4-2021

Using Machine Learning for Detection of Covid-19

Justin Rickert
Grand Valley State University

Using Machine Learning for Detection of Covid-19

Justin Rickert
Advisor: Dr. Greg Wolffe

Abstract

Currently, the most widely used diagnostic tool for COVID-19 is the RT-PCR nasal swab test recommended by the CDC. However, some studies have shown that chest CT scans have the potential to be more accurate and are also capable of detecting the virus in its earlier stages. Unfortunately, CT results are not instantaneously available as it may be days before a radiologist can review the scan. This delay is one of the factors preventing the widespread use of CT scans for COVID detection. To address the delay, this project investigated Convolutional Neural Networks, an advanced form of machine learning used for image classification. CNNs have proven very effective at extracting patterns from images and have been used to detect clinical signs of COVID. The goal of this project was to develop an improved CNN that could accurately predict whether a patient is COVID-19 positive based on their CT scan. This could potentially provide a valuable pre-screening tool for overwhelmed radiologists.

1. Background Information

Convolutional Neural Networks (CNNs) are extremely effective and popular as a tool for image classification. These networks are generally composed of two main sections: a hierarchy of Convolutional and Pooling Layers and a Dense Neural Network. The input image is first passed through the Convolutional and Pooling Layers to extract the important features from an input image. The feature map output of the final layer is then passed into a Dense Neural Network for classification.

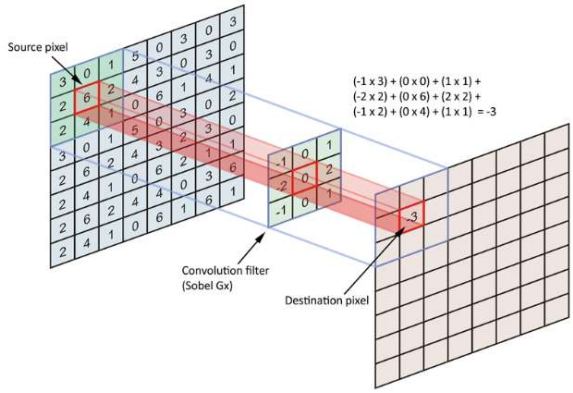

Figure 1: Convolutional Filter (Dertat, 2017)

Convolutional Layers extract the important features in an image using a series of filters that traverses all the pixels in an input image. New values are output at each step in the traversal that signify whether a specific feature is present in that section of the image (see Fig. 1). Using stacked Convolutional Layers, highly semantic features can be extracted in hierarchical and increasingly abstract fashion. For example, the first Convolutional Layer might extract a simple feature like an edge, while the final Convolutional Layer detects the presence of a face.

Pooling Layers are used to condense the output feature maps of Convolutional Layers. This helps reduce the number of parameters present in the network, which reduces computational cost. Pooling Layers generally occur between groups of Convolutional Layers.

Dense Neural Networks perform the classification on the final feature map output by the Convolutional and Pooling Layers. These neural networks are composed of layers of neurons. Each neuron is connected to all of the neurons in the previous layer (see Fig. 2), and

each connection is given an initial weight value. The output value of a node is calculated by passing the big sum of each previous nodes weighted value into an activation function (as given in Eq. 1).

Equation 1: Neuron Output Value

$$y = f_n(b + \sum_{i=1}^{d} w_i x_i)$$

To perform a classification all neurons' values propagate through the network until reaching the output layer. The output node with the largest value represents the final classification.

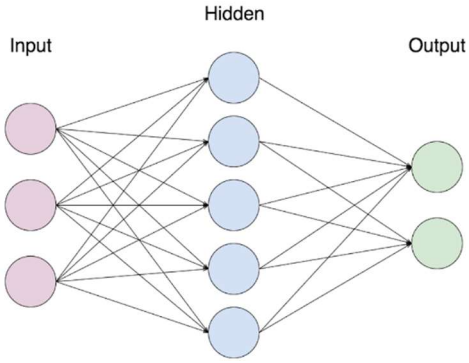

Figure 2: Dense Neural Network (Wiederer, 2016)

When a misclassification occurs, the errors are backpropagated through the network and used to update the weights between each neuron. The updated value for each weight is calculated by multiplying the Learning Rate, the difference between target and output values, and the input value (see Eq. 2).

Equation 2: Backpropagation Function

$$\Delta w_{ij} = \eta \left(t_j - y_j\right) x_i$$

This process iteratively adjusts edge weights so that future classifications will be correct. After enough misclassifications, the weights become optimized, maximizing the accuracy of the network's predictions.

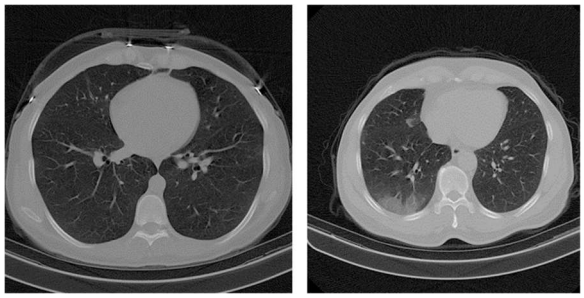

Figure 3: Normal CT (left) vs Covid CT (right)

2. Methodology

The dataset contained 12,058 Chest CT-scan images: 2,282 of these images were Covid-positive, while 9,776 were not (see Fig. 3). The dataset was split into two subsets: a training set of 4,000 images was split 50:50 between Covid and Normal scans, as was the validation set of size 500.

The network was comprised of four main pieces. First, a Normalization Layer recomputed all pixel values in the input image onto a (-1 to 1) interval. These values were then passed to ResNet50V2 (a prebuilt CNN architecture). Third, a Global Average Pooling Layer transformed the final feature map into an array of average values for each feature. Finally, a Dense Neural Network, with no hidden layers, determined the classification using the values output by the Global Average Pooling layer

Although the number of images was sufficient for training the Dense Neural Network classifier, it was not large enough to train the fifty Convolutional Layers built into ResNet50V2. This problem was solved using Transfer Learning. The ResNet50V2 network was pretrained on the ImageNet dataset, which contains over 14 million images and 30,000 classification classes. The parameters of the pretrained model were frozen before training the Dense Neural Network.

3. Experiments

All experiments ran for 100 epochs, a process that generally took 1.5 hours. The software used was Tensorflow 2.4.1 with Python 3.7 running on a Threadripper 3960x CPU, an RTX 2070 Super GPU, and 128GB of DDR4 RAM. Two hyperparameters of the model were investigated: the activation function and the learning rate.

3.1. Activation Functions

The activation function affects how the output value is ultimately calculated. Experiments used the Linear, Softmax, ReLu, and Leaky ReLu activation functions on the output nodes of the Dense Neural Network. The input value for each of these activation functions is the big sum of the weighted values of all the nodes in the previous layer.

Equation 3: Neuron Sigma Value

$$\sigma = b + \sum_{i=1}^{d} w_i x_i$$

The Linear activation function simply returns the raw input value. The Softmax activation function normalizes all nodes on a probability distribution on the interval 0 to 1. The ReLu activation function either returns the max of the input value, or zero. The Leaky ReLu activation function operates similarly to ReLu but has a slight gradient in the negative range.

Each experiment was repeated twice using two separately-trained networks running for 100 epochs with a learning rate of 0.001. The classification accuracy on the validation set was logged after each epoch.

Activation Function	Max Accuracy
Linear	0.94
Softmax	0.94
ReLu	0.91
Leaky ReLu	0.94

There was little to no significant difference in accuracy between the Linear, Softmax, and Leaky ReLu activation functions. All three functions produced a network with an overall accuracy of 94% on the validation set. However, training with the ReLu activation function produced a less accurate model of 91% accuracy.

This highlighted a larger issue occurring with the ReLu activation function. One of the trained networks maxed out at 50% accuracy, suggesting that the network was making the same prediction regardless of the input (recall that the validation set was split exactly 50:50). This substantial decrease in accuracy was likely caused by a problem called Dying ReLu. If a network initializes with a strong negative bias, both output nodes in the Neural Network will have a value of 0. This prevents the network from training its weights and making an intelligent classification.

Overall, it appeared that the Linear, Softmax, and Leaky ReLu activation functions were equally suitable for this classification task.

3.2. Learning Rates

The learning rate of a model affects how drastically the weights of a Neural Network are altered when backpropagating errors. This parameter is the η value in the backpropagation function (Eq. 2). It is typically a very small value on the interval 0 to 1, with the TensorFlow default being 0.001.

Choosing a learning rate is a balancing act. A learning rate that is too high will cause erratic changes in network weights, causing accuracy to be unstable over time and making it difficult for the network to optimally converge. Conversely, a learning rate that is too low will cause the network's validation accuracy to plateau before perfectly optimizing the

network's weights. A well-chosen learning rate should mitigate both of these issues.

Again, each learning rate experiment repeated network training for 100 epochs with a linear activation function. Each network's classification accuracy on the validation set was logged after each epoch.

Learning Rate	Max Accuracy
0.1	0.91
0.01	0.94
0.001	0.93
0.0001	0.89
0.00001	0.80

When looking at max accuracy on the validation set, the 0.01 and 0.001 learning rates produced the most accurate networks. However, when comparing the accuracy over time, the accuracy of the model that used a 0.01 learning rate was quite erratic (see Fig. 4). Thus, the 0.001 learning rate appeared to be optimal out of those tested.

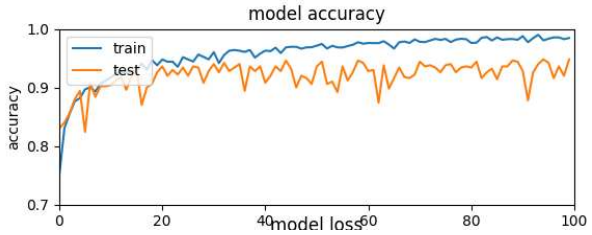

Figure 4: Learning Rate of 0.01; Accuracy Over Time

4. Findings

The most accurate network produced was trained for 100 epochs with a learning rate of 0.001 and the Leaky ReLu activation function. This network had a **max validation accuracy of 94.4%** and minimum loss value of 0.18. The model's accuracy over time can be seen in Figure 5.

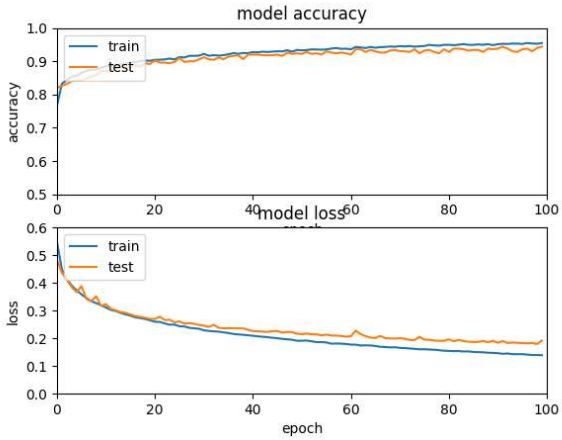

Figure 5: Best Network; Accuracy Over Time

5. Future Work

Future plans call for further test set analysis, utilizing the remaining CT scan images in the base dataset that currently are not being used in either the training or validation sets. This will allow for more detailed analytics as to where the network fails. For example, what percentage of misclassifications are due to false positives vs. false negatives.

Future experiments will also employ a Feature Pyramid Network (FPN) in place of a basic CNN. The main difference between an FPN and a CNN is that FPNs make multiple predictions, after the convolutional layers at different stages in the network, while CNNs make a single prediction after the final convolutional layer.

Bibliography

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature (London), 521(7553), 436-444. https://doi.org/10.1038/nature14539

Rahimzadeh, M., Attar, A., & Sakhaei, M. (2020). A Fully Automated Deep Learning-based Network For Detecting COVID-19 from a New And Large Lung CT Scan Dataset. Preprints 2020, 2020060031. https://doi.org/10.20944/preprints202006.0031.v3

Baykal, E., Dogan, H., Ercin, M. E, Ersoz, S., & Ekinci, M. (2020). Transfer learning with pre-trained deep convolutional neural networks for serous cell classification. Multimed Tools Appl, 79, 15593–15611. https://doi.org/10.1007/s11042-019-07821-9

Tang, Y., Huang, J., Zhang, f., & Gong, W. (2020). Deep residual networks with a fully connected reconstruction layer for single image super-resolution. Neurocomputing, 405, 186-199. https://doi.org/10.1016/j.neucom.2020.04.030

Suresh, R. & Keshava, N. (2019). A Survey of Popular Image and Text analysis Techniques. 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), 1-8. https://doi.org/10.1109/CSITSS47250.2019.9031023

Shao, J. M., Ayuso, S. A., Deerenberg, E. B., Elhage, S. A., Augenstein, V. A. & Heniford, B. T. (2020). A systematic review of CT chest in COVID-19 diagnosis and its potential application in a surgical setting. Colorectal Dis, 22, 993-1001. https://doi.org/10.1111/codi.15252

Bai, L., Zhao, Y., & Huang, X. (2018). A CNN Accelerator on FPGA Using Depthwise Separable Convolution. IEEE Transactions on Circuits and Systems II: Express Briefs, 65(10), 1415-1419. https://doi.org/10.1109/TCSII.2018.2865896

Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 936-944. https://doi.org/10.1109/CVPR.2017.106

Wiederer, T., (2016). “Neuronal Network Scheme”. Webkid, https://webkid.io/blog/neural-networks-in-javascript

Dertat, A., (2017). “Convolutional layer filter”. Towards Data Science, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2