DETECTING PNEUMONIA IN X-RAYS USING NEURAL NETWORKS

Mark Martir-Irizarry, Garima Singh, Erika Andrade, Tony Tran

University of Houston

UHSPE Machine Learning BootCamp 2020

**Problem Introduction**

Determining whether a patient has pneumonia using Neural Networks can have significant advantages. For starters, there are multiple tests needed to detect pneumonia such as listening to lungs with a stethoscope, chest X-rays, blood tests, etc. If it can be identified with an X-ray, it could potentially decrease the number of tests that have to be run. Therefore, tests can be run faster, and potentially be cheaper.

The goal of the analysis is to identify the differences between a patient with pneumonia and ones without using neural networks. We will train our code using 1341 pictures of X-rays that have been identified as normal and 3875 pictures of patients who have pneumonia and use 624 pictures of patients to test it. We will adjust the number of nodes needed to have the most accurate identifier of pneumonia possible. The goals of this project will be:

(1) Identify whether a patient has pneumonia using old X-ray image data and neural networks.

(2) Evaluate the effectiveness and accuracy of neural networks for this problem

(3) Find out which neural network combination will have the most accurate identifier of pneumonia

(4) Show whether X-rays alone are reliable enough to identify patients with pneumonia

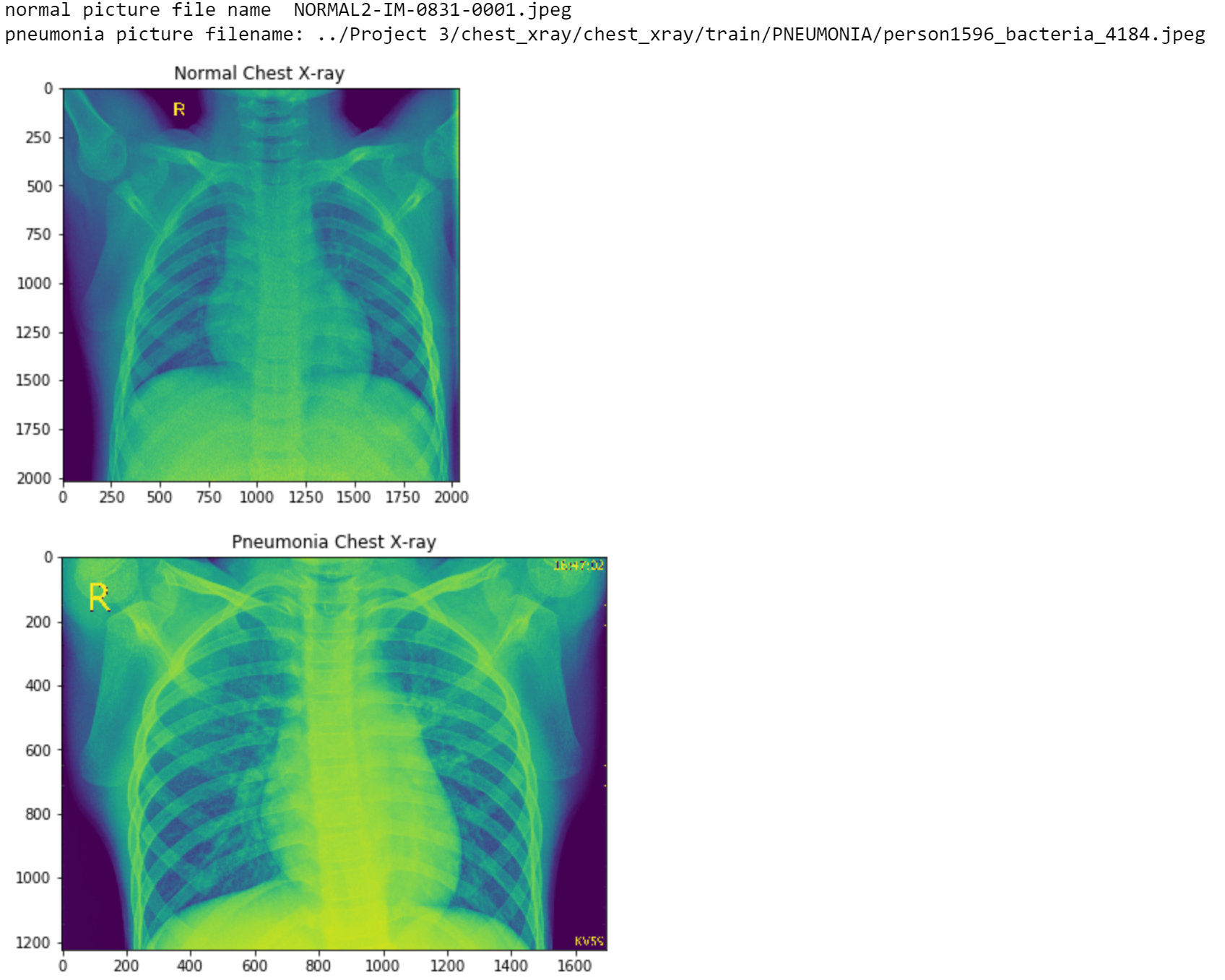
# Dataset

The dataset used for this project was obtained through the Kaggle repository under the data section. The dataset contains a total of 5856 image samples categorized into testing, training, and validation folders. Within these folders, they are categorized based on whether the image is of a patient that had pneumonia.

# Features and Processing

In this case, features are only based on the chest x-ray images, therefore we could consider each pixel in the image matrix individual features.

We imported the folders individually, giving them names based on their location. In order to understand the data and see if the data was successfully imported, we visualized the data. Due to the sheer number of images in the folders, we could not view the images individually, since we can only process so many images in our code. Therefore, we decided to view the images based on a randomized integer value. We looked for a random picture inside the training folder from both the pneumonia and normal category and presented them in the console, as well as its filename, shown in ***Figure 1***. This way we can compare each of the images themselves and know the naming conventions of the images.



***Figure 1*: Random image inside the train folder of both pneumonia and normal X-rays, and their image name.**

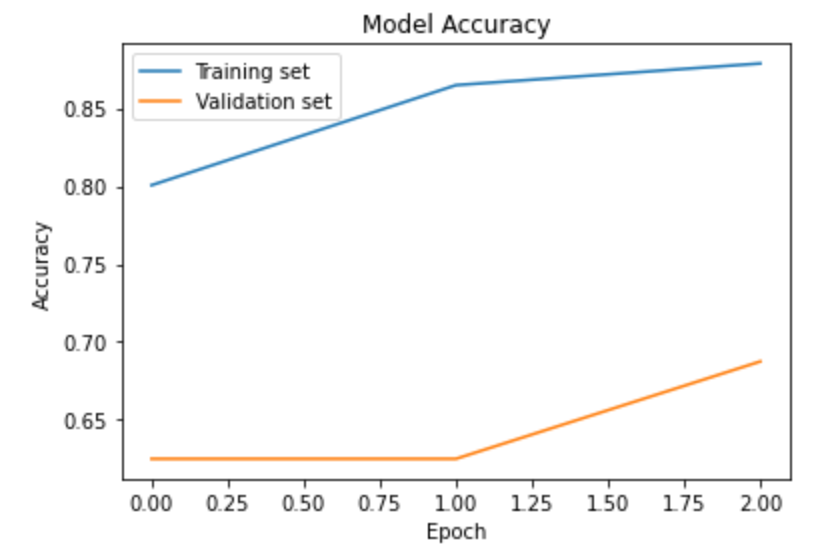
# Models and Techniques

The dataset was already split into training set and testing set. We only used neural networks for this project. Specifically, we used CNN, convolutional neural network. This method, instead of having weights for each pixel, CNN only has enough weights to look at a small part of an image. We also used pyplot to plot the model’s history in order to check and analyze which epoch values give the best accuracy and model loss. Therefore giving us the most optimal solution.

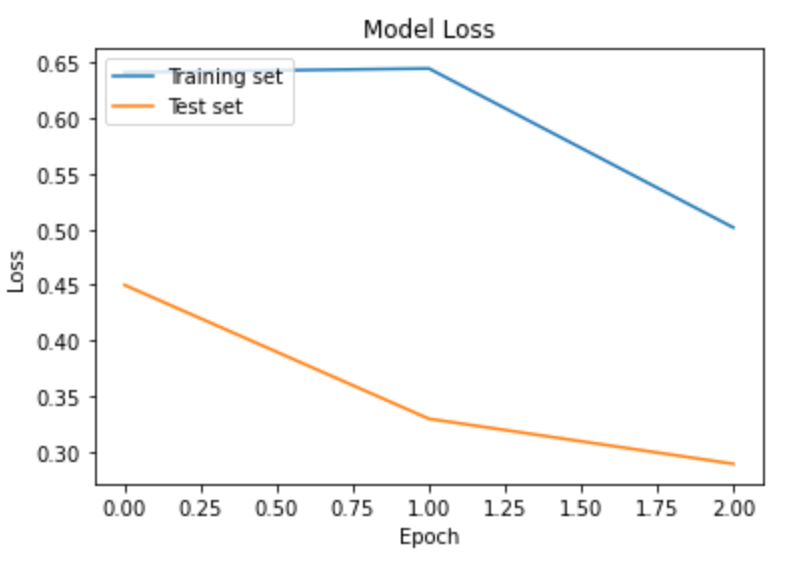
# Results and Discussion

We decided to go with 2 convolution layers on a trial-and-error basis. As we know, convolution layers are the building blocks in convolutional neural networks. Since not every single pixel is not needed to be accurate, we want our program to run efficiently and smooth. Convolution and pooling layers do just that. Our images are too big; therefore, we only want to identify using a small batch. We determined that 2 convolution layers was best because there was less loss of precision. The model accuracy was 80% with 3 convolution layers while it was 86% with 2 convolution layers.

We decided to normalize the data to make the pixels more distinct therefore more visible and train the CNN model based on that. Our CNN model went through a lot of changes based on trial and error as well. In ***Figure 2*** and ***Figure 3*** below, you can see that we changed the epoch values and got these results. The following graphs show the results of the model accuracy and model loss based on epoch values 0,1, and 2. Our steps per epoch was 200 and our amount of convolution layers was 2 as previously discussed. A value of 3 epoch was decided which gave us 86%, but, as can be seen, the other epoch values gave poor results.



***Figure 2*: Graph of Model Accuracy vs Epoch Values.**



***Figure 3*: Graph of Model Loss vs Epoch Values.**

Later, we decided to check different parameters. In ***Table 1*** below, convolutions, epochs, epoch steps, and time per epoch values are being tested. The values in row 1 were used as standard values to compare to others. We initially tested the epoch values by increasing them significantly, and our model accuracy increased. This makes sense since multiple passes through the training data should yield better results. Then we tested 3 convolutions shown in rows 3 and 4 with different epoch values. Since there is a loss of data with the third convolution, we can see both do not yield better results than the 2 convolutions. Then the steps were tested with different epoch values in rows 5 and 6. This one is random, since the smaller the steps the less datapoints are checked with each run. These values yielded our most accurate model at 89.282% model accuracy. Finally, we did some random testing across the board, and yielded a maximum 89.082% model accuracy value, shown in row 11.

***Table 1*: Convolution, epoch steps, and time per epoch relationship to model accuracy.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Rows | Convolutions | Steps | epoch | time/epoch | percentage |  |  |
| 1 | 2 | 200 | 1 | 207 | 84.624% |  | Standard/ control |
| 2 | 2 | 200 | 5 | 207 | 88.964% |  | testing epoch |
| 3 | 3 | 200 | 1 | 207 | 76.120% |  | Testing convolution |
| 4 | 3 | 200 | 5 | 207 | 87.336% |  |  |
| 5 | 2 | 150 | 1 | 183 | 82.518% |  | Testing Steps |
| 6 | 2 | 150 | 5 | 183 | 89.282% |  |  |
| 7 | 2 | 250 | 1 | 229 | 84.922% |  | Random testing |
| 8 | 2 | 250 | 5 | 229 | 85.918% |  |  |
| 9 | 2 | 250 | 3 | 229 | 79.006% |  |  |
| 10 | 2 | 200 | 3 | 207 | 86.072% |  |  |
| 11 | 3 | 250 | 5 | 220 | 89.082% |  |  |
| 12 | 3 | 250 | 3 | 220 | 86.874% |  |  |

# Future Work

In the future, we should:

1. Use the equation train\_length / batch\_size for our epoch steps and test it out. This equation should change it such that it tests for all the datapoints.
2. Do some data manipulation, this could include thresholding, filtering, and masking, to make the training data much easier for our model.
3. Test more epoch values to attempt to increase accuracy.

References

Last Name, F. M. (Year). Article Title. *Journal Title*, Pages From - To.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.

Footnotes

1[Add footnotes, if any, on their own page following references. For APA formatting requirements, it’s easy to just type your own footnote references and notes. To format a footnote reference, select the number and then, on the Home tab, in the Styles gallery, click Footnote Reference. The body of a footnote, such as this example, uses the Normal text style. (Note: If you delete this sample footnote, don’t forget to delete its in-text reference as well. That’s at the end of the sample Heading 2 paragraph on the first page of body content in this template.)]

Tables

Table 1

[Table Title]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Head | Column Head | Column Head | Column Head | Column Head |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |

Note: [Place all tables for your paper in a tables section, following references (and, if applicable, footnotes). Start a new page for each table, include a table number and table title for each, as shown on this page. All explanatory text appears in a table note that follows the table, such as this one. Use the Table/Figure style, available on the Home tab, in the Styles gallery, to get the spacing between table and note. Tables in APA format can use single or 1.5 line spacing. Include a heading for every row and column, even if the content seems obvious. A default table style has been setup for this template that fits APA guidelines. To insert a table, on the Insert tab, click Table.]

Figures title:

Figure 1. [Include all figures in their own section, following references (and footnotes and tables, if applicable). Include a numbered caption for each figure. Use the Table/Figure style for easy spacing between figure and caption.]

For more information about all elements of APA formatting, please consult the APA Style Manual, 6th Edition.