

Capstone Project 2 Final Report

Predicting Pneumonia with Chest X-Rays

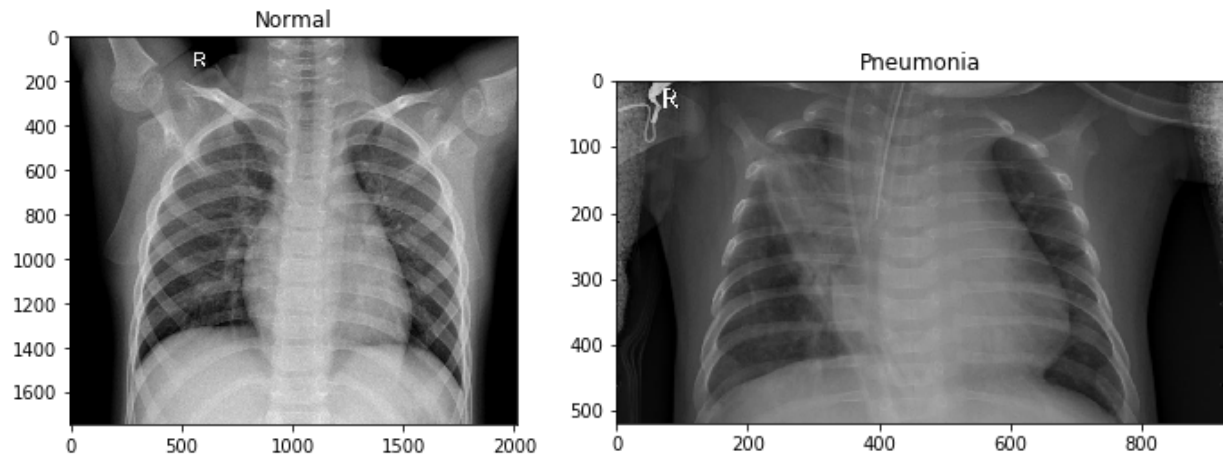
Pneumonia is an infection of the lungs that affects millions of people a year as one of the most common causes for admission to a hospital. To diagnose a patient with pneumonia, chest x-rays are typically taken to a highly trained specialist in order to diagnose the extent and location of the infection. If the process for such a common infection can be streamlined and made more efficient through automation and machine learning, patients would be able to receive treatment faster and doctors would be able to diagnose more patients.

We will be using data sourced from:

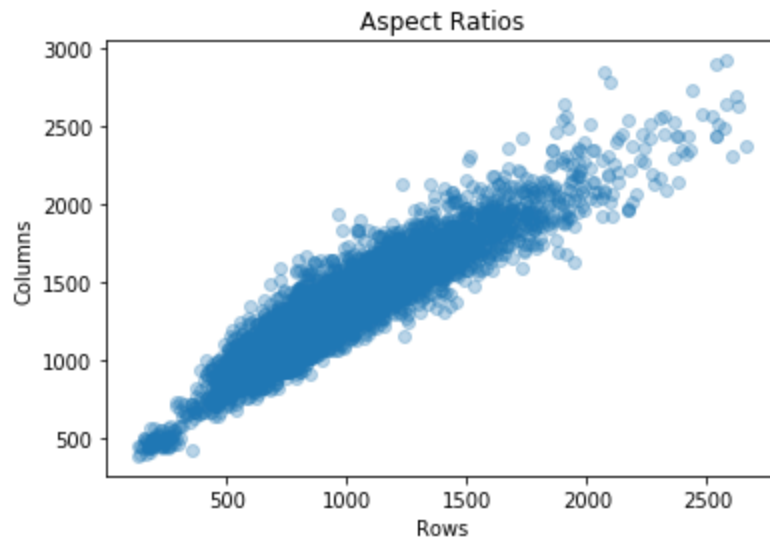
<https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>

The images are grouped by doctors as being normal, or with pneumonia. The photos labeled as pneumonia are also labeled as bacterial or viral. Low quality or unreadable scans were also removed from the data set.

Let's first see some examples of our images labelled pneumonia and normal.



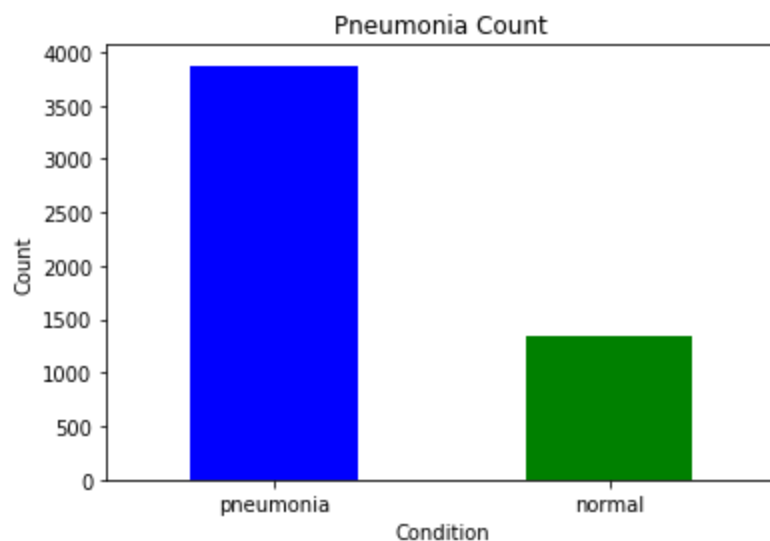
The first thing we note is that the images in the set all vary in size. Let's see what the sizes generally look like.



Mean Aspect Ratio = 1.44

Even though the image sizes vary, we can see that there is a linear relationship between the rows and the columns of an image. Will need to resize the images for our deep learning network, so it may be useful to maintain a similar aspect ratio so that we do not alter the images too much.

Next we should see how many instances of each category (pneumonia and normal) exist in our data set.



We can see that our data set is slightly imbalanced, having more pneumonia images than normal images. This suggests that we may need to be careful when selecting our evaluation metrics, and that simply using accuracy may not be sufficient. We will need to explore the use of recall, precision and F1 scores.

First Model

The data is first shuffled and loaded using an image data generator with a validation split of 0.3. Using the data generators, we also resize the images to (200,300) maintaining an aspect ratio close to that of our mean from exploring the data, and convert the images to grayscale. Using these parameters, we first try to use a simple model as a test run.

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 198, 298, 32)	320
max_pooling2d_4 (MaxPooling2D)	(None, 99, 149, 32)	0
conv2d_5 (Conv2D)	(None, 97, 147, 32)	9248
max_pooling2d_5 (MaxPooling2D)	(None, 48, 73, 32)	0
conv2d_6 (Conv2D)	(None, 46, 71, 64)	18496
max_pooling2d_6 (MaxPooling2D)	(None, 23, 35, 64)	0
batch_normalization_2 (Batch Normalization)	(None, 23, 35, 64)	256
flatten_2 (Flatten)	(None, 51520)	0
dense_3 (Dense)	(None, 128)	6594688
dense_4 (Dense)	(None, 1)	129

With this model architecture we get some great looking results:

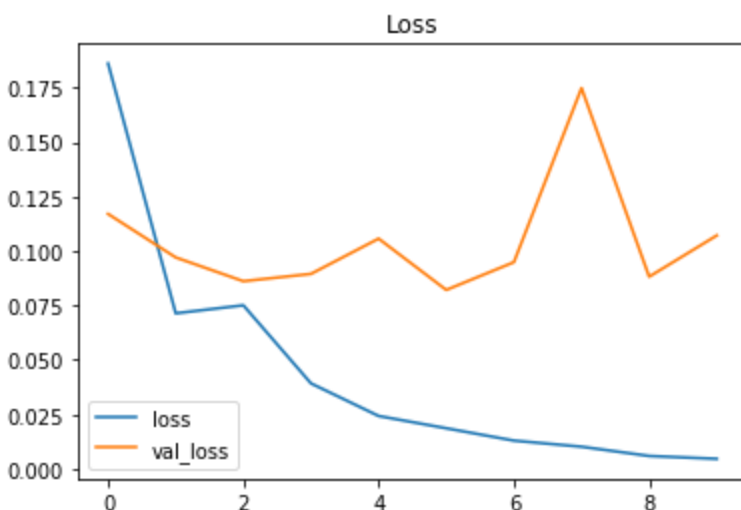
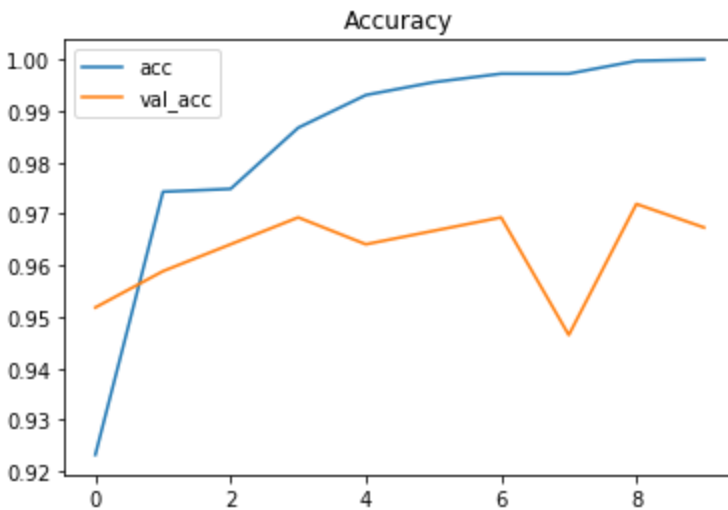
Loss: 0.1072 acc: 0.9674 f1_m: 0.9782 recall_m: 0.9860 precision_m: 0.9714

Upon testing however:

Loss: 1.65 Acc: 0.74 F1 Score: 0.83 Recall: 1.00 Precision: 0.71

This clearly suggests that the model is overfitting.

Taking a look at our accuracy and loss plots during our training epochs, it's clear that our model is not very good. The difference between our validation and training accuracy shows an overfit issue, and our validation loss does not seem to be minimizing very well.



Second Model

Let's try a more complex model and see if that yields better results. We can try adding more layers, adjust the number of filters on the CNN layers, and adjust the number of nodes in the Dense layers. We will also try to address the overfit issue by adding dropout and regularization.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 198, 298, 32)	320
max_pooling2d_7 (MaxPooling2D)	(None, 99, 149, 32)	0
conv2d_8 (Conv2D)	(None, 97, 147, 32)	9248
max_pooling2d_8 (MaxPooling2D)	(None, 48, 73, 32)	0
conv2d_9 (Conv2D)	(None, 46, 71, 64)	18496
max_pooling2d_9 (MaxPooling2D)	(None, 23, 35, 64)	0
conv2d_10 (Conv2D)	(None, 21, 33, 64)	36928
max_pooling2d_10 (MaxPooling2D)	(None, 10, 16, 64)	0
conv2d_11 (Conv2D)	(None, 8, 14, 128)	73856
batch_normalization_3 (Batch Normalization)	(None, 8, 14, 128)	512
max_pooling2d_11 (MaxPooling2D)	(None, 4, 7, 128)	0
flatten_3 (Flatten)	(None, 3584)	0
dense_5 (Dense)	(None, 256)	917760
dense_6 (Dense)	(None, 512)	131584
dropout_1 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 1)	513

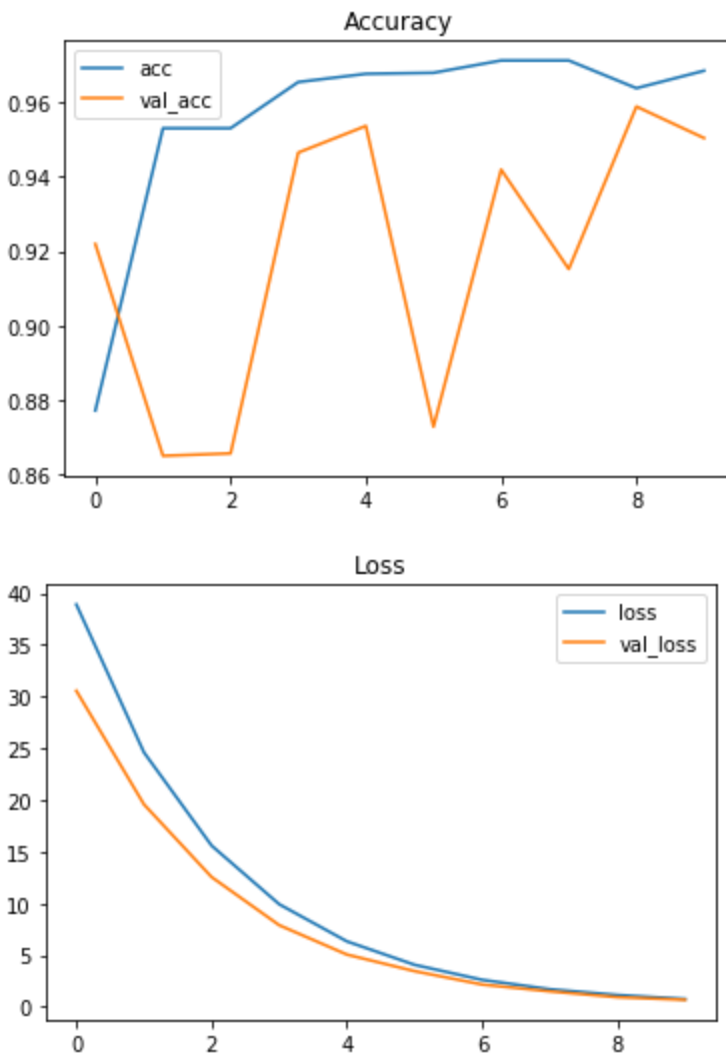
Validation Metrics

loss: 0.6699 acc: 0.9504 f1_m: 0.9671 recall_m: 0.9909 precision_m: 0.9456

Test Metrics

Loss: 1.68 Acc: 0.69 F1 Score: 0.80 Recall: 1.00 Precision: 0.67

We can see that we still have clear overfitting problems based on the differences in the acc and f1 scores.



The accuracy plot looks like a step in the right direction, as the validation accuracy seems to be trending upward. The loss definitely looks much better. The training and validation loss seems to have flattened out and converged, suggesting we are moving in the right direction.

We can continue tweaking and adjusting our model until we get some reasonable results. We can save the models as h5 files so that they can be revisited at a later date.

After several iterations, we finally land on a model architecture:

Saved Model

Layer (type)	Output Shape	Param #
conv2d_46 (Conv2D)	(None, 198, 298, 32)	320
max_pooling2d_37 (MaxPooling)	(None, 99, 149, 32)	0
conv2d_47 (Conv2D)	(None, 97, 147, 32)	9248
max_pooling2d_38 (MaxPooling)	(None, 48, 73, 32)	0
conv2d_48 (Conv2D)	(None, 46, 71, 32)	9248
max_pooling2d_39 (MaxPooling)	(None, 23, 35, 32)	0
conv2d_49 (Conv2D)	(None, 21, 33, 64)	18496
max_pooling2d_40 (MaxPooling)	(None, 10, 16, 64)	0
conv2d_50 (Conv2D)	(None, 8, 14, 128)	73856
batch_normalization_10 (Batch Normalization)	(None, 8, 14, 128)	512
flatten_10 (Flatten)	(None, 14336)	0
dense_19 (Dense)	(None, 64)	917568
dense_20 (Dense)	(None, 1)	65

From the model metrics we can see that this model with our saved weights does not have an obvious overfit issues, while maintaining a fairly high F1 score, recall, and precision scores.

TRAINING

Loss: 0.25 Acc: 0.92 F1 Score: 0.94 Recall: 0.89 Precision: 1.00

VALIDATION

Loss: 0.26 Acc: 0.92 F1 Score: 0.94 Recall: 0.90 Precision: 0.99

TEST

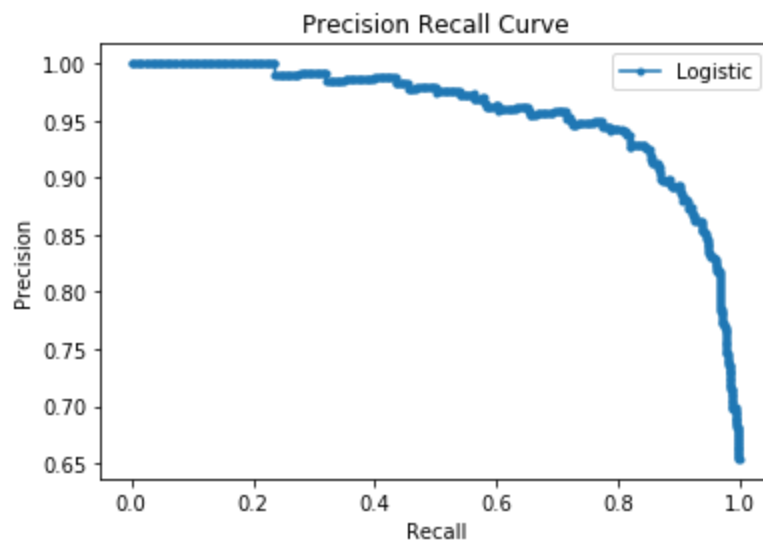
Loss: 0.37 Acc: 0.87 F1 Score: 0.89 Recall: 0.92 Precision: 0.87

We can see that the difference between the validation and test scores is not as extreme, suggesting that this model is not overfitting as much as the previous ones and is doing a much better job of generalizing to new data.

In this model we have maintained a reasonably high recall score of 0.92. It should be noted that when dealing with healthcare predictions, it may be prudent to keep a high recall to limit the number of false negatives since missing a diagnosis could potentially have much higher costs than incorrectly diagnosing a case as positive. Any changes or improvements made to the model should take this into consideration and try to keep the recall of the model as high as possible.

Precision Recall Curve

Since we have a slight imbalance, we should look at a precision recall curve and AUC score to get an idea of how our model performs instead of the usual ROC curve.

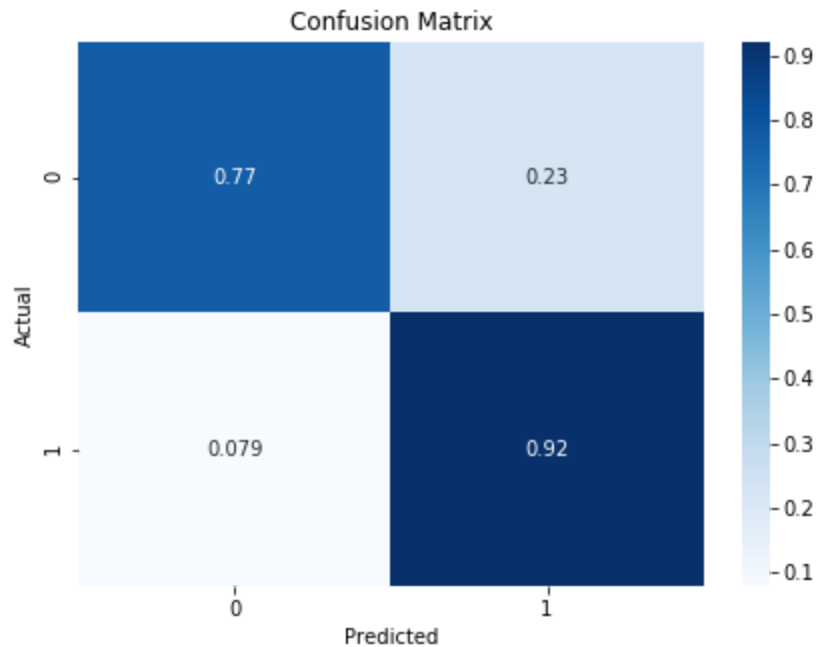


AUC Score: 0.957

The precision recall curve looks good, as does the AUC score.

Confusion Matrix

Finally, let's take a look at our confusion matrix so that we can visualize how our classification model performs.



Conclusion

We have a model that predicts pneumonia from chest x-rays reasonably well. The initial model architecture did not generalize and learn well based on the metrics and loss plots, and the 2nd iteration of the model also did not perform well. An overly complex model may have contributed to the overfitting in some of the iterations. There seems to be a sweet spot that balances overfitting a complex model with producing high quality results. We can continue to adjust the model and its parameters to see if better results can be achieved.