Project Title: Transfer learning for covid-19 diagnosis

1. Goal of the project

The goal of this project is to develop a 2D Convolutional Neural Network (CNN) for COVID-19 diagnosis using CT images of lungs. In a 2D CT image, the indicator of COVID-19 positive is the ground class opacity at both lobes. In deep learning, convolutional neural network is applied for image classification. A CNN consists of many hidden layers, and each layer learns particular features of the input data. As each input neutron is connect to the next hidden layer, different features are learned gradually, which enables the neural network to analyze the whole input 2D image.

2. Hardware and software Environment

To build a neural network for this project, we need to configure the hardware and software.

Software requirement:

- Jupyter notebook
- PyTorch package and torchvision package
- macOS

Hardware requirement:

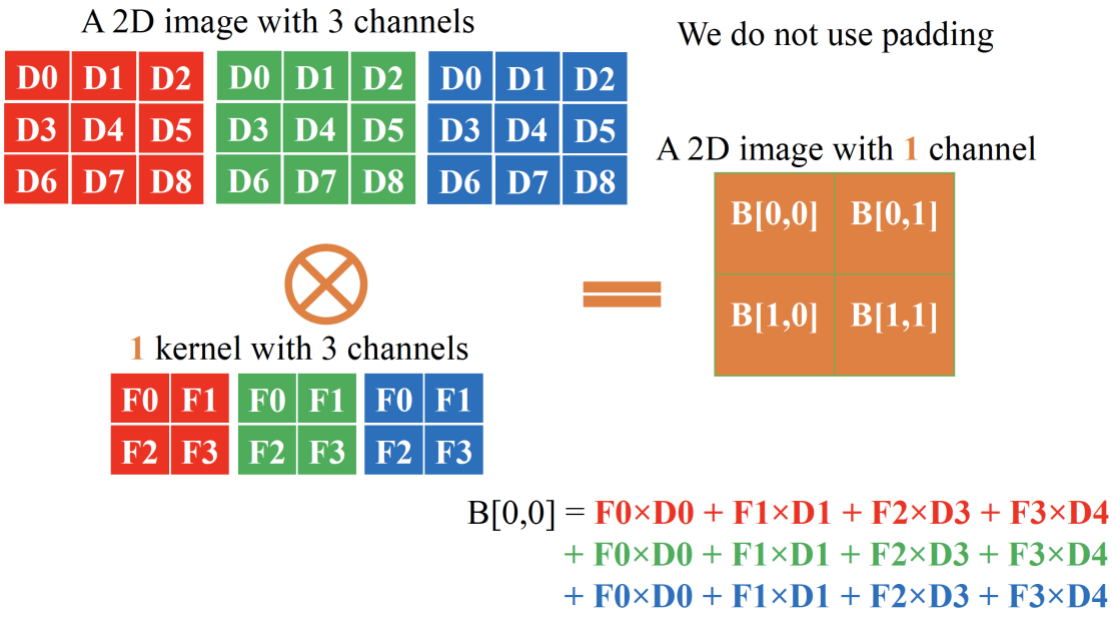
- i5 CPU or higher
- memory space at minimum of 10GB
- Graphic Processing Unit (GPU) is preferred

3. The rationale of using CNN for this project

CNN is a class of artificial neural network designed for processing structured arrays of data such as images. Differing from conventional classifiers like KNN classifier, and decision tree, a CNN can analyze images that have complex patterns. CNNs can leverage massive compute resources to ferret out tiny and inconspicuous visual patterns that might go unnoticed to the human eye. When it comes to the field of medicine, it may help the human doctors to diagnosis CT images from a large number of patients. With a huge amount of training data, a trained CNN can detect important features automatically without human supervision. Well-trained CNNs might even outperform human experts at detecting relevant patterns.

4. The basic components of CNN

A CNN consists of many hidden layers. Each layer has many neurons. Neurons' behavior depends on the weight of the neuron. The number of neurons is determined by the size of the input layer. Normally, the image with high resolutions needs more neurons to extract the patterns from the pixel values.



4.1 Convolution operation function

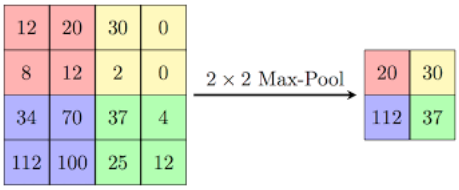
The processing of getting a weighted sum of the input in a small region is convolution. As the above figure shows, the 2D image has 3 channels: Red, Green and Blue (RGB). One neuron can have several weights. Take 2*2 red matrix as an example, the F0, F1, F2, F3 are weights in a neuron. The process of convolution is performing the matrix multiplication between the weight matrix and pixel matrix. It is repeated for every channel in this layer. The weighted sum is the sum of the value. But the 2*2 matrix in not enough to feed the 3*3 matrix. Then the matrix moves 1 step right and do the same thing for D1, D2, D4, D5. 1 is stride here. Then do the same process for D3, D4, D6, D7. The orange matrix is the matrix of weighted sum (net output). But it's still not the output of this layer.

4.2 Activation function

There are different kinds of activation functions, including sigmoid, Tanh, and ReLU. In this project, each convolutional layer is followed by a ReLU layer. ReLU means rectified linear unit. Why not passing the net output from the conv layer directly to the next conv layer? The rectified linear activation function overcomes the vanishing gradient problem. It allows the CNN model to learn faster and perform better. How does ReLU work? For the input larger than zero, it returns the input. For the input smaller than 0, it returns 0. Why does ReLU become my choice for this neural network? Comparing with sigmoid and Tanh, ReLU does not need to use the exponential function. I use it for computational efficiency. ReLU also looks like linear function. For a CNN, it's easier to optimize when it is close to linear. In the last layer, SoftMax function is used because it converts the output of the last layer in neural network into a probability distribution. It turns a vector of k real values into a vector of k real values that have a summation of 1.

4.3 Maxpooling function

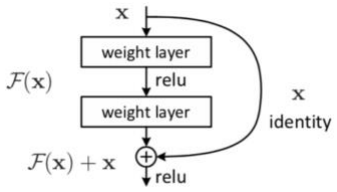
Images from reality have many details, and some of them are similar but different in some details. However, convolutional layers record the precise position of the features. I do not want the CNN to classify them into a new category just because the image feature is in a different location. The approach is to create a picture with lower resolution. For the example shown below, Maxpooling does the job of choosing the largest value in red, yellow, purple, and green. Then the values form a new matrix. The pooling layer will always reduce the size of the map by a factor of 2. It is useful as small changes in the details of the feature from the convolutional layer will result in a pooled feature map in the same location.



4.4 The full connection layer

At the last full connection layer, we need to convert the net output size to the number of classes. In this project, we need to use linear function to convert the net output to a 2-dimensional vector and use SoftMax to convert the probability with a summation of 1 for 2 classes. (COVID vs. Normal).

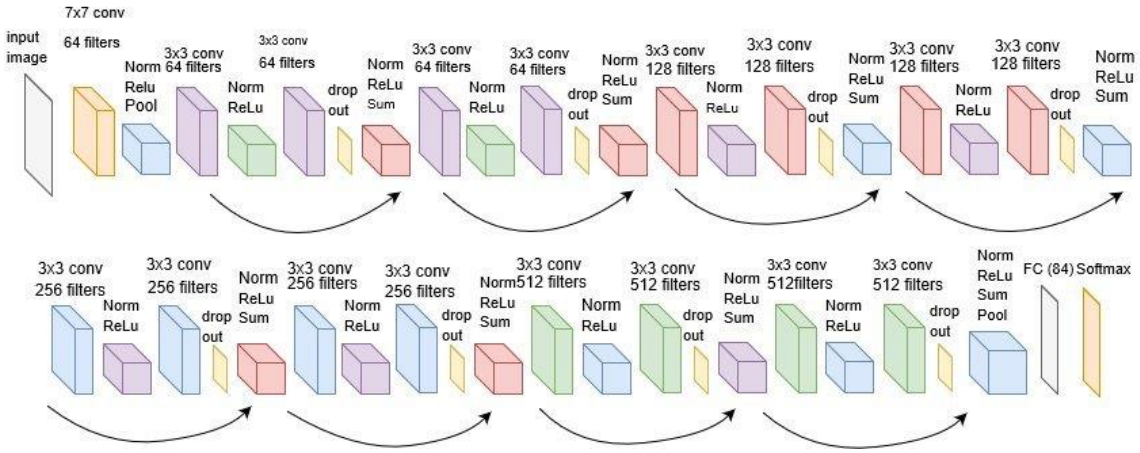
4.5 Residual block



This figure shows a residual block. It takes the best output from x itself and the output from ReLU. The optimization of a deep network with residual blocks will be easier.

5. The structure of the CNN and transfer learning

This project is the implementation of a CNN for classifying CT images. There are two approaches of building the neural network. The first one is to train a CNN from scratch. The second one is to apply transfer learning. In this project, I tried both approaches and used Resnet-18 as the reference design, which is shown in the figure below.



this diagram shows the flowchart of resnet-18.

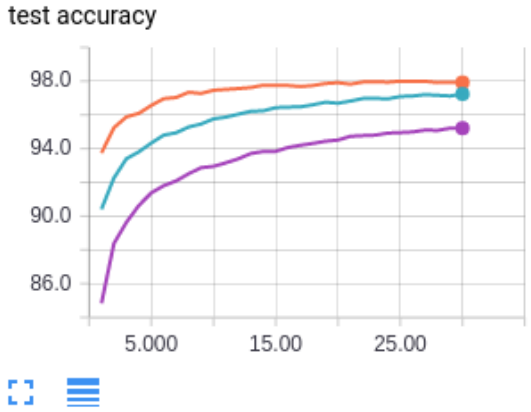
5.1 Preprocessing for dataset

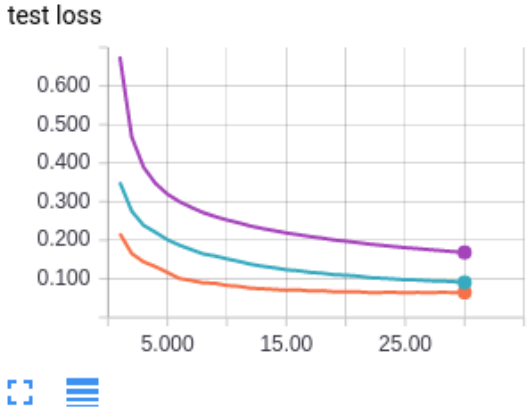
I resize the image to (224,224,3) and append it to the array. Then I convert input format to torch format which is (X, 3, 224, 224). X is the total number of images in the set.

Secondly, I create layers by following the pattern of residual block. The network is shallow because it has 1+4*4+1 layers. The first one is initialization of convolution. Each layer from 4 layers in the middle represents 2 conv layers plus 2 ReLU layers.

5.2 Training Parameters

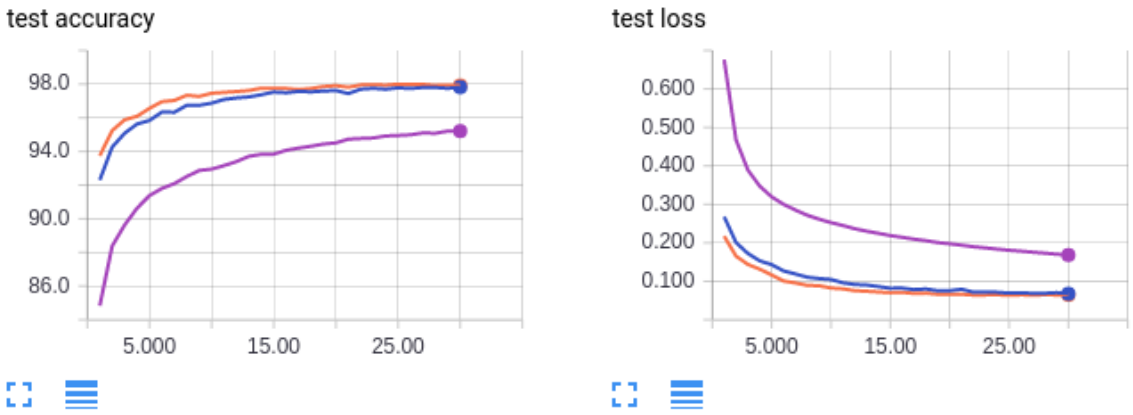
- **Batch Size**



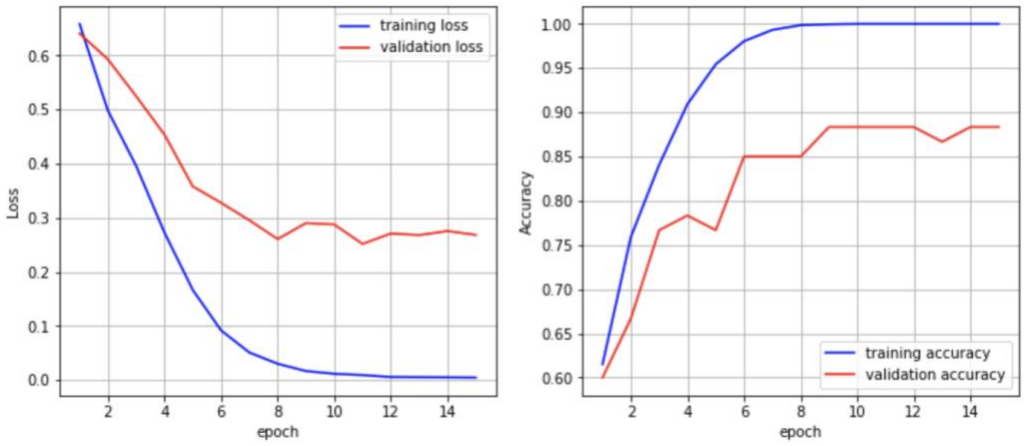


Orange curves: batch size 64
Blue curves: batch size 256
Purple curves: batch size 1024
Larger batch size increases the training speed but lower asymptotic accuracy.

- **Learning Rate**



Orange curves: batch size 64, learning rate 0.01 (reference)
Purple curves: batch size 1024, learning rate 0.01 (reference)
Blue: batch size 1024, learning rate 0.1
Lowering the learning rate can increase test accuracy. We can use it to recover the lost test accuracy by reducing the learning rate. However, a lower learning rate requires more epochs for each weight to update. In a nutshell, the learning rate determines how quickly the model can converge to the best accuracy.
Then I train the model using an optimizer for 16 epochs. I also recorded the training and validation loss in each iteration.



5.3 Classification Reports

Then I train the model using an optimizer for 16 epochs. I also recorded the training and validation loss in each iteration.

```
test accuracy:   0.88671875
classification reports:
                    precision    recall  f1-score   support

class 0: Non-Covid       0.91      0.91      0.91       200
    class 1: Covid       0.91      0.91      0.91       200

          accuracy                           0.91       400
         macro avg       0.91      0.91      0.91       400
      weighted avg       0.91      0.91      0.91       400
```

6. Train the CNN using transfer learning

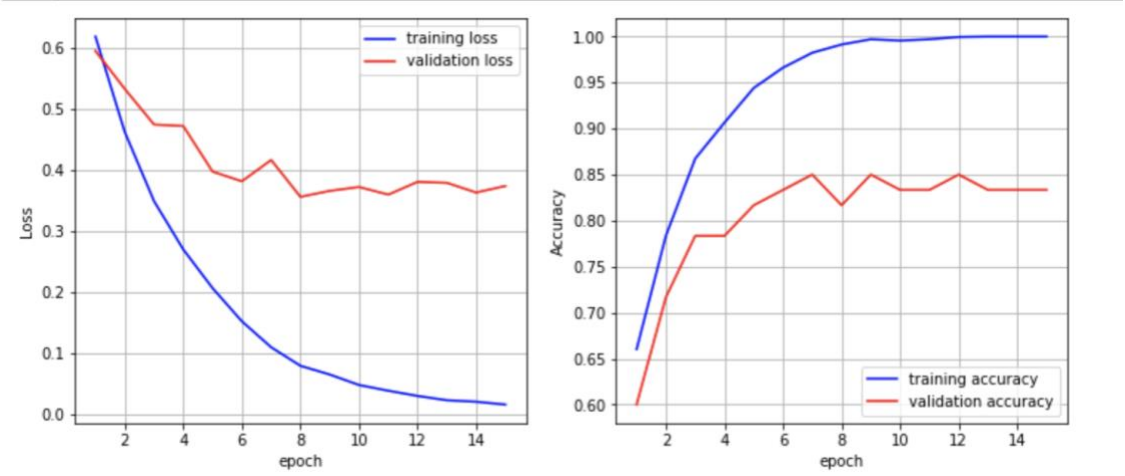
Transfer learning is using the pretrained model trained on a large dataset and transfer its knowledge to the smaller database.

6.1 Approaches for Transfer Learning

There are two ways to transfer learning. The first one is freezing some layers and train other layers. The second one is fine-tuning, which means modify the parameters to gain better performance.

In this case, I attempted to freeze the first 16 layers and retrain the last 2 layers. The accuracy is around 80%, which is low. So, I fine-tuned the model by modifying the dimension of the last layer to 2, which is to replace the original fully connected layer with a fully connected layer.

The accuracy is around 0.85 to 0.92 depends on the random initialization of weights. This score is higher than the accuracy from the scratched model.



The training is very slow on my computer because the learning rate is very low and GPU is not available. That's the reason to use a computer with GPU to do machine learning.

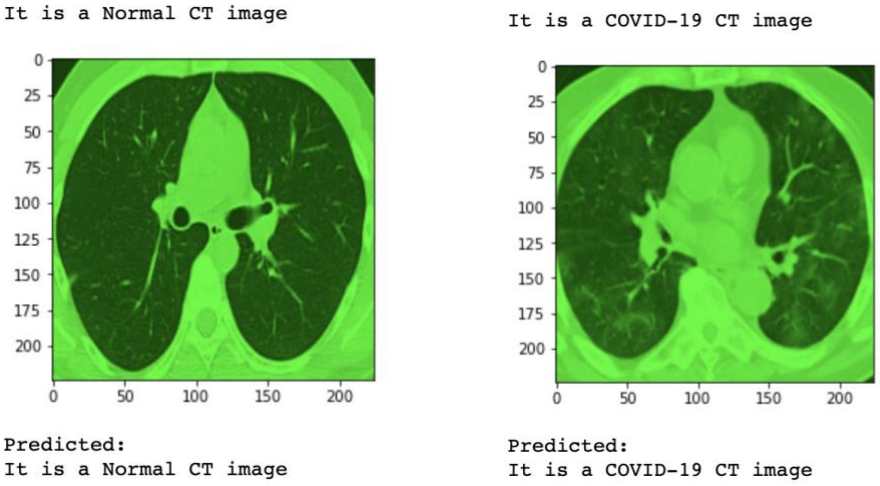
```
test accuracy:   0.8828125
classification reports:
                    precision    recall  f1-score   support

class 0: Non-Covid       0.89      0.92      0.90       200
    class 1: Covid       0.92      0.89      0.90       200

          accuracy                           0.90       400
         macro avg       0.90      0.90      0.90       400
      weighted avg       0.90      0.90      0.90       400
```

baseline(learning from scrach): 88.67%

transfer learning(with pretrained ResNet18): 90.23%

6.2 Result of the prediction

To get the result of the prediction, I visualize the model first. After the full connection layer. The model outputs 2 vectors of probability. However, it's not showing the relationship between the 2 classes. So, I used SoftMax to take the exponential of the output and use a max function to get the larger one from 2 probabilities. The sum of the probabilities is one.



The left one is the normal CT image, and the prediction shows a normal CT image, which is correct.

The right one is the covid-19 CT image, and the prediction displays a COVID positive image, which is correct.

7. Conclusion

I Developed a CNN for COVID-19 CT image classification and fine-tuned the resnet-18 pretrained model. After comparing two training methods, it shows the accuracy of two training methods are similar based on the given dataset. However, running time for transfer learning is smaller. The running time for training of the scratched model is 3 hours and the running time for training of transfer learning is 30 minutes.

So, Transfer learning is preferred in this case. If the image patterns in the new dataset is similar to the image patterns in dataset for pretrain, then the base model can be modified for the classification/regression task on new dataset. The models are only based on the database given. The results are exploratory. Medical validations have not been given for this model. This model can be further improved with a larger database, so that the model can be more well-rounded and generalizable.