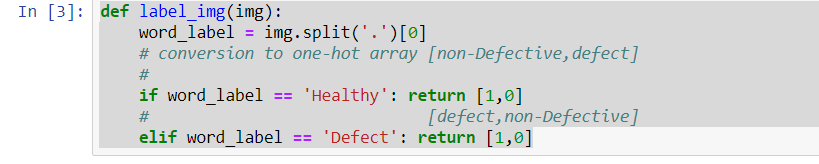
Model for JBM

Step1: Splitting Images into Different Folders so as to pre-process the Data

Directory 1 – Healthy Training Images and Test Images

Directory 2 – Defect Training Images and Test Images

Step2: Labelling of Images into Healthy and Non – Healthy Images by using the Below Function

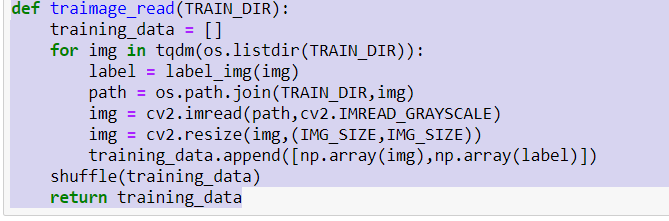


Step3: Transformed the images by using below techniques to make it ready for fitting into Convolution Neural Network.

1) Greyscale - So that less memory be used

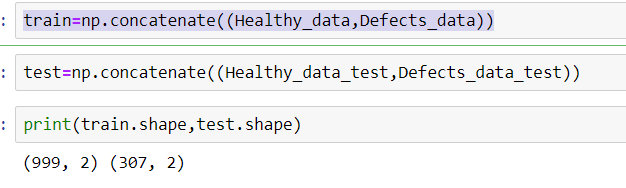
2)Resized the images to 64 \* 64 Size

3)Convert images to Numpy Array so as to make it fit for Training

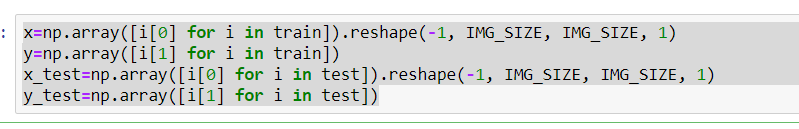


Step 4:

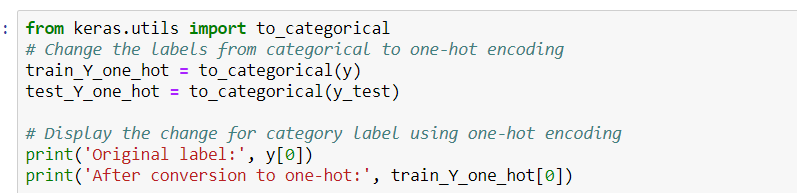
Divide the Data into train and Test to make the model Generalize the model well to make accurate Predictions.



Step5: Reshaping the Array to Make it fit for the model.



Step6: Used one hot Encoding to make it fit for the model



Step7: Classifier Used – Keras



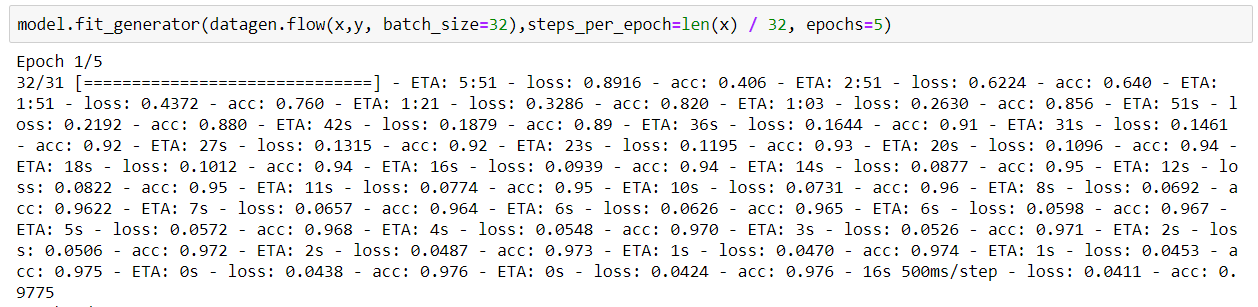
Here we are using 3 convolution layers with single stride, zero padding and relu activation (you can even try changing the activation functions and see how the model behaves). If you notice for each layer the filter count is increasing, because the initial layer represents the high level features of the image and the deeper layers will represent more detailed features and so they usually have more number of filters. And the filter count is an arbitrary number and we can play with it and see the model behaviour.

After three convolution layers we have one dropout layer and this is to avoid overfitting problem. And once the image passes through the convolution layers it has to be flattened again to be fed into fully connected layers (it’s called a dense layer in keras, here all the neurons in first layer is connected to all the neurons in the second layer.

We have 2 dense layers and the first one is having 512 neurons and relu activation, this is also arbitrary and we can have the neuron count as per our choice. the second dense layer will have only 2 neurons as we have only two classes to classify, usually the number of neurons in the output layer will be equal to number of classes in our problem. This layer will use softmax activation. softmax activation will calculate the probabilities of each target class over all possible target classes and the sum of all the probabilities will always be 1. The input will be classified into any of the target class based on the higher probability value in softmax.

We are using Adam optimizer with “categorical\_crossentropy” as loss function and learning rate of 0.001. We train our model for 5 epochs (for every epoch the model will adjust its parameter value to minimize the loss) and the accuracy we got here is around 97%.

Step7: Used Keras for Tensor Flow Library to Fit the model we got an accuracy of 97 % accuracy



Step:8As the model was not Able to generalize well and was Overfitting we used Data Augmentation and add more layers to our Classifier to Achieve better and Accurate Predictions

