COVID-19 DETECTION IN X-RAY IMAGE OF LUNG PARENCHYMA USING DEEP NEURAL NETWORK

Sachin Verma, Saurabh Pandey, Avni Kothari,Pradip Gupta,Prashant Kumar Giri

School of Computer Science and Engineering, Lovely Professional University

Abstract: *Covid-19**[1]** a type of virus generally found in animals mainly in bats. Scientist said that this virus get transferred from Bats to Humans. Normally this process of mutation is not possible and even if possible then it occurs in thousands of year. This virus is originated from Wuhan city of China and spread across the whole world in 2019. Due to its origin in 2019, it is named as covid19. A new kind of virus whose Anti-Virus is not present in human body. Researchers are working to make its vaccine. Russia is the first country to make vaccine earliest, launched in August 2020. With the use of Machine Learning, we can make Models that can be used to Predict/Classify the diseases in Humans. For example Model that can take Physiological parameter and then predict the level of fitness in Humans. There are numerous Model that were already built and Peoples across the world are working over new Model by using the Platform like ImageNet, Kaggle, Pytorch and Tensorflow. Machine Learning has also huge application in other fields like Manufacturing industries, Robotics Automation Process, CNC technology, UAVs and many more. In Medical fields, Machine Learning has wide application for example Detect the presence of tumor**[2]**, classify between diseased or Healthy person using X-ray images, classification of multi class diseases in human being, Predict RBC/WBC count in Humans Blood even a lot more things can be done.*

Keywords: Lung Parenchyma, Image Convolution, Neural Network, Hyper-parameter Tuning, Optimizers, Global Average Pooling, Deep Learning.

1. INTRODUCTION:

We designed a Neural Network Model using CNN, MaxPooling, Dense, Flatten and other layers. It classifies the X-Ray images of lung parenchyma into three classes namely NiCT, pCT and nCT. pCT means Positive CT Scan images and it refers to image of Covid19 patient, nCT means Negative CT Scan images i.e Normal Person and NiCT means Non-informative CT Scan images i.e no decision can be made for this kind of image because of very less informative features present in image.

In Simple term NiCT, pCT and nCT represents three classes as non-informative sample, Positive case and negative case for covid19. The Model we build extract the features from X-Ray image and uses final layer as Dense with three Neuron to predict the target class out of three (NiCT, nCT, pCT) discussed above.

Over normal training with epoch value equals to 20 model reached at accuracy of 96.83% and when tested on Testing Data accuracy is 98.91% and loss of model is reduced to nearly 4% i.e from 1 to 4.28. Good choice of Activation Function, Loss Function, Optimizers affect the performance of the Neural Network to a great extent. Normally for multiclass classification sigmoid works best but more ever, it also depend on dataset. Using appropriate Loss Function helps to reduce the loss computed by Neural Network at faster rate and hence increases the accuracy in less Epoch value. Since the Accuracy of our Model is very Good i.e 96% and even on random training for just 20 epochs, the accuracy goes up between 97% to 99%, it means the parameter that we have selected are perfect for our Model. Anyone who is unable to decide the number of neuron to use in Model, they can take help from Hyper-Parameter Tuning technique.

All the results were obtained on 64 bit System with 8GB RAM, Core i5 7th Generation Intel Processor, Clock Speed of 2.5 - 2.71 GHz having Nvidia GeForce 920MX GPU.

2. ABOUT THE DATASET:

Dataset contains X-Ray image of Lung Parenchyma divided into three class NiCT, nCT, pCT. Figure 1.1 shows the number of samples, classes available in Dataset. It has total size of 4.0 GB.

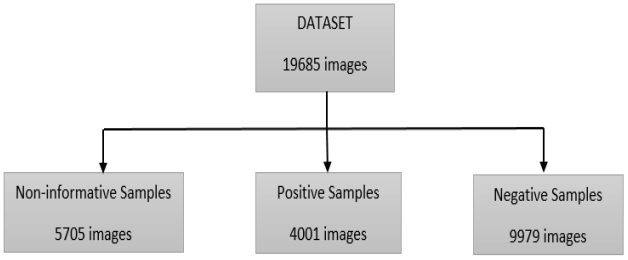


Figure 1.1 Dataset Classifications

Data were collected from 2 hospitals, Union Hospital (HUST-UH) and Liyuan Hospital (HUST-LH) which is describes in detail in the paper *iCTCF: An integrative resource of chest computed tomography images and clinical features of patients with covid-19 pneumonia***[3].**

The authors of the above paper classified individual CT images into three types-
(I) 5705 non-informative CT (NiCT) images where lung parenchyma was not captured for any judgment.
(ii) 4001 positive CT (pCT) images where imaging features associated with COVID-19 pneumonia could be unambiguously discerned, and
(iii) 9979 negative CT (nCT) images where imaging features in both lungs were irrelevant to COVID-19 pneumonia.

Some sample images are shown in below figure 2.1 with their labels.

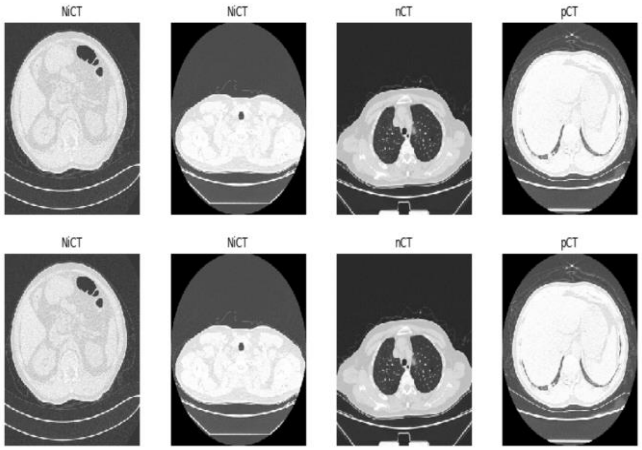
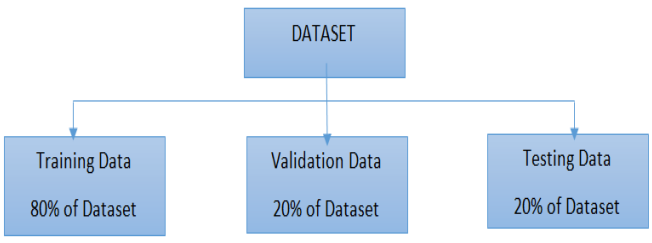


Figure 2.1 Sample images from Dataset

3. METHODS:

3.1 Load the Dataset:

All the images from the Dataset were read from the given path where dataset is located. There were total 19685 X-Ray images present in the dataset and further we divided out dataset into three part as shown below and then created batched dataset by using 30 images in single batch.



Training Data and Validation Data was used during Training of Model and Testing Data was used for Testing the Accuracy and Loss of Model. If split is not applied then whole Dataset will be loaded for Training Data. That will not be good scenario therefore we had split our whole dataset in three part as discussed above. Normally, split that is most widely used is 80% training data, 10% testing and 10% validation data. Validation Dataset and Testing Dataset together knows as Hold Out Sets**[4]** and Validation Dataset helps for parameter optimization during training.

```
Found 19685 files belonging to 3 classes.
Using 13780 files for training.
```

3.2 Data Preprocessing:

The minimum value of Pixel for the image is 0 and maximum value is 255 so rescaling of pixel values are required because pixel value in [0, 1] will reduce our model complexity and easier to work and model would tend to have less variation. This technique is also known as Normalization**[5]**.

For output range in [0, 1] multiply input by 1.0/255.

For output range in [-1, 1] multiply input by 1.0/127.5 and set offset equals to -1 in Rescaling layer.

In our case, we converted the pixel value in range of 0 to 1.

After that reshaped all the images of dataset to value (300, 300, 3) as (HEIGHT, WIDTH, CHANNEL) as shown in figure 3.2.1.

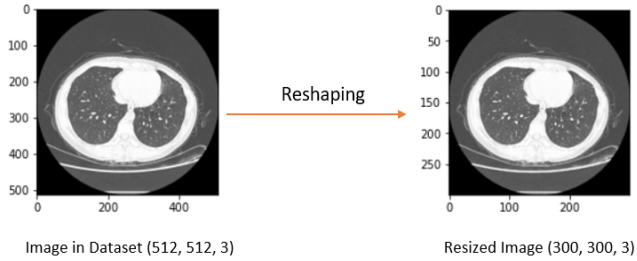


Figure 3.2.1 Reshaping of Image

3.3 Designing Custom Network of Layers (Classification Model):

First layer is INPUT_LAYER which defines the shape for all the incoming images to our Model i.e (300, 300, 3). This means that only image of shape (300, 300, 3) are allowed to pass in Model. Here batch is not defined in INPUT_LAYER means it can take any number of input images. By default Model will get shape as (None, 300, 300, 3). The output shape of INPUT_LAYER is (None, 300, 300, 3). Second layer is Rescaling Layer which rescales

the pixel value in range [0, 1] and its output is (None, 300, 300, 3). After that Convolution i.e Conv2D is applied to each image of shape (300, 300, 3). Let take a look how convolution work on input image. Convolution**[6]** is Dot Product of Image Matrix and Kernel (a.k.a Filter) using Window**[7]** method as shown in figure 3.3.1.

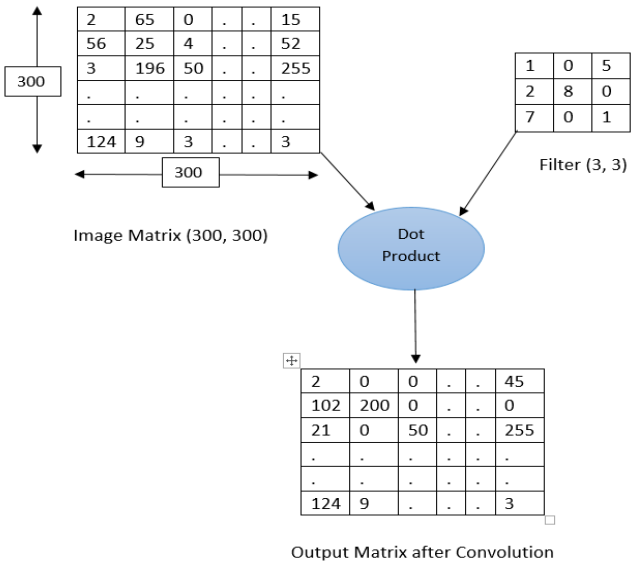


Figure 3.3.1 Working of Convolution layer on Image

At one time (3, 3) filter occupies (3, 3) area of image matrix and dot product is applied between $[A]_{ir}$ element of Image and $[A]_{ir}$ element of filter. Here (i, r) are index of row and column of matrix. For the first window, filter occupies shape of (3, 3) part of image in leftmost side and dot product will take place between each element of filter and each element of covered image matrix. The values obtained in output matrix are calculated as follows-

At (1, 1) equals to 1*2 = 2, At (1, 2) equals to 65*0 = 0 and so on for remaining elements.

Note: Dimension of output Matrix in convolution depend on value of padding and stride, given as-

$$x = \left\lfloor \frac{n - m + 2p}{s} \right\rfloor + 1$$

x: Output dimension,

n: Input image shape,

m: kernel or filter shape,

p: padding value,

s: stride value

A Convolution over a single image is shown in figure 3.3.2. Conv2D is used with number of kernels equals to 1 with filter shape of (3, 3) and activation function equals to “relu”.

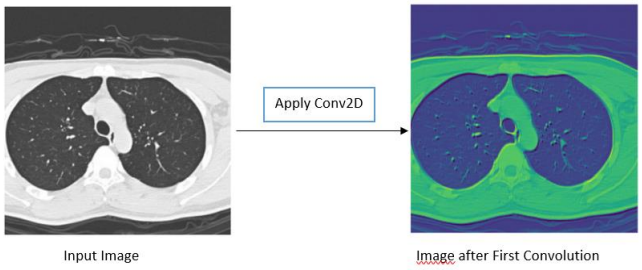


Figure 3.3.2 Applying Convolution on Input image

After Convolution Layer (Conv2D) Sub Sampling Layer[8] type is applied i.e MaxPooling2D**.** The application of MaxPooling2D Layer is for Dimensionality Reduction**[9]** as well as to reduce the number of parameters which results in less complex Model. More the number of parameters, more will be computation time and hence more will be training time of model. Model that are deep like containing 200-300 layer with 200-500 neuron units may have Millions of parameters in such cases it required some techniques to reduce the number of parameter like Global Average Pooling, MaxPooling2D etc. Output shape of Conv2D is (None, 300, 300, 1). Since 1 Neurons used in Conv2D as Conv2D(1, 3,activation=’relu’).

After that Flatten Layer**[10]** is used to merger all the features as (1, n) where n is total number of features obtained from multiplying image height, width and channel value. At the end Dense Layer with three neurons is used for classification since we have three classes [NiCT, pCT, nCT]. Therefore three neurons are required. The summary of Model is shown below in figure 3.3.3

```
Model: "COVID_19_Classifier"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (InputLayer)     [(None, 300, 300, 3)]     0
_________________________________________________________________
rescaling_61 (Rescaling)     (None, 300, 300, 3)       0
_________________________________________________________________
conv2d_108 (Conv2D)          (None, 300, 300, 1)       28
_________________________________________________________________
max_pooling2d_108 (MaxPoolin (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_109 (Conv2D)          (None, 150, 150, 2)       20
_________________________________________________________________
max_pooling2d_109 (MaxPoolin (None, 75, 75, 2)         0
_________________________________________________________________
conv2d_110 (Conv2D)          (None, 75, 75, 2)         38
_________________________________________________________________
max_pooling2d_110 (MaxPoolin (None, 37, 37, 2)         0
_________________________________________________________________
flatten_36 (Flatten)         (None, 2738)              0
_________________________________________________________________
dense_15 (Dense)             (None, 8)                 21912
_________________________________________________________________
dense_16 (Dense)             (None, 16)                144
_________________________________________________________________
Output_Layer (Dense)         (None, 3)                 51
=================================================================
Total params: 22,193
Trainable params: 22,193
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 3.3.3 Summary of Model

Analyzing the Input and Output Dimension to each layer:

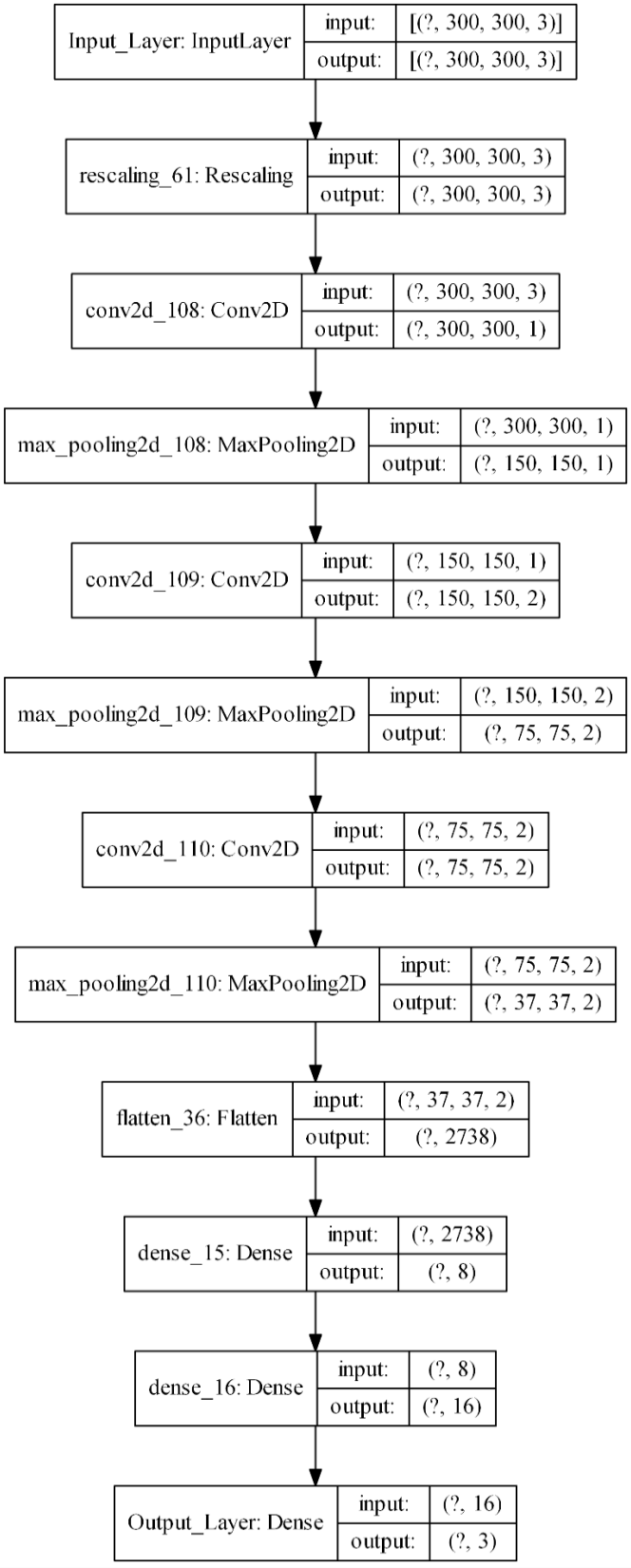


Figure 3.3.4 Graph showing Input and Output Dimension at each layer

3.4 Compile the Model:

Loss function and optimizers**[11]** affect the performance of model, so it is necessary to select the right ones which gives best accuracy. We used Adam as optimizer, although there are many optimizers available in Keraslike SGD, RMSprop, and Adagrad. We can use any of these which fits best with our Dataset. Learning rate supplied in Adam Optimizer is 0.001.

Loss function is Sparse Categorical Corssentropy**[12]** because target having three classes i.e [NiCT, pCT, nCT] even we can go for CategoricalCrossentropy function but that will require our label to be preprocessed with OneHotEncoding**[13]** of shape (n, 3) where n is our number of label for n images. Metric value is Accuracy, multiple option are available like MSE, MAE etc.

3.5 Training the Model:

Model is trained for first 20 epochs only using model fit method. Validation Data is used for Validation of model. 30 batches are trained in one epoch means 32*30 images passes in single epoch. Figure 3.5.1 shows the training of Model, maximum accuracy achieved by model for 20 epoch only is 95.83%. And corresponding validation accuracy achieved is 94.9%. Only starting epochs (from 1 to 2) and last epochs (from 19 to 20) are show in figure because training for 20 epoch will be large and will not be fitted in this paper.

```
Epoch 19/20
30/30 [==============================] - 39s 1s/step - loss: 0.5859 - accuracy: 0.9667 - val_loss: 0.5822 - val_accuracy: 0.972
0
Epoch 20/20
30/30 [==============================] - 37s 1s/step - loss: 0.5917 - accuracy: 0.9583 - val_loss: 0.6004 - val_accuracy: 0.949
1
```

Figure 3.5.1 Training of Model for 20 Epoch value.

4. OUTPUTS AND RESULT:

Final Dense Layer with three Neuron gives the class to which the particular X-Ray image belongs, and using activation function as Softmax**[14]** which give the probability distribution for all target class and the class with maximum probability will be considered as output.

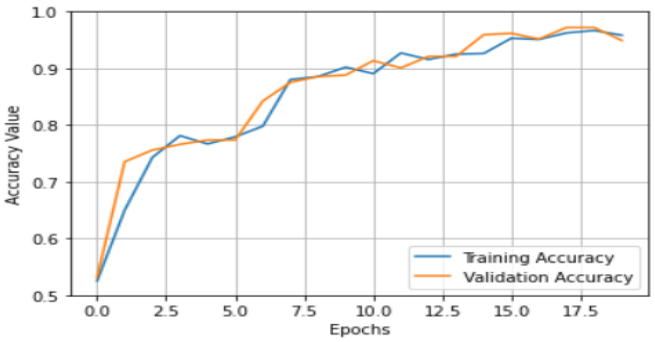


Figure 4.1 Training and Validation Accuracy

Figure 4.1 Show the graph of epoch versus accuracy during training, the Training accuracy value start from approx. 0.4 and slightly converging toward value 1.0 i.e 100% accuracy.

Below figure 4.2 show the decrease in Training and Validation loss value with increase of number of epochs. Loss value is approaching downward. With increased in epoch, both Training and Validation loss can be minimized to very lower extent.

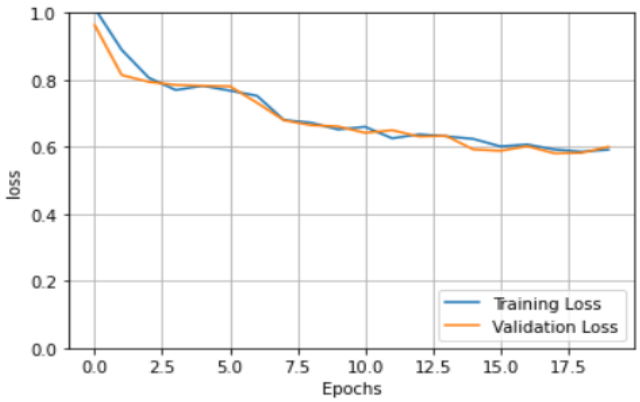


Figure 4.2 Training and validation loss.

Finally Training Accuracy, Validation Accuracy, Training loss and Validation loss are plotted together.

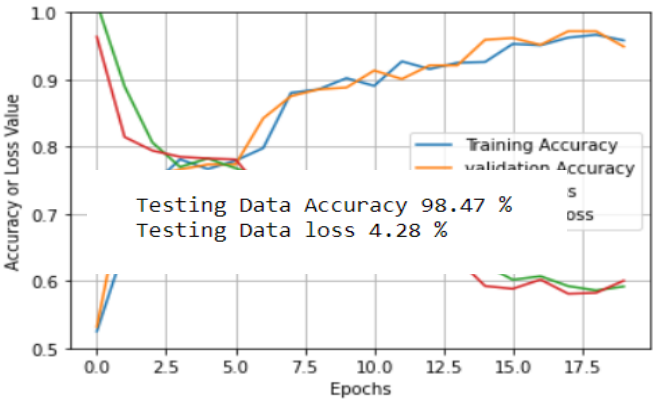


Figure 4.3 Training Accuracy, Validation Accuracy, Training loss and Validation loss

We have results of Training Data and Validation Data, now let look over the results of the model over Testing Data. Below figure 4.4 show the model loss and accuracy on testing data.

Figure 4.3 Accuracy and Loss over testing data.

Confusion Matrix**[15]** of Testing Data is shown in figure 4.5. The value of Recall and Precision are calculated for each class present in our Dataset i.e for NiCT, pCT, nCT.

```
------------Confusion Matrix--------
      NiCT  nCT  pCT
NiCT [[ 99    2    1]
nCT   [  1  206    0]
pCT   [  1    1   82]]

------------MODEL ACCURACY--------

Class_Names      Precision       Recall

  NiCT            0.980198       0.970588
  nCT             0.985646       0.995169
  pCT             0.987952       0.976190
```

Figure 4.5 Confusion Matrix, Recall and Precision

Calculation of Precision and Recall is done as follows

For class1 (i.e NiCT) Precision = 99 / (99+1+1) = 0.980

Recall = 99 / (99+2+1) = 0.970

And similarly calculation is done for all other two classes.

5. CONCLUSION:

Model's maximum accuracy is about 96% during training, this accuracy can further be increased by increasing epoch value, Hyper-tuning of model, selecting the right learning rate and many more factor are there which decides the model performance. Loss is reduced to 0.6 for just 20 epochs and decreasing slowly, using the above techniques, loss value can be decreased to greater extent i.e equals to zero.

Model performance over the testing dataset is also good, about 98% testing accuracy with 40% of loss. Confusion Matrix is also being shown above in figure 4.5 with corresponding Precision and Recall values for all the three classes, using this also we can calculate the accuracy for correct classification in each classes.

Various plots in Section 4 "Outputs and Results" show the behavior of Model over Training Data, Validation Data and Testing Data.

This model can find application in medical for detecting Covid-19 infected or non-infected lungs and after that we can judge whether the person have Covid-19 or not. Only the X-Ray image of lung parenchyma is required as input and model can give the output. For every training, accuracy changes and the maximum accuracy encountered during random training was about 99% so by changing the some layer architecture, model accuracy

can easily cross the Accuracy value of 99%. Above results shows that model does not require any hyper parameter tuning but still we can go for if model accuracy decreases for higher epoch values and more dataset are added as input. Code is available on below Github link-

https://github.com/SACHIN446/Covid-19-Detection-using-Custom-CNN-in-Chest-X-ray-Images.git

6. REFERENCES:

[1] CDC. 2019 Novel Coronavirus, Wuhan, China. CDC.
[2] Artificial Intelligence in Cancer imaging: Clinical challenges and applications, Wenya Libda Bi,MD.
[3] iCTCF: An integrative resource of chest computed tomography images and clinical feature of patient with covid-19 pneumonia, Ning W, Lei S, Yang J, Cao Y, Jiang P, Yang Q, Zhang J, Wang X, Chen F, Geng Z, Xiong L, Zhou H, Guo Y, Zeng Y, Shi H, Wang L, Xue Y, Wang Z.
[4] On Holdout and Cross Validation: A Comparison between Neural Network and Support Vector Machine. (By Jamilu Awwalu and 2Ogwueleka Francisca Nonyelum 1,2 Department of Computer Science, Nigerian Defence Academy, Kaduna, Nigeria).
[5] Pixel Normalization from Numeric Data as Input to Neural Networks, Parth Sane and Ravindra Agrawal, Department of Computer Engineering SIES GST, University of Mumbai, India.
[6] Rader, C.M. (December 1972). "Discrete Convolutions via Mersenne Transforms". IEEE Transactions on Computers. **21** (12):1269–1273. doi:10.1109/T-C.1972.223497.
[7] Spectral Audio Signal Processing, by Julius O. Smith III, W3K Publishing,2011, ISBN 978-0-9745607-3-1.
[8] Deep Generalized Max Pooling, Vincent Christlein, Lukas Spranger, Mathias, Seuret, Anguelos, Nicolaou, Pavel Král, Andreas Maier.
[9] Boehmke, Brad; Greenwell, Brandon M. (2019). "Dimension Reduction". Hands-On Machine Learning with R. Chapman & Hall. pp. 343–396. ISBN 978-1-138-49568-5.
[10] Flattened Convolutional Neural Networks for Feedforward Acceleration Jonghoon Jin, Aysegul Dundar, Eugenio Culurciello.
[11] On Empirical Comparisons of Optimizers for Deep Learning Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, George E. Dahl.
[12] Rethinking Softmax with Cross-Entropy: Neural Network Classifier as Mutual Information Estimator Zhenyue Qin, Dongwoo Kim, Tom Gedeon.
[13] A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers, Kedar Potdar Taher S. Pardawala Chinmay D. Pai.
[14] Activation Functions: Comparison of Trends in Practice and Research for Deep Learning Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall.
[15] Generalized Confusion Matrix for Multiple Classes, Cinmayii Manliguez, University of the Philippines.