

# CT Scan COVID Diagnosis

CS 4641 Fall 2020: Team 39's Machine Learning Project

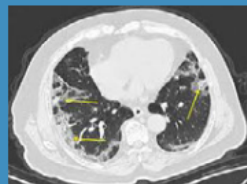
[View Cs4641-group39.GitHub.io on GitHub](https://github.com/aazmi6/CS4641-Group39.github.io/)

## Summary

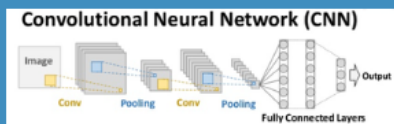
### COVID-19 Testing Through Lung CT Scans

#### Goal

Using Images of Lung CT-Scans to detect COVID-19 more rapidly



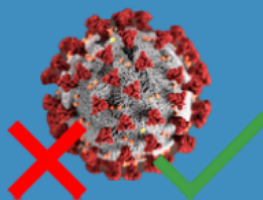
#### Methodology



Using a dataset that involves both healthy lungs and COVID-19 inflicted lungs, we plan to train a CNN-based model to detect whether or not the lungs show signs of COVID-19

Our model will detect with some confidence value on whether or not the lung CT scan image that we test has COVID-19 or not.

#### Results



Lung CT scans typically take 5 minutes to complete the process. Current testing methods typically take half an hour. Using the results of this project, medical professionals would be able to rapidly detect COVID-19 directly after the scan.

#### Discussion



## Introduction

Our goal is to create a model that will learn from the lung CT scan dataset to determine whether or not new patients are at risk for pneumonia as a result of being COVID-19 positive. With the new knowledge surrounding the virus, it becomes much more crucial that patients seek special attention if the virus has infected the

lungs. With this model, doctors only need to upload the CT scan from the patient and the model will give vital, potentially life-saving information on the next step of treatment.

## Methods

We plan to use lung CT Scans provided on kaggle to determine whether a patient has COVID-19. We're also planning to use a CNN-based model through Pytorch in order to train our model to identify COVID-19 through CT lung scans.

## Results

Given a set of lung images (some patients positive for Covid-19, some patients exposed to other conditions such as pneumonia, and some healthy patients) our model will determine (to some % of confidence) whether or not a given patient's CT lung scan is Covid positive.

## Discussion

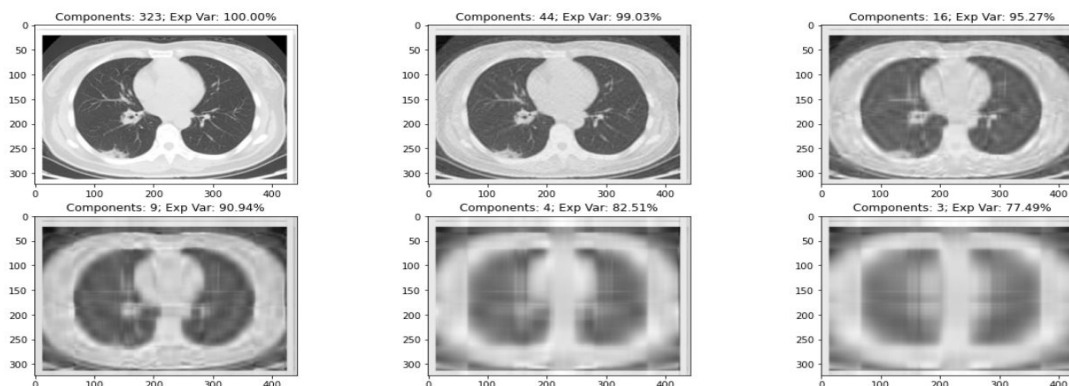
The significance of this relates back to the ease with which doctors will be able to rapidly confirm patients' condition, and subsequently be able to use this information for deciding which route to take with treatment. This will be especially significant for at-risk patients if the CT scan can reveal the results faster than a mouth or nose swab test.

## Unsupervised Learning

### PCA

PCA allows us to remove features in our data that contribute the least to the variance in the data. This can help quicken the training of our supervised models. In addition, it can reveal the most significant features within our data. Here is an example of applying PCA to a CT scan image:

Applying PCA (Negative Patient)



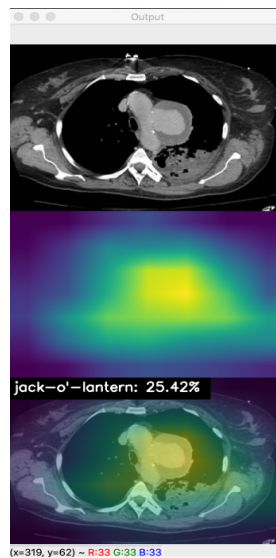
PCA was successfully implemented, but no use cases have been explored yet. We are potentially considering using it for our CNN, which will be implemented in the near future for supervised learning. For most test cases, 99% of variance was able to be recovered while dropping a majority of the components. This indicates that we can apply PCA to reduce the number of components to be analyzed; as a result, fewer computations have to be made to still yield an accurate model.

We can make some observations from the PCA result however. We can immediately tell that important features look like white spaces within the lung scan. It could be picking up on ground glass opacities, which are white-ish/hazy areas indicative of illness in the lung. We will talk more about ground glass opacity in the next few sections.

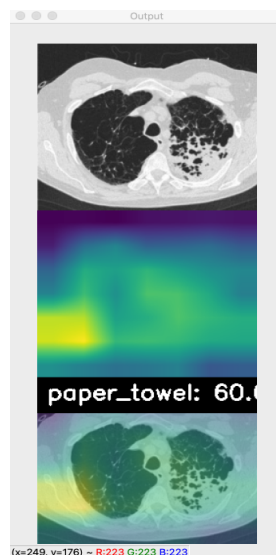
### CAM

Class Activation Maps highlight the parts of the image that the Neural Network focuses on so we can better understand how our neural network works in supervised learning when we reach that stage. CAMs gives us an idea of what part of the data has to be focused on by building a heat map. We pasted this outputted heat map over our original image to understand important visual features.

When we worked with The VGG16 architecture, the output seemed to focus more on the heart area than the rest of the lung scan. The model also didn't have the best accuracy.



When we worked with The resnet architecture, our output highlighted the ground glass opacity in the lungs with a much better accuracy.



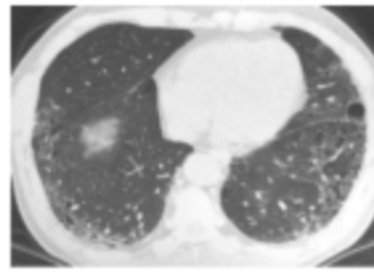
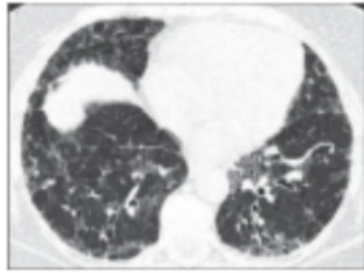
In conclusion, the CAM model tells us that ground glass opacity and the heart are possibly important features, a trend we observe with GMM and DBSCAN as well.

## DBSCAN

DBSCAN is probably not the best way to go for image segmentation because the epsilon parameter is too sensitive and would need to take time to find the perfect value to segment properly. Each image that we ran through DBSCAN needed a drastically different epsilon in order to show results. Even though the resulting images are interesting and helped us realize that there are differences between COVID-positive and negative, it's not the best to reduce noise and portions of the images. Some additional understandings that we found through DBSCAN on the other hand was that many of the CT scans clustered the center area much more larger for negative cases and the center area was smaller for positive COVID cases. This could potentially related to the methods on how medical professionals take COVID CT-Scans but it was one of the observations that we made through our clustering.

### COVID Negative DBSCAN Clustering

$(-0.5, 21.5, 15.5, -0.5)$   $(-0.5, 20.5, 14.5, -0.5)$

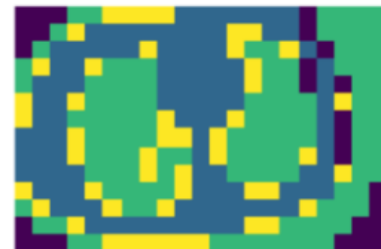
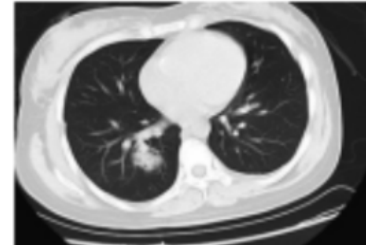
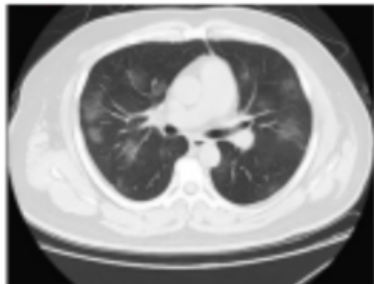


COVID Positive DBSCAN Clustering

$(-0.5, 36.5, 27.5, -0.5)$

e.py'>

Out[97]:  $(-0.5, 20.5, 13.5, -0.5)$



## GMM

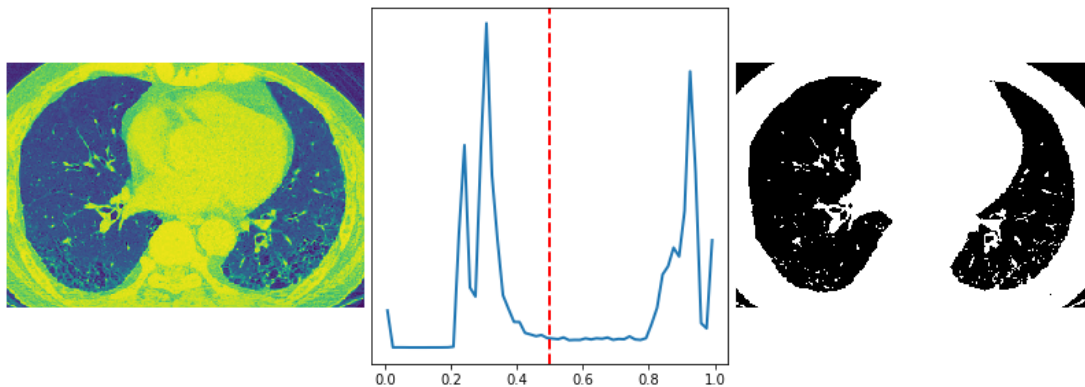
The results found from GMM has several implications towards further understanding what happens to a human lung when coming in contact with the Covid-19. In order to understand the findings from GMM, we must first discuss the capabilities and limitations of GMM in regards to clustering grayscale images.

In terms of capabilities, GMM has the ability to pick up ground glass opacity, which is abnormal findings on a CT scan which result in a hazy, obscured view of the underlying structure of the lungs. Because GMM can recognize these subtle changes, the algorithm bypasses the loss of information from this phenomenon, allowing it to draw focus towards anomalies in the lungs. Now, we certainly are no medical experts, but the ability of the algorithm to filter out the "noise" associated with the CT scans certainly has positive implications in the medical field, not just in terms of Covid-19, but CT scans in general.

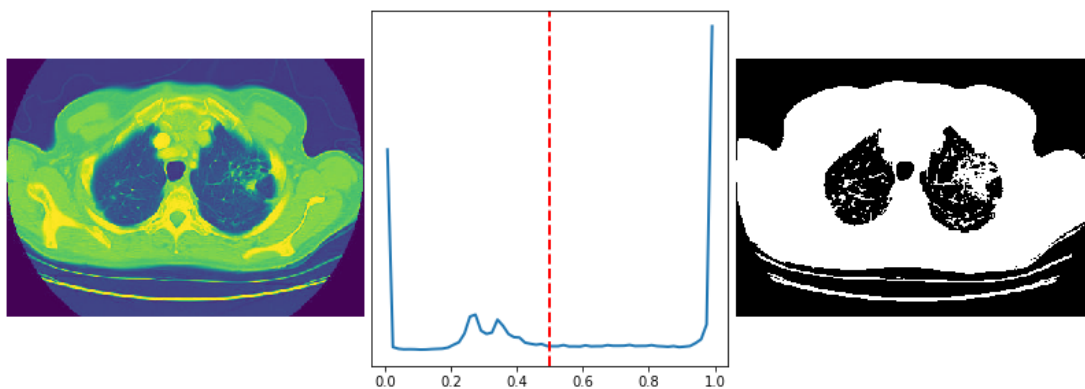
With regards to limitations, GMM was not able to process multiple images at one time, which would be excellent for classifying incoming CT scans. We had to run each image individually, one at a time. This was not only a time consuming process, but also provided the inability for the algorithm to prove it could correctly group negative scans against positive scans, which would certainly prove beneficial for supervised learning.

Below are samples of images generated from the GMM algorithm. The images are labeled as either a Covid-19 negative patient or Covid-19 positive patient. The leftmost image is the original CT scan, the graph represents the bin clustering, indicating how many clusters may exist in the data. The rightmost image is the clustered, grayscale image generated by GMM. We can draw the conclusion that, in general, the center between the lungs was bigger in Covid positive patients and smaller in Covid negative patients, which aligns with our findings from DBSCAN. Whether this is medically significant would be left up to the doctors, but it was consistent across many images.

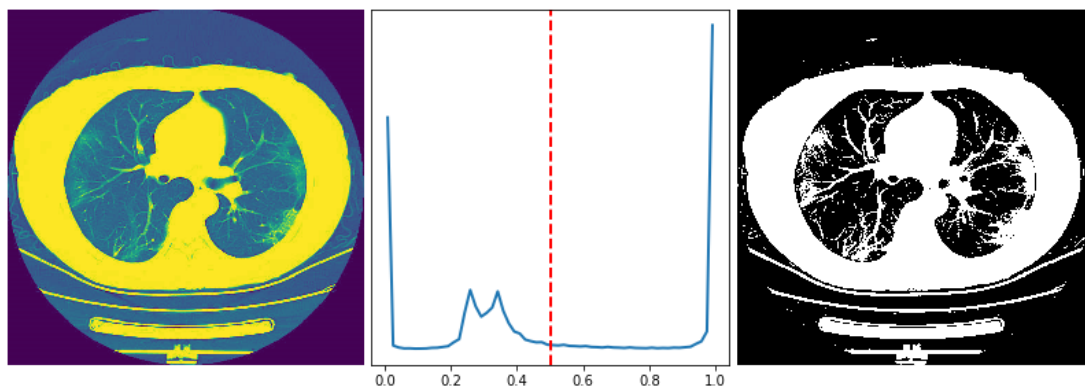
COVID Negative GMM Clustering (Image 1)



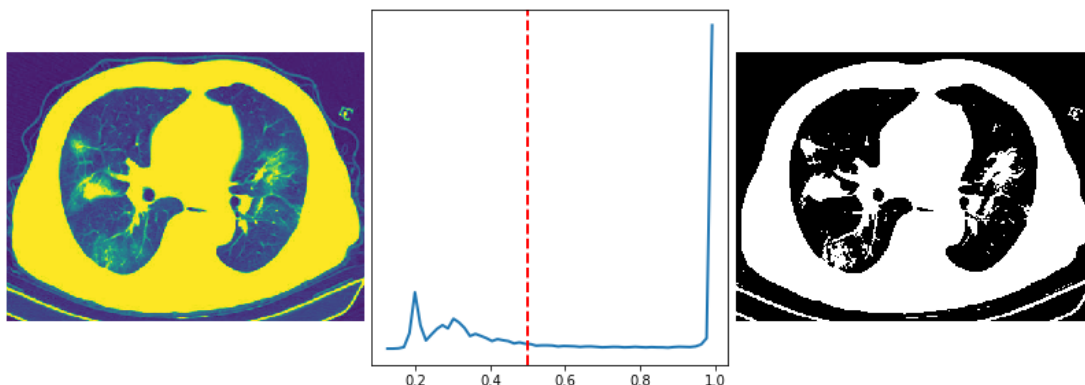
COVID Negative GMM Clustering (Image 2)



COVID Positive GMM Clustering (Image 1)



COVID Positive GMM Clustering (Image 2)



## Supervised Learning

### Convolutional Neural Network

For the supervised portion, we decided to use convolutional neural networks in order to predict whether or not someone has COVID based on a lung CT scan.

#### Initial values:

- Data Cleaning
  - Scaled from 0 to 255
  - Random Shear of 0.2
  - Random Zoom of 0.2
- Data Splitting
  - 70% training
  - 20% validation
  - 10% testing
- Model Structure

#### CNN Structure

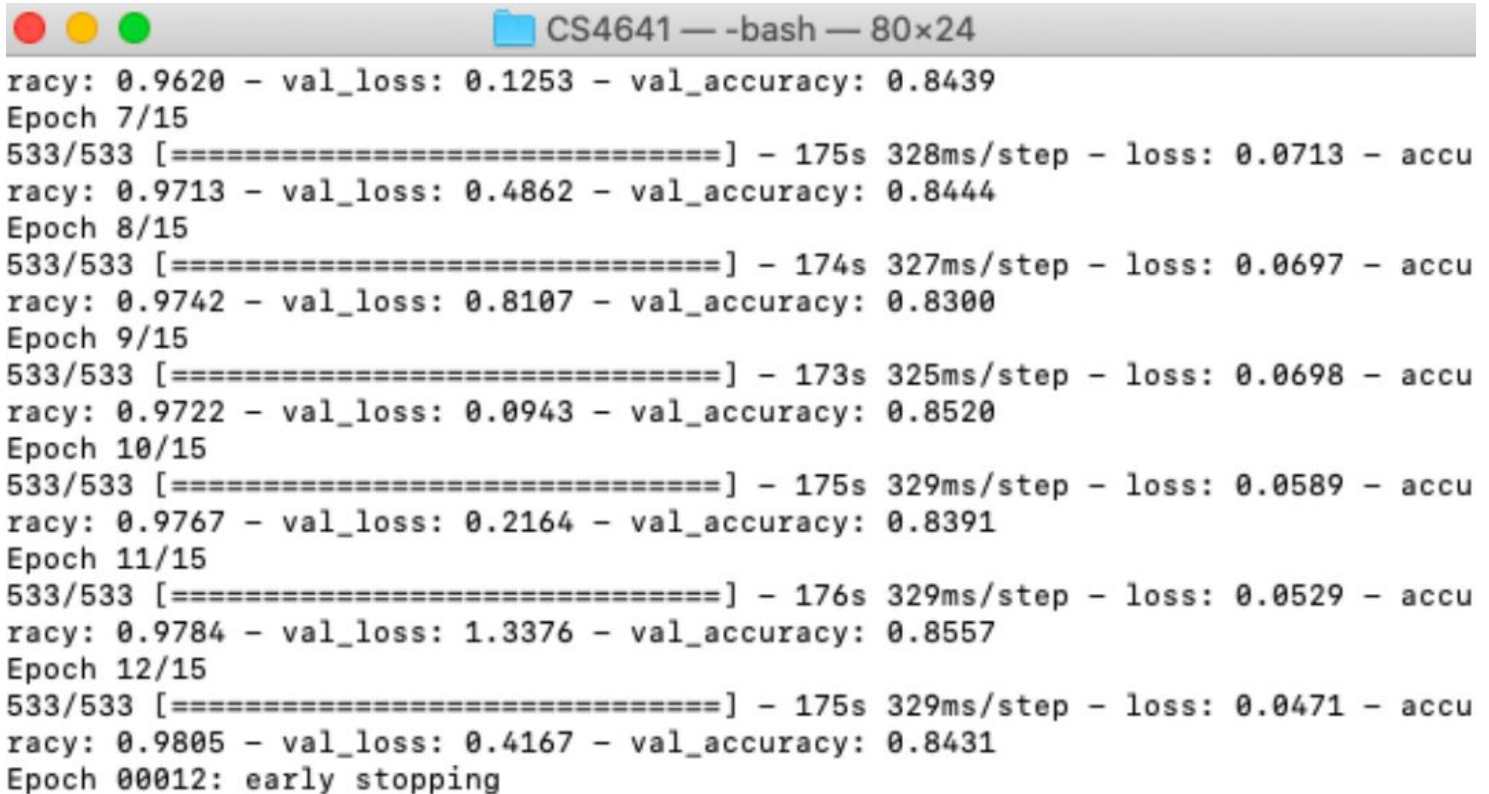
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 32)	9248
max_pooling2d_2 (MaxPooling2)	(None, 36, 36, 32)	0
conv2d_3 (Conv2D)	(None, 34, 34, 32)	9248
max_pooling2d_3 (MaxPooling2)	(None, 17, 17, 32)	0
flatten_1 (Flatten)	(None, 9248)	0
dense_1 (Dense)	(None, 64)	591936
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65
=====		
Total params: 611,393		
Trainable params: 611,393		
Non-trainable params: 0		



As you can see on our model structure, we have 3 sets of Conv2D and then max pooling, then the model flattens the data. Then there's 2 layers in the fully connected layers. The first layer uses relu activation function and the second layer uses sigmoid activation function. We used an adam optimizer as well as a binary cross entropy loss function.

Before our touchpoint, we were able to achieve 84.36% accuracy as shown below:

#### First Accuracy Result



```

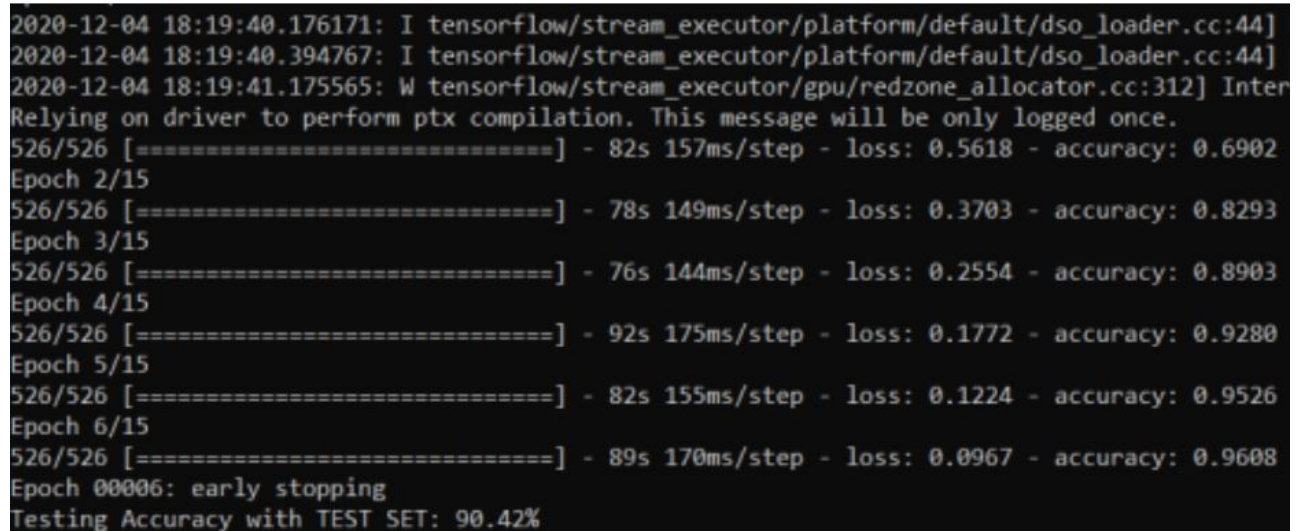
racy: 0.9620 - val_loss: 0.1253 - val_accuracy: 0.8439
Epoch 7/15
533/533 [=====] - 175s 328ms/step - loss: 0.0713 - accu
racy: 0.9713 - val_loss: 0.4862 - val_accuracy: 0.8444
Epoch 8/15
533/533 [=====] - 174s 327ms/step - loss: 0.0697 - accu
racy: 0.9742 - val_loss: 0.8107 - val_accuracy: 0.8300
Epoch 9/15
533/533 [=====] - 173s 325ms/step - loss: 0.0698 - accu
racy: 0.9722 - val_loss: 0.0943 - val_accuracy: 0.8520
Epoch 10/15
533/533 [=====] - 175s 329ms/step - loss: 0.0589 - accu
racy: 0.9767 - val_loss: 0.2164 - val_accuracy: 0.8391
Epoch 11/15
533/533 [=====] - 176s 329ms/step - loss: 0.0529 - accu
racy: 0.9784 - val_loss: 1.3376 - val_accuracy: 0.8557
Epoch 12/15
533/533 [=====] - 175s 329ms/step - loss: 0.0471 - accu
racy: 0.9805 - val_loss: 0.4167 - val_accuracy: 0.8431
Epoch 00012: early stopping

```

#### Testing Accuracy with TEST SET: 84.36%

Afterwards, in order to improve the accuracy of the model after the touchpoint and their feedback, we decided to increase our shear to 0.25 and our zoom to 0.25 and achieved an accuracy of >90%:

#### Second Accuracy Result



```

2020-12-04 18:19:40.176171: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
2020-12-04 18:19:40.394767: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
2020-12-04 18:19:41.175565: W tensorflow/stream_executor/gpu/redzone_allocator.cc:312] Inter
Relying on driver to perform ptx compilation. This message will be only logged once.
526/526 [=====] - 82s 157ms/step - loss: 0.5618 - accuracy: 0.6902
Epoch 2/15
526/526 [=====] - 78s 149ms/step - loss: 0.3703 - accuracy: 0.8293
Epoch 3/15
526/526 [=====] - 76s 144ms/step - loss: 0.2554 - accuracy: 0.8903
Epoch 4/15
526/526 [=====] - 92s 175ms/step - loss: 0.1772 - accuracy: 0.9280
Epoch 5/15
526/526 [=====] - 82s 155ms/step - loss: 0.1224 - accuracy: 0.9526
Epoch 6/15
526/526 [=====] - 89s 170ms/step - loss: 0.0967 - accuracy: 0.9608
Epoch 00006: early stopping
Testing Accuracy with TEST SET: 90.42%

```

Introducing a random rotation of 15 degrees further improved our accuracy to 93.68%:

#### Third Accuracy Result

A terminal window with a black background and yellow text. The text shows a progress bar at the top, followed by 'Epoch 00008: early stopping' and 'Testing Accuracy with TEST SET: 93.68%'.

```
507/507 [=====]  
Epoch 00008: early stopping  
Testing Accuracy with TEST SET: 93.68%
```

We also attempted to use the ResNet50 architecture. Unfortunately, it did not work as we anticipated. Training took a long time, and the model was quite inaccurate as well.

## Conclusions

Being able to achieve an accuracy of 93.68% has great implications in terms of being able to quickly identify COVID-19 for medical professionals and their patients. By using our model, the decision making process is cut down to mere seconds and can further affirm a professional's analysis of lung CT scans. In addition, as COVID-19 lung CT scan datasets begin to diversify to severity and cross comparisons to other diseases, this project demonstrates that machine learning can be applicable in identifying other respiratory diseases outside of COVID-19 and even differentiate between COVID-19 and other diseases. Furthermore, our project also demonstrates the usefulness of unsupervised techniques on lung ct scans as well. When running DBSCAN and GMM as ways to segment images, they both helped outline patterns that one can find between COVID-19 negative and positive images. The CAM unsupervised technique could also be essential as they show that neural networks seem to respond to ground glass opacities. Both our unsupervised and supervised techniques could potentially aid medical professionals in making better conclusions about lung CT scans overall.

## References

- [Description of pneumonia, and its impact on lungs](#)
- [Types of damages to lungs](#)
- [Extent of injury due to COVID-19 on the lungs](#)
- [COVID-19 lung CT scan image dataset](#)



[aazmi6](#) maintains [Cs4641-group39.GitHub.io](#)

Find the code for our project at <https://github.gatech.edu/ssubramanian87/Lung-COVID-Detection> This page generated using [GitHub Pages](#) theme by [Jon Rohan](#)