How Does Image Size or a Skip Connection Impact a Neural Network's Effectiveness at Diagnosing COVID-19 from CT Scans?

Author*: Ethan Biegeleisen
Advisor†: Dr. Angel R. Pineda
Department of Mathematics, Manhattan College, Bronx, NY

December 15, 2021

Abstract

This project utilizes convolutional neural networks and residual networks to diagnose whether a patient has COVID-19 from a CT scan of the patient's lungs. The networks are trained and tested on a data set consisting of CT scans from patients with COVID-19 and CT scans from patients without COVID-19. The results from training and testing with each network are compared with each other to determine if a standard network or a residual network is more effective for this application. The process is repeated using smaller versions of the scans to determine which image size is necessary for either network to make an effective diagnosis.

Introduction

Computer vision is a field with numerous potential applications. One such use is diagnosing patients from medical imaging. With COVID-19 unfortunately becoming a part of everyday life, having additional techniques for diagnosing patients quickly and effectively remains a priority. The question is not whether a neural network can properly diagnose COVID-19 from medical imaging, but which structures for neural networks provide the most accurate diagnoses. Two specific types of models are used for this project. The first is a standard convolutional neural network (CNN). This is a type of neural network that is traditionally used for image classification. This is compared with a residual network (ResNet) that utilizes a skip connection. Both of these networks are trained and tested using parts of a CT scan data set that contains lung images of patients with COVID-19 and patients without COVID-19. An additional question is how large these CT scans need to be in order for either network to accurately diagnose patients. The training and testing is repeated for the scans at different image sizes to determine how small medical imaging could be for making a proper diagnosis.

*ebiegeleisen01@manhattan.edu
†angel.pineda@manhattan.edu

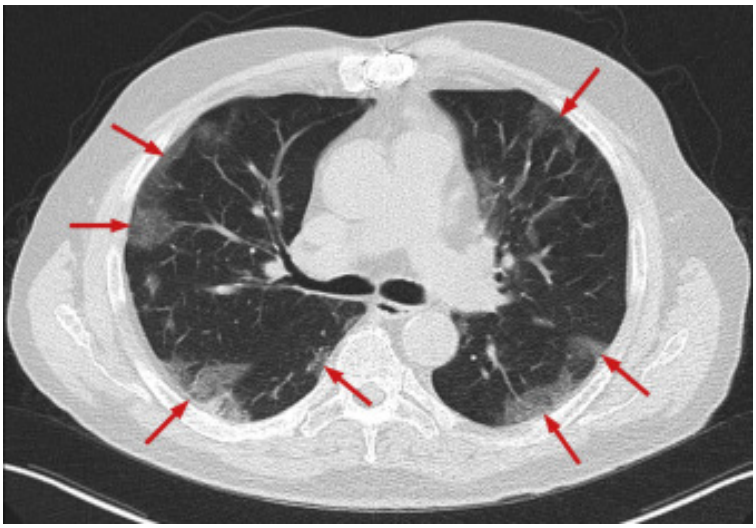


Figure 1: Medical imaging of a patient with COVID-19. The red arrows highlight ground-glass opacities visible in the lungs [1].

Background

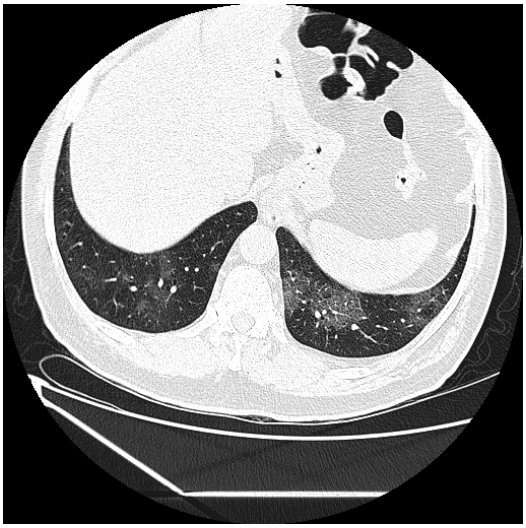
Lung CT scans of patients with COVID-19 often contain ground-glass opacities [1]. These show in the medical imaging as grey sections inside the lungs [2]. A normal lung scan would be black in those sections, while a lung that is heavily diseased would be white from the infection filling the lung with pus or fluid [2]. Figure 1 highlights the ground-glass opacities found in the medical imaging of a patient with COVID-19 [1].

The data set for this project was provided by M. Aria, M. Ghaderzadeh, F. Asadi, and R. Jafari on Kaggle [3]. The data contains 7,495 lung CT scans of patients who have COVID-19 and 944 lung CT scans of patients who do not have COVID-19. Each image has 3 slices of 512 pixels by 512 pixels, with the COVID-19 positive images and the COVID-19 negative images being provided in separate folders. Aside from the folders and the image names, the scans do not include any labels. Figure 2 shows a COVID-19 positive CT scan and a COVID-19 negative CT scan from the data set [3].

Methods

A CNN is a neural network that utilizes convolutional layers, pooling layers, and fully connected layers. A convolutional layer involves placing at least one convolutional filter over a section of the input array, conducting element wise multiplication between the filter and the input section, and summing the results to calculate the corresponding entry in the output. The filter is then moved to cover a different section of the input and the process repeats. The length and width of the filter is represented as f while the stride of the filter, which refers to how much the filter is moved over the input at each step, is represented as s. Figure 3 shows an example of a convolutional layer [4].

Pooling layers are more straightforward than convolutional layers. A max pooling layer involves placing a filter over a section of the input, and the entry in that section of the input with the highest value becomes the entry in the corresponding section of the output. Figure 4 shows an



Positive

Negative

Figure 2: On the left is a CT scan from the data set showing the lungs of a patient infected with COVID-19 [3]. On the right is a CT scan from the data set showing the lungs of a patient without COVID-19 [3].

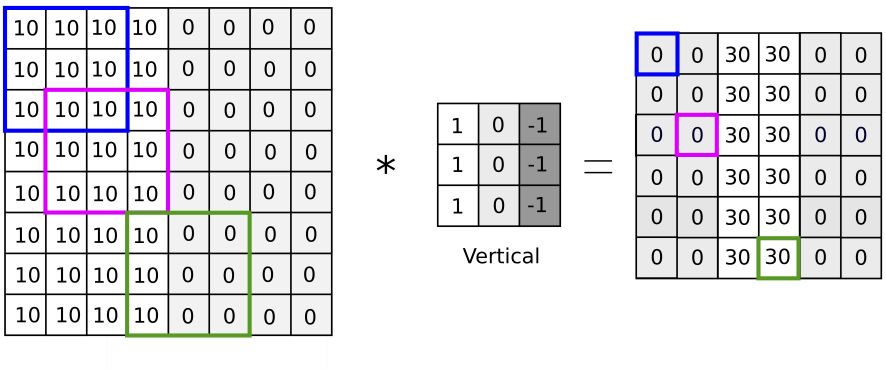


Figure 3: The input on the left is run through a convolutional filter in the middle for detecting vertical edges, resulting in the output on the right [4].

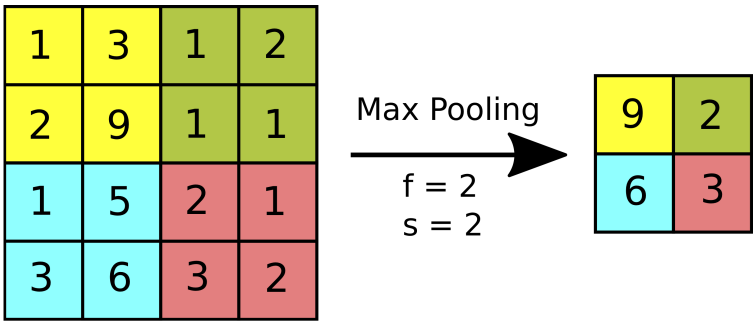


Figure 4: The input on the left is run through a max pooling filter with length 2 and stride 2, resulting in the output on the right [4].

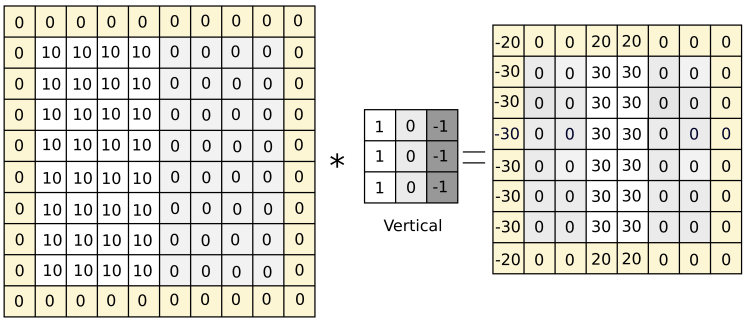


Figure 5: The input on the left is run through a convolutional filter in the middle for detecting vertical edges, resulting in the output on the right. Notice the output has the same dimension as the input did before zero padding was applied [4].

example of a max pooling layer [4]. An average pooling layer instead uses the mean of all entries in that section of the input as the entry in the corresponding section of the output. Potential issues with filters are that they can shrink the size of the input too quickly and that information from the outer edges of the input can be lost early in the network. One solution, known as zero padding, is to add one or more layers of entries consisting of 0 around the perimeter of the input. Figure 5 shows an example of zero padding [4].

At least one fully connected layer is found in the deepest part of the network. The input array is first flattened into a one dimensional vector before it can run through the first fully connected layer. Unlike convolutional layers or pooling layers, fully connected layers utilize every individual input for calculating every individual node. Figure 6 shows a fully connected layer [5]. The calculations for each input i at each node l are the following:

$$z_i^{[l]} = w_i^{[l]T} x + b_i^{[l]} \tag{1}$$

$$a_i^{[l]} = activation(z_i^{[l]}) \tag{2}$$

w and b are weight and bias parameters, while *activation* refers to a nonlinear activation function like sigmoid, hyperbolic tangent, or Rectified Linear Unit (ReLU).

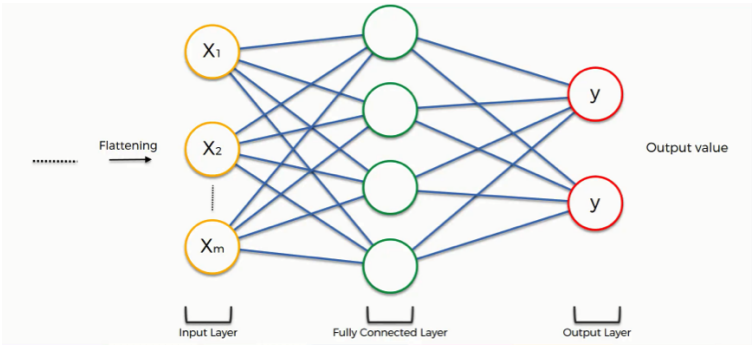


Figure 6: A fully connected layer in a CNN [5].

Everything discussed so far is part of the forward propagation process. After an iteration of every layer in the network during training, the cost is calculated. Its purpose is to measure the difference between the network's predicted classifications for the training set and the actual classifications for the training set. The cost is then used to initiate an iteration of backpropagation, which is a series of derivations starting in the deepest layer of the network and ending in the outermost layer of the network. The derivatives that result are used to update the network's parameters like the entries in convolutional filters and weight and bias parameters in the fully connected layers. These updated parameters are then used for the next iteration of forward propagation.

ResNets have the same general structure as CNNs aside from one key difference. ResNets include skip connections where the input for a block of layers gets stored and added to the block's output. A CNN needs to keep track of initial information from the input while also keeping track of the changes made from the following layers of the network. A CNN can have trouble with this task when a large number of layers is involved. Since a ResNet has information from the input stored in a skip connection, it only needs to focus on keeping track of the later changes made to the input. Figure 7 shows the difference between a block in a standard network and a block in a ResNet [6].

Results

All coding was done in Python using Jupyter Notebook, with the neural networks utilizing the tensorflow and keras libraries. The CT scans were first imported as images of size 64 x 64 x 3 and converted into arrays. Labels were then generated to indicate whether or not each patient had COVID-19, and the data was shuffled into training and test sets. The initial CNN and ResNet models implemented were based on models used during assignments earlier in this course [7]. The main difficulties running the CNN model involved getting the tensorflow and keras libraries installed without any errors and properly organizing the data set for compatibility with the model. This is why the initial image size used was 64 x 64. The ResNet model required more adjustments to run properly that mainly involved changing it from a categorical classifier to a binary classifier. The number of classes in the model was reduced, the final activation layer was changed from softmax to sigmoid, and the loss calculation was changed from categorical cross entropy to binary cross entropy. The models were then running with the data set, but any training took a long time. From this point onward, 944 COVID-19 positive CT scans were used instead of all 7,945 positive scans. Training became much faster and the results did not appear to be impacted by the reduction of positive images. This also allowed for an equal amount of positive and negative images to be used from the

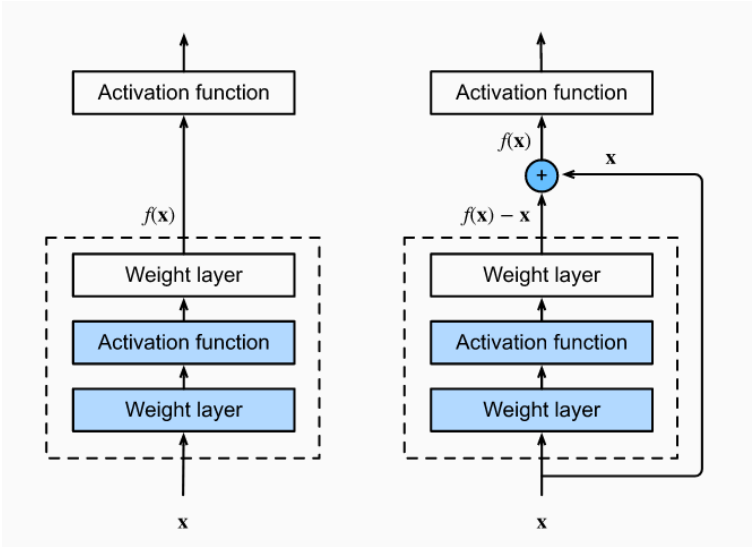
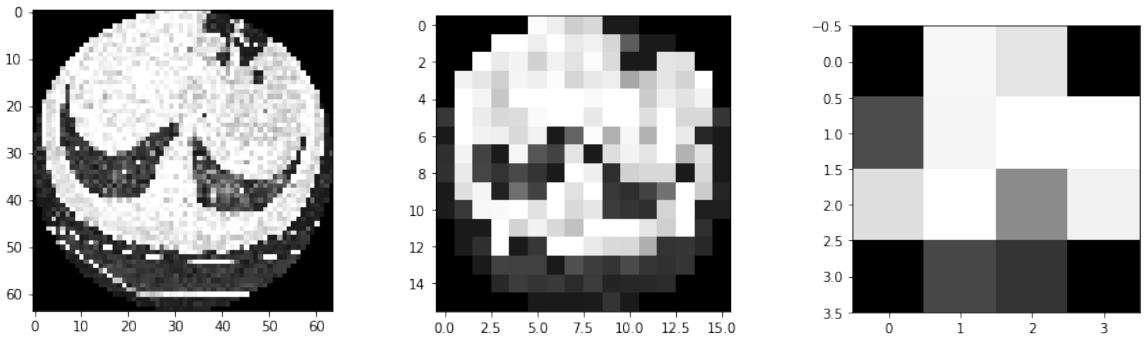


Figure 7: A block from a standard neural network on the left and a block from a ResNet on the right [6].

data set instead of the positive images heavily outnumbering the negative images.

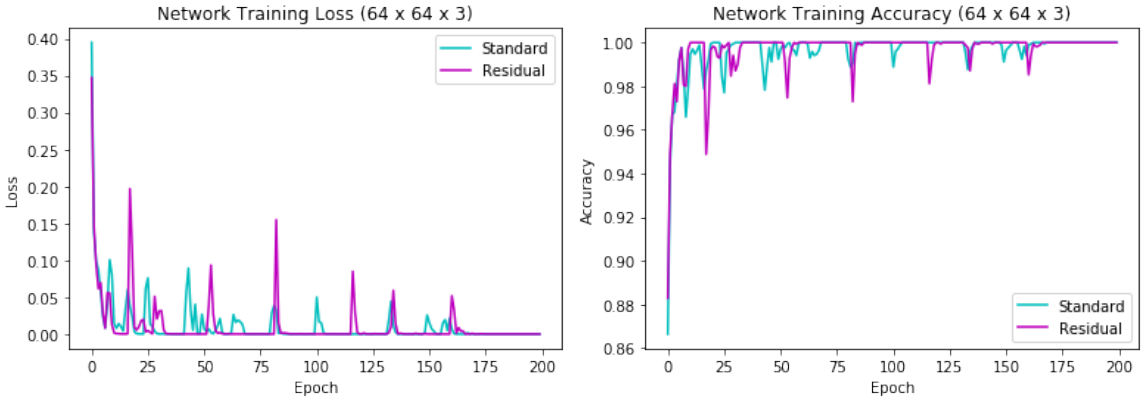
After obtaining initial results, the next step was to use the networks with different image sizes as inputs. Progressively smaller image sizes were attempted first, but later work was attempted with larger image sizes as well. Figure 8 shows smaller image sizes of the positive CT scan from Figure 2 [3]. The only change the CNN needed was the input size of the image array, but the ResNet required additional adjustments. Average pooling and max pooling layers deeper in the network ran into errors when smaller image sizes were used as inputs. By the time the inputs reached those layers, the input size was too small for the filters to work properly. The first attempt at fixing this was to shrink the filter size when smaller images were used. This let the model run without errors, but the input was still shrinking in surface size quickly after the initial layers. The filter size changes were undone, and instead the initial zero padding layer was increased in size as smaller images were used as inputs. This allowed the ResNets to run properly without the input's surface drastically shrinking from the first few layers. For image sizes larger than 64 x 64, the amount of zero padding was unchanged while the filter size of the final average pooling layer was increased so the output layer would have the same surface size before getting flattened for the fully connected layer. Unfortunately, any attempt at using images with size 512 x 512 led to immediate freezing so the largest image size used was 256 x 256 instead.

After these changes, there were still issues with the networks. The CNN could train fairly quick, but the ResNet took much longer to train for even the smallest image sizes. Even running 3 epochs with the ResNet for any image size took longer than running 10 epochs with the CNN for any image size. This was likely due to the ResNet having many more layers than the CNN, which tied back to another problem. With the CNN and the ResNet using substantially different structures, the project was not properly comparing the effectiveness of the networks. The solution was to redo both networks so they could be as close to each other as possible while also having a training speed close to the prior CNN. The ResNet that had been utilized for the 64 x 64 images was used as a basis, and most of its repeated layers were removed to bring its size closer to the prior



64 x 64 16 x 16 4 x 4

Figure 8: The positive CT scan from Figure 2 [3] scaled to smaller image sizes.

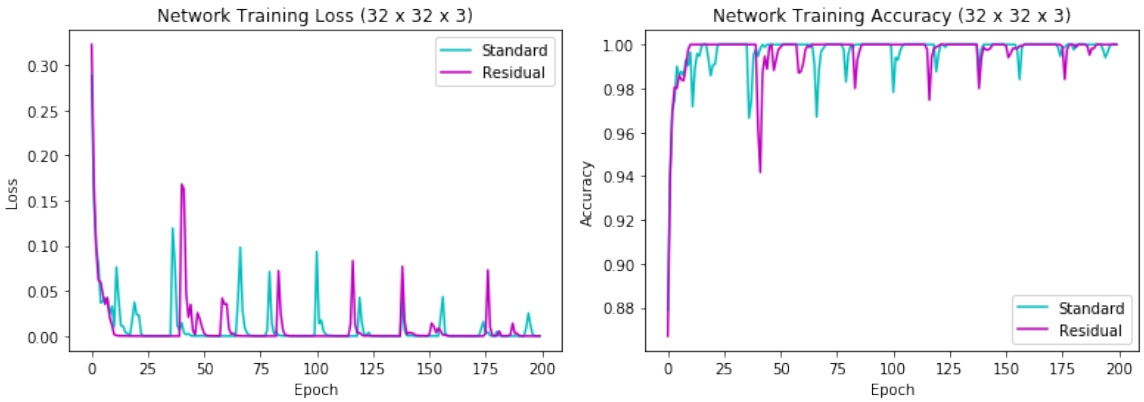


Training Loss Training Accuracy

Figure 9: Training loss on the left and training accuracy on the right using a CNN (teal) and a ResNet (purple) with 64 x 64 image sizes.

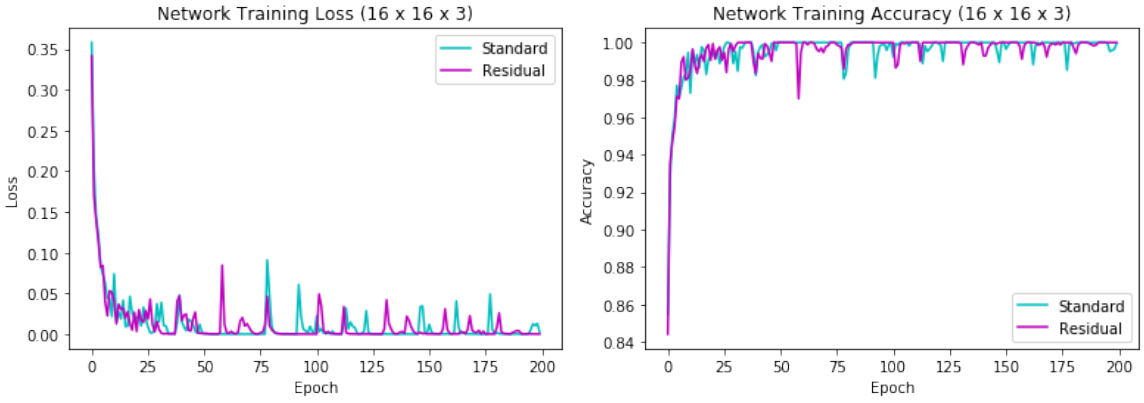
CNN. This new ResNet was then copied as a basis for the new CNN. The only additional change made for the new CNN was removing the skip connection. Aside from changing the input size, the only necessary adjustment for inputs of different image sizes was increasing the initial zero padding for both networks as the image size decreased.

The new CNN and ResNet were able to train on the 64 x 64 data set for 200 epochs. The training results are shown in Figure 9. This process was then repeated using smaller image sizes of 32 x 32 (Figure 10), 16 x 16 (Figure 11), 8 x 8 (Figure 12), 4 x 4 (Figure 13), and 2 x 2 (Figure 14). The networks were able to train on 128 x 128 and 256 x 256 image sizes as well, but the test results with these image sizes were less consistent. The test results for image sizes 64 x 64 and smaller using both the CNNs and the ResNets are shown in Figure 15 and Table 1.



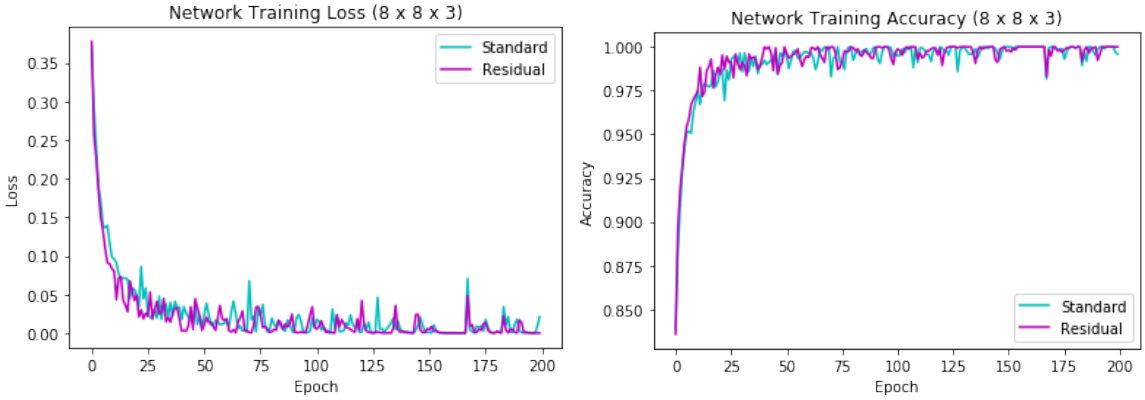
Training Loss Training Accuracy

Figure 10: Training loss on the left and training accuracy on the right using a CNN (teal) and a ResNet (purple) with 32 x 32 image sizes.



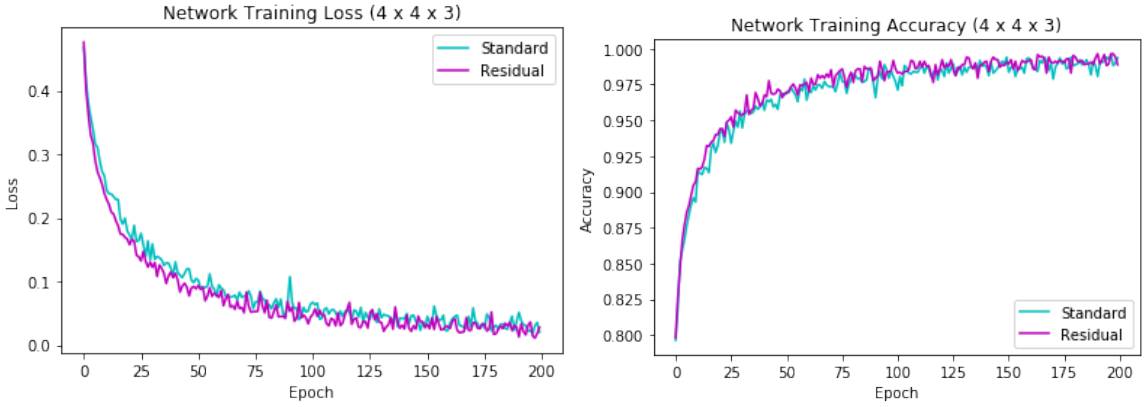
Training Loss Training Accuracy

Figure 11: Training loss on the left and training accuracy on the right using a CNN (teal) and a ResNet (purple) with 16 x 16 image sizes.



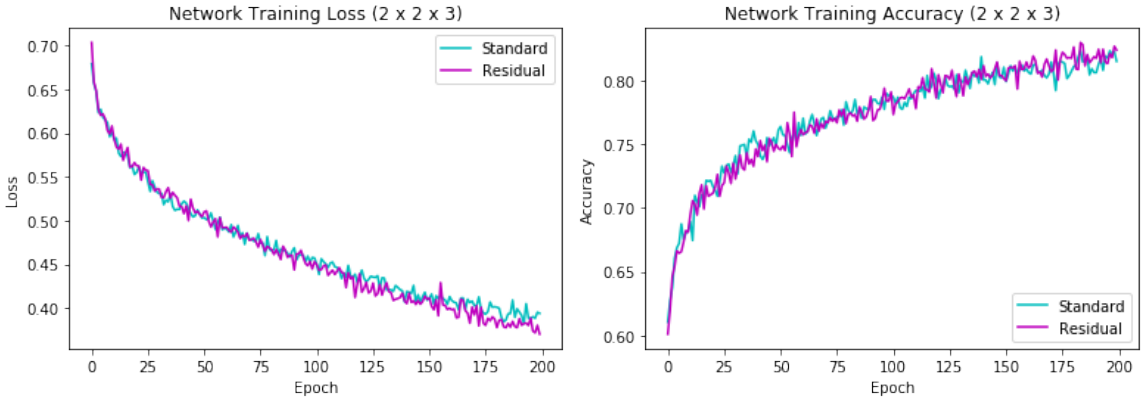
Training Loss Training Accuracy

Figure 12: Training loss on the left and training accuracy on the right using a CNN (teal) and a ResNet (purple) with 8 x 8 image sizes.



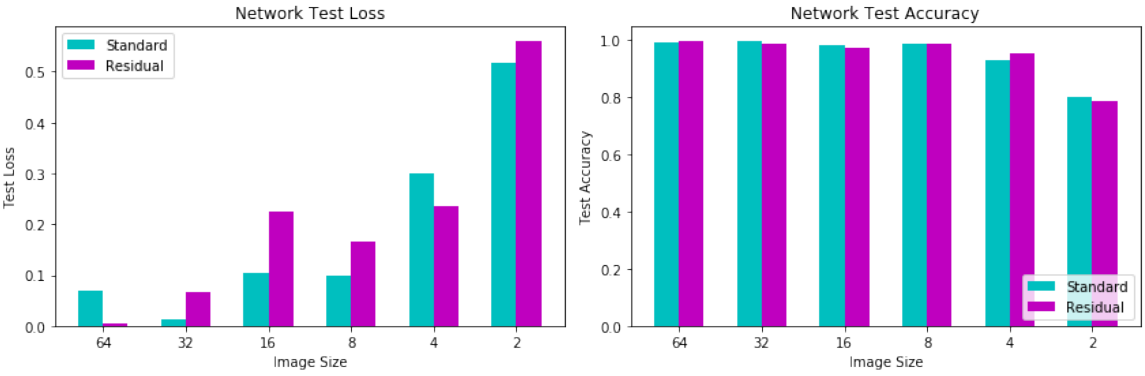
Training Loss Training Accuracy

Figure 13: Training loss on the left and training accuracy on the right using a CNN (teal) and a ResNet (purple) with 4 x 4 image sizes.



Training Loss Training Accuracy

Figure 14: Training loss on the left and training accuracy on the right using a CNN (teal) and a ResNet (purple) with 2 x 2 image sizes.



Test Loss Test Accuracy

Figure 15: Test loss on the left and test accuracy on the right using CNNs (teal) and ResNets (purple) with each image size.

Image Size	Loss (CNN)	Accuracy (CNN)	Loss (ResNet)	Accuracy (ResNet)
64 x 64	0.068	0.989	0.006	0.995
32 x 32	0.012	0.995	0.066	0.984
16 x 16	0.105	0.979	0.224	0.968
8 x 8	0.099	0.984	0.166	0.984
4 x 4	0.301	0.926	0.235	0.952
2 x 2	0.516	0.799	0.561	0.783

Table 1: Test loss and test accuracy using CNNs and ResNets with each image size.

Discussion

Figure 9 shows that for both networks using 64 x 64 images, the training loss approached 0 and the training accuracy approached 1 as the training continued. Neither network immediately appeared to have performed better than the other at this image size. However, both networks had spikes during training with the ResNet experiencing the largest spikes. This appears to be related to the mini batch size used during training. When larger mini batch sizes are used, the network can continue to move back and forth around the minimum cost instead of consistently remaining at the minimum cost. The mini batch size used during training was 8. Smaller mini batch sizes were also attempted and the training results had smaller spikes, but performance with the test set appeared to work best with the mini batch size of 8. In addition, the spikes might appear to have been a bit more drastic than they actually were due the plot windows. When the largest spike occurred with the ResNet, the model's training accuracy was approximately 0.95.

Table 1 shows that after training both networks using 64 x 64 images, the test loss was 0.068 using the CNN and 0.006 using the ResNet. Although the ResNet had a substantially lower test loss, the CNN's test loss of 0.068 does not appear to be particularly poor. Using the 64 x 64 images, the test accuracy was 0.989 using the CNN and 0.995 using the ResNet. These test accuracy results from the two networks are much closer than the test loss results. For this image size, the ResNet appears to have performed a bit better with the test set overall than the CNN. However, both networks seemed to perform well.

Figure 10 shows that training the networks using 32 x 32 images had similar training results to using 64 x 64 images, with the CNN and the ResNet having a similar level of performance to each other here as well. The main difference using this image size is that there seemed to be more spikes during training when compared to using the 64 x 64 images. At the 32 x 32 image size, the spikes were still occurring near the final epochs as well. Figure 11 shows that there were noticeably more spikes when training the networks using 16 x 16 images. At this image size, the largest spikes were roughly half the size of the largest spike from using 64 x 64 images. Using 16 x 16 images, the general trends during training, the training loss at the final epoch, and the training accuracy at the final epoch for both networks appeared to have similar results to using larger image sizes.

Figure 12 shows that training the networks using 8 x 8 images yielded similar results to training with 16 x 16 images, except the number of spikes during training continued to increase as the image size decreased. However, the final epoch of the CNN occurred during a training spike with a training loss close to 0.04 and a training accuracy close to 0.99. Despite this, the training loss for both networks was still trending towards 0 and the training accuracy for both networks was still trending towards 1. Figure 13 shows that using 4 x 4 images had similar trends to using 8 x 8 images, but the training loss and training accuracy on the final epoch started to change more noticeably. When 4 x 4 images were used, the training loss for both networks was close to 0.05 and the training accuracy was close to 0.99 during the final epoch. These results were similar to the final

epoch of the CNN using 8 x 8 images, except they did not appear to occur during a training spike. Figure 14 shows that decreasing the image size from 4 x 4 to 2 x 2 caused a more drastic shift in the training results. When 2 x 2 images were used, the training loss for both networks was close to 0.38 and the training accuracy for both networks was close to 0.82 during the final epoch. The overall slope when training with 2 x 2 images also appeared to be more gradual in comparison to the overall slope when training with 4 x 4 images.

Figure 15 and Table 1 show the loss and accuracy of each network on the test set using each image size. Aside from the CNN with 64 x 64 images and both networks with 8 x 8 images, the test loss appeared to gradually increase overall for both networks as the image size decreased. When the image size decreased from 4 x 4 to 2 x 2, test loss appeared to experience a sharper increase for both networks. One observation of note is that the test loss for the CNN was similar at the 16 x 16 and 8 x 8 image sizes, while the test loss for the ResNet was similar at the 16 x 16 and 4 x 4 image sizes. The test accuracy appeared to remain relatively flat for the larger image sizes used with each network. As the image size continued to decrease from 16 x 16 to 8 x 8, the test accuracy went from slightly decreasing and slightly increasing. When the image size decreased from 8 x 8 to 4 x 4 and 2 x 2, the test accuracy for both networks decreased more consistently. The sharpest decrease in test accuracy occurred for both networks with the 2 x 2 image size. In general, both networks appeared to perform similarly. There was not one type of network that had consistently better test results than the other.

One detail that stands out in Table 1 is how close the test accuracy was when using image sizes of 32 x 32, 16 x 16, and 8 x 8 for both networks. The lowest test accuracy at any of these image sizes was 0.968 for the ResNet using 16 x 16 images. The test accuracy for both networks using 8 x 8 images was 0.984, which was higher than the test accuracy for either network using 16 x 16 images. In addition, the test accuracy for the CNN was slightly higher using 32 x 32 images than 64 x 64 images. In general, both networks appeared to hold up well when using image sizes as low as 8 x 8. Even with 4 x 4 images, however, both networks had a test accuracy greater than 0.92. The networks did not appear to have major issues until the image size was brought down to 2 x 2.

One limitation of these results is that each individual network was only able to be trained and tested once when using 200 epochs. A more ideal situation might involve training and testing each type of network using each image size hundreds of times, recording the test loss and test accuracy at each instance, and calculating the mean and standard deviation for the test loss and test accuracy for each type of network with each image size. This might allow for more reliable results on the overall performance of the networks.

Conclusions

The CNN and the ResNet both appeared to perform well at using lung CT scans to determine whether patients had COVID-19. The skip connection did not appear to substantially help or hinder the ResNet at this task when compared to the CNN. This might be related to the networks not having a large number of layers. If deeper networks were used, the use of skip connections might have given the ResNet a visible advantage over the CNN. As the image size decreased, both networks retained surprisingly robust diagnostic accuracy overall. If a health care provider was looking for an automated method of diagnosing COVID-19 with an accuracy of 0.98, CT scans could potentially be utilized with image sizes as low as 8 x 8 pixels instead of 512 x 512 pixels. This could expose patients and health care workers to less radiation while also saving a substantial amount of storage space in patient databases.

References

[1] Adams, H. J.A., Kwee T. C., Yakar D., Hope M. D., Kwee R. M. (2020). Chest CT imaging signature of coronavirus disease 2019 infection: In pursuit of the scientific evidence. *Chest, 158*(5), 1885-1895. https://doi.org/10.1016/j.chest.2020.06.025.

[2] Fiore, K. (2020). Hazy on ground-glass opacities? Here's what they are. https://www.medpagetoday.com/pulmonology/generalpulmonary/86751.

[3] Aria M., Ghaderzadeh M., Asadi F., Jafari R. (2021) COVID-19 Lung CT Scans: A large dataset of lung CT scans for COVID-19 (SARS-CoV-2) detection. Kaggle. DOI: 10.34740/kaggle/dsv/1875670.

[4] Reynolds A. (2019). Convolutional neural networks (CNNs). https://anhreynolds.com/blogs/cnn.html.

[5] SuperDataScience Team (2018). Convolutional neural networks (CNN): Step 4 - full connection. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection.

[6] Dive Into Deep Learning (ND). 7.6. Residual networks (ResNet). https://d2l.ai/chapter_convolutional-modern/resnet.html#fig-residual-block.

[7] Ng A., Katanforoosh K., Mourri Y. B. (2021). Deep learning specialization. DeepLearning.AI. https://www.coursera.org/specializations/deep-learning.