Diagnosing Pneumonia from Chest X-Rays

Valerie Liu, Cherise McMahon, Tianyi Wen, Huiruo Zou

November 2019

Abstract

We apply deep learning methods to chest x-ray images to diagnose pneumonia. The goal is to provide an accurate tool for medical professionals to be able to classify pneumonia within patients expressing symptoms given a chest x-ray. Ideally, this tool will also be able to differentiate between bacterial and viral pneumonia, thereby increasing the diagnosis efficiency and aiding in treatment suggestions. The results of our project suggest that we are moderately successful in predicting pneumonia via chest x-rays.

1 Introduction

Pneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus (purulent material), causing someone to have a fever, a harsh cough, chills, and difficulty breathing.

A quarter of a million people contract pneumonia every year, and a fifth of them will die due to it. The graph from National Center for Disease Control reveals the age-adjusted death rate of pneumonia and influenza by year, this trend needs to be payed attention to as it doesn't seem to be going away significantly anytime soon. Hopefully, our project has the potential to aid in the diagnosing process to increase correct treatment and decrease deaths.

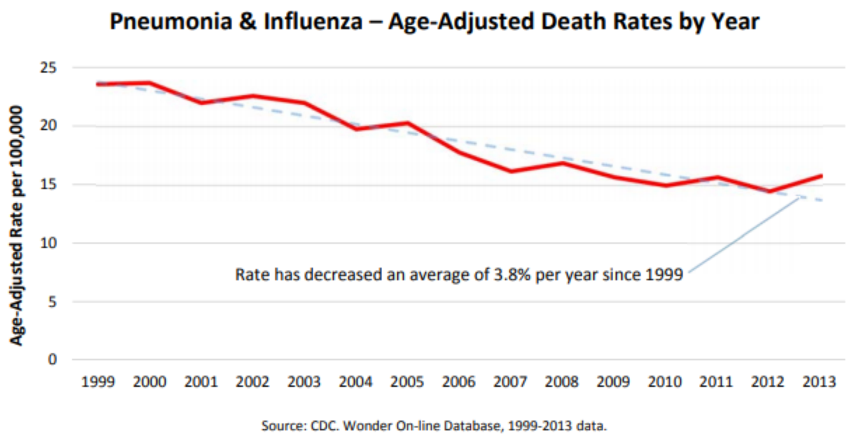

Figure 1: Influenza, pneumonia-related deaths [3]

2 Pre-Processing the Data

Our data set [1] consisted of 5,863 images between training, testing, and validating data. However, in order to properly process the images, they needed to have consistent dimensions across the board. All of the images were originally color encoded, resulting in them having a third dimension, which we kept in order to have more analysis options. The images contained the label for their strain of pneumonia in their filenames which allowed us to assign labels while loading the images.

Originally, the images could range anywhere from approximately 300x300 to 1500x2000 pixels. We decided that we didn't want to stretch any of the images or have images with too few pixels. We decided on resizing all images that were greater than 256x256 pixels to be 256x256; this would allow for no image stretching.

Once the images were resized and labeled, they would be turned into a *numpy* array and appended to the rest of the list of processed images. The final list would be loaded into an .npz file so the data could be read and manipulated with different analysis methods. This process was applied to the train, test, and validation images.

3 Base Line and Analysis Methods

Base Line To best understand the quality of learning on the data set, we wanted to compare the accuracy of the different methods to both each other and a baseline value. We had a significant amount of data and therefore did a very basic linear classifier. The test error rate of the base line is **0.465** with the classifier. We originally attempted classifying the images to provide a layperson general error, but the x-ray images were very difficult to differentiate between. It was like we were just guessing, so the human error baseline is **0.66** given that there are three possible labels.

Overview of Methods Given that we had train, test, and validation data, we were able to choose from a wide range of analysis methods. We decided on three different processes; we used Logistic Regression and the VGG16 and modified LeNet neural networks.

Logistic Regression Our method for Logistic Regression was utilizing am import from *sklean* called *LogisticRegressionCV*. We loaded the train and test files and separated the images and the labels for each set. The original image shapes were (m, 256, 256, 3), where m is their unique images in each set. We reshaped each image set to be (m, 256*256*3) in order to process the data. Once reshaped, we normalized both training and testing image data by dividing the sets by their respective maximum value. The training data set (both labels and images) was shuffled and the top 1000 values were selected to be processed. It applied three fold cross-validation and minimized loss with Stochastic Average Gradient decent. For L2 Regularization, the optimal alpha was found to be 1000. Once the model was created, we fit it to the training data set and then used it to predict the labels for both the training and testing images.

LeNet and our Convolution Model First, we wanted to directly use LeNet and check its performance. Since it did not work very well, we wanted to try building our own model. We tried different combinations of Conv, Maxpooling, and Dense layers and extracted the best model. Our final model ended up with 13 layers: the first two layers are Conv layers with 20 nodes, followed by a MaxPooling layer of 20 nodes. The next two sets of two layers are Conv and MaxPooling layers, which all have 50 nodes.The eighth and ninth layers are Conv layers with 10 nodes, followed by a MaxPooling layer with 10 nodes. The last three layers are flatten and two Dense layers.

VGG16 We did not have enough images to train a VGG16 network from scratch, so we used Transfer Learning with pre-trained weights ’image-net’ by utilizing a model imported from Keras VGG16. We connected the VGG16 layer to 3 Dense layers with ReLU activation function. These Dense layers have 1024 nodes each. They are then connected to another Dense layer with Softmax activation function. Before going through the Dense layers, we flatten the data into a 2-d array. The model was compiled with learning rate 0.01, batch size 32 and sparse categorical cross entropy as loss function. We also utilized Early Stopping in this model to stop the model from over-fitting. Everything else was left as Keras default values.

4 Results

Logistic Regression Using Logistic Regression with L2 Regularization and Cross Validation resulted in a train error rate of **0.121** and a test error of **0.338**. Since it ran with only 1000 images, we ran it again with the entire data set. The training error was **0.153** and the test error was **0.33**. This is better than our baselines by a bit, but the lack of difference between the two runs suggested we were underfitting and needed to move forward.

LeNet and our Convolution Model First, we tried to use the LeNet layers to train out data set. Each epoch trained on 4172 samples and validated on 1044 samples. The total trainable parameters were 102,428,573. After training on 50 epochs with 0.001 learning rate, our model’s result had a train error of **0.186** with a test error of **0.378**. The result was not good enough. The better loss we got from the training set, the worse test error we got on the test set. Because of this, we thought that we were facing an over-fitting issue.

We then tried to revise our model and use Earlystopping to save time, so now our model had 7 layers. The first was a Conv layer with 20 node, the second was a MaxPooling layer, the following one was another Conv layer with 50 nodes, maxpooling again, and the last two were fully-connected layers. With the new model, we reduced the errors to have a train error of

0.114 and a test error of **0.30**. Still wanting to improve, we built another model. Finally the model with 13 layers works best. The first several layers were convolutions and maxpooling, concatenating with 2 dense layers. The total trainable parameters were 1,396,163, which is largely reduced from the numbers from LeNet. We got train error of **0.199** and a test error of **0.255**

VGG16 We trained the model on 4172 samples and validated on 1044 samples. The model was set to run 100 epochs, but it was terminated by early stopping after 9 epochs as the loss function did not improve for 3 continuous epochs. After running 9 epochs with 0.01 learning rate, we had training error **0.123** and test error **0.236**. This result is much better than the baseline, and by far the model with best results.

5 Conclusion

Based on all of the results, our newly trained convolution model has a good result, but VGG16 has the best performance and lowest test error rate. We believe that this is caused by Keras VGG16 being pre-trained; it may adapt to the features of the chest x-ray images using previous weights better than the model we trained from scratch. As for Logistic Regression, it is adapted from a linear classifier with a softmax layer and therefore may not fit the data as well as the other models. It requires the most improvement.

By processing the data and analyzing the error rate of our analysis methods, our project seems that it can be useful in real life. Our goal is to help medical professionals save time in the diagnosing process in order to help more patients, which improves efficiency of illness treatment.

We were discouraged with a 76.4% accuracy, but then we did some research on how accurately doctors predict pneumonia with chest x-rays. In one study, only 57% of clinically diagnosed pneumonia cases were able to be confirmed with chest x-rays [2]. If this is typical, then our tool would be able to strongly help doctors in their diagnosing process; at least for confirmation on what they believe. However, it should be considered that other illnesses which cause respiratory inflammation could be registered as pneumonia given that our training data does not account for that. Nonetheless, there is a lot of potential!

References

[1] Chest X-Ray Images. `https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`, 2017. Accessed 11/8/2019.

[2] The diagnosis of pneumonia requires a chest radiograph (x-ray)—yes, no or sometimes? `https://pneumonia.biomedcentral.com/articles/10.15172/pneu.2014.5/464`, Dec 2014.

[3] Trends in Pneumonia and Influenza Morbidity and Mortality. `https://www.lung.org/assets/documents/research/pi-trend-report.pdf`, Nov 2015. Accessed 11/8/2019.