A CNN Architecture with Adaptive Histogram Equalization on COVID-19 Prediction

Gökhan Koçmarlı *(150720821)*
gokhankocmarli@marun.edu.tr
Artificial Intelligence on Health Sciences Project (2022-2023 Fall)

***Abstract*—A deep learning model can achieve high accuracy results on classifying categories. In this paper, I have proposed a custom convolutional neural network with preprocessing input method to classify chest X-rays if they have COVID-19 disease. The preprocessing of the image input allow the model to have better contrast on the chest part – which allows it to classify cloudiness of viral infection in the lungs, better. The proposed model trained with 1200 CXR (chest x-ray) images, which are selected from 1000 COVID CXR (chest x-ray) and 100 non-COVID CXRs. These non-COVID CXRs includes normal patients' and viral pneumonia patients'. The accuracy on the test data found as 87.68% accuracy, which is not good as other state-of-art networks.**

***Index Terms*—COVID-19 detection, chest x-ray, adaptive histogram, CNNs**

I. Introduction

In the course of Artificial Intelligence on Health Sciences, we have been introduced to the ideas of machine learning- and neural network-based problem solutions. These problems mostly includes either a regression or a classification problem. On the project part of the course, we are asked to design a system which gets chest X-ray inputs, and give the prediction of COVID-19 disease status on the patient. I have chosen to solve the problem with convolutional neural networks by knowing their popular use on image-based classifications.

On the other hand, designing a CNN from scratch needs to perform a trail-and-error approach to find the best model. To handle this problem, a lot of state-of-art articles uses the technique called Transfer Learning, and it's fine-tuning. I have performed trails on transfer learning with ResNet-50 model (which is better on CXR classification [3]) and custom models to see which performs the best.

I have used a pre-processing technique which helps to increase contrast of the image to help the AI for finding informative features.

II. Preprocessing of the Dataset

One of the most important features that distinguishes the artificial intelligence model I created is that the data to be used as input has first gone through normalization and then adaptive histogram equalization. This pre-processing of the data helps the model with better contrast on chest x-ray images. The high contrast value in the images causes the shapes in the image to be more prominent, and this can make the model easier to diagnose. The method I used is found on the Reference [5].

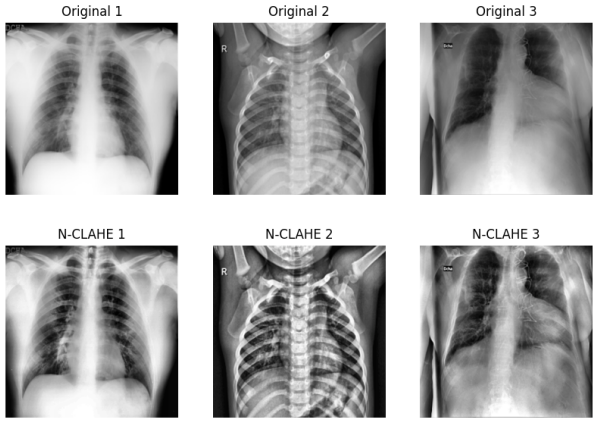

Fig. 1. The N-CLAHE method's output on CXR images.

Firstly, I have applied min-max normalization to the images to have them in a range of value to make the result of CLAHE to more predicable equalization. Afterward, I have applied contrast limited adaptive histogram equalization on the images to have better contrast. You can see the resulting images for 3 image in the Fig. 1

III. Transfer Learning Approach

Transfer learning is a machine learning technique in which a model trained on one task is re-purposed on a second related task. This can be useful in cases where collecting and annotating a large amount of data for the second task is infeasible, or where the second task is much smaller in scale than the first. Transfer learning allows the model to use its knowledge of the first task to improve performance on the second task, even if the second task is quite different from the first. Transfer learning has been successful in a variety of fields including computer vision, natural language processing, and speech recognition.

Fine-tuning is the process of adapting a pre-trained model on a new task through training. It involves updating the weights of the pre-trained model using the data for the new task. Fine-tuning can be used in transfer learning when the new task is similar to the original task for which the pre-trained model was trained. This can be an effective way to leverage the knowledge learned by the pre-trained model and improve performance on the new task.

For example, suppose a model was pre-trained on a large dataset of images of animals and then fine-tuned on a smaller

dataset of images of birds. The model would use its knowledge of animals, which it learned during the pre-training phase, to improve its performance on the bird classification task. Fine-tuning usually involves training only a portion of the layers in the pre-trained model, rather than training the entire model from scratch, since the lower layers of the model contain more general features that are likely to be useful across a wide range of tasks.

ResNet (short for "Residual Network") is a type of neural network architecture that was introduced in 2015 by a research team at Microsoft. It is designed to be able to train very deep neural networks, with tens or even hundreds of layers, while still being able to achieve good performance. One of the key ideas behind ResNet is the use of "skip connections," which allow the network to more easily propagate gradients back through the layers, making it easier to train deep networks. This has made ResNet a popular choice for many image classification tasks, and it has been used to win several image classification competitions.

A. Fine-Tuning on ResNet-50 for COVID-19 Dataset

I have used the CNN architecture part of the ResNet-50, and append a dense neural network afterward. With 64-neuron, followed by 10 neurons and ending with 1 neuron with sigmoid activation function. On my first trail with batch size of 32, epoch of 10, and with a small dataset portion (500 COVID, 463 non-COVID).

The performing test had resulted as 77.03% validation data accuracy.

IV. Custom CNN Model Approach

Designing a convolutional neural network from start needs a methodological approach to have better results on each trail-and-error step to not lose huge times on training useless models. The process can be simplified on this algorithm:

1) Use the modern convolutional block with the same depth and the same hyperparameters, and choose the best validation accuracy.
2) Use different hyperparameters and depth on the best model found in Step 1.
3) Train the network with larger amount of data to get the weights.
4) If it over-fits, use dropout layers, and if it did not work as expected, use regularization.

A. The Standard Blocks for Constructing CNNs

The most used building blocks of CNNs are

- C(x)C(x)M—C(2x)C(2x)M—...
- C(x)M—C(2x)M—..."
- C(x)M—C(x)M—...

which $C(x)$ indicates convolutional layer with x filters, M indicates max pooling. In this part, I have constructed two-blocks of networks using the above most-used blocks. Each network ends with 64-neurons, followed by 10-neurons and followed by 1-neuron dense network which is the same as transfer learning approach. The validation accuracy of each model can be found in the Table I.

TABLE I
THE CNN ARCHITECTURES USED TO PREDICT WHICH STANDARD BLOCKS ARE BETTER FOR COVID-19 DATASET.

Input Size	Network Architecture	Accuracy
(299, 299, 3) RGB	C(32)C(32)M — C(64)C(64)M — D	0.731707
	C(32)M — C(64)M — D	0.882114
	C(32)M — C(32)M — D	0.873984
(299, 299, 1) Grayscale	C(32)C(32)M — C(64)C(64)M — D	0.873984
	C(32)M — C(64)M — D	0.869919
	C(32)M — C(32)M — D	0.878049

As you may see from the table, the best resulted model is C(32)M — C(64)M — D on RGB image input. Therefore, I decided to continue on that model with tuning its hyperparameters. In the next section, the methodology used to predict the best hyperparameters are introduced to the reader.

B. Hyperparameter Tuning

In the context of convolutional neural networks (CNNs), a hyperparameter is a parameter that is set before training the model and whose value cannot be directly learned from the data. Hyperparameters are typically chosen by human designers, and the process of choosing good hyperparameters is known as hyperparameter tuning. Here are the hyperparameters of CNNs described:

- **Filter size**: This hyperparameter determines the size of the filters that are used in the convolutional layers of the network. Larger filters can capture more information about the input, but also increase the number of parameters in the model and may require more computational resources to train.
- **Stride**: This hyperparameter determines the step size that the filter moves across the input when performing the convolution. A larger stride will reduce the spatial dimensions of the output, which can be useful for down sampling the input.
- **Padding**: This hyperparameter determines whether to add padding to the input before applying the convolutional filters. Padding can be used to preserve the spatial dimensions of the input, and may be necessary to ensure that the output has the same dimensions as the input. It is useful to have some padding if the borders are important for the feature extraction.
- **Number of filters**: This hyperparameter determines the number of filters that are used in each convolutional layer of the network. More filters can capture more information about the input, but also increase the number of parameters in the model and may require more computational resources to train.
- **Dropout rate**: This hyperparameter determines the probability that a neuron will be "dropped out" or turned off during training. Dropout is a regularization technique that can help prevent overfitting by forcing the model to learn multiple independent representations of the input data.
- **Learning rate**: This hyperparameter determines the step size at which the optimizer updates the model parameters

during training. A larger learning rate can lead to faster convergence, but may also cause the model to oscillate or diverge. A smaller learning rate may converge more slowly, but can help the model find a better local minimum of the loss function. The most convenient optimizer is Adam's with 0.001 learning rate.

Hyperparameter tuning is an important aspect of building and training CNNs, as the choice of hyperparameters can significantly affect the performance of the model on the task at hand. There are many approaches to hyperparameter tuning, including manual trial and error, grid search, and random search. More recently, techniques such as Bayesian optimization and evolutionary algorithms have also been applied to hyperparameter tuning.

In this paper, I have used the trail and error approach with an automated software tool. This tool called as *Keras Tuner* and available on Python as a pip package.

The validation loss was 0.3945 and the training loss was 0.0128 of the model I have chosen. This huge gap between losses of train and validation set means that, the model is overfitted. I have added a dropout layer after each max pooling layer to overcome this issue.

On this section, I have constructed different network architecture with different hyperparameters to see which one performs better. Note that, it is quite heavy process and since the trainings held using Google Colab' Tesla 4 GPUs and Intel Core i7 CPUs, only limited amount of the epochs could be achieved (less than or equal to 10) with the small dataset explained before.

The hyperparameters tested can be shown in the Table II.

TABLE II
THE TRAILS OF HYPERPARAMETERS FOR TUNING THE ARCHITECTURE

Hyperparameter	Tested Values	The Best
Dropout Rate	[0.0 0.1, 0.2, 0.3, 0.4, 0.5]	0.1
Model Depth	[2, 3]	3
Starting Filter Size	[16, 32, 64]	32
1st Dense Layer Size	[32, 64, 128]	128
Increasing Kernel	[True, False]	False
Starting Kernel Size	[3, 5]	3

Let's investigate each hyperparameter name, and what I mean by changing their values.

Dropout Rate is the fraction of all neurons to be "disabled" during tuning. From the results, we can say that disabling the 10% of neurons is effective to generalize the model. **Model Depth** is configured to have two model blocks (CMCM) or (CMCMCM), which is important to have more features kept in the model. The tuning process shows us that three-blocks depth is more efficient. Note that, there is no dropout layer added after the last block's max pooling layer. **Starting Filter Size** is the filter size on the first convolutional layer. On each convolutional layer, this filter size is double comparing to previous layer. So, it is better to have filters of 32, 64, 128 for three convolution layers.

First Dense Layer Size is the amount of neurons on the first dense layer. In our architecture, I have used three dense layers after convolutional blocks flattening, which last two are already set to 10-ReLU and 1-sigmoid. Within the results, we can understand that dense layer with 128 neurons is better. **Increasing Kernel** is the hyperparameter, which checks that should we double the kernel size on each convolutional layer. For example, if the kernel size used in the first layer used (3x3), the second should be (5x5) if this parameter says True. However, it seems that increasing does not affect our results that much. The last hyperparameter I have tuned is **Starting Kernel Size**. It is the kernel size of the all convolutional layers, since doubling kernel parameters returned False.

The final state of the model can be shown on the Figure 2. The model has 20,165,077 parameters.

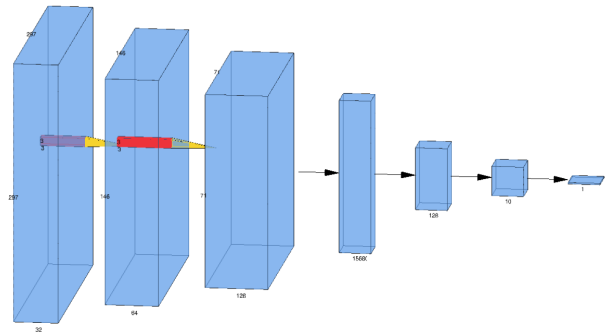

Fig. 2. The 3D Representation of the Model

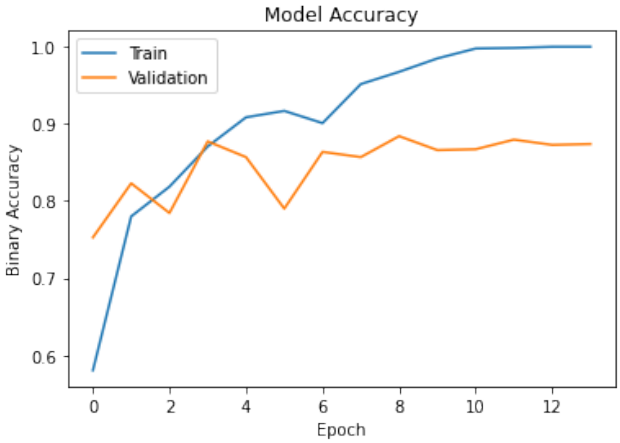

Fig. 3. Accuracy Graph of the Model

C. Training Using All Dataset

After the training done using a bigger dataset (consists of 1300 CXR images), the model accuracy found as 87.68% for the best loss difference (epoch = 4). The plots can be seen on Figures 3 and 4.

V. CONCLUSION

As we discussed on the Sections III and IV, within the same output-dense layer and the same training hyperparameters, the custom CNN network I have designed resulted better on the same small dataset. On the other hand, the proposed model does not successfully perform as the other state-of-art techniques.

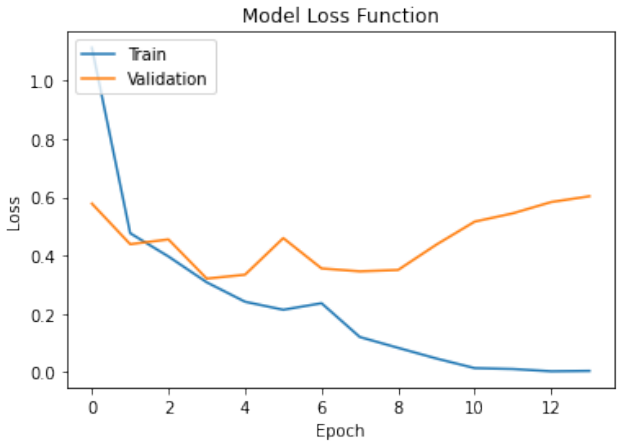

Fig. 4. Loss Graph of the Model

VI. Future Work

- Data augmentation is crucial to have generalized models. In my project, I have not used any artificial data since the dataset already huge enough. However, it may increase the validation accuracy.
- It may be also a better idea to perform hypertunning processes using genetic algorithms instead of trail and error approach.
- It would perform better if we would use all the dataset, which contains 27 thousands labeled CXR images. Due to the problems of low-efficent training hardware modules.
- Hyperparameter tuning on epoch and batch size would be very intelligent.

References

[1] M.E.H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M.A. Kadir, Z.B. Mahbub, K.R. Islam, M.S. Khan, A. Iqbal, N. Al-Emadi, M.B.I. Reaz, M. T. Islam, "Can AI help in screening Viral and COVID-19 pneumonia?" IEEE Access, Vol. 8, 2020, pp. 132665 - 132676.

[2] Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S. and Chowdhury, M.E., 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images. arXiv preprint arXiv:2012.02238.

[3] Ali Narin, Ceren Kaya, & Ziynet Pamuk (2021). Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks. Pattern Analysis and Applications, 24(3), 1207–1220.

[4] Guefrechi, S., Jabra, M. B., Ammar, A., Koubaa, A., & Hamam, H. (2021). Deep learning based detection of COVID-19 from chest X-ray images. Multimedia tools and applications, 80(21-23), 31803–31820. https://doi.org/10.1007/s11042-021-11192-5

[5] Koonsanit, Kitti & Thongvigitmanee, Saowapak & Pongnapang, Napapong & Thajchayapong, Pairash. (2017). Image enhancement on digital x-ray images using N-CLAHE. 1-4. 10.1109/BMEiCON.2017.8229130.