Deep Learning-based Detection for COVID-19

Jinyu Yu **Zhiying Xu** **Lifang Wang** **Sevket Melih Zenciroglu**

Abstract

COVID-19 disease has widely spread all over the world since the beginning of 2020. It is desirable to develop automatic and accurate detection of COVID-19 using chest X-ray images. Therefore, in this project we proposed an improved deep convolutional neural network (CNN) model for the detection of COVID-19 cases from chest X-rays. Four different convolutional neural network based models (simple CNN, VGG19, improved CNN and Transfer Learning) have been implemented with several optimizations, such as cyclic learning rate, cross validation, hyper-parameter tuning an so on. Four models together with optimizations are trained using One big data-sets and one small data-sets. Through 23 experiments, the improved CNN model is proved to have high accuracy and be less sensitive to the size of data-sets. The code of our project is:

`https://drive.google.com/drive/folders/1IfWeCOD8Rxz6V93IfsB94umeECc-0X44`

1 Introduction

With more than 4 billion confirmed cases worldwide and nearly 300 thousand deaths, the COVID-19 pandemic has emerged as an unprecedented healthcare crisis. Since it is expected that the number of daily cases worldwide will increase rapidly, the important factors limiting the diagnosis is the duration and accuracy of viral pathological examinations. Patients cannot be tested effectively, they may spread the virus to more people. Another problem will be the expensive large-scale implementation of current diagnostic procedures. Medical conditions are poor in some areas. For the people there, there is not enough technical and economic ability to detect. This will have serious consequences. The existence of deep learning technology can improve these problems to a certain extent. Since deep learning has powerful feature extraction capabilities, it has achieved great success in the field of medical imaging. For example, deep learning is used to detect and distinguish between bacterial and viral pneumonia based on chest X-ray. Therefore, it is meaningful and necessary to use this technology to implement a better network model for detecting COVID-19.

In this report, we are focused on the following useful methods: keras tuner, cross validation, cycle learning rate and transfer learning. Keras tuner performs automatic hyper-parameter tuning to increase the accuracy on a computer version problem. There are two types of parameters: trainable parameters and hyper-parameters. Hyper-parameters can be numerous even for small models. Tuning them can be a challenge. Keras tuner can help us to search for a good hyper-parameter combination to boost the accuracy. Cross validation is an important method in the data science and data analysis. It is used to minimize or prevent over-fitting. Cross validation has three different types. Here we use K-fold cross-validation. We divide the data into two parts first and keep the second part (15%) to evaluate the success of the model and never use it during the training. K-fold is applied to the first part. From k subsets, we train k-1 subsets and use the remaining subset for the validation. Different values for k was tested, but mainly we used 5-fold cross validation. The last method is cycle learning rate, as it was declared by Leslie N. Smith[1], instead of monotonically decreasing the learning rate, this method lets the learning rate cyclically vary between reasonable boundary values. Training with cyclical learning rates instead of fixed values achieves improved classification accuracy without a

need to tune and often in fewer iterations. In this report, we also discuss the methods to implement the CNN network with transfer learning and some other methods to improve VGG19.

1.1 Contribution

In this project, we proposed an improved CNN model which is less sensitive to the size of data-sets and has high accuracy. Firstly, we implement three models, simple CNN, VGG19, transfer learning and improved CNN model with hyper-parameter tuning, cyclical learning rate and cross validation to train both small and big data-sets. Secondly, we compare the performance of the same model between data-sets as well as among optimizations. We also compare the results among three models. Finally, we proved that our proposed improved CNN model has a better performance.

1.2 Outline

In section 2, this report referred to some related research about model compression. In section 3, there is a simple introduction to the data-set we use. The methods in this report have been explained clearly in Section 4. Moreover, the experiment configuration and results are shown in Section 5. At last, a brief summary is given in Section 6.

2 Related Work

In the literature, a wide range of algorithms has been proposed for COVID-19 detection based Convolutional Neural Network (CNN) which can automatically detect the important features without any human supervision.

Linda Wang[2] proposed a new architecture of CNN, called COVID-net. They use a much larger data-set. The authors report an accuracy of 92.4overall and sensitivity of 80% for COVID-19. Chuansheng Zheng[3] developed a weakly-supervised deep learning-based software system using 3D CT volumes to detect COVID-19. The Proposed DeCoVNet includes stem, two 3D ResBlocks and Classifier. They applied extensive data augmentation on training CT volumes to obtain more training examples. Ali Narin[4] proposed three different convolutional neural network based models (ResNet50, InceptionV3 and InceptionResNetV2) for the detection of COVID-19 using 5-fold cross validation. Halgurd S. Maghdid[5] proposed a simple convolution neural network (CNN) with transfer learning algorithms and modified pretrained AlexNet model and the utilized models can provide accuracy up to 98% via pretrained network and 94.1% accuracy by using the modified CNN. Eduardo José da Silva Luz[6] proposed to explore and extend the EfficientNet family of models using chest X-ray images to perform COVID-19 detection. [7] introduces a new deep learning framework, named COVIDX-Net to assist radiologists to automatically diagnose COVID-19 in X-ray images. A comparison among seven different well-known deep learning neural networks architectures was presented. Experiments and evaluation of the COVIDX-Net have been successfully done based on 80-20% of X-ray images for the model training and testing phases, respectively. The VGG19 and Dense Convolutional Network (DenseNet) models showed a good and similar performance of automated COVID-19 classification with f1-scores of 0.89 and 0.91 for normal and COVID-19, respectively.

In our project, we implemented several CNN models and use some optimizations mentioned above and did several interesting experiments.

3 Data

In this project, two data-set are used. A small data-set of covid-19 from GitHub [8] is utilized in our tests . This data-sets consist of 92 Covid-19 and 99 non-Covid-19 chest X-ray images picked from this link.

A big data-set of pneumonia from Kaggle [9] is also used in our tests.This data-set has 5870 images in total, 1583 normal images and 4287 pneumonia images.

The data-set is split into three parts, 75% of data as train data, 15% data as validation data and 15% data as test data. For 5-fold cross validation, we use 20% data as test data and the remaining 80% of data for 5-fold cross validation. So, practically 16% of data is used for validation and 64% for training. In some cases, we picked the test ratio as 15% during our experiments.

To read the images, we use OpenCV's cv2 interface. After that, with the same interface, we adjust the order of colors from BGR (blue, green, red) to RGB (red, green, blue) because imread() reads the colors in BGR order

$$cv2.cvtColor(image, cv2.COLOR_BGR2RGB)$$

Matplotlib uses RGB format to plot the images, so as many other libraries.

4 Methods

4.1 Convolutional neural network(CNN)

Convolutional neural network (CNN) plays an important role in image classification and detection. The classic CNN architectures have a few layers stacked up on top of each other. The architecture may involve a convolutional layer with activation functions, mostly ReLU, followed by a pooling layer. This process is repeated for a few layers. Then the final (top) layer involves a fully connected Dense layer with a softmax activation function. We choose cross-entropy as the loss function.

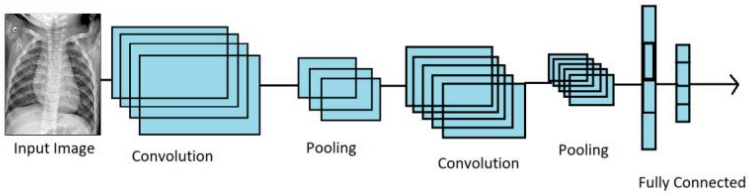

Figure 1: CNN Architecture

4.1.1 Simple CNN model

In this simple CNN model, we used pre-processed images in the form of (224, 224, 3) matrices as our inputs by using the OpenCV library. There are four convolutional layers which use ReLU activation function in the model having max_pooling, dropout, flatten and dense layers in between. Finally, we flatten our features and use softmax activation function for the classification. The model compiles with Adam as optimizer, categorical cross-entropy as loss function. See table 4.

4.1.2 VGG19 model

VGG19 has 19 layers which is characterized by its simplicity, using only 3x3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier[10]

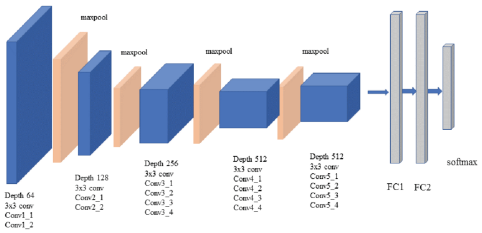

Figure 2: VGG19 Architecture

4.1.3 Improved CNN model

We propose an improved CNN model based on VGG19. In this model, after VGG19, there is a progressive classifier including three 2D convolution layers and a a fully-connected (FC) layer with the softmax activation function.The classifier can progressively abstract the information in the X-ray volumes by 2D max-pooling and finally output the probabilities of detection.

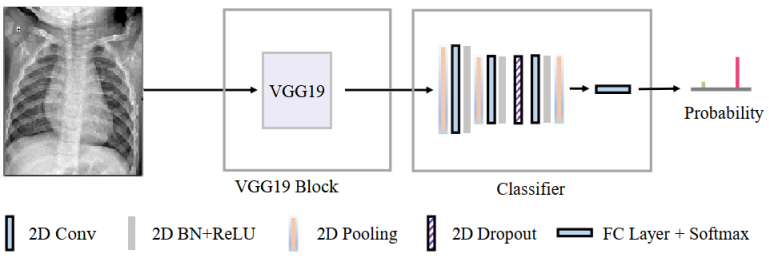

Figure 3: Improved CNN Architecture

4.2 Transfer Learning

While operating with Neural Networks, it is possible to take advantage of the previous learning in a similar data. It is convenient to use one or more layers of a successfully trained and stored model and use it in your newly created model. This technique is called as Transfer Learning (TL). Practically, reusing the prior knowledge potentially increases the chances of the accuracy of the model [11]. In our study, we first generated a model by using a VGG19 layer as the base layer and adding a flattening layer, two dense layers (not in a row) and a dropout layer on top of it. After training and saving the model in h5 format, we created another model. In this new model, we started to build the model with two Conv2D layer with 64 filters, 3x3 kernel_size, and ReLU activation. After that, we loaded our previously trained model. We removed first two layers of VGG19 in this previous model and added all other layers of VGG19 on top of those Conv2D layers we created for the second model. We also added the rest of the layers from the previous model and finished the design of our new model. In both models, we used 5-fold cross validation. We marked all the old layers as non-trainable and trained the new two layers.

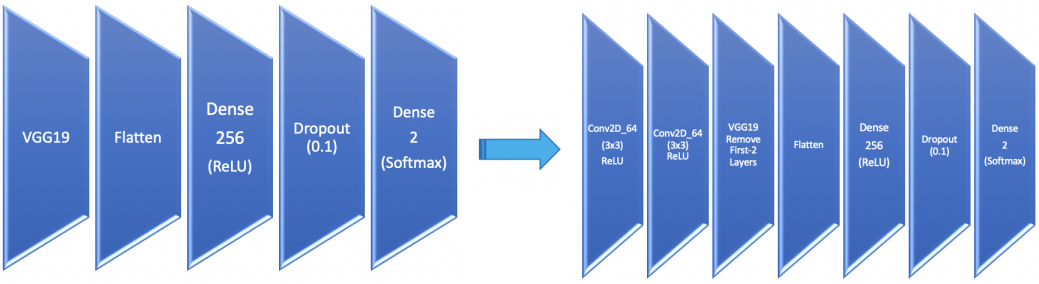

Figure 4: Transfer Learning

4.3 Optimizations

As it is mentioned in the introduction, we mainly use some methods to optimize our network. Here are the details of them.

4.3.1 Data augmentation

For augmentation, we basically use ImageDataGenerator class to generating different versions of the same images. It is very convenient to use that class especially one has a very limited data-set in hand to use. We only apply this to the training data with the properties of rotation_range 20, width_shift_range and height_shift_range 0.2, an horizontal_flip True. Then, we feed our model with this generator to train.

4.3.2 K-fold cross validation

Over-fitting occurs when the function is too suitable for a limited set of data. This is what we do not want to see. The solution to this problem is a process called cross-validation (CV). Fig 4.is the visual

representation of K-folds. The test set should still be retained untouched for the final evaluation. 5-fold CV is used in our project. We divide the training set into 5 smaller sets and use 4 (k-1) subsets to train our data and the last subset is used as for validation. Then the performance (final accuracy) reported by 5-fold CV is the average of the values calculated in the loop.

4.3.3 Cycle learning rate

We use Cycle Learning Rate (CLR) instead of monotonically decreasing our learning rate for two reasons: on the one hand, the network may fall into a local minimum, and a lower learning rate may not be enough to leave this area and descend to a lower loss area; on the other hand, the model and optimizer may be very sensitive to the initial learning rate we select. If we choose a bad initial value of the learning rate, then our model may be in trouble from the beginning.

First of all, we need to define some values. For example, batch size, Cycle, step size and so on. Leslie Smith[7] recommends that the step_size should be

$$(2 - 8) * training\_iterations\_in\_epoch \tag{1}$$

In our project, the value of step_size is 8. CLR also has some different types. the "triangular" policy is a simple triangular cycle. The learning rate starts at the base value and then starts to increase. It reaches to the maximum in the half of the cycle (the step size). Once the maximum learning rate is reached, it will be reduced back to the basic value. Similarly, it takes half a cycle to return to the basic learning rate. Last, repeat the entire process until the training is finished. We use the "triangular2" policy in our project. It is similar to the "triangular" policy, but it reduces the maximum learning rate by half after each cycle. Because decreasing the maximum learning rate over time helps stabilize training.

4.3.4 Hyper-parameter tuning

A good hyper-parameter combination can highly improve model's performance.We can do hyper-parameter tuning with Keras Tuner which searches function performs the iteration loop, which evaluates a certain number of hyper-parameter combinations. Evaluation is performed by computing the trained model's accuracy on a held-out validation set[12].

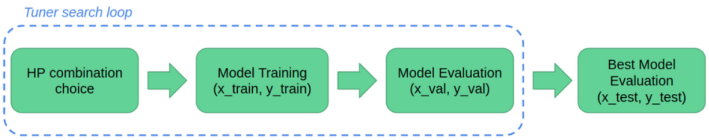

Figure 5: Hyperparameter tuning process with Keras Tuner

5 Experiments

Based on the above models and optimizations,we did 21 kinds of different experiments using two data-sets. For covid-19 data-sets, we did 12 experiments using three models combined with three optimizations,shown as table 5. For X-ray data-sets, we did 9 experiments using three models combined with two optimizations, see table 6. In the following section, we will describe the details of these experiments.

5.1 Parameter Settings

Firstly, we train the data-sets using three models without any optimization with Adam default learning rate 0.001, then we train the data-sets using keras tuner to choose a better learning rate.The parameters are min_value=1e-4, max_value=1e-2, sampling='LOG'. Next we use cyclic learning rate. We also train three models using 5-fold cross validation and keras tuner. The batch size for all the experiments is 32.

And for CLR, as we mention in section 4.2.3, there are different types of CLR. Here we only compare ”triangular“ and "triangular2". The accuracy of the model with triangular2 is 1.36% higher than the model with triangular. So we used the “triangular2” policy instead of the “triangular” policy in CLR.

5.2 Experiments Result

Using the above parameters, we did experiments for different models and optimizations. In the following sections we will elaborate on the experimental results.

5.2.1 Results on simple CNN

In this section, we focus on learning rate and size of data-sets which can have a big influence on the accuracy. The model can get the best performance that the test accuracy is 93.10% with learning rate 0.000422744 when we apply keras tuner to search a good learning rate. Also, when we apply cyclic learning rate (CLR), the test accuracy is a little higher, but not too much.The results are shown in table 1.

Table 1: Test accuracy of different learning rate

Learning rate	0.001	0.00061051	CLR
Test acc (%)	75.86	82.76	79.31

5-fold cross validation (CV) is used and the test accuracy is 81.53% which is much higher compared to 75.86%.

Big data-sets make sense to improve the performance, because when using X-ray data-sets,the test accuracy is 94.54% which is much higher than small data-sets of covid-19.

Table 2: Test accuracy of different optimizations

Optimizations	none	CLR	CV	Big datasets
Test acc (%)	75.86	79.31	81.53	94.54

5.2.2 Results on VGG19

As the same with the last section, we used some methods to optimize VGG19. At the same time, two different size data-sets are also used for comparison to get a better model and higher accuracy. And the parameters we used in this part are the same like before.

For the small size of data-set, only using the VGG19 model without any optimization method, the accuracy we got is 86.20%. Later, by comparing the test accuracy after using the two optimization methods (CLR and CV). It is not difficult to notice that these two methods have a certain effect on optimizing VGG19. They are able to increase the accuracy of the model by about 4%.

It is worth mentioning that using a large-size data-set can significantly increase the accuracy of the model. The accuracy for simple VGG19 with the large-size of data-set is 98.36% while it is 86.20% for the model use small-size of data-set. It is more than 12% higher than the accuracy of the model uses the small-size of data-set. At the same time, the model using large-size of data-set also has a more prominent performance in loss. Fig 6 is the loss and accuracy plots for VGG19+CLR. The left two plots are the model uses small-size of data-set and the right two are the model uses small-size of data-set. Compared with the model using the small-size of data-set, the optimization method does not particularly increase the accuracy for the model using the large-size of data-set.

5.2.3 Results on Improved CNN

Compared with other models, improved CNN can significantly improved the performance and it is less sensitive to the size of data-sets.The test accuracy is 89.66% for small data-sets and 98.36% for big data data-sets.When applying cyclic learning rate, the gap of test accuracy between small

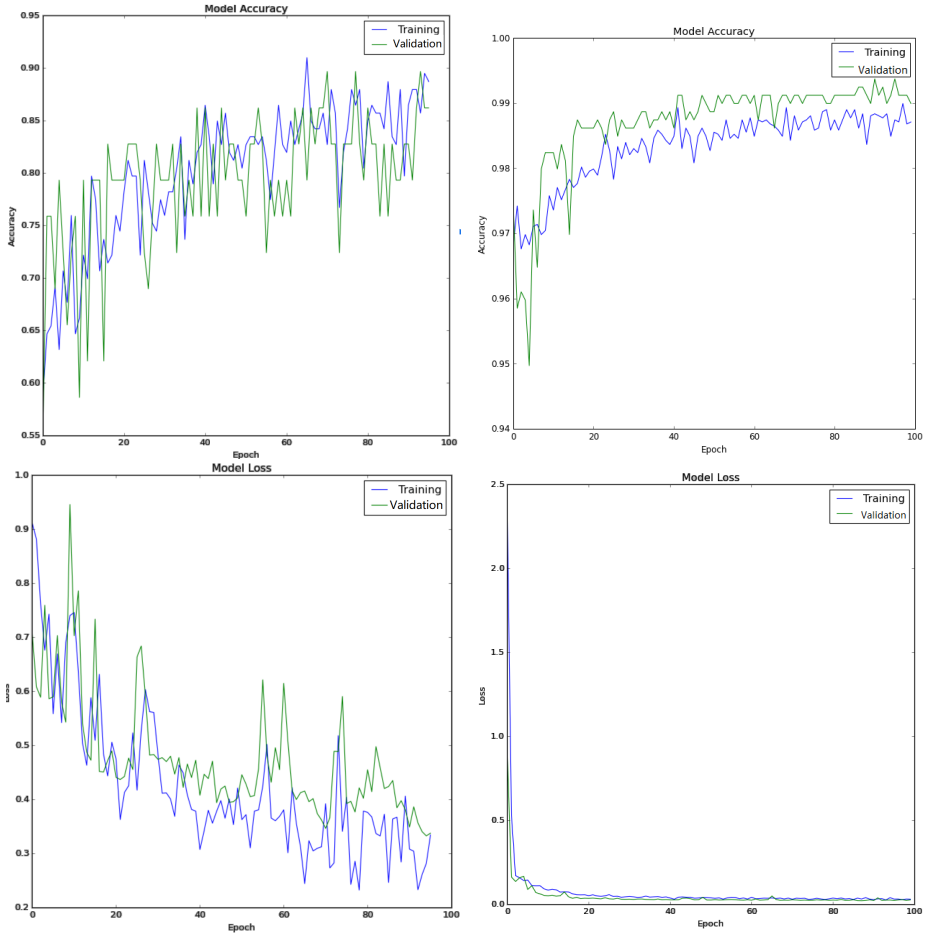

Figure 6: loss & acc for VGG19+CLR

data-sets and big data-sets is smaller, which is 93.1% for small data-sets and 98.74% for big data-sets. What surprises us is that when using keras tuner to search a better learning rate, the test accuracy is 96.55% with learning rate 0.003385118, which is almost the same as the test accuracy 97.38% using big data-sets.

Table 3: Test accuracy of different data-sets & optimizations

Datasets	none	CLR	CV	tuner
small datasets (%)	89.66	93.10	94.32	96.55
big datasets (%)	97.38	98.74	98.89	-

5.3 Transfer learning

The first model we trained as the base model concluded a test accuracy of 93.1%, having 97.58% of validation and 97.04% of training accuracy. The second model transferring some layers, indeed most of the layers, from the first one achieved a 95.17% of test accuracy, 93.33% of validation and 94.32% of training accuracy. For both of the models, 5-fold cross validation is used during the training and 15% of the data was reserved for evaluating the test results.

6 Conclusion

Based on our 21 experiments, we can conclude that the improved CNN model we proposed can predict the X-ray image a covid-19 or normal with the accuracy 96.55%.

We also proved that some optimizations can improve the performance.

- Proper learning rate can significantly improve the performance. What is worth to mention is that keras tuner is a good choice to tune the learning rate as well as other parameters. Cyclic learning rate is also a good way to improve the performance.
- Simple CNN model and VGG19 model rely on size of data-sets a lot, large-size data-sets can significantly improve the performance
- Cross Validation: By increasing the number of data trained to achieve the purpose of improving network performance. For those networks that use small-size of data-sets, CV has a more significant effect.
- Transfer Learning is a very powerful way of taking advantage of the previous experiences and saving time while reaching to higher performance values.

Our improved CNN model is much less sensitive to the size of data-sets. It can get almost the same test accuracy between big data-sets and small data-sets.

References

[1] Leslie N. Smith, (2015). Cyclical Learning Rates for Training Neural Networks. arXiv:1506.01186[cs.CV].

[2] Wang, L., & Wong, A. (2020). COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest Radiography Images.

[3] Chuansheng Zheng.Deep Learning-based Detection for COVID-19 from Chest CT using Weak Label

[4] Ali Narin. Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks

[5] Halgurd S. Maghdid. Diagnosing COVID-19 Pneumonia from X-Ray and CT Images using Deep Learning and Transfer Learning Algorithms

[6] Eduardo José da Silva Luz.Towards an Effective and Efficient Deep Learning Model for COVID-19 Patterns Detection in X-ray Images

[7] E. E.-D. Hemdan, M. A. Shouman, and M. E. Karar, "Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images," arXiv preprint arXiv:2003.11055, 2020.

[8] https://github.com/casperbh96/COVID-19-Detection

[9] https://academictorrents.com/details/95588a735c9ae4d123f3ca408e56570409bcf2a9

[10] Building VGG19 with Keras.https://medium.com/@saicharanars/building-vgg19-with-keras-f516101c24cf

[11] George Karimpanal, Thommen; Bouffanais, Roland (2019). "Self-organizing maps for storage and transfer of knowledge in reinforcement learning". Adaptive Behavior. 27 (2): 111–126. arXiv:1811.08318. doi:10.1177/1059712318818568. ISSN 1059-7123.

[12] Hyperparameter tuning with Keras Tuner.https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner

Appendix

Table 4: Simple CNN Model Summary

Layer (type)	Output Shape	Param
conv2d_1 (Conv2D)	(None, 222, 222, 32)	896
conv2d_2 (Conv2D)	(None, 220, 220, 128)	36992
max_pooling2d_1(MaxPooling2D)	(None, 110, 110, 128)	0
dropout_1 (Dropout)	(None, 110, 110, 128)	0
conv2d_3 (Conv2D)	(None, 108, 108, 64)	73792
max_pooling2d_2(MaxPooling2D)	(None, 54, 54, 64)	0
dropout_2 (Dropout)	(None, 54, 54, 64)	0
conv2d_4 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_3(MaxPooling2D)	(None, 26, 26, 128)	0
dropout_3 (Dropout)	(None, 26, 26, 128)	0
flatten_1(Flatten)	(None, 86528)	0
dense_1 (Dense)	(None, 64)	5537856
dropout_4 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 2)	130
Total params: 5,723,522	Trainable params: 5,723,522	Non-trainable params: 0

Table 5: Experiments for covid-19 datasets

No.	Experiments	Train Acc(%)	Validation Acc(%)	Test Acc(%)
1	Simple CNN	83.46	79.31	75.86
2	Simple CNN+tuner	86.37	84.23	82.76
3	Simple CNN + CLR	82.18	86.21	79.31
4	Simple CNN +CV	82.08	81.99	81.53
5	VGG19	88.72	86.21	86.20
6	VGG19+tuner	90.36	88.42	89.65
7	VGG19+CLR	84.96	93.10	90
8	VGG19+CV	93.44	96.67	93.85
9	Improved CNN	98.50	93.10	89.66
10	Improved CNN+tuner	97.74	93.10	96.55
11	Improved CNN+CLR	96.24	92.21	93.10
12	Improved CNN+CV	97.01	93.28	94.32

Table 6: Experiments for X-ray datasets

No.	Experiments	Train Acc(%)	Validation Acc(%)	Test Acc(%)
1	Simple CNN	92.45	91.67	94.54
2	Simple CNN + CLR	97.16	95.48	96.78
3	Simple CNN +CV	97.85	97.15	96.95
4	VGG19	98.84	99.12	98.36
5	VGG19+CLR	98.58	98.74	98.99
6	VGG19+CV	98.49	98.16	98.76
7	Improved CNN	99.62	98.79	97.38
8	Improved CNN+CLR	99.50	98.99	98.74
9	Improved CNN + CV	99.54	99.03	98.89

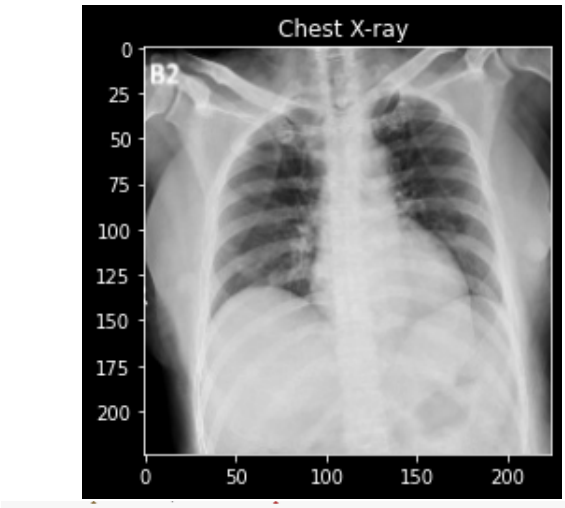

Figure 7: Input X-ray(covid-19) and the prediction from our network