COVID-19 Detection Using CT Scans: STAT453 Project Report

James Thomason
jethomason@wisc.edu

Steven Xia
sxia35@wisc.edu

Saniya Khullar
skhullar2@wisc.edu

Abstract

For the past year, COVID-19 has had a major impact on the entire globe, resulting in high numbers of deaths and hospitalizations. With over 3 million deaths worldwide and millions more fighting for their health, this project is important because it is an opportunity to make a real positive impact on people's lives. Communities of lower socioeconomic status are especially at risk of COVID-19 and given their lack of health and safety resources, there is a clear struggle for physical resources such as test kits and vaccines. An image detection model could potentially keep many people safe while being incredibly efficient in use of resources, as use of CT scans for detection of COVID-19 would help ease the strain of lack of available testing in areas as well as reduce the waste created by the use of swabs and spit containers. The objective for this project is to develop an object detection model which can successfully determine whether or not a given subject is positive or negative for COVID-19 through data extracted from that individual's given Computed Tomography (CT) radio-graph digital scan.

1. Introduction

CT Chest Scans are non-invasive tests that are enhanced versions of X-rays and provide detailed information (ex. bones, fats, muscles, organs) on an individual's chest [8]. These are also known as lung CT scans.

Chest CT is very crucial in a wide variety of scenarios, especially for processes such as medical diagnosis, both COVID-19 related health issues (acute respiratory distress syndrome, pulmonary embolism, and heart failure) as well as traditional medical problems. While CT Chest Scans serve a wide scope of medical purposes, its role in predicting future medical events is still under scrutiny and investigation [4].

Data was obtained with CT scan images of 742 peoples chest cavities, each testing either positive or negative for COVID-19. The data is divided into 374 patients diagnosed with COVID-19 and 397 who were not. This means that the data was not paired. Certain images contain certain extra identifiers such as letters, arrows, and circles. By default the data has three color channels: red, green, and blue. The data was used as it was given, leaving in these extra indicators. Both of the groups of images contained a relatively similar number of images with extra indicators. Because of this and time restraints, the group chose to analyze the data as it was given.

From a personal perspective the group is interested in learning how to create and execute image classification neural networks. Image classification neural networks have large real world applications like self driving cars, disease prevention, security risk detection etc. It is also something that can be seen visually and therefore can be easily presented and explained as opposed to non-visual data sets.

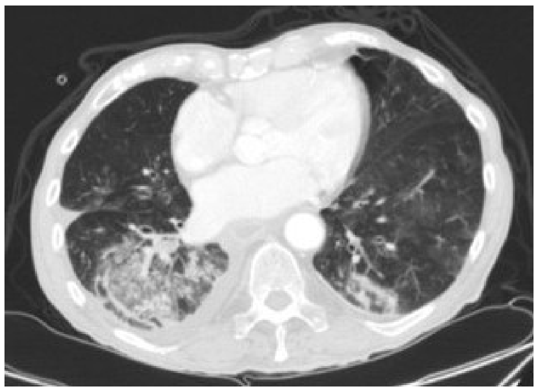


Figure 1. CT scan of a patient without COVID-19

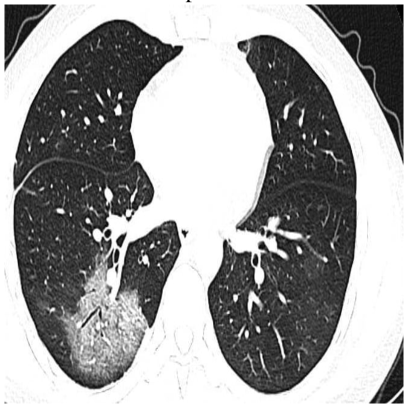


Figure 2. CT scan of patient with COVID-19

2. Related Work

There has been a variety of related work that has been done in this field.Computer Vision approaches have been focused on answering different questions pertaining to Covid-19, such as image features to correctly identify bacteria, viral, Covid-19, and pneumonia conditions; screening infected individuals to better enforce social distancing protocols; maintaining healthcare equipment for Covid-19 patients, and developing a Covid-19 vaccine [8]. Other studies have focused on individuals who survived Covid-19 and on building medical image classifiers to better differentiate between individuals who suffered from severe Covid-19 symptoms and those who did not (not severe). In a study conducted by the University of Waterloo, a human-machine collaborative design strategy was utilized to create a neural network where human-driven principled network design prototyping is combined with machine-driven design exploration to produce a network architecture tailored for the detection of COVID-19 cases from CXR images [9].

Additionally, in another recent study, other parameters are taken into account to diagnose COVID-19, such as the typical radiologic appearance of the lungs as detected by a CT-scan.

The authors of the dataset that was used [6] trained 5 different deep CNN network models (AlexNet, VGGNet16, VGGNet19, GoogleNet, and ResNet50) and found that ResNet50 was the top deep learning model. They had a test accuracy of 82.91 percent. [6]. We will be expanding on their work by applying other CNN models.

Since this is an image detection problem, we will be utilizing concepts in Computer Vision, such as state-of-the art implementations of You Only Look Once (YOLO) object detection [7].

3. Proposed Method

There were some markers in a few of the original CT scans, such as labels (A, B, C) and other shadings (possibly due to technical issues in scanning or markings by radiologists). To address this, the images were converted to grayscale. Since both Covid positive and negative CT scans contained these labels, the risk of retaining those labels in terms of possible overfitting was reduced. Therefore, these CT scans were not edited to remove those labels as those could introduce possible errors and corruption of the data.

Our images were shrunk down to being 64 by 64 pixels in size. Since we applied grayscale to our image, we only had 1 channel instead of the 3 channels (R, G, B). Thus, please note that our input image is: 64 x 64 x 1. Usually in CNNs, the height and width of the images tend to be the same.

Existing Convolutional Neural Network architectures were used, such as LeNet-5, AlexNet, and ResNet.

3.1. CNN Models Used

LeNet-5 Consisting of 3 convolutional layers, 2 subsampling layers, and 2 fully connected layers, LeNet-5 is a 7-layer Convolutional Neural Network architechture [5]. This is a relatively simple model that can be run on the CPU.

The first layer of LeNet-5 is a Convolutional Layer, which takes in the number of input channels, which is 1. There is a kernel size of 5, and this layer contains all of the weights. It produces an output of 6 feature maps,

The first layer of LeNet-5, a convolutional layer, has a kernel size of 5x5 and produces an output of 6 feature maps, each having size 60 by 60 (as there were 4 pixels lost in height and in width due to the padding). Following this convolutional layer an activation function and then a subsampling layer, where the dimension of the feature maps it receives from the previous layer is cut in half. Thus, the dimensions become 60/2 = 30, and therefore the dimensions are 30 by 30. This subsampling layer also produces 6 feature maps as an output, each one corresponding to the feature maps passed as input from the previous layer [5]. There is another Convolutional layer that then reduces the dimensionality by 4 pixels in each direction, and then the dimensionality becomes 26 by 26. Another activation function is applied. Lastly, there is a maximum pooling layer with a kernel (or stride) of 2, essentially halving the width and height from 26 by 26 down to 13 by 13. The feature extractors are both of these convolutional layers.

Thus, this final Convolutional Layer output dimension is 16 channels (or feature maps) that are 13 by 13 in dimension. Overall, there are (Height) * (Width) * (16 channels) as the dimension, and this is: (13)*(13)*(16) = 2,704 inputs. In between the convolutional layer and the classifier is a flattening layer, which takes these 2,704 inputs and then transforms it into a vector that can be used by the Classifier part of LeNet-5. Essentially, LeNet-5 has a classifier that functions similar to a fully-connected neural network, a Multi-Layer Perceptron. This takes the vector of 2,704 units and then connects it to 120 neurons in the 1st hidden layer. Then, the 120 neurons in the 1st hidden layer are connected to the 84 neurons in the 2nd hidden layer. Lastly, the 2nd hidden layer connects to the output layer, which is a classifier for both classes: Covid Positive CT Scan versus Covid Negative CT scan.

AlexNet Consisting of 8 learned layers, the first 5 layers of AlexNet are convolutional layers (some have maxpooling layers) and are the feature extraction part. Additionally, the last 3 layers of AlexNet are fully-connected multi-layer perceptron. Main method using CNNs and GPUs in Deep Learning. This architecture is a breakthrough for CNNs.

Some modifications were made to the original AlexNet architecture, especially to the first layer, since the input images have a smaller size (dimensions). Thus, the first layer of AlexNet instead filters our 64x64x1 input image with a stride of 4 pixels and a padding of 2 pixels. The stride provides a measure of the distance between neighbor neurons in a kernel map, with respect to the receptive field centers [3]. In our 2nd convolutional layer, we use the response-normalized and pooled 1st convolutional layer output and filter it with 256 kernels First feature map has 64 channels, then 192 channels, then 384 channels, and lastly 256 channels in the final convolutional layer.

In between the last convolutional layer and the 1st multilayer perceptron layer, there is an adaptive average pooling that will take any size out of the last convolutional layer and pools it such that the output feature matrix will be 6x6. Thus, this 6x6 matrix can be used in the 1st layer of the Multilayer Perceptron, to get a Linear layer that has 256x6x6 = 9,216 input neurons. This 6x6 dimension was used instead of a reduced 3x3 matrix.

The last 3 layers, the Multilayer Perceptron Layers, will begin with a linear layer. Dropout rate of 0.5 is incorporated.

ResNet Since the introduction of AlexNet, recent Convolutional Neural Network architectures have used increasingly more layers in deep neural networks in order to reduce rate of error. However, when the number of layers increases, the problem of exploding and vanishing gradients arises, causing the gradient to either be too large or become zero. Because of this, when the number of layers within a deep neural network increases, the rate of error for training and testing increases as well. ResNet architechture uses a a technique called skip connections, which skips over the training of several layers and is connected directly to the output layer. The major advantage of skips connections in ResNet is that if there are any particular layers that are lowering performance of a neural network architecture, then that layer will be skipped over by regularization. This allows deep neural networks to be trained without the problem of an exploding or vanishing gradient.

CONVNet Additional architectures were considered, such as a pre-built COVID-19 Detection Neural Network (CONVNet) (https://github.com/bkong999/COVNet). This code was based on a paper using Artificial Intelligence and Chest CT Scans to distinguish COVID-19 from Community Acquired Pneumonia. [4].

3.2. Optimizers Used

Please note that the different optimizer functions were used in the CNNs for adaptive learning: the ADAM optimizer (Adam) and Stochastic Gradient Descent (SGD).

ADAM (Adaptive Moment Estimation) The Adam optimizer has 3 tunable hyperparameters, including the learning rate. The other hyperparameters are ϵ and $[\beta_1,\ \beta_2]$ (corresponding to momentum and RMSprop, respectively). There are some advantages to using Adam. To begin with, the default settings are typically optimal and hence do not require much adjustment or tuning. In fact, keeping the default settings can usually result in very high performance, even the 2nd best out of the suite of other possible optimizers. Adam is also the default optimizer that is considered by a lot of new deep learning architectures and therefore is currently the most commonly used Deep Learning optimization algorithm. There are only rare cases when Adam defaults do not perform well.

Nevertheless, there are some disadvantages associated with Adam, such as the computational burden. That is, Adam involves a lot more memory for the state (such as approximately 8n) and is more computationally intense. It also has a worse generalization (higher testing error) and converges to sharp minima. There are alternative methods, like AdamW, that adapt on ADAM to improve the weak generalization (test) accuracy, but those may involve even more memory.

Stochastic Gradient Descent (SGD) SGD has only 1 tunable parameter, which is the Learning Rate. There are some advantages to using SGD. For example, SGD usually requires 0 memory on the Graphical Processing Unit (GPU), making it very friendly for computation. In addition, it often provides the best generalization (lowest test error) Nonetheless, there are some disadvantages associated with SGD. While it typically may perform better than Adam and other optimizers, it does require an extensive amount of training and epochs to reach those high levels of test accuracy. Furthermore, more time is allocated to tune the Learning Rate to achieve those results, and SGD updates can be rather noisy. There is also a risk that the resulting model could be lost in saddle points or local minima, and therefore might not find the globally optimal solution. Moreover, is very sensitive to the initial values and the learning rate, α.

3.3. Activation Functions Used

For each CNN Model that was trained, the activation functions were standardized and used the same activation function throughout (for all of the layers in the model).

Sigmoid Function Mathematically, the logistic sigmoid function can be represented accordingly:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

An advantage of the logistic sigmoid activation function is that it is a smooth function that is differentiable every-where, meaning that it can facilitate gradient-based back propagation as well as being suitable for forward propagation. The logistics sigmoid activation is also a non-linear function, so it can be used to activate hidden layers within a given neural network. The logistic sigmoid function also has outputs ranging from 0 to 1, so it can generate probabilities as well.

Contrarily, a major disadvantage of the sigmoid activation is that there is a problem of vanishing gradients. When the activation is near the region of saturation, the change is slow, the derivative is close to 0, and the backward transfer is based upon the chain rule of calculus. The derivative needs the product of the previous derivative of each layer. Once lower values are multiplied together, the derivative result is very close to 0, and the training of the deep network cannot be completed. Additionally, the output of the sigmoid function is not centered around zero. What happens as a result of this is that the input of the neural network's back layer neurons become on average a non-zero signal. Lastly, the exponential operator within the sigmoid activation is generally quite time-consuming.

Rectified Linear Unit (ReLU) The Rectified Linear Unit, or ReLU activation function can be represented accordingly:

$$\sigma(x) = \begin{cases} 0, \text{ if } x < 0 \\ x, \text{ otherwise} \end{cases}$$

With the ReLU activation function, all negative x-values become zero, so there exists no negative signals. Here, the values of Maximum Threshold are Infinity. Thus, the vanishing gradient problem is solved as the gradient does not saturate. Therefore, the prediction accuracy of the output and its efficiency is optimized. Furthermore, the computational complexity of the ReLU activation is relatively low so the convergence speed is faster than that of the sigmoid and tanh activation functions.

Similar to the sigmoid activation, a drawback of the ReLU activation function is that its output is not centered around zero. During backpropagation, the weight gradient can only either be completely positive or completely negative. As a result of this, gradient updates for the weights start to increase and decrease rapidly, damaging neural network performance. Furthermore, the ReLU activation function is unbounded non-differentiable at zero. The gradients for negative input are zero, which means the weights are not updated during backpropagation for the activations within that particular region. This can create dead neurons that never get activated, which can play a role similar to regularization in linear regression (in terms of helping with pruning of neurons and reducing overfitting). This activation function is currently the most widely used in Deep Learning applications.

Hyperbolic Tangent Function (Tanh) Another widely used activation function is the hyperbolic tangent function, or tanh. This activation function can be represented as:

$$\sigma(x) = \frac{e^x - e^{-x}}{e^z + e^{-z}}$$

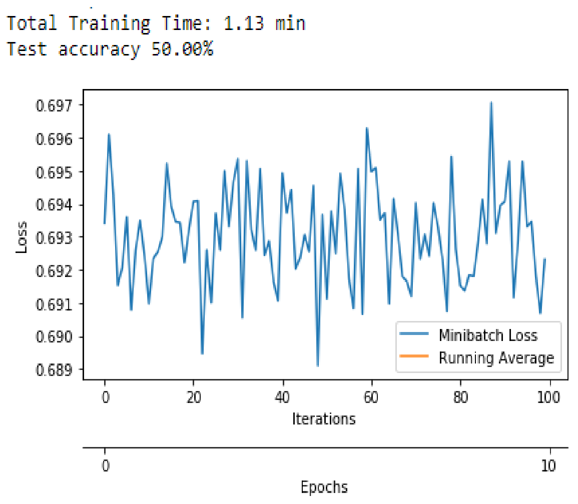
An advantage of the Tanh activation function is advantages: mean centering at 0 (incorporates positive and negative starting weights, so there can be more combinations). It also has larger gradients and is steeper, and has a simpler derivative. Moreover, it has a shape similar to the sigmoidal shape but has a mean at 0 (instead of 0.5, which the sigmoid has at net input = 0), so both negative and positive values can be considered. Furthermore, it improves on the issue that Sigmoid activation functions face, which is the fact that the gradients tend to saturate near the tails of the distribution (around -4 and below and 4 and above).

A major flaw of the Tanh activation function is that it has vanishing gradients. The gradients of the network's output with respect to the parameters in the early layers become extremely small, and thus training of the network becomes inhibited. Additionally, the power operations within the Tanh activation function are also quite time-consuming.

We did not use edge detection as that was not useful.

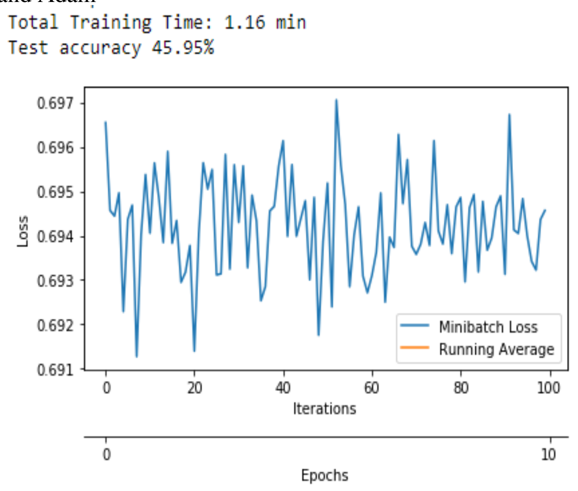
4. Experiments

Please note that random seed was set to 123. The data was randomly divided up into a training,testing, and validation set. The training set contained 80 percent of the data while 5 percent was reserved for validation, and 10 percent for the testing set. The validation and testing sets were prepared in the same way as both are evaluation data sets. The data was converted to grayscale (1 channel instead of 3) was resized to be 64x64 pixels and centered in order to standardize the data. The DataLoader utilized a batch size of 32. A normalized version of the data and a version that was not normalized was used. The normalized version of the data preformed better in every architecture tested. The architectures tested were Alexnet and LeNet 5. With each architecture a different combinations of activation and optimization function was tested. Parameters such as batch size and number of epochs were kept the same in order to expedite testing. After a few combinations were tried the idea of normalization was tested. This time only LeNet 5 was run as it preformed significantly better than Alexnet without testing the effect of normalization.



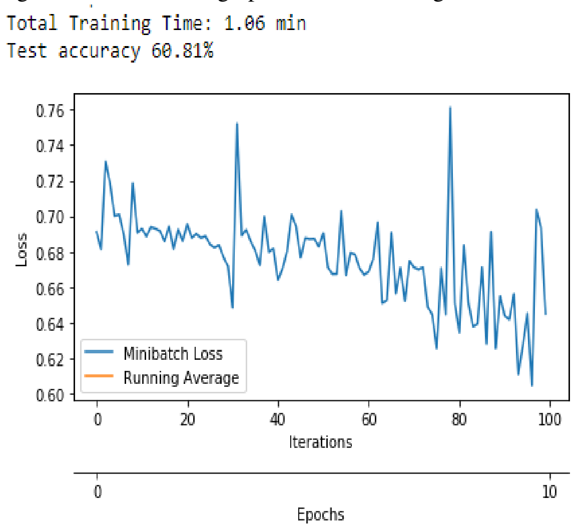
```
Total Training Time: 1.13 min
Test accuracy 50.00%
```

Figure 3. Performance graph for the normalized data using ReLU and Adam



```
Total Training Time: 1.16 min
Test accuracy 45.95%
```

Figure 4. Performance graph for the data using ReLU and Adam



```
Total Training Time: 1.06 min
Test accuracy 60.81%
```

Figure 5. Performance graph for the data using ReLU and Adam

From figure 3 and 4 it is seen that the normalized data has about a four percent increase in performance; however both models performed worse than than default percentage of COVID vs. no-COVID of about 54 percent. It is important to note that the best performance is from the figure 5 model.

4.1. Dataset

This dataset is available on Kaggle.com, where the author, has collected and also generated (data augmentation) several images related to chest CT radiographs of coronavirus-infected patients. Here, 345 of the 742 images are for COVID-19 infected individuals, and the remaining 397 images are for COVID-19 negative individuals (individuals who do not have COVID). The author was also focused on data augmentation to provide additional image data on COVID positive versus COVID negative CT Scans. Many of the images were generated using an unsupervised data augmentation approach known as CGAN, which involves General Adversarial Neural Networks, to create more images to better detect the presence of Covid-19 using CT data [6]. In this project, the focus will instead be focused on the original data set of 742 images and on performing image detection using only that set of images.

Unfortunately, the sample size is rather limited due to the difficulty in obtaining lung or chest CT images, especially during Covid-19 times. Not only is it costly to obtain these images, but also rigorous and challenging to obtain patients, to ensure that patients receive high doses, that scanner rooms are disinfected, and that proper protocols are followed to minimize possible Covid-19 spread [8].

Normal lung or chest CT scans are typically black, representing healthy lungs. Individuals with diseased lungs usually have a lot of white flecks or whitish regions in their lung CT scans because they tend to have more pus (result of neutrophil immune cells fighting infection) or fluid [2]. Covid-19 wreaks havoc on several regions in the body, especially the lung, and clinicians and researchers have observed that the CT scans for these individuals tend to have shades of grey. These lungs are in between healthy (all black) and diseased (a lot of white regions) lungs. Usually, CT Scans for Covid-19 positive patients have been found to include markers such as Ground-Glass Opacity (GGO) and/or mixed GGO along with mixed consolidation [1]. GGO are the hazy, white-flecked patterns on these CT scans, which is indicative of greater density [2]. These GGOs are also characteristic of other viral-induced pathologies and other diseases (ex. chronic lung disease, fibrosis), inflammatory conditions, and cancers; they are not likely to result from air pollution or smoking. Hence radiologists have been focused on using these GGOs to better identify Covid-19 infected individuals from non-infected individuals.

Currently, chest CT scans are not official diagnostic tools for Covid-19, but hopefully these efforts can help on-going research efforts into using Artificial Intelligence tools as an aid in this devastating global pandemic.

4.2. Software

Please note that for this project, all models were built within Jupyter Lab Notebooks (Python 3), utilizing the PyTorch, NumPy, Torchvision, pandas, and matplotlib packages. We adapted on code shared on github by Professor Sebastian Raschka (https://github.com/rasbt/stat453-deep-learning-ss21). These include his code for the standard LeNet-5 and AlexNet architecture as well as his helper files (helper_evaluation, helper_train, and helper_plotting).

4.3. Hardware

Code was executed on a laptop running windows. The code was implemented utilizing a GTX 980m nvidia graphics card.

5. Results and Discussion

The group learned from this experiment that not all data sets are equal. It was much easier to develop models in order to analyze things like the mnist data set. Using data from other untested sources can require a lot more tinkering and design time in order to achieve a model with a high classification accuracy. Just looking at the images without the use of any algorithms there weren't any features that appeared that gave clear indications of COVID. No one in the group is experienced with examining CT scans and a trained professional might be able to spot something that the group did not. After trying many different architectures with very poor accuracy a few conclusions could be drawn.

CT scans of COVID patients do not have notable features. This could the case as the group could not distinguish any notable features, however to reiterate the group does not contain medical experts. Additionally, the sample only contained 742 samples. Other experiments attempting to do similar research used x-rays of chest scans which produced completely different images and therefore the group could not find multiple sources to supplement the initial data. With more time the group could have researched networks and strategies to maximize results with smaller sample sizes.

The project undertaken here was very different from the conditions of what was done for homework and in class. In class students knew what to expect from the data and it was already known what architecture to use. The project highlighted that while the group knew about the different architectures there was not a fluency to that knowledge that allowed for and easy solution to the problem. It would be interesting to see how quickly a more experienced machine learning engineer would be able to generate a model that was better than the one made by this group. It was interesting that most models that were tested preformed worse than baseline percentage of COVID to non-COVID patients.

```
Total Training Time: 1.16 min
Test accuracy 36.49%
```

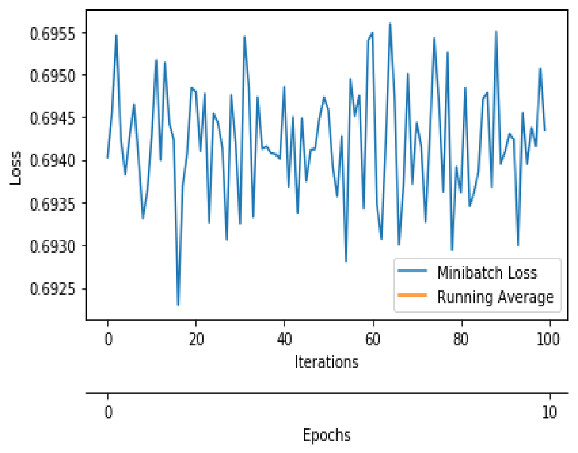
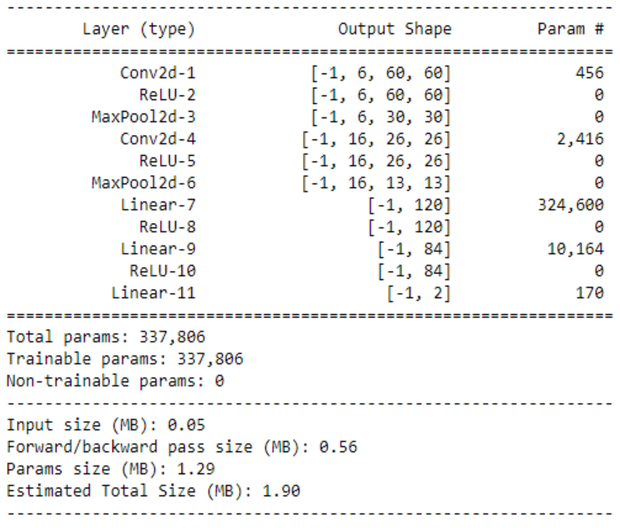


Figure 6. Performance graph for the data using Tanh and Adam



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 60, 60]             456
              ReLU-2            [-1, 6, 60, 60]               0
         MaxPool2d-3            [-1, 6, 30, 30]               0
            Conv2d-4           [-1, 16, 26, 26]           2,416
              ReLU-5           [-1, 16, 26, 26]               0
         MaxPool2d-6           [-1, 16, 13, 13]               0
            Linear-7                  [-1, 120]         324,600
              ReLU-8                  [-1, 120]               0
            Linear-9                   [-1, 84]          10,164
             ReLU-10                   [-1, 84]               0
           Linear-11                    [-1, 2]             170
================================================================
Total params: 337,806
Trainable params: 337,806
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.56
Params size (MB): 1.29
Estimated Total Size (MB): 1.90
----------------------------------------------------------------
```

Figure 7. Model summary for figure 5

As seen in figure 6 the lenet architecture that used the Tanh activation function and an Adam optimizer performed far worse the the default 54 percent. Why did this model perform so poorly? One problem with trying to find solutions to make a network better for this project is that is difficult to know what the features are that the model is picking out as important. Much of the image is black with white dots, not to mention what the model is using is vectors so it difficult to distinguish what the models are determining as important for classification. The fact the most models performed better with normalized would make it seem that the small spots with high contrast to nearby spots on the scans aren't important however the best model by far, shown in figure 5, was generated using data that was not normalized. Figure 7 shows the model summary for this model. It shows the number of inputs and their size and shape at each layer of the network.

A project that would be interesting to see is a program that once the type of data being read in is indicated it ran through and exhaustive list of settings in order to find the best model. The compute time on the would be large however.

6. Conclusions

In the end the group was not able to create a model that would help ease the burden of COVID testing in the United States. In reality with emerging strains and the large inconsistency of symptoms among patients a model that would be able to reliably detect COVID is probably not feasible. As more data becomes available maybe methods will emerge that are faster, cheaper, and more accurate for testing. The biggest limiter of this project was time. With only one group member coding the progress of the model was severely hindered. However, every group member learned a lot about what it takes to actually do data science.

7. Acknowledgements

A big thank you to Professor Sebastian Raschka for his lectures (STAT 453), which were very helpful. Code was adapted from his lectures and from his feedback on the project proposal. The time that he spent to helping with debugging is also really appreciated. In addition, the Teaching Assistant, Zhongjie Yu, helped with other debugging challenges.

8. Contributions

James found and cleaned up the data set. James put together the initial code for the network and after Saniya and James went to office hours James then executed and made all changes to the code. The abstract and introduction were worked on equally by everyone. Related work, proposed method, and dataset was primarily wokred on by Steven and Saniya. James wrote and produced figures for the experiment section as well as wrote the results and discussion. James wrote the conclusion. The slides were evenly divided and commentated on by each group member.

References

[1] M. Dai, L. Ouyang, F. Yang, H. Shi, J. Wang, X. Han, O. Alwalid, Y. Cao, D. Yang, and Y. Li. Chest ct imaging features of typical covert covid-19 cases. *International Journal of Medical Sciences*, 18(10):2128–2136, 2021.

[2] K. Fiore. Hazy on ground-glass opacities? here's what they are, May 2020.

[3] A. E. Krizhevsky, I. E. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[4] T. C. Kwee and R. M. Kwee. Chest ct in covid-19: What the radiologist needs to know. *RadioGraphics*, 40(7):1848–1865, 2020.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] M. Loey. Covid-19 chest ct image augmentation gan dataset, Jan 2021.

[7] J. Redmon. Yolo: Real-time object detection, Jan 2021.

[8] A. Ulhaq, J. Born, A. Khan, D. P. S. Gomes, S. Chakraborty, and M. Paul. Covid-19 control by computer vision approaches: A survey. *IEEE Access*, 8:179437–179456, 2020.

[9] L. Wang, Z. Q. Lin, and A. Wong. Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1), 2020.