Jake Prichard

Deep Learning: Chest X-ray Preprocessing Pipeline

Project Objective: Build an advanced preprocessing pipeline for chest x-ray images. Build models that can determine where the lungs are in an x-ray and filter out anatomical structures that are not needed for the discovery of various lung conditions.

Project Purpose: Having a pipeline for processing chest x-ray images and removing areas such as the heart, head, diaphragm,etc, is a necessary step towards building models that can detect the presence of various lung conditions. A pipeline of this kind could be incredibly useful for the purpose of building models that can diagnose both bacterial/viral pneumonia as well as covid-19 pneumonia.

Data set :Covid-19 radiography -database

https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

The dataset provides chest x-ray images and corresponding lung masks for patients with a variety of lung conditions, as well as normal chest x-rays. The size of each of the folders is shown below.

Normal

Size: 10192

COVID

Size: 3616

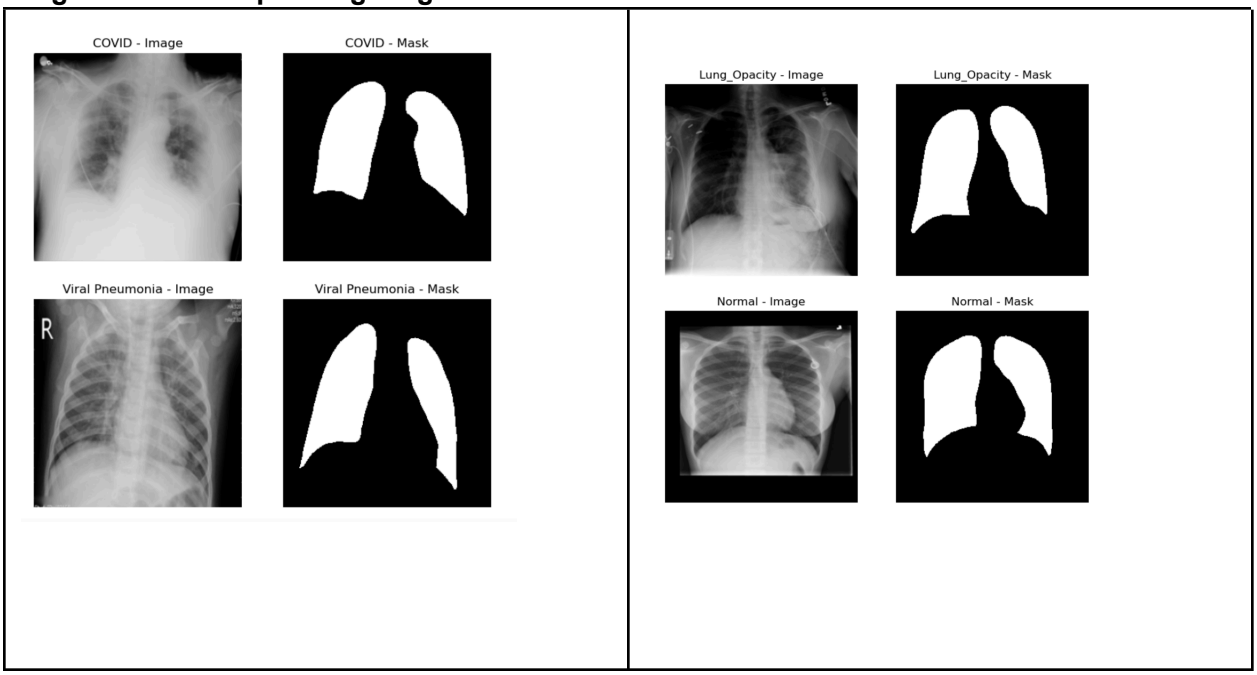
Viral Pneumonia

Size: 1345

Lung_Opacity

Size: 6012

Images with corresponding lung masks



Segmentation Problem

The first task was to build a model that could predict the lung mask for the images. This would be a classifier that predicts whether each pixel represents lung or does not represent lung. To do this I used the Unet architecture. Unet is a model architecture that is commonly used in medical imaging segmentation problems. The model architecture was first introduced in 2015 in the paper "U-Net: Convolutional Networks for Biomedical Image Segmentation". Since then, Unet has been widely used for problems that involve segmenting certain organs or lesions.

Model: "U-Net"

Layer (type)	Output Shape	Param #	Connected to
input_layer_6 (InputLayer)	(None, 256, 256, 1)	0	-
conv2d_114 (Conv2D)	(None, 256, 256, 64)	640	input_layer_6[0]…
conv2d_115 (Conv2D)	(None, 256, 256, 64)	36,928	conv2d_114[0][0]
max_pooling2d_24 (MaxPooling2D)	(None, 128, 128, 64)	0	conv2d_115[0][0]
conv2d_116 (Conv2D)	(None, 128, 128, 128)	73,856	max_pooling2d_24…
conv2d_117 (Conv2D)	(None, 128, 128, 128)	147,584	conv2d_116[0][0]
max_pooling2d_25 (MaxPooling2D)	(None, 64, 64, 128)	0	conv2d_117[0][0]
conv2d_118 (Conv2D)	(None, 64, 64, 256)	295,168	max_pooling2d_25…
conv2d_119 (Conv2D)	(None, 64, 64, 256)	590,080	conv2d_118[0][0]
max_pooling2d_26 (MaxPooling2D)	(None, 32, 32, 256)	0	conv2d_119[0][0]
conv2d_120 (Conv2D)	(None, 32, 32, 512)	1,180,160	max_pooling2d_26…

conv2d_121 (Conv2D)	(None, 32, 32, 512)	2,359,808	conv2d_120[0][0]
max_pooling2d_27 (MaxPooling2D)	(None, 16, 16, 512)	0	conv2d_121[0][0]
conv2d_122 (Conv2D)	(None, 16, 16, 1024)	4,719,616	max_pooling2d_27…
conv2d_123 (Conv2D)	(None, 16, 16, 1024)	9,438,208	conv2d_122[0][0]
conv2d_transpose_24 (Conv2DTranspose)	(None, 32, 32, 512)	2,097,664	conv2d_123[0][0]
concatenate_24 (Concatenate)	(None, 32, 32, 1024)	0	conv2d_transpose… conv2d_121[0][0]
conv2d_124 (Conv2D)	(None, 32, 32, 512)	4,719,104	concatenate_24[0…
conv2d_125 (Conv2D)	(None, 32, 32, 512)	2,359,808	conv2d_124[0][0]
conv2d_transpose_25 (Conv2DTranspose)	(None, 64, 64, 256)	524,544	conv2d_125[0][0]
concatenate_25 (Concatenate)	(None, 64, 64, 512)	0	conv2d_transpose… conv2d_119[0][0]
conv2d_126 (Conv2D)	(None, 64, 64, 256)	1,179,904	concatenate_25[0…
conv2d_127 (Conv2D)	(None, 64, 64, 256)	590,080	conv2d_126[0][0]

conv2d_transpose_26 (Conv2DTranspose)	(None, 128, 128, 128)	131,200	conv2d_127[0][0]
concatenate_26 (Concatenate)	(None, 128, 128, 256)	0	conv2d_transpose… conv2d_117[0][0]
conv2d_128 (Conv2D)	(None, 128, 128, 128)	295,040	concatenate_26[0…
conv2d_129 (Conv2D)	(None, 128, 128, 128)	147,584	conv2d_128[0][0]
conv2d_transpose_27 (Conv2DTranspose)	(None, 256, 256, 64)	32,832	conv2d_129[0][0]
concatenate_27 (Concatenate)	(None, 256, 256, 128)	0	conv2d_transpose… conv2d_115[0][0]
conv2d_130 (Conv2D)	(None, 256, 256, 64)	73,792	concatenate_27[0…
conv2d_131 (Conv2D)	(None, 256, 256, 64)	36,928	conv2d_130[0][0]
conv2d_132 (Conv2D)	(None, 256, 256, 1)	65	conv2d_131[0][0]

Total params: 31,030,593 (118.37 MB)

Trainable params: 31,030,593 (118.37 MB)

Non-trainable params: 0 (0.00 B)

Training

The model was trained in 5 different stages, the first 3 of which are summarized below.

Train 1 - Sample of 2000 images, 1000 from the Normal folder and 1000 from the Lung_Opacity folder. 20% of the data was used for validation.

Epochs:: 20
Batch_Size=16

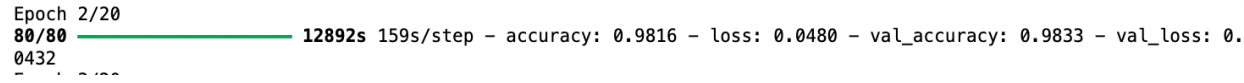
Train 2 -Sampled 1600 images, 400 from each folder

Epochs: 14
Batch_Size=16

Train 3 -Sampled 1600 image, 400 from each folder

Epochs: 2
Batch_Size=16

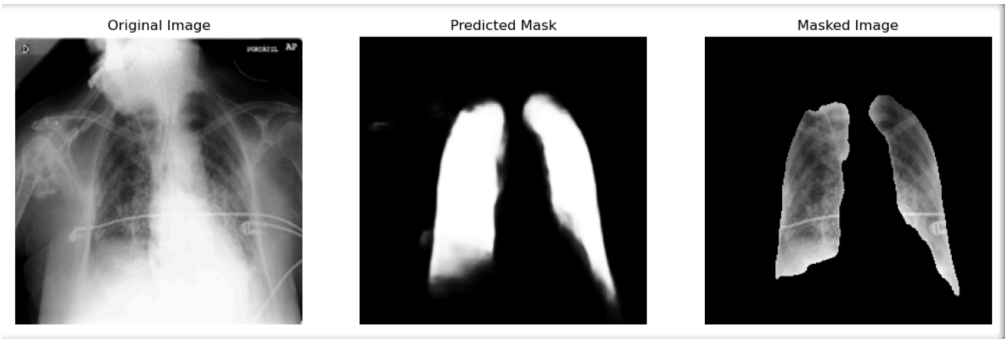
Train 1 and train 2 used data augmentation that included horizontal random flips, random rotations and random brightness. The third train added random contrast, random zooms, and gaussian noise to the data augmentation. After training was stopped on the third run, the accuracy of the model was 98.33% on the specific validation set set up during training.



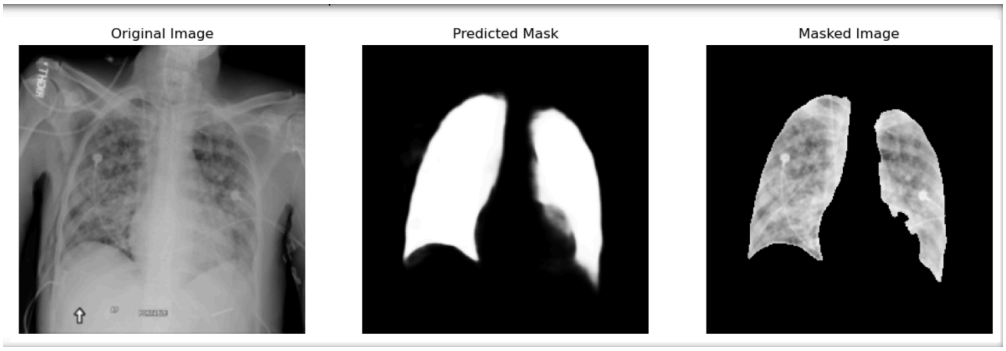
Predicted Masks/masked images After 3 rounds of training

Normal

Covid



Lung Opacity



Viral Pneumonia

Morphological Closing

After visual inspection of the predicted masks it was clear that the model was very accurate. However, there were some images where the model did have trouble distinguishing between lungs and other structures present in the image. Specifically, I noticed that there were images with holes/gaps in the predicted mask or the masked image. There were some instances of images that were more zoomed which caused some of the anatomical structures near the waist to be misclassified as lung. There were others where the intensity of the opacity was so high that it was misclassified as non-lung. I decided to handle these issues using morphological closing. By strategically filling gaps and holes in lung areas, I could better ensure that my pipeline was more accurately finding lung areas. I put together a list of problematic images and evaluated the morphological closing, by visually inspecting them after the operations were performed. The diagram below shows the images that were found to be problematic.

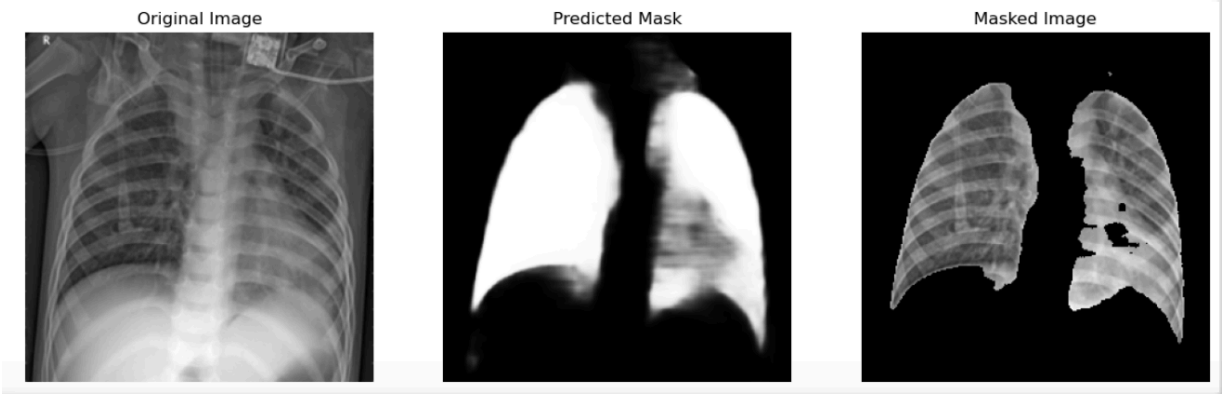
Images with holes/gaps in predicted mask

Problematic Image 1

Problematic Image 2

Problematic Image 3

Problematic Image 4

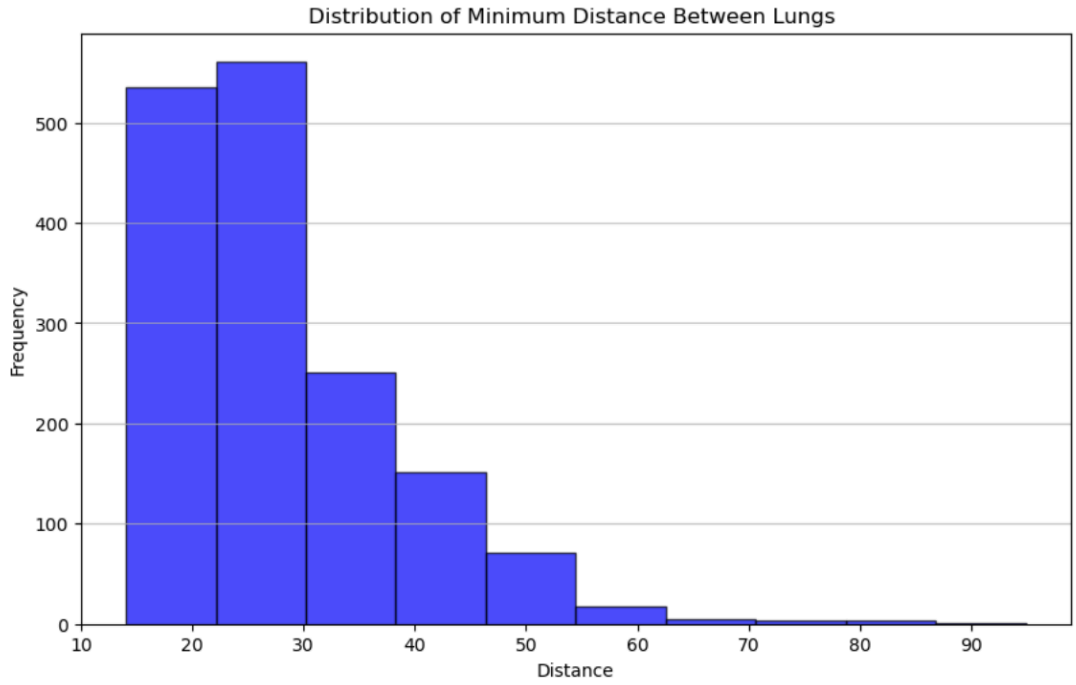


This presented many challenges, for example, sometimes there were disconnected components (islands) that shouldn't be there (aka they are not part of the ground truth mask), and others where perhaps the island should be connected to the main part of the lungs. In image 1 for example, the inlet seen at the top is clearly something that should be filled in, as it is part of the lungs. In image 2 the inlet on the left side should also be filled, as it represents an opacity that could be significant. On the other hand in image

4, the hole that you see is actually most likely the correct part of the mask. In other words the surrounding area that is predicted as lung represents false positives. This is most likely because this is a viral pneumonia case where the opacity overlaps with the heart. The area of the remaining two images have islands that clearly should not be part of the image. I decided to use distance thresholds to determine if a gap should be filled. Filling gaps meant that I needed to be careful to not connect the two lungs, as removing the heart and spine are important parts of the preprocessing pipeline.

I sampled 1600 masks and found the shortest distance between the two lungs in each mask.

```
Mean Distance: 28.38972059899717
Mimimum Distance: 14.035668847618199
```



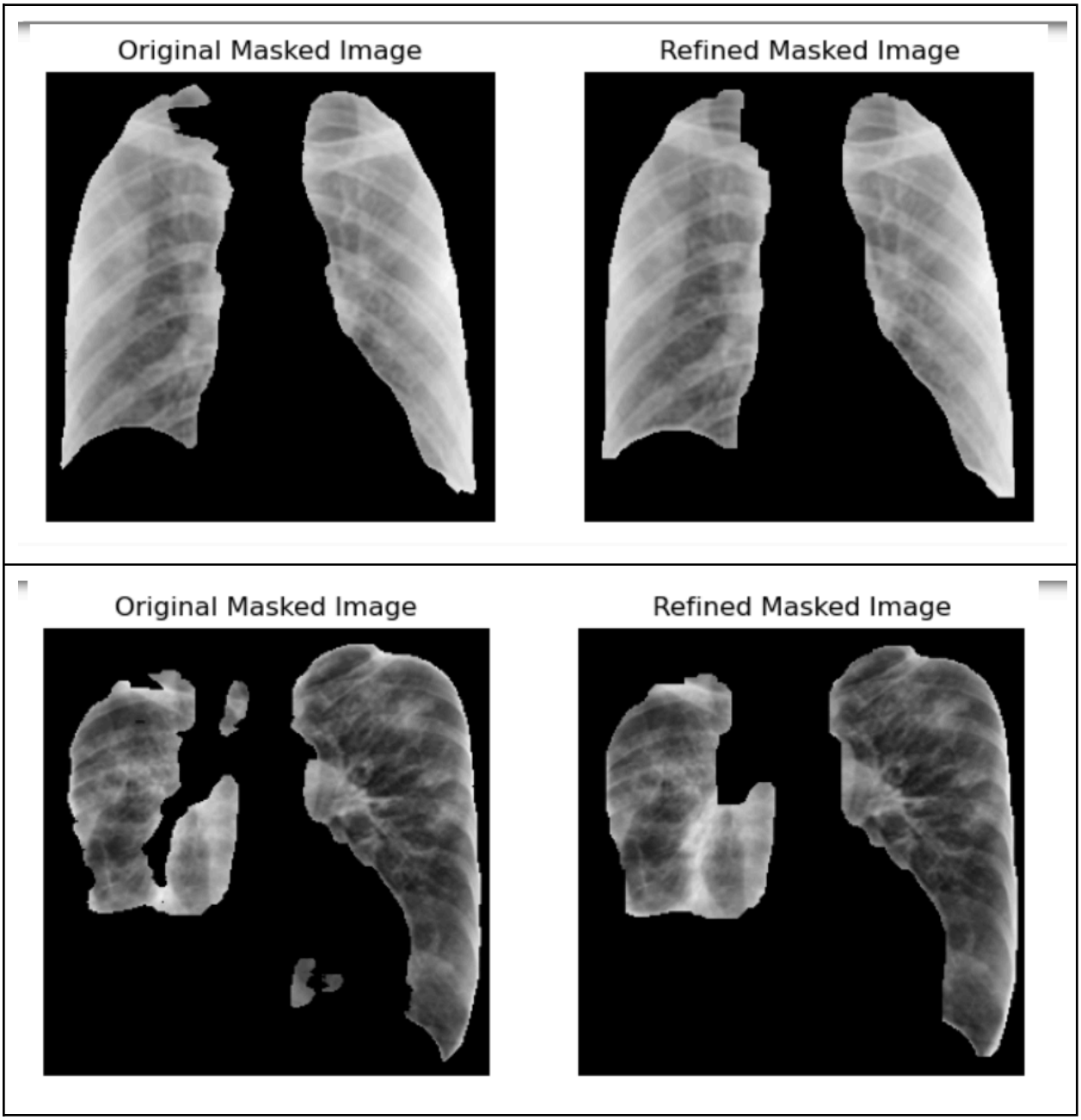
Since the minimum distance between the two lungs was roughly 14 pixels, I used this value to determine the values in the horizontal kernel to fill gaps horizontally. The final pipeline for morphological operations is the following.

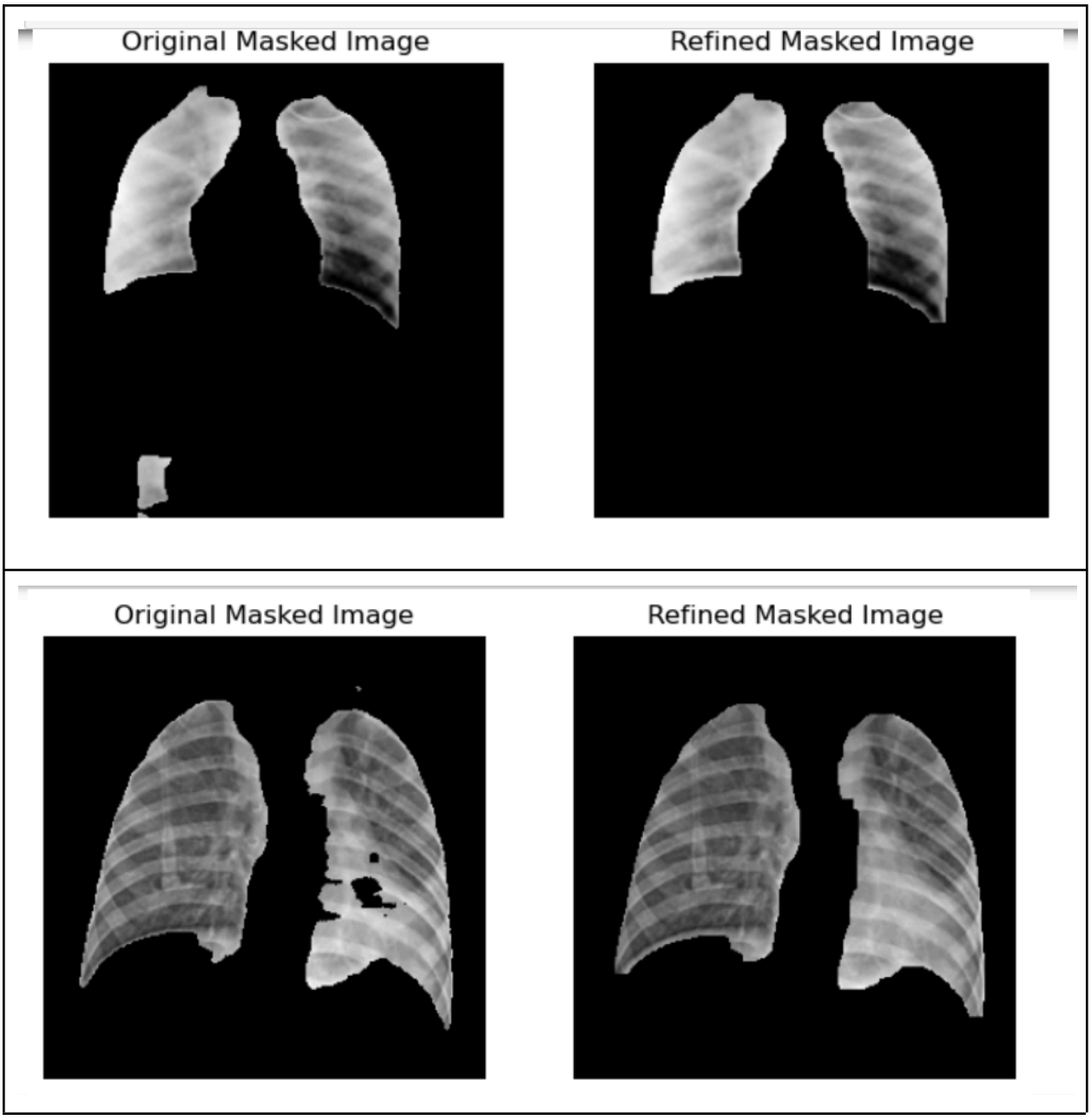
1. Apply vertical filling with kernel 19X1, filling gaps vertically when they are less than 19 pixels away.

2. Apply Horizontal filling with kernel 1X13, filling horizontal gaps that are less than 13 pixels away.

3. Apply vertical filling again with kernel 12X1, filling remaining vertical gaps that are less than 12 pixels away.

4. Remove any remaining any disconnected components (islands)





The results visually looked favorable, but I needed to evaluate the results to a larger set of images. To assess both the results of the Unet model and the model after refinement I created an additional test set of 800 images. The test set included 400 images from the normal class, 200 from the covid class, and 200 from the viral pneumonia class. It was split this way so that I could also use it to test the various preprocessing pipelines on models that classify pneumonia.

Analyzing model accuracy using the Jaccard Index

The jaccard index is a better metric for evaluating the performance of segmentation models. This is because the accuracy metric gives too much weight to predicting the background (negative class). The Jaccard score evaluates the amount of correctly classified positives (aka pixels that are classified as lung and actually are lung) and divides by the sum of all positive pixels and pixels misclassified as positive. This effectively eliminates true negatives from the equation and evaluates how much overlap there is between the ground truth mask and the predicted mask.
Definition of the Jaccard Score: TP / (TP + FP + FN)
The results after the filling operations looked favorable for the particular images that were identified as problematic. Furthermore, upon visual inspection of other images, they all mostly appeared to be unchanged by this refinement. However, the mean Jaccard index on the test set was reduced by this action, going from 0.9277 with no morphological operations to 90.61 with the morphological operations.

Mean Jaccard Index before filling gaps/removing isolated components - 0.9277
Mean Jaccard Index After refinement: : 0.9061

As a result, I decided to apply the vertical/horizontal filling conditionally, only performing the operations if holes were detected in the predicted mask. This produced a very slight increase in the Jaccard index compared to the Jaccard score with no refinement.

Mean Jaccard Index before refinement : 0.9277
Mean Jaccard Index after conditional refinement : 0.9280

More model Fitting

Train 4: Sample 400 of each class. Same augmentation as train 3:

Epochs: 3

Batch_Size=16

Train 5: Sample 400 of each class. Same augmentation as train 3:

Epochs: 14

Batch_Size=16

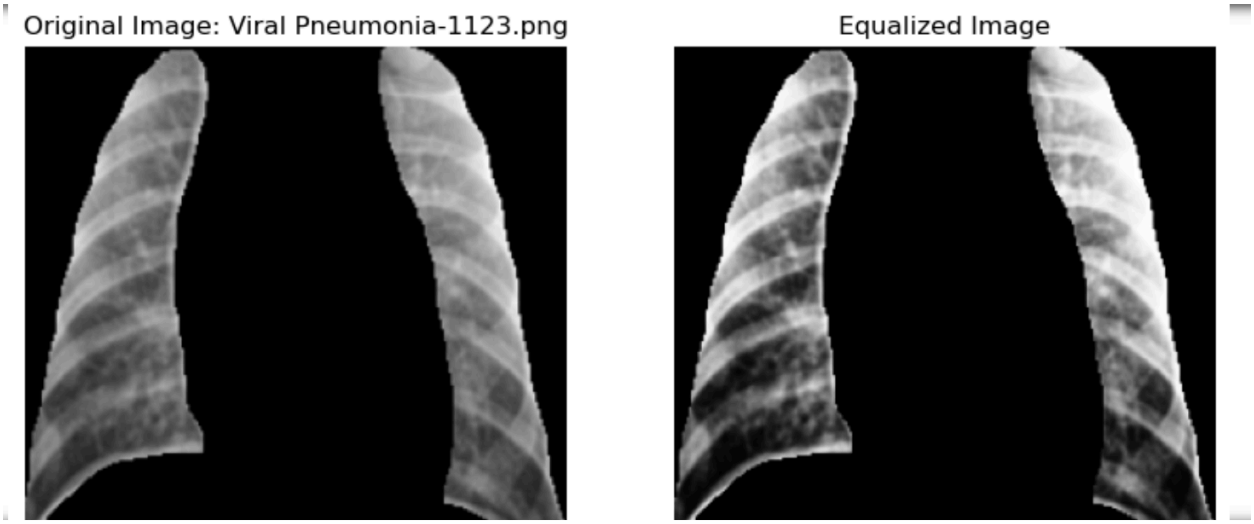
At this point I decided to focus my attention back to fitting the segmentation model. The model, up to this point, was 98.33%. However, early stopping had not yet been realized and fitting was only stopped for time reasons. I sampled a different training set and ran roughly 12 more epochs, using the same data augmentation as the last train. After the final train, the model accuracy improved to 98.66% on the current validation set. The jaccard score on the test set was also improved.

Jaccard Score after model 6 (no refinement): 0.946713022845746

Jaccard Score after model 6 (with refinement): 0.9467327391745909

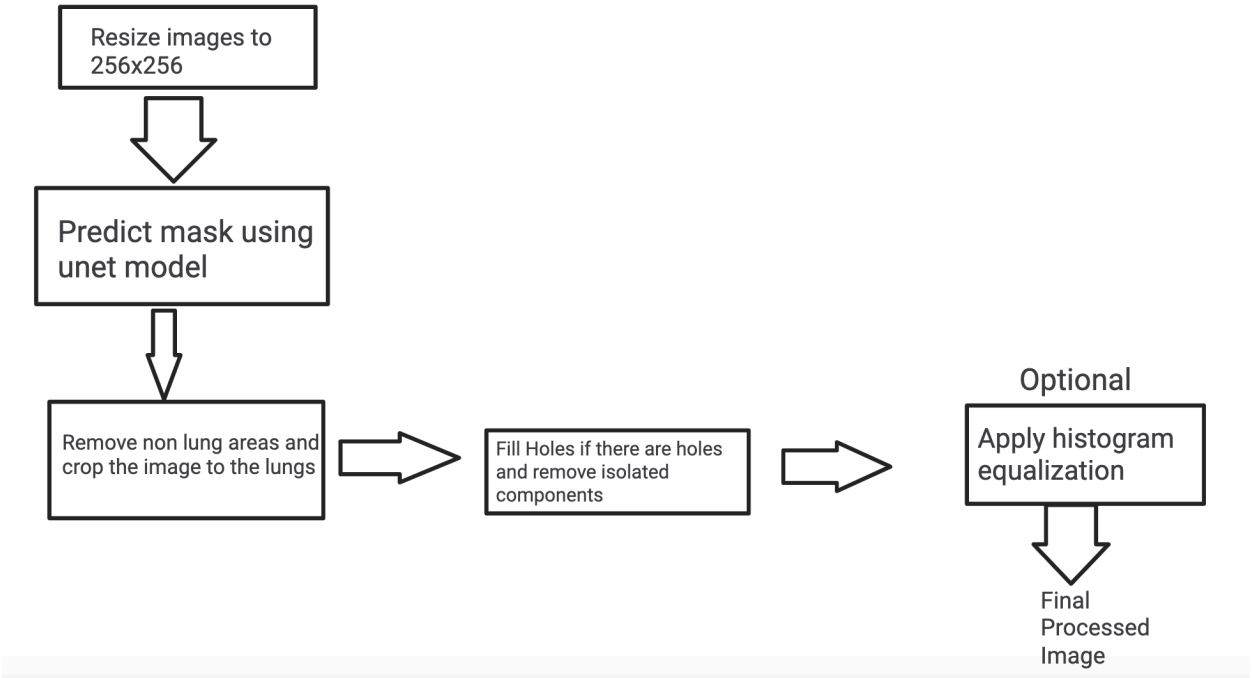
Histogram Equalization

I decided to add an optional parameter for histogram equalization. By enhancing the contrast of the image, histogram equalization could potentially be used to bring out certain features in the image and highlight opacities. During this process I fit several models to predict pneumonia using covid and viral pneumonia images as the positive class. Histogram equalization had the effect of improving model performance with respect to the f1 and recall scores. The model was perfect with respect to predicting the positive class, however it also generated more false positives than did the model without histogram equalization. I think that this could be further optimized to reduce the false positives, so for now I am leaving the histogram equalization as an option in the overall pipeline. Below is an example of a processed image from the viral pneumonia class with and without histogram equalization.



It is noteworthy that the size and color of the background will change the results of the equalized image. Because each person's lungs are not the same size, the amount of background is varying from image to image. Because of this It may be worth exploring histogram equalization with different backgrounds (e.g all white background, or gray background). Another thing that could be explored is figuring out how to cut out the middle of the image and creating a uniformly sized background.

Description of Final Pipeline

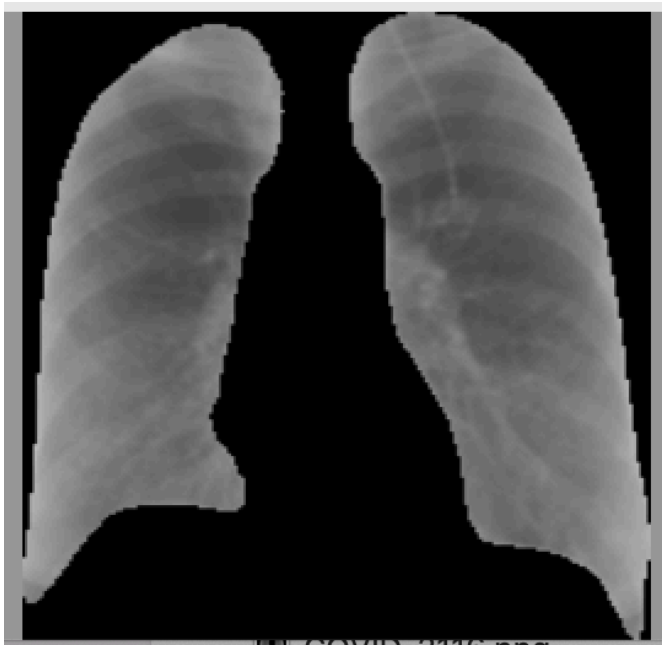


Conclusion

In the end I was able to create a pipeline that will be useful for further modeling with respect to lung diseases. The segmentation model that was made has an accuracy of somewhere around 98% and a jaccard score of somewhere around 94%. The model can still be improved with more training. In order to make further improvements to the pipeline, more information is needed on the exact preprocessing requirements needed to carry out tasks such as fitting models that predict pneumonia or covid-19 pneumonia. With more time, it would be useful to explore various background choices that could improve the effect of the histogram equalization. Furthermore, exploring other types of histogram equalization such as Clahe, could potentially produce better results. I believe this pipeline is a valuable step towards the goal of creating better models that can detect lung diseases. Below are images that would be produced by the pipeline.

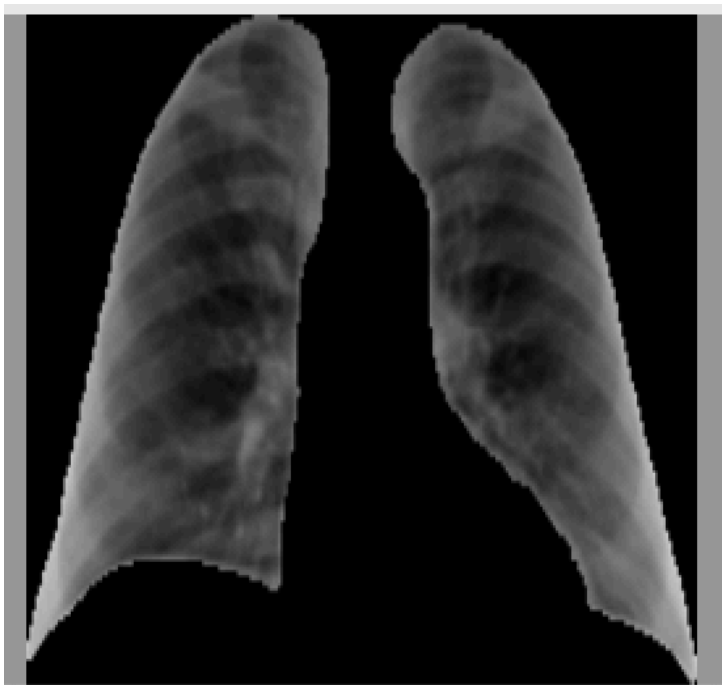
Final Results

Covid 19 (No Equalization)

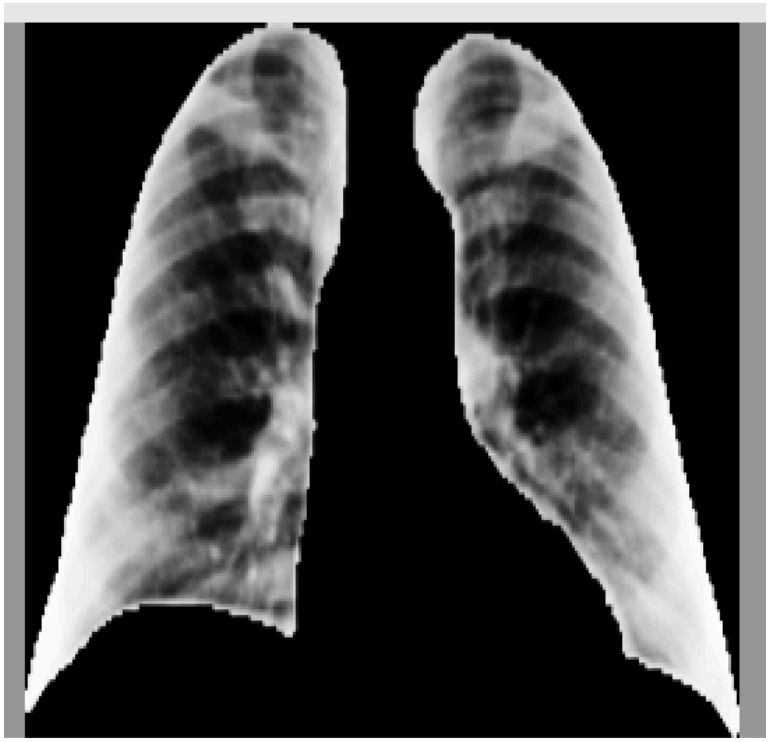


Covid 19 (With Equalization)

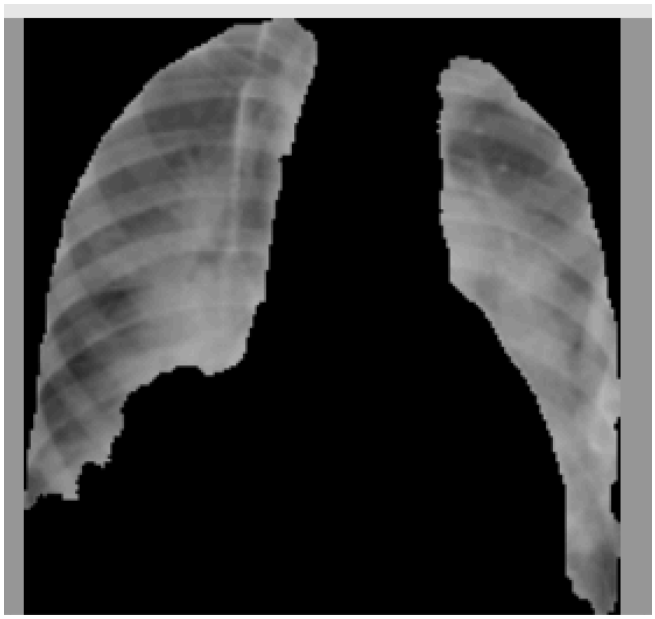
Normal (No equalization)



Normal (With equalization)



Viral Pneumonia (No equalization)



Viral Pneumonia (With equalization)