.optimizers.RMSprop(0.01),
                  metrics=METRICS)

    return model

cnn_model_v4 = make_cnn_model()
cnn_model_v4.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 254, 254, 32)      320

 max_pooling2d_11 (MaxPoolin  (None, 127, 127, 32)     0
 g2D)

 conv2d_12 (Conv2D)          (None, 125, 125, 64)      18496

 max_pooling2d_12 (MaxPoolin  (None, 62, 62, 64)       0
 g2D)

 conv2d_13 (Conv2D)          (None, 60, 60, 64)        36928

 max_pooling2d_13 (MaxPoolin  (None, 30, 30, 64)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 28, 28, 128)       73856

 max_pooling2d_14 (MaxPoolin  (None, 14, 14, 128)      0
 g2D)

 flatten_3 (Flatten)         (None, 25088)             0

 dense_10 (Dense)            (None, 256)               6422784

 dense_11 (Dense)            (None, 1)                 257

=================================================================
Total params: 6,552,641
Trainable params: 6,552,641
Non-trainable params: 0
_________________________________________________________________
```

In [49]:

```
cnn_model_v4_results = cnn_model_v4.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 583.2070 - tp: 2322.0000 - fp:
436.0000 - tn: 913.0000 - fn: 1503.0000 - accuracy: 0.6252 - precision: 0.8419 - recall:
0.6071 - auc: 0.6558 - prc: 0.8140
Epoch 1: val_loss did not improve from 0.29326
41/41 [==============================] - 181s 4s/step - loss: 583.2070 - tp: 2322.0000 -
fp: 436.0000 - tn: 913.0000 - fn: 1503.0000 - accuracy: 0.6252 - precision: 0.8419 - rec
all: 0.6071 - auc: 0.6558 - prc: 0.8140 - val_loss: 1.0111 - val_tp: 0.0000e+00 - val_f
p: 0.0000e+00 - val_tn: 8.0000 - val_fn: 8.0000 - val_accuracy: 0.5000 - val_precision:
0.0000e+00 - val_recall: 0.0000e+00 - val_auc: 0.8125 - val_prc: 0.8461
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 1.6290 - tp: 2732.0000 - fp: 38
8.0000 - tn: 953.0000 - fn: 1085.0000 - accuracy: 0.7144 - precision: 0.8756 - recall:
0.7157 - auc: 0.7574 - prc: 0.8623
Epoch 2: val_loss did not improve from 0.29326
41/41 [==============================] - 187s 5s/step - loss: 1.6290 - tp: 2732.0000 - f
p: 388.0000 - tn: 953.0000 - fn: 1085.0000 - accuracy: 0.7144 - precision: 0.8756 - reca
ll: 0.7157 - auc: 0.7574 - prc: 0.8623 - val_loss: 0.6657 - val_tp: 8.0000 - val_fp: 7.0
000 - val_tn: 1.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.5625 - val_precision: 0.5333
- val_recall: 1.0000 - val_auc: 0.8750 - val_prc: 0.9085
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 2.0830 - tp: 2544.0000 - fp: 50
7.0000 - tn: 834.0000 - fn: 1273.0000 - accuracy: 0.6549 - precision: 0.8338 - recall:
0.6665 - auc: 0.6456 - prc: 0.7960
Epoch 3: val_loss did not improve from 0.29326
41/41 [==============================] - 180s 4s/step - loss: 2.0830 - tp: 2544.0000 - f
p: 507.0000 - tn: 834.0000 - fn: 1273.0000 - accuracy: 0.6549 - precision: 0.8338 - reca
ll: 0.6665 - auc: 0.6456 - prc: 0.7960 - val_loss: 0.4074 - val_tp: 8.0000 - val_fp: 2.0
000 - val_tn: 6.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8750 - val_precision: 0.8000
- val_recall: 1.0000 - val_auc: 0.9844 - val_prc: 0.9853
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 11.2761 - tp: 2889.0000 - fp: 5
84.0000 - tn: 757.0000 - fn: 928.0000 - accuracy: 0.7069 - precision: 0.8318 - recall:
0.7569 - auc: 0.6840 - prc: 0.8182
Epoch 4: val_loss did not improve from 0.29326
41/41 [==============================] - 180s 4s/step - loss: 11.2761 - tp: 2889.0000 -
fp: 584.0000 - tn: 757.0000 - fn: 928.0000 - accuracy: 0.7069 - precision: 0.8318 - reca
ll: 0.7569 - auc: 0.6840 - prc: 0.8182 - val_loss: 0.5929 - val_tp: 8.0000 - val_fp: 7.0
```

```
000 - val_tn: 1.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.5625 - val_precision: 0.5333
- val_recall: 1.0000 - val_auc: 1.0000 - val_prc: 1.0000
Epoch 5/25
41/41 [==============================] - ETA: 0s - loss: 0.8408 - tp: 3176.0000 - fp: 48
5.0000 - tn: 856.0000 - fn: 641.0000 - accuracy: 0.7817 - precision: 0.8675 - recall: 0.
8321 - auc: 0.7799 - prc: 0.8713
Epoch 5: val_loss did not improve from 0.29326
41/41 [==============================] - 174s 4s/step - loss: 0.8408 - tp: 3176.0000 - f
p: 485.0000 - tn: 856.0000 - fn: 641.0000 - accuracy: 0.7817 - precision: 0.8675 - recal
l: 0.8321 - auc: 0.7799 - prc: 0.8713 - val_loss: 10.5880 - val_tp: 8.0000 - val_fp: 8.0
000 - val_tn: 0.0000e+00 - val_fn: 0.0000e+00 - val_accuracy: 0.5000 - val_precision: 0.
5000 - val_recall: 1.0000 - val_auc: 0.5000 - val_prc: 0.5000
```

In [50]:

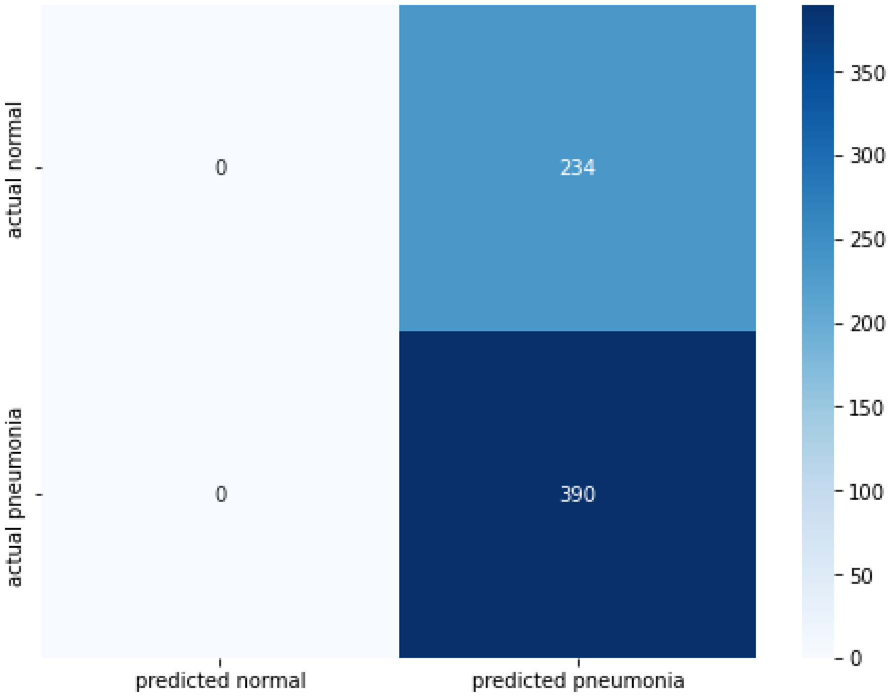
```
full_report(cnn_model_v4, cnn_model_v4_results)
```

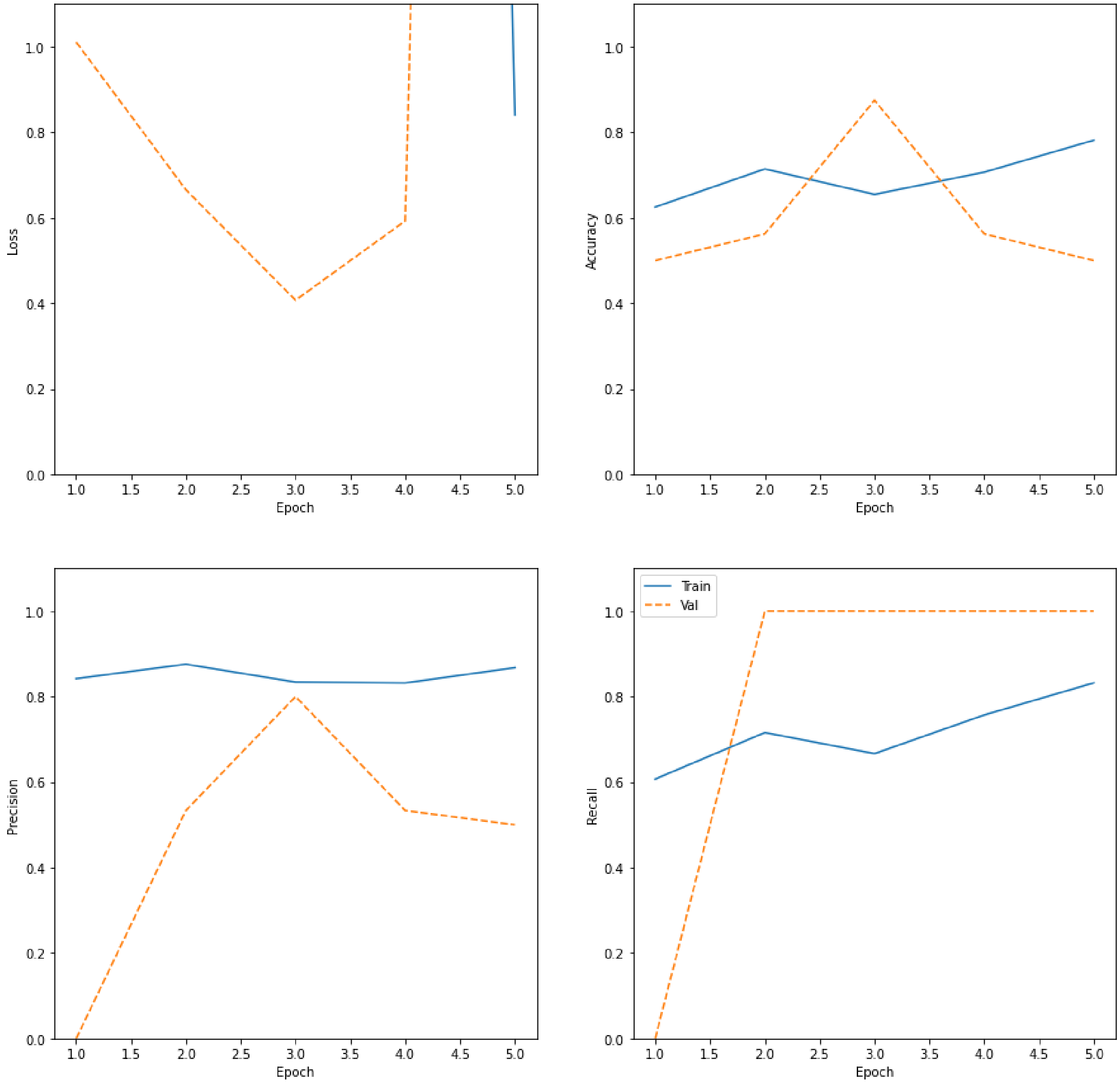
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.00      0.00      0.00       234
   pneumonia       0.62      1.00      0.77       390

    accuracy                           0.62       624
   macro avg       0.31      0.50      0.38       624
weighted avg       0.39      0.62      0.48       624
```





CNN model v5

Decrease the learning rate by a factor of 10.

In [51]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))
    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=keras.optimizers.RMSprop(0.001),
                  metrics=METRICS)

    return model

cnn_model_v5 = make_cnn_model()
cnn_model_v5.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_15 (Conv2D)          (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d_15 (MaxPoolin  (None, 127, 127, 32)     0         
 g2D)                                                            
                                                                 
 conv2d_16 (Conv2D)          (None, 125, 125, 64)      18496     
                                                                 
 max_pooling2d_16 (MaxPoolin  (None, 62, 62, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_17 (Conv2D)          (None, 60, 60, 64)        36928     
                                                                 
 max_pooling2d_17 (MaxPoolin  (None, 30, 30, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_18 (Conv2D)          (None, 28, 28, 128)       73856     
                                                                 
 max_pooling2d_18 (MaxPoolin  (None, 14, 14, 128)      0         
 g2D)                                                            
                                                                 
 flatten_4 (Flatten)         (None, 25088)             0         
                                                                 
 dense_12 (Dense)            (None, 256)               6422784   
                                                                 
 dense_13 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 6,552,641
Trainable params: 6,552,641
Non-trainable params: 0
_________________________________________________________________
```

In [52]:

```
cnn_model_v5_results = cnn_model_v5.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
```

```
41/41 [==============================] - ETA: 0s - loss: 0.7913 - tp: 2250.0000 - fp: 40
9.0000 - tn: 940.0000 - fn: 1575.0000 - accuracy: 0.6165 - precision: 0.8462 - recall:
0.5882 - auc: 0.6969 - prc: 0.8437
Epoch 1: val_loss did not improve from 0.29326
41/41 [==============================] - 183s 4s/step - loss: 0.7913 - tp: 2250.0000 - f
p: 409.0000 - tn: 940.0000 - fn: 1575.0000 - accuracy: 0.6165 - precision: 0.8462 - reca
ll: 0.5882 - auc: 0.6969 - prc: 0.8437 - val_loss: 0.4092 - val_tp: 8.0000 - val_fp: 4.0
000 - val_tn: 4.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.7500 - val_precision: 0.6667
- val_recall: 1.0000 - val_auc: 0.9531 - val_prc: 0.9643
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.3409 - tp: 3316.0000 - fp: 18
5.0000 - tn: 1156.0000 - fn: 501.0000 - accuracy: 0.8670 - precision: 0.9472 - recall:
0.8687 - auc: 0.9323 - prc: 0.9678
Epoch 2: val_loss did not improve from 0.29326
41/41 [==============================] - 180s 4s/step - loss: 0.3409 - tp: 3316.0000 - f
p: 185.0000 - tn: 1156.0000 - fn: 501.0000 - accuracy: 0.8670 - precision: 0.9472 - reca
ll: 0.8687 - auc: 0.9323 - prc: 0.9678 - val_loss: 0.3299 - val_tp: 7.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - v
al_recall: 0.8750 - val_auc: 0.9375 - val_prc: 0.9565
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.2178 - tp: 3474.0000 - fp: 10
8.0000 - tn: 1233.0000 - fn: 343.0000 - accuracy: 0.9126 - precision: 0.9698 - recall:
0.9101 - auc: 0.9711 - prc: 0.9887
Epoch 3: val_loss did not improve from 0.29326
41/41 [==============================] - 181s 4s/step - loss: 0.2178 - tp: 3474.0000 - f
p: 108.0000 - tn: 1233.0000 - fn: 343.0000 - accuracy: 0.9126 - precision: 0.9698 - reca
ll: 0.9101 - auc: 0.9711 - prc: 0.9887 - val_loss: 0.8164 - val_tp: 8.0000 - val_fp: 6.0
000 - val_tn: 2.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6250 - val_precision: 0.5714
- val_recall: 1.0000 - val_auc: 0.9688 - val_prc: 0.9737
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 0.2927 - tp: 3467.0000 - fp: 11
8.0000 - tn: 1223.0000 - fn: 350.0000 - accuracy: 0.9093 - precision: 0.9671 - recall:
0.9083 - auc: 0.9556 - prc: 0.9746
Epoch 4: val_loss improved from 0.29326 to 0.25460, saving model to model.epoch04-loss0.
25.hdf5
41/41 [==============================] - 181s 4s/step - loss: 0.2927 - tp: 3467.0000 - f
p: 118.0000 - tn: 1223.0000 - fn: 350.0000 - accuracy: 0.9093 - precision: 0.9671 - reca
ll: 0.9083 - auc: 0.9556 - prc: 0.9746 - val_loss: 0.2546 - val_tp: 6.0000 - val_fp: 0.0
000e+00 - val_tn: 8.0000 - val_fn: 2.0000 - val_accuracy: 0.8750 - val_precision: 1.0000
- val_recall: 0.7500 - val_auc: 0.9844 - val_prc: 0.9853
Epoch 5/25
41/41 [==============================] - ETA: 0s - loss: 0.1743 - tp: 3548.0000 - fp: 8
2.0000 - tn: 1259.0000 - fn: 269.0000 - accuracy: 0.9320 - precision: 0.9774 - recall:
0.9295 - auc: 0.9807 - prc: 0.9928
Epoch 5: val_loss did not improve from 0.25460
41/41 [==============================] - 181s 4s/step - loss: 0.1743 - tp: 3548.0000 - f
p: 82.0000 - tn: 1259.0000 - fn: 269.0000 - accuracy: 0.9320 - precision: 0.9774 - recal
l: 0.9295 - auc: 0.9807 - prc: 0.9928 - val_loss: 0.3517 - val_tp: 8.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8750 - val_precision: 0.8000
- val_recall: 1.0000 - val_auc: 0.9531 - val_prc: 0.9493
Epoch 6/25
41/41 [==============================] - ETA: 0s - loss: 0.1363 - tp: 3595.0000 - fp: 5
9.0000 - tn: 1282.0000 - fn: 222.0000 - accuracy: 0.9455 - precision: 0.9839 - recall:
0.9418 - auc: 0.9878 - prc: 0.9956
Epoch 6: val_loss did not improve from 0.25460
41/41 [==============================] - 178s 4s/step - loss: 0.1363 - tp: 3595.0000 - f
p: 59.0000 - tn: 1282.0000 - fn: 222.0000 - accuracy: 0.9455 - precision: 0.9839 - recal
l: 0.9418 - auc: 0.9878 - prc: 0.9956 - val_loss: 0.3066 - val_tp: 7.0000 - val_fp: 1.00
00 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - va
l_recall: 0.8750 - val_auc: 0.9531 - val_prc: 0.9493
```

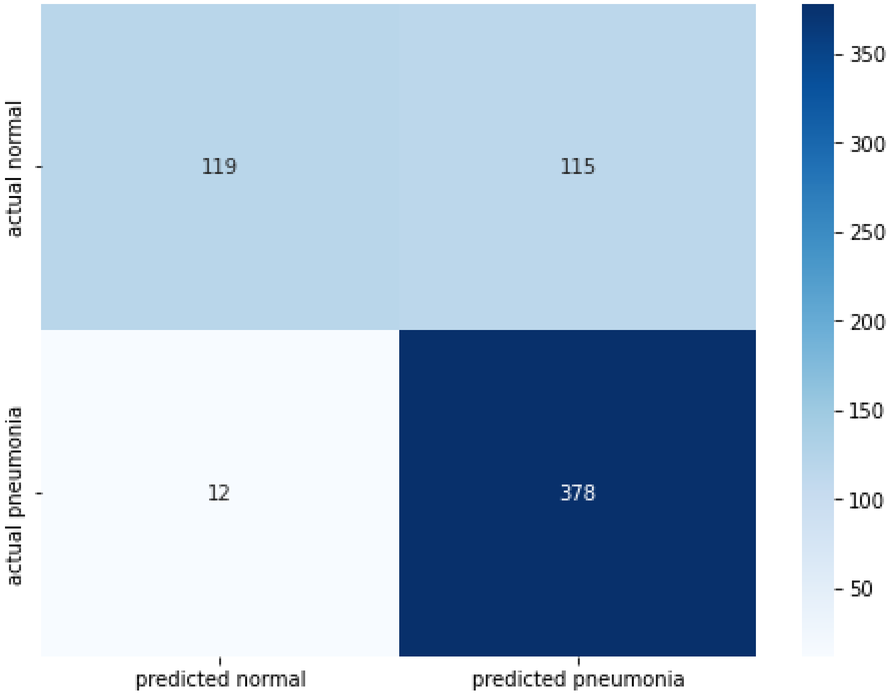
```
In [53]: full_report(cnn_model_v5, cnn_model_v5_results)
```

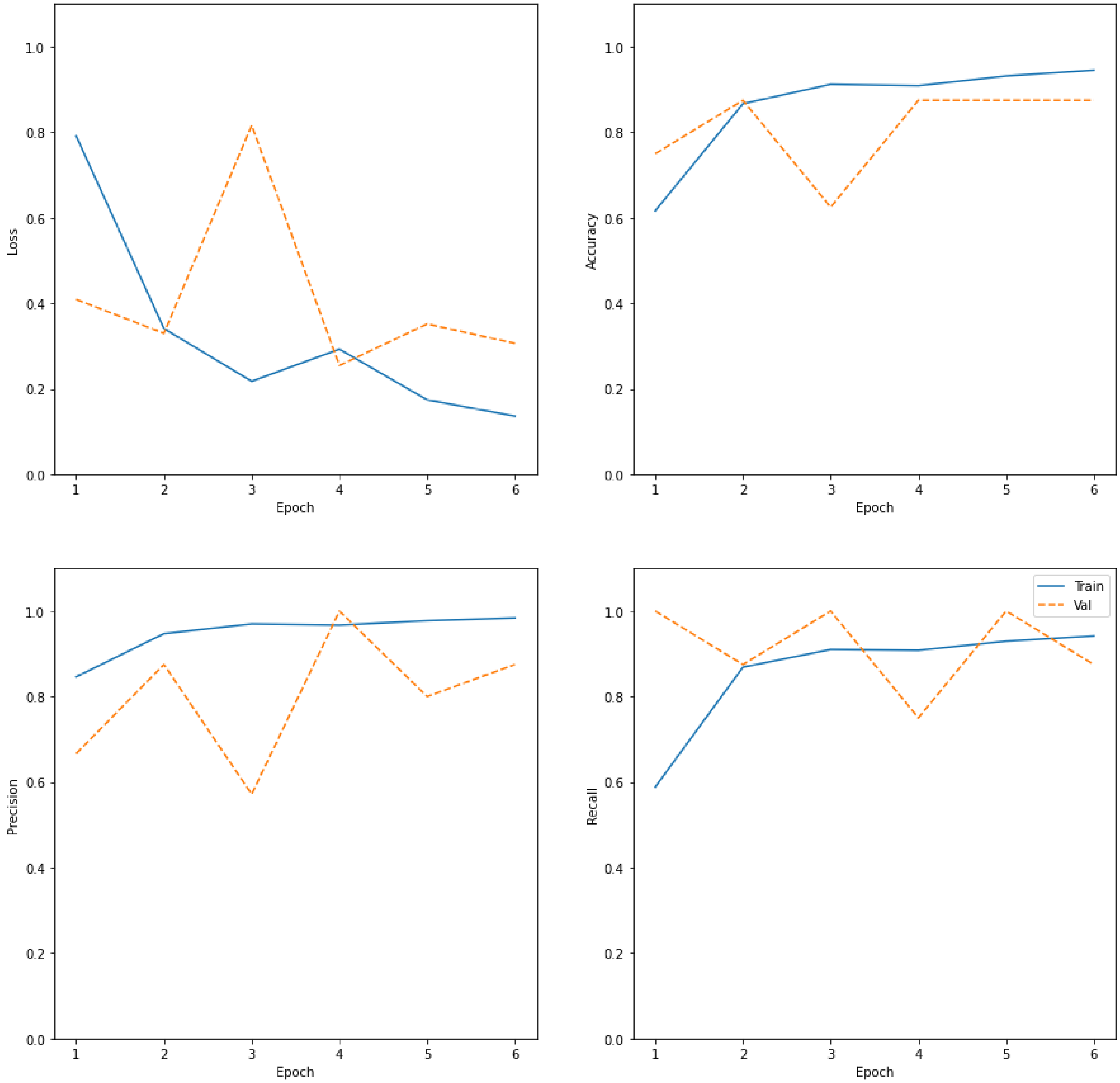
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.91      0.51      0.65       234
   pneumonia       0.77      0.97      0.86       390

    accuracy                           0.80       624
   macro avg       0.84      0.74      0.75       624
weighted avg       0.82      0.80      0.78       624
```





CNN model v6

So far our implemented changes have resulted in worse performing models than our first model.

We are going to go back to that and change filters on the convolutional layers, decreasing each by half.

In [54]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))
    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

    return model

cnn_model_v6 = make_cnn_model()
cnn_model_v6.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_19 (Conv2D)          (None, 254, 254, 16)      160       
                                                                 
 max_pooling2d_19 (MaxPoolin  (None, 127, 127, 16)     0         
 g2D)                                                            
                                                                 
 conv2d_20 (Conv2D)          (None, 125, 125, 32)      4640      
                                                                 
 max_pooling2d_20 (MaxPoolin  (None, 62, 62, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_21 (Conv2D)          (None, 60, 60, 64)        18496     
                                                                 
 max_pooling2d_21 (MaxPoolin  (None, 30, 30, 64)       0         
 g2D)                                                            
                                                                 
 flatten_5 (Flatten)         (None, 57600)             0         
                                                                 
 dense_14 (Dense)            (None, 256)               14745856  
                                                                 
 dense_15 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 14,769,409
Trainable params: 14,769,409
Non-trainable params: 0
_________________________________________________________________
```

In [55]:

```
cnn_model_v6_results = cnn_model_v6.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.4282 - tp: 2981.0000 - fp: 20
6.0000 - tn: 1143.0000 - fn: 844.0000 - accuracy: 0.7971 - precision: 0.9354 - recall:
```

```
0.7793 - auc: 0.8940 - prc: 0.9536
Epoch 1: val_loss did not improve from 0.25460
41/41 [==============================] - 95s 2s/step - loss: 0.4282 - tp: 2981.0000 - f
p: 206.0000 - tn: 1143.0000 - fn: 844.0000 - accuracy: 0.7971 - precision: 0.9354 - reca
ll: 0.7793 - auc: 0.8940 - prc: 0.9536 - val_loss: 0.4718 - val_tp: 8.0000 - val_fp: 3.0
000 - val_tn: 5.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8125 - val_precision: 0.7273
- val_recall: 1.0000 - val_auc: 0.9219 - val_prc: 0.9283
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1791 - tp: 3569.0000 - fp: 9
9.0000 - tn: 1242.0000 - fn: 248.0000 - accuracy: 0.9327 - precision: 0.9730 - recall:
0.9350 - auc: 0.9802 - prc: 0.9928
Epoch 2: val_loss did not improve from 0.25460
41/41 [==============================] - 93s 2s/step - loss: 0.1791 - tp: 3569.0000 - f
p: 99.0000 - tn: 1242.0000 - fn: 248.0000 - accuracy: 0.9327 - precision: 0.9730 - recal
l: 0.9350 - auc: 0.9802 - prc: 0.9928 - val_loss: 0.3126 - val_tp: 6.0000 - val_fp: 1.00
00 - val_tn: 7.0000 - val_fn: 2.0000 - val_accuracy: 0.8125 - val_precision: 0.8571 - va
l_recall: 0.7500 - val_auc: 0.9219 - val_prc: 0.9149
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1138 - tp: 3652.0000 - fp: 5
2.0000 - tn: 1289.0000 - fn: 165.0000 - accuracy: 0.9579 - precision: 0.9860 - recall:
0.9568 - auc: 0.9911 - prc: 0.9968
Epoch 3: val_loss did not improve from 0.25460
41/41 [==============================] - 93s 2s/step - loss: 0.1138 - tp: 3652.0000 - f
p: 52.0000 - tn: 1289.0000 - fn: 165.0000 - accuracy: 0.9579 - precision: 0.9860 - recal
l: 0.9568 - auc: 0.9911 - prc: 0.9968 - val_loss: 0.3612 - val_tp: 8.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8750 - val_precision: 0.8000
- val_recall: 1.0000 - val_auc: 0.9375 - val_prc: 0.9265
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 0.1053 - tp: 3672.0000 - fp: 5
7.0000 - tn: 1284.0000 - fn: 145.0000 - accuracy: 0.9608 - precision: 0.9847 - recall:
0.9620 - auc: 0.9926 - prc: 0.9973
Epoch 4: val_loss did not improve from 0.25460
41/41 [==============================] - 102s 2s/step - loss: 0.1053 - tp: 3672.0000 - f
p: 57.0000 - tn: 1284.0000 - fn: 145.0000 - accuracy: 0.9608 - precision: 0.9847 - recal
l: 0.9620 - auc: 0.9926 - prc: 0.9973 - val_loss: 0.2903 - val_tp: 7.0000 - val_fp: 1.00
00 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - va
l_recall: 0.8750 - val_auc: 0.9531 - val_prc: 0.9493
Epoch 5/25
41/41 [==============================] - ETA: 0s - loss: 0.0917 - tp: 3688.0000 - fp: 3
6.0000 - tn: 1305.0000 - fn: 129.0000 - accuracy: 0.9680 - precision: 0.9903 - recall:
0.9662 - auc: 0.9944 - prc: 0.9979
Epoch 5: val_loss did not improve from 0.25460
41/41 [==============================] - 99s 2s/step - loss: 0.0917 - tp: 3688.0000 - f
p: 36.0000 - tn: 1305.0000 - fn: 129.0000 - accuracy: 0.9680 - precision: 0.9903 - recal
l: 0.9662 - auc: 0.9944 - prc: 0.9979 - val_loss: 0.3546 - val_tp: 7.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - va
l_recall: 0.8750 - val_auc: 0.8750 - val_prc: 0.8723
Epoch 6/25
41/41 [==============================] - ETA: 0s - loss: 0.0669 - tp: 3721.0000 - fp: 3
1.0000 - tn: 1310.0000 - fn: 96.0000 - accuracy: 0.9754 - precision: 0.9917 - recall: 0.
9748 - auc: 0.9967 - prc: 0.9987
Epoch 6: val_loss did not improve from 0.25460
41/41 [==============================] - 94s 2s/step - loss: 0.0669 - tp: 3721.0000 - f
p: 31.0000 - tn: 1310.0000 - fn: 96.0000 - accuracy: 0.9754 - precision: 0.9917 - recal
l: 0.9748 - auc: 0.9967 - prc: 0.9987 - val_loss: 0.2822 - val_tp: 8.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8750 - val_precision: 0.8000
- val_recall: 1.0000 - val_auc: 0.9531 - val_prc: 0.9570
Epoch 7/25
41/41 [==============================] - ETA: 0s - loss: 0.0685 - tp: 3718.0000 - fp: 2
9.0000 - tn: 1312.0000 - fn: 99.0000 - accuracy: 0.9752 - precision: 0.9923 - recall: 0.
```

```
9741 - auc: 0.9966 - prc: 0.9989
Epoch 7: val_loss did not improve from 0.25460
41/41 [==============================] - 93s 2s/step - loss: 0.0685 - tp: 3718.0000 - f
p: 29.0000 - tn: 1312.0000 - fn: 99.0000 - accuracy: 0.9752 - precision: 0.9923 - recal
l: 0.9741 - auc: 0.9966 - prc: 0.9989 - val_loss: 0.2750 - val_tp: 7.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - va
l_recall: 0.8750 - val_auc: 0.9375 - val_prc: 0.9442
Epoch 8/25
41/41 [==============================] - ETA: 0s - loss: 0.0606 - tp: 3728.0000 - fp: 2
1.0000 - tn: 1320.0000 - fn: 89.0000 - accuracy: 0.9787 - precision: 0.9944 - recall: 0.
9767 - auc: 0.9975 - prc: 0.9992
Epoch 8: val_loss did not improve from 0.25460
41/41 [==============================] - 103s 3s/step - loss: 0.0606 - tp: 3728.0000 - f
p: 21.0000 - tn: 1320.0000 - fn: 89.0000 - accuracy: 0.9787 - precision: 0.9944 - recal
l: 0.9767 - auc: 0.9975 - prc: 0.9992 - val_loss: 0.4908 - val_tp: 8.0000 - val_fp: 3.00
00 - val_tn: 5.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8125 - val_precision: 0.7273
- val_recall: 1.0000 - val_auc: 0.8906 - val_prc: 0.9055
Epoch 9/25
41/41 [==============================] - ETA: 0s - loss: 0.0425 - tp: 3757.0000 - fp: 2
0.0000 - tn: 1321.0000 - fn: 60.0000 - accuracy: 0.9845 - precision: 0.9947 - recall: 0.
9843 - auc: 0.9988 - prc: 0.9996
Epoch 9: val_loss did not improve from 0.25460
41/41 [==============================] - 99s 2s/step - loss: 0.0425 - tp: 3757.0000 - f
p: 20.0000 - tn: 1321.0000 - fn: 60.0000 - accuracy: 0.9845 - precision: 0.9947 - recal
l: 0.9843 - auc: 0.9988 - prc: 0.9996 - val_loss: 0.5180 - val_tp: 8.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8750 - val_precision: 0.8000
- val_recall: 1.0000 - val_auc: 0.8750 - val_prc: 0.8383
```

In [56]:

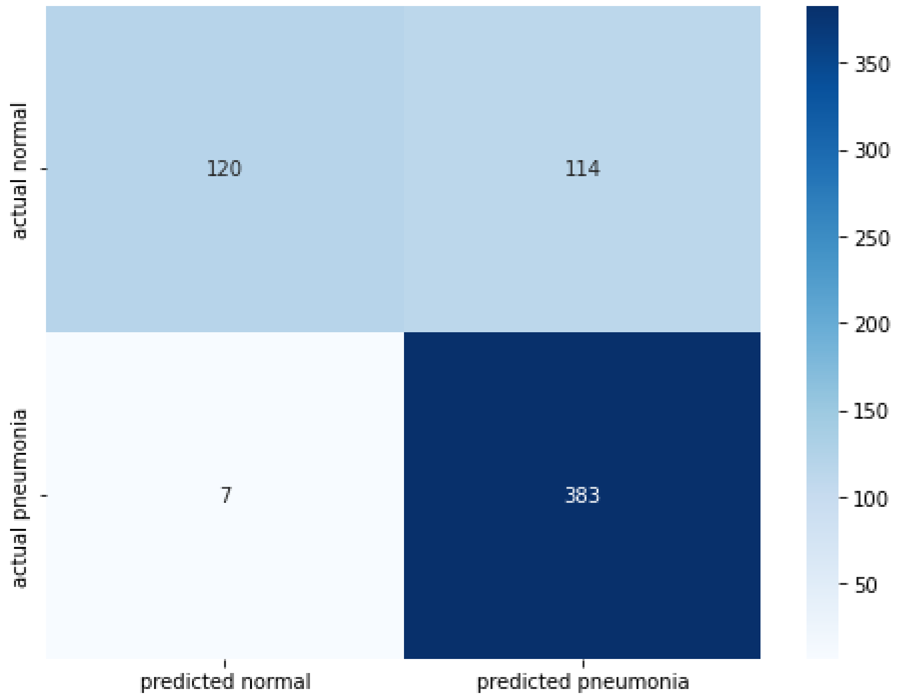
```
full_report(cnn_model_v6, cnn_model_v6_results)
```

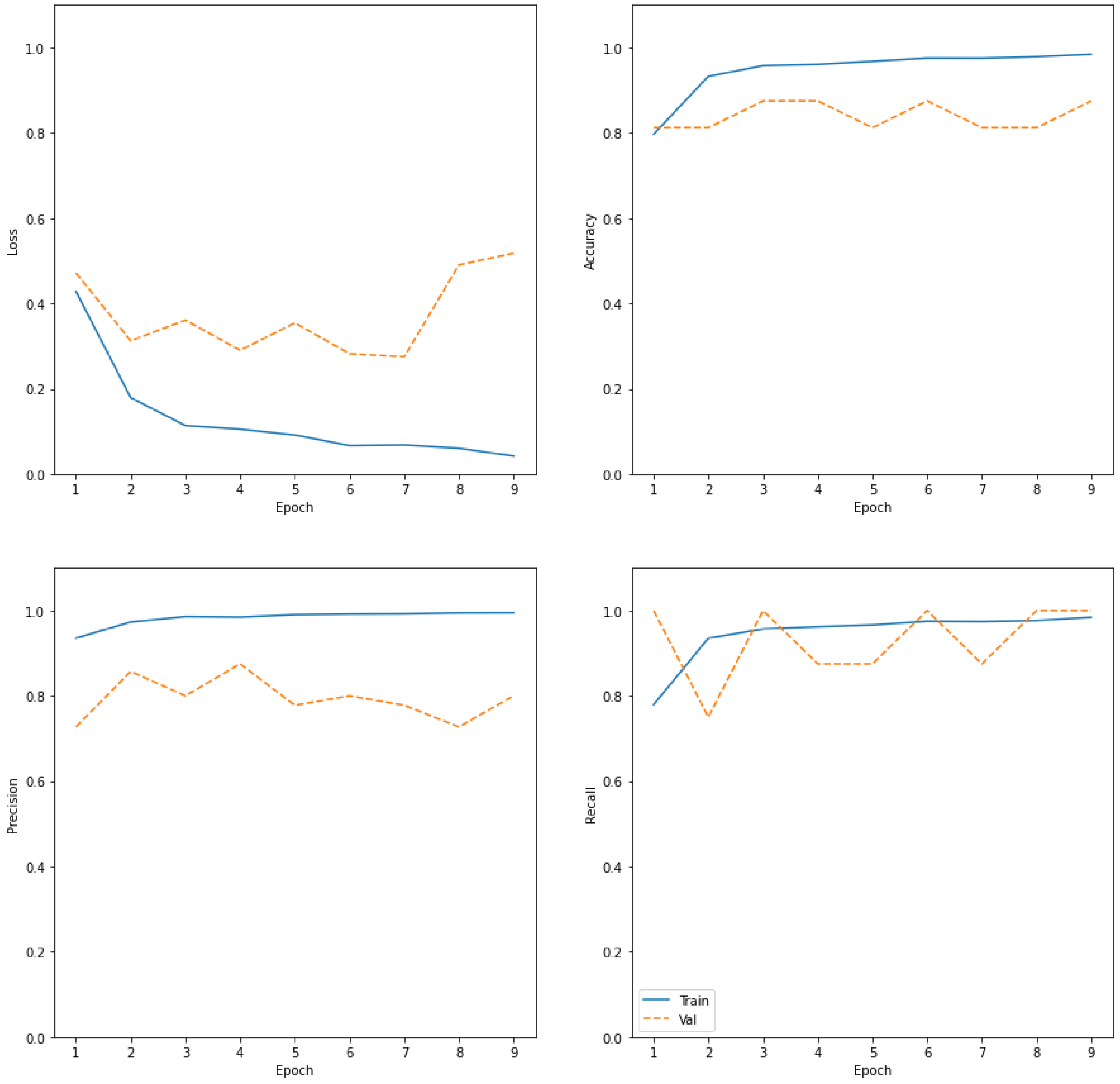
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.94      0.51      0.66       234
   pneumonia       0.77      0.98      0.86       390

    accuracy                           0.81       624
   macro avg       0.86      0.75      0.76       624
weighted avg       0.84      0.81      0.79       624
```


actual normal
actual pneumonia
predicted normal
predicted pneumonia
120
114
7
383
350
300
250
200
150
100
50


Loss
Accuracy
Precision
Recall
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
6
7
8
9
Train
Val

CNN model v7

Lets add in a 4th convolutional layer, duplicating the middle layer with 32 filters.

In [57]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(16, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v7 = make_cnn_model()
cnn_model_v7.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_22 (Conv2D)          (None, 254, 254, 16)      160       
                                                                 
 max_pooling2d_22 (MaxPoolin  (None, 127, 127, 16)     0         
 g2D)                                                            
                                                                 
 conv2d_23 (Conv2D)          (None, 125, 125, 16)      2320      
                                                                 
 max_pooling2d_23 (MaxPoolin  (None, 62, 62, 16)       0         
 g2D)                                                            
                                                                 
 conv2d_24 (Conv2D)          (None, 60, 60, 32)        4640      
                                                                 
 max_pooling2d_24 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_25 (Conv2D)          (None, 28, 28, 64)        18496     
                                                                 
 max_pooling2d_25 (MaxPoolin  (None, 14, 14, 64)       0         
 g2D)

 flatten_6 (Flatten)         (None, 12544)             0

 dense_16 (Dense)            (None, 256)               3211520

 dense_17 (Dense)            (None, 1)                 257

=================================================================
Total params: 3,237,393
Trainable params: 3,237,393
Non-trainable params: 0
_________________________________________________________________
```

In [58]:

```
cnn_model_v7_results = cnn_model_v7.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.4235 - tp: 2992.0000 - fp: 29
4.0000 - tn: 1055.0000 - fn: 833.0000 - accuracy: 0.7822 - precision: 0.9105 - recall:
0.7822 - auc: 0.8830 - prc: 0.9527
Epoch 1: val_loss did not improve from 0.25460
41/41 [==============================] - 82s 2s/step - loss: 0.4235 - tp: 2992.0000 - f
p: 294.0000 - tn: 1055.0000 - fn: 833.0000 - accuracy: 0.7822 - precision: 0.9105 - reca
ll: 0.7822 - auc: 0.8830 - prc: 0.9527 - val_loss: 0.5510 - val_tp: 7.0000 - val_fp: 3.0
000 - val_tn: 5.0000 - val_fn: 1.0000 - val_accuracy: 0.7500 - val_precision: 0.7000 - v
al_recall: 0.8750 - val_auc: 0.8906 - val_prc: 0.9055
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1928 - tp: 3513.0000 - fp: 9
1.0000 - tn: 1250.0000 - fn: 304.0000 - accuracy: 0.9234 - precision: 0.9748 - recall:
0.9204 - auc: 0.9772 - prc: 0.9914
Epoch 2: val_loss did not improve from 0.25460
41/41 [==============================] - 82s 2s/step - loss: 0.1928 - tp: 3513.0000 - f
p: 91.0000 - tn: 1250.0000 - fn: 304.0000 - accuracy: 0.9234 - precision: 0.9748 - recal
l: 0.9204 - auc: 0.9772 - prc: 0.9914 - val_loss: 0.4212 - val_tp: 8.0000 - val_fp: 3.00
00 - val_tn: 5.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8125 - val_precision: 0.7273
- val_recall: 1.0000 - val_auc: 0.9219 - val_prc: 0.9249
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1592 - tp: 3577.0000 - fp: 8
1.0000 - tn: 1260.0000 - fn: 240.0000 - accuracy: 0.9378 - precision: 0.9779 - recall:
0.9371 - auc: 0.9841 - prc: 0.9940
Epoch 3: val_loss did not improve from 0.25460
41/41 [==============================] - 84s 2s/step - loss: 0.1592 - tp: 3577.0000 - f
p: 81.0000 - tn: 1260.0000 - fn: 240.0000 - accuracy: 0.9378 - precision: 0.9779 - recal
l: 0.9371 - auc: 0.9841 - prc: 0.9940 - val_loss: 0.3904 - val_tp: 6.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 2.0000 - val_accuracy: 0.7500 - val_precision: 0.7500 - va
l_recall: 0.7500 - val_auc: 0.8906 - val_prc: 0.9014
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 0.1192 - tp: 3641.0000 - fp: 5
8.0000 - tn: 1283.0000 - fn: 176.0000 - accuracy: 0.9546 - precision: 0.9843 - recall:
0.9539 - auc: 0.9907 - prc: 0.9965
Epoch 4: val_loss did not improve from 0.25460
41/41 [==============================] - 80s 2s/step - loss: 0.1192 - tp: 3641.0000 - f
p: 58.0000 - tn: 1283.0000 - fn: 176.0000 - accuracy: 0.9546 - precision: 0.9843 - recal
l: 0.9539 - auc: 0.9907 - prc: 0.9965 - val_loss: 0.4330 - val_tp: 6.0000 - val_fp: 2.00
```

```
00 - val_tn: 6.0000 - val_fn: 2.0000 - val_accuracy: 0.7500 - val_precision: 0.7500 - va
l_recall: 0.7500 - val_auc: 0.8906 - val_prc: 0.9014
Epoch 5/25
41/41 [==============================] - ETA: 0s - loss: 0.1040 - tp: 3673.0000 - fp: 5
1.0000 - tn: 1290.0000 - fn: 144.0000 - accuracy: 0.9622 - precision: 0.9863 - recall:
0.9623 - auc: 0.9926 - prc: 0.9972
Epoch 5: val_loss did not improve from 0.25460
41/41 [==============================] - 81s 2s/step - loss: 0.1040 - tp: 3673.0000 - f
p: 51.0000 - tn: 1290.0000 - fn: 144.0000 - accuracy: 0.9622 - precision: 0.9863 - recal
l: 0.9623 - auc: 0.9926 - prc: 0.9972 - val_loss: 0.5486 - val_tp: 7.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - va
l_recall: 0.8750 - val_auc: 0.8281 - val_prc: 0.8173
```

In [59]:

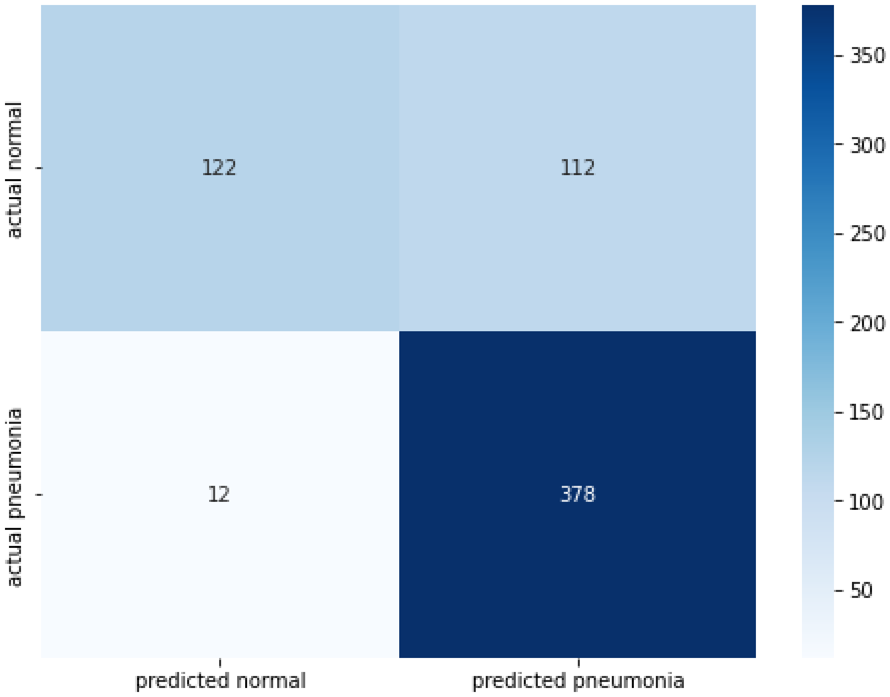
```
full_report(cnn_model_v7, cnn_model_v7_results)
```

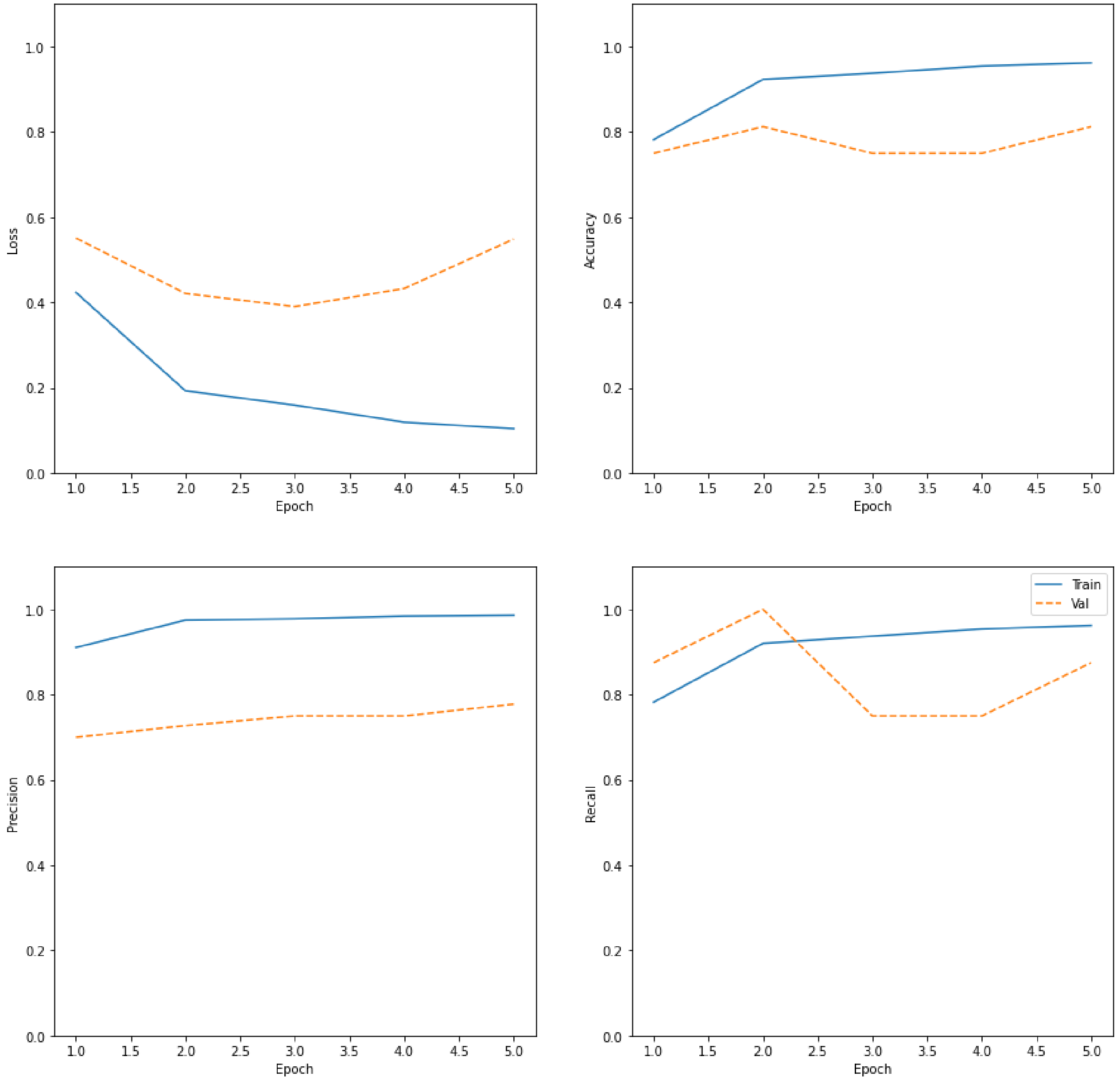
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.91      0.52      0.66       234
   pneumonia       0.77      0.97      0.86       390

    accuracy                           0.80       624
   macro avg       0.84      0.75      0.76       624
weighted avg       0.82      0.80      0.79       624
```





CNN model v8

Same model as version 7 with decreased batch size

In [60]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(16, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))
    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v8 = make_cnn_model()
cnn_model_v8.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 254, 254, 16)      160

 max_pooling2d_26 (MaxPoolin  (None, 127, 127, 16)     0
 g2D)

 conv2d_27 (Conv2D)          (None, 125, 125, 16)      2320

 max_pooling2d_27 (MaxPoolin  (None, 62, 62, 16)       0
 g2D)

 conv2d_28 (Conv2D)          (None, 60, 60, 32)        4640

 max_pooling2d_28 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 28, 28, 64)        18496

 max_pooling2d_29 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_7 (Flatten)         (None, 12544)             0

 dense_18 (Dense)            (None, 256)               3211520

 dense_19 (Dense)            (None, 1)                 257

=================================================================
Total params: 3,237,393
Trainable params: 3,237,393
Non-trainable params: 0
_________________________________________________________________
```

In [61]:

```
cnn_model_v8_results = cnn_model_v8.fit(X_train, y_train,
                                        batch_size=64,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
```

```
81/81 [==============================] - ETA: 0s - loss: 0.2935 - tp: 3124.0000 - fp: 10
2.0000 - tn: 1247.0000 - fn: 701.0000 - accuracy: 0.8448 - precision: 0.9684 - recall:
0.8167 - auc: 0.9510 - prc: 0.9817
Epoch 1: val_loss did not improve from 0.25460
81/81 [==============================] - 82s 982ms/step - loss: 0.2935 - tp: 3124.0000 -
fp: 102.0000 - tn: 1247.0000 - fn: 701.0000 - accuracy: 0.8448 - precision: 0.9684 - rec
all: 0.8167 - auc: 0.9510 - prc: 0.9817 - val_loss: 1.1838 - val_tp: 8.0000 - val_fp: 7.
0000 - val_tn: 1.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.5625 - val_precision: 0.533
3 - val_recall: 1.0000 - val_auc: 0.9219 - val_prc: 0.9264
Epoch 2/25
81/81 [==============================] - ETA: 0s - loss: 0.1519 - tp: 3614.0000 - fp: 7
8.0000 - tn: 1263.0000 - fn: 203.0000 - accuracy: 0.9455 - precision: 0.9789 - recall:
0.9468 - auc: 0.9858 - prc: 0.9945
Epoch 2: val_loss did not improve from 0.25460
81/81 [==============================] - 79s 971ms/step - loss: 0.1519 - tp: 3614.0000 -
fp: 78.0000 - tn: 1263.0000 - fn: 203.0000 - accuracy: 0.9455 - precision: 0.9789 - reca
ll: 0.9468 - auc: 0.9858 - prc: 0.9945 - val_loss: 0.5065 - val_tp: 8.0000 - val_fp: 3.0
000 - val_tn: 5.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8125 - val_precision: 0.7273
- val_recall: 1.0000 - val_auc: 0.9375 - val_prc: 0.9265
Epoch 3/25
81/81 [==============================] - ETA: 0s - loss: 0.1069 - tp: 3682.0000 - fp: 5
6.0000 - tn: 1285.0000 - fn: 135.0000 - accuracy: 0.9630 - precision: 0.9850 - recall:
0.9646 - auc: 0.9920 - prc: 0.9970
Epoch 3: val_loss did not improve from 0.25460
81/81 [==============================] - 79s 973ms/step - loss: 0.1069 - tp: 3682.0000 -
fp: 56.0000 - tn: 1285.0000 - fn: 135.0000 - accuracy: 0.9630 - precision: 0.9850 - reca
ll: 0.9646 - auc: 0.9920 - prc: 0.9970 - val_loss: 0.3370 - val_tp: 7.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - v
al_recall: 0.8750 - val_auc: 0.9531 - val_prc: 0.9493
Epoch 4/25
81/81 [==============================] - ETA: 0s - loss: 0.0964 - tp: 3682.0000 - fp: 4
6.0000 - tn: 1295.0000 - fn: 135.0000 - accuracy: 0.9649 - precision: 0.9877 - recall:
0.9646 - auc: 0.9930 - prc: 0.9974
Epoch 4: val_loss did not improve from 0.25460
81/81 [==============================] - 80s 991ms/step - loss: 0.0964 - tp: 3682.0000 -
fp: 46.0000 - tn: 1295.0000 - fn: 135.0000 - accuracy: 0.9649 - precision: 0.9877 - reca
ll: 0.9646 - auc: 0.9930 - prc: 0.9974 - val_loss: 0.7860 - val_tp: 8.0000 - val_fp: 6.0
000 - val_tn: 2.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6250 - val_precision: 0.5714
- val_recall: 1.0000 - val_auc: 0.9141 - val_prc: 0.9003
Epoch 5/25
81/81 [==============================] - ETA: 0s - loss: 0.0853 - tp: 3709.0000 - fp: 3
6.0000 - tn: 1305.0000 - fn: 108.0000 - accuracy: 0.9721 - precision: 0.9904 - recall:
0.9717 - auc: 0.9945 - prc: 0.9979
Epoch 5: val_loss did not improve from 0.25460
81/81 [==============================] - 79s 971ms/step - loss: 0.0853 - tp: 3709.0000 -
fp: 36.0000 - tn: 1305.0000 - fn: 108.0000 - accuracy: 0.9721 - precision: 0.9904 - reca
ll: 0.9717 - auc: 0.9945 - prc: 0.9979 - val_loss: 0.3375 - val_tp: 7.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - v
al_recall: 0.8750 - val_auc: 0.9219 - val_prc: 0.9149
```

In [62]:

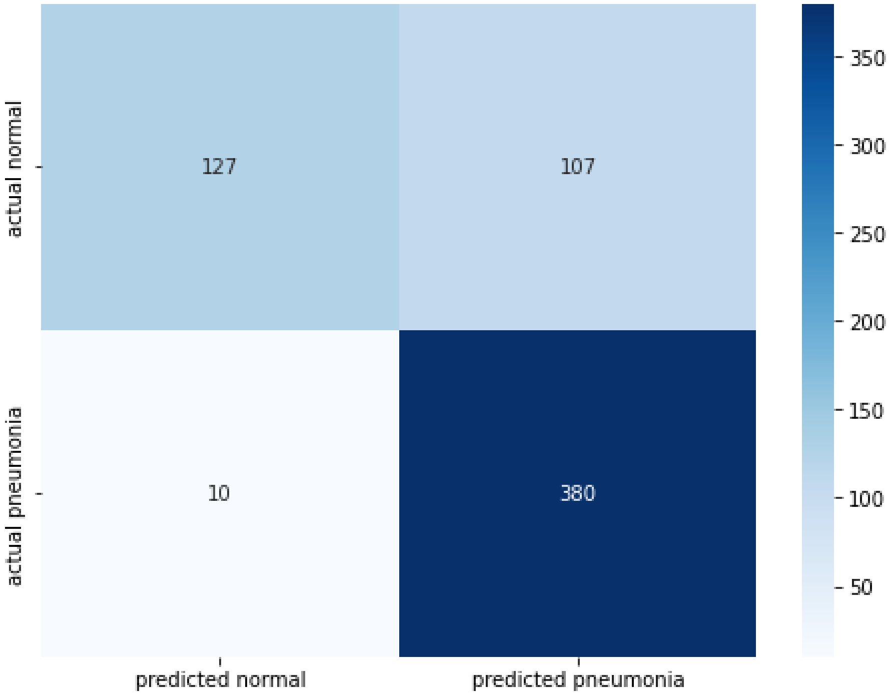
```
full_report(cnn_model_v8, cnn_model_v8_results)
```

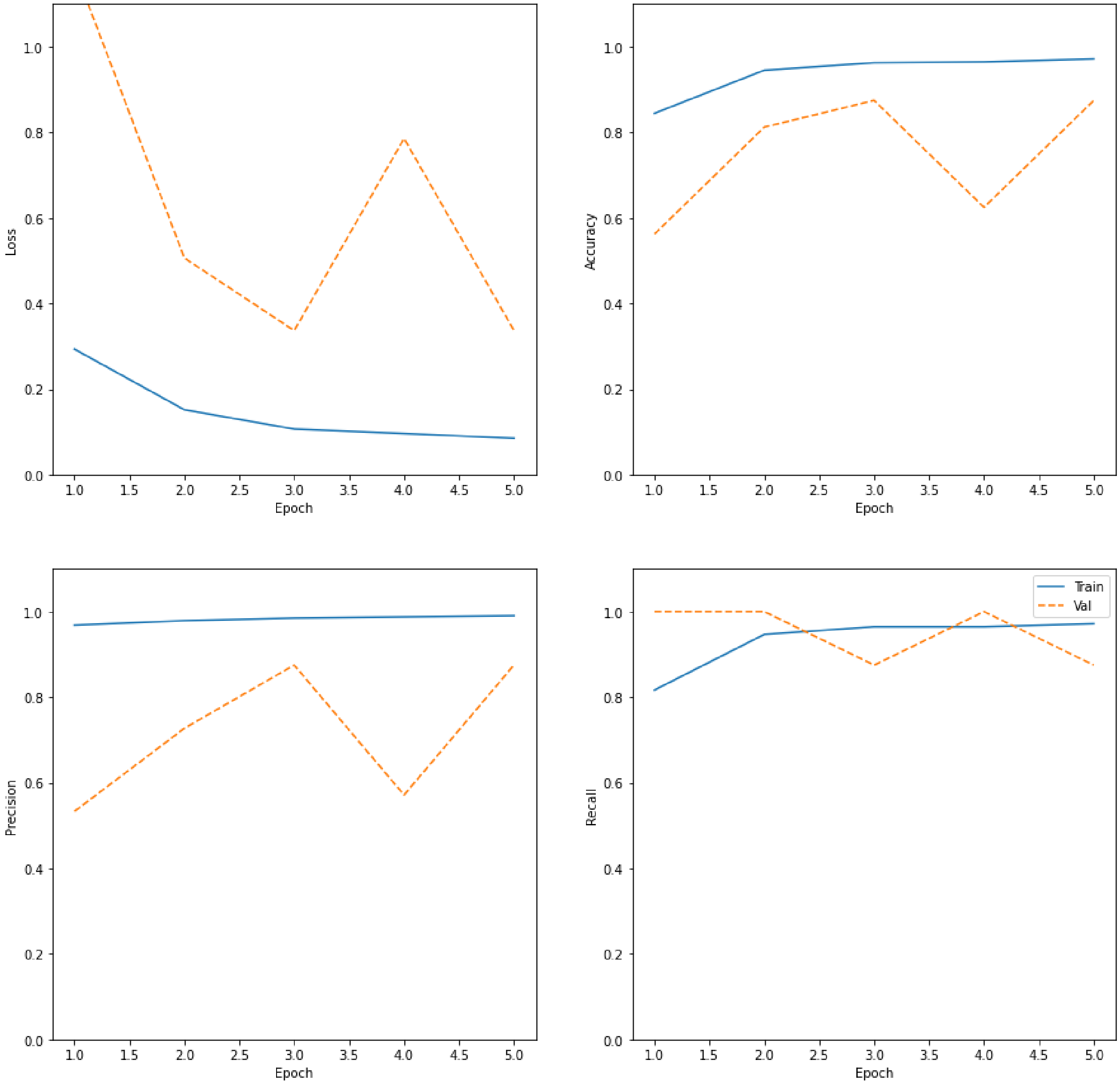
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------
```

```
              precision    recall  f1-score   support

      normal       0.93      0.54      0.68       234
   pneumonia       0.78      0.97      0.87       390

    accuracy                           0.81       624
   macro avg       0.85      0.76      0.78       624
weighted avg       0.84      0.81      0.80       624
```





CNN model v9

Version 8 was the best one yet! Increase in total accuracy to 81%. It does come with a cost of increasing type II errors. Lets increase the batch size to double what we used in version 7

In [63]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(16, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v9 = make_cnn_model()
cnn_model_v9.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_30 (Conv2D)          (None, 254, 254, 16)      160

 max_pooling2d_30 (MaxPoolin  (None, 127, 127, 16)     0
 g2D)

 conv2d_31 (Conv2D)          (None, 125, 125, 16)      2320

 max_pooling2d_31 (MaxPoolin  (None, 62, 62, 16)       0
 g2D)

 conv2d_32 (Conv2D)          (None, 60, 60, 32)        4640

 max_pooling2d_32 (MaxPoolin  (None, 30, 30, 32)       0
 g2D)

 conv2d_33 (Conv2D)          (None, 28, 28, 64)        18496

 max_pooling2d_33 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 flatten_8 (Flatten)         (None, 12544)             0

 dense_20 (Dense)            (None, 256)               3211520

 dense_21 (Dense)            (None, 1)                 257

=================================================================
Total params: 3,237,393
Trainable params: 3,237,393
Non-trainable params: 0
_________________________________________________________________
```

In [64]:

```
cnn_model_v9_results = cnn_model_v9.fit(X_train, y_train,
                                        batch_size=256,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
21/21 [==============================] - ETA: 0s - loss: 0.5910 - tp: 1998.0000 - fp: 17
4.0000 - tn: 1175.0000 - fn: 1827.0000 - accuracy: 0.6133 - precision: 0.9199 - recall:
0.5224 - auc: 0.7973 - prc: 0.9064
Epoch 1: val_loss did not improve from 0.25460
21/21 [==============================] - 81s 4s/step - loss: 0.5910 - tp: 1998.0000 - f
p: 174.0000 - tn: 1175.0000 - fn: 1827.0000 - accuracy: 0.6133 - precision: 0.9199 - rec
all: 0.5224 - auc: 0.7973 - prc: 0.9064 - val_loss: 0.5301 - val_tp: 3.0000 - val_fp: 0.
0000e+00 - val_tn: 8.0000 - val_fn: 5.0000 - val_accuracy: 0.6875 - val_precision: 1.000
0 - val_recall: 0.3750 - val_auc: 0.9531 - val_prc: 0.9643
Epoch 2/25
21/21 [==============================] - ETA: 0s - loss: 0.3104 - tp: 3270.0000 - fp: 14
1.0000 - tn: 1200.0000 - fn: 547.0000 - accuracy: 0.8666 - precision: 0.9587 - recall:
0.8567 - auc: 0.9470 - prc: 0.9804
Epoch 2: val_loss did not improve from 0.25460
21/21 [==============================] - 78s 4s/step - loss: 0.3104 - tp: 3270.0000 - f
p: 141.0000 - tn: 1200.0000 - fn: 547.0000 - accuracy: 0.8666 - precision: 0.9587 - reca
ll: 0.8567 - auc: 0.9470 - prc: 0.9804 - val_loss: 0.2955 - val_tp: 7.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - v
al_recall: 0.8750 - val_auc: 0.9844 - val_prc: 0.9853
Epoch 3/25
21/21 [==============================] - ETA: 0s - loss: 0.1906 - tp: 3503.0000 - fp: 8
5.0000 - tn: 1256.0000 - fn: 314.0000 - accuracy: 0.9226 - precision: 0.9763 - recall:
0.9177 - auc: 0.9786 - prc: 0.9925
Epoch 3: val_loss did not improve from 0.25460
21/21 [==============================] - 78s 4s/step - loss: 0.1906 - tp: 3503.0000 - f
p: 85.0000 - tn: 1256.0000 - fn: 314.0000 - accuracy: 0.9226 - precision: 0.9763 - recal
l: 0.9177 - auc: 0.9786 - prc: 0.9925 - val_loss: 0.3217 - val_tp: 6.0000 - val_fp: 1.00
00 - val_tn: 7.0000 - val_fn: 2.0000 - val_accuracy: 0.8125 - val_precision: 0.8571 - va
l_recall: 0.7500 - val_auc: 0.9531 - val_prc: 0.9493
Epoch 4/25
21/21 [==============================] - ETA: 0s - loss: 0.1344 - tp: 3607.0000 - fp: 5
9.0000 - tn: 1282.0000 - fn: 210.0000 - accuracy: 0.9478 - precision: 0.9839 - recall:
0.9450 - auc: 0.9889 - prc: 0.9959
Epoch 4: val_loss did not improve from 0.25460
21/21 [==============================] - 79s 4s/step - loss: 0.1344 - tp: 3607.0000 - f
p: 59.0000 - tn: 1282.0000 - fn: 210.0000 - accuracy: 0.9478 - precision: 0.9839 - recal
l: 0.9450 - auc: 0.9889 - prc: 0.9959 - val_loss: 0.6329 - val_tp: 8.0000 - val_fp: 3.00
00 - val_tn: 5.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.8125 - val_precision: 0.7273
- val_recall: 1.0000 - val_auc: 0.8594 - val_prc: 0.8686
```

In [65]:

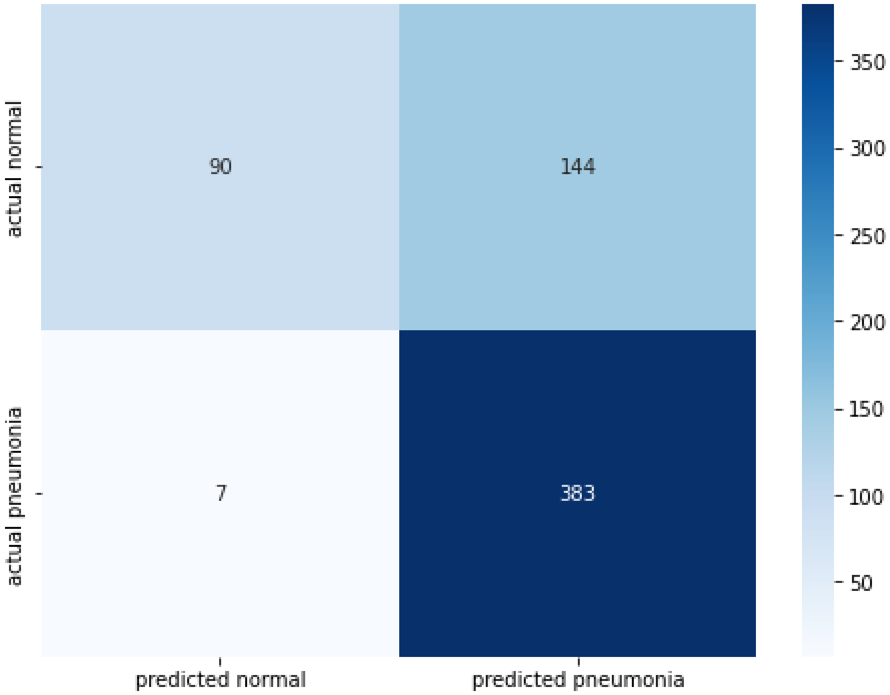
```
full_report(cnn_model_v9, cnn_model_v9_results)
```

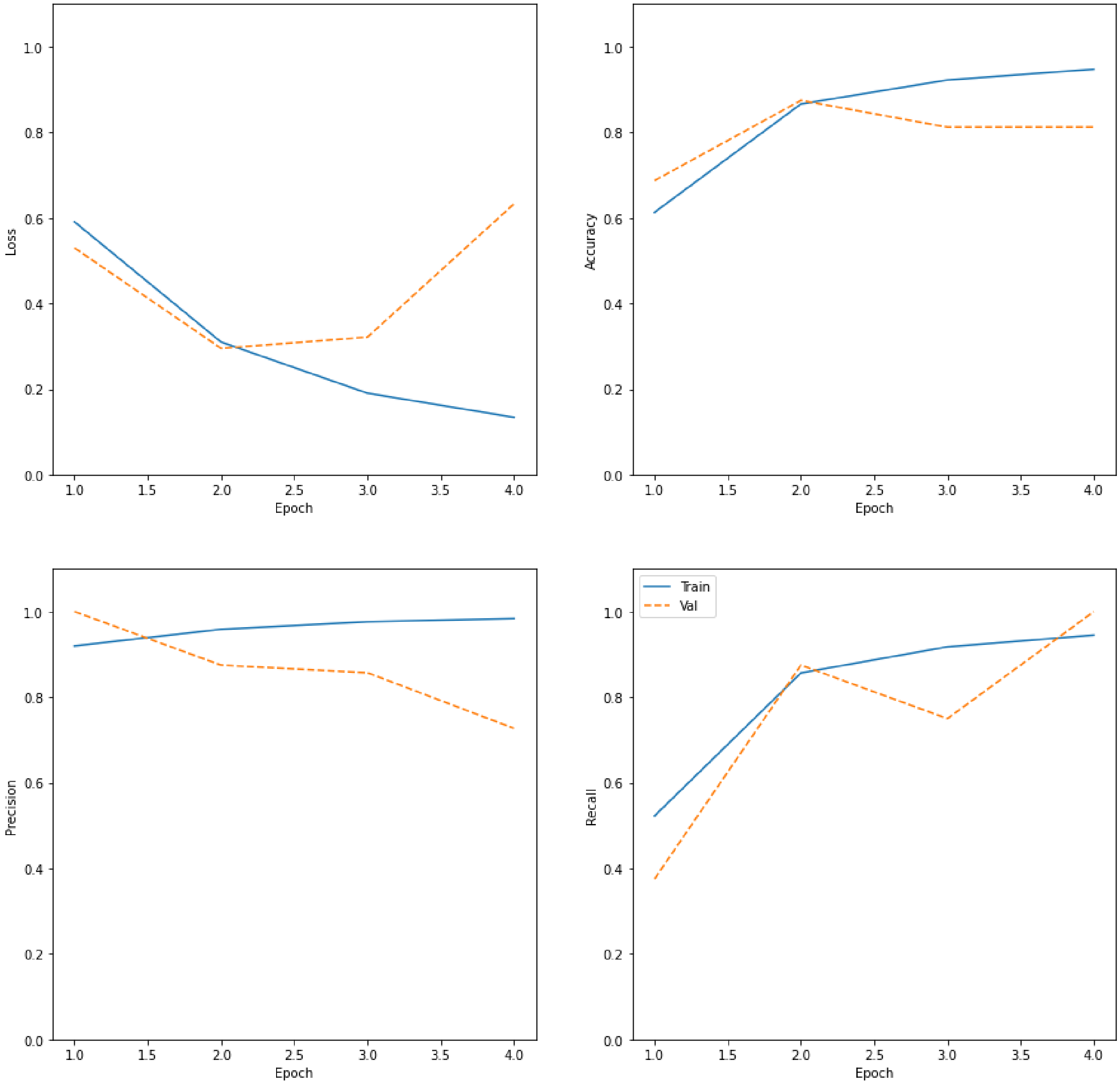
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.93      0.38      0.54       234
   pneumonia       0.73      0.98      0.84       390

    accuracy                           0.76       624
   macro avg       0.83      0.68      0.69       624
weighted avg       0.80      0.76      0.73       624
```


actual normal
actual pneumonia
predicted normal
predicted pneumonia
90
144
7
383
350
300
250
200
150
100
50


Loss
Accuracy
Precision
Recall
Epoch
Train
Val
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

CNN model v10

We are going to take version 8 and increase the complexity of our model. We will leave the same number of convolution layers but increase the filters on each layer by double.

In [66]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v10 = make_cnn_model()
cnn_model_v10.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_34 (Conv2D)          (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d_34 (MaxPoolin  (None, 127, 127, 32)     0         
 g2D)                                                            
                                                                 
 conv2d_35 (Conv2D)          (None, 125, 125, 32)      9248      
                                                                 
 max_pooling2d_35 (MaxPoolin  (None, 62, 62, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_36 (Conv2D)          (None, 60, 60, 64)        18496     
                                                                 
 max_pooling2d_36 (MaxPoolin  (None, 30, 30, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_37 (Conv2D)          (None, 28, 28, 128)       73856     
 max_pooling2d_37 (MaxPoolin  (None, 14, 14, 128)      0
 g2D)

 flatten_9 (Flatten)         (None, 25088)             0

 dense_22 (Dense)            (None, 256)               6422784

 dense_23 (Dense)            (None, 1)                 257

=================================================================
Total params: 6,524,961
Trainable params: 6,524,961
Non-trainable params: 0
_________________________________________________________________
```

In [67]:

```
cnn_model_v10_results = cnn_model_v10.fit(X_train, y_train,
                                        batch_size=BATCH_SIZE,
                                        epochs=EPOCHS,
                                        callbacks=CALLBACKS,
                                        class_weight=class_weights,
                                        validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.5282 - tp: 2558.0000 - fp: 32
3.0000 - tn: 1026.0000 - fn: 1267.0000 - accuracy: 0.6927 - precision: 0.8879 - recall:
0.6688 - auc: 0.8123 - prc: 0.9193
Epoch 1: val_loss did not improve from 0.25460
41/41 [==============================] - 148s 4s/step - loss: 0.5282 - tp: 2558.0000 - f
p: 323.0000 - tn: 1026.0000 - fn: 1267.0000 - accuracy: 0.6927 - precision: 0.8879 - rec
all: 0.6688 - auc: 0.8123 - prc: 0.9193 - val_loss: 0.3709 - val_tp: 6.0000 - val_fp: 0.
0000e+00 - val_tn: 8.0000 - val_fn: 2.0000 - val_accuracy: 0.8750 - val_precision: 1.000
0 - val_recall: 0.7500 - val_auc: 0.9531 - val_prc: 0.9643
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1958 - tp: 3509.0000 - fp: 10
4.0000 - tn: 1237.0000 - fn: 308.0000 - accuracy: 0.9201 - precision: 0.9712 - recall:
0.9193 - auc: 0.9770 - prc: 0.9914
Epoch 2: val_loss did not improve from 0.25460
41/41 [==============================] - 147s 4s/step - loss: 0.1958 - tp: 3509.0000 - f
p: 104.0000 - tn: 1237.0000 - fn: 308.0000 - accuracy: 0.9201 - precision: 0.9712 - reca
ll: 0.9193 - auc: 0.9770 - prc: 0.9914 - val_loss: 0.5579 - val_tp: 8.0000 - val_fp: 4.0
000 - val_tn: 4.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.7500 - val_precision: 0.6667
- val_recall: 1.0000 - val_auc: 0.8906 - val_prc: 0.9014
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1398 - tp: 3612.0000 - fp: 7
7.0000 - tn: 1264.0000 - fn: 205.0000 - accuracy: 0.9453 - precision: 0.9791 - recall:
0.9463 - auc: 0.9878 - prc: 0.9955
Epoch 3: val_loss did not improve from 0.25460
41/41 [==============================] - 179s 4s/step - loss: 0.1398 - tp: 3612.0000 - f
p: 77.0000 - tn: 1264.0000 - fn: 205.0000 - accuracy: 0.9453 - precision: 0.9791 - recal
l: 0.9463 - auc: 0.9878 - prc: 0.9955 - val_loss: 0.6484 - val_tp: 7.0000 - val_fp: 3.00
00 - val_tn: 5.0000 - val_fn: 1.0000 - val_accuracy: 0.7500 - val_precision: 0.7000 - va
l_recall: 0.8750 - val_auc: 0.8359 - val_prc: 0.8158
```

In [68]:

```
full_report(cnn_model_v10, cnn_model_v10_results)
```

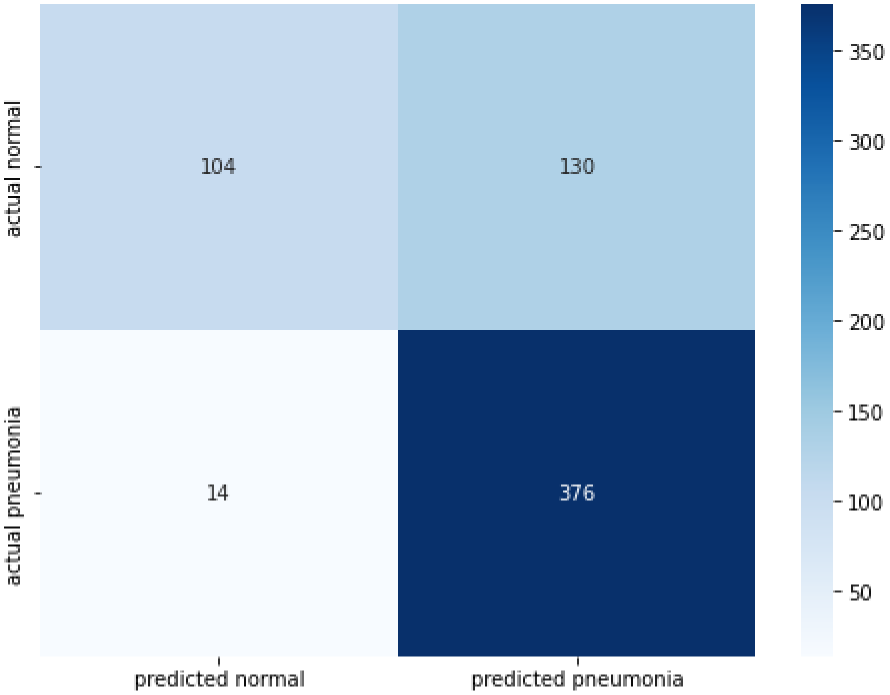
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
```

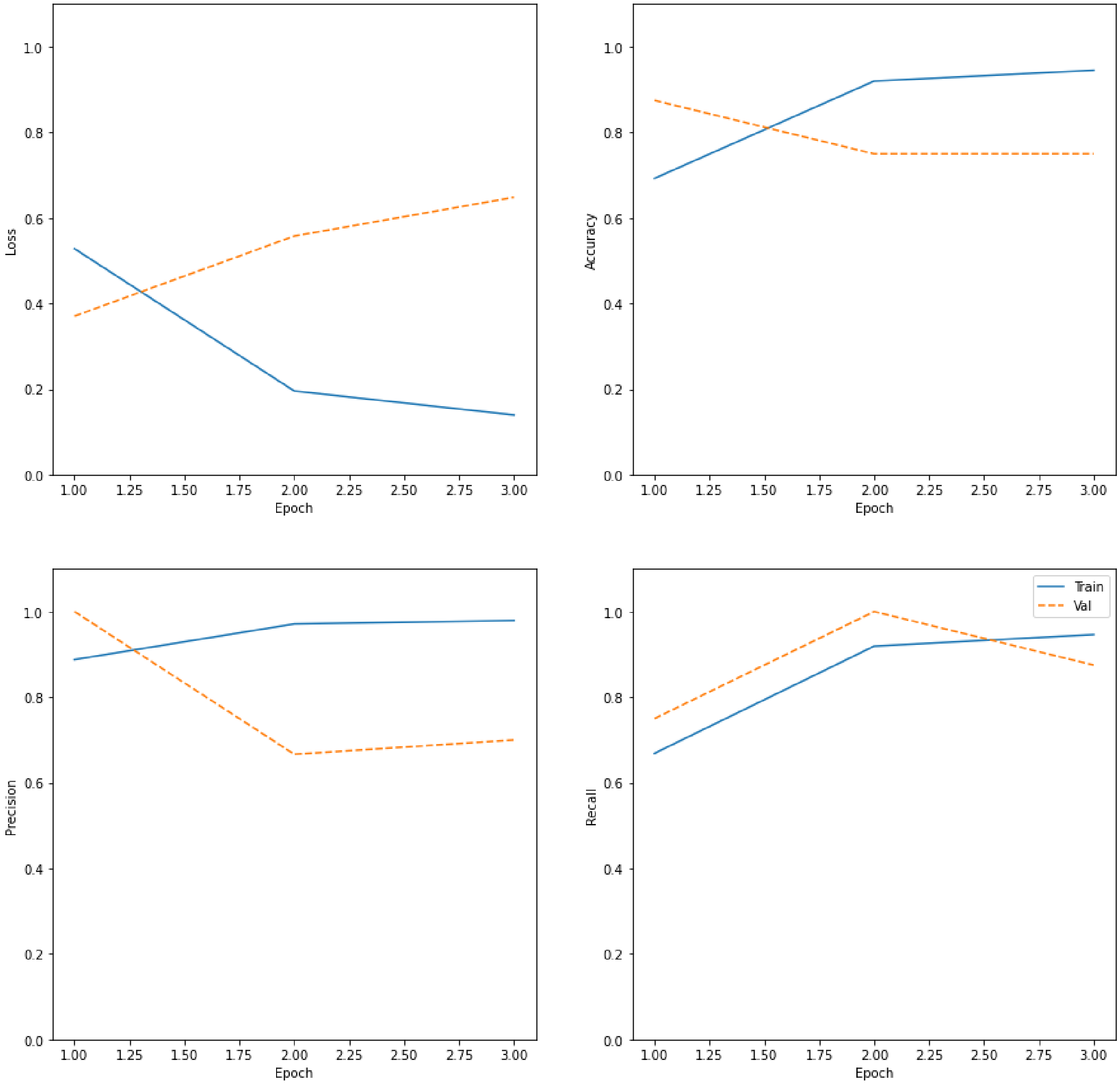
```
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.88      0.44      0.59       234
   pneumonia       0.74      0.96      0.84       390

    accuracy                           0.77       624
   macro avg       0.81      0.70      0.72       624
weighted avg       0.79      0.77      0.75       624
```





CNN model v11 - current best overall accuracy

Still not too much improvement off our best model, lets add one more layer of convolution with 32 filters to our model v8.

In [69]:
```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(16, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v11 = make_cnn_model()
cnn_model_v11.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_38 (Conv2D)          (None, 254, 254, 16)      160       
                                                                 
 max_pooling2d_38 (MaxPoolin  (None, 127, 127, 16)     0         
 g2D)                                                            
                                                                 
 conv2d_39 (Conv2D)          (None, 125, 125, 16)      2320      
                                                                 
 max_pooling2d_39 (MaxPoolin  (None, 62, 62, 16)       0         
 g2D)                                                            
                                                                 
 conv2d_40 (Conv2D)          (None, 60, 60, 32)        4640      
                                                                 
 max_pooling2d_40 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_41 (Conv2D)          (None, 28, 28, 32)        9248      
                                                                 
 max_pooling2d_41 (MaxPoolin  (None, 14, 14, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_42 (Conv2D)          (None, 12, 12, 64)        18496     
                                                                 
 max_pooling2d_42 (MaxPoolin  (None, 6, 6, 64)         0         
 g2D)                                                            
                                                                 
 flatten_10 (Flatten)        (None, 2304)              0         
                                                                 
 dense_24 (Dense)            (None, 256)               590080    
                                                                 
 dense_25 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 625,201
Trainable params: 625,201
Non-trainable params: 0
_________________________________________________________________
```

In [70]:

```
cnn_model_v11_results = cnn_model_v11.fit(X_train, y_train,
                                          batch_size=BATCH_SIZE,
                                          epochs=EPOCHS,
                                          callbacks=CALLBACKS,
                                          class_weight=class_weights,
                                          validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.5210 - tp: 2086.0000 - fp: 15
1.0000 - tn: 1198.0000 - fn: 1739.0000 - accuracy: 0.6347 - precision: 0.9325 - recall:
0.5454 - auc: 0.8434 - prc: 0.9345
Epoch 1: val_loss did not improve from 0.25460
41/41 [==============================] - 85s 2s/step - loss: 0.5210 - tp: 2086.0000 - f
p: 151.0000 - tn: 1198.0000 - fn: 1739.0000 - accuracy: 0.6347 - precision: 0.9325 - rec
all: 0.5454 - auc: 0.8434 - prc: 0.9345 - val_loss: 0.2822 - val_tp: 7.0000 - val_fp: 0.
0000e+00 - val_tn: 8.0000 - val_fn: 1.0000 - val_accuracy: 0.9375 - val_precision: 1.000
0 - val_recall: 0.8750 - val_auc: 1.0000 - val_prc: 1.0000
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1963 - tp: 3499.0000 - fp: 9
6.0000 - tn: 1245.0000 - fn: 318.0000 - accuracy: 0.9197 - precision: 0.9733 - recall:
0.9167 - auc: 0.9770 - prc: 0.9921
Epoch 2: val_loss did not improve from 0.25460
41/41 [==============================] - 91s 2s/step - loss: 0.1963 - tp: 3499.0000 - f
p: 96.0000 - tn: 1245.0000 - fn: 318.0000 - accuracy: 0.9197 - precision: 0.9733 - recal
l: 0.9167 - auc: 0.9770 - prc: 0.9921 - val_loss: 0.2719 - val_tp: 8.0000 - val_fp: 0.00
00e+00 - val_tn: 8.0000 - val_fn: 0.0000e+00 - val_accuracy: 1.0000 - val_precision: 1.0
000 - val_recall: 1.0000 - val_auc: 1.0000 - val_prc: 1.0000
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1719 - tp: 3557.0000 - fp: 9
0.0000 - tn: 1251.0000 - fn: 260.0000 - accuracy: 0.9321 - precision: 0.9753 - recall:
0.9319 - auc: 0.9822 - prc: 0.9938
Epoch 3: val_loss did not improve from 0.25460
41/41 [==============================] - 86s 2s/step - loss: 0.1719 - tp: 3557.0000 - f
p: 90.0000 - tn: 1251.0000 - fn: 260.0000 - accuracy: 0.9321 - precision: 0.9753 - recal
l: 0.9319 - auc: 0.9822 - prc: 0.9938 - val_loss: 0.3318 - val_tp: 7.0000 - val_fp: 3.00
00 - val_tn: 5.0000 - val_fn: 1.0000 - val_accuracy: 0.7500 - val_precision: 0.7000 - va
l_recall: 0.8750 - val_auc: 0.9531 - val_prc: 0.9643
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 0.1148 - tp: 3641.0000 - fp: 4
7.0000 - tn: 1294.0000 - fn: 176.0000 - accuracy: 0.9568 - precision: 0.9873 - recall:
0.9539 - auc: 0.9917 - prc: 0.9972
Epoch 4: val_loss did not improve from 0.25460
41/41 [==============================] - 87s 2s/step - loss: 0.1148 - tp: 3641.0000 - f
p: 47.0000 - tn: 1294.0000 - fn: 176.0000 - accuracy: 0.9568 - precision: 0.9873 - recal
l: 0.9539 - auc: 0.9917 - prc: 0.9972 - val_loss: 0.6745 - val_tp: 8.0000 - val_fp: 6.00
00 - val_tn: 2.0000 - val_fn: 0.0000e+00 - val_accuracy: 0.6250 - val_precision: 0.5714
- val_recall: 1.0000 - val_auc: 0.9062 - val_prc: 0.9333
```

In [71]:

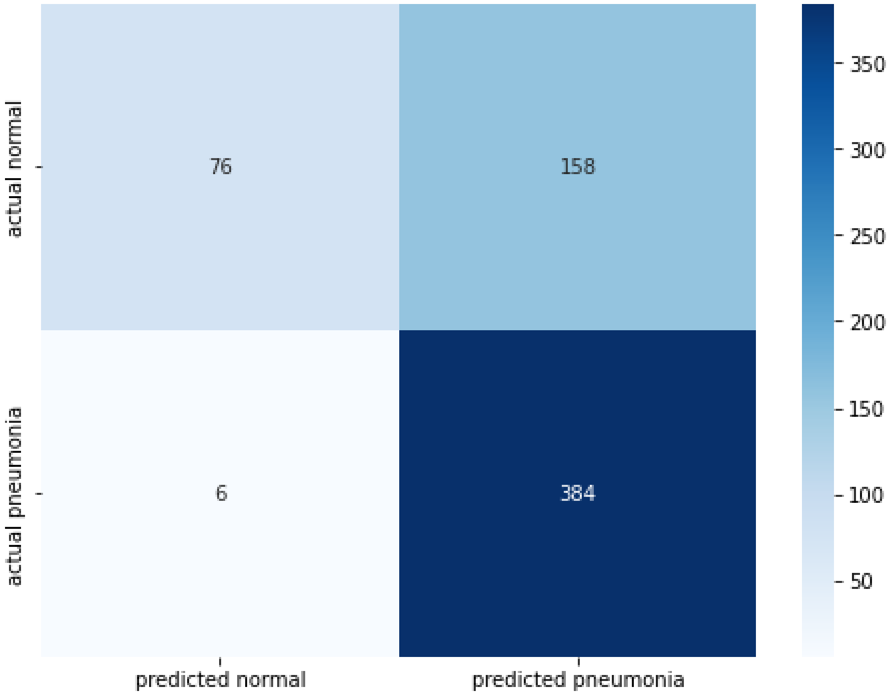
```
full_report(cnn_model_v11, cnn_model_v11_results)
```

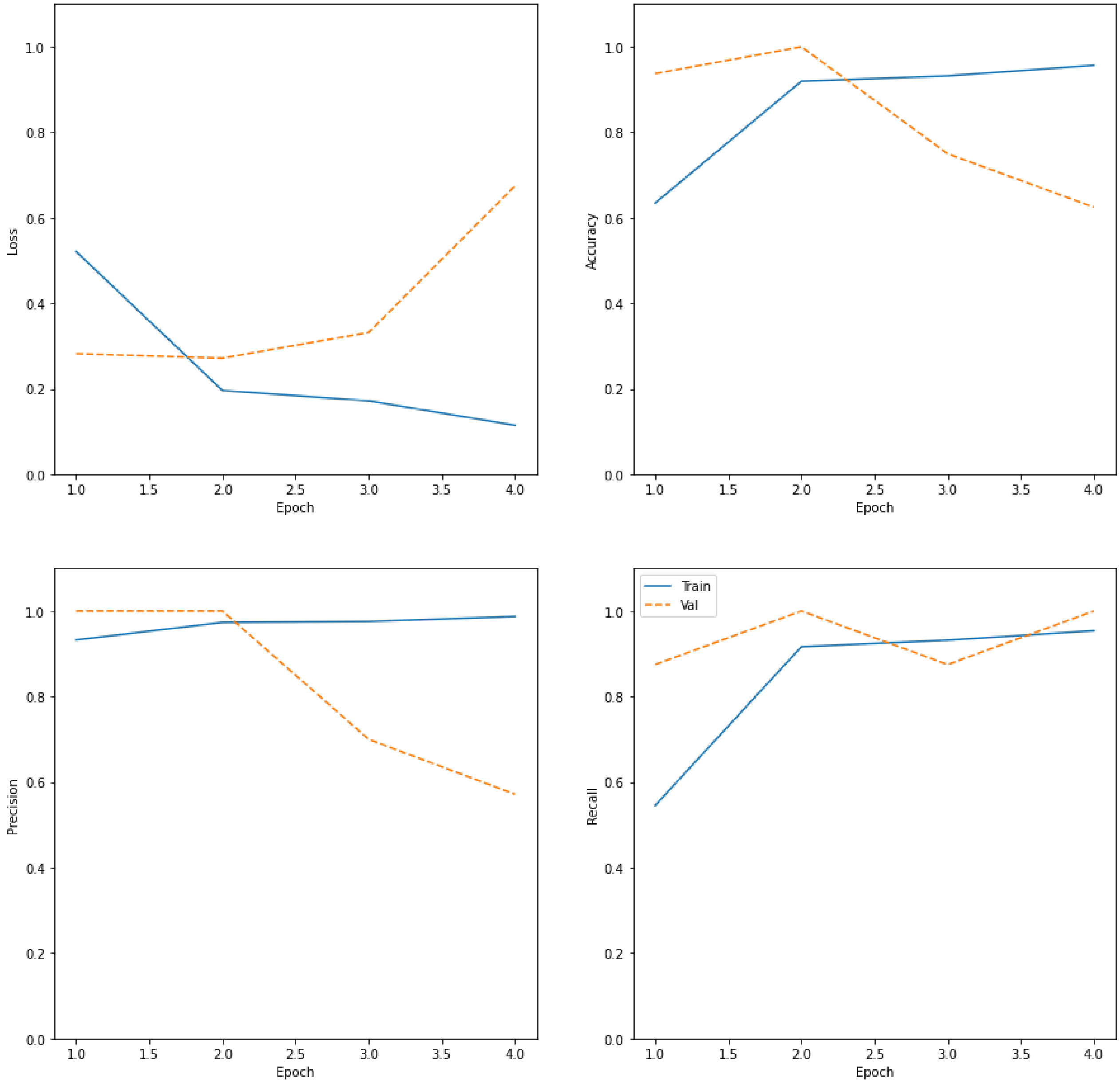
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------


              precision    recall  f1-score   support
```

```
      normal       0.93      0.32      0.48       234
   pneumonia       0.71      0.98      0.82       390

    accuracy                           0.74       624
   macro avg       0.82      0.65      0.65       624
weighted avg       0.79      0.74      0.70       624
```





The first time this model was trained it was the best overall. The computer restarted and in order to prepare the necessary deliverables I needed to rerun all the cells. This time the version 11 model performed much worse than originally seen.

In [ ]:

```
# cnn_model_v11.save(filepath="./models/")
```

CNN model v12 (11.1)

Version 11 was pretty good, and our early stopping callback was not triggered. Lets run it for more epochs to see if we can get a new best model.

*this statement is no longer valid given the changes when we reran the training for model v11. Thankfully, that model was saved for us to bring back in at the end of the notebook. We also therefore don't need to run this model as it's the same as the last.

In [72]:

```
def make_cnn_model():
    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(16, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v12 = make_cnn_model()
cnn_model_v12.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_43 (Conv2D)          (None, 254, 254, 16)      160       
                                                                 
 max_pooling2d_43 (MaxPoolin  (None, 127, 127, 16)     0         
 g2D)                                                            
                                                                 
 conv2d_44 (Conv2D)          (None, 125, 125, 16)      2320      
                                                                 
 max_pooling2d_44 (MaxPoolin  (None, 62, 62, 16)       0         
 g2D)                                                            
                                                                 
 conv2d_45 (Conv2D)          (None, 60, 60, 32)        4640      
                                                                 
 max_pooling2d_45 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_46 (Conv2D)          (None, 28, 28, 32)        9248      
                                                                 
 max_pooling2d_46 (MaxPoolin  (None, 14, 14, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_47 (Conv2D)          (None, 12, 12, 64)        18496     
                                                                 
 max_pooling2d_47 (MaxPoolin  (None, 6, 6, 64)         0         
 g2D)

 flatten_11 (Flatten)        (None, 2304)              0

 dense_26 (Dense)            (None, 256)               590080

 dense_27 (Dense)            (None, 1)                 257

=================================================================
Total params: 625,201
Trainable params: 625,201
Non-trainable params: 0
_________________________________________________________________
```

In [73]:

```
# cnn_model_v12_results = cnn_model_v12.fit(X_train, y_train,
#                                           batch_size=BATCH_SIZE,
#                                           epochs=EPOCHS,
#                                           callbacks=CALLBACKS,
#                                           class_weight=class_weights,
#                                           validation_data=(X_val, y_val))
```

In [74]:

```
# full_report(cnn_model_v12, cnn_model_v12_results)
```

CNN model v13 (11.2)

Lets try increasing the value of each layers filter by double.

In [75]:

```
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(32, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(128, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)
    return model


cnn_model_v13 = make_cnn_model()
cnn_model_v13.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_48 (Conv2D)          (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d_48 (MaxPoolin  (None, 127, 127, 32)     0         
 g2D)                                                            
                                                                 
 conv2d_49 (Conv2D)          (None, 125, 125, 32)      9248      
                                                                 
 max_pooling2d_49 (MaxPoolin  (None, 62, 62, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_50 (Conv2D)          (None, 60, 60, 64)        18496     
                                                                 
 max_pooling2d_50 (MaxPoolin  (None, 30, 30, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_51 (Conv2D)          (None, 28, 28, 64)        36928     
                                                                 
 max_pooling2d_51 (MaxPoolin  (None, 14, 14, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_52 (Conv2D)          (None, 12, 12, 128)       73856     
                                                                 
 max_pooling2d_52 (MaxPoolin  (None, 6, 6, 128)        0         
 g2D)                                                            
                                                                 
 flatten_12 (Flatten)        (None, 4608)              0         
                                                                 
 dense_28 (Dense)            (None, 256)               1179904   
                                                                 
 dense_29 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 1,319,009
Trainable params: 1,319,009
Non-trainable params: 0
_________________________________________________________________
```

In [76]:

```
cnn_model_v13_results = cnn_model_v13.fit(X_train, y_train,
                                          batch_size=BATCH_SIZE,
                                          epochs=EPOCHS,
                                          callbacks=CALLBACKS,
                                          class_weight=class_weights,
                                          validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.4991 - tp: 2045.0000 - fp: 14
6.0000 - tn: 1203.0000 - fn: 1780.0000 - accuracy: 0.6278 - precision: 0.9334 - recall:
0.5346 - auc: 0.8402 - prc: 0.9335
Epoch 1: val_loss did not improve from 0.25460
```

```
41/41 [==============================] - 165s 4s/step - loss: 0.4991 - tp: 2045.0000 - f
p: 146.0000 - tn: 1203.0000 - fn: 1780.0000 - accuracy: 0.6278 - precision: 0.9334 - rec
all: 0.5346 - auc: 0.8402 - prc: 0.9335 - val_loss: 0.3772 - val_tp: 7.0000 - val_fp: 0.
0000e+00 - val_tn: 8.0000 - val_fn: 1.0000 - val_accuracy: 0.9375 - val_precision: 1.000
0 - val_recall: 0.8750 - val_auc: 0.8906 - val_prc: 0.9396
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1986 - tp: 3516.0000 - fp: 10
1.0000 - tn: 1240.0000 - fn: 301.0000 - accuracy: 0.9221 - precision: 0.9721 - recall:
0.9211 - auc: 0.9761 - prc: 0.9913
Epoch 2: val_loss did not improve from 0.25460
41/41 [==============================] - 171s 4s/step - loss: 0.1986 - tp: 3516.0000 - f
p: 101.0000 - tn: 1240.0000 - fn: 301.0000 - accuracy: 0.9221 - precision: 0.9721 - reca
ll: 0.9211 - auc: 0.9761 - prc: 0.9913 - val_loss: 0.3719 - val_tp: 7.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - v
al_recall: 0.8750 - val_auc: 0.8906 - val_prc: 0.9396
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1560 - tp: 3579.0000 - fp: 7
7.0000 - tn: 1264.0000 - fn: 238.0000 - accuracy: 0.9389 - precision: 0.9789 - recall:
0.9376 - auc: 0.9851 - prc: 0.9944
Epoch 3: val_loss did not improve from 0.25460
41/41 [==============================] - 176s 4s/step - loss: 0.1560 - tp: 3579.0000 - f
p: 77.0000 - tn: 1264.0000 - fn: 238.0000 - accuracy: 0.9389 - precision: 0.9789 - recal
l: 0.9376 - auc: 0.9851 - prc: 0.9944 - val_loss: 0.4429 - val_tp: 7.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - va
l_recall: 0.8750 - val_auc: 0.8750 - val_prc: 0.9229
Epoch 4/25
41/41 [==============================] - ETA: 0s - loss: 0.1176 - tp: 3632.0000 - fp: 5
6.0000 - tn: 1285.0000 - fn: 185.0000 - accuracy: 0.9533 - precision: 0.9848 - recall:
0.9515 - auc: 0.9909 - prc: 0.9966
Epoch 4: val_loss did not improve from 0.25460
41/41 [==============================] - 168s 4s/step - loss: 0.1176 - tp: 3632.0000 - f
p: 56.0000 - tn: 1285.0000 - fn: 185.0000 - accuracy: 0.9533 - precision: 0.9848 - recal
l: 0.9515 - auc: 0.9909 - prc: 0.9966 - val_loss: 0.5132 - val_tp: 7.0000 - val_fp: 4.00
00 - val_tn: 4.0000 - val_fn: 1.0000 - val_accuracy: 0.6875 - val_precision: 0.6364 - va
l_recall: 0.8750 - val_auc: 0.8750 - val_prc: 0.8937
```

In [77]:

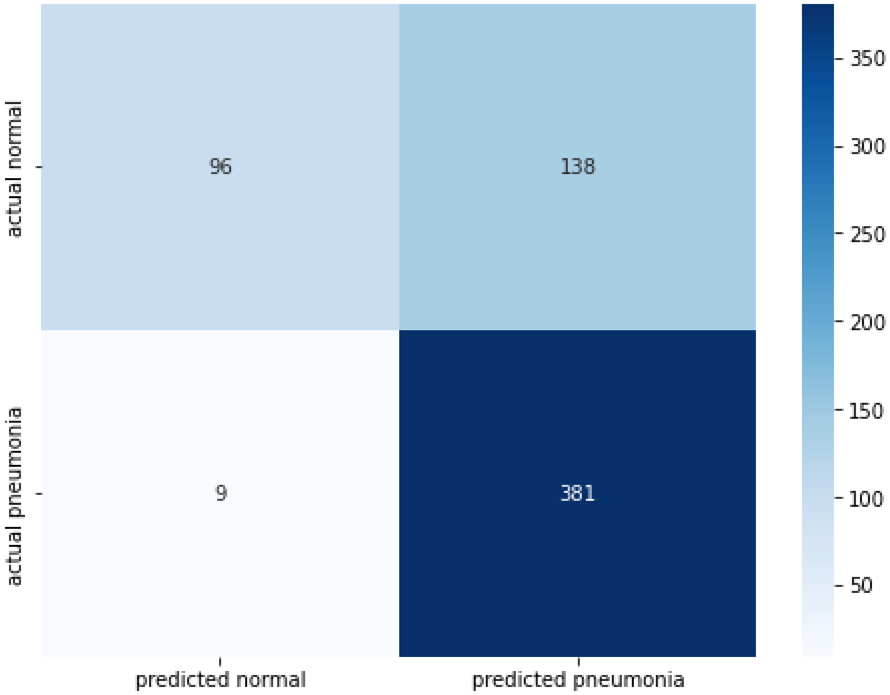
```
full_report(cnn_model_v13, cnn_model_v13_results)
```

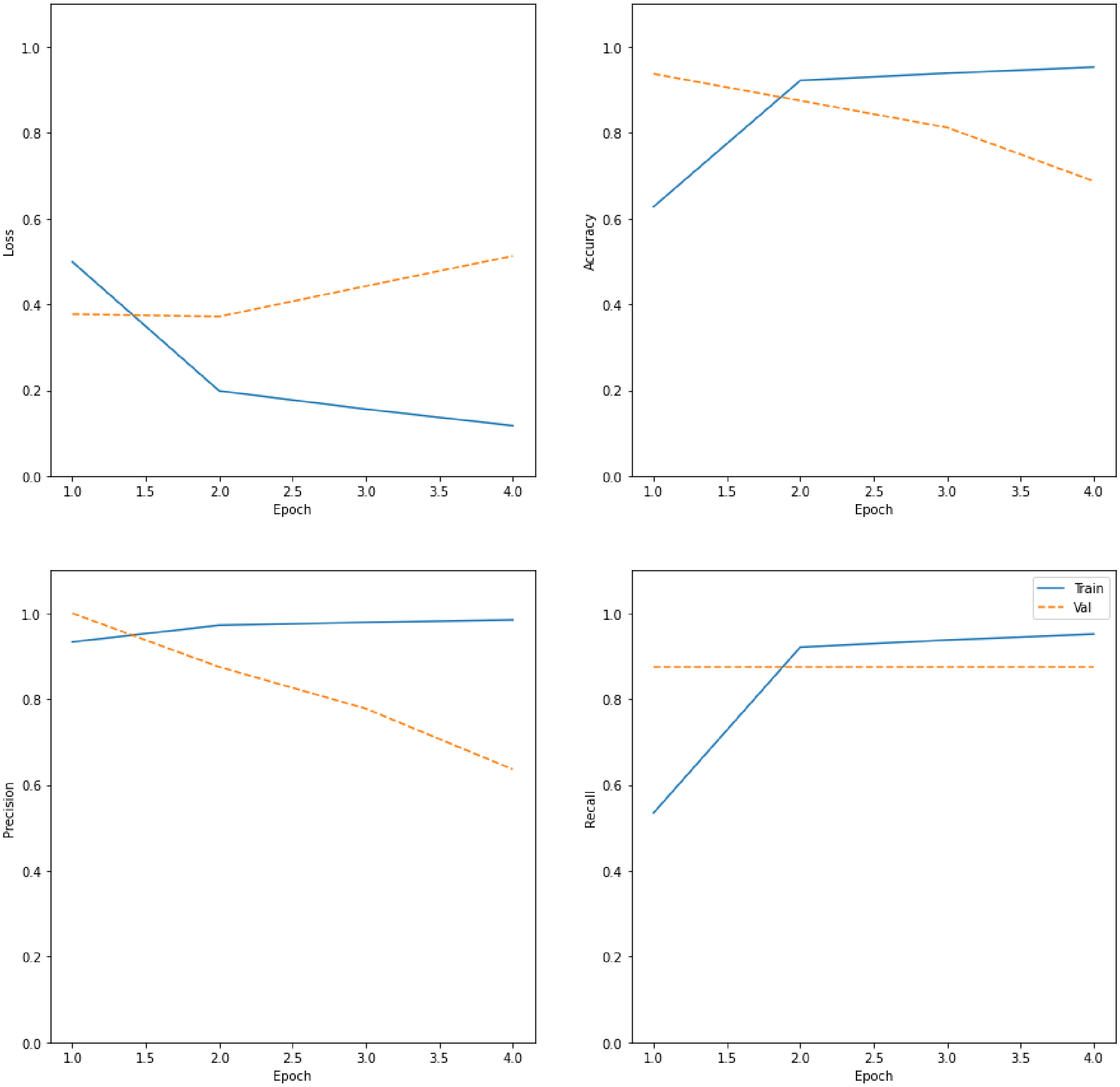
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.91      0.41      0.57       234
   pneumonia       0.73      0.98      0.84       390

    accuracy                           0.76       624
   macro avg       0.82      0.69      0.70       624
weighted avg       0.80      0.76      0.74       624
```


actual normal
actual pneumonia
predicted normal
predicted pneumonia
96
138
9
381
350
300
250
200
150
100
50


Loss
Accuracy
Precision
Recall
Epoch
Train
Val
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

CNN model v14 (11.3)

Lets add some dropout to our version 11 model.

In [78]:

```python
def make_cnn_model():

    model = Sequential()

    model.add(Conv2D(16, (3,3), activation=ACTIVATION, input_shape=INPUT_SHAPE))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(16, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(32, (3,3), activation=ACTIVATION))
    model.add(MaxPooling2D((2,2)))

    model.add(Conv2D(64, (3,3), activation=ACTIVATION))

    model.add(Flatten())

    model.add(Dense(256, activation=ACTIVATION))
    model.add(Dropout(0.3))

    model.add(Dense(1, activation=OUTPUT))

    model.compile(loss=LOSS,
                  optimizer=OPTIMIZER,
                  metrics=METRICS)

    return model

cnn_model_v14 = make_cnn_model()
cnn_model_v14.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_53 (Conv2D)          (None, 254, 254, 16)      160       
                                                                 
 max_pooling2d_53 (MaxPoolin  (None, 127, 127, 16)     0         
 g2D)                                                            
                                                                 
 conv2d_54 (Conv2D)          (None, 125, 125, 16)      2320      
                                                                 
 max_pooling2d_54 (MaxPoolin  (None, 62, 62, 16)       0         
 g2D)                                                            
                                                                 
 conv2d_55 (Conv2D)          (None, 60, 60, 32)        4640      
                                                                 
 max_pooling2d_55 (MaxPoolin  (None, 30, 30, 32)       0         
 g2D)                                                            
 conv2d_56 (Conv2D)          (None, 28, 28, 32)        9248

 max_pooling2d_56 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 conv2d_57 (Conv2D)          (None, 12, 12, 64)        18496

 flatten_13 (Flatten)        (None, 9216)              0

 dense_30 (Dense)            (None, 256)               2359552

 dropout_3 (Dropout)         (None, 256)               0

 dense_31 (Dense)            (None, 1)                 257

=================================================================
Total params: 2,394,673
Trainable params: 2,394,673
Non-trainable params: 0
_________________________________________________________________
```

In [79]:

```
cnn_model_v14_results = cnn_model_v14.fit(X_train, y_train,
                                          batch_size=BATCH_SIZE,
                                          epochs=EPOCHS,
                                          callbacks=CALLBACKS,
                                          class_weight=class_weights,
                                          validation_data=(X_val, y_val))
```

```
Epoch 1/25
41/41 [==============================] - ETA: 0s - loss: 0.3562 - tp: 3104.0000 - fp: 19
1.0000 - tn: 1158.0000 - fn: 721.0000 - accuracy: 0.8237 - precision: 0.9420 - recall:
0.8115 - auc: 0.9215 - prc: 0.9698
Epoch 1: val_loss did not improve from 0.25460
41/41 [==============================] - 90s 2s/step - loss: 0.3562 - tp: 3104.0000 - f
p: 191.0000 - tn: 1158.0000 - fn: 721.0000 - accuracy: 0.8237 - precision: 0.9420 - reca
ll: 0.8115 - auc: 0.9215 - prc: 0.9698 - val_loss: 0.4167 - val_tp: 7.0000 - val_fp: 1.0
000 - val_tn: 7.0000 - val_fn: 1.0000 - val_accuracy: 0.8750 - val_precision: 0.8750 - v
al_recall: 0.8750 - val_auc: 0.8750 - val_prc: 0.9085
Epoch 2/25
41/41 [==============================] - ETA: 0s - loss: 0.1889 - tp: 3505.0000 - fp: 8
3.0000 - tn: 1258.0000 - fn: 312.0000 - accuracy: 0.9234 - precision: 0.9769 - recall:
0.9183 - auc: 0.9780 - prc: 0.9915
Epoch 2: val_loss did not improve from 0.25460
41/41 [==============================] - 86s 2s/step - loss: 0.1889 - tp: 3505.0000 - f
p: 83.0000 - tn: 1258.0000 - fn: 312.0000 - accuracy: 0.9234 - precision: 0.9769 - recal
l: 0.9183 - auc: 0.9780 - prc: 0.9915 - val_loss: 0.4857 - val_tp: 7.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - va
l_recall: 0.8750 - val_auc: 0.8438 - val_prc: 0.8816
Epoch 3/25
41/41 [==============================] - ETA: 0s - loss: 0.1337 - tp: 3612.0000 - fp: 6
5.0000 - tn: 1276.0000 - fn: 205.0000 - accuracy: 0.9477 - precision: 0.9823 - recall:
0.9463 - auc: 0.9884 - prc: 0.9957
Epoch 3: val_loss did not improve from 0.25460
41/41 [==============================] - 86s 2s/step - loss: 0.1337 - tp: 3612.0000 - f
p: 65.0000 - tn: 1276.0000 - fn: 205.0000 - accuracy: 0.9477 - precision: 0.9823 - recal
l: 0.9463 - auc: 0.9884 - prc: 0.9957 - val_loss: 0.5309 - val_tp: 7.0000 - val_fp: 2.00
00 - val_tn: 6.0000 - val_fn: 1.0000 - val_accuracy: 0.8125 - val_precision: 0.7778 - va
l_recall: 0.8750 - val_auc: 0.8281 - val_prc: 0.8588
```

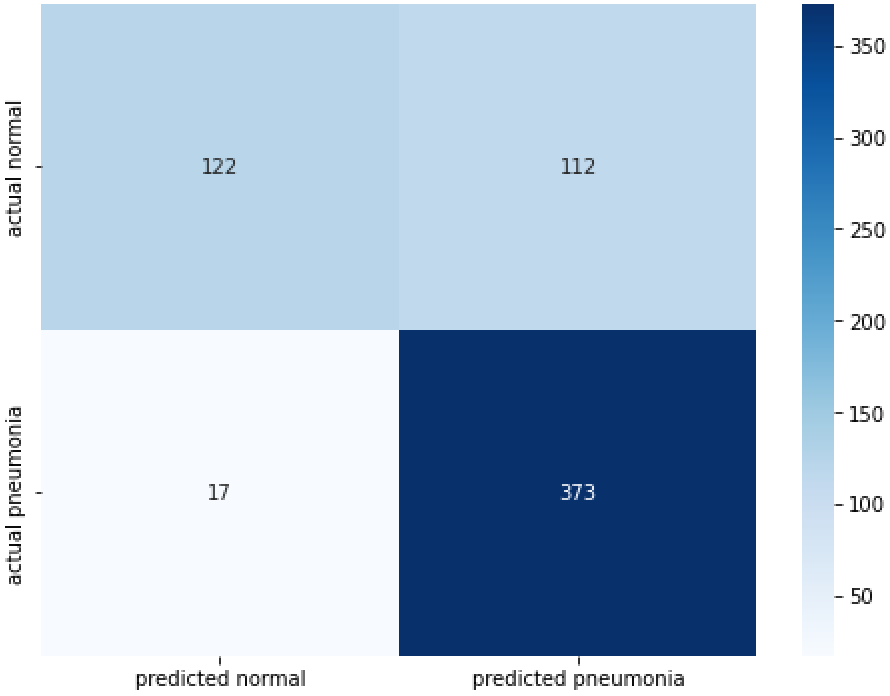
```
In [80]: full_report(cnn_model_v14, cnn_model_v14_results)
```

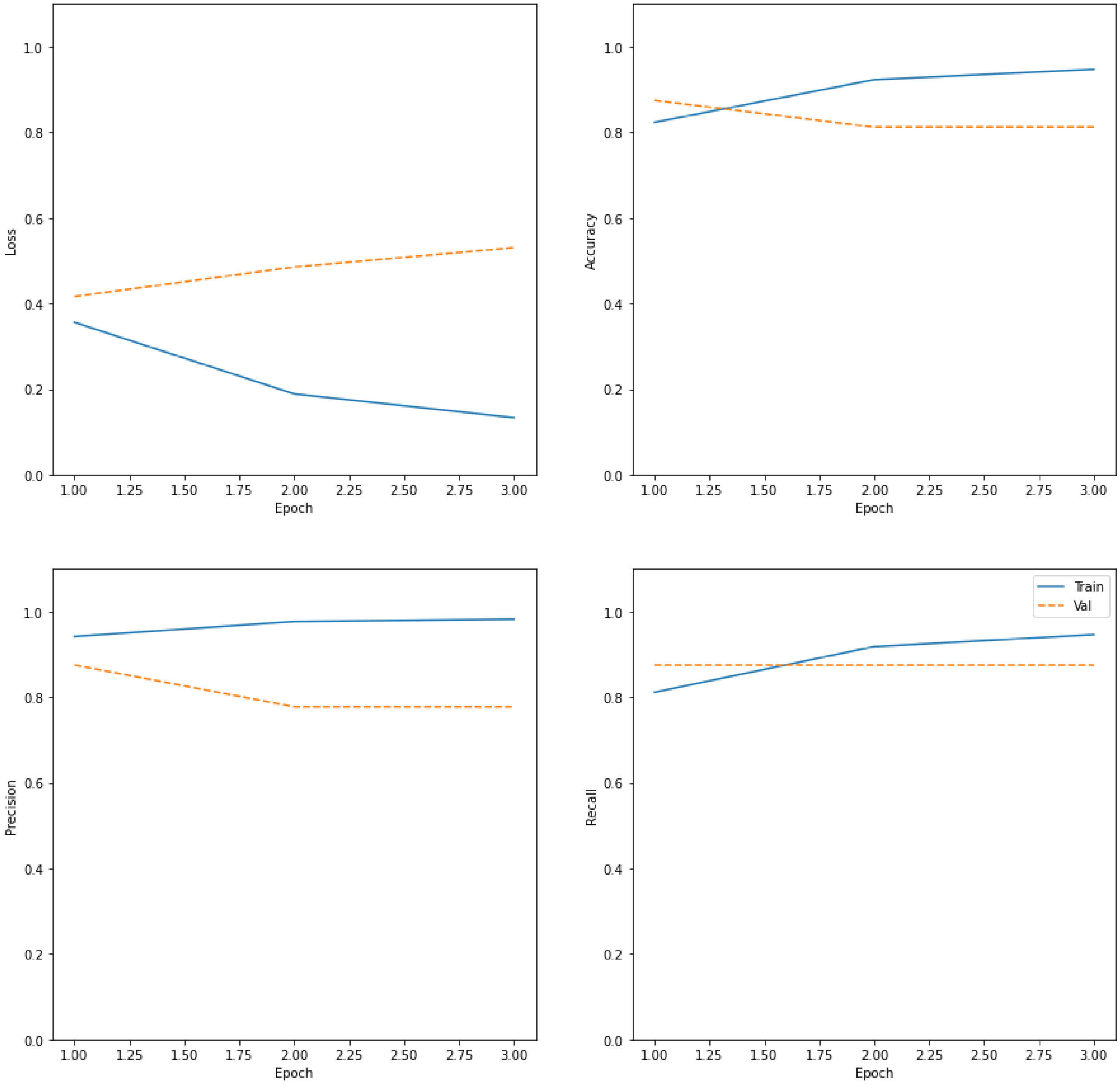
```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support

      normal       0.88      0.52      0.65       234
   pneumonia       0.77      0.96      0.85       390

    accuracy                           0.79       624
   macro avg       0.82      0.74      0.75       624
weighted avg       0.81      0.79      0.78       624
```





Final model

In [81]:

```
final_model = keras.models.load_model('./models/')
```

In [82]:

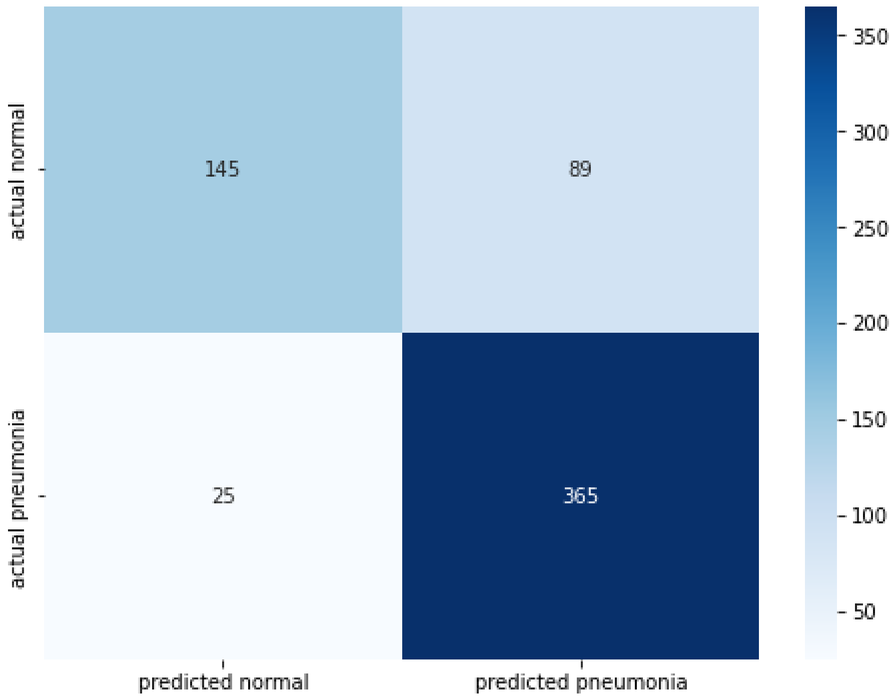
```
full_report(final_model)
```

```
Choose which testing dataset to use for reporting
Input S for training data
Input T for test data
Input V for validation data:
t
---------------------------------------------------------------

              precision    recall  f1-score   support


      normal       0.85      0.62      0.72       234
   pneumonia       0.80      0.94      0.86       390
```

```
    accuracy                           0.82       624
   macro avg       0.83      0.78      0.79       624
weighted avg       0.82      0.82      0.81       624
```



```
In [ ]:
```