

# COVID-19 CT SCAN DATA CLASSIFICATION BY DEEP LEARNING



---

**Name :** AYUSHI JAIN  
**Reg. Number :** 202639  
**Roll Number :** 202CS003  
**Mobile :** 9829337176  
**E-mail :** [ayushijain.202cs003@nitk.edu.in](mailto:ayushijain.202cs003@nitk.edu.in)

**Submitted To.:** Dr. Jenny Rajan (CS737-Deep Learning)

National Institute of Technology Karnataka, Surathkal, Mangalore

## Data Preparation:

The Data is provided by the instructor which is as given below:

Train		Test	
Covid-19	Non-Covid-19	Covid-19	Non-Covid-19
700	700	302	284
Total=1400		Total=586	

Data Set preparation is a very important part in order to train the deep learning model:

1. In this model the CT-SCAN images of both COVID and non-COVID resized into (100\*100), as all the images must not be of same dimensions and fixed scaling of each image is required for which I have used OpenCV library,
2. Normalization of the data set is done by dividing it by 255 so that range becomes (0-1)
3. Data is augmented to increase the size training samples
4. I have classified the COVID images as 0 and non-COVID images as 1.
5. After this I have converted the images into feature lists

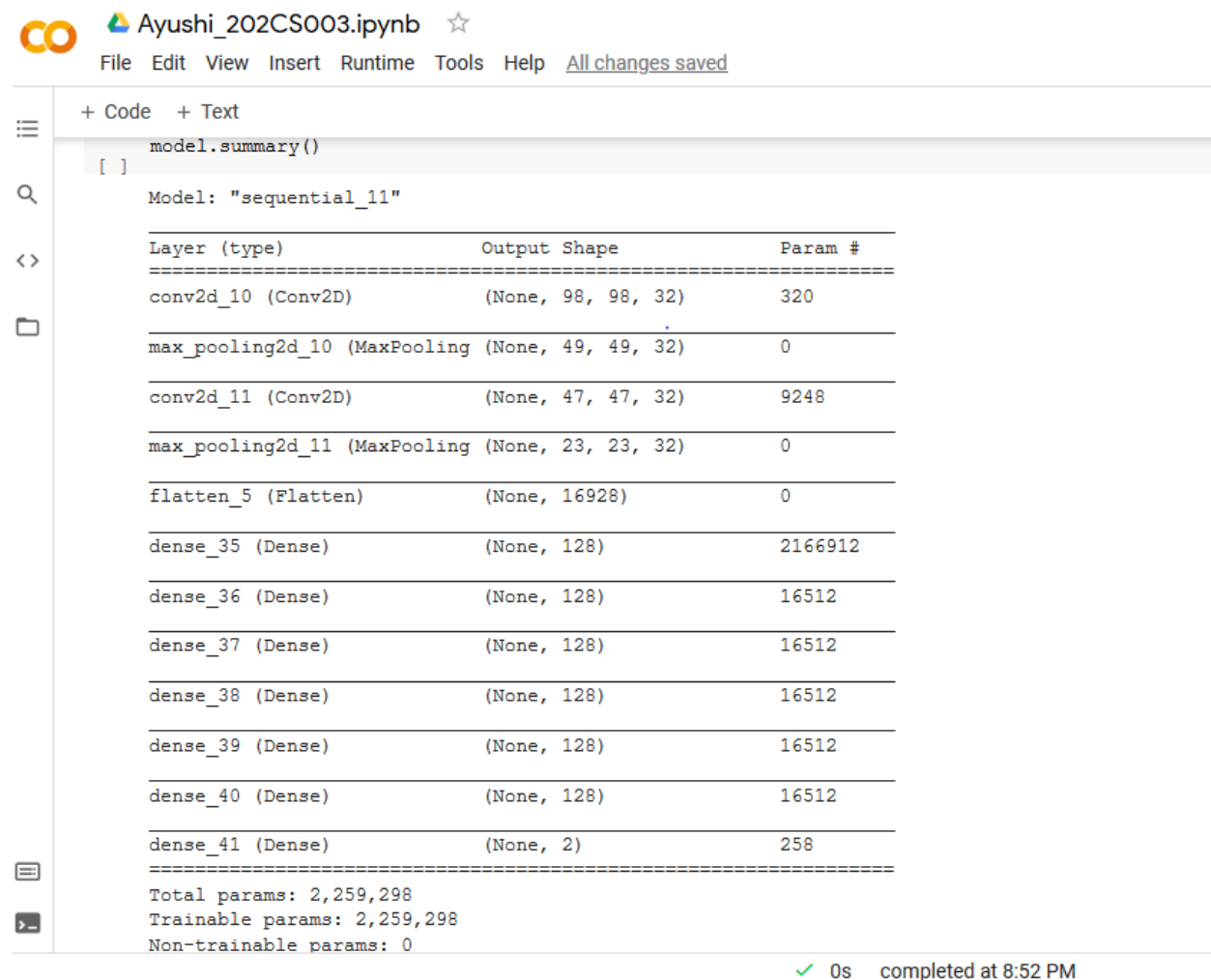
## Architecture Used:

I have selected a sequential model for covid and non-covid classification. Model have 14 layers :

1. Convolution layer that will be having 32 filters and his kernel size will be 3\*3 and the activation function used is RELU which is good as compared to other activation functions in a hidden layer.
2. dropout layer with dropout rate 0.5.
3. max-pooling of size 2\*2 .
4. Convolution layer which will be having filters of size 32 and kernel size are having 3\*3 and the activation function is RELU.
5. dropout layer with dropout rate 0.5.
6. max-pooling of size 2\*2 .
7. a flattening layer that helps to convert the 2D size into a 1D size which will flatten the image.
8. 5 dense layers with RELU as activation function
9. the last layer is a dense layer which is having a activation as SIGMOID.

I have used an ADAM optimizer with a learning rate of 0.001 and have a loss function as a categorical cross-entropy and having a batch size of 32 and run for 30 epochs with validation split as 0.1

In the above screenshot image the number of trainable parameters and size is mentioned.



```
model.summary()

[ ]

Model: "sequential_11"

Layer (type)                 Output Shape              Param #
=====
conv2d_10 (Conv2D)           (None, 98, 98, 32)        320
max_pooling2d_10 (MaxPooling (None, 49, 49, 32)        0
conv2d_11 (Conv2D)           (None, 47, 47, 32)       9248
max_pooling2d_11 (MaxPooling (None, 23, 23, 32)        0
flatten_5 (Flatten)          (None, 16928)             0
dense_35 (Dense)              (None, 128)              2166912
dense_36 (Dense)              (None, 128)              16512
dense_37 (Dense)              (None, 128)              16512
dense_38 (Dense)              (None, 128)              16512
dense_39 (Dense)              (None, 128)              16512
dense_40 (Dense)              (None, 128)              16512
dense_41 (Dense)              (None, 2)                 258
=====
Total params: 2,259,298
Trainable params: 2,259,298
Non-trainable params: 0
```

✓ 0s completed at 8:52 PM

## Novelty Adopted: I

While designing the architecture for the model, I have applied several standard architectures like VGG, AlexNet and GoogLeNet and found that accuracy of the model was not satisfactory hence I designed my own model. Other than that, for optimizers I have used Adam Optimizer. Also for the outer layer earlier I have used SoftMax activation function and then after experimenting switched to Sigmoid activation function. Lastly the loss function which is used is Categorical Cross Entropy loss function.

# Results:

## Test Results

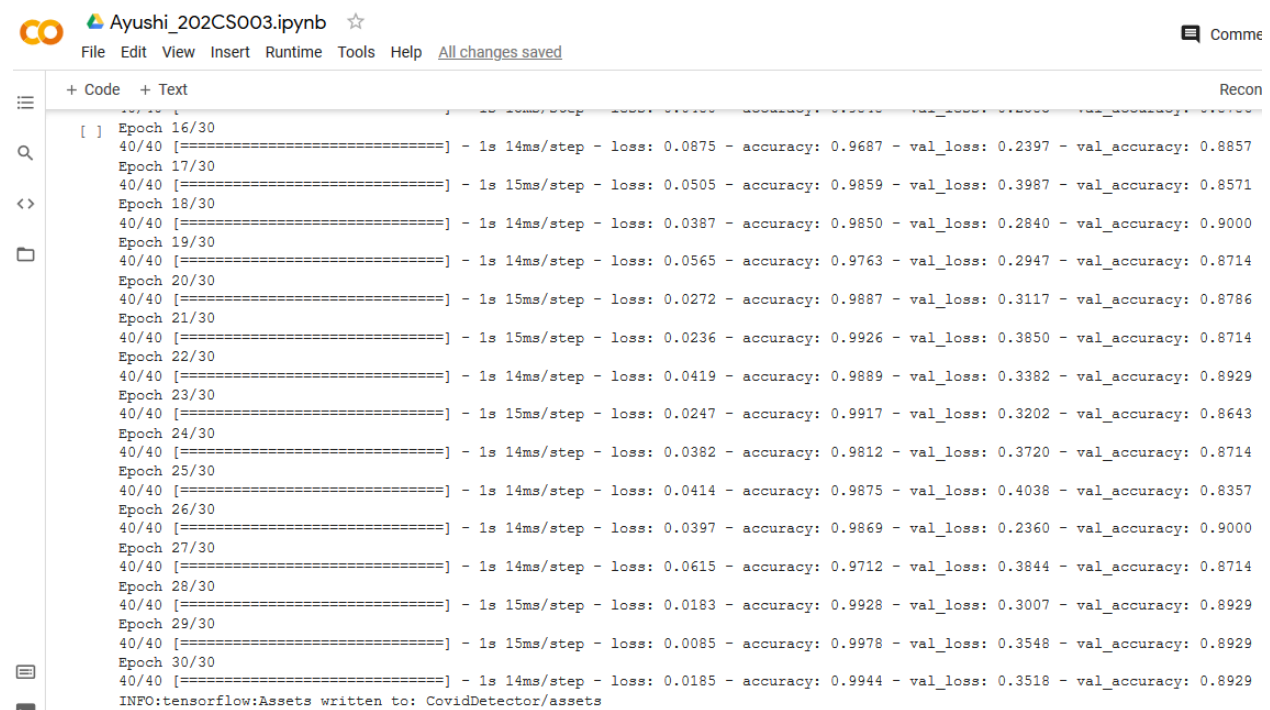
I ran the model for 30 epoch and got

Training accuracy = 99.44%.

Validation accuracy = 89.29%.

Training loss = 0.0185.

*screenshots of the results:*



The screenshot shows a Jupyter Notebook interface with a file named 'Ayushi\_202CS003.ipynb'. The notebook contains a list of 30 epochs, each with its corresponding training and validation metrics. The metrics include training loss, training accuracy, validation loss, and validation accuracy. The training accuracy is consistently high, starting at 0.9687 and ending at 0.9944. The validation accuracy is consistently lower, starting at 0.8857 and ending at 0.8929. The training loss is consistently low, starting at 0.0875 and ending at 0.0185. The validation loss is consistently higher, starting at 0.2397 and ending at 0.3518.

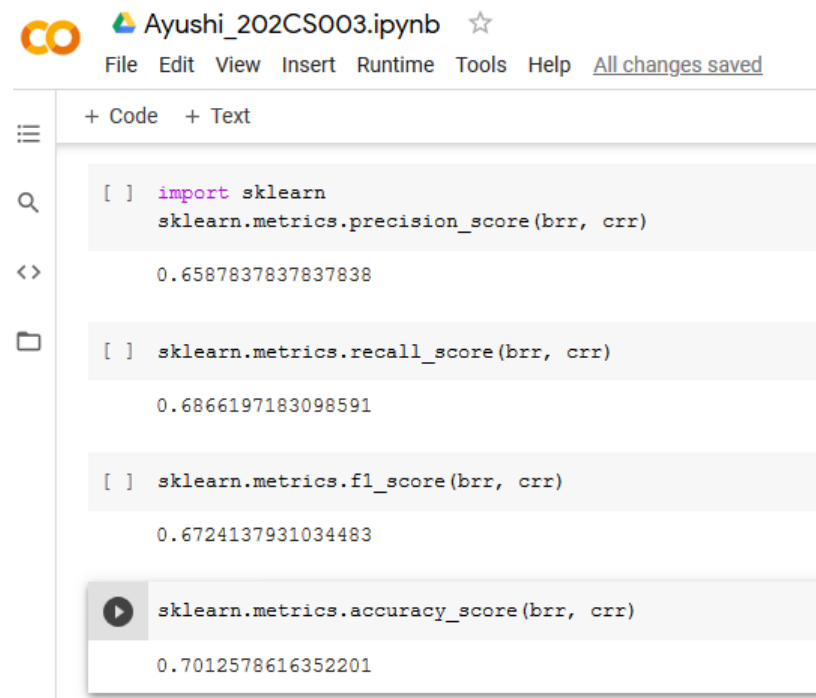
Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
16/30	0.0875	0.9687	0.2397	0.8857
17/30	0.0505	0.9859	0.3987	0.8571
18/30	0.0387	0.9850	0.2840	0.9000
19/30	0.0565	0.9763	0.2947	0.8714
20/30	0.0272	0.9887	0.3117	0.8786
21/30	0.0236	0.9926	0.3850	0.8714
22/30	0.0419	0.9889	0.3382	0.8929
23/30	0.0247	0.9917	0.3202	0.8643
24/30	0.0382	0.9812	0.3720	0.8714
25/30	0.0414	0.9875	0.4038	0.8357
26/30	0.0397	0.9869	0.2360	0.9000
27/30	0.0615	0.9712	0.3844	0.8714
28/30	0.0183	0.9928	0.3007	0.8929
29/30	0.0085	0.9978	0.3548	0.8929
30/30	0.0185	0.9944	0.3518	0.8929

INFO:tensorflow:Assets written to: CovidDetector/assets

## Testing Results

Parameter	Value
Accuracy	<b>0.7012578616352201</b>
Precision	<b>0.6587537837837838</b>
Recall	<b>0.6866197183098591</b>
F1 score	<b>0.6724137931034483</b>
Confusion Matrix	<b>[[251 101] [ 89 195]]</b>

screenshots of the results.



The screenshot shows a Jupyter Notebook titled "Ayushi\_202CS003.ipynb". The interface includes a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a status bar indicating "All changes saved". On the left, there is a sidebar with icons for a list, search, expand/collapse, and a file explorer. The main area displays four code cells, each followed by its output:

```
[ ] import sklearn
sklearn.metrics.precision_score(brr, crr)

0.6587837837837838

[ ] sklearn.metrics.recall_score(brr, crr)

0.6866197183098591

[ ] sklearn.metrics.f1_score(brr, crr)

0.6724137931034483

[ ] sklearn.metrics.accuracy_score(brr, crr)

0.7012578616352201
```



Ayushi\_202CS003.ipynb ☆

File Edit View Insert Runtime Tools Help



+ Code + Text



```
[ ] cm=confusion_matrix(brr, crr)
    print(cm)
```

```
[[251 101]
 [ 89 195]]
```



```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
sn.set(font_scale=1.2) # for label size
sn.heatmap(cm, annot=True, annot_kws={"size": 24})
plt.show()
#categories=['COVID','non-COVID']
#           =[ 0      , 1      ]
```

