All the graphs can be seen in the jupyter notebook file

Here batch gradient descent has been used

SHUFFLING:

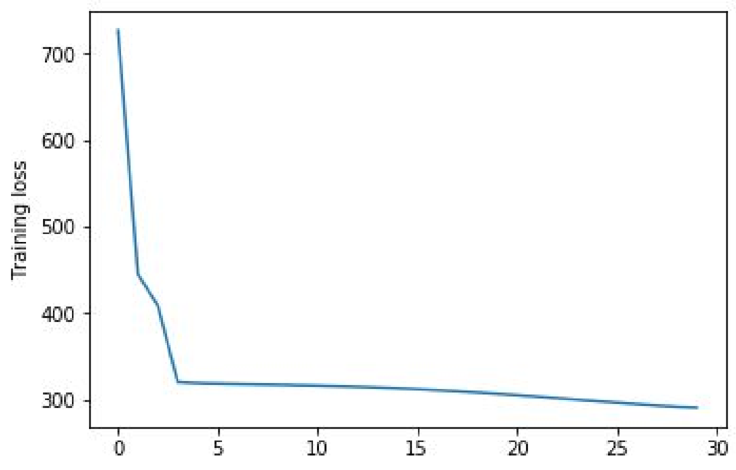
The Code 'CNN_1layer' contains the code without shuffling of data.

The Code 'CNN_1layer_Shuffle' contains the code with shuffling of data.

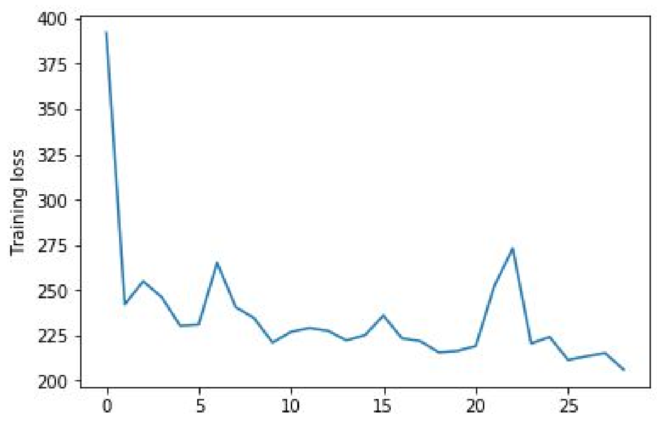
It can be clearly said from the graph of training loss that shuffling increases better adaptability of the model and avoids overfitting.

TRAINING LOSS GRAPH FOR SAME ARCHITECTURE BUT WITH SHUFFLE AND WITHOUT SHUFFLE

Without shuffling reduces smoothly indicating overfitting

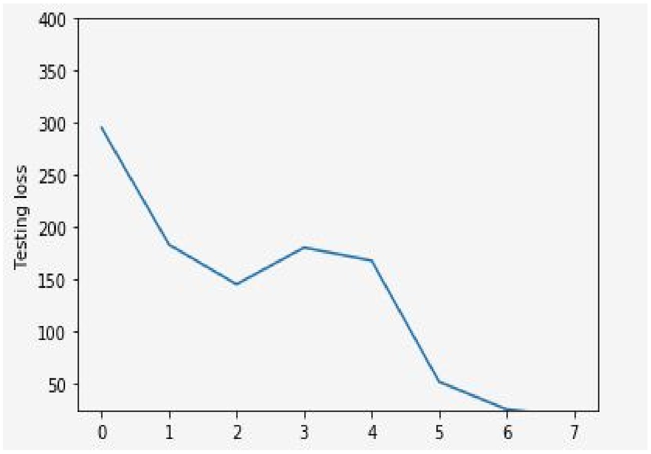


With shuffling reduces over a period of timeindicating better model

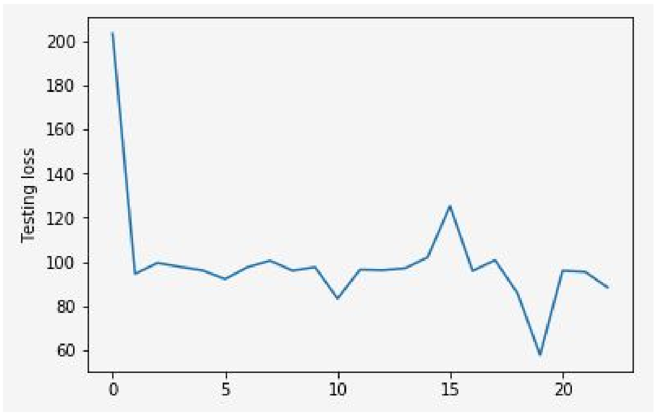


Test loss graph for 2 different architectures(1CNN LAYER ; 2 CNN LAYER)

CNN 2 LAYER



CNN 1 LAYER



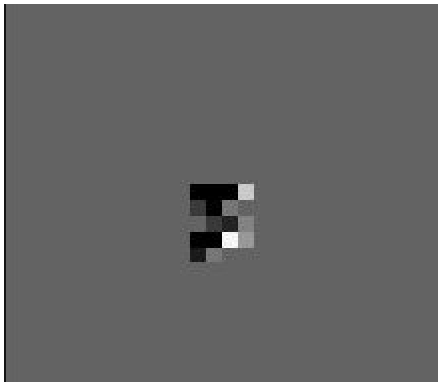
Accuracy: (BEST ACCURACY ON TEST DATA 0.2)

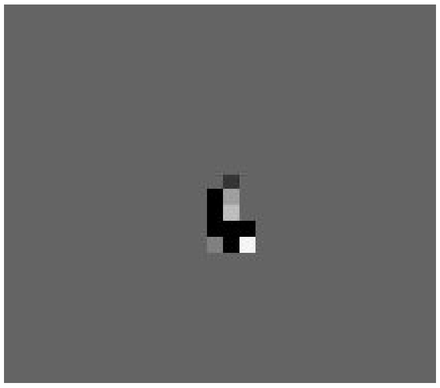
POINT TO UNDERSTAND THE ACCURACY IS LOW DUE TO SMALL ARCHITECTURE IE ONLY ONE CNN LAYER AND LESS DATA GIVEN TO TRAIN DUE COMPUTATIONAL ISSUES ON MY LAPTOP

EVEN WITH 2 CNN LAYER THE FILTERS WERE LOW IN NUMBER AND TRAINED WITH LESS TRAIN DATA

Accuracy has been computed for the model at the end only once because computing accuracy at the end of each epoch is consuming a lot of time and also accuracy at the end of the complete training has been reported for 2 different architectures; also with shuffling and without shuffling with in same architecture

Some of the **activation maps** has been given below





More activation maps have been given in jupyter notebook