COL341 Assignment-2 PART 1

Neural Network Implementation

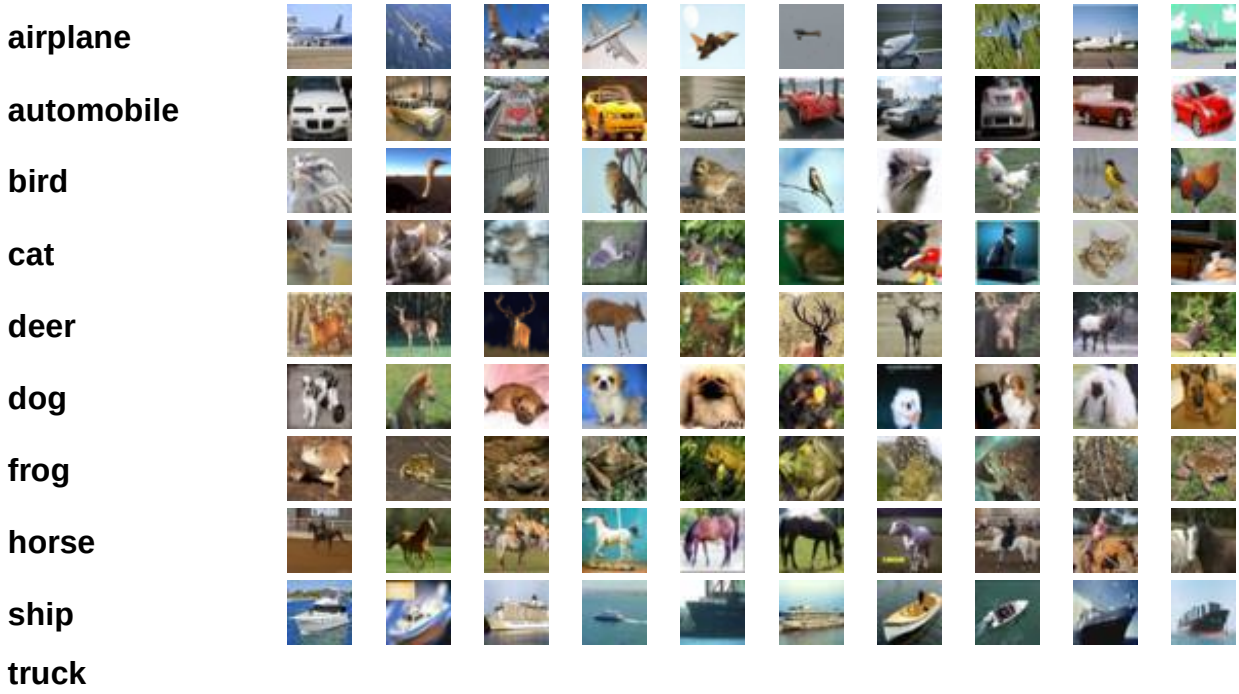
Name- Vikram Kumar **Entry No. 2016ce10237**

Part C.)

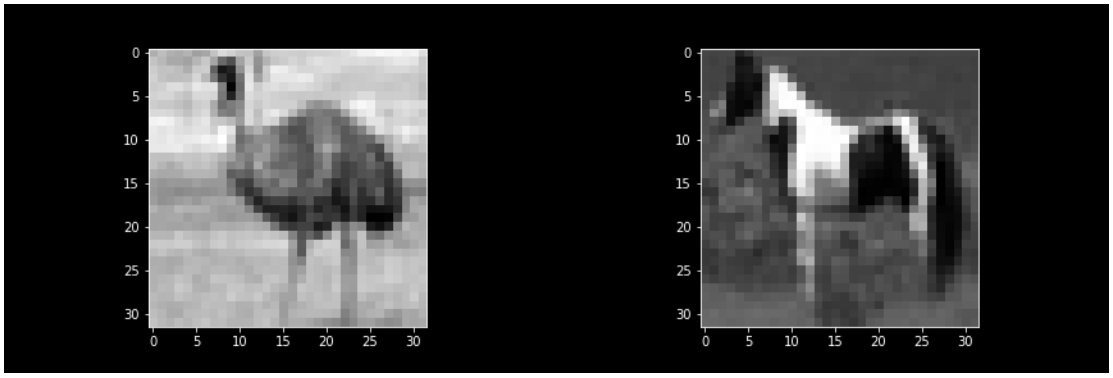
Implemented multi layer neural network class which can take inputs for Neural Network Architecture such as:

- Options for number of hidden layers with their number of neurons
- Options for Activation functions for hidden layers such as ReLu, Sigmoid and tanh
- Learning rate and Number of iterations and batch size
- Adaptive Gradient descent or Fixed Gradient Descent
- Options for number of output neurons according to classification task
- Options for Batch Normalization and Dropout for regularization, keep it or can avoid it

Classification on CIFAR 10 dataset



Cifar 10 dataset contains 10 classes which are these, we are using only 1024 features(gray scale image of 32x32) for single image, for examples:-



Bird Horse

First I created a simple neural network with:-

- One hidden layer of 517 neurons (choice of number of neurons is based on formula which is number of neurons should be (input_neurons +output_neurons)/2)
- Choosed learning rate 0.001 and batch size 100 and number of iteration was 5000
- Activation function for hidden layer was relu and output activation was softmax
- Initialized weights with random uniform distribution between -1 to 1

For this architecture test accuracy was around 24%

For the same parameters except initialization of weights now changed, now weight initialization was done by random uniform distribution **between -1 to 1** and multiplied it with the factor

$$\sqrt{(2/(\boldsymbol{Number\ of\ input\ neurons})}$$

Which increases test accuracy upto **35.96%**

Increasing number of epochs from **25 epochs to 50 epochs** increases test accuracy upto **38%**.

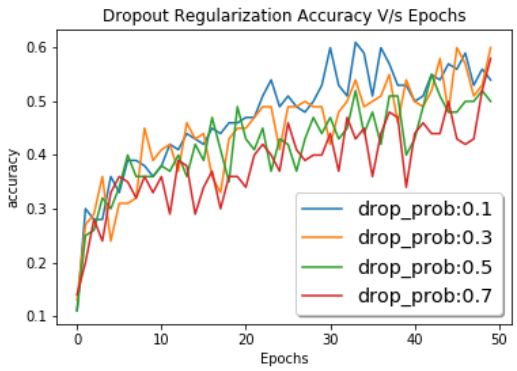
Other architecture which I used,(all of these networks uses 1 hidden layer), for all of these weight initialization was same as mentioned above

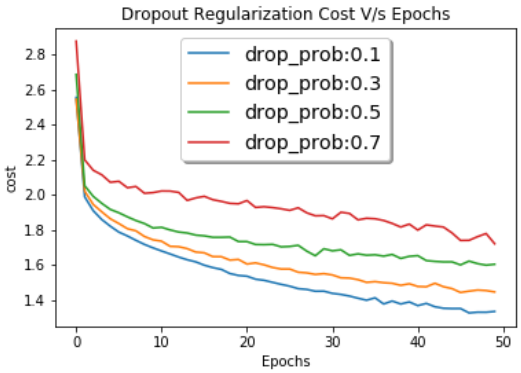
Activation function of hidden layer	Neurons in hidden layers	Learning rate	Batch size	Drop Prob	Regularization parameter	Train Accuracy(%)	Test Accuracy(%)	epochs
tanh	517	0.01	100	0.5	0	38.1	35.72	50
Relu	517	0.01	100	0.5	0	67.20	41.18	50
Sigmoid	517	0.01	100	0.5	0	35.874	33.56	50
Leaky relu	517	0.01	100	0.5	0	67.915	41.6	50

Dropout Regularization:-

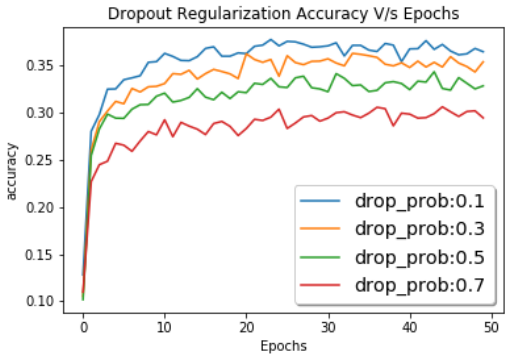
Performed dropout regularization for **ReLu Activation** on hidden layers for 4 different drop probabilities **0.1,0.3,0.5 and 0.7**, as I train the model for only 8000 iteration (50 epochs) and it can be seen that still training cost is reducing significantly which can be seen by increasing number of iteration which leads to highest train accuracy obtained upto **72.58%** and test accuracy **42.95% for** 12000 iterations and **drop probability 0.5**, dropout does not actually regularize very well because there is significant gap between train and test accuracy which actually represent overfitting of the model

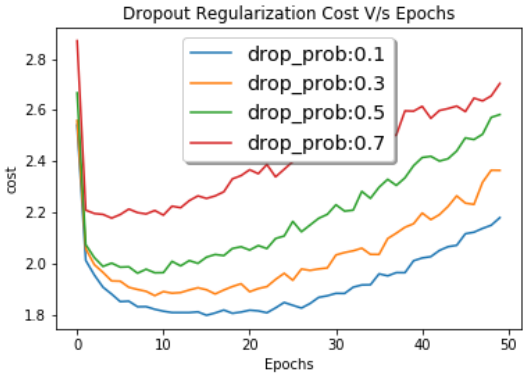
Training Set





Validation Set



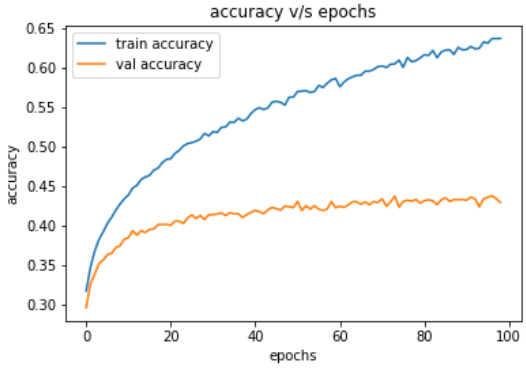


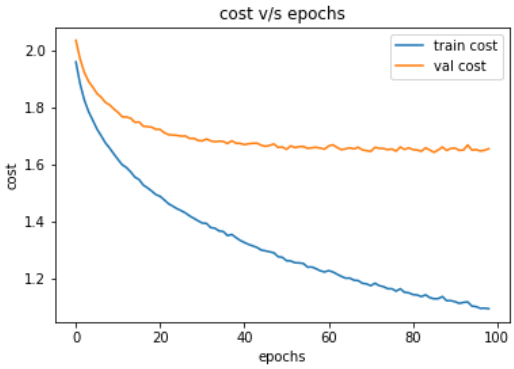
Best Architecture :-

Till now best architecture is getting test accuracy around **42%**

Weight initialization as described above, activation function is leaky-relu, drop_prob=0.5, learning_rate=0.01, batchsize=100, regularization_para = 1.5, one hidden layer with 517 neurons and epochs was 100

Plots for best architecture for accuracy and cost with respect to No of epochs





Confusion matrix :-

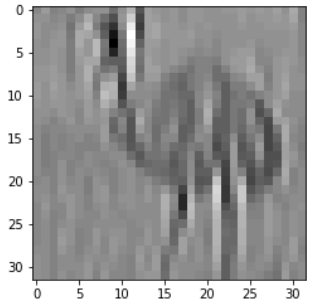
	precesion	recall	F1_scores
0	0.6264	0.6193	0.62
1	0.809	0.7057	0.75
2	0.5143	0.4276	0.47
3	0.5838	0.4898	0.53
4	0.4535	0.5922	0.51
5	0.6487	0.5931	0.62
6	0.6046	0.7014	0.65
7	0.7893	0.7097	0.75
8	0.6551	0.7153	0.68
9	0.7079	0.7792	0.74

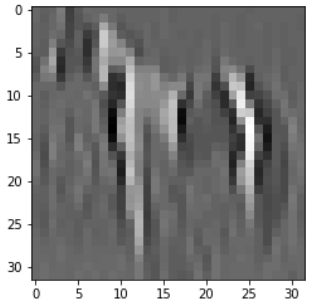
Part D) Feature Generation

Gained test accuracy around – 41%

Gabor filters:-

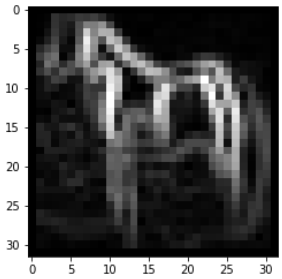
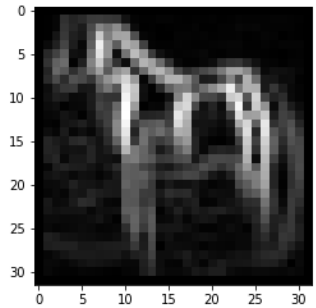
Gabor filters are **orientation-sensitive** filters, used for texture analysis. A gabor filter set with a given direction gives a strong response for locations of the target images that have structures in this given direction. These are some filtered images which basically identified the textured of the image and highlight that texture.

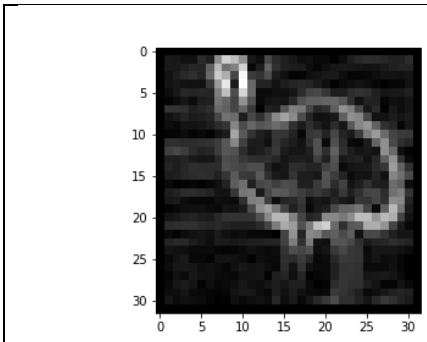


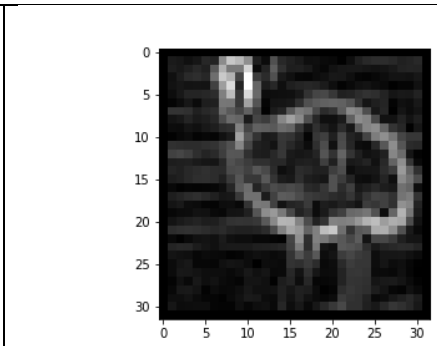


Sobel and Prewitt Filters:-

These are basically edge detector filters, the results can be seen here

Sobel	Prewitt
	





HOG(Histogram of Oriented Gradients):-

The technique counts occurrences of gradient orientation in localized portions of an image, it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

Pixels per cell-> (2,2)	**Pixels per cell->(8,8)**