# Death Rate Correlation with COVID-19 X-Ray Images

## Overview

### Team Members

1. Harshal Sharma

   I'm currently pursuing a BS in CS at the Texas Tech University. (https://www.linkedin.com/in/harshalsharma08)

2. Shreyash Shrivastava

   I am an incoming Analyst at Bank of America in Plano, TX. I have completed a BSCS from UT Arlington. (https://www.linkedin.com/in/shreyash-shrivastava/)

### Project Development libraries

Numpy, Pandas, Matplotlib, Jupyter notebook, Seaborn, PIL, Keras, OpenCV, Scikit-learn

### Dataset Source

1. https://www.kaggle.com/bachrr/covid-chest-xray/data
2. https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

### Problem Statement

To find a correlation between chest x-ray images of COVID-19 infected patients and their subsequent mortality.

## Procedureand Efficacy

### Dataset Explanation:

Dataset1

We have labeled dataset of COVID-19 infected patient's chest X-ray depicting the mortality label for each image. In the same set, we also have unlabeled images of chest X-rays as well.

Dataset2

We have three sets of folders, namely; Viral Pneumonia, COVID-19, and Normal. The three folders consist of chest X-rays of people with the said conditions.

### CNN (Convolution Neural Network):

1. Clean and preprocess the raw data using Keras preprocessing library. Separate the data with missing values. Then, create a CNN model as per the described architecture.
2. Use the clean data to train the model with specified hyperparameters.
3. The model calculates the chance of survival from the input images and classifies it into two classes: *WILL_SURVIVE, WILL_NOT_SURVIVE*
4. Use out of setting data (Dataset 2) which consists of three categories: *COVID-19, NORMAL, VIRAL PNEUMONIA*
5. Clean and preprocess all three categorical data, and use the trained model to classify it. Then, based on the classification calculate the death rate and survival rate.
6. (Next steps) Use GAN to generate artificial images and use that to again train the model and evaluate it.

### Constrains

Limited Data available
Limited Computational power
Time constraint

## Results

The CNN model performed extremely well on online datasets for COVID-19 patients. The results were astonishingly close to the real world cases despite the computational and data scarcity constraints we possessed.

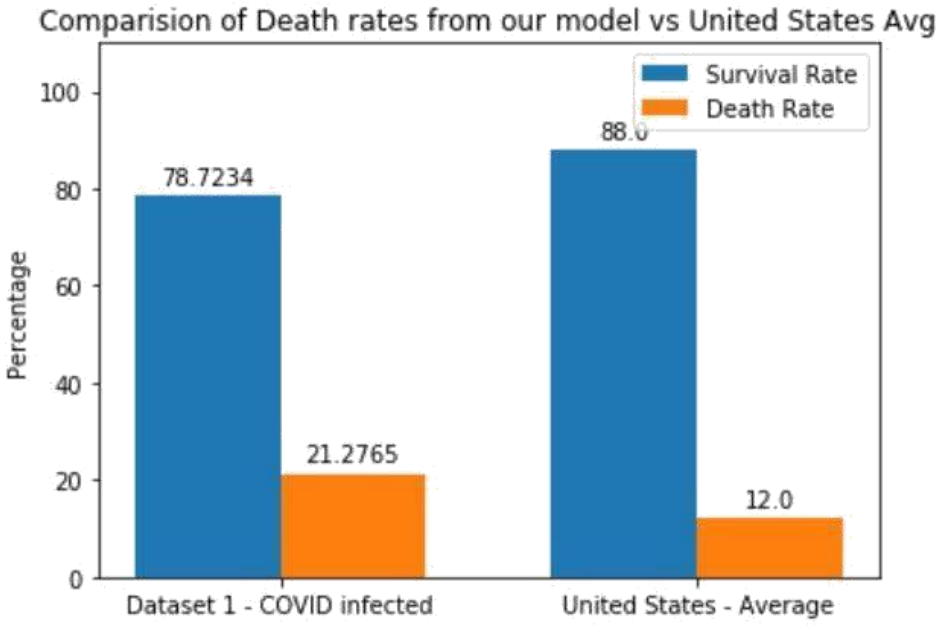



**Figure 1 -** Comparison of Death rate prediction of our model vs the United States closed cases

**NOTE:**

The definition of the real-world death rate should be pondered on. The death rate should be calculated on the number of closed cases. This implies that the on-going cases of COVID-19 infection should be deliberately excluded from the calculation. The closed cases signify two possibilities, either the infection recovers or its consequences into a fatality.

Therefore, the death rate is calculated as (Total Deaths) /(Total number of people recovered + Total Deaths)

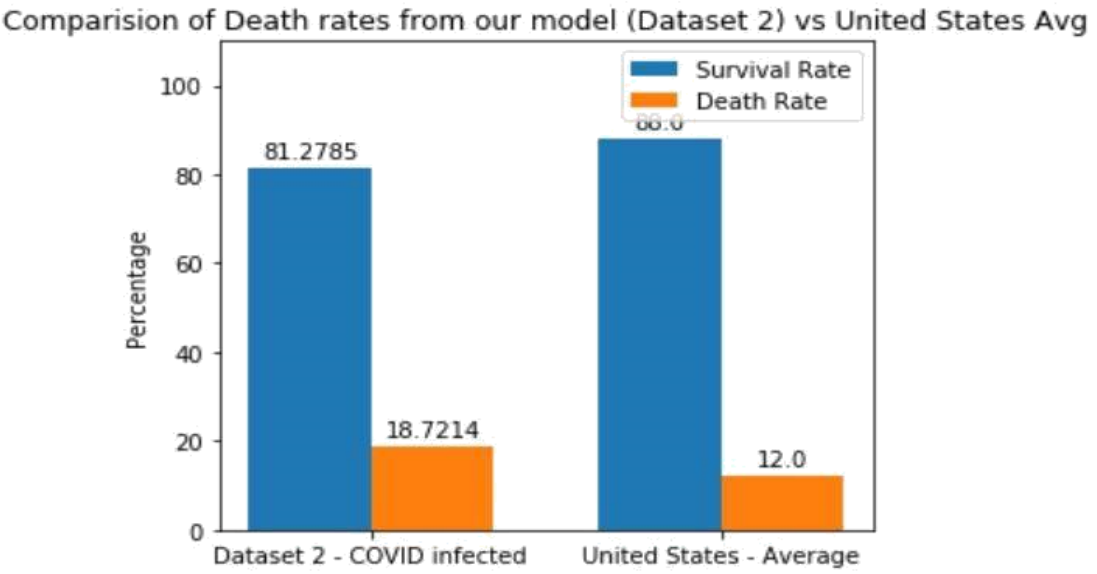



**Figure 2 -** Comparison of Death rate prediction of our model vs the United States closed cases

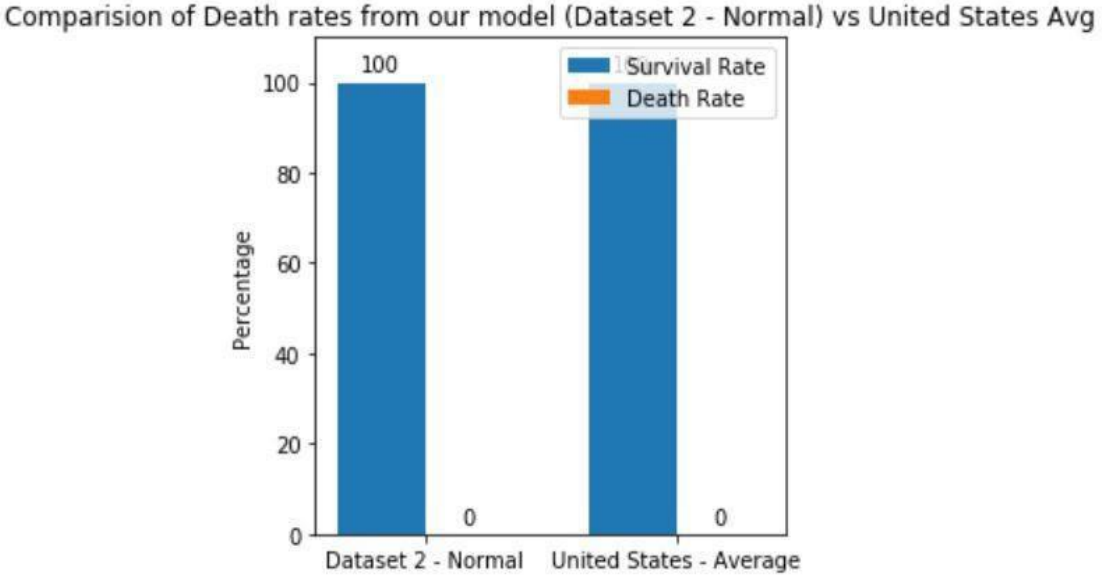



**Figure 3 -** Comparison of Death rate prediction of our model vs the United States case for ***Non-COVID-19***Infected patients

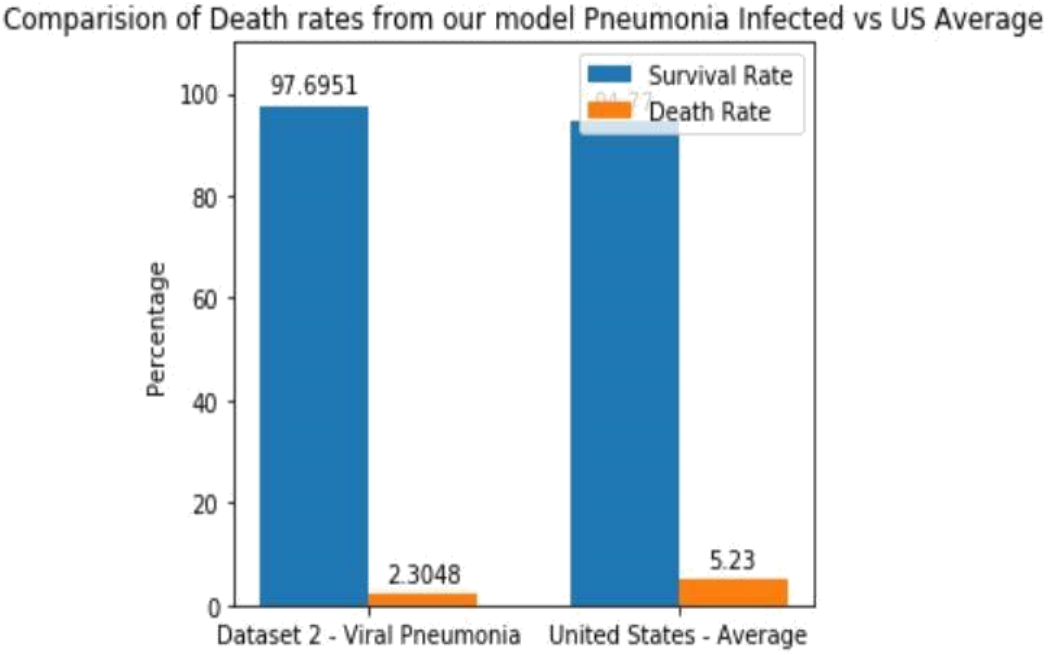



**Figure 4 -** Comparison of Death rate prediction of our model vs the United States closed cases for Viral Pneumonia Infected patients

**CNN's model summary and architecture**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_11 (Conv2D) | (None, 62, 62, 32) | 896 |
| max_pooling2d_11 (MaxPooling | (None, 31, 31, 32) | 0 |
| conv2d_12 (Conv2D) | (None, 29, 29, 32) | 9248 |
| max_pooling2d_12 (MaxPooling | (None, 14, 14, 32) | 0 |
| flatten_6 (Flatten) | (None, 6272) | 0 |
| dense_11 (Dense) | (None, 128) | 802944 |
| dense_12 (Dense) | (None, 2) | 258 |

Total params: 813,346
Trainable params: 813,346
Non-trainable params: 0

**Figure 5**- Architecture of the CNN model

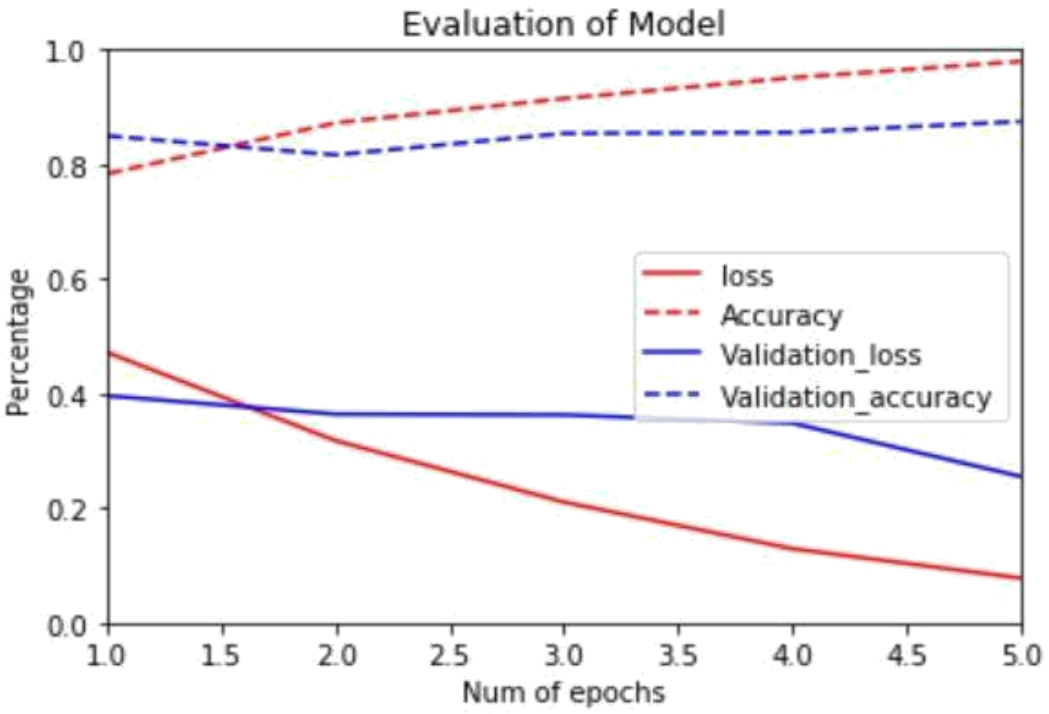



**Figure 6**- Evaluation of the CNN model

***GAN ARCHITECTURE AND SUMMARY***

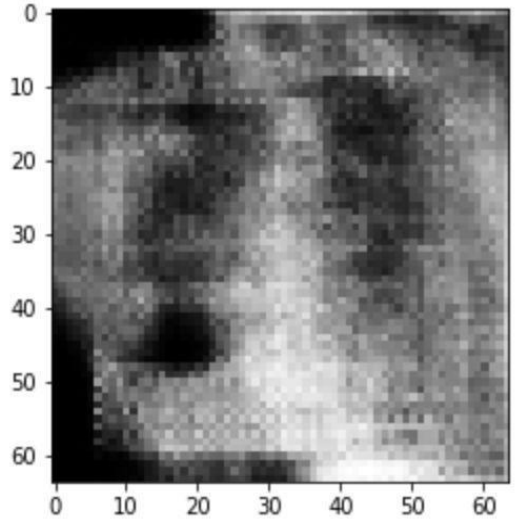
Generative Model Summary

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 32768)             3309568
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 32768)             0
_________________________________________________________________
reshape_1 (Reshape)          (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 32, 32, 128)       262272
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 64, 64, 128)       262272
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 1)         32769
=================================================================
Total params: 3,866,881
Trainable params: 3,866,881
Non-trainable params: 0
_________________________________________________________________
```

Discriminative Model Summary

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 64)        640
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 16385
=================================================================
Total params: 107,906
Trainable params: 53,953
Non-trainable params: 53,953
_________________________________________________________________
```

The following X-Ray image is artificially generated using a GAN. The GAN network was trained on a local CPU with 50 iterations. We can drastically improve the quality of generated images given computation power and time.



***Conclusion***

Our model shows enough promise to be considered significant. The real-world death rate is around 12 percent, and our model predicts 19. The model also predicts ***zero mortality***for ***normal X-ray images***.The skewness can be accounted for numerous factors, such as a biased sample size, or lack of training data. Such a close correlation between chest X-ray and mortality is an interesting endeavor to pursue and our investigation performs the task of showing noteworthy indication.

Next Steps:

The next step in our project would be to use trained and tuned GAN to further generate the training data. The generation of labeled training data would significantly increase the accuracy of our model, and we will be able to test it widely across different datasets.