Convolutional Neural Network Models Application: The Efficiency of Pre-trained CNN Models on COVID-19 Detection Using Lung Scans

University of Colorado at Denver

Kailun Lin

Advisors:

Erin Austin

Francis Newman

Yaning Liu

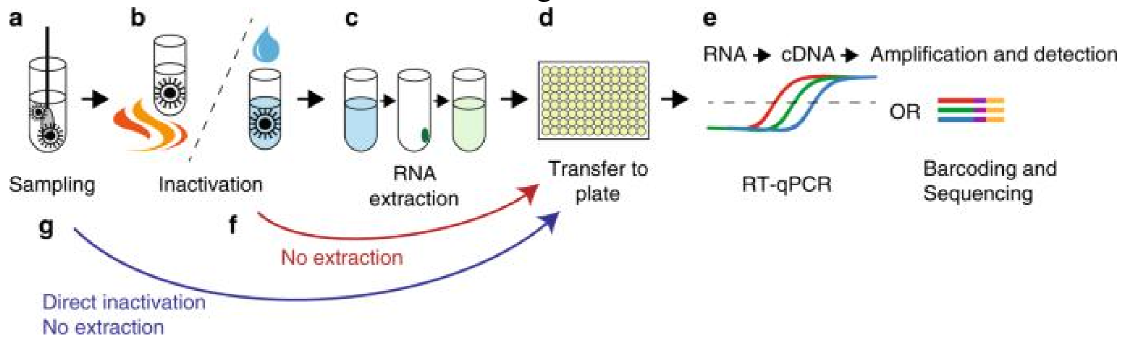
Abstract

COVID-19 (Coronavirus-2019) is one of the most serious pandemics in human history. The current most commonly used method for COVID-19 detection is called RT-PCR (Reverse Transcription–Polymerase Chain Reaction). However, the problem with a biological test such as RT-PCR is that the test is time-consuming and has high risk of misdiagnosing cases at an early stage of COVID-19. I propose using CNN (Convolutional Neural Network), an image cognition method, as an alternative method for COVID-19 detection. In the field of CNN study, CNN method has largely been used for lung cancer detection, especially for early stage lung cancer detection using lung scans. Even if lung cancer and COVID-19 have different presentation on lung scans, these two diseases still have a very similar pathway, increasing the potential successful application of transfer learning considering the difference between human vision and computer vision. Thus, CNN potentially will avoid misdiagnosing early stage COVID-19. In this project, I will use two CNN models: AlexNet, and SqueezeNet; and pre-train these models with lung cancer images and COVID-19 images separately. That is, use a pre-trained model for COVID-19 detection, which means both regular CNN and transfer learning in CNN will be used. Then based on test accuracy, precision, recall, F1-score, I compare the performance of these models. It turned out that SqueezeNet performs better than AlexNet overall and small cell carcinoma (a type of lung cancer) is more suitable for transfer learning.

Background

At the end of 2019, COVID-19 quietly spread around the world. Within months, millions of people were infected by this virus (World Health Organization, 2019). On average, about 2 people out of 100 in the population died due to this virus: COVID-19. COVID-19 is one of the most cunning and tenacious viruses in human history. Until April 1, 2022, about 489.19 million people worldwide have been infected by COVID-19, and around 6.17 million people died due to COVID-19 within 3 years (World Health Organization, 2019). Compared to other serious pandemics in human history, Black Death killed at least 25 million Europeans within 4 years (Patterson et al, 2021) and Smallpox killed up to 300 million people in a century (Henderson, 2011). With no doubt, we should take COVID-19 seriously and find a sustainable way to detect COVID-19.

Figure 1



The most commonly used method for COVID-19 detection is called RT-PCR (Reverse Transcription–Polymerase Chain Reaction), and it typically takes a few day to return the diagnosis result. Basically, RT-PCR is a biological test describes in Figure 1. RT-PCR generally has around 95% accuracy rate for COVID-19 detection (Brihn et al., 2020). However, when people are initially infected by COVID-19, they typically have a 14 to 21 day incubation period. During the incubation period, research showed RT-PCR has high risk to misdiagnose cases (Winkelhake, 2021), classifying a positive case as negative and there were many reported cases due to this same reason (Tahamtan&Ardebili, 2020).

Thus, the potential issues for RT-PCR are that it is time-consuming and likely to misdiagnose cases at early stages COVID-19. In this project, I am proposing an alternative way for COVID-19 detection: CNN (Convolutional Neural Network). This potentially would avoid these issues because Tahamtan and Ardebili used CT scans to validate the accuracy of RT-PCR tests for COVID-19 detection at early stage (2020). If human vision can distinguish the normal lung scan and COVID lung scan at early stage, then I hypothesize that computer vision is capable of doing the same thing, and the overall accuracy of CNN should be very close to, and potentially higher than RT-PCR without retesting data to return test results.

Introduction

CNN (Convolutional Neural Network) is an image cognition method that can classify images with different features. For instance, CNN can distinguish dogs and cats just by training with a set of dog images and cat images. Training is a process for CNN models to learn the features from representative training dataset. In the past decade, CNN has been used in lung cancer detection, and there are more than 10 thousand studies containing the key words "CNN" and "lung cancer". According to work by Dr. Thomas, lung cancer and COVID-19 have similar pathway (2021). In terms of COVID-19 detection, CNN are rarely implemented. Considering the current pandemic era, finding sufficient COVID-19 lung scans to train CNN models is simple and the major condition for CNN implementation is having sufficient data. Thus, CNN could be used to diagnose COVID-19 cases by analyzing of training with lung scans.

In this project, I will use both transfer learning in CNN and regular CNN for COVID-19 detection. Regular CNN for COVID-19 detection, as mentioned before, is the method that uses a set of COVID lung scans and normal lung scans to pre-train CNN models, and then uses the pre-trained CNN models to classify the unknown lung scans as a COVID lung or a normal lung. Transfer learning is a method that uses the pre-trained CNN models in the issues similar to what we are studying and then applies the pre-trained CNN models into the issue that we are interested in.

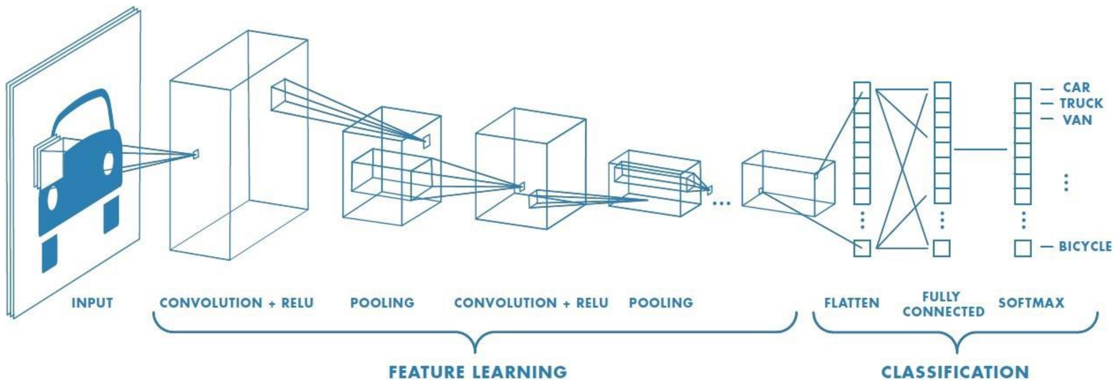
In this project, I will first use regular CNN for COVID-19 detection and lung cancer detection to validate my approach and code. Next, I will pre-train the CNN models with a set of cancerous lung scans and normal lung scans. At the same time, I will pre-train the CNN models in another set of lung scans that contains normal lung and COVID lung. I will then use these two sets of pre-trained CNN models to classify whether a different, testing set of the unknown lung scans are COVID lung or normal lung and compare the statistical results to check the performance of two sets of pre-trained CNN models.

The point of doing lung cancer pre-trained CNN models is to check whether the lung diseases with high similarity would have analogous presentation on CNN analysis. If so, we can anticipate statistical results such as the accuracy of these two sets of pre-trained CNN models are similar. More importantly, if we encounter an unknown lung disease whose mechanism is similar to lung cancer or COVID-19, we will have shown that it is possible to use transfer learning for detection of the unknown disease by running the lung cancer pre-trained CNN models or COVID-19 pre-trained CNN models. This would save a lot of time and help control the disease before it breaks out.

Methodology

Generally, CNN models have an input layer, hidden layers, and an output layer. Recall the goal of using these different layer is classifying images. An example is given in Figure 2. The input layer is the car image; the hidden layers are the feature learning part; the output layer is the classification part. There are more details in each layer that are explained next.

Figure 2

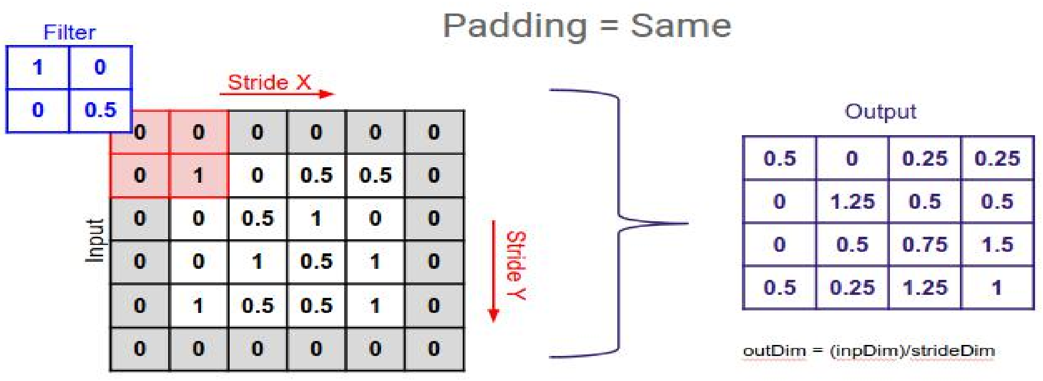


The input layer is the input image to be classified by CNN models. The size of input image usually would be four dimensions (numbers of input image× height of the input image (measured in pixels) × width of the input image (measured in pixels) × channels of input image, where the channels of input image typically refer to RGB channels used for representing colors of the image). Each image contains at least thousands of pixels and the range of each single pixel is in [0, 255]. The input layer normalizes the value of each pixel to the range of [0, 1] so the cost of further calculation would be less expensive.

In addition, the input layer divides the input image into several smaller volumes so the feature learning process is more accurate and more efficient compared to directly analyzing the whole image. Last, the input layer will pass the normalized and partitioned input image to the next layer.

The hidden layers consist of (multiple) convolutional layers, and pooling layers.
A convolutional layer typically is the layer right after either the input layer or the pooling layer. It convolves the input volume, which is the output volume passed from the last layer, and then passes the output volume to the next layer. The detailed mechanism of convolving is in Figure 3.

Figure 3

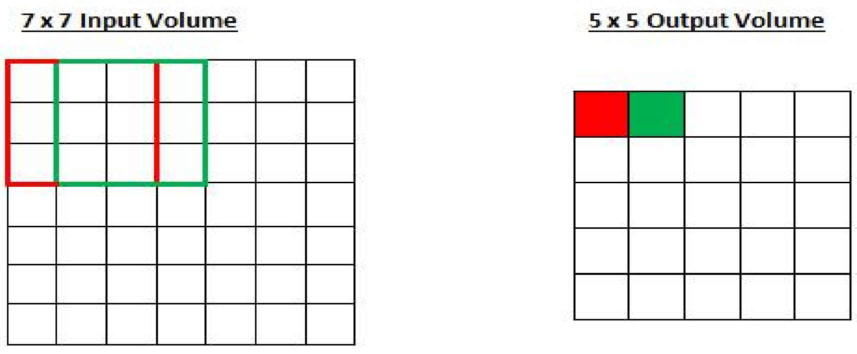


The convolving process will have an input volume, a filter, a determined value of stride, and a padding. The input volume simply is the output volume passed from the last layer. An example is given in Figure 3. The filter, in Figure 3, is a 2 × 2 kernel representing one feature extracted from pre-training images. Stride is the number of pixels the filter will move along the axis at a time. Padding basically increases the size of the input volume, which is the shaded pixels with values of zero. The function of padding is to guarantee all the values in the original input volume will be scanned by the filter for the same amount of time so the weights will be the same. The calculation of convolving, looking at the filter and the red shaded part in Figure 3, is simply taking the product of value from the filter and value from the red shaded part in the same positions and then summing the products such that $1 \times 0 + 0 \times 0 + 0 \times 0 + 0.5 \times 1 = 0.5$, where 0.5 is the value in the left upper corner of the output volume.

Generally, a convolutional layer with padding will keep the size of output volume the same as the original input volume. After convolving the input volume into output volume, the convolutional layer will either use a Sigmoid function or ReLU function to manipulate the values in the output volume depending on different CNN models and then pass the adjusted output volume to the next layer. The Sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$, is a normalization function where $f(x) \in [0, 1]$. In Figure 3, some of the output values can still be greater than one even if all the values in the filter and input volume have been normalized. The goal of using the Sigmoid function is using 0 and 1 to represent the likelihood of the output volume that has the same features as the filter. If the value is close or equal to one, then it has high chance of having the same features with the filter. If the value is close or equal to zero, then it is unlikely to have the same features as the filter. The ReLU function, $f(x^{+}) = \max(0, x^{+}), f(x^{-}) = 0$, uses a similar idea as a Sigmoid function but has fewer vanishing gradient issues. The graph of the Sigmoid function and the ReLU function are in Appendix 1 and Appendix 2.

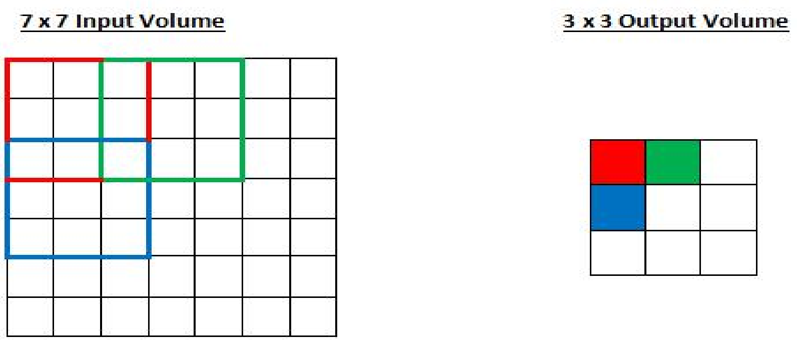
Pooling layer typically is the layer right after convolutional layer and it aims to pass a condensed extracted feature volume to the next layer. This means the size of output volume would be decreased by not using padding and by increasing stride. More details are shown in the following Figure 4 and Figure 5.

Figure 4



Generally, there are two ways of pooling: max pooling and average pooling. In Figure 4, the filter is a 3 × 3 kernel using to find either maximum value or mean value for each one of the 3 × 3 area (such as red frame or green frame in Figure 4) and then pass the values into the output volume. Looking at 3 × 3 red frame in the input volume, if the pooling layer performs max pooling, it will find the maximum value out of these nine values and fill the maximum into the red shaded pixel in the output volume; If the pooling layer performs average pooling, then it will calculate the mean of these nine values and fill the mean into the red shaded pixel in the output volume.

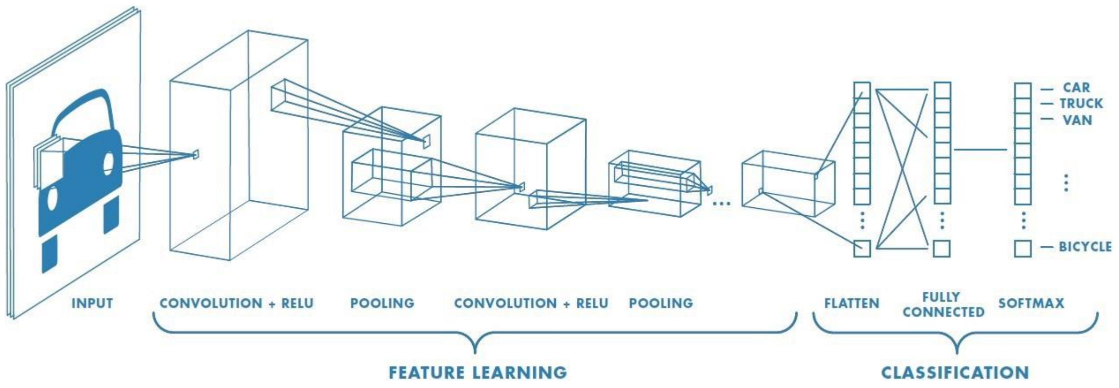
Figure 5



In Figure 5, if pooling layer sets stride to be two, then the size of the output volume will be 3 × 3 instead of 5 × 5.

As the example showed in Figure 2, so far, we have already known that CNN models will divide the input image into multiple smaller volumes and extract the features of these partitioned image. That is how input layer and hidden layers work in a CNN model. Note that after pooling layer, if the size of output volume is still too large or the learnable parameters (refer to the features of image) are too many, then the hidden layers can have more convolutional layers and pooling layers to perform the convolving process and pooling process again. Eventually, the hidden layers will pass the output volume to the next layer.

Figure 2



The output layer will flatten the input volume, which is the output volume passed from the last layer, into the vector-like form, and then use a fully connected layer connect with the flattened input volume. A fully connected layer is a cheap way to represent the non-linear correlation of each feature, and the features in the image typically are non-linearly correlated. Consider mapping in mathematics, a fully connected layer works in a similar way. Then comparing the flatten results (extracted features from input image) and the results (extracted features from pre-training process) stored in fully connected layer, determine which class the input image belongs to based on how close these two results are.

This is the how generally CNN model works but the exact architecture can vary. In this project, I choose two representative CNN models: AlexNet, and SqueezeNet.

Figure 6

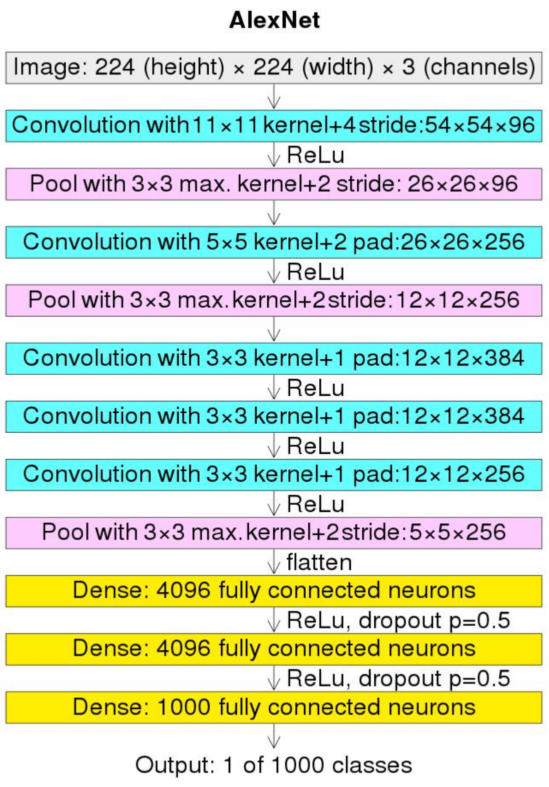

AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max.kernel+2stride:12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max.kernel+2stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

Figure 6 shows the architectures of AlexNet, which contains all the layers mentioned earlier. AlexNet is a typical and a fairly new model released in 2012, and it can deal with colored image because it contains three channels (typically refer to RGB channel).

Figure 7

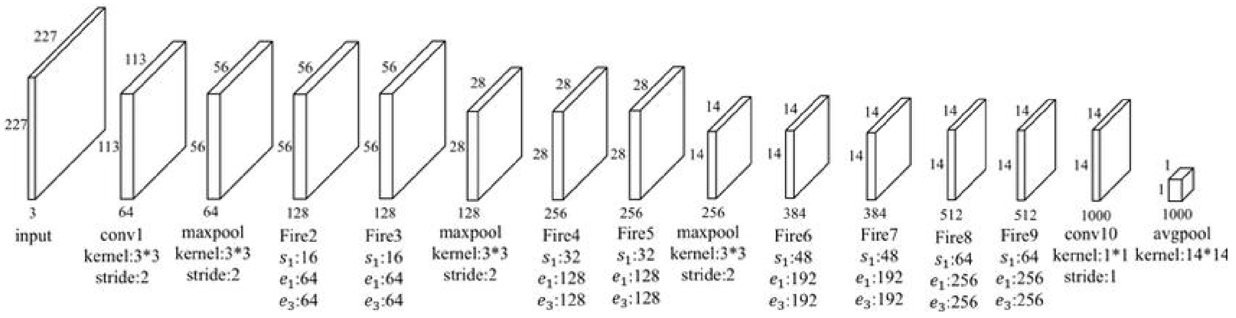

227
227
3 input
113
113
64 conv1 kernel:3*3 stride:2
56
56
64 maxpool kernel:3*3 stride:2
56
56
128 Fire2 s_1:16 e_1:64 e_3:64
56
56
128 Fire3 s_1:16 e_1:64 e_3:64
28
28
128 maxpool kernel:3*3 stride:2
28
28
256 Fire4 s_1:32 e_1:128 e_3:128
28
28
256 Fire5 s_1:32 e_1:128 e_3:128
14
14
256 maxpool kernel:3*3 stride:2
14
14
384 Fire6 s_1:48 e_1:192 e_3:192
14
14
384 Fire7 s_1:48 e_1:192 e_3:192
14
14
512 Fire8 s_1:64 e_1:256 e_3:256
14
14
512 Fire9 s_1:64 e_1:256 e_3:256
14
14
1000 conv10 kernel:1*1 stride:1
1
1
1000 avgpool kernel:14*14

Figure 7 shows the architecture of SqueezeNet, whose structure is slightly different from a typical CNN model because it contains fire layers. The function of a fire layer is to squeeze the convolutional filters into a smaller filter and then expand the filters. As a result, the learnable parameters will be significantly decreased compared with not using a fire layer. Again, SqueezeNet is a fairly new CNN model released in 2016.

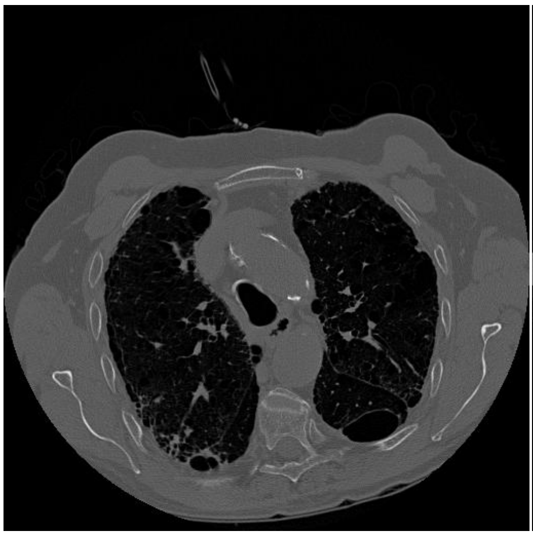
As mentioned in the introduction section, I will do both transfer learning in CNN and regular CNN for COVID-19 detection. Note that CNN models can only classify images into certain classes that were set up in the pre-training process. Using transfer learning in this project needs to be more careful because the class names for cancerous class and COVID class are different, and CNN models cannot classify a cancerous lung scan into COVID class (it will report bug in coding). When using these pre-trained CNN models for COVID-19 detection, they cannot classify COVID lung scans as easy as regular CNN because the presentation of a COVID lung and a cancerous lung in lung scans is quite different (More details about differences between COVID lung and cancerous lung are in the Data and Data Visualization section). To account for this, I will change the class name from cancerous class to abnormal class. Even if the cancerous class may not fit COVID lung scans, compared with normal class, the cancerous class is a contrast class that tells how different the abnormal lung scans and normal lung scans are. The COVID lung cannot fit the normal class because the features of COVID lung scans are distinguishable from a normal lung, and a cancerous lung cannot fit a normal class for the same reason. Thus, COVID lung scans have a high probability to be classified into abnormal class rather than normal class.

Changing the class name will not affect the result of classification but it affects the strictness. Yet the problem of doing that is that we cannot know whether the abnormal lung is cancerous lung or COVID lung unless we know which dataset the lung scans belong to. In reality, when we implement transfer learning for COVID-19 detection, we can only tell if the abnormal lung is either a cancerous lung or a COVID lung or both cancerous and COVID, but we cannot specifically know which type or types of lung diseases the abnormal lung has. Transfer learning in CNN, compared with regular CNN, generally has higher costs. To restate this project's goals, the reason I am doing transfer learning is to verify the efficiency of transfer learning. If the efficiency of transfer learning is relatively high, such as not taking too much extra time than using regular CNN, then we can use transfer learning to save significant amount of time when we cannot perform regular CNN, for instance, when a new kind of lung disease similar to lung cancer or COVID-19 emerges but we don't have enough lung disease scans to do regular CNN.

Data and Data Visualization

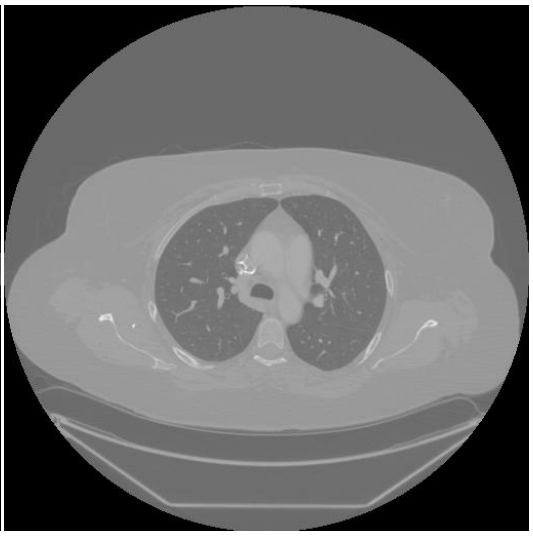
In this project, I have total three datasets. The first dataset is a COVID dataset from Kaggle, which contains 7481 COVID lung scans and 933 normal lung scans. The second dataset is a COVID dataset from Dr. Francis, which contains 106 COVID lung scans and 1551 normal lung scans. Note that Dr. Francis actually provided me a total of 1660 COVID lung scans, but all the COVID data I have contains more COVID lung scans than normal lung scans. For the sake of balancing the data sources, I only use 106 out of 1660 COVID lung scans from Dr. Francis in my project. The last dataset is a cancer dataset from Kaggle, which contains 785 cancerous lung scans (three types) and 215 normal lung scans. The three types of lung cancer are adenocarcinoma (338), large cell carcinoma (187), and squamous cell carcinoma (260).

Figure 8



(COVID Lung Scan)

Figure 9



(Normal Lung Scan)

To understand exactly how CNN works in a computer, we need to first know that the computer vision is different from human vision. Figure 8 is a COVID lung scan, and Figure 9 is a normal lung scan. From human vision, people can tell the difference between COVID and normal lungs just by looking at features of the images. There is a thicker wall along two lobes for the COVID lung compared with normal lung. For computer vision, it will transfer the whole image into numbers, perform a series of calculations, and extract the key features from pre-training process. It will then compare those features with the extracted features from the input image and classify the image to the class with the closest features. In regular CNN, even if human vision and computer vision work differently, the classification results would be similar. That is, it would classify Figure 8 as COVID class and classify Figure 9 normal class. However, the transfer learning in CNN would work differently in this case.

Examine the following Figure 10 and Figure 9. Figure 10 is the cancerous lung scan with a specific lung cancer type called large cell carcinoma, and Figure 9 is the same normal lung scan. There are several white in two lobes for large cell carcinoma scan. For human vision, we can easily tell the differences between Figure 9 and Figure 10, as well as the differences between Figure 9 and Figure 8. For computer vision, if computer only learned the features of cancerous lung scans and normal lung scans, then it can only classify these two classes, rather than COVID and normal lung scans. This means in transfer learning a computer can only tell the differences between Figure 10 and Figure 9. In other words, computer vision can tell whether the image belongs to the normal class or not. Recall I will change cancer class and COVID class into abnormal class for transfer learning. Although changing the name of class would not return the specific type of lung disease if the result is abnormal, the project goal is to evaluate whether the patient has lung disease or not. In practice we could perform further tests to determine which type of lung disease the patient has.

Figure 10

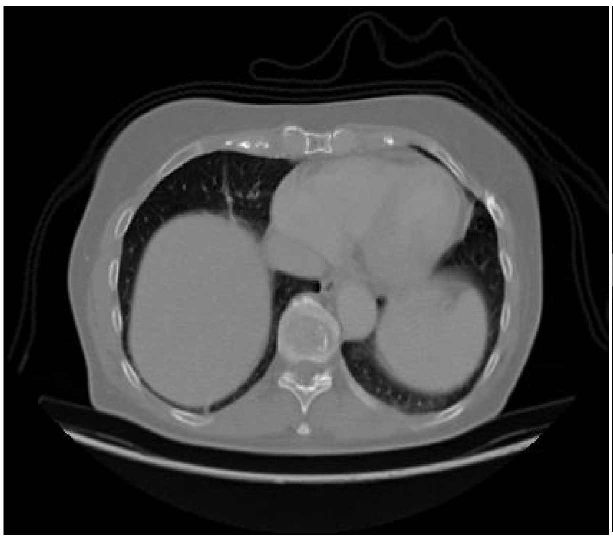
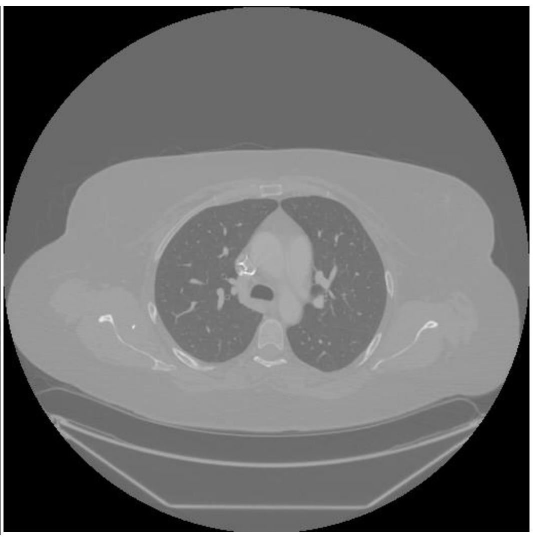

(Cancerous Lung Scan: large cell carcinoma)

Figure 9


(Normal Lung Scan)

Coding

The coding for this project has uploaded into GitHub and the link is: https://github.com/Kailun322/Final-Project

Before officially running the code, there are several parameters for CNN that need to be fixed to improve performance. Note that there are three parameters: batch size, lr (learning rate), and epoch. They are changeable, affecting the result. For batch size, typically it will chosen from the range [1, number of images]. The smaller batch size you choose, the time each epoch takes will be longer. I just chose 32 as batch size initially, where 32 is a typical number for batch size similar to 32 for random seeds. Dr. Bengio's work also verified a batch size of

32 is a good default value and is typically chosen between 1 and a few hundred (2012). However, a batch size of 32 did not work well for both AlexNet and SqueezeNet. After systematic experimenting different values of batch size such as 2, 4, 8, and 16, I finally fix a batch size of 4 for AlexNet and a batch size of 16 for SqueezeNet. According to the researchers (Masters&Luschi, 2018), the best results will occur by choosing batch size of 32 or a smaller batch size, such as 2 and 4.

For the selection of lr (learning rate), I chose 0.001. According to Bishop (1996), he mentioned "The choice of the value for the learning rate can be fairly critical, since if it is too small the reduction in error will be very slow, while, if it is too large, divergent oscillations can result" (p.95). In addition, other articles recommended choosing learning rate in a logarithmic scale such as 0.1, 0.01, 0.001, 0.0001, and 0.00001 (Goodfellow et al., 2016).

Epoch is the number of training processes the computer will go through. When the epoch is small, the trained model will have underfitting issue. When the epoch is large, the trained model will have overfitting issue. Both underfitting and overfitting will lower the test accuracy. In this project, choosing 50 as epoch is enough because when the train accuracy approaches 100%, the test accuracy will peak at some value and then start falling, and that is where the overfitting issue appears. The peak value of test accuracy is the best epoch for training the model, which in this case will be reached before 50 epochs, but the best epoch could be changeable. Thus, 50 epoch will likely be the smallest epoch that will include the best epoch in this case. Note that I don't have early stop in my project because I want to see the complete trend of train-test accuracy curve as epochs increase.

Recall that what I will do in the coding part is apply the regular CNN for COVID-19 detection and lung cancer detection to ensure the parameters I have fixed is efficient. Next, I start using transfer learning in CNN for COVID-19 detection; that is, train CNN models with cancerous and normal lung scans and use the pre-trained CNN models to classify COVID lung and normal lung. Last, I evaluate the performance of CNN models in transfer learning.

Since I have sufficient data in the COVID-19 dataset, I will also investigate whether good performance of CNN models is due to the adjustment of parameters or to coincidence of data combination. So, I created four different combination of training and testing datasets, and each of the datasets will be used for regular CNN. For each combination of a COVID-19 dataset, the size of the training set is 5000, which contains 3000 abnormal (COVID lung) scans and 2000 normal lung scans, and the size of testing set is 500, which contains 250 abnormal (COVID lung) scans and 250 normal lung scans.

Because the size of lung cancer dataset is really small comparing with the COVID-19 dataset, and I will not create different combinations of lung cancer scans. Since there are three types of lung cancer and one normal class, there will be four classes for the lung cancer dataset. Recall that the goal of this project is to distinguish the difference between normal and abnormal lung. That is, use pre-trained CNN models to determine whether a lung is healthy or has at least one of these diseases (three types of lung cancer, COVID-19). Therefore, I split each type of lung cancer from the lung cancer dataset, and I created three binary classification sets for lung cancer detection instead.

Based on the three lung cancer types (adenocarcinoma, large cell carcinoma, small cell carcinoma), I split the lung cancer data into three subsets. For adenocarcinoma data, the size of the training set is 343, which contains 195 abnormal (adenocarcinoma) scans and 148 normal lung scans, and the size of the testing set is 196, which contains 136 abnormal (adenocarcinoma) scans and 60 normal lung scans. For the large cell carcinoma data, the size of the training set is 263, which contains 115 abnormal (large cell carcinoma) scans and 148 normal lung scans, and the size of the testing set is 118, which contains 58 abnormal (large cell carcinoma) scans and 60 normal lung scans. For the small cell carcinoma data, the size of the training set is 303, which contains 155 abnormal (small cell carcinoma) scans and 148 normal lung scans, and the size of the testing set is 160, which contains 100 abnormal (small cell carcinoma) scans and 60 normal lung scans.

Similarly, in the transfer learning process, I will use three sets of lung cancer datasets (one type of lung cancer for each) as training set, but the testing set will include COVID lung scans rather than cancerous lung scans. For adenocarcinoma transfer learning, the size of the training set is 474, which contains 326 abnormal (adenocarcinoma) scans and 148 normal lung scans, and the size of the testing set is 120, which contains 60 abnormal (COVID lung) scans and 60 normal lung scans. For large cell carcinoma transfer learning, the size of the training set is 311, which contains 163 abnormal (large cell carcinoma) scans and 148 normal lung scans, and the size of the testing set is 120, which contains 60 abnormal (COVID lung) scans and 60 normal lung scans. For small cell carcinoma, the size of the training set is 400, which contains 252 abnormal (small cell carcinoma) scans and 148 normal lung scans, and the size of the testing set is 120, which contains 60 abnormal (COVID lung) scans and 60 normal lung scans.

Table 1. Summary of All Datasets Info

Type of CNN	Type of Images in the Training Set	Type of Images in the Testing Set	Size of Dataset	Size of the Training Set	Size of the Testing set
Regular CNN	COVID (3000) Normal (2000)	COVID (250) Normal (250)	5500	5000	500
Regular CNN	COVID (3000) Normal (2000)	COVID (250) Normal (250)	5500	5000	500
Regular CNN	COVID (3000) Normal (2000)	COVID (250) Normal (250)	5500	5000	500

Regular CNN	COVID (3000) Normal (2000)	COVID (250) Normal (250)	5500	5000	500
Regular CNN	Adenocarcinoma (195) Normal (148)	Adenocarcinoma (136) Normal (60)	539	343	196
Regular CNN	Large Cell Carcinoma (115) Normal (148)	Large Cell Carcinoma (58) Normal (60)	381	263	118
Regular CNN	Small Cell Carcinoma (155) Normal (148)	Small Cell Carcinoma (100) Normal (60)	463	303	160
Transfer Learning in CNN	Adenocarcinoma (326) Normal (148)	COVID (60) Normal (60)	594	474	120
Transfer Learning in CNN	Large Cell Carcinoma (163) Normal (148)	COVID (60) Normal (60)	431	311	120
Transfer Learning in CNN	Small Cell Carcinoma (252) Normal (148)	COVID (60) Normal (60)	520	400	120

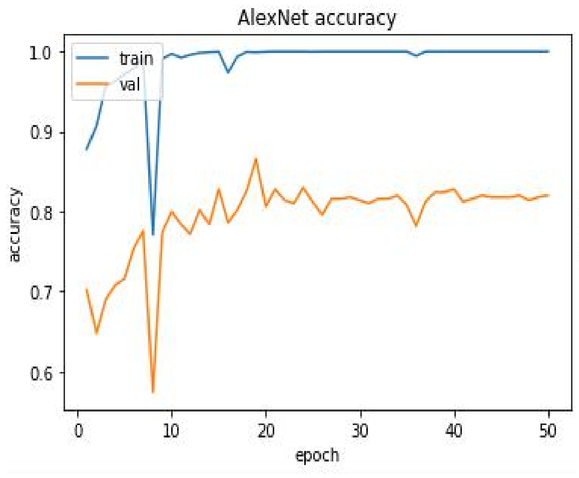
This also raises a question because I basically have three datasets for COVID-19 (two from Dr. Francis and one from Kaggle) and these three datasets are from different sources. Different sources mean that the lung scans are photoed slightly differently by different machines even if the images are all called lung scans. Here is the question: whether the different source issue will influence the performance of the CNN models? The answer is yes. I will show more details in results and image visualizations in the following section to explain the reason.

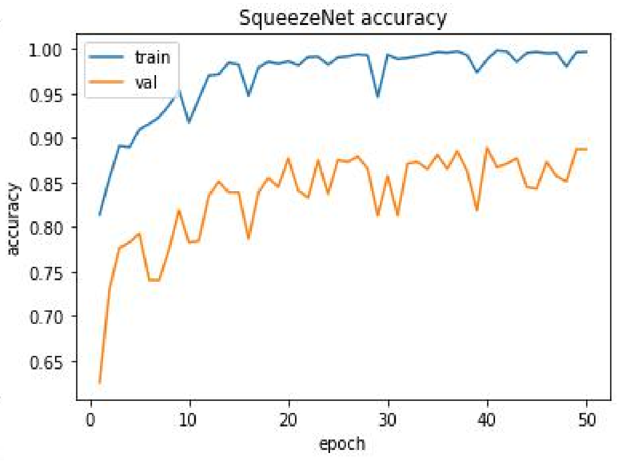
Results

To begin with this section, I will first have three analogous paragraphs and each paragraph following with a graph will explain the statistical results for training and testing two models in terms of one set of COVID data. However, none of these three set of COVID data is well-distributed. That is, none of these three set of COVID data are created by evenly mixing the COVID data from three different sources, and you will see the final performance is "abnormal" in the following three paragraphs. More details will be included in the following 4th paragraph.

The following results have showed the training and testing accuracy of two models (AlexNet and SqueezeNet) for the first set of COVID data as epoch is increasing. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy (the judgment of whether convergence occurs is when the loss for the specific epoch equals to or approaches 0) occurred around the 20th epoch, the highest test accuracy is around 85%, and each epoch takes approximate 220 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around the 50th epoch, the highest test accuracy is around 88%, and each epoch takes approximate 180 seconds. Since the selection of batch size for these two models is different, which means AlexNet and SqueezeNet will go over different amounts of batches (in this case, AlexNet will go over three times more batches than SqueezeNet), the comparison of time execution is unfair for AlexNet. Nevertheless, SqueezNet still performs better than AlexNet in terms of time execution and test accuracy. AlexNet takes less epoch to converge (based on the loss for each epoch) and reaches the highest test accuracy. For a simple calculation, AlexNet will take approximate 20 epochs to converge, which will take total around 1.2 hours where SqueezeNet will take approximate 50 epochs to converge, which will take total around 2.5 hours, yet SqueezeNet has the higher test accuracy.

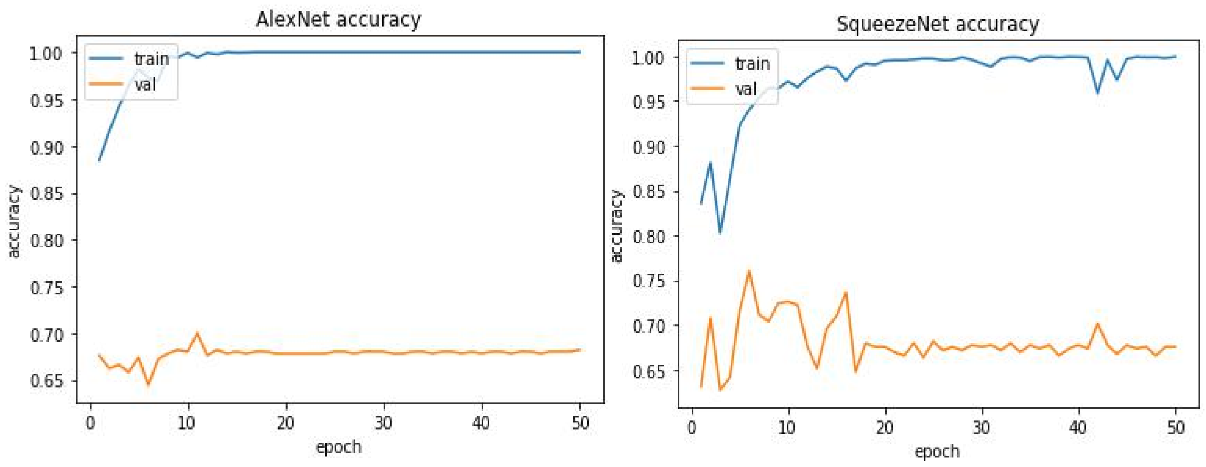
Regular CNN: training (train) and testing (val) accuracy of two models
for the first set of COVID data





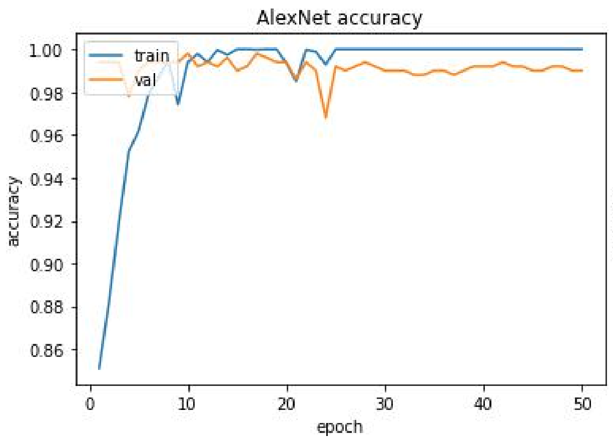
The following results have showed the training and testing accuracy of two models for the second set of COVID data as epoch increases. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around the 10th epoch, the highest test accuracy is around 67%, and each epoch takes approximate 223 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around the 20th epoch, the highest test accuracy is around 67%, and each epoch takes approximate 186 seconds. For a simple calculation, AlexNet will take approximate 10 epochs to converge, which will take total around 0.6 hours where SqueezeNet will take approximate 20 epochs to converge, which will take total around 1 hour.

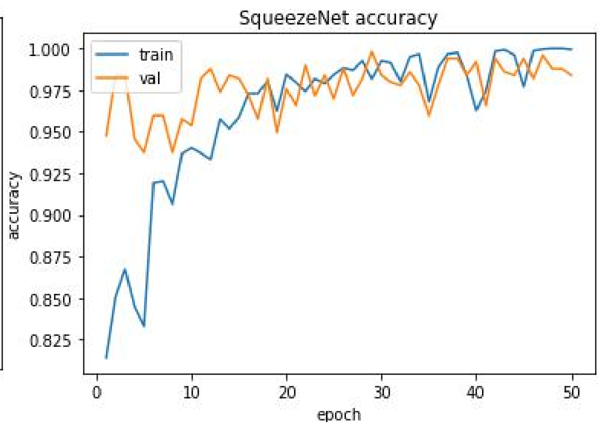
Regular CNN: training (train) and testing (val) accuracy of two models for the second set of COVID data



The following results have showed the training and testing accuracy of two models for the third set of COVID data as epoch increases. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around the 15th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 222 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around the 50th epoch, the highest test accuracy is around 98%, and each epoch takes approximate 188 seconds. For a simple calculation, AlexNet will take approximate 15 epochs to converge, which will take total around 0.9 hours where SqueezeNet will take approximate 50 epochs to converge, which will take total around 2.6 hours.

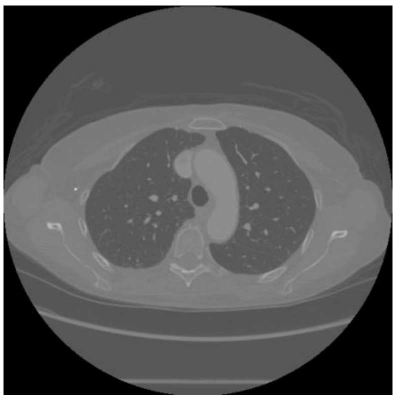
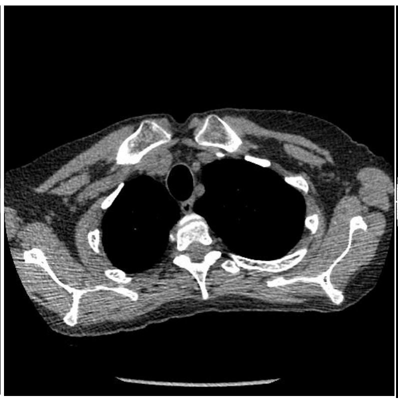
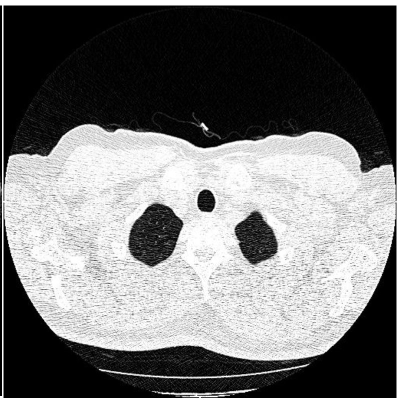
Regular CNN: training (train) and testing (val) accuracy of two models for the third set of COVID data





Recall that I have three COVID data from different sources and the following images are three images from different sources. All three use images of normal lung scans, but from different sources. As the results, they look very different, even if they are all normal lung scans. For the COVID class, the case is the same. The problem for the first three set of data is I didn't evenly mix up the images from three sources in each set of training and testing data, but the situations are slightly different. For the first set of data, I use source A, source B, and source C data for training, but I only use Source C data for testing, which means source A and source B data used for training will slightly lower the test accuracy (the test accuracy is still above 80%) for source C data detection. For the second set of data, I use source A and source B data for training and use source C data for testing, meaning CNN models cannot distinguish the difference between COVID lung and normal lung in terms of source C data by learning from source A and source B data. Therefore, the results of the second dataset are around 70%, which is not a good performance. For the third set of data, I use source A, source B, and source C data for both of COVID class and normal class in the training set, but I only include source B data in COVID class and only source C data in normal class for the testing set. The test accuracy is approaching around 100% even before training accuracy converges, which means CNN models probably classify the image correctly based on the different sources of image rather than based on the features of COVID and normal lung in this case.

Figure 11 Figure 12 Figure 13

(Source A: Normal Lung Scan)(Source B: Normal Lung Scan)(Source C: Normal Lung Scan)

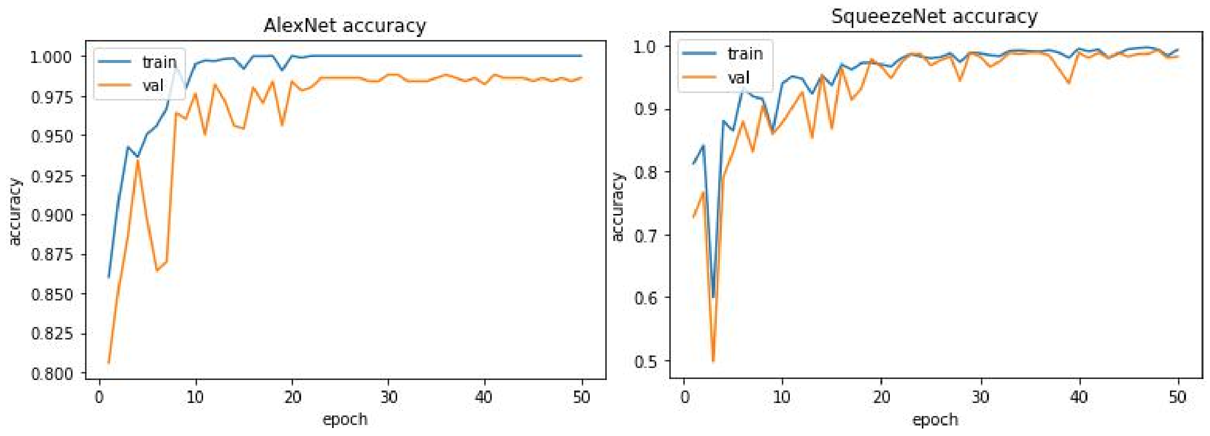
To summarize the results so far, the source of data will influence CNN models' performance if and only if the data from different sources look very different. From the first three set of data I used for training and testing, I generalize three assumptions for the "abnormal" performance:1. If you have more than one sources of data, using all sources of data for training and only one source of data for testing will slightly lower the performance (in this project, the test accuracy is lowered by around 15%); 2. If you have more than one sources of data, using partial sources of data for training and the rest sources of data for testing will totally mess up with the performance and the CNN models cannot distinguish the difference between/among classes in testing set by learning from the training set; 3. If you have more than one sources of data, using all sources of data for training, using partial sources of data for one class and the rest sources of data for the other class will result an unfairly high performance because the CNN models classify the image correctly based on the source of data rather than the features of classes.

Table 2. Summary of the First three COVID Datasets Info

COVID Dataset No.	Sources of Images Included in the Training set	Sources of Images Included in the testing set
First (No. 1)	Source A, Source B, and Source C for Abnormal Class Source A, Source B, and Source C for Normal Class	Source C for Abnormal Class Source C for Normal Class
Second (No. 2)	Source A, and Source B for Abnormal Class Source A, and Source B for Normal Class	Source C for Abnormal Class Source C for Normal Class
Third (No. 3)	Source A, Source B, and Source C for Abnormal Class Source A, Source B, and Source C for Normal Class	Source B for Abnormal Class Source C for Normal Class

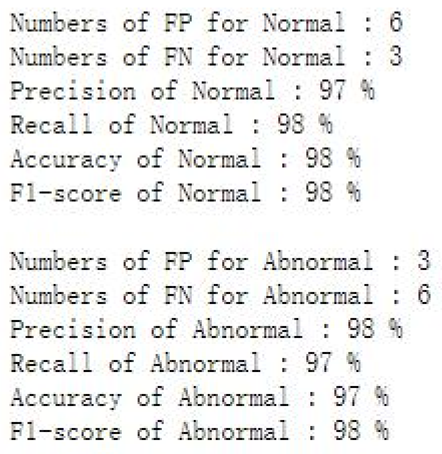
Therefore, to have a fair result for both of CNN models. I created the fourth set of data by evenly and randomly mixing up the data from three different sources for both classes, and for both training and testing set. The following graph shows the results. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 98%, and each epoch takes approximate 222 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 50th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 188 seconds. For a simple calculation, AlexNet will take approximate 20 epochs to converge, which will take total around 1.2 hours to have a 98% of test accuracy where SqueezeNet will take approximate 50 epochs to converge, which will take total around 2.6 hours to have a 99% of test accuracy. The test accuracy for these two models are similar yet AlexNet take less time to converge. Let's look at more detail of performance of these two models by looking at its partial confusion metrics.

Regular CNN: training (train) and testing (val) accuracy of two models
for the fourth set of COVID data



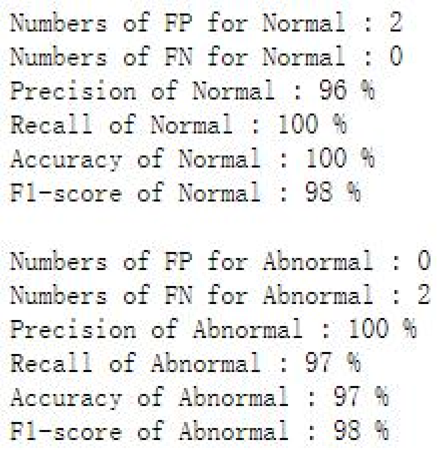
The overall results such as precision, recall, accuracy, and F1-score for both AlexNet and SqueezeNet are similar so let's focus on FP and FN (false positive and false negative). FP for normal class means the image denotes a normal lung where the image is classified as COVID lung. FN for normal class means the image denotes a COVID lung where the image is classified as normal. In reality, the misdiagnosis for FP cases is more acceptable than FN cases because people who are not infected by COVID are asked to have a quarantine will not have negative influence on society. However, if people who are infected by COVID but diagnosis as normal will not have a quarantine to protect themselves and others, and the negative impact will be significant. Considering the numbers of FP and FN cases, SqueezeNet has fewer FN and FP cases than AlexNet even if SqueezeNet takes more time to converge. Therefore, SqueezeNet seems to be a better choice over AlexNet.

AlexNet SqueezeNet



```
Numbers of FP for Normal : 6
Numbers of FN for Normal : 3
Precision of Normal : 97 %
Recall of Normal : 98 %
Accuracy of Normal : 98 %
F1-score of Normal : 98 %

Numbers of FP for Abnormal : 3
Numbers of FN for Abnormal : 6
Precision of Abnormal : 98 %
Recall of Abnormal : 97 %
Accuracy of Abnormal : 97 %
F1-score of Abnormal : 98 %
```



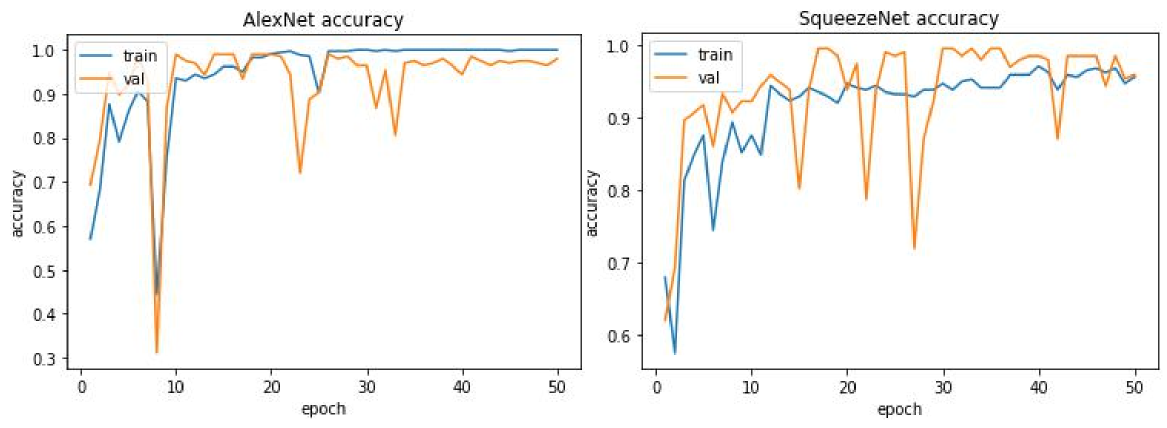
```
Numbers of FP for Normal : 2
Numbers of FN for Normal : 0
Precision of Normal : 96 %
Recall of Normal : 100 %
Accuracy of Normal : 100 %
F1-score of Normal : 98 %

Numbers of FP for Abnormal : 0
Numbers of FN for Abnormal : 2
Precision of Abnormal : 100 %
Recall of Abnormal : 97 %
Accuracy of Abnormal : 97 %
F1-score of Abnormal : 98 %
```

Before doing transfer learning in CNN, I will also perform regular CNN for lung cancer detection first in order to have a contrast result for transfer learning. Then compare with the results, we can determine whether transfer learning process is efficient. Since there are three types of lung cancer (adenocarcinoma, large cell carcinoma, small cell carcinoma), I will have three paragraphs, and show the training and testing results for each type of lung cancer.

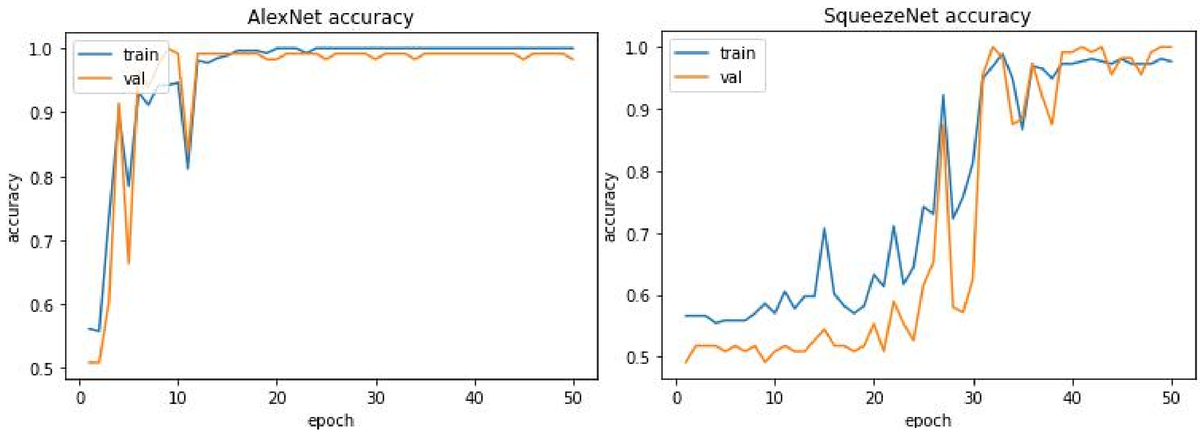
The following results have showed the training and testing accuracy of two models for adenocarcinoma data as epoch increasing. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 25th epoch, the highest test accuracy is around 97%, and each epoch takes approximate 15 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 98%, and each epoch takes approximate 10 seconds. For a simple calculation, AlexNet will take approximate 25 epochs to converge, which will take total around 6.2 minutes where SqueezeNet will take approximate 20 epochs to converge, which will take total around 3.3 minutes.

Regular CNN: training (train) and testing (val) accuracy of two models for adenocarcinoma data



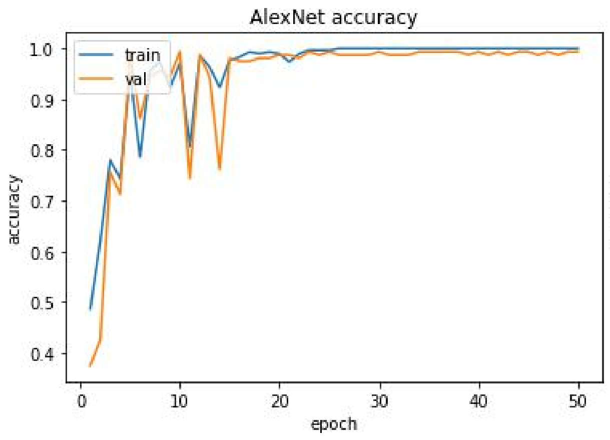
The following results have showed the training and testing accuracy of two models for large cell carcinoma data as epoch increasing. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 15th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 15 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 40th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 10 seconds. For a simple calculation, AlexNet will take approximate 15 epochs to converge, which will take total around 3.75 minutes where SqueezeNet will take approximate 40 epochs to converge, which will take total around 6.6 minutes.

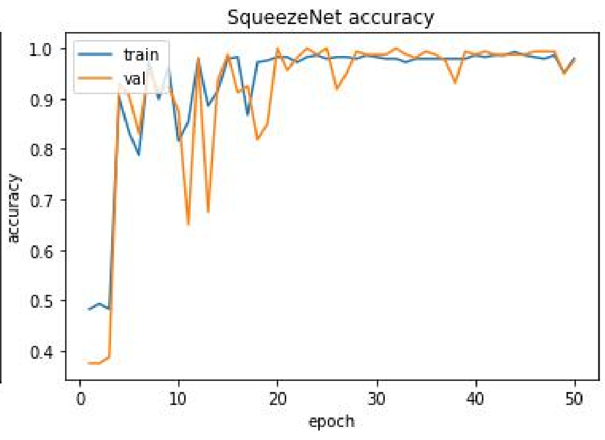
Regular CNN: training (train) and testing (val) accuracy of two models for large cell carcinoma data



The following results have showed the training and testing accuracy of two models for small cell carcinoma data as epoch increasing. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 98%, and each epoch takes approximate 15 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 10 seconds. For a simple calculation, AlexNet will take approximate 20 epochs to converge, which will take total around 5 minutes where SqueezeNet will take approximate 20 epochs to converge, which will take total around 3.3 minutes.

Regular CNN: training (train) and testing (val) accuracy of two models
for small cell carcinoma data

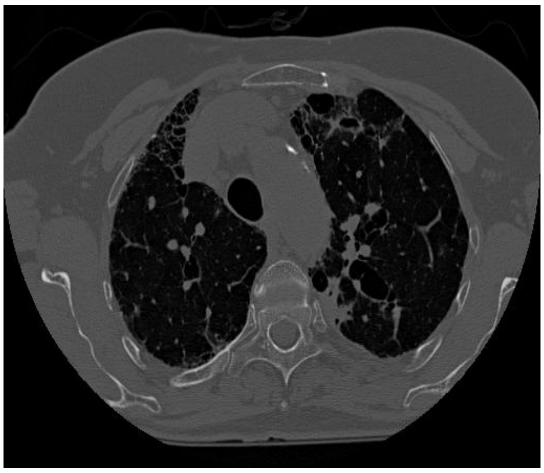




As the results showed so far, all of training and testing results for these three types of lung cancer seem to be fairly good for both AlexNet and SqueezeNet. For adenocarcinoma data, using SqueezeNet will be more effiecient than AlexNet where using SqueezeNet takes less time to converge and has fairly good test accuracy. For large cell carcinoma data and small cell carcinoma data, using AlexNet will be more efficient than using SqueezeNet where using AlexNet takes less time to converge and has fairly good test accuracy.

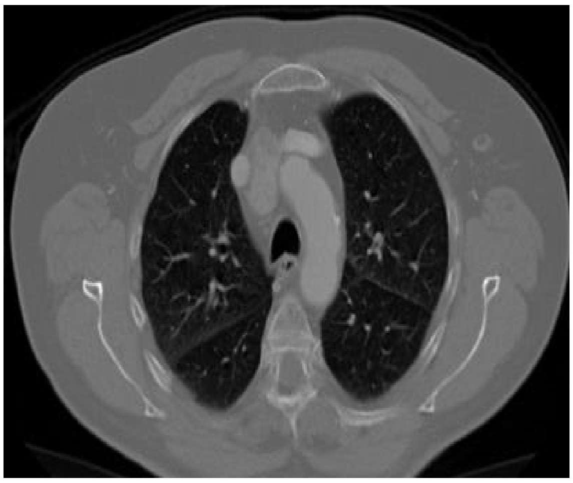
So far it seems like regular CNN for both COVID-19 detection and lung cancer detection work well. Recall that there is the source of image issue that will influence CNN models performance. Thus, in transfer learning process, I need to ensure that the images of lung cancer dataset and the images of COVID dataset looks like from the same source. Luckily, I do have one source of COVID images look like similar (in terms of source) with lung cancer image as the images have showed in Figure 14 and Figure 15. I used COVID data from source A in testing set in terms of transfer learning process.

Figure 14



(Source A: COVID Lung Scan)

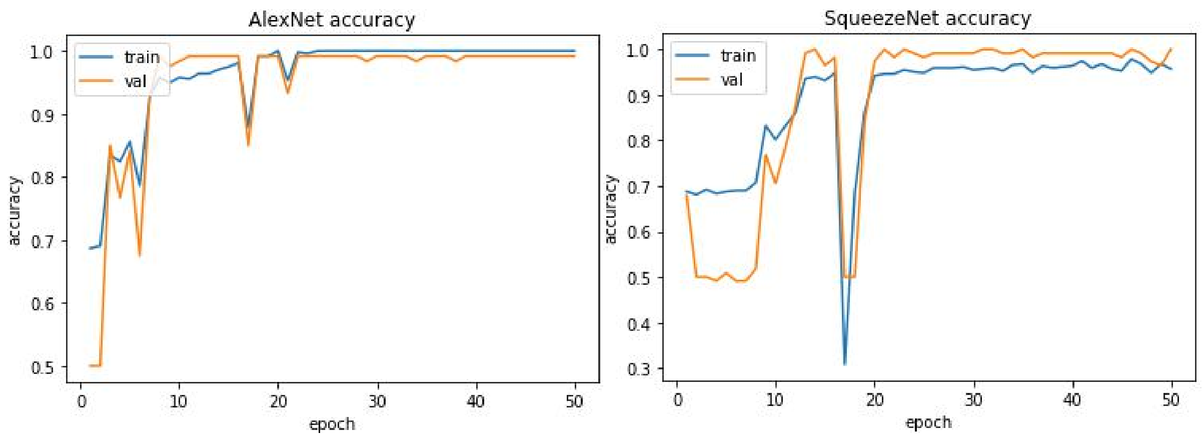
Figure 15



(cancerous Lung Scan: adenocarcinoma)

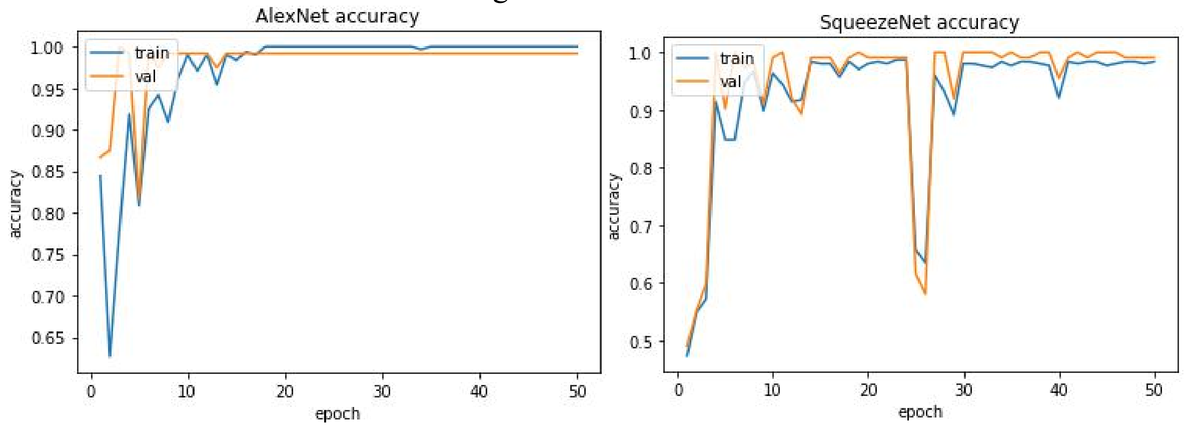
Now let’s look at the results for transfer learning process. The following results have showed the training and testing accuracy of two models for adenocarcinoma data as epoch increasing. For AlexNet’s performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 15 seconds. For SqueezeNet’s performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 10 seconds. For a simple calculation, AlexNet will take approximate 20 epochs to converge, which will take total around 5 minutes where SqueezeNet will take approximate 20 epochs to converge, which will take total around 3.3 minutes.

Transfer Learning: training (train) and testing (val) accuracy of two models for adenocarcinoma data



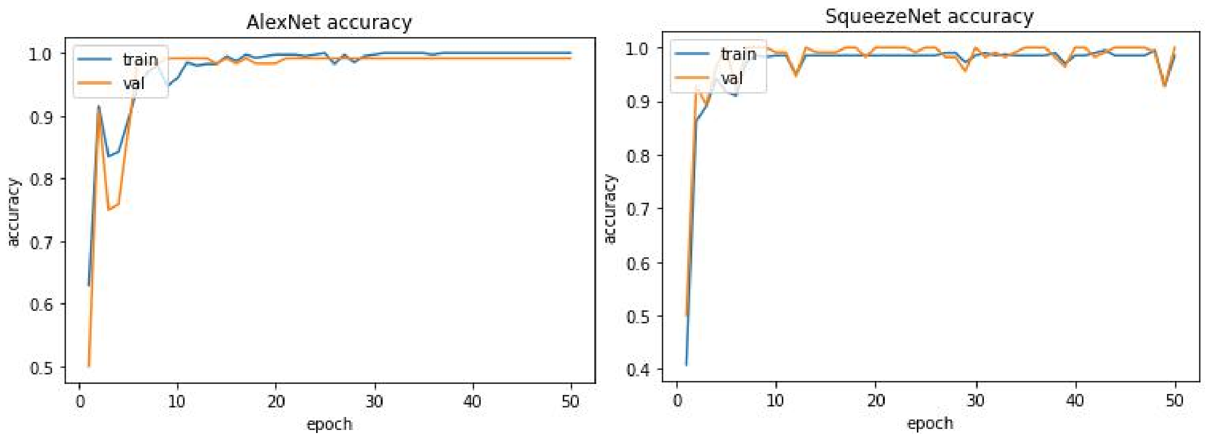
The following results have showed the training and testing accuracy of two models for large cell carcinoma data as epoch increasing. For AlexNet’s performance, the highest test accuracy as well as the convergence of train accuracy occurred around 20th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 15 seconds. For SqueezeNet’s performance, the highest test accuracy as well as the convergence of train accuracy occurred around 15th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 10 seconds. For a simple calculation, AlexNet will take approximate 20 epochs to converge, which will take total around 5 minutes where SqueezeNet will take approximate 15 epochs to converge, which will take total around 2.5 minutes.

Transfer Learning: training (train) and testing (val) accuracy of two models for large cell carcinoma data



The following results have showed the training and testing accuracy of two models for small cell carcinoma data as epoch increasing. For AlexNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 15th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 15 seconds. For SqueezeNet's performance, the highest test accuracy as well as the convergence of train accuracy occurred around 10th epoch, the highest test accuracy is around 99%, and each epoch takes approximate 10 seconds. For a simple calculation, AlexNet will take approximate 15 epochs to converge, which will take total around 3.75 minutes where SqueezeNet will take approximate 10 epochs to converge, which will take total around 1.6 minutes.

Transfer Learning: training (train) and testing (val) accuracy of two models for small cell carcinoma data



The transfer learning results for three types of lung cancer seems to be good. However, for large cell carcinoma data, the test accuracy is approaching 100% when the train accuracy is far away from 100%, which is abnormal. For adenocarcinoma data and small cell carcinoma, the train accuracy and test accuracy curve are normal so the selection of source A data of COVID lung scans work fine. The abnormal performance in large cell carcinoma is not due to source of data. I need to identify the abnormality of large cell carcinoma results and determine whether the transfer learning process for large cell carcinoma is valid.

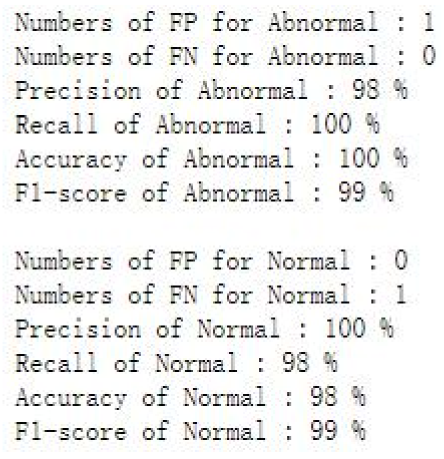
At the end of methodology section and middle of data and data visualization section, I did mention the possible reasons that why I am thinking using transfer learning will work to justify myself. Let me recap the ideas here: 1. For the binary classification in my project, there is one tricky part for computer to perform binary classification in terms of transfer learning. If there are only two classes to be classified in transfer learning, the computer doesn't necessarily need to learn the abnormal class from transfer learning process. All the computer needs to do is learn the features of normal class and then determine whether the images belong to normal class or not (if not belong to normal class, then it has belong to abnormal class because this is binary classification), which means transfer learning by training lung cancer scans has less influence than training CNN models by using normal lung scans; 2. For COVID lung and cancerous lung, they have similar pathway so I was thinking they probably will have similar presentation to human vision or to computer vision.

The two ideas above will support transfer learning will be a success. However, if the first idea is the main cause of transfer learning's success, it means the CNN models didn't classify the images correctly only based on learning the similarities from cancerous lung scans and COVID lung scans. Instead, CNN model classifies the images correctly in transfer learning only because the training of normal lung scans. In addition, the transfer learning part didn't provide any learnable features to support binary classification, which means transfer learning part is less meaningful. In this case, I can use any types of lung disease scans for transfer learning to detect COVID-19 if and only if the abnormal (any types of lung disease) lung scans look different from normal lung scans.

Therefore, if the success of transfer learning is due to the second idea or the second idea is the main cause, then it is what I want for this project. Obviously, the abnormal performance for large cell carcinoma results (that is, the test accuracy is approaching 100% even if the train accuracy is far away from 100%) is due to the first idea. Thus, the results for large cell carcinoma is invalid.

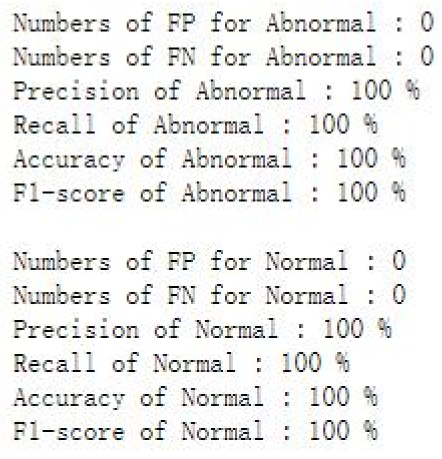
The following table shows the confusion metrics for the other two valid types of lung cancer transfer learning. The results are the same for adenocarcinoma and small cell carcinoma so I just list one table here. There is only 1 FP case for AlexNet and no misdiagnosis case for SqueezeNet. SqueezeNet generally takes less time to converge than AlexNet. At the same time, comparing with the train/test curves for adenocarcinoma and small cell carcinoma, the curves for small cell carcinoma take less epoch as well as less time to converge and seems to be more stable after the curves' convergence. Thus, SqueezeNet is a better choice for transfer learning process in COVID-19 detection.

AlexNet SqueezeNet



```
Numbers of FP for Abnormal : 1
Numbers of FN for Abnormal : 0
Precision of Abnormal : 98 %
Recall of Abnormal : 100 %
Accuracy of Abnormal : 100 %
F1-score of Abnormal : 99 %

Numbers of FP for Normal : 0
Numbers of FN for Normal : 1
Precision of Normal : 100 %
Recall of Normal : 98 %
Accuracy of Normal : 98 %
F1-score of Normal : 99 %
```



```
Numbers of FP for Abnormal : 0
Numbers of FN for Abnormal : 0
Precision of Abnormal : 100 %
Recall of Abnormal : 100 %
Accuracy of Abnormal : 100 %
F1-score of Abnormal : 100 %

Numbers of FP for Normal : 0
Numbers of FN for Normal : 0
Precision of Normal : 100 %
Recall of Normal : 100 %
Accuracy of Normal : 100 %
F1-score of Normal : 100 %
```

The following Table 3 summarize all the results from training the CNN models via different datasets (the first three COVID datasets are excluded) so far. As you can see, the transfer learning process takes similar time as the regular CNN for lung cancer detection and even shorter. They have the similar test accuracy as well. Thus, transfer learning by training cancerous lung scans for COVID-19 detection is a good substitution of regular CNN.

Table 3. Summary of the Results

Dataset and Type of CNN	Highest Testing Accuracy	Case of FP and FN	F1-Score	Numbers of Epochs to Converge	Approximate Time to Converge
Regular CNN for the Fourth COVID Dataset	AlexNet: Around 98% SqueezeNet: Around 99%	AlexNet: 6, 3 SqueezeNet: 2, 0	AlexNet: Around 98% SqueezeNet: Around 98%	AlexNet: Around 20th Epoch SqueezeNet: Around 50th Epoch	AlexNet: Around 1.2 hours SqueezeNet: Around 2.6 hours
Regular CNN for Adenocarcinoma Dataset	AlexNet: Around 97% SqueezeNet: Around 98%	AlexNet: 2, 2 SqueezeNet: 1, 2	AlexNet: Around 98% SqueezeNet: Around 95%	AlexNet: Around 25th Epoch SqueezeNet: Around 20th Epoch	AlexNet: Around 6.2 mins SqueezeNet: Around 3.3 mins
Regular CNN for Large Cell Carcinoma Dataset	AlexNet: Around 99% SqueezeNet: Around 99%	AlexNet: 1, 0 SqueezeNet: 0, 0	AlexNet: Around 99% SqueezeNet: 100%	AlexNet: Around 15th Epoch SqueezeNet: Around 40th Epoch	AlexNet: Around 3.75 mins SqueezeNet: Around 6.6 mins

Regular CNN for Small Cell Carcinoma Dataset	AlexNet: Around 98% SqueezeNet: Around 99%	AlexNet: 0, 2 SqueezeNet: 0, 0	AlexNet: Around 98% SqueezeNet: 100%	AlexNet: Around 20th Epoch SqueezeNet: Around 20th Epoch	AlexNet: Around 5 mins SqueezeNet: Around 3.3 mins
Transfer Learning for Adenocarcinoma Dataset	AlexNet: Around 99% SqueezeNet: Around 99%	AlexNet: 1, 0 SqueezeNet: 0, 0	AlexNet: Around 99% SqueezeNet: 100%	AlexNet: Around 20th Epoch SqueezeNet: Around 20th Epoch	AlexNet: Around 5 mins SqueezeNet: Around 3.3 mins
Transfer Learning for Large Cell Carcinoma Dataset (Invalid)	AlexNet: Around 99% SqueezeNet: Around 99%	AlexNet: 1, 0 SqueezeNet: 0, 0	AlexNet: Around 99% SqueezeNet: 100%	AlexNet: Around 20th Epoch SqueezeNet: Around 15th Epoch	AlexNet: Around 5 mins SqueezeNet: Around 2.5 mins
Transfer Learning for Small Cell Carcinoma Dataset	AlexNet: Around 99% SqueezeNet: Around 99%	AlexNet: 1, 0 SqueezeNet: 0, 0	AlexNet: Around 99% SqueezeNet: 100%	AlexNet: Around 15th Epoch SqueezeNet: Around 10th Epoch	AlexNet: Around 3.75 mins SqueezeNet: Around 1.6 mins

Conclusion

All in all, using CNN models (AlexNet and SqueezeNet) for COVID-19 detection has around from 98% to 99% of test accuracy, which indicates convolutional neural network can potentially be an alternative way of RT-PCR (Reverse Transcription–Polymerase Chain Reaction) for COVID-19 detection. However, in this project, I didn't specify the early stage of COVID-19 scans and we cannot know how CNN would work for early stage COVID-19 detection. In reality, when actual implement CNN into COVID-19 detection, you may also want to make sure your images are from the same source. If not, you want to randomly and evenly split the images from different sources into both/all of the classes for both training set and testing set. That is the premise to correctly implement CNN for COVID-19 detection.

In the case that transfer learning for binary classification, and transfer learning process will only substitute one class and remain the other class the same, you may want to ensure the two substituted classes have similarity/similarities for computer vision because you want to have the correct binary classification results by training both classes rather than only one class's training working.

For both regular CNN and transfer learning in CNN, SqueezeNet performs better than AlexNet overall and small cell carcinoma (a type of lung cancer) is more suitable for transfer learning than adenocarcinoma (a type of lung cancer). The success of transfer learning by using small cell carcinoma data to train the model infers that implement sufficient existing diseases scans for training CNN models to detect the similar disease that has insufficient data is workable. In the future, when a new kind of lung disease break out, by analyzing the limited infected lung scans if we find this new disease has similar pathway and presentation to some existing lung disease, then we can use the existing lung disease for transfer learning and detect the new lung disease. As the result, we can perform quick lung disease detection before it largely breaks out yet we still need to perform further detection to identify which exact type of lung disease (at least one type of lung disease will have) the patient have.

Limitation and Further Study

There are some limitations for this project. If you want to study the same field as this project, consider the following limitations:

1. This project only proved transfer learning in terms of lung diseases is workable. Whether transfer learning can be more generally used in the other types of diseases such as brain diseases is unknown as it needs further studies to determine.

2. The three assumptions regarding different sources issue in the results section might not able to generalize the situation. They need more combinations of datasets with different sources to verify the assumptions.

3. Since there are multiple sources of lung scans, the lung scans from different sources may contain the human organs other than lung, which will be the irrelevant learnable parameters that will influence the performance of CNN models.

4. The selection of parameters is a time-consuming process, to distinguish good or bad choices of parameters, using a small size dataset probably cannot show the gap between good and bad choices significantly. In this project, I adjust the parameters by training and testing a large dataset (total of 5500 images). However, there are many other possible combinations of parameters potentially giving a better performance for CNN models. I just found one combination, which provides good results yet it doesn’t mean I optimized the selection of parameters.

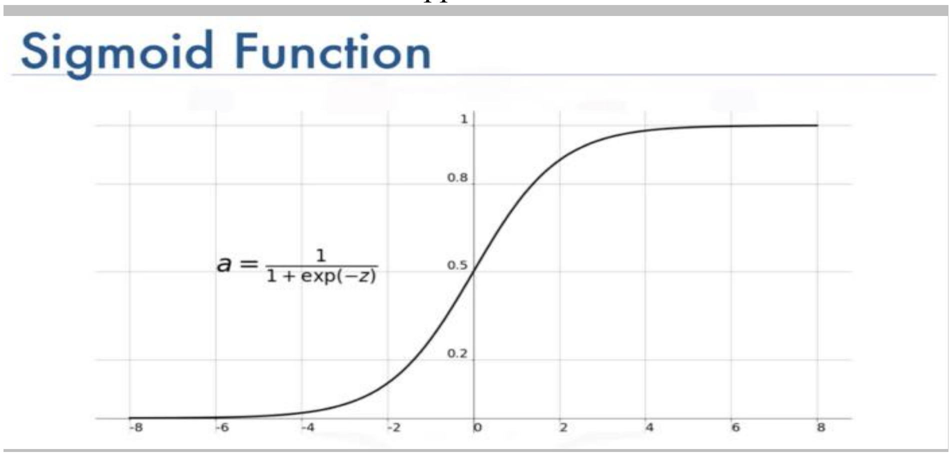
5. In this project, I used only my own PC to run the code. There are other factors that could influence the performance of CNN models such as CPU and GPU. To make sure the results are objective, using multiple devices (with different configurations) to run the same codes could be a good way to justify the objectivity of the results.

6. For the lung cancer dataset, I didn't find sufficient lung scans (more than 5000) so I only used one combination of lung cancer dataset for regular CNN and transfer learning CNN to get the conclusion. This still needs more combinations of data to verify my conclusion.

7. In reality, if using transfer learning diagnosis the disease as abnormal, we still cannot know which exact type of disease the patient have (either have one of the two diseases or both) and we need to perform the further diagnosis to identify the type of disease.

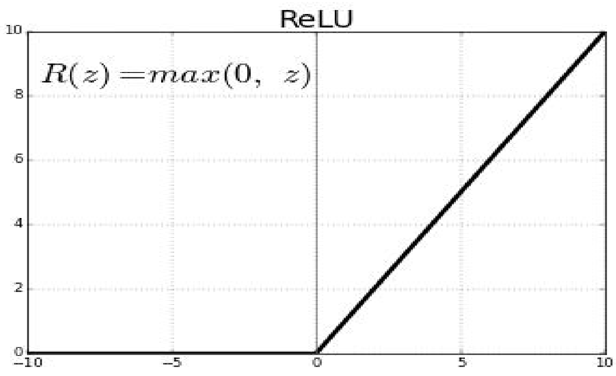
8. In terms of transfer learning for binary classification (keeping one class the same in both training and testing set), CNN models could classify the images correctly either learning the features from one class or learning the features from two classes. We want to accept the results when the second reason makes transfer learning for binary classification success, but we cannot ignore that to what degree the first reason influences the performance of CNN models. Unfortunately, there is no way to fully get rid of the effect of first reason in this project. In the future, considering uses transfer learning for muti-class classification and that probably would get rid of the effect of the first reason.

Appendix 1



(Cited from: https://medium.com/@toprak.mhmt/activation-functions-for-deep-learning-13d8b9b20e)

Appendix 2



(Cited from: https://medium.com/@kanchansarkar/ReLU-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec)

References

World Health Organization. (2019). WHO Coronavirus (COVID-19) Dashboard. World Health Organization. Retrieved April 1, 2022, from https://covid19.who.int/

Henderson, D. A. (2011, December). The eradication of smallpox – an overview of the past, present, and future. Vaccine. 29 Suppl 4: D7–9. DOI: https://doi.org/10.1016/j.vaccine.2011.06.080

Patterson, G. E., McIntyre, K. M., Clough, H. E., & Rushton, J. (2021). Societal Impacts of Pandemics: Comparing COVID-19 With History to Focus Our Response. Frontiers in public health, 9, 630449. DOI: https://doi.org/10.3389/fpubh.2021.630449

Winkelhake, H. (2021, September 3). If you're fully vaccinated, wait a few days after a COVID-19 exposure before getting tested. *Norton Healthcare*. Retrieved from: https://nortonhealthcare.com/news/how-long-after-exposure-to-test-positive-for-covid/

Brihn, A., Chang, J. O., Yong, K., et al. (2020). Diagnostic Performance of an Antigen Test with RT-PCR for the Detection of SARS-CoV-2 in a Hospital Setting — Los Angeles County, California, June – August 2020. MMWR Morb Mortal Wkly Rep 2021;70:702 – 706. DOI: http://dx.doi.org/10.15585/mmwr.mm7019a3

Bengio, Y. (2012, June 24). Practical recommendations for gradient-based training of deep architectures. Cornell University. Retrieved from: http://dx.doi.org/10.15585/mmwr.mm7019a3

Masters, D., & Luschi, C, (2018, April 20). Revisiting Small Batch Training for Deep Neural Networks. Cornell University. Retrieved from: https://arxiv.org/abs/1804.07612

Tahamtan, A., & Ardebili, A. (2020). Real-time RT-PCR in COVID-19 detection: issues affecting the results. Expert review of molecular diagnostics, 20(5), 453 – 454. DOI: https://doi.org/10.1080/14737159.2020.1757437

Thomas, L. (2021, February 19). COVID-19 and lung cancer have a common pathway, say researchers. *News Medical Life Sciences*. Retrieved from: https://www.news-medical.net/news/20210219/COVID-19-and-lung-cancer-have-a-common-pathway-say-researchers.aspx

Bishop, C. (1996). *Neural Networks for Pattern Recognition* (1st ed.).
Oxford University Press.

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*.
The MIT Press.

Figure 1, Cited from: https://www.nature.com/articles/s41467-020-18611
Figure 2, Cited from:
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
Figure 3, Cited from:
https://ayeshmanthaperera.medium.com/what-is-padding-in-cnns-71b21fb0dd7
Figure 4, Cited from:
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/
Figure 5, Cited from:
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/
Figure 6, Cited from: https://en.wikipedia.org/wiki/Convolutional_neural_network
Figure 7, Cited from:
https://www.researchgate.net/figure/The-architecture-of-SqueezeNet-11_fig4_335357358