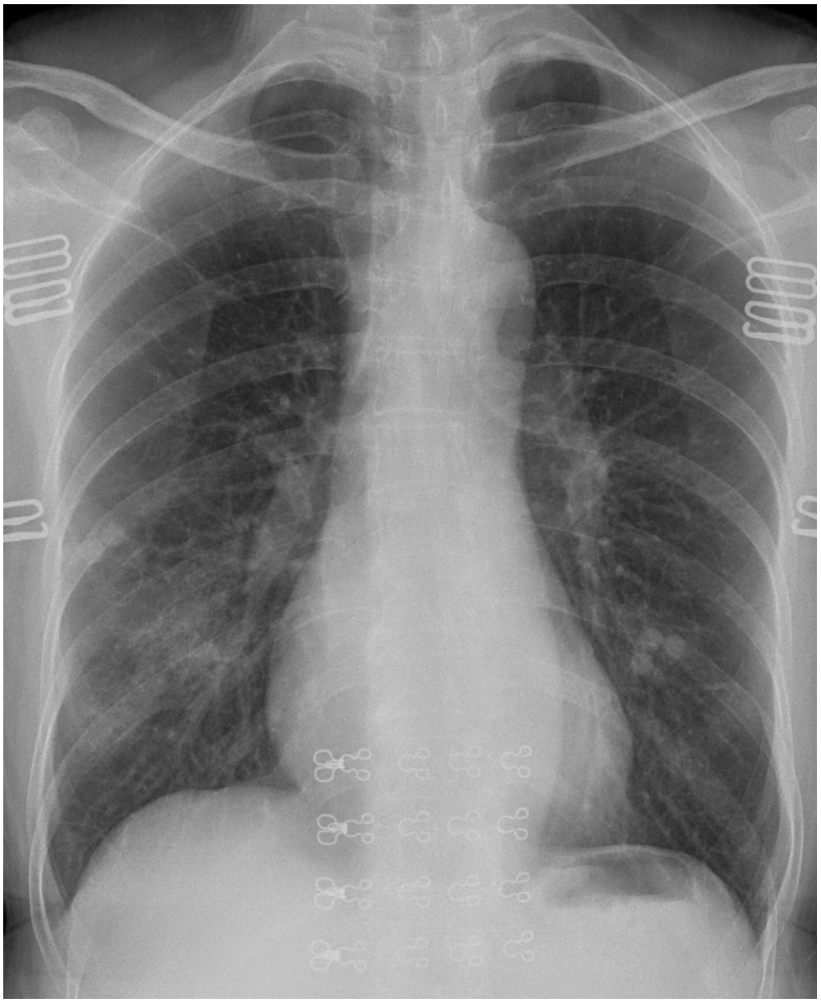
Predicting Lung and Heart Disease From Chest X-Rays



Angad Pfuetzner

Table of Contents

Predicting Lung and Heart Disease From Chest X-Rays

Problem Statement

Chest X-rays have long been used to identify potential heart and lung problems in patients. Chest X-rays are quite common and are often among the first procedures a doctor recommends if they suspect heart or lung disease. Radiologists, doctors that are trained in interpreting X-rays, then analyze the images to determine any potential problems.

Since X-rays are so common, there are countless X-ray images produced each year. Doctors therefore spend many hours analyzing the images and can still make mistakes. Developing an application that would be able to analyze these X-ray images and label potential diseases that the patient may have would therefore be invaluable. It could serve as an independent tool to free up doctors' time so they may see more patients or as an assistive tool to pre-screen images and give doctors the opportunity to focus on patients that may be at much higher risk.

Data Access

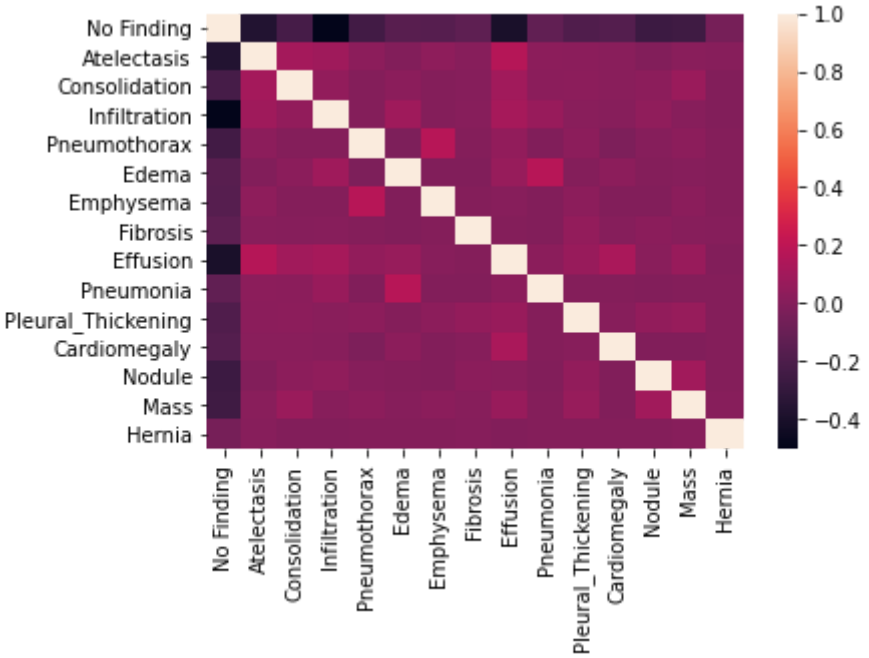
The data I used is the National Institutes of Health Chest X-Ray Dataset. I used an unofficial TFRecord version of the dataset which can be found here https://www.kaggle.com/nickuzmenkov/nih-chest-xrays-tfrecords. The dataset contains 112 120 chest x-ray images from 30 805 unique patients each depicting none, one or several of the 14 diseases in the dataset. The original dataset was labeled using NLP which may have led to some incorrect labels but the labeling accuracy is believed to be >90%.

I used the Kaggle API to load the dataset into a cloud based notebook and performed the necessary cleaning and wrangling steps there.

Data Wrangling and Exploratory Data Analysis

After downloading and unzipping the dataset I had access to 256 TFRecord files containing the images and their respective disease labels, and a csv file summarizing the data by listing every image name and the corresponding diseases. Each X-ray image can be labelled by a No Finding label or by one or more of the following disease labels: Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural Thickening, Cardiomegaly, Nodule, Mass, Hernia.

I began by exploring the data contained in the csv file. There were no missing or null values in the dataset. Next, I checked for any correlations between diseases and produced the following heatmap:



From the heatmap it can be seen that there are no strong correlations other than with No Finding, which makes sense since if the No Finding variable is True then all other disease variables would be false.

Next, I looked at how many data points there were for each of the diseases and produced the following table:

Label	# of Images
No Finding	60 361
Atelectasis	11 559
Consolidation	4 667
Infiltration	19 894
Pneumothorax	5302
Edema	2303
Emphysema	2516
Fibrosis	1 686
Effusion	13 317
Pneumonia	1 431
Pleural Thickening	3 385
Cardiomegaly	2 776
Nodule	6 331

Mass	5 782
Hernia	227

Since some of the diseases have very few data points it would be more logical to treat this as a binary classification problem distinguishing between a patient having a disease, and a patient having no disease (No Finding). This also eliminates the complexity of having to predict multiple class labels for a single image.

Preprocessing and Modelling

Equalization

The first preprocessing method I investigated was equalization which is very common for medical images. The two types of equalization explored were histogram equalization and adaptive histogram equalization. The effects of these equalizations can be seen here:

Original Image

Histogram Equalization

Adaptive Histogram Equalization

The adaptive histogram equalized image seemed to be the best so adaptive histogram equalization was used as a preprocessing step. However after comparing models using the original image against models using the adaptive equalized image there was no significant improvement in classification accuracy and a slight increase in training time, so I decided to stick with the original images for model building.

Resizing

The next parameter to look at was the size of the images. The original images were all 768 x 768 pixels which is decently large. I visualized the images after downsizing to 256 x 256 and 128 x 128 which both seemed to retain enough of the images features. The models built using the 256 x 256 pixel images had slightly increased accuracy predictions at the cost of increased model training time. Images smaller than 128 x 128 pixels did not retain enough features and the models built were therefore much less accurate.

Neural Network Architecture

Several network architectures were considered for the model. The two that performed the best are described here.

Network 1

The first network considered was a structure utilizing the MobileNet architecture, since it is often a good starting point. The MobileNet structure was followed by a global average pooling layer, a dropout layer, a dense layer, another dropout layer and a final dense output layer. The dropout layers were essential for avoiding overfitting. The structure of the network made using Keras, can be seen here:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_256 (Function (None, 8, 8, 1024)        3228288
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1024)              0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 1026
=================================================================
Total params: 3,754,114
Trainable params: 3,732,226
Non-trainable params: 21,888
_________________________________________________________________
```

Network 2

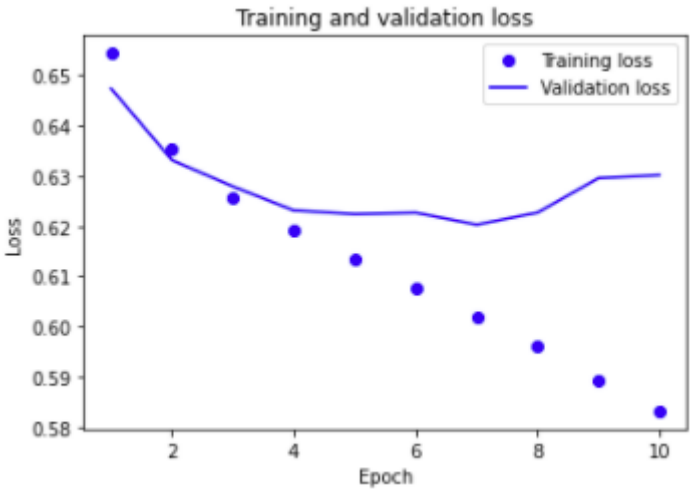
The architecture that performed the best was a network with 4 convolution/MaxPooling layers followed by a flattening layer feeding into a dense layer and finally the output layer. The structure of the network made using Keras, can be seen here:

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 (MaxPooling2	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense_4 (Dense)	(None, 256)	6422784
dense_5 (Dense)	(None, 2)	514

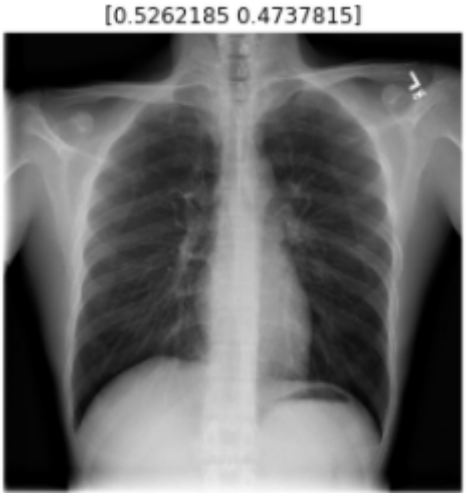
Total params: 6,663,554
Trainable params: 6,663,554
Non-trainable params: 0

Results

The network built on the MobileNet architecture was outperformed by the network built from scratch so only the results of the latter will be explored. For the final model, the batch size was 128 and it was trained for 10 epochs. The results of the accuracy and loss can be seen here:



As seen in the above plots, the optimal number of epochs to avoid overfitting was 7. The peak validation accuracy was 67% with the minimum loss at 0.62. After 7 epochs we can see the accuracy for the training set increasing while the validation accuracy decreases, indicating overfitting. With the final layer in the network having a softmax activation, the model outputs prediction similar to what is shown below, where the first number is the likelihood of no finding and the second number corresponds to the presence of a disease.



20% of the data was reserved for testing and the final prediction accuracy of the model on the test data was 67%.

Further Study

A predictive accuracy of 67% is a good start for prioritizing the highest disease likelihood X-rays, but improvements can certainly be made. Some of the things that can be investigated in further research include:

- Additional preprocessing strategies. The equalization did not seem to significantly improve accuracy but other strategies such as contrast or brightness adjustments could be explored

- Train the model with a mix of image types. Include some rotated or randomly preprocessed images to increase model robustness
- Include additional dense layers to the network. The network used was limited by processor capabilities, but more complex architectures can be investigated using better GPUs
- Include regularization parameters to avoid overfitting the network