Adam Villarreal
January 20, 2020

Capstone Report

Classifying Pneumonia

Definition:

Project Overview

Building an optimal Convolutional Neural Network, or rather an AI System, that is able to learn from images of X-ray images that contain pneumonia and images that do not. While it learns, it finds patterns within the images on its own. After learning, it will be able to guess if an image contains pneumonia or not.

Problem Statement

According to the American Lung Association (ALA), pneumonia is the cause of death of 50,000 people each year and more than a million hospitalizations per year in the United States alone. One new popular Deep Learning technique called, Convolutional Neural Networks (CNNs), are being used to solve classification problems. Convolutional Neural Networks are known to be advancing Computer Vision, which is the possibility for a computer to be able to read and learn from images. There are datasets that contain thousands of X-ray images that contain pneumonia and those who do not have pneumonia. We can use Convolutional Neural Networks to solve this problem. We can create different Deep Learning models that use CNNs to learn from

the images for the task to predict images that it did not learn from so that it may predict images at high efficiency.

The problem will be solved by these steps:

1. Explore the dataset
2. Define different CNNs with different architectures
3. Train the CNNs
4. Test the CNNs
5. Evaluate which CNN performed the best with defined metrics

Metrics

The metrics that will be used are by measuring these:

1. True Positives(TP) - The number of correctly identified images with pneumonia.
2. False Positives(FP) - The number of images without pneumonia falsely identified with having pneumonia.
3. True Negatives(TN) - The number of correctly identified images that do not have pneumonia.
4. False Negatives(FN) - The number of images with pneumonia falsely identified with not having pneumonia.

We want to decrease FP and FN as much as possible. Especially FN, because if a patient that has pneumonia is classified as if not having pneumonia, that is a big problem!

Using those measures we can compute these ratios:

1. Sensitivity (or Recall) - The percentage of correctly classified images with pneumonia of all images of pneumonia.

$$\frac{TP}{TP + FN}$$

2. Specificity - The percentage of correctly classified images without pneumonia of all images that do not contain pneumonia.

$$\frac{TN}{TN + FP}$$

3. False Positive Rate (FPR)

$$1 - Specificity = \frac{FP}{TN+FP}$$

Just as above, we would like to decrease FP and FN as much as possible. We should want to maximize Sensitivity as much as possible. This is because we want to maximize the number of images that have pneumonia to be correctly classified.

Using Sensitivity and FPR, we can use the AUC metric. The Reciever Operating Characteristics (ROC) measures the performance of the model on different thresholds and the AUC (Area Under the Curve) accesses that performance. You can watch this video here to learn more. Or you can read this here.

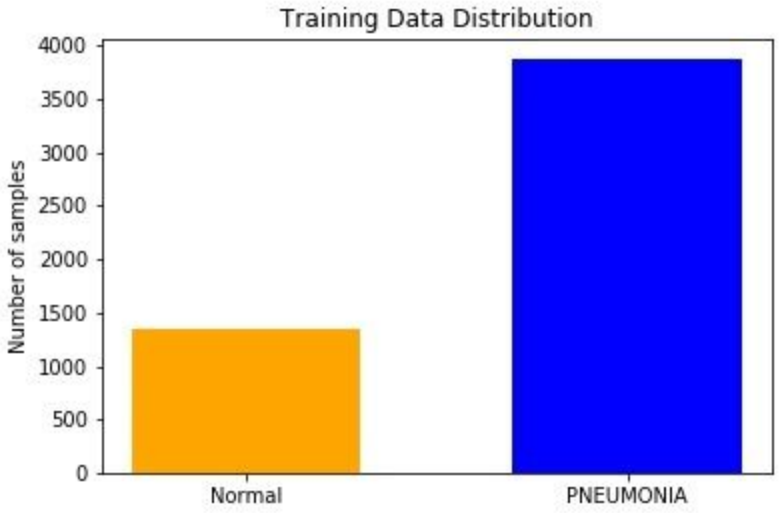
For a quick guideline of these metrics please refer to this Lecture Slideshow by James D. Wilson Ph.D. from the University of San Francisco here.

Analysis:

Data Exploration and Visualization

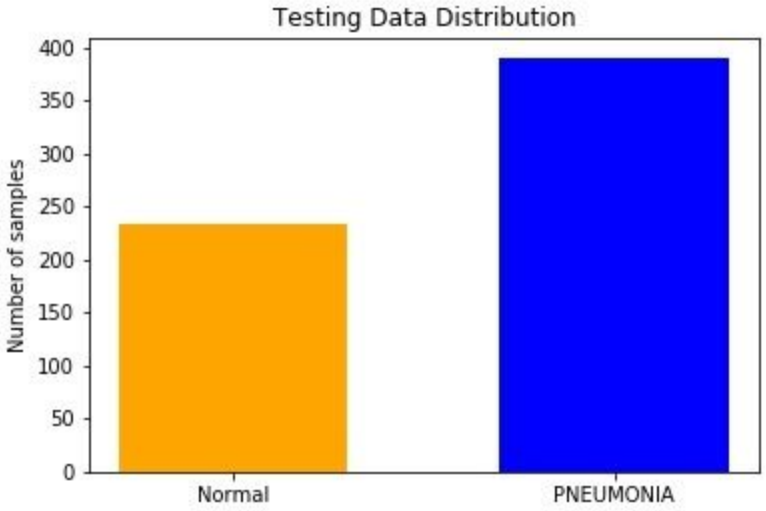
The dataset contains thousands of pictures of images that contain pneumonia and images that do not contain pneumonia. The training set contains a total of 5,216 images, of which 1,341 of them are normal images and 3,875 are images with pneumonia. The testing set contains a total of 624 images, of which 234 of them are normal images and 390 are images with pneumonia. The validation set contains a total of 16 images, of which 8 of them are normal images and 8 are images with pneumonia. That makes a total dataset of 5,856 images.

Here is the distribution of the training set:



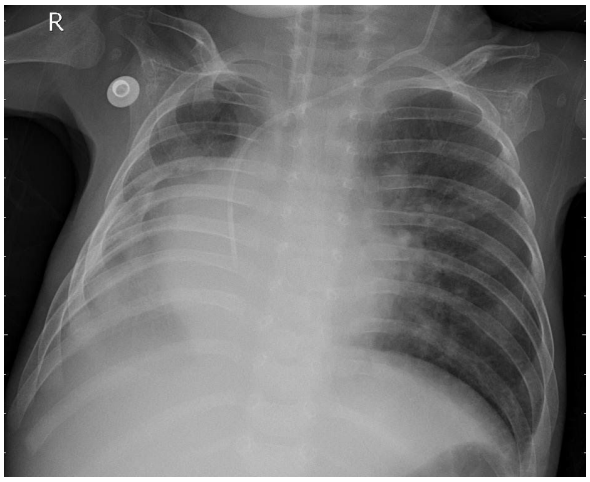
As we see here the training data is very unbalanced.

Here is the distribution of the testing set:



As we see here the testing data is very unbalanced.

Here is an example of an image that contains pneumonia:



Another issue for data to be used in a Convolutional Neural Network is to ensure that all images have the same dimension and color channels (Greyscale or RGB). As we see above, we would predict that most of the images are greyscale images, however, 283 images were RGB images. Also, the images did not contain the same

dimensions. To solve this problem, I converted all the images to be 224x224 and to have the RGB color-channel, therefore the final dimension of every image should be 224x224x3.

Algorithms and Techniques

The algorithm that will be used is a Convolutional Neural Network, which is one of the best tools for image classification based on recent research. The task of the algorithm will be to classify images of having pneumonia or not having pneumonia. This is technically a Binary Classifier since there are only two classes to choose from. Therefore, the output should either be a probability distribution of each class or a direct output that provides the class.

Parameters of the Convolutional Neural Network:

- Training Parameters
    - Epochs - The number of times that the model would train on the training data
    - Batch Size - The number of images that the model will train on in a single step
    - Learning Rate - The magnitude of the step for the model to converge to the lowest error it can find
- Architecture
    - Number of Layers
    - Types of Layers (Convolutional, Pooling, Fully-Connected)
    - Read here to understand the types of layers

Since converting all images to RGB and to have the dimensions of 224x224x3, all the models that I will build will take an input of that dimension. The models will be made with AWS Sagemaker Built-In Image Classification Algorithm and with PyTorch.

Benchmark

The benchmark model will be AWS Image Classification Algorithm because it will be best suited to use the HyperparameterTuner to create multiple models by modifying these hyperparameters. The tuning job will train and provide multiple models, and it will use the validation set to select the best model.

Methodology:

Data Preprocessing

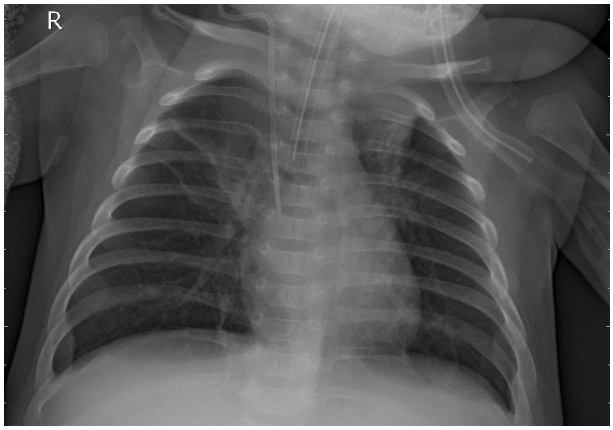
These are the issues with the unclean dataset:

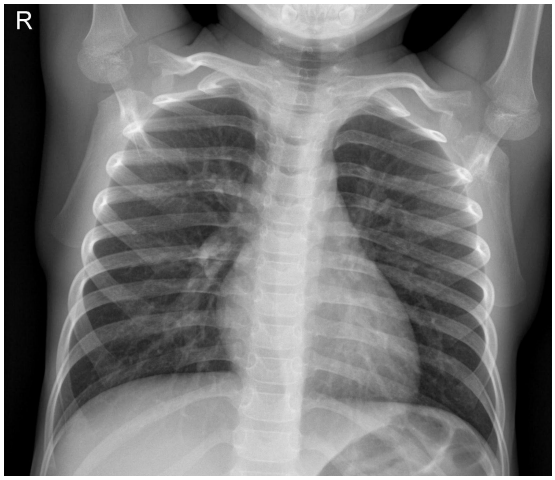
1. The data does not have the same color-channel (Greyscale or RGB)
2. The data do not have universal dimensions

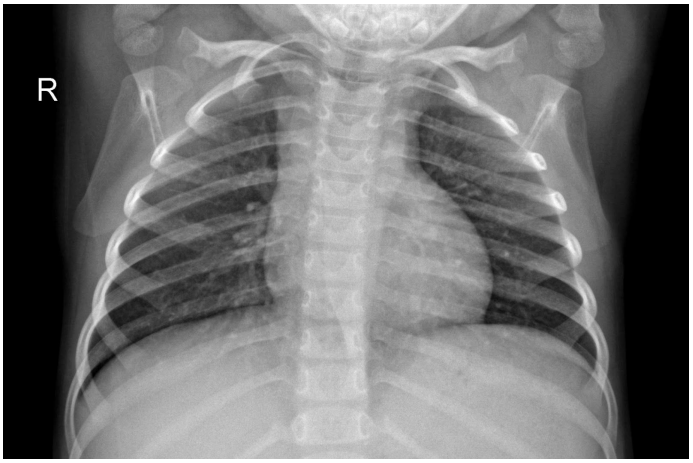
Steps I took to clean the dataset:

1. I converted all images to have the RGB color-channel so that the CNN can expect a depth of 3 for all images
2. I resized all images to have dimensions of 224x224x3

Here are examples of the unclean data:

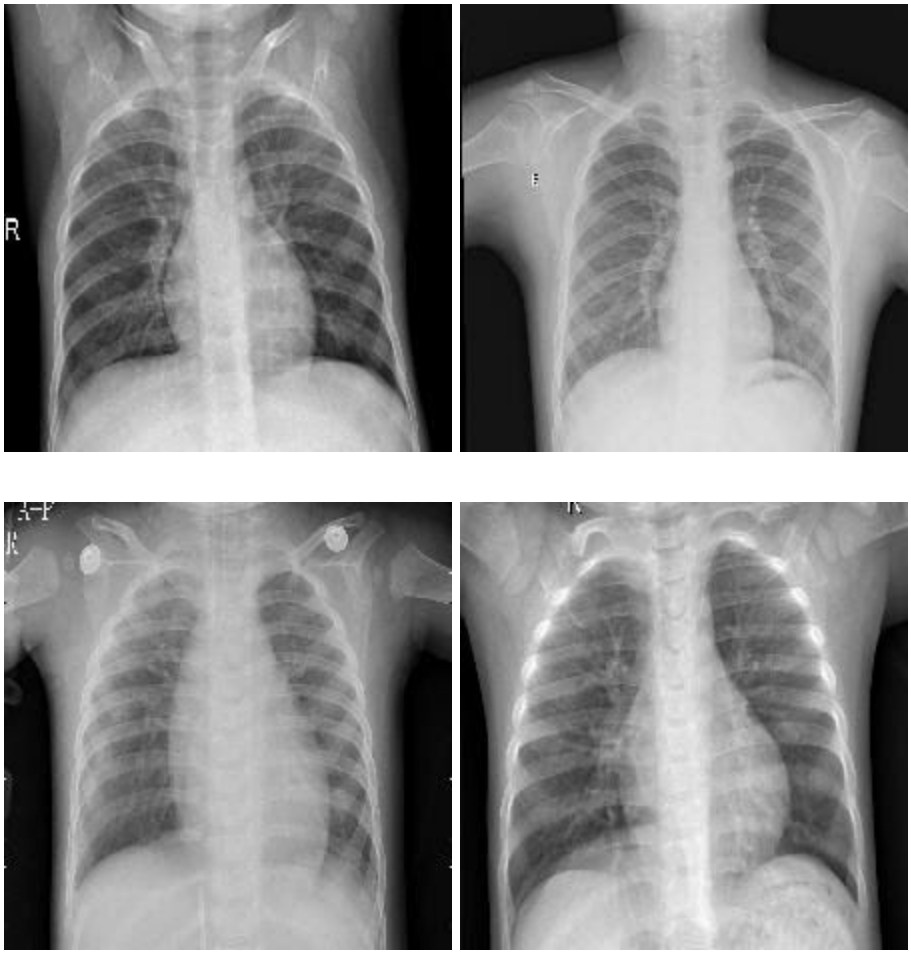






These images do not have the same height and length dimensions.

Here are examples of the cleaned data:



As we see, the clean images are now in uniform to be both RGB and to have the dimensions of 224x224x3.

Implementation

After the dataset was cleaned and explored, it was uploaded to the AWS S3 storage platform which will be used to train the models. All models were trained, validated, and tested on the same dataset using AWS SageMaker. Below is more information about each of the models.

AWS Model:

- Uses ResNet-152 Architecture
- Used AWS SageMaker HyperparameterTuner tool to find the best model by tweaking these parameters
    - Learning Rate
    - Optimizer (SGD, ADAM, RMSPROP, NAG)
- The metric for selecting the best model from the HyperparameterTuner was the minimization of the validation loss
- Deploy the best model
- Test the model using the cleaned images saved on the local machine
- Calculate the metrics stated from ‘Metrics’ section from above

These are the other PyTorch models:

1. My own Convolutional Neural Network
    a. There are three Convolutional and Pooling Layers
    b. The optimizer uses Stochastic Gradient Descent
    c. One node output that is the probability of the image having pneumonia
        i. The output is rounded to be either 0 or 1, where 1 symbolizes that the image contains pneumonia
    d. When the training job is called, it trains, validates, and tests the models all within AWS Sagemaker. While testing the PyTorch models, the metrics from the ‘Metrics’ section from above were calculated. The calculated metrics can be seen in the training job’s logs

2. Retrained VGG-16
    a. 16 Weight Layers
    b. The optimizer uses Stochastic Gradient Descent
    c. Reconfigured to also have one output as above
    d. It also was trained, validated and tested as the model from above.
    e. The metrics were also calculated using the same method as the other method above

Refinement

AWS Model - There was no improvement for the AWS model since it was able to tune itself using the HyperparameterTuner tool.

My own CNN - I trained 5 separate models using learning rates of 0.1, 0.05, 0.01 and two other random learning rates.

VGG-16 - I trained 5 separate models using the same method as above.

After training the models, I compared the calculated metrics especially the ROC-AUC score.

Results:

Model Evaluation and Validation

After training and testing, I selected the best model for each of the three methods.

1. AWS Model
    a. Learning Rate: 0.05561869868809854
    b. Epochs: 10
    c. Optimizer: adam
    d. Metrics
        i. True Positives: 379
        ii. False Positives: 81
        iii. True Negatives: 153
        iv. False Negatives: 11
        v. AUC Score: 0.8128205128205128
2. My CNN Model
    a. Learning Rate: 0.045
    b. Epochs: 20
    c. Metrics
        i. True Positives: 370
        ii. False Positives: 73
        iii. True Negatives: 161
        iv. False Negatives: 20
        v. AUC Score: 0.8183760683760684

3. VGG-16 Model
    a. Learning Rate: 0.05
    b. Epochs: 20
    c. Metrics
        i. True Positives: 389
        ii. False Positives: 128
        iii. True Negatives: 106
        iv. False Negatives: 1
        v. AUC Score: 0.7252136752136752

After comparing these results, the AUC score of the VGG-16 had a noticeable difference compared to the other models, therefore it is not the best model. One interesting finding is that the number of False Negatives for the VGG-16 model is very small compared to the other models, but the amount of False Positives is not optimal since it is failing to predict most of the images that do not contain pneumonia. The AUC scores between the AWS model and my CNN model is very small, but the AWS was able to receive a lower number of False Negatives. Therefore, it is best to choose the AWS model.

Justification

It is justified to choose the AWS model because it is less likely for someone with pneumonia to be diagnosed as not having pneumonia. The AWS does have more False Positives than the CNN model, however it is not as dangerous as having a higher False Negative count. It is not as dangerous because if a healthy person was incorrectly

diagnosed with having pneumonia, it is not as catastrophic as not receiving medical care when the patient has pneumonia.