TEST DATA: Test dataset contains 193 test images. Here it is. Download or view.

Model	Input Shape	No of Parameters (in Million)	Model Size (in MB)	Accuracy On Colab	Accuracy On Raspberry Pi	Average Runtime on Pi of a single Image Using Push Button (in sec)	Average Runtime on Pi of a single Image from Terminal (in sec)
Normal Original	(240x240x3)	7.25	55	98.45	x	x	x
TFLite	do	do	28	97.93	97.92	-	4.8414
Quantized	do	do	7	97.93	97.92	5.38	4.8931
DWT Original	(121x121x1)	0.786	6	96.37	x	x	x
TFLite	do	do	3.1	96.37	95.85	-	0.3346
Quantized	do	do	0.75	95.85	95.33	0.24	0.3342
CS Original	(100x240x1)	0.8		89	x	x	x
TFLite	do	do	3.2	89	-	-	-
Quantized	do	do	0.8	89	-	-	-

x marked parts are not necessary.

Blue coloured words are hyperlink. So, click on that to get the desired file or folder.

Code for testing accuracy on Raspberry Pi:

1. **Normal:** click here to view or download.
2. **DWT:** click here to view or download.
3. **CS:**

The codes are in .py format. Remember one thing. When you run the DWT code, one more library is required. You must install the PyWavelets library. View documentation for installing.

The links and results for CS will be updated very soon after some fine tuning.

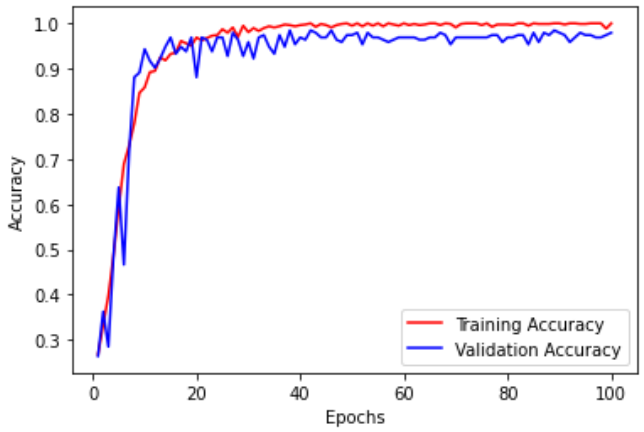
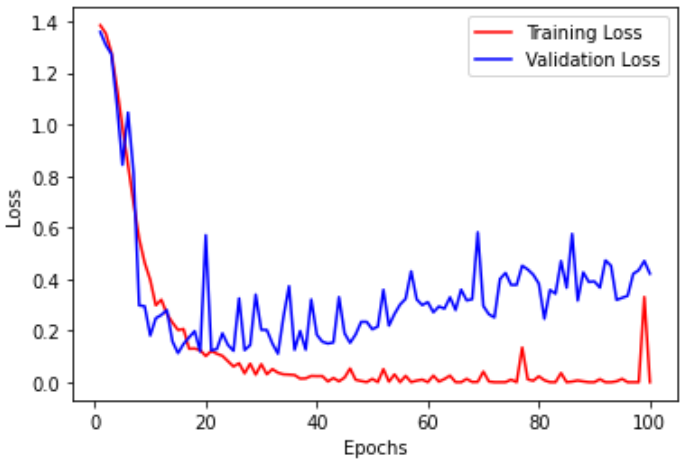
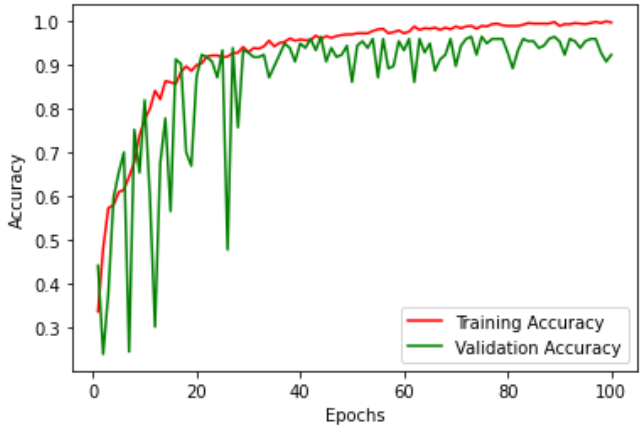
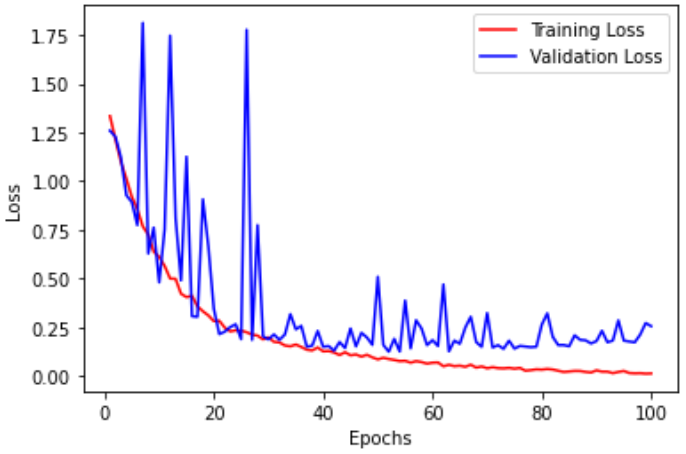
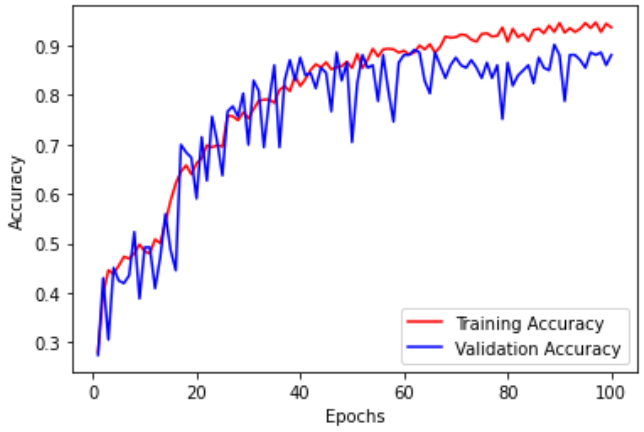
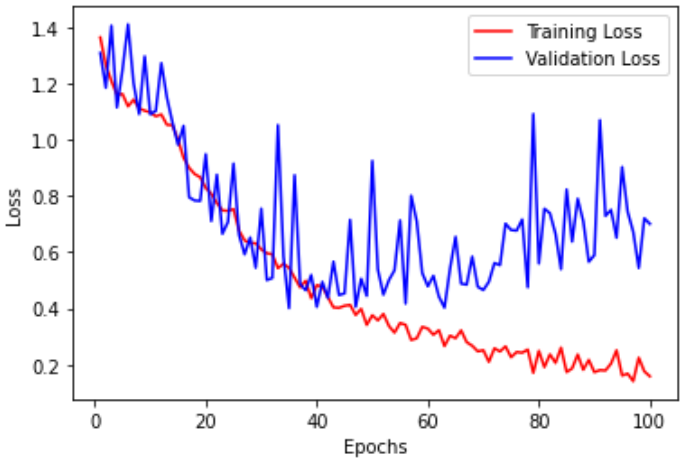
The folder which contains everything is here, view it.

Colab Notebook for training:

- **Normal:** view here

- **DWT:** view here
- **CS:** view here

Graphs:

Model	Accuracy	Loss
Normal		
DWT		
CS		

Flowchart:

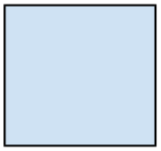
Import libraries → Read all images → Feed them into a model with 3 conv layers → save trained model → convert into TFLite → Quantization

Import libraries → Read images and perform DWT operation→ Feed them into a model with 3 conv layers → save trained model → convert into TFLite → Quantization

Import libraries → Read all images → Take every image and transform them into DCT domain → Compress them by columns → Feed them into a model with 3 conv layers → save trained model → convert into TFLite → Quantization

Pre-Conv layer:

240 x 240 x 1



16 x 16 x 1
3 x 3

80 x 80 x 1

Parameters: `233,157`
`Input shape:` `80 x 80 x 1`
Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	95.34%	94.82%	94.3%
Model size		0.89 MB	0.23 MB

The colab notebook is here.

DWT_Reconstructed:
Read input as 240 x 240. Rescale to 256 x 256.
HWT upto 8th level. Set 0 to the LL subband and reconstruct then so size becomes 256 x 256.

Parameters: `8,688,068`
`Input shape:` `256 x 256 x 1`
Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	97.92%	97.41%	96.89%
Model size	66 MB	33 MB	8.3 MB

This DWT decomposition and reconstruction method will help us to reduce the noise. As our current dataset was a manipulated dataset so that, when we will use real-world data this method will help us.
As here, I have decomposed upto 8th level and then reconstructed upto 8th level, so the output image has a size of 256 x 256. But if we reconstruct upto an intermediate level like 7th or 6th level, then the output image size will be 128 x 128 or 64 x 64 respectively. So we can reduce both noise and size.

The colab notebook is here.

DWT RECONSTRUCTION 7th LEVEL:

Parameters: `896,116`

`Input shape:` `128 x 128 x 1`

Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	94.3	93.78	94.3
Model size	7	3.42	0.86

As here, we have decomposed upto 8th level and then reconstructed upto 8th level, so the output image has a size of 128 x 128 x 1.

The colab notebook is here.

DWT RECONSTRUCTION 6th LEVEL:

Parameters: `175,556`

`Input shape:` `64 x 64 x 1`

Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	91.70	91.19	91.19
Model size (MB)	1	0.68	0.18

As here, we have decomposed upto 8th level and then reconstructed upto 6th level, so the output image has a size of 64 x 64 x 1.

The colab notebook is here.

DWT Compression + Pre-Conv:

Parameters: `135,109`

`Input shape:` `121 x 121 x 1`

Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	92.23	91.70	91.70
Model size (MB)	1	0.52	0.14

The colab notebook is here.

SPARK MODEL:

Parameters: `136,500`

`Input shape:` `240 x 240 x 3`

Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	92.75	92.23	92.23
Model size (MB)	2	0.53	0.18

The colab notebook is here.

DWT compression + Spark:

Parameters: `18,484`

`Input shape:` `121 x 121 x 1`

Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	96.37	95.85	95.85
Model size (MB)	0.27	0.08	0.042

The colab notebook is here.

Pre-Conv + Spark

Parameters: `18,173`

`Input shape:` `240 x 240 x 1`

Accuracy & Model size:

	Original	TFLite	Quantized
Accuracy	90.67%	90.15%	89.12%
Model size (MB)	0.26	`0.079`	`0.04`

The colab notebook is here.

Model		Accuracy (%)	Avg Runtime (s)	Size (MB)
Baseline	Original	93.45	0.19	55
	TFLite	93.93	0.16	28
	Quantized	93.92	10	7
Baseline + DWT	Original	94.37	0.09	6
	TFLite	94.85	0.11	3
	Quantized	94.34	0.12	0.756
Spark	Original	95.74	0.052	2
	TFLite	95.22	0.057	0.534
	Quantized	95.27	0.58	0.188
DWT + Spark	Original	96.37	0.065	0.265
	TFLite	95.85	0.04	0.08
	Quantized	95.85	0.045	0.04199
DWT Reconstruction upto level 6	Original	91.7	0.034	1
	TFLite	91.19	0.015	0.67
	Quantized	90.67	0.018	0.18
DWT Reconstruction upto level 7	Original	94.30	0.121	7
	TFLite	93.78	0.137	3
	Quantized	93.78	0.134	0.87
Pre-Conv	Original	95.34	0.065	2
	TFLite	94.81	0.056	0.896
	Quantized	94.3	0.061	0.232
Pre-Conv + Spark	Original	90.67	0.057	0.263
	TFLite	90.16	0.042	0.079
	Quantized	89.12	0.039	0.041
Pre-Conv + DWT	Original	92.23	0.035	1
	TFLite	91.71	0.016	0.521
	Quantized	91.71	0.018	0.144

The red coloured data is not from Raspberry Pi. They are done on my Ubuntu Terminal because these 2 models are not running properly on Raspberry Pi. Although they are running properly on my (Arindam Majee) own Ubuntu terminal.

Pink-coloured data is the overall best performance and blue data is best on respective columns.

DWT TIME = 0.00488 SEC
SPARK = 0.0657 SEC
DWT + SPARK = 0.05985 SEC

Problem:
The hardware team needs the shape of each layer to be in multiples of 14. There are certain problems to achieve that:

- The spark module needs to have a zero padding since we are concatenating a 3x3 and 1x1 convolution at some point in its execution, so the spark module does not downsample the size.
- If we do away with the spark module and use separable convolution layers instead, then too, to achieve 14x height and width in each layer will need a 15x15 filter for the convolution, which will hamper the results and accuracy of the model greatly since we are using 3x3 layers here generally.

To do:

- A pipeline to create an executable code where an input image dumped into the model will give out all the stats and segmented image in the same folder. This pipeline consists of two components:
    - The classification model which will produce a probability of which scan type this is.
    - The segmentation model which will output a segmented image.
- This model will be uploaded to github and it will later be used to train our model using a different kind of data.
- Make a presentation depicting the total pipeline of the work to be done

Done:
Dimensions changed to multiples of 15(240,120,60,30,15)

Done:
After changing dimensions in the classification model the accuracy has dropped drastically so we will need a bit more time tuning the parameters.