Deep learning to train and classify CT-scans of COVID19

Vidyasagar Kummarikunta
DSC 680 – Applied Data science

Professor Catie Williams
Bellevue University

Contents

Introduction

On March 11, 2020, the World Health Organization (WHO) declared the novel coronavirus (also referred to as COVID-19), a pandemic. As we all are well aware that COVID-19 is an infectious disease that has caused and is continuing to cause numerous deaths all over the world infecting millions of people. The ultimate solution to put an end to the pandemic is to find an effective vaccine. Scientific community around the world is racing to find a vaccine and meanwhile, many measures are being taken by countries to contain this pandemic. To contain the pandemic, one of the important steps to be taken is to test the infected patients.

COVID-19 causes severe respiratory symptoms and is associated with relatively high ICU admission and mortality. Testing COVID-19 involves analyzing samples that indicate the present or past presence of severe acute respiratory syndrome-associated coronavirus 2 (SARS-CoV-2). The test is done to detect either the presence of the virus or of antibodies produced in response to infection. COVID-19 diagnostic approach is mainly divided into two broad categories, a laboratory-based approach, which includes point of care-testing, nucleic acid testing, antigens tests, and serology (antibody) tests. The other approach is using medical imaging diagnostic tools such as X-ray and computed tomography (CT). CT scan involves transmitting X-rays through the patient's chest, which are then detected by radiation detectors and reconstructed into high-resolution medical images. There are certain patterns to look out for in a chest CT scans which present themselves in different characteristic manifestations. (Ilker Ozsahin, 2020).

Right now, the gold standard of testing COVID-19 in infected patients is a nucleic acid testing called - RT-PCR (Reverse Transcription – Polymerase Chain Reaction). However, due to some underlying technicalities, the test is not always reliable and is shown to have sensitivity between 60-71% (H. Bai, 2020). Also, during the peak of the pandemic, there was shortage of test kits and healthcare providers had to look for alternative methods for confirming the disease in an infected patient. Among the alternative methods, CT scans have been successfully used for screening and diagnosing COVID-19. Very recently, these CT scans have been open sourced to seek help from general public from various scientific backgrounds. Various studies reported that sensitivity for CT scans is about 98% while for RT-PCR, it is about 71% (Yicheng Fang, 2020).

For instance, **Fig.1** shows what a CT scan of a healthy person looks like –

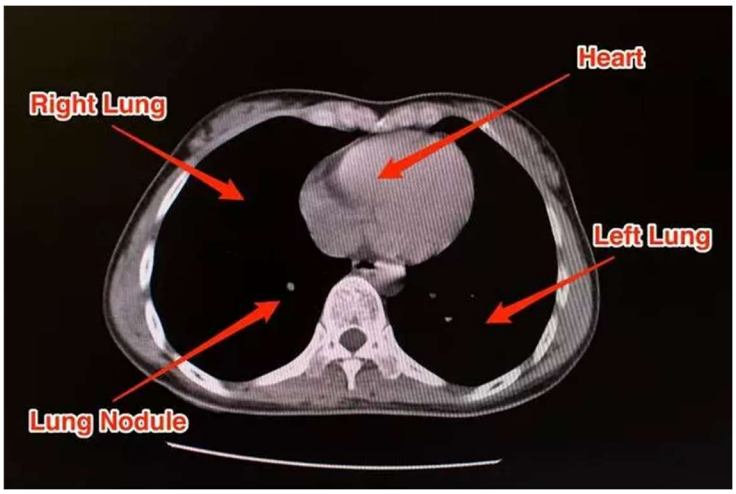


Fig.1 (Kathuria, 2020)

Fig.2 shows what a CT scan of a patient who is suffering from pneumonia brought on by COVID-19.

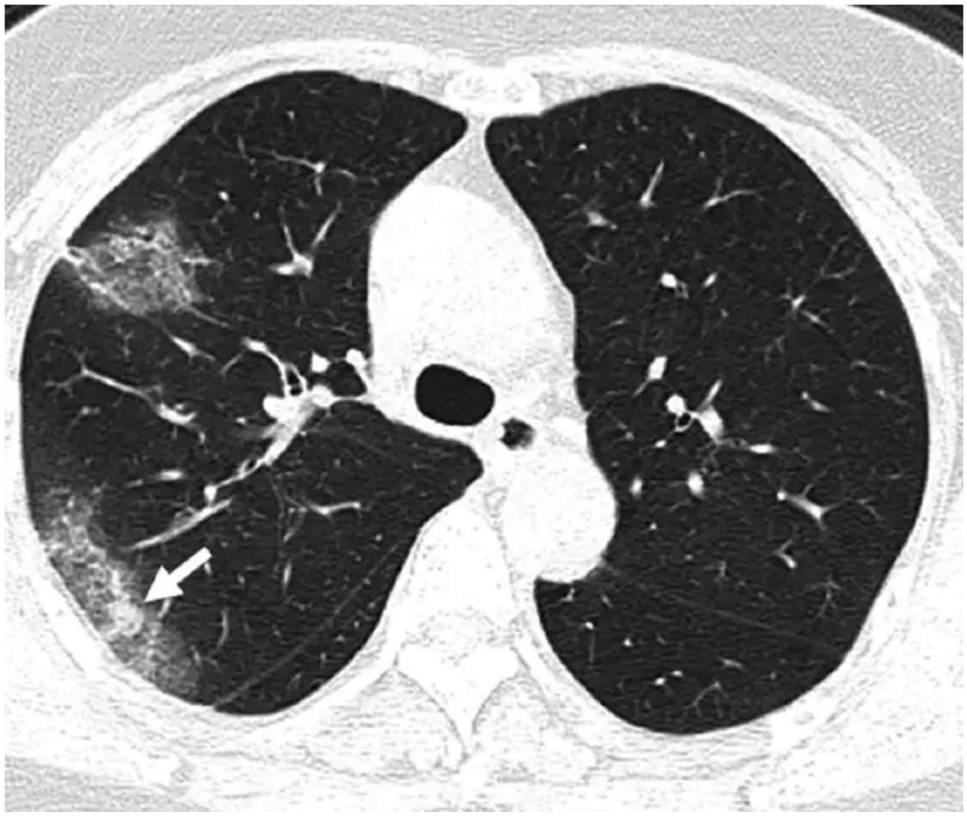


Fig.2 (Kathuria, 2020)

If we observe closely, the opacity in the lungs of infected CT scan is caused by pneumonia. However, it is pointed out (Xiaowei Xu, et al., 2020) that the CT scans of COVID-19 have some distinct features that separate them from other types of pneumonia resulting from different causes. The three main distinctive features of COVID-19 CT scans observed are –

1. Ground-glass appearance - Opacities in the lung look like ground-glass
2. Striking peripheral distribution along with the pleura – Implies that the majority of these opacities occur along the edge of the lung
3. More than one independent focus of infections for one case – Implies that there can be more than one such cluster of opacities

Now, what does artificial learning have to do with all this? Machine learning has proven to be useful in medical applications since its inception, and it became widely accepted due to its high prediction and accuracy rates. In the diagnosis stage of COVID-19, machine learning algorithms can be used to recognize patterns on medical images taken by CT. Machine learning algorithms can be used to segment regions of interest and capture fine structures in chest CT images, self-learned features can easily be extracted for diagnosis and other applications. A recent study showed that machine learning models can accurately detect COVID-19 in CT images and was also able to differentiate it from other lung diseases and community-acquired pneumonia (Ilker Ozsahin, 2020). Deep learning technologies, such as convolutional neural network (CNN) with the strong ability of nonlinear modeling, have extensive applications in medical image processing. For this study, I will be using transfer learning to build a model that will classify COVID-19 and non-COVID-19 images.

Data sources

The CT scan images are obtained from Github page provided by University of California, San Diego (Zhao, 2020). According to the Github page, the images are collected from COVID19-related papers from medRxiv, bioRxiv, NEJM, JAMA, Lancet, etc. CTs containing COVID-19 abnormalities are selected by reading the figure captions in the papers. All copyrights of the data belong to the authors and publishers of these papers.

Here is the link to the UCSD Github page: https://github.com/UCSD-AI4H/COVID-CT

The data was downloaded as a zip file. There are two classes of images –

1. First class (positive) consists of CT scans of COVID-19 positive patients.
2. Second class (negative) consists of a mixture CT scans of healthy patients and patients suffering from other non-COVID-19 related diseases that may cause opacities in the lungs.

The dataset is divided into three splits: the train set (425 examples), validation set (118 examples), and the test set (203 examples). Information for this split has been provided in the folder Data-split folder. This folder contains text files which explain which files belong to each split.

Problem statement

The objective of this project is to apply computer vision-based deep learning algorithms as a tool to help diagnose COVID-19 by using CT scan of a patient. This project is focused on testing a model that can classify the CT scan images of COVID-19 and non-COVID-19 with greater accuracy.

Approach

For this project, I will be using Python program, Jupyter notebook and PyTorch to build the classifier. First step is to understand the images. After the images are loaded, they have to be preprocessed to evaluate the model. Performance metrics such as sensitivity, specificity and area under ROC are calculated.
It is challenging to train a new deep learning model targeting COVID19 medical images, due to insufficient amounts of sample data available publicly. In this scenario, the best alternative is to apply transfer learning where a deep learning network is pre weighted with the results of a previous training cycle from a different domain. For this study, I would like to use the pretrained VGG-19 with batch normalization. Next step is to train the hyperparameters. Once the model is run, the model with best validation accuracy will be chosen to compute performance metrics. Finally, I intend to use another pretrained model (VGG-16) which has fewer layers of architecture compared to VGG-19 to observe if we can achieve better or diminished accuracy.

Methods and Results

Loading COVID-19 positive images –

```
covid_files_path = '/Users/12702/Desktop/MODatascience/DSC-680/Project2/COVID-CT/Images-processed/CT_COVID/'
covid_files      = [os.path.join(covid_files_path, x) for x in os.listdir(covid_files_path)]
covid_images     =  [cv2.imread(x) for x in random.sample(covid_files, 5)]

plt.figure(figsize=(20,10))
columns = 5
for i, image in enumerate(covid_images):
    plt.subplot(len(covid_images) / columns + 1, columns, i + 1)
    plt.imshow(image)
```

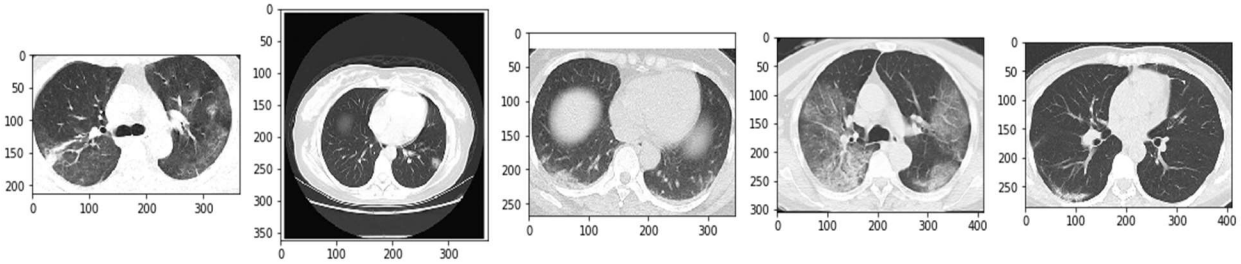


Fig.3: Sample images of the COVID-19 positive CT scans

Loading non-COVID-19 images –

```
non_covid_files_path = '/Users/12702/Desktop/MODatascience/DSC-680/Project2/COVID-CT/Images-processed/CT_NonCOVID/'
non_covid_files      = [os.path.join(non_covid_files_path, x) for x in os.listdir(non_covid_files_path)]
non_covid_images     = [cv2.imread(x) for x in random.sample(non_covid_files, 5)]

plt.figure(figsize=(20,10))
columns = 5
for i, image in enumerate(non_covid_images):
    plt.subplot(len(non_covid_images) / columns + 1, columns, i + 1)
    plt.imshow(image)
```

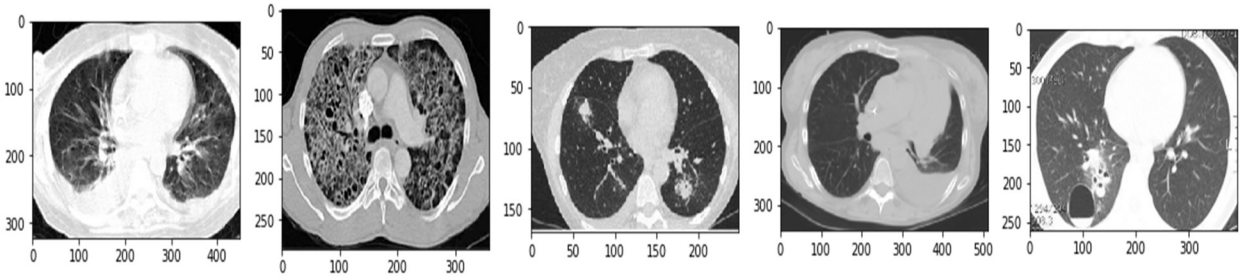


Fig.4: Sample images of the non-COVID-19 CT scans

The dataset is already divided into train, validation and test sets. Information for this split is provided in the 'Data-split' folder of the source dataset. After loading the data, the dataset returns a dictionary containing the image tensor, the label tensor and list of image paths that were included in the batch.

For training data, the following steps were taken to prepare images. Images were resized to 256 while maintaining the aspect ratio. Images were cropped to the random size ranging from 50% to 100%. Cropped images were resized to 224x224 and applied horizontal flip of the image. Normalized the image to have 0 mean and standard deviation of 1.
For the test data, images were resized to 224x224 and normalized to have 0 mean and standard deviation 1. The non-COVID cases are labeled '0' and COVID positive cases are labeled '1'.

```
normalize = transforms.Normalize(mean=[0,0,0], std=[1,1,1])
train_transformer = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomResizedCrop((224),scale=(0.5,1.0)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    normalize
])

val_transformer = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    normalize
])
```

To calculate accuracy of the model, performance metrics were defined. For this study, I have chosen to compute sensitivity, specificity, accuracy, confusion matrix and area under ROC.

For the model, I decided to go with pretrained VGG-19 with batch normalization. The final linear layer which had one neuron as its output is replaced with 2 to perform transfer learning.

```
model = vgg19_bn(pretrained=True)
model.classifier[6] = nn.Linear(4096, 2)
model.to(device)
```

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mo
de=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
)
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
    (9): ReLU(inplace=True)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
```

```
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (19): ReLU(inplace=True)
    (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (22): ReLU(inplace=True)
    (23): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (24): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (25): ReLU(inplace=True)
    (26): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
    (27): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (32): ReLU(inplace=True)
    (33): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (34): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (35): ReLU(inplace=True)
    (36): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (37): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (38): ReLU(inplace=True)
    (39): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
    (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (42): ReLU(inplace=True)
    (43): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (44): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (45): ReLU(inplace=True)
```

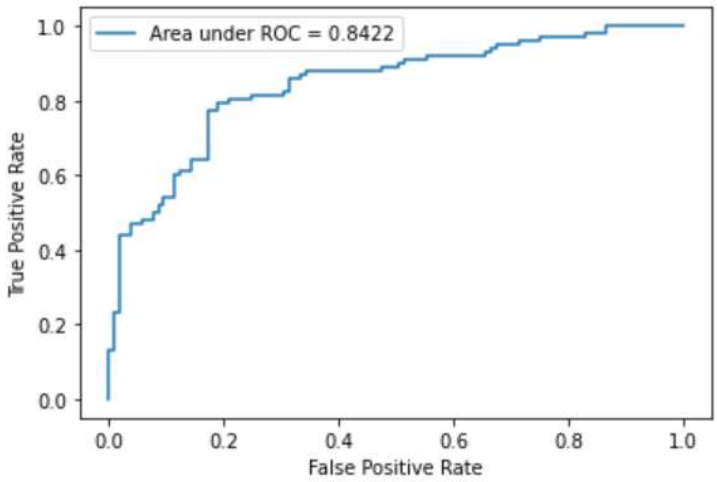
```
    (46): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (47): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (48): ReLU(inplace=True)
    (49): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    (50): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    (51): ReLU(inplace=True)
    (52): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_m
ode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=2, bias=True)
  )
)
```

For training hyperparameters, an initial learning rate of 0.01 is applied. A class called ‘EarlyStopping’ is implemented to keep the running averages of both loss and accuracy. If the metric does not improve beyond a set number of epochs then the method stops. The model is being trained for a maximum of 60 epochs.

After the training loop, the best performing model is selected for testing performance. The model with the best validation accuracy is stored and is accessed as shown below –

```
model = torch.load('best_model.pkl')
```

The accuracy results of the best performing model are shown below –




```
------------------- Test Performance -----------------------------------
Accuracy         0.764
Sensitivity      0.847
Specificity      0.686
Area Under ROC   0.842
------------------------------------------------------------------------
```

Fig.5: Performance metrics of model trained with VGG-19

As we can see in **Fig.5** that the accuracy of the model predictions is promising at 76 %. The area under ROC is 0.842 which implies that there is 84% chance that the model will predict the CT scans correctly. In comparison to the accuracy of RT-PCR testing sensitivity for COVID-19 which is at 60-71% (H. Bai, 2020), the model we trained is showing higher accuracy.

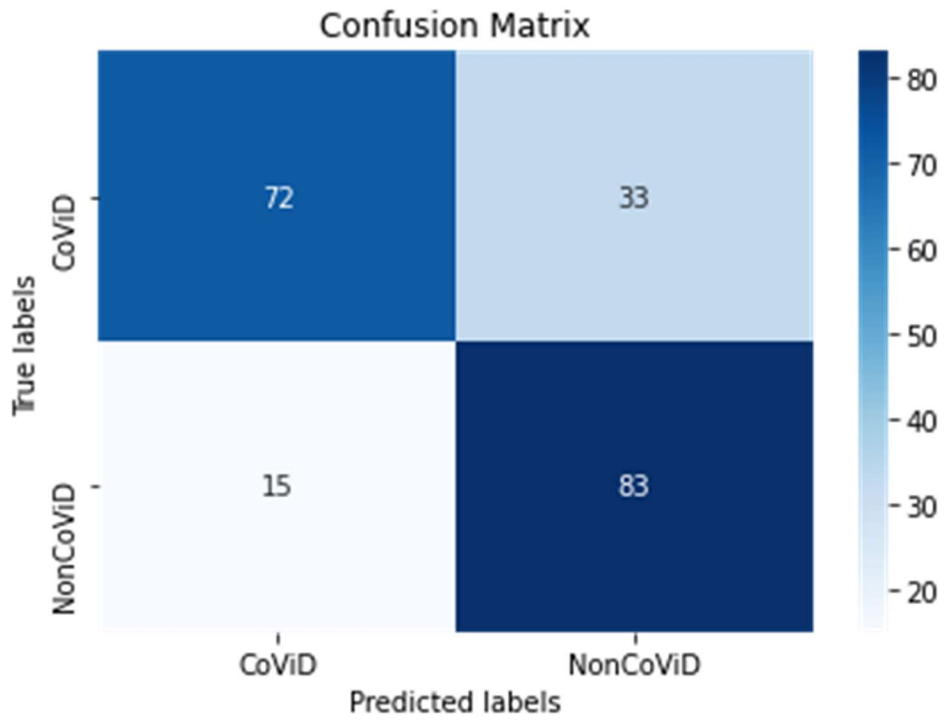


Fig.6: Confusion matrix of model trained with VGG-19

According to the confusion matrix (**Fig.6**), the true positives are at 72, true negatives are 83, false positives are 33 and false negatives are 15.

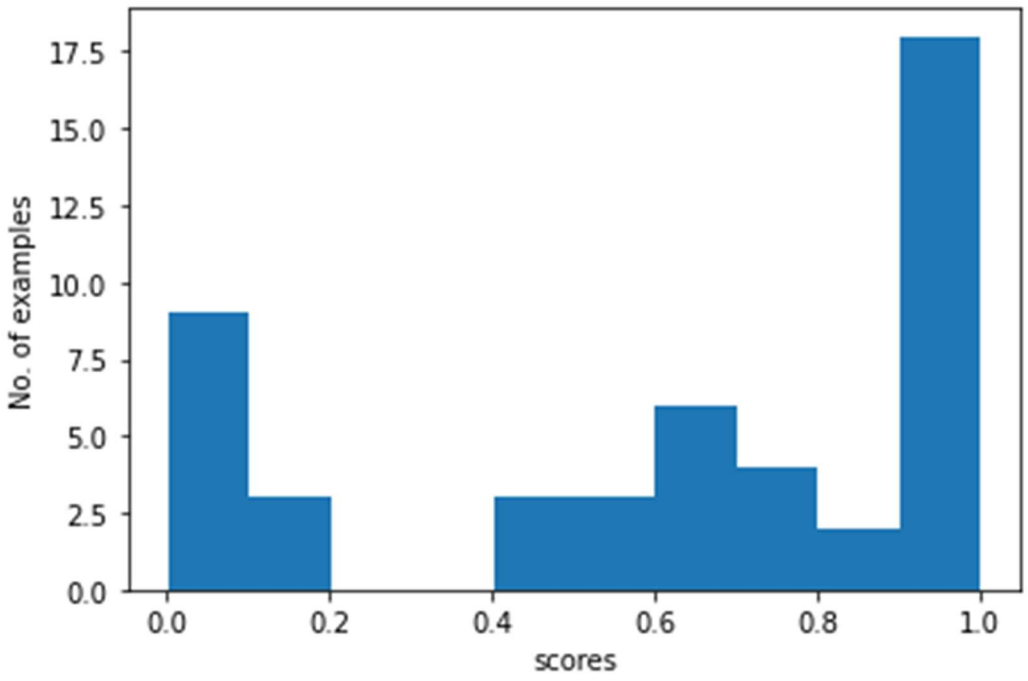
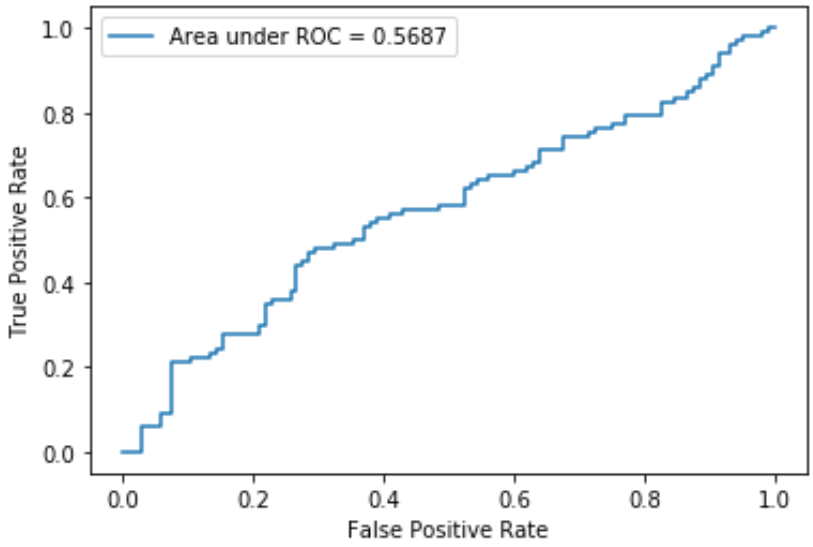


Fig. 7

To visualize the performance of model in a different angle, we'll look at the mistakes that the model has committed. The scores assigned to the misclassified examples are plotted to a histogram and the results are shown in **Fig. 7**. Since the threshold for this model is at 0.5, the scores near 0.5 implies that the model is ambiguous about these examples. The spikes near the ends 0 and 1 implies that the model is very confident in misclassifying these examples.

To further this project, I tried to define another model with pretrained VGG-16. Although, VGG-19 has more parameters than VGG-16, I still wanted to check how it may impact the accuracy. The VGG-16 transfer learning model is trained with similar parameters as used for VGG-19. The performance metrics are shown below –




```
------------------- Test Performance ---------------
-------
Accuracy          0.542
Sensitivity       0.133
Specificity       0.924
Area Under ROC    0.569
-----------------------------------------------------
```

Fig.8: Performance metrics of model trained with VGG-16

From the results of model trained with VGG-16 (**Fig.8**), it is evident that the accuracy is low at 54%. The area under ROC is 0.569 which implies that there is only a 57% chance that the model will predict the CT scans correctly. In comparison to the accuracy of RT-PCR testing sensitivity for COVID-19 which is at 60-71% (H. Bai, 2020), the model trained with VGG-16 is showing lower accuracy.

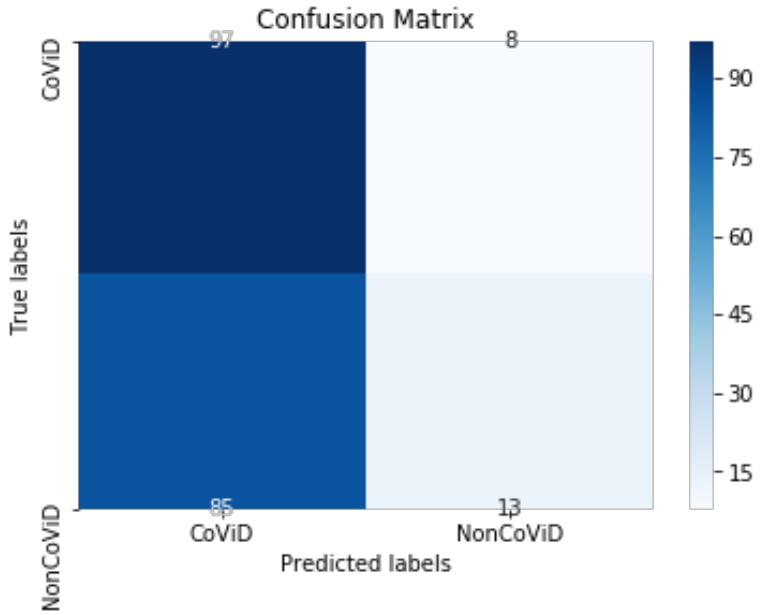


Fig.9: Confusion matrix of model trained with VGG-16

According to the confusion matrix (**Fig.9**), the true positives are at 97, true negatives are 13, false positives are 8 and false negatives are 85.

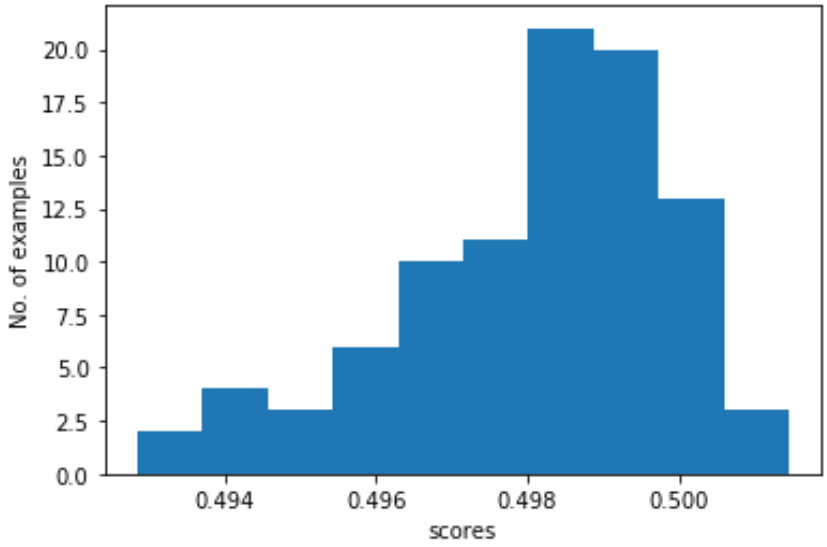


Fig.10

The scores assigned to the misclassified examples (VGG-16) are plotted to a histogram and the results are shown in **Fig. 10**. Since the threshold for this model is at 0.5, the scores near 0.5 implies that the model is ambiguous about these examples. The absence of spikes near the ends 0 and 1 implies that the model does not have any predictions that it is confident of.

Q & A

With the results/findings in hand, here are some questions and answers that I can derive from the observations of the project -

Was the objective of this project achieved?

Yes, the objective of this project is to apply deep learning algorithms to help diagnose COVID-19 by using CT scan of a patient. The pretrained VGG-19 model used to classify the COVID-19 and non-COVID-19 scans demonstrates 81% accuracy which is higher than the RT-PCR diagnostics.

Are the results of this project good enough to start implementing the algorithm for real world COVID-19 diagnosis?

While the model does demonstrate decent metrics, the dataset that the model is trained on has only 746 images. The dataset is too small for a model trained on it to be deployed in the real world.

Are there any issues with the data? Do you think the data used for this project is accurate for this type of analysis?

The images for the negative class are a mix of healthy patients and non-COVID disease CT scans. The performance could be much better if we had labels separating the healthy patients from the non-healthy non-COVID scans. This is perhaps the reason why this model mistakes non-COVID opacities for COVID ones.

Why was transfer learning chosen for this project?

Transfer learning is typically used when there is not enough data to train a model. That is the case in this scenario. There are only 746 images of CT scans available and it requires a lot more images to train a model from scratch.

Why did you choose VGG-19 and VGG-16 pretrained models?

VGG-19 has more layers of architecture compared to other pretrained models and it will result in better accuracy than models that have fewer layers like VGG-16.

Which model performs better?

The model pre trained with VGG-19 shows better performance at 81%

Concluding Remarks

There has been lot of chaos all around the world since the beginning of the pandemic. Healthcare providers, research scientists and Data science community have all come together with one focus and that is to contain and prevent the COVID19 infections. RT-PCR is one of the widely used testing method for this infection and the extensive testing reveals that there are many anomalies in the sensitivity of this method. With the cold weather approaching, experts are predicting a new wave of infections in addition to existing flu season.

The aim of this project is to find a second method of testing by using CT-scans of patients and deep learning that can serve as a 'second opinion' in cases where the RT-PCR fails. The results of this project are promising. While the model does give decent metrics, the outcome of this project will serve as a stepping stone to expand the analysis by including more data and fine tuning the parameters.

References

H. Bai, B. H. (2020). Performance of radiologists in differentiating COVID-19 from viral pneumonia on chest CT. *Radiology, 296*.

Ilker Ozsahin, B. S. (2020). Review on Diagnosis of COVID-19 from Chest CT Images Using Artificial Intelligence. *Computational and Mathematical Methods in Medicine*, 10.

Kathuria, A. (2020, april 01). *PaperspaceBlog*. Retrieved from https://blog.paperspace.com/fighting-corona-virus-with-ai-medical-imaging-testing/

Xiaowei Xu, M., Xiangao Jiang, M. C., Du, P., Li, X., Shuangzhi Lv, M., Liang Yu, M., . . . Lingxiang Ruan, M. (2020). Deep Learning System to Screen Coronavirus Disease 2019 Pneumonia . *arxiv.org*.

Yicheng Fang, H. Z. (2020). Sensitivity of Chest CT for COVID-19: Comparison to RT-PCR. *RSNA, 296*.

Zhao, J. a. (2020). COVID-CT-Dataset: a CT scan dataset about COVID-19. *arXiv preprint arXiv:2003.13865*.

Appendix A - Web resources

- The primary article that served as an inspiration for this project. The study conducted in this article aims to establish an early screening model to distinguish COVID-19 pneumonia from Influenza-A viral pneumonia and healthy cases with pulmonary CT images using deep learning techniques. --- *Deep learning system to screen coronavirus disease 2019 Pneumonia*, Xu et. al., https://arxiv.org/pdf/2002.09334.pdf
- This is another important article that I am using as a guide for this project which highlights how machine learning is applied to help fight the pandemic. --- *Fighting Coronavirus With AI, Part 1: Improving Testing with Deep Learning and Computer Vision*, Ayoosh Kathuria, April 2020. https://blog.paperspace.com/fighting-corona-virus-with-ai-medical-imaging-testing/
- This is the reference article that provides information about the dataset that I will be using for this project. This open source dataset is provided by the researchers of UC San Diego. --- *COVID-CT-Dataset: A CT Image Dataset about COVID-19*, Yang et.al., June 2020, https://arxiv.org/pdf/2003.13865.pdf
- This is the secondary reference article that I will be using as a guide for this project structure and evaluation methods --- *Fighting Coronavirus With AI, Part 2: Building a CT Scan COVID-19 Classifier Using PyTorch*, Ayoosh Kathuria, July 2020, https://blog.paperspace.com/fighting-coronavirus-with-ai-building-covid-19-classifier/
- This link directs to the github dataset that will be used for this project. https://github.com/UCSD-AI4H/COVID-CT
- For classifying/identifying CT scans that are of COVID19 infected patients, I will be looking for the 'ground glass' effect in the scans. This is a news article that talks about how scientists have identified that coronavirus CT scans have white patches referred as 'ground glass opacity' that would differentiate COVID19 scans from normal and all other diseases. https://www.businessinsider.in/science/news/china-is-diagnosing-coronavirus-patients-by-looking-for-ground-glass-in-their-lungs-take-a-look-at-the-ct-scans-/articleshow/74126644.cms
- Now that the source data is identified, I have used the following article to help me decide which deep learning library is best suitable. This article compares and contrasts the two most used deep learning libraries – *Tensorflow and PyTorch.* --- *Tensorflow or PyTorch: The force is strong with which one?*, Yashwardhan Jain, April 2018, https://medium.com/@UdacityINDIA/tensorflow-or-pytorch-the-force-is-strong-with-which-one-68226bb7dab4
- Since there is not enough data available to train a new model, I will be using transfer learning method for this project. This is the article that I am referring to for transfer learning in deep learning. --- *COVID-19 Detection Through Transfer Learning Using Multimodal Imaging Data*, M. J. Horry et al., IEEE Access, vol. 8, pp. 149808-149824, 2020. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9167243&isnumber=8948470
- Finally, I have enlisted the help of these tutorials of PyTorch to help me with the project. https://pytorch.org/tutorials/