07 May 2019

Build an AI Model to detect the crack Defect
Deploy the model on Smartphone/Cloud

Sagar Bhure

Table of Content

JBM Group Assignment

I would first like to share a research work from my past in the field of Image Processing and then keeping this work from past I would proceed to the assignment of Developing AI model to Model to detect the crack Defect from sample. After which I would share a Mobile application which will predict the crack Defect in real time.

1. Review of Masters Thesis

Masters Thesis : *Dispersion analysis of CNT/Epoxy composites and architecture of porous scaffold using Digital Image Processing*
Tools/Library Used : OpenCV, python, numpy, matplotlib, pandas, matlab

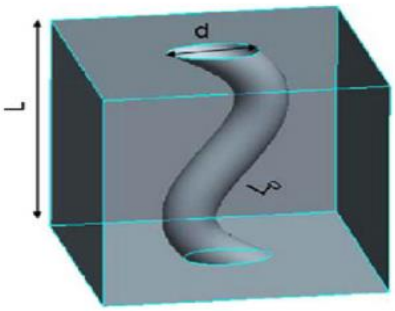
In my thesis titled ***Dispersion analysis of CNT/Epoxy composites and architecture of porous scaffold using Digital Image Processing*** wherein I wrote a code in python using OpenCV library to detect horizontal/vertical lines(fibers) using canny edge detection,
Which was further extended to measure the size of objects in the image. For the research purpose my object was an scaffold material which is used for bone regeneration. Various fabrication techniques have been used to prepare 3d scaffolds. Some of the traditional techniques being salt leaching, gas forming, phase separation and freeze dying. All of the above-mentioned techniques do not allow precise controlled fabrication of internal scaffold structure. Thus with the advancement in manufacturing methods, with the use of rapid prototyping and 3D printing techniques using computer-aided-design(CAD) modeling fabrication of such complex architectures has become feasible. Several scaffold fabrication techniques have been developed based on the end requirements of quantitative properties of the scaffold. The Base requirements of scaffolds which we have mainly focused in my current research were mainly porosity, interconnectivity and tortuosity of pores. My research aims to determine the mentioned three quantitative properties of the scaffold which are porosity , interconnectivity and tortuosity, using non-destructing mechanism (micro-CT scan). Which gives the stack wise images of any 3d object, scaffold in our case.

The 2D calculation of porosity is made by dividing the area occupied by pores by the total area of the image this can be simply done by determining the ratio of pixels associated with pores PP_i to the number contained in whole image PP_t. We evaluated 2D porosity in the stacks of images/frames, in XY-plane from z = 0 to z = thickness, obtained from micro-CT analysis of scaffold material. (refer fig 1) below to have a rough idea how pore detection algorithm was developed.

Pore interconnectivity is considered an important structural parameter that may affect the biological performance of a material by influencing fluid permeation, cell migration tissue ingrowth. In this section I developed Throat Detection algorithm on Scaffold material which was awarded for being the breakthrough algorithm in the field of Bioengineering by Dept of Mechanical Engineering, IIT Kanpur. Which basically identifies if we have two or pores in a surface which shares a common channel (refer Fig 3) if yes we should be able to determine the diameter of the common portion, so called throat.

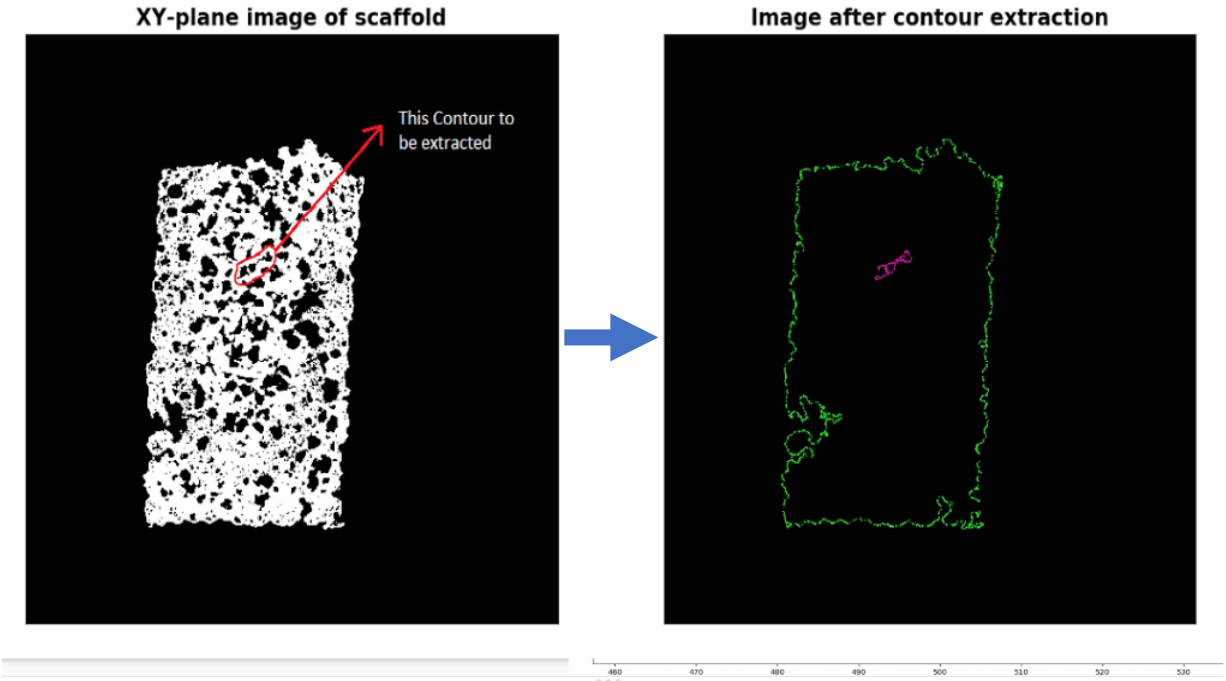
Last of them was tortuosity measurement of 3D printed sample, simplest definition of tortuosity is defined as the ratio of shortest distance take by fluid molecule to the shortest straight line that connects the entry point and exit point In current research work a 3d printed solid cube sample was manufactured using Acrylonitrile butadiene styerene(ABS) plastic with a tapered hole in the center (refer fig 4), where in We will virtually recreate his model using image processing and 3D modelling software (inventor in our case),thus validating or measurements virtually followed by tortuosity calculation of the sample.
When shape detection algorithm is applied to the bottom/top view of this sample to extract contour (or profile) of outer square and inscribed circle we can calculate the circumference of the bottom/top view of this sample .

This work was highly recognized in the field of digital image processing in the department of Aerospace Engineering and Mechanical Engineering of Indian Institute of Technology Kanpur.

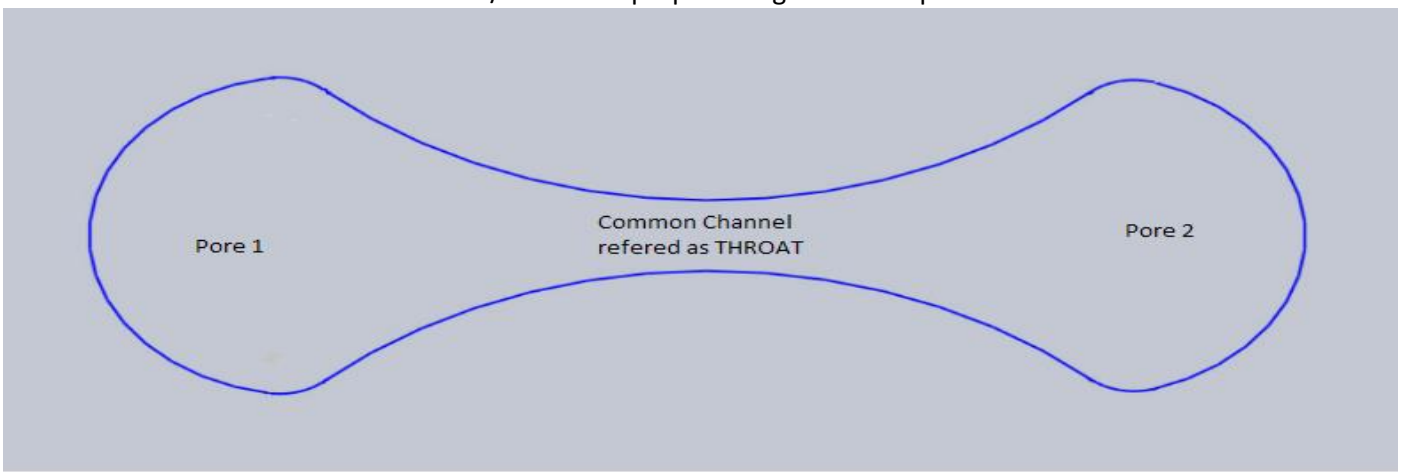


$$T = \frac{L_0}{L}$$

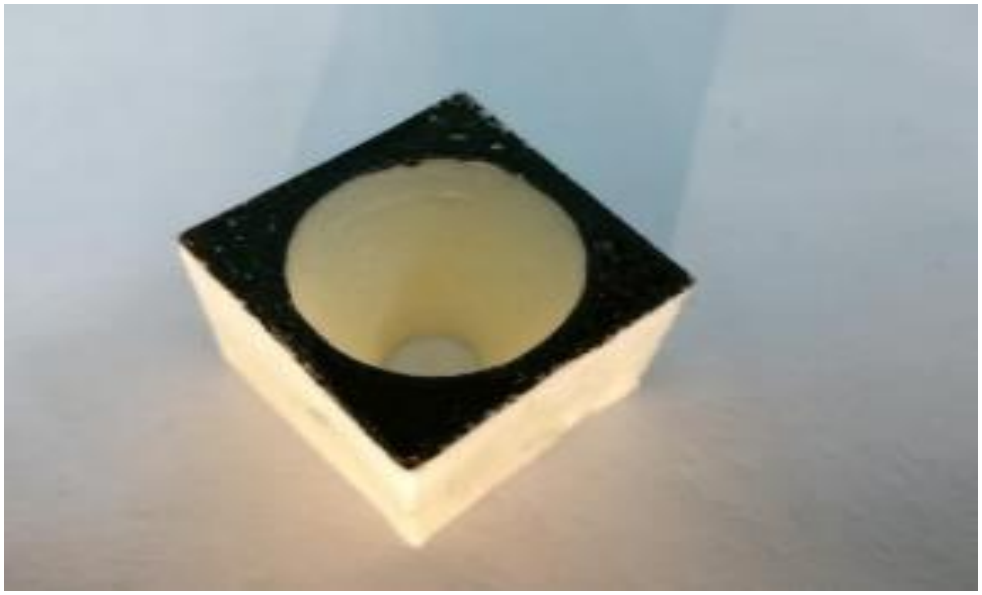
Fig(1) Schematic representation of tortuosity taking account of single channel within the scaffold



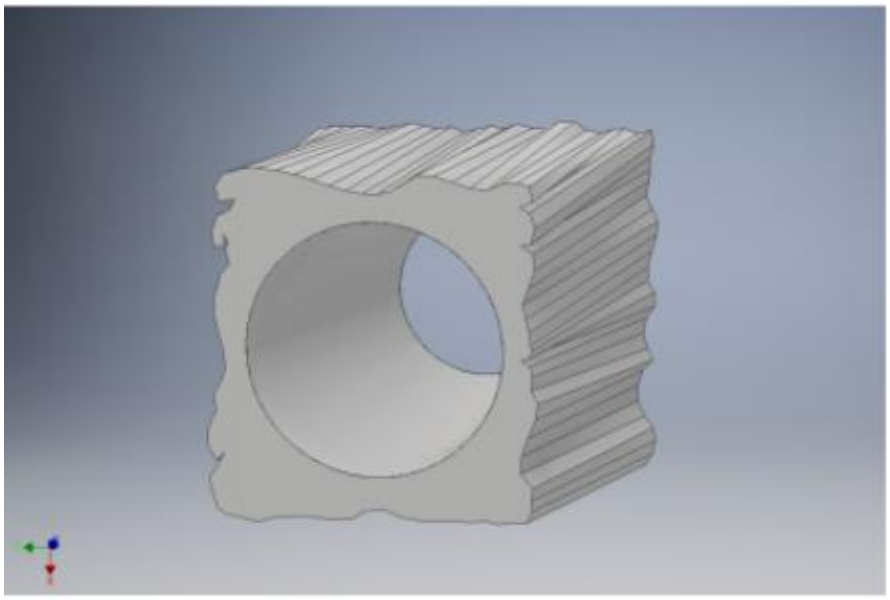
Fig(2) One of the image from image stack and algorithm to extract/detect pore from the image For further processing.
Curve/contour in purple on right is called pore



Fig(3) Two pore defination



Fig(4) 3D printed sample



Fig(5) Virtual Model of above 3D printed sample

2. Problem Statement: **Build an AI Model to detect the crack Defect**

In this Assignment I will try to build an AI model which detects the AI Model that can successfully predict the quality of a given sample(Healthy/Defect) with the validation **accuracy of 93.55%** (refer appendix A), and will deploy this model into my android smartphone. I will also touch upon how we can deploy this model in Microsoft Azure Cloud, so that the Model is accessible Globally on any device in the end of the assignment

2.1 Introduction

The machine parts recognition in automated assembly systems is entirely different from general object recognition; moreover the ability of human to distinguish between healthy and unhealthy machine parts are good but it is a complicated task for a machine. In general manual defect detection by human inspectors are impractical with fast moving machine parts on conveyor in addition it is expensive, subjective, inaccurate, eye straining and other health issues to quality control inspectors

By considering these issues, a computer vision/Machine Learning based non-contact inspection technique is developed in this assignment for defect detection in industrial machine parts by image processing and Machine Learning techniques. This can be implemented in industries with the help of industrial robot used in assembly process and industrial inspection systems (Fig 1).

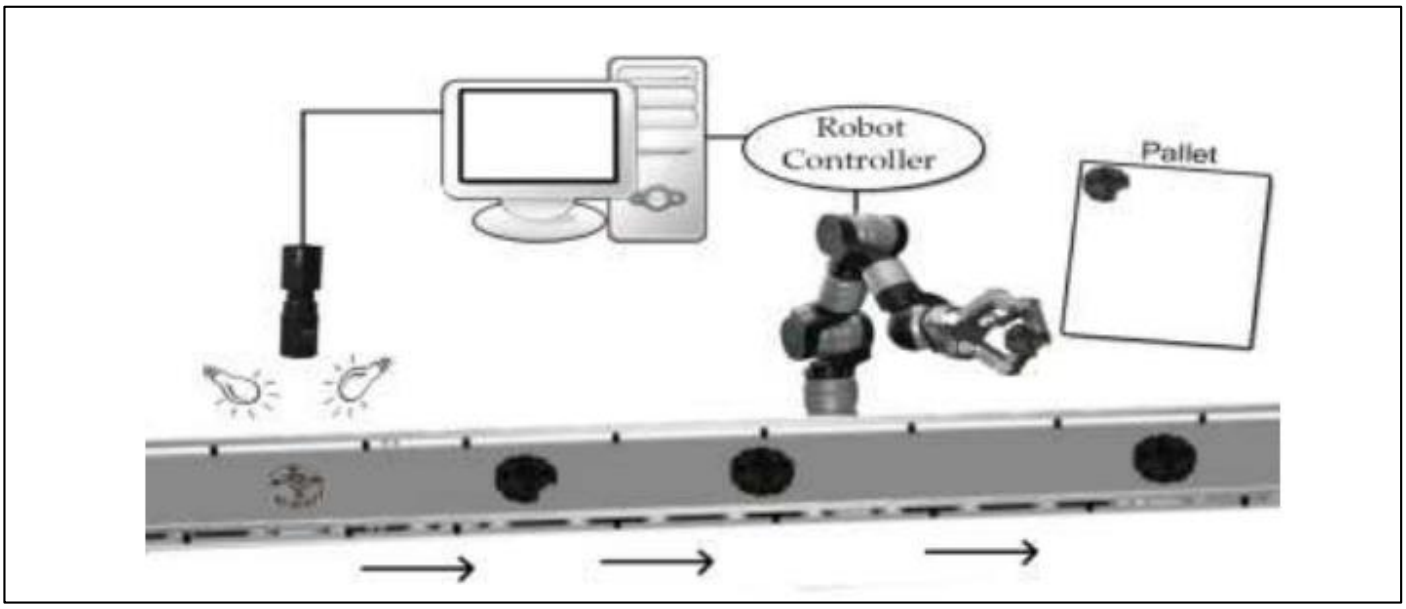
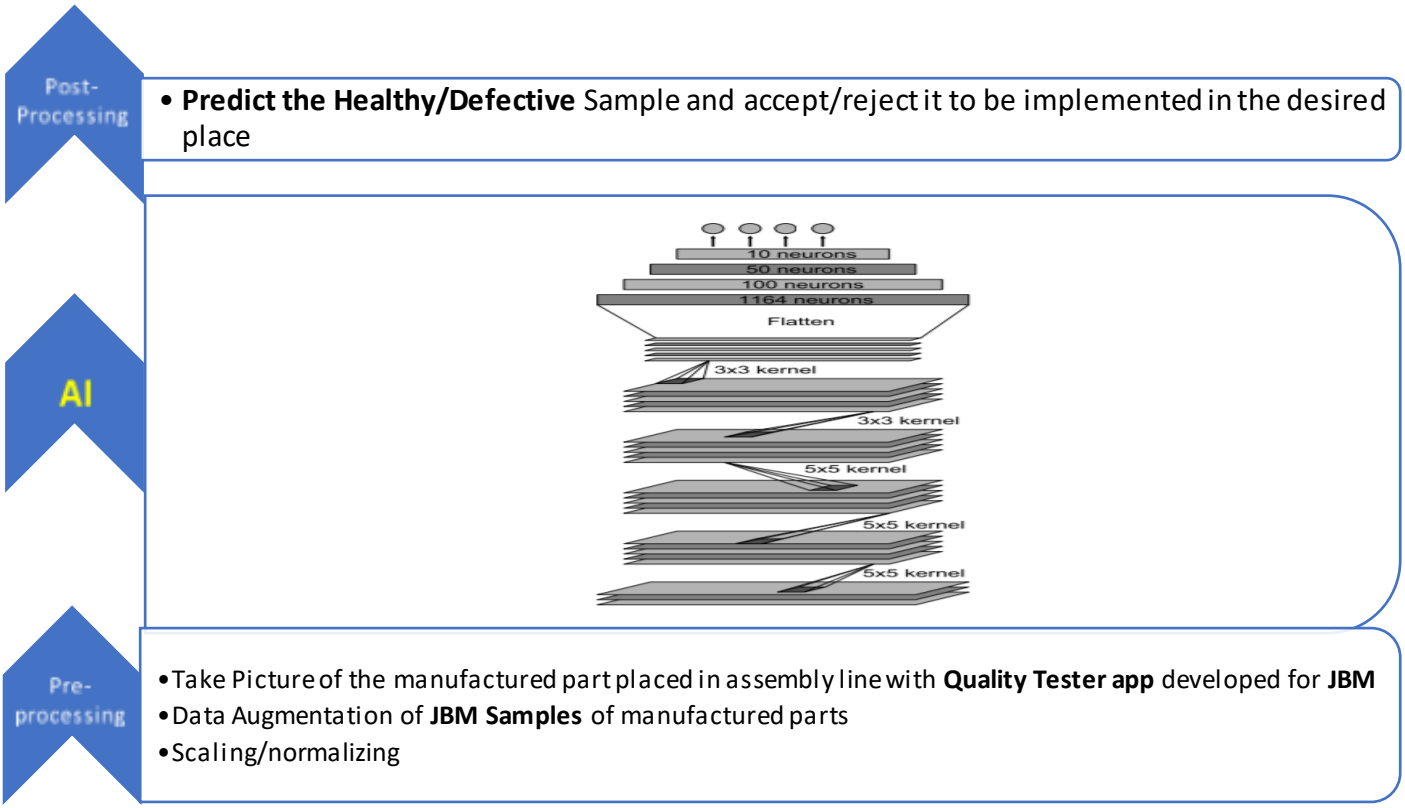


Fig 1. A proposed diagram for machine parts defect detection in model assembly line

2.2 Proposed Approach: Defect Identification

The objects moving in the conveyor as shown in Fig.1 are captured by vertical direction of the overhead camera in the automated assembly lines. The machine parts are placed on the conveyor with predetermined sequence and the timing is synchronized with operation of assembly system. Here the objects on the conveyor are not occluded and they posses rotational orientations. That is, machine part to be captured and the camera are stationary; this simulates the situation in which the camera will extract the exact shape from the top view and only one part is present per view.

Images from camera is passed into the pipeline as shown in Fig(2), at the end we can detect the quality(healthy/defective) of the part instantaneously with the app developed in real time as I have implemented integrated camera in the app.



Fig(2) Above pipeline shows what need to be done from taking a picture from the app to predicting its quality

2.3 Pre-Processed Data set and Network Implementation

Shape of training dataset 188
shape of validation set is 62

Train dataset shape: (188, 224, 224, 3) Validation dataset shape: (62, 224, 224, 3)
Train Bottleneck Features: (188, 51200) Validation Bottleneck Features: (62, 51200)

Layer (type)	Output Shape	Param #
dense_127 (Dense)	(None, 524)	26829324
dropout_106 (Dropout)	(None, 524)	0
dense_128 (Dense)	(None, 128)	67200
dropout_107 (Dropout)	(None, 128)	0
dense_129 (Dense)	(None, 64)	8256
dropout_108 (Dropout)	(None, 64)	0
dense_130 (Dense)	(None, 32)	2080

dropout_109 (Dropout) (None, 32) 0

dense_131 (Dense) (None, 32) 1056

dropout_110 (Dropout) (None, 32) 0

dense_132 (Dense) (None, 32) 1056

dropout_111 (Dropout) (None, 32) 0

dense_133 (Dense) (None, 1) 33

Total params = Sum of number of parameters from each layer
Total params = 26829324 + 67200+ 8256 +2080 + 1056 +1056 + 33

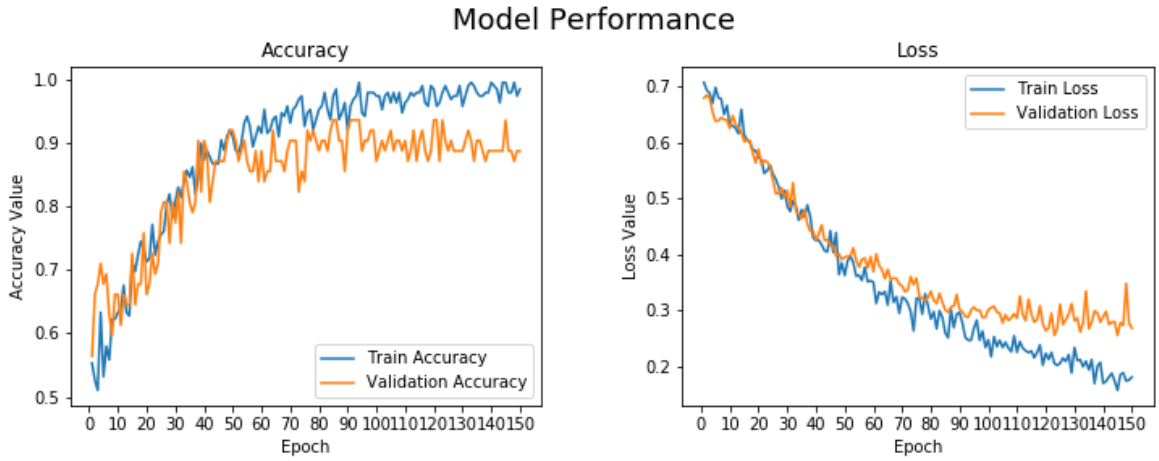
Total params: 26,909,005
Trainable params: 26,909,005
Non-trainable params: 0

- Since the size of the dataset was small so training the model from scratch might not give good accuracy scores so the concept of **Transfer Learning** as used to train the model.
- During Pre-Processing entire dataset was divided between training and validation sets according to Split Ratio of 0.25 and later images was loaded with size (244,244) to be fed into the network.
- By default, it uses the feature vectors computed by Inception V3 trained on ImageNet. The developed AI Model uses sequential or KNF model which allows us to create models layer-by-layer.
- Dense layer used
    - 5 dense layers used with relu activation
    - 1 dense layers used with backend.tanh (or elementwise tanh) activation
    - 1 dense layers used with sigmoidactivation
- Between two dense layers Dropout of rate 0.3 is used to avoid overfitting
- Hyperparameters used
    - batch_size = 32
    - epoch = 150
    - input_shape = (224, 224, 3)
    - Learning Rate = 1e-5 (RMSprop)

  After Train on 188 samples, validate on 62 samples we get the following Accuracy(refer Appendix A)

Loss : 0.2895
Accuracy: 0.9355 (or 93.55 %)

Following is the graph of Training/validation accuracy(left)
And the Graph for Training/validation loss(right]) for training of 150 epoch



2.4 Guide to use Project

2.4.1 Training Classifier

The folder `./JBMClassification2` contains subfolder YE358311_Healthy & YE358311_defects which has our dataset

- Open playingwithmodel.py
- Cd to JBMClassification2
- Make sure to modify the path to your local dir in Line No. 43
- Now the Classifier can be trained by executing the below command.
  - python playingwithmodels.py
- Saved network at the end or training is generated with file name JBM_Classification.h5

2.4.2 Testing the model

Already trained model present in `./JBMClassificatrion2/` (with name JBM_Classification.h5)directory can be used for testing on new images. Execute the below command.

- Open test_model.py
- Cd to JBMClassification2
- Modify the path for trained_model_path (`./JBMClassificatrion2/JBM_Classification.h5`), incept_model_path(`./JBMClassificatrion2/incept_model.h5`) Line No. 31 & 32
- Make a Test folder inside the JBMClassification2 and keep test images(Healthy/Defective) inside this folder with image name being test.img Line No. 33 in test_model.py

- OR

```
python ./test_model.py \
--images_path={Path of the Image file or Folder of images} \
--trained_model_path  = ./ JBM_Classification.h5\
--incept_model_path = ./incept_model.h5
```

3. Mobile Application: Quality Tester by JBM

I deployed above model on a smartphone using android application which I named as **Quality Tester by JBM**
Below is the app implementation and the UI, Refer to the video for live demo

The application has an integrated camera which take images of the manufactured parts uses TensorFlow library on the app to classify images.

3.1 Usage

Flow of the app is simple:

- Take a photo.
- Classify if it's Healthy or Defective.
- Show the results.

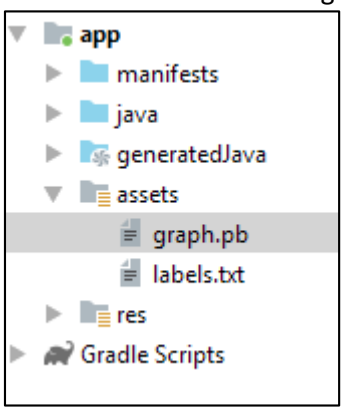
3.2 Classifier

The assets parameter is an AssetManager instance. The other parameters are placed in Constants class.

```
const val GRAPH_FILE_PATH = "file:///android_asset/graph.pb"
const val LABELS_FILE_PATH = "file:///android_asset/labels.txt"

const val GRAPH_INPUT_NAME = "Placeholder"
const val GRAPH_OUTPUT_NAME = "final_result"

const val IMAGE_SIZE = 224
const val COLOR_CHANNELS = 3

const val ASSETS_PATH = "file:///android_asset/"
```

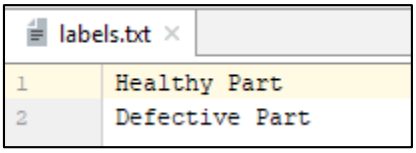
Ok, so what are they?

- GRAPH_FILE_PATH — path to our classifier file in the assets folder. We'll be using words 'classifier' or 'graph' alternately. Graph because our classifier is a saved neural network graph. You can also call it a 'model'.



Graph and labels files in assets folder

- LABELS_FILE_PATH — path to our labels file in assets folder. Simply put, labels are our possible results. Our classifier was trained to tell if a photo is hot or not. This is all it can do, so our labels are just 'hot' and 'not'.



Labels file

- GRAPH_INPUT_NAME — name of the classifier's input. We pass the image to the classifier via input.
- GRAPH_OUTPUT_NAME — name of the classifier's output. We get the result of the classification from the output.
- IMAGE_SIZE — the size of the image in pixels. Our classifier can understand images which are 224x224 pixels — that's how it was trained.

`ImageClassifier` properties:

- `inputName` - the name of the classifier's input (the photo pixels goes in there),
- `outputName` - the name of the classifier's output (the results can be found there),
- `imageSize` - the size of the photo,
- `labels` - the list of the labels (in our case "Healthy" and "Defective"),
- `imageBitmapPixels` - the array with bitmap pixels (int values before normalization),
- `imageNormalizedPixels` - the array with normalized pixels,
- `results` - the list with the results,
- **`tensorFlowInference`** - the TensorFlow API object (which is used for inference).

3.3 Classification process

The model can be found inside the `assets` folder together with the labels file. Before classification the photo needs to be prepared to fit the input of the classifier which is 224x224 pixels. Because of that the photo is resized and cropped which is happening inside the `ImageUtils`.

Prepared photo is passed to the `ImageClassifier`. The class responsibilities are as follows:

1. Nomalizing pixels of the photo - `preprocessImageToNormalizedFloats()` method.

2. Classifying - `classifyImageToOutputs()` method.
3. Getting the results - `getResults()` method.

For the classification process the instance of the `TensorFlowInferenceInterface` is used. The classification looks as follows:

1. Put the data to the classifier:
   `tensorFlowInference.feed(inputName, imageNormalizedPixels, 1L, imageSize, imageSize, COLOR_CHANNELS.toLong())`
2. Run the classifier:
   `tensorFlowInference.run(arrayOf(outputName), ENABLE_LOG_STATS)`
3. Get the results from the output:
   `tensorFlowInference.fetch(outputName, results)`

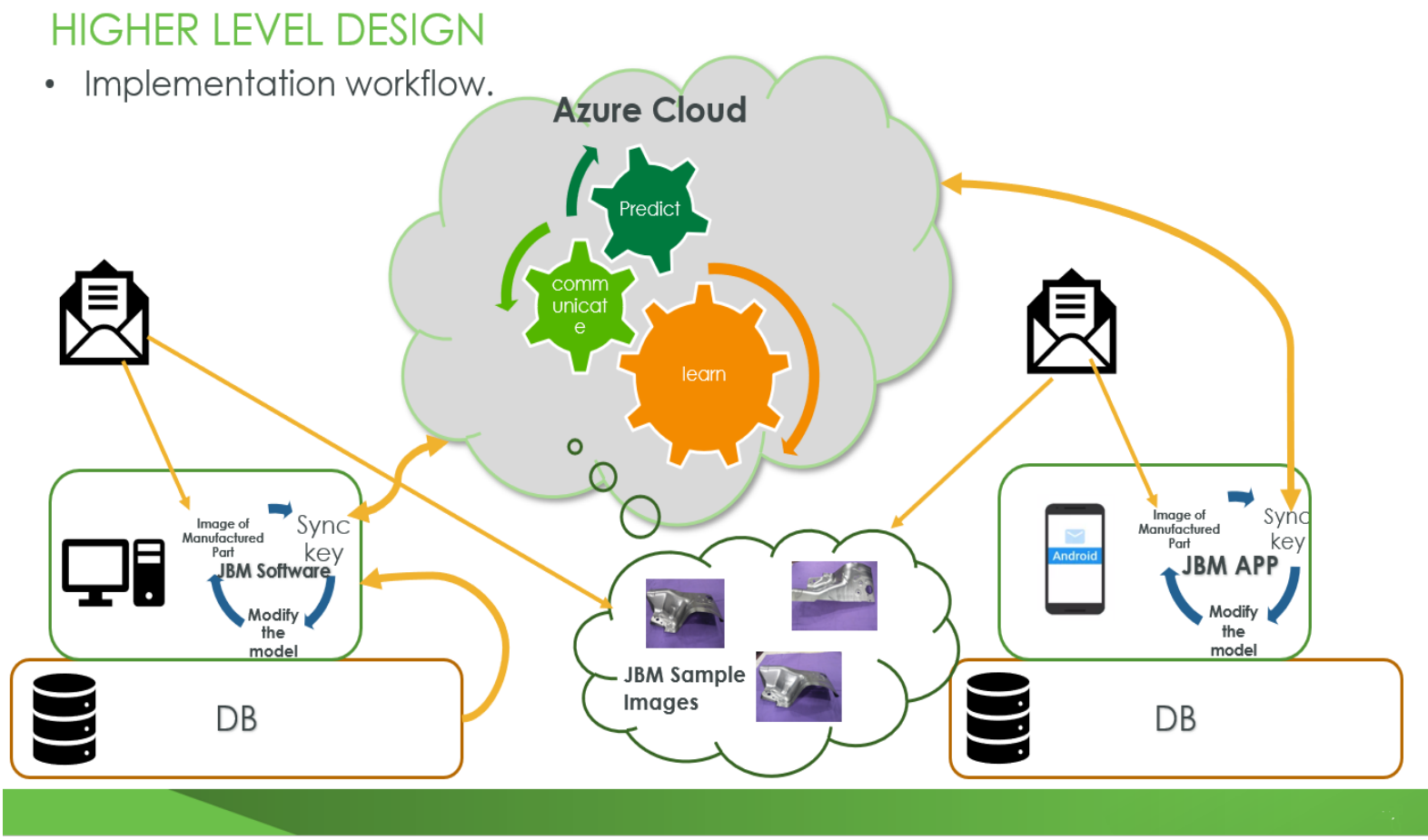
The results are then passed to the `MainActivity` and shown on the screen.

Refer to the Video in `./JBMClasification2/QualityTesterByJBM.mp4` for Demo.

This is an example of how to use TensorFlow library to classify images as Healthy or Defective, following is the link to github project to deploy the model on android app.

[IMP] Keep your saved network as a **".pb"** file inside assets folder

4. Deploying the Quality Tester App in Azure Cloud Network



- Train the model in the background on Azure
- AI model's for different parts being manufactured by JBM could be implemented in using Cloud.
- Different machines having the quality tester app could do the task irrespective of the geographical location
- Can be implemented in any smartphone having the app installed.

Appendix A

For verbose = 1

Epoch 1/150
188/188 [==============================] - 134s 711ms/step - loss: 0.7115 - acc: 0.5851 - val_loss: 0.6781 - val_acc: 0.5323
Epoch 2/150
188/188 [==============================] - 0s 3ms/step - loss: 0.6642 - acc: 0.5638 - val_loss: 0.6271 - val_acc: 0.6774
Epoch 3/150
188/188 [==============================] - 0s 3ms/step - loss: 0.6844 - acc: 0.5798 - val_loss: 0.6809 - val_acc: 0.6290
Epoch 4/150
188/188 [==============================] - 0s 3ms/step - loss: 0.6605 - acc: 0.6543 - val_loss: 0.6289 - val_acc: 0.7097
Epoch 5/150
188/188 [==============================] - 0s 3ms/step - loss: 0.6418 - acc: 0.6064 - val_loss: 0.5692 - val_acc: 0.7419
Epoch 6/150
188/188 [==============================] - 0s 3ms/step - loss: 0.6220 - acc: 0.6436 - val_loss: 0.5602 - val_acc: 0.7903
Epoch 7/150
188/188 [==============================] - 0s 3ms/step - loss: 0.5531 - acc: 0.7713 - val_loss: 0.5353 - val_acc: 0.7742
Epoch 8/150
188/188 [==============================] - 0s 3ms/step - loss: 0.5283 - acc: 0.7447 - val_loss: 0.7876 - val_acc: 0.5806
Epoch 9/150
188/188 [==============================] - 0s 3ms/step - loss: 0.7369 - acc: 0.5106 - val_loss: 0.6928 - val_acc: 0.4355
Epoch 10/150
188/188 [==============================] - 1s 3ms/step - loss: 0.5945 - acc: 0.7340 - val_loss: 0.5856 - val_acc: 0.7258
Epoch 11/150
188/188 [==============================] - 1s 3ms/step - loss: 0.5563 - acc: 0.7819 - val_loss: 0.4830 - val_acc: 0.7742
Epoch 12/150
188/188 [==============================] - 0s 3ms/step - loss: 0.4856 - acc: 0.7872 - val_loss: 0.4164 - val_acc: 0.8226
Epoch 13/150
188/188 [==============================] - 0s 3ms/step - loss: 0.3622 - acc: 0.8564 - val_loss: 0.5316 - val_acc: 0.7742
Epoch 14/150
188/188 [==============================] - 0s 3ms/step - loss: 0.3282 - acc: 0.8723 - val_loss: 0.3519 - val_acc: 0.8548
Epoch 15/150
188/188 [==============================] - 1s 3ms/step - loss: 0.2243 - acc: 0.9362 - val_loss: 0.7950 - val_acc: 0.6290
Epoch 16/150
188/188 [==============================] - 0s 3ms/step - loss: 0.4269 - acc: 0.8138 - val_loss: 0.5746 - val_acc: 0.6935
Epoch 17/150
188/188 [==============================] - 0s 3ms/step - loss: 0.4101 - acc: 0.8085 - val_loss: 0.4133 - val_acc: 0.8387
Epoch 18/150
188/188 [==============================] - 0s 3ms/step - loss: 0.2547 - acc: 0.9043 - val_loss: 0.5307 - val_acc: 0.7419
Epoch 19/150
188/188 [==============================] - 0s 3ms/step - loss: 0.1561 - acc: 0.9521 - val_loss: 0.2914 - val_acc: 0.8871
Epoch 20/150
188/188 [==============================] - 0s 3ms/step - loss: 0.1495 - acc: 0.9574 - val_loss: 0.3035 - val_acc: 0.9194
Epoch 21/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0699 - acc: 0.9787 - val_loss: 0.3342 - val_acc: 0.9032
Epoch 22/150
188/188 [==============================] - 0s 3ms/step - loss: 0.2049 - acc: 0.9149 - val_loss: 0.3243 - val_acc: 0.8710
Epoch 23/150
188/188 [==============================] - 0s 3ms/step - loss: 0.1580 - acc: 0.8989 - val_loss: 0.2986 - val_acc: 0.9032
Epoch 24/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0893 - acc: 0.9628 - val_loss: 0.2865 - val_acc: 0.9032

```
Epoch 25/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0181 - acc: 1.0000 - val_loss: 0.3796 - val_acc: 0.8548
Epoch 26/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0112 - acc: 1.0000 - val_loss: 0.3199 - val_acc: 0.9355
Epoch 27/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0053 - acc: 1.0000 - val_loss: 0.3403 - val_acc: 0.9355
Epoch 28/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0036 - acc: 1.0000 - val_loss: 0.3504 - val_acc: 0.9355
Epoch 29/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0021 - acc: 1.0000 - val_loss: 0.3622 - val_acc: 0.9194
Epoch 30/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0016 - acc: 1.0000 - val_loss: 0.3708 - val_acc: 0.9194
Epoch 31/150
188/188 [==============================] - 1s 3ms/step - loss: 0.0013 - acc: 1.0000 - val_loss: 0.3745 - val_acc: 0.9194
Epoch 32/150
188/188 [==============================] - 0s 3ms/step - loss: 0.0011 - acc: 1.0000 - val_loss: 0.3759 - val_acc: 0.9194
Epoch 33/150
188/188 [==============================] - 0s 3ms/step - loss: 9.3518e-04 - acc: 1.0000 - val_loss: 0.3764 - val_acc: 0.9355
Epoch 34/150
188/188 [==============================] - 0s 3ms/step - loss: 8.1332e-04 - acc: 1.0000 - val_loss: 0.3786 - val_acc: 0.9355
Epoch 35/150
188/188 [==============================] - 0s 3ms/step - loss: 7.4068e-04 - acc: 1.0000 - val_loss: 0.3816 - val_acc: 0.9355
Epoch 36/150
188/188 [==============================] - 0s 3ms/step - loss: 6.7510e-04 - acc: 1.0000 - val_loss: 0.3858 - val_acc: 0.9355
Epoch 37/150
188/188 [==============================] - 0s 3ms/step - loss: 6.2612e-04 - acc: 1.0000 - val_loss: 0.3888 - val_acc: 0.9355
Epoch 38/150
188/188 [==============================] - 0s 3ms/step - loss: 5.8649e-04 - acc: 1.0000 - val_loss: 0.3911 - val_acc: 0.9355
Epoch 39/150
188/188 [==============================] - 0s 3ms/step - loss: 5.5232e-04 - acc: 1.0000 - val_loss: 0.3933 - val_acc: 0.9355
Epoch 40/150
188/188 [==============================] - 0s 3ms/step - loss: 5.2361e-04 - acc: 1.0000 - val_loss: 0.3951 - val_acc: 0.9355
Epoch 41/150
188/188 [==============================] - 0s 3ms/step - loss: 4.9825e-04 - acc: 1.0000 - val_loss: 0.3968 - val_acc: 0.9355
Epoch 42/150
188/188 [==============================] - 0s 3ms/step - loss: 4.7416e-04 - acc: 1.0000 - val_loss: 0.3984 - val_acc: 0.9355
Epoch 43/150
188/188 [==============================] - 1s 3ms/step - loss: 4.5374e-04 - acc: 1.0000 - val_loss: 0.4000 - val_acc: 0.9355
Epoch 44/150
188/188 [==============================] - 0s 3ms/step - loss: 4.3531e-04 - acc: 1.0000 - val_loss: 0.4017 - val_acc: 0.9355
Epoch 45/150
188/188 [==============================] - 0s 3ms/step - loss: 4.1665e-04 - acc: 1.0000 - val_loss: 0.4030 - val_acc: 0.9355
Epoch 46/150
188/188 [==============================] - 0s 3ms/step - loss: 4.0079e-04 - acc: 1.0000 - val_loss: 0.4047 - val_acc: 0.9355
Epoch 47/150
188/188 [==============================] - 1s 3ms/step - loss: 3.8537e-04 - acc: 1.0000 - val_loss: 0.4064 - val_acc: 0.9355
Epoch 48/150
188/188 [==============================] - 0s 3ms/step - loss: 3.7194e-04 - acc: 1.0000 - val_loss: 0.4075 - val_acc: 0.9355
Epoch 49/150
188/188 [==============================] - 0s 3ms/step - loss: 3.5791e-04 - acc: 1.0000 - val_loss: 0.4086 - val_acc: 0.9355
Epoch 50/150
188/188 [==============================] - 1s 3ms/step - loss: 3.4574e-04 - acc: 1.0000 - val_loss: 0.4093 - val_acc: 0.9355
```

Epoch 51/150
188/188 [==============================] - 0s 3ms/step - loss: 3.3400e-04 - acc: 1.0000 - val_loss: 0.4103 - val_acc: 0.9355
Epoch 52/150
188/188 [==============================] - 0s 3ms/step - loss: 3.2359e-04 - acc: 1.0000 - val_loss: 0.4119 - val_acc: 0.9355
Epoch 53/150
188/188 [==============================] - 0s 3ms/step - loss: 3.1335e-04 - acc: 1.0000 - val_loss: 0.4128 - val_acc: 0.9355
Epoch 54/150
188/188 [==============================] - 0s 3ms/step - loss: 3.0386e-04 - acc: 1.0000 - val_loss: 0.4142 - val_acc: 0.9355
Epoch 55/150
188/188 [==============================] - 1s 3ms/step - loss: 2.9508e-04 - acc: 1.0000 - val_loss: 0.4152 - val_acc: 0.9355
Epoch 56/150
188/188 [==============================] - 0s 3ms/step - loss: 2.8708e-04 - acc: 1.0000 - val_loss: 0.4163 - val_acc: 0.9355
Epoch 57/150
188/188 [==============================] - 0s 3ms/step - loss: 2.7939e-04 - acc: 1.0000 - val_loss: 0.4175 - val_acc: 0.9355
Epoch 58/150
188/188 [==============================] - 0s 3ms/step - loss: 2.7195e-04 - acc: 1.0000 - val_loss: 0.4191 - val_acc: 0.9355
Epoch 59/150
188/188 [==============================] - 1s 3ms/step - loss: 2.6510e-04 - acc: 1.0000 - val_loss: 0.4203 - val_acc: 0.9355
Epoch 60/150
188/188 [==============================] - 0s 3ms/step - loss: 2.5879e-04 - acc: 1.0000 - val_loss: 0.4213 - val_acc: 0.9355
Epoch 61/150
188/188 [==============================] - 0s 3ms/step - loss: 2.5262e-04 - acc: 1.0000 - val_loss: 0.4227 - val_acc: 0.9355
Epoch 62/150
188/188 [==============================] - 0s 3ms/step - loss: 2.4671e-04 - acc: 1.0000 - val_loss: 0.4242 - val_acc: 0.9355
Epoch 63/150
188/188 [==============================] - 0s 3ms/step - loss: 2.4143e-04 - acc: 1.0000 - val_loss: 0.4258 - val_acc: 0.9355
Epoch 64/150
188/188 [==============================] - 0s 3ms/step - loss: 2.3581e-04 - acc: 1.0000 - val_loss: 0.4271 - val_acc: 0.9355
Epoch 65/150
188/188 [==============================] - 0s 3ms/step - loss: 2.3090e-04 - acc: 1.0000 - val_loss: 0.4287 - val_acc: 0.9355
Epoch 66/150
188/188 [==============================] - 0s 3ms/step - loss: 2.2571e-04 - acc: 1.0000 - val_loss: 0.4303 - val_acc: 0.9355
Epoch 67/150
188/188 [==============================] - 0s 3ms/step - loss: 2.2129e-04 - acc: 1.0000 - val_loss: 0.4324 - val_acc: 0.9355
Epoch 68/150
188/188 [==============================] - 0s 3ms/step - loss: 2.1694e-04 - acc: 1.0000 - val_loss: 0.4343 - val_acc: 0.9355
Epoch 69/150
188/188 [==============================] - 0s 3ms/step - loss: 2.1212e-04 - acc: 1.0000 - val_loss: 0.4359 - val_acc: 0.9355
Epoch 70/150
188/188 [==============================] - 0s 3ms/step - loss: 2.0800e-04 - acc: 1.0000 - val_loss: 0.4381 - val_acc: 0.9355
Epoch 71/150
188/188 [==============================] - 0s 3ms/step - loss: 2.0362e-04 - acc: 1.0000 - val_loss: 0.4403 - val_acc: 0.9355
Epoch 72/150
188/188 [==============================] - 0s 3ms/step - loss: 1.9930e-04 - acc: 1.0000 - val_loss: 0.4420 - val_acc: 0.9355
Epoch 73/150
188/188 [==============================] - 0s 3ms/step - loss: 1.9544e-04 - acc: 1.0000 - val_loss: 0.4432 - val_acc: 0.9355
Epoch 74/150
188/188 [==============================] - 0s 3ms/step - loss: 1.9177e-04 - acc: 1.0000 - val_loss: 0.4452 - val_acc: 0.9355
Epoch 75/150
188/188 [==============================] - 0s 3ms/step - loss: 1.8793e-04 - acc: 1.0000 - val_loss: 0.4469 - val_acc: 0.9355
Epoch 76/150
188/188 [==============================] - 0s 3ms/step - loss: 1.8415e-04 - acc: 1.0000 - val_loss: 0.4482 - val_acc: 0.9355

Epoch 77/150
188/188 [==============================] - 0s 3ms/step - loss: 1.8142e-04 - acc: 1.0000 - val_loss: 0.4496 - val_acc: 0.9355
Epoch 78/150
188/188 [==============================] - 1s 3ms/step - loss: 1.7816e-04 - acc: 1.0000 - val_loss: 0.4508 - val_acc: 0.9355
Epoch 79/150
188/188 [==============================] - 1s 3ms/step - loss: 1.7531e-04 - acc: 1.0000 - val_loss: 0.4517 - val_acc: 0.9355
Epoch 80/150
188/188 [==============================] - 1s 3ms/step - loss: 1.7263e-04 - acc: 1.0000 - val_loss: 0.4522 - val_acc: 0.9355
Epoch 81/150
188/188 [==============================] - 1s 3ms/step - loss: 1.7007e-04 - acc: 1.0000 - val_loss: 0.4528 - val_acc: 0.9355
Epoch 82/150
188/188 [==============================] - 1s 3ms/step - loss: 1.6758e-04 - acc: 1.0000 - val_loss: 0.4533 - val_acc: 0.9355
Epoch 83/150
188/188 [==============================] - 1s 3ms/step - loss: 1.6524e-04 - acc: 1.0000 - val_loss: 0.4539 - val_acc: 0.9355
Epoch 84/150
188/188 [==============================] - 1s 3ms/step - loss: 1.6308e-04 - acc: 1.0000 - val_loss: 0.4542 - val_acc: 0.9355
Epoch 85/150
188/188 [==============================] - 1s 3ms/step - loss: 1.6084e-04 - acc: 1.0000 - val_loss: 0.4547 - val_acc: 0.9355
Epoch 86/150
188/188 [==============================] - 1s 3ms/step - loss: 1.5885e-04 - acc: 1.0000 - val_loss: 0.4551 - val_acc: 0.9355
Epoch 87/150
188/188 [==============================] - 1s 3ms/step - loss: 1.5666e-04 - acc: 1.0000 - val_loss: 0.4555 - val_acc: 0.9355
Epoch 88/150
188/188 [==============================] - 1s 3ms/step - loss: 1.5467e-04 - acc: 1.0000 - val_loss: 0.4559 - val_acc: 0.9355
Epoch 89/150
188/188 [==============================] - 1s 3ms/step - loss: 1.5281e-04 - acc: 1.0000 - val_loss: 0.4563 - val_acc: 0.9355
Epoch 90/150
188/188 [==============================] - 0s 3ms/step - loss: 1.5089e-04 - acc: 1.0000 - val_loss: 0.4567 - val_acc: 0.9355
Epoch 91/150
188/188 [==============================] - 0s 3ms/step - loss: 1.4917e-04 - acc: 1.0000 - val_loss: 0.4570 - val_acc: 0.9355
Epoch 92/150
188/188 [==============================] - 0s 3ms/step - loss: 1.4733e-04 - acc: 1.0000 - val_loss: 0.4575 - val_acc: 0.9355
Epoch 93/150
188/188 [==============================] - 0s 3ms/step - loss: 1.4565e-04 - acc: 1.0000 - val_loss: 0.4578 - val_acc: 0.9355
Epoch 94/150
188/188 [==============================] - 0s 3ms/step - loss: 1.4393e-04 - acc: 1.0000 - val_loss: 0.4581 - val_acc: 0.9355
Epoch 95/150
188/188 [==============================] - 0s 3ms/step - loss: 1.4232e-04 - acc: 1.0000 - val_loss: 0.4585 - val_acc: 0.9355
Epoch 96/150
188/188 [==============================] - 0s 3ms/step - loss: 1.4073e-04 - acc: 1.0000 - val_loss: 0.4589 - val_acc: 0.9355
Epoch 97/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3917e-04 - acc: 1.0000 - val_loss: 0.4592 - val_acc: 0.9355
Epoch 98/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3760e-04 - acc: 1.0000 - val_loss: 0.4596 - val_acc: 0.9355
Epoch 99/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3617e-04 - acc: 1.0000 - val_loss: 0.4600 - val_acc: 0.9355
Epoch 100/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3479e-04 - acc: 1.0000 - val_loss: 0.4603 - val_acc: 0.9355
Epoch 101/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3333e-04 - acc: 1.0000 - val_loss: 0.4607 - val_acc: 0.9355
Epoch 102/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3196e-04 - acc: 1.0000 - val_loss: 0.4611 - val_acc: 0.9355

Epoch 103/150
188/188 [==============================] - 0s 3ms/step - loss: 1.3065e-04 - acc: 1.0000 - val_loss: 0.4615 - val_acc: 0.9355
Epoch 104/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2931e-04 - acc: 1.0000 - val_loss: 0.4619 - val_acc: 0.9355
Epoch 105/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2806e-04 - acc: 1.0000 - val_loss: 0.4623 - val_acc: 0.9355
Epoch 106/150
188/188 [==============================] - 1s 3ms/step - loss: 1.2678e-04 - acc: 1.0000 - val_loss: 0.4627 - val_acc: 0.9355
Epoch 107/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2558e-04 - acc: 1.0000 - val_loss: 0.4631 - val_acc: 0.9355
Epoch 108/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2437e-04 - acc: 1.0000 - val_loss: 0.4635 - val_acc: 0.9355
Epoch 109/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2320e-04 - acc: 1.0000 - val_loss: 0.4638 - val_acc: 0.9355
Epoch 110/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2200e-04 - acc: 1.0000 - val_loss: 0.4642 - val_acc: 0.9355
Epoch 111/150
188/188 [==============================] - 0s 3ms/step - loss: 1.2089e-04 - acc: 1.0000 - val_loss: 0.4647 - val_acc: 0.9355
Epoch 112/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1980e-04 - acc: 1.0000 - val_loss: 0.4649 - val_acc: 0.9355
Epoch 113/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1867e-04 - acc: 1.0000 - val_loss: 0.4654 - val_acc: 0.9355
Epoch 114/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1768e-04 - acc: 1.0000 - val_loss: 0.4658 - val_acc: 0.9355
Epoch 115/150
188/188 [==============================] - 1s 3ms/step - loss: 1.1657e-04 - acc: 1.0000 - val_loss: 0.4662 - val_acc: 0.9355
Epoch 116/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1553e-04 - acc: 1.0000 - val_loss: 0.4666 - val_acc: 0.9355
Epoch 117/150
188/188 [==============================] - 1s 3ms/step - loss: 1.1450e-04 - acc: 1.0000 - val_loss: 0.4670 - val_acc: 0.9355
Epoch 118/150
188/188 [==============================] - 1s 3ms/step - loss: 1.1355e-04 - acc: 1.0000 - val_loss: 0.4674 - val_acc: 0.9355
Epoch 119/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1261e-04 - acc: 1.0000 - val_loss: 0.4677 - val_acc: 0.9355
Epoch 120/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1164e-04 - acc: 1.0000 - val_loss: 0.4682 - val_acc: 0.9355
Epoch 121/150
188/188 [==============================] - 0s 3ms/step - loss: 1.1074e-04 - acc: 1.0000 - val_loss: 0.4685 - val_acc: 0.9355
Epoch 122/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0979e-04 - acc: 1.0000 - val_loss: 0.4688 - val_acc: 0.9355
Epoch 123/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0884e-04 - acc: 1.0000 - val_loss: 0.4693 - val_acc: 0.9355
Epoch 124/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0797e-04 - acc: 1.0000 - val_loss: 0.4697 - val_acc: 0.9355
Epoch 125/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0710e-04 - acc: 1.0000 - val_loss: 0.4701 - val_acc: 0.9355
Epoch 126/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0626e-04 - acc: 1.0000 - val_loss: 0.4705 - val_acc: 0.9355
Epoch 127/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0539e-04 - acc: 1.0000 - val_loss: 0.4709 - val_acc: 0.9355
Epoch 128/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0457e-04 - acc: 1.0000 - val_loss: 0.4713 - val_acc: 0.9355

Epoch 129/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0377e-04 - acc: 1.0000 - val_loss: 0.4717 - val_acc: 0.9355
Epoch 130/150
188/188 [==============================] - 1s 3ms/step - loss: 1.0298e-04 - acc: 1.0000 - val_loss: 0.4722 - val_acc: 0.9355
Epoch 131/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0213e-04 - acc: 1.0000 - val_loss: 0.4726 - val_acc: 0.9355
Epoch 132/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0137e-04 - acc: 1.0000 - val_loss: 0.4730 - val_acc: 0.9355
Epoch 133/150
188/188 [==============================] - 0s 3ms/step - loss: 1.0065e-04 - acc: 1.0000 - val_loss: 0.4734 - val_acc: 0.9355
Epoch 134/150
188/188 [==============================] - 1s 3ms/step - loss: 9.9875e-05 - acc: 1.0000 - val_loss: 0.4737 - val_acc: 0.9355
Epoch 135/150
188/188 [==============================] - 1s 3ms/step - loss: 9.9100e-05 - acc: 1.0000 - val_loss: 0.4742 - val_acc: 0.9355
Epoch 136/150
188/188 [==============================] - 1s 3ms/step - loss: 9.8402e-05 - acc: 1.0000 - val_loss: 0.4745 - val_acc: 0.9355
Epoch 137/150
188/188 [==============================] - 0s 3ms/step - loss: 9.7658e-05 - acc: 1.0000 - val_loss: 0.4749 - val_acc: 0.9355
Epoch 138/150
188/188 [==============================] - 0s 3ms/step - loss: 9.6912e-05 - acc: 1.0000 - val_loss: 0.4754 - val_acc: 0.9355
Epoch 139/150
188/188 [==============================] - 0s 3ms/step - loss: 9.6236e-05 - acc: 1.0000 - val_loss: 0.4757 - val_acc: 0.9355
Epoch 140/150
188/188 [==============================] - 1s 3ms/step - loss: 9.5529e-05 - acc: 1.0000 - val_loss: 0.4762 - val_acc: 0.9355
Epoch 141/150
188/188 [==============================] - 0s 3ms/step - loss: 9.4849e-05 - acc: 1.0000 - val_loss: 0.4765 - val_acc: 0.9355
Epoch 142/150
188/188 [==============================] - 0s 3ms/step - loss: 9.4176e-05 - acc: 1.0000 - val_loss: 0.4769 - val_acc: 0.9355
Epoch 143/150
188/188 [==============================] - 0s 3ms/step - loss: 9.3511e-05 - acc: 1.0000 - val_loss: 0.4772 - val_acc: 0.9355
Epoch 144/150
188/188 [==============================] - 0s 3ms/step - loss: 9.2823e-05 - acc: 1.0000 - val_loss: 0.4776 - val_acc: 0.9355
Epoch 145/150
188/188 [==============================] - 0s 3ms/step - loss: 9.2175e-05 - acc: 1.0000 - val_loss: 0.4780 - val_acc: 0.9355
Epoch 146/150
188/188 [==============================] - 0s 3ms/step - loss: 9.1535e-05 - acc: 1.0000 - val_loss: 0.4785 - val_acc: 0.9355
Epoch 147/150
188/188 [==============================] - 0s 3ms/step - loss: 9.0938e-05 - acc: 1.0000 - val_loss: 0.4788 - val_acc: 0.9355
Epoch 148/150
188/188 [==============================] - 0s 3ms/step - loss: 9.0301e-05 - acc: 1.0000 - val_loss: 0.4792 - val_acc: 0.9355
Epoch 149/150
188/188 [==============================] - 0s 3ms/step - loss: 8.9686e-05 - acc: 1.0000 - val_loss: 0.4796 - val_acc: 0.9355
Epoch 150/150
188/188 [==============================] - 0s 3ms/step - loss: 8.9097e-05 - acc: 1.0000 - val_loss: 0.4800 - val_acc: 0.9355

Accuracy Achieved towards epoch 150 is 93.55 %