About code file

The HW3_[redacted].py is main file. Executing this file gives the outputs for all the questions. All the functions are stated below along with their functionality.

Read_images: this function read all the data.
Onehot: this function converts the labels into one hot vectors.
Preprocess: this function resize, rescale, check channels and remove noise if necessary.
Augmentation: this function augments the data by flipping all the images.
Weight_vector: this function generates weight tensor of a shape required for a layer.
Bias_vector: this function generates Bias tensor of a shape required for a layer.
Layer_convolution creates a convolutional layer, number of inputs, channels, size of filters, number of filters and to use pooling or not is required for tgis function.
Layer_flatten: flattens the output from convolutional layer
Dense layer: create a dense layer of input and output connection should be passed.
Makelayer: creates the whole Network architecture
Batch: creates batch training data
TestBatch: randomly takes a batch of test data for validation.
Optimize: Optimize function is heart of the program which trains the network, Data, number of iterations and batch size and whether regularization should be done or is to be passed to this function.
Weights_plot: It plots the histograms of the weights of a layer.
Accuracy_plot: It plots the training and validation accuracy, calculated in every epoch.
Loss_plot: It plots the training and validation loss, calculated in every epoch.
Plot_conv_layer: This function plots outputs of the convolution layers.
Question1: read and preprocess the data
Question2: it trains the Network
Question3: plots the accuracy, loss and weights in layers.
Question4: it plots all the images.
Question5: it trains the network with Regularization
Question6: it applies data augmentation and then train a network.

Assignment Answers

Question 1.
The preprocessing file is saved in data prepo.py. In this file I first read the images and then resize all the images to a specific size, so that all the images have same resolution. More ever I also converted all one channel images to three channel images. So that that grayscale image act like RGB. The reason we do rescaling of images is that our CNN model accepts a fixed size image. If images in a has different shapes then size of each layer also needs to change dynamically. this is almost impossible to handle. So, for a static network we need fixed size images. That’s why I have resized images to a fixed size resolution and also checked for channels of images and changed it to 3, if it is less than 3.

Question2:

I tried several experiments with different parameters. Some experiments are given below.

Experiment 1:

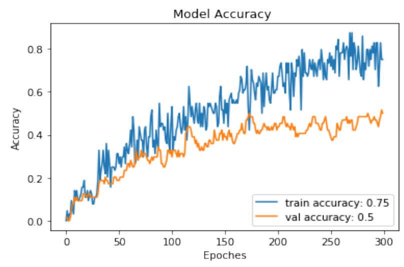
In this experiment we achieved highest training accuracy of 0.91 at epoch 229. At the same epoch the validation accuracy was almost 0.40.

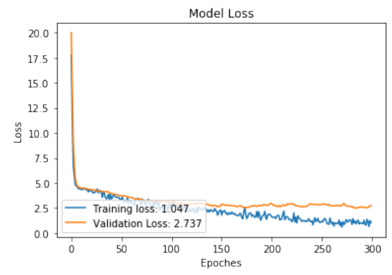
Input image size: 128 * 128

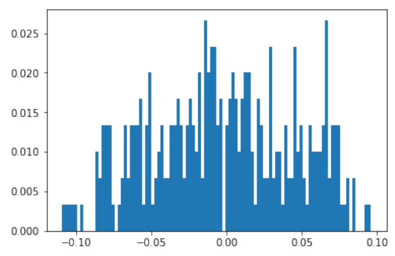
Convolutional layers: 2, number of filters in first layer: 4, number of filters in second layers: 4

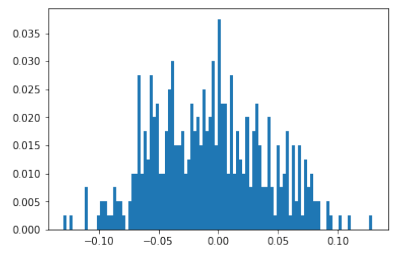
Dense layers: 3, first layer neurons: 4096, second layer neurons: 256 third layer neurons: 256

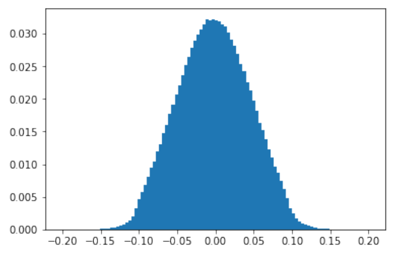
Epochs:300

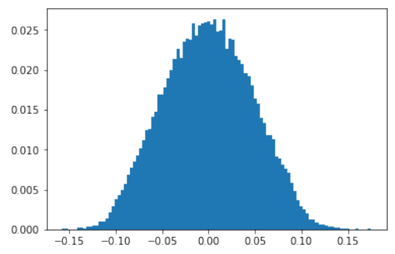


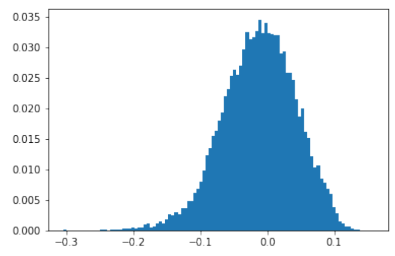












Experiments 2:

All the parameters are same as in experiment 1 except that it was trained for 600 epochs. The model starts overfitting. Accuracy dropped from 0.5 to 0.4 and validation loss increased from 2.7 to 4.48. We can clearly see the model overfits after 250 epochs from the accuracy and loss plots.

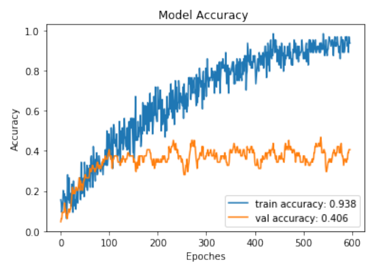
Model:

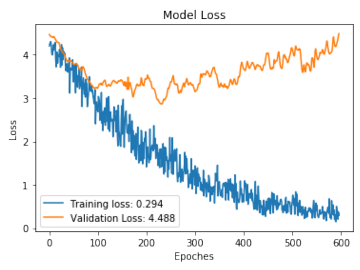
Input image size: 128 * 128

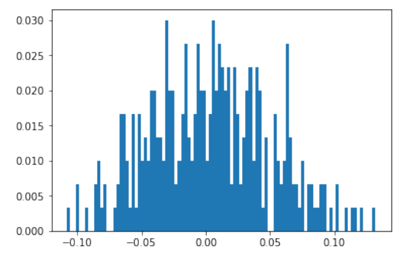
Convolutional layers: 2, number of filters in first layer: 16, number of filters in 2nd layers: 16

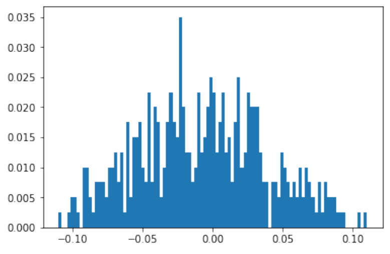
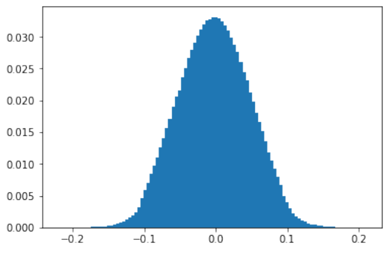
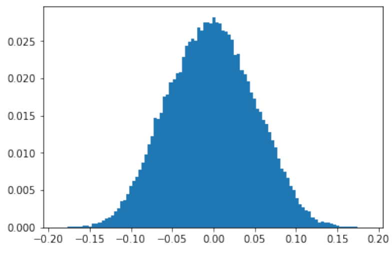
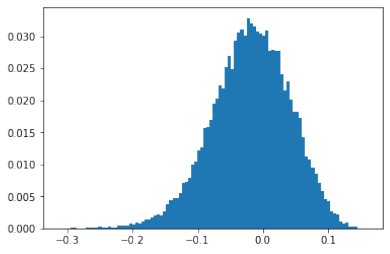
Dense layers: 3, first layer neurons: 4096, second layer neurons: 256 third layer neurons: 256

Epochs:300







Experiment 3:

In this experiment we changed the model a little bit and increased number of filters to 16 in each layer. The rest of the network was same as in Experiment 1. The model hyper parameters are shown below. Best training accuracy recorded was 87.4 and validation accuracy at that epoch was 29.2. The model started overfitting after 150 epochs. The validation loss started increasing after that.

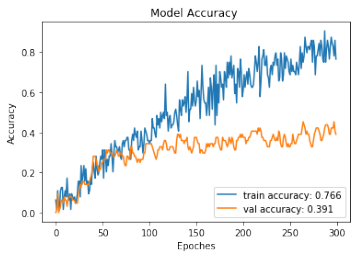
Model:

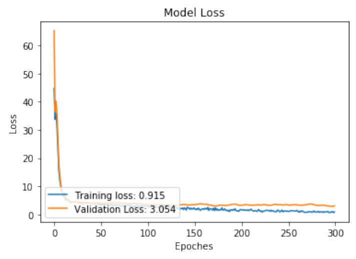
Input image size: 128 * 128

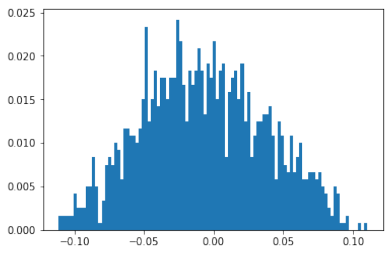
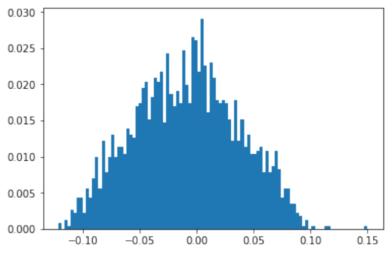
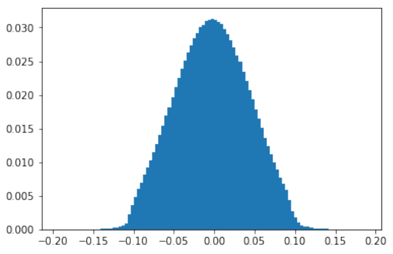
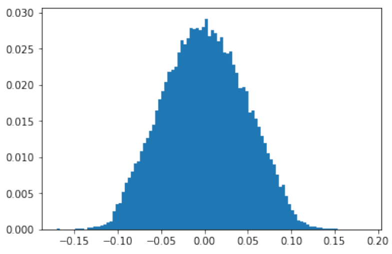
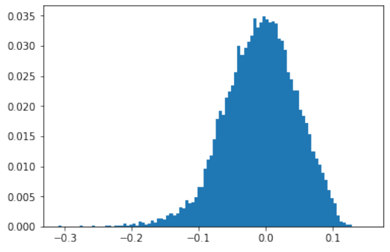
Convolutional layers: 2, number of filters in first layer: 16, number of filters in 2nd layers: 16

Dense layers: 3, first layer neurons: 4096, second layer neurons: 256 third layer neurons: 256

Epochs:300

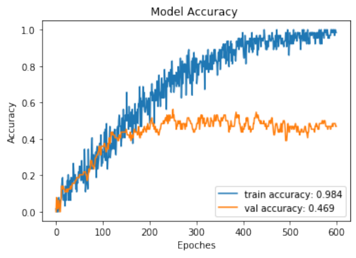


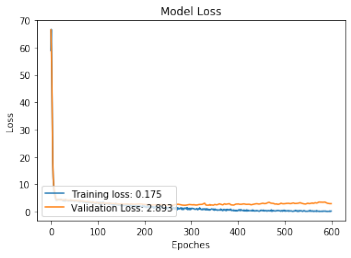


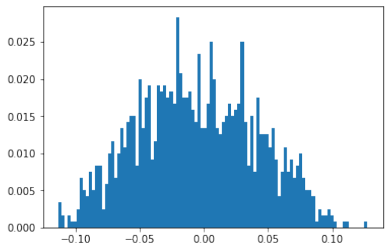






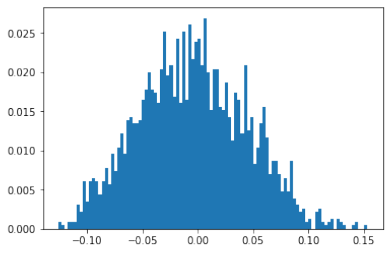
Experiment 4:

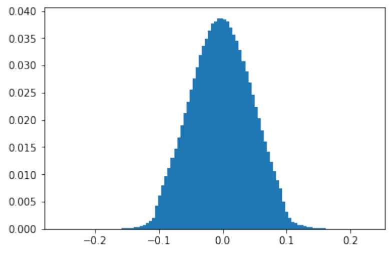
In this experiment the strides were changed to [2,2] and the model was trained for 600 epochs. Rest of the parameters were same as experiment 3. Training for more Increased the training accuracy to 98% but test accuracy starts decreasing after 400 epochs. The validation loss starts increasing and it seems the model is overfitting.

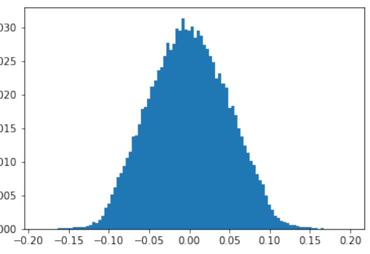


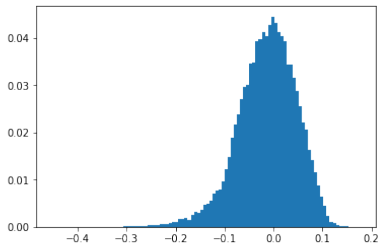












Question 3:

In this question we are asked to plot accuracy, loss and also plot the weights. Although I Have plotted these things in question 2. But Here I will present the best performance which we achieved is shown in figure below. The model parameters are shown below

Model:

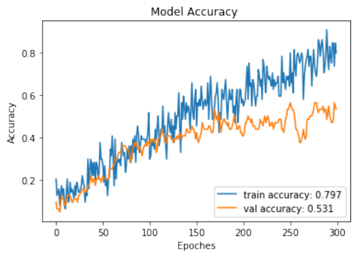
Input image size: 128 * 128, learning rate=0.01

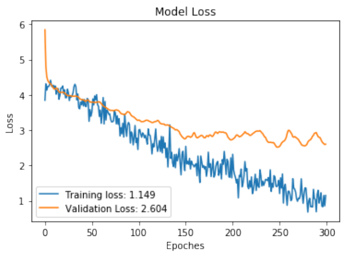
Convolutional layers: 2, number of filters in first layer: 16, number of filters in second layers: 16, stride in convolution= [1,1].

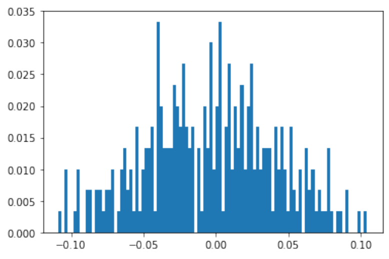
Dense layers: 3, first layer neurons: 4096, second layer neurons: 256 third layer neurons: 256

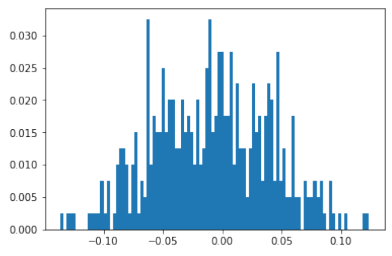
Epochs:300

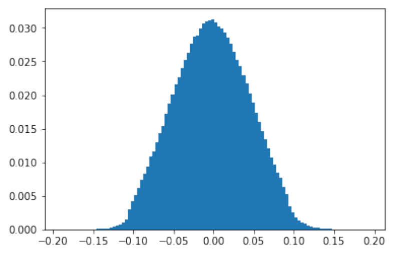
As we have huge fluctuation so I traced back and found the best train and validation accuracy. It was found that Train accuracy was 79.3 and validation accuracy is 53.2. This accuracy was achieved at epoch 245 shown in plot below. As when we look at loss curve the model seems to start overfitting after 240 epochs. We have somewhat removed this overfitting by regularization explained in question 5.

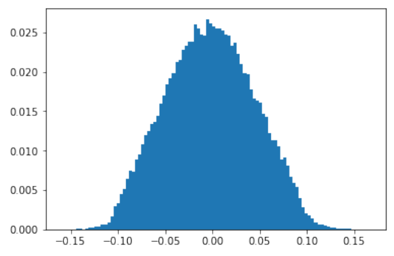






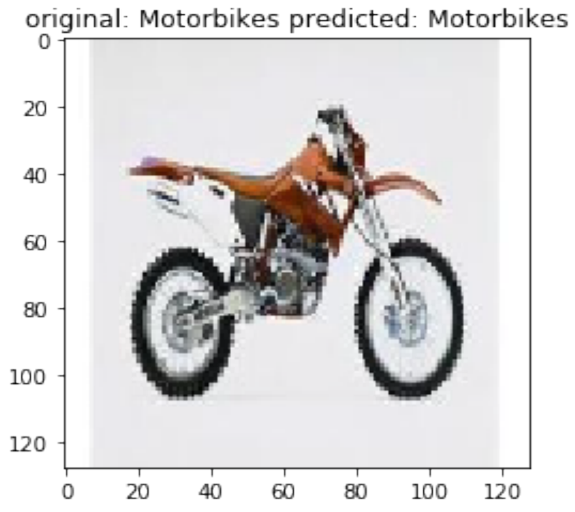




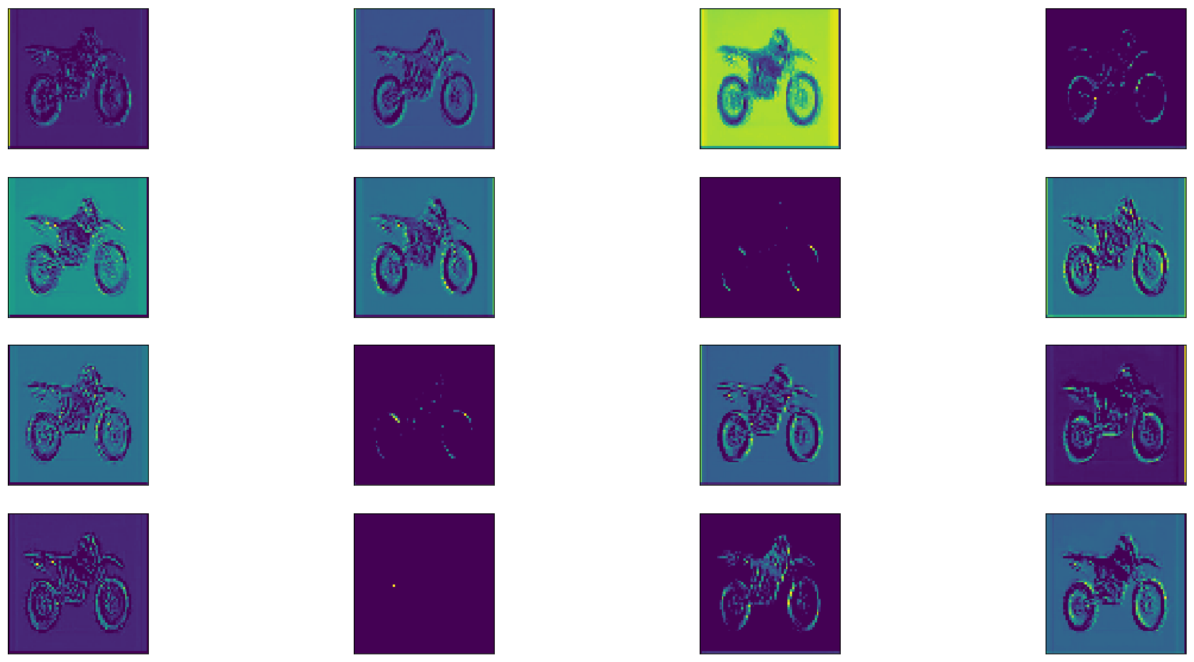


Question 4:

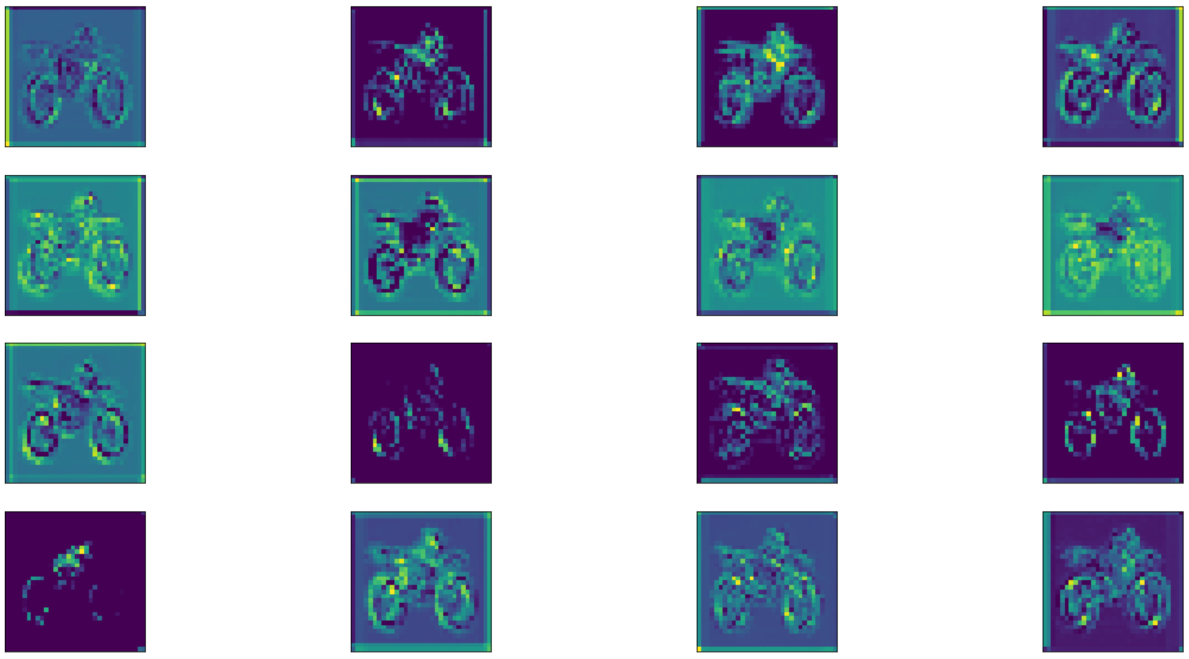
In Image of motorbike is shown below, the network correctly predicted the class Motorbike.



In first convolution layer we used 16 filters. The output of these of all these convolutions is shown below. As in initial layers CNN learn simple features like lines, circles etc. In this layer CNN has learned the edges of the object. We can see from the image that all the filters have extracted some features. Each image shows a separate feature map representing motorbike.

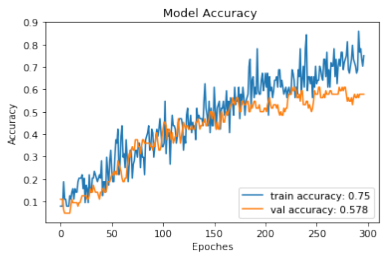


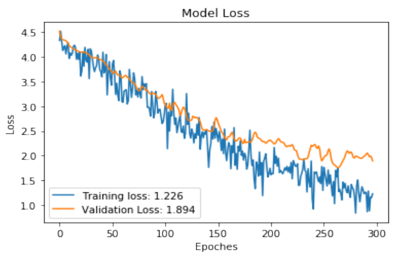
In Second convolution layer we again used 16 filters. The output is shown below. We can see the difference between the features map learned in first layer and feature map extracted in second layer. The second layer features seem more complex than earlier layer. As we go more into deep network the CNN will learn more complex features which will beyond human understanding.

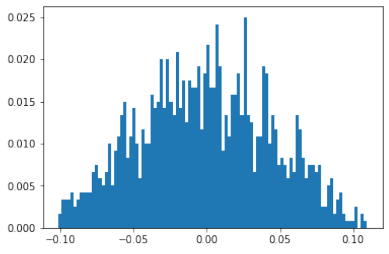


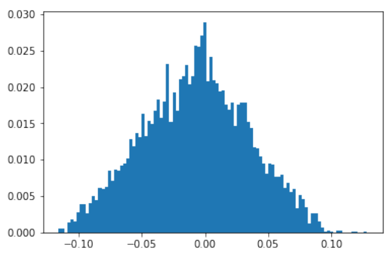
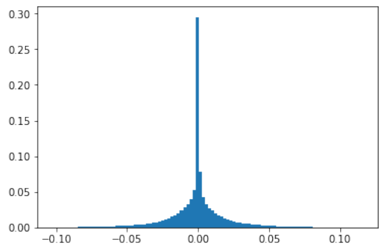
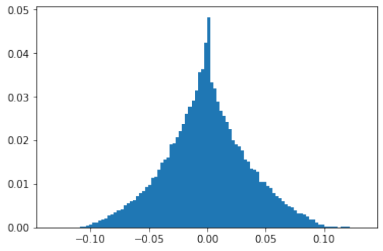
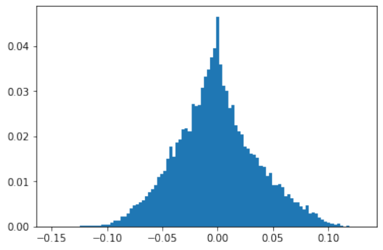
Question5. The experiments in Question 3 was repeated again with regularization. The following results were obtained.

The model was same as shown below. The only change is that we used regularization in this. The best training accuracy and testing accuracy increased to 82.3 from 79.3 and validation accuracy was increased to 59.1 from 53.2. With regularization we see from Loss plot that the training and validation also decreased. The curve now of both Train and Validation seems to be decreasing now. We see that regularization has somewhat benefited the overall a model and removed some overfitting also.







Question 4: Model overfitting can be solved by many ways. Three of which is discussed below.

- Data augmentation: data augmentation is creating more data from the existing available data. From the already present instances in in data new data instances are created with some modifications. If the data are images, data can be augmented by translation, rotation, flipping, zooming, right shift, left shift etc.
- Batch Normalization: It is used to normalize convolutional layers and dense layers output. Batch normalization decreases the network dependency on weight initialization. As means and variances are calculated from the samples in mini batch, it adds some regularization to the model. Batch normalization helps in overcoming the dead ReLU problem by shifting and scaling the standard normal distribution accordingly. Batch normalization increases the test accuracy and helps in overcoming the overfitting problem

- Dropout: it's a method that trains a large number of neural networks with different architectures in parallel. In each architecture some number of layer or nodes or switched off which call dropped out. These nodes and its connections are temporarily removed from the network. Dropout acts like adding a noise, which enforce nodes take on more or less responsibility for the inputs in accordance with a probability.

In the code I implemented Data augmentation and dropout methods. For data augmentation I carried four operations rotation by 90, rotation by 180, translating image and flipping the image. Out of these four operations, I applied one operation to each image and duplicated it in data set. After that the same Training process was carried out on our best model. The plots below show accuracy and loss.

I have also added dropout of 0.2% to fully connected layers.

Accuracies and loss summary:

The overall accuracy summary of assignment is given here. We have taken best model on basis of test accuracy.

	Train Accuracy%	Test Accuracy%	Train loss	Test loss
Experiment 1 (No regularization)	75	5	1.04	2.73
Experiment 2 (No regularization)	93.8	40.6	0.294	4.48
Experiment 3 (No regularization)	76.6	39.1	0.915	3.054
Experiment 4 (No regularization)	98.4	46.9	0.175	2.893
Best Model (no regularization)	79.7	53.0	1.149	2.604
Best Model (with regularization)	75.0	57.8	1.226	1.894
Best Model (Data Augmentation+ Dropout)	0.81	59.8	1.01	1.69