DEEP LEARNING

CS7015

Programming Assignment : 2

Convolutional Neural Networks

Shweta Bhardwaj
CS16S003

Shikha Singh
CS16D008

March 25, 2017

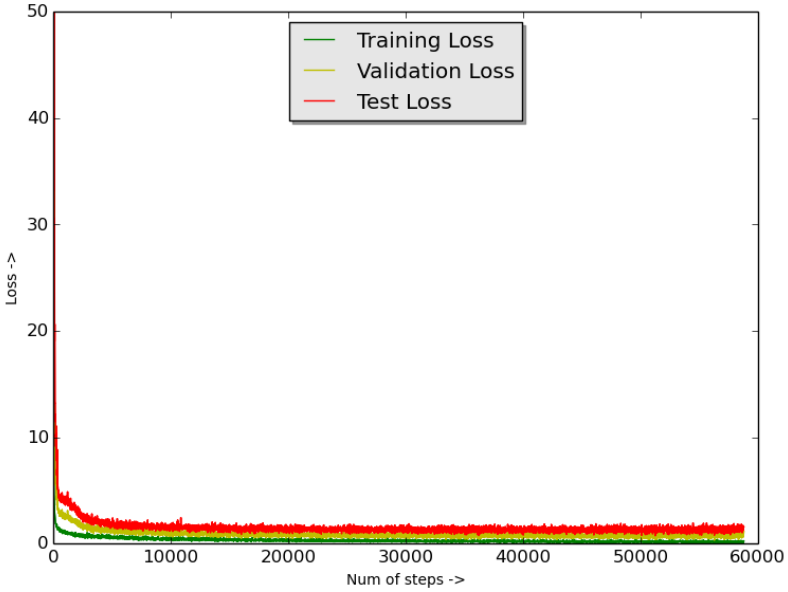
SET-UP :

- : DataSet - CIFAR10
- : Network - As given in Problem
- LOSS : Cross-entropy
- Optimiser : Adam
- Experimented with Initialisations : [Xavier, He]
- Experimented with learning rate : [0.001,0.01,0.1]
- Experimented with Batch size :
- Batch Normalisation : [with/ without]
- Early Stopping with patience parameter 5

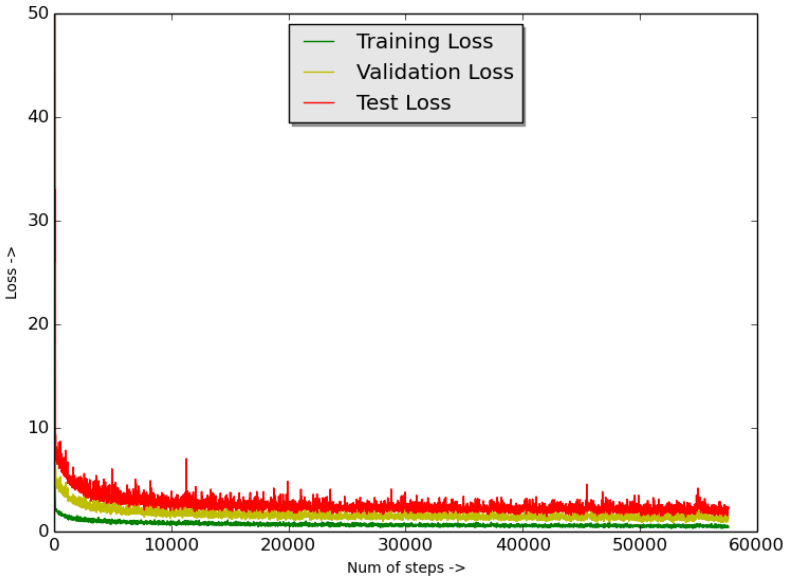
Question : 1

PLOTS : LOSS

Performance of CNN in terms of loss $\{alpha$:0.001$\}$ with Batch Normalisation



Performance of CNN in terms of loss $\{alpha$:0.01$\}$ with Batch Normalisation



Performance of CNN in terms of LOSS at different learning rates with Batch Normalisation and Xavier Initialisation

OBSERVATIONS :

- Large number of spikes are observed in loss while working with high learning rate.
- The model gave higher accuracy at α : 0.001

Question : 2,3

Test Accuracy : 88.2

Best parameters tuned which achieves highest accuracy on test data are :
Learning rate : 0.001
Initialisation : Xavier
Batch-size : 128
Batch- Normalisation : YES Other : Batch Shuffling, Input Distortions

Question : 4,5

CNN layer	Number of Inputs	Weights-Parameters + Biases	Number of Outputs
Layer:	Depth×height×width	Outputs×height×width×depth + Bias	Depth×height×width
CONV1	3×32×32	64×3×3×3 + 64	64×32×32
POOL1	64×32×32	0	64×16×16
CONV2	64×16×16	128×3×3×64 + 128	128×16×16
POOL2	128×16×16	0	128×8×8
CONV3	128×8×8	256×3×3×128 + 256	256×8×8
CONV4	256×8×8	256×3×3×256 +256	256×8×8
POOL3	256×8×8	0	256×4×4
FC1	256×4×4	1024×4×4×256 + 1024	1024
FC2	1024	1024×1024 + 1024	1024
SOFTMAX	1024	1024×10 + 10	10

Dimensions of Inputs and outputs at each fully-connected and convolutional layer alongwith number of parameters

Total Number of Parameters : 6216074
No. of Parameters in convolutional Layers : 960896 (15.45 percent)
No. of Parameters in fully connected Layers : 5255178 (84.54 percent)

Question : 6

Number of "**neurons**" in each layer is equal to number of outputs from that layer.

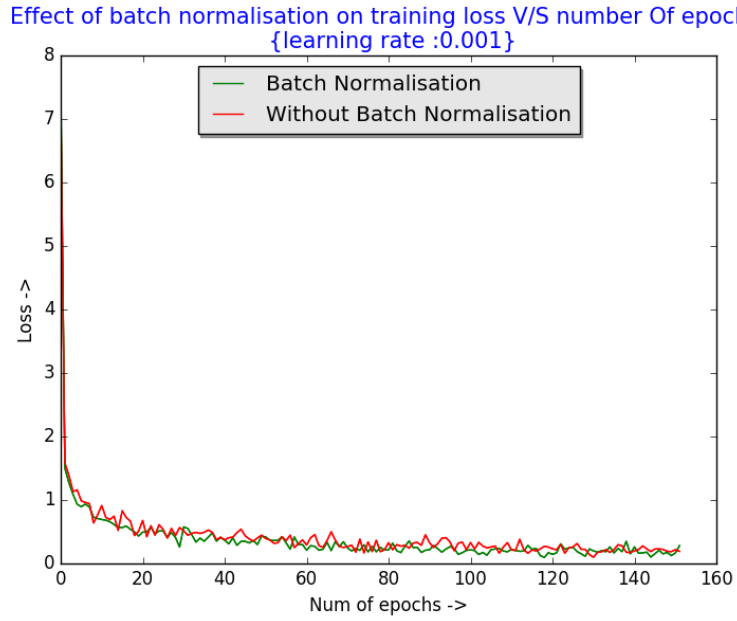
- INPUTS: 32×32×3
- CONV1: 64×32×32
- POOL1: 64×16×16
- CONV2: 128×16×16
- POOL2: 128×8×8
- CONV3: 256×8×8
- CONV4: 256×8×8
- POOL3: 256×4×4
- FC1: 1024
- FC2: 1024
- SOFTMAX: 10

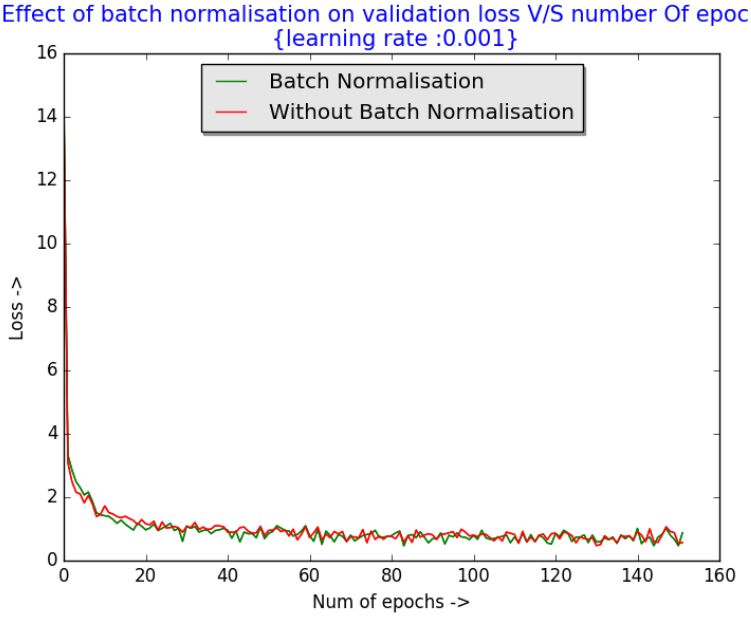
Total no. of neurons : 133130
no. of neurons in convolutional layers : 131072 (98.45 percent)
no. of neurons in fully connected layers : 2058 (1.54 percent)

[Excluding the representation volume in Input and poolng layers]

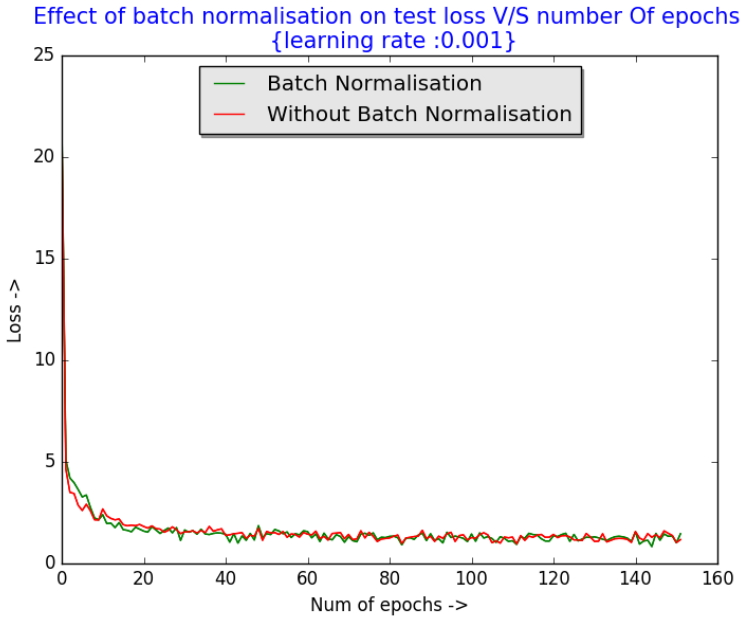
Question : 7

Effect of Batch Normalisation :





Effect of Batch Normalisation

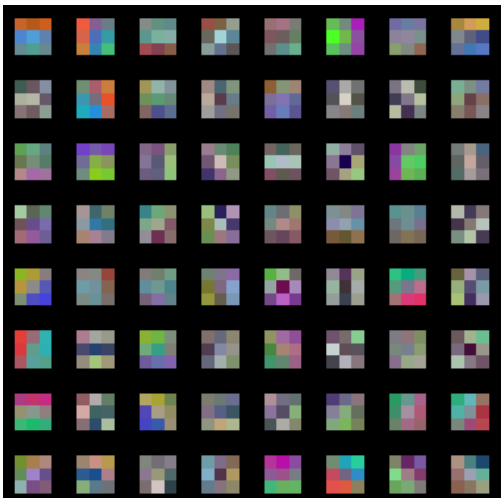


OBSERVATIONS :

- Our experiment results : Test accuracy = 88.2 % with BN and 87 % without BN (when run for 150 epochs)
- The reason being that Batch Normalisation speeds up training, and converges faster. We might have got same accuracy without BN also, if the experiment was run for more epochs.
- One advantage of using BN (what we studied theoretically) is that we could use higher learning rates to speed up training without getting oscillations in loss, but that didn't happen with us as we found many fluctuations in loss curve with increased learning rate (from 0.001 to 0.01).
- The plots clearly shows the above effects.
- Another observation with BN is that Validation and Test loss showed a drastic increase while training loss continued to decrease in initial training steps, but that did not happen while training with out using BN.

Question : 8

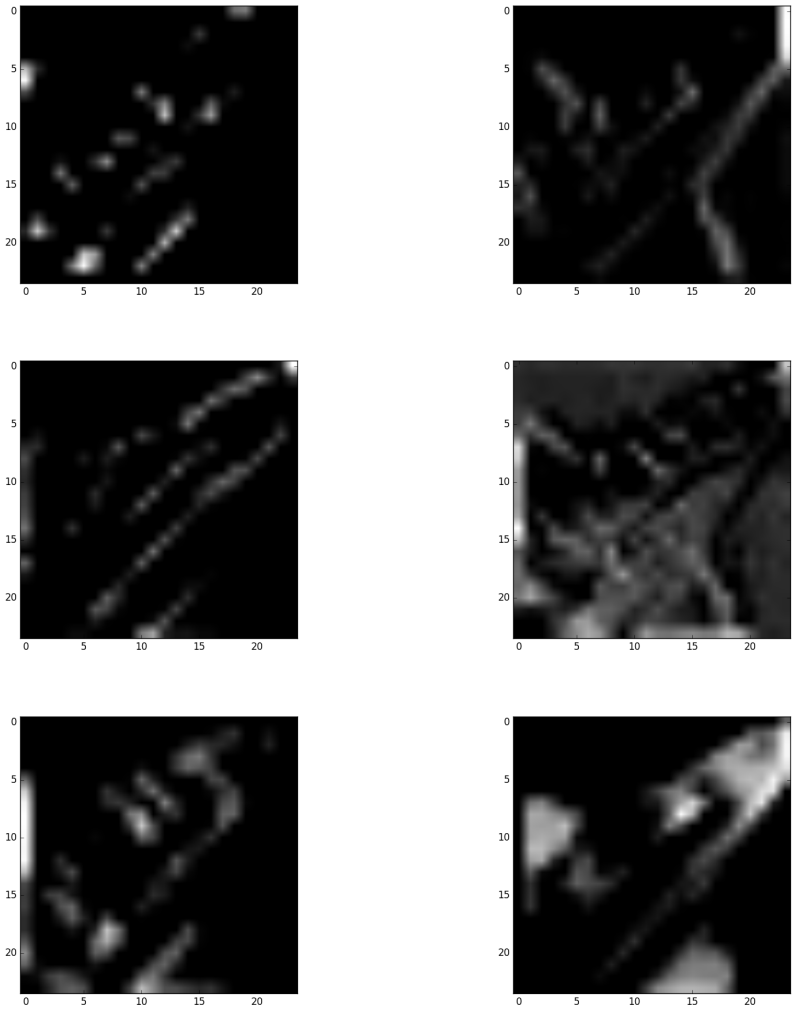
Convolution filters learnt at CONV1 layer at the end of training :



Visualisation of CONV1 layer filters (8×8 Grid)

OBSERVATIONS :

- We tried to visualise the features maps after the conv1 filters were applied.
- Although the patterns are not that clear in colored feature maps, the grayscale feature maps are much easy to interpret.



Gray scale feature maps

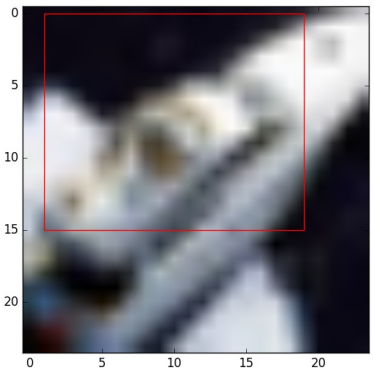
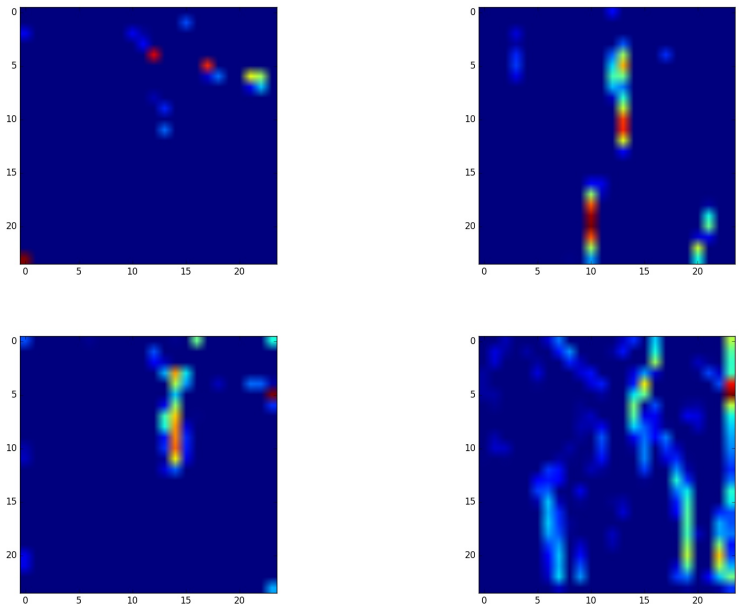


Figure 1: Caption



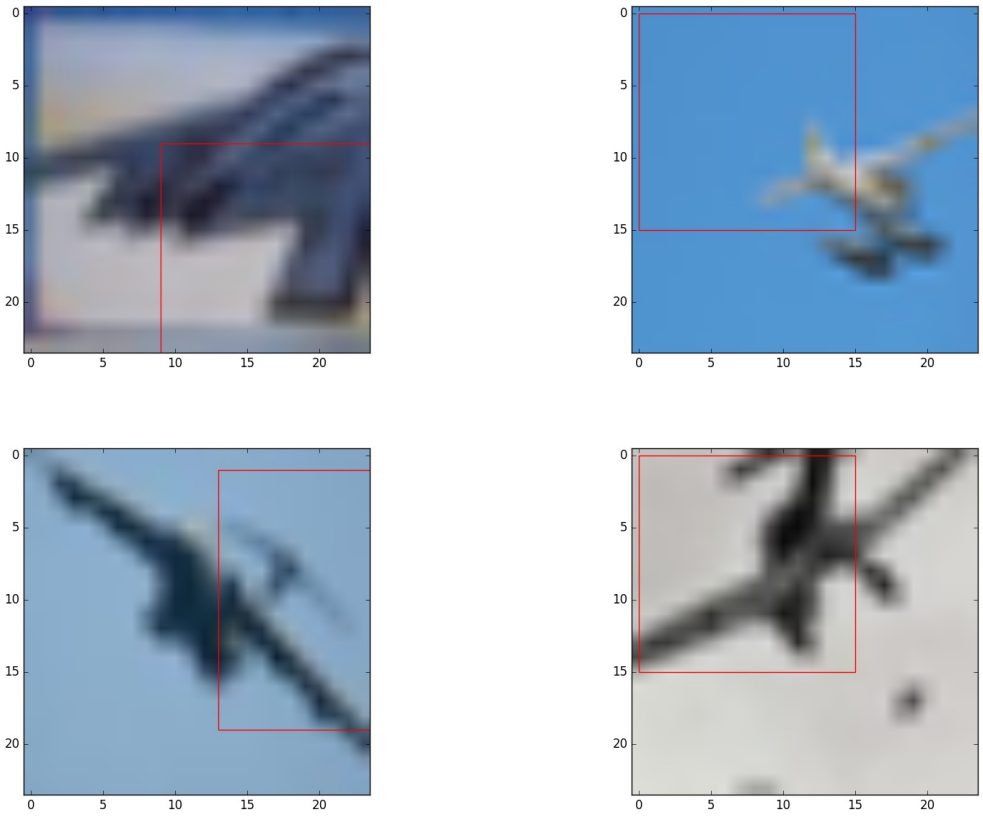
Coloured feature maps

The above feature maps correspond to the image shown below :

Question : 9

Implementation :

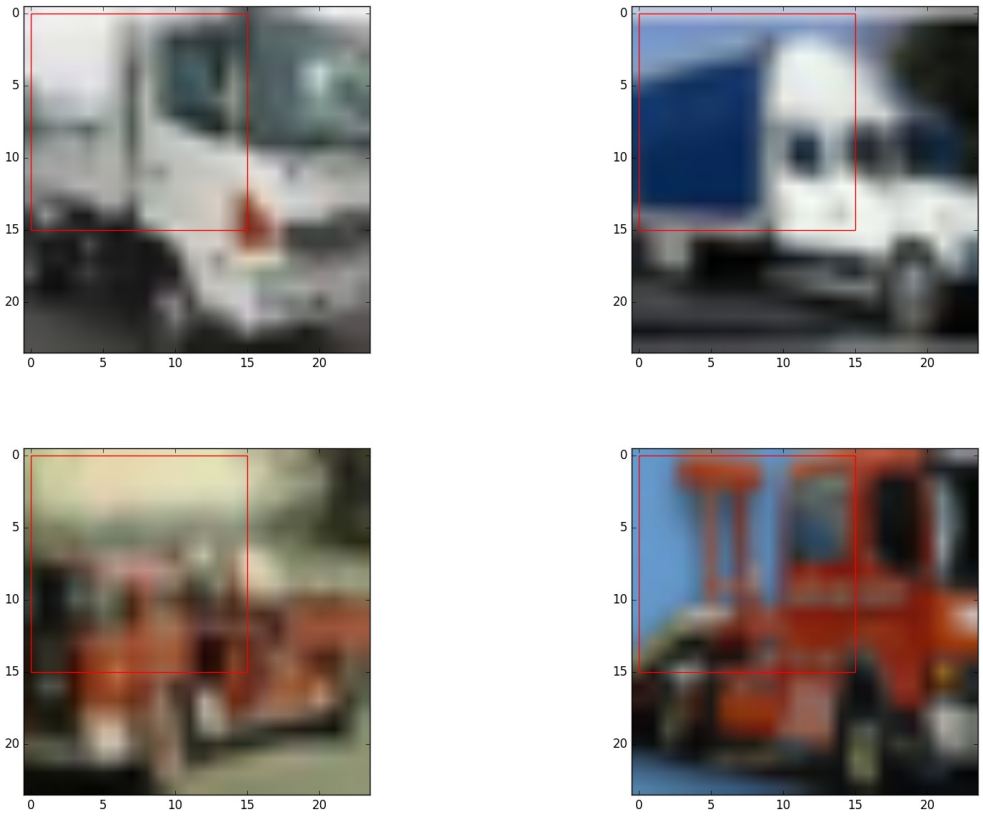
- We obtained patches/portions of images which excite some neuron at CONV3 by passing the images of **"same class"** to network and observing maximally excited neurons at output of CONV3.
- Then we computed the coordinates of image patch responsible for it by back tracing and calculating region of influence in each previous layer
- Finally we arrive at coordinates of the responsible portion/patch in original image (Code attached)
- Some of the exciting patterns observed are :



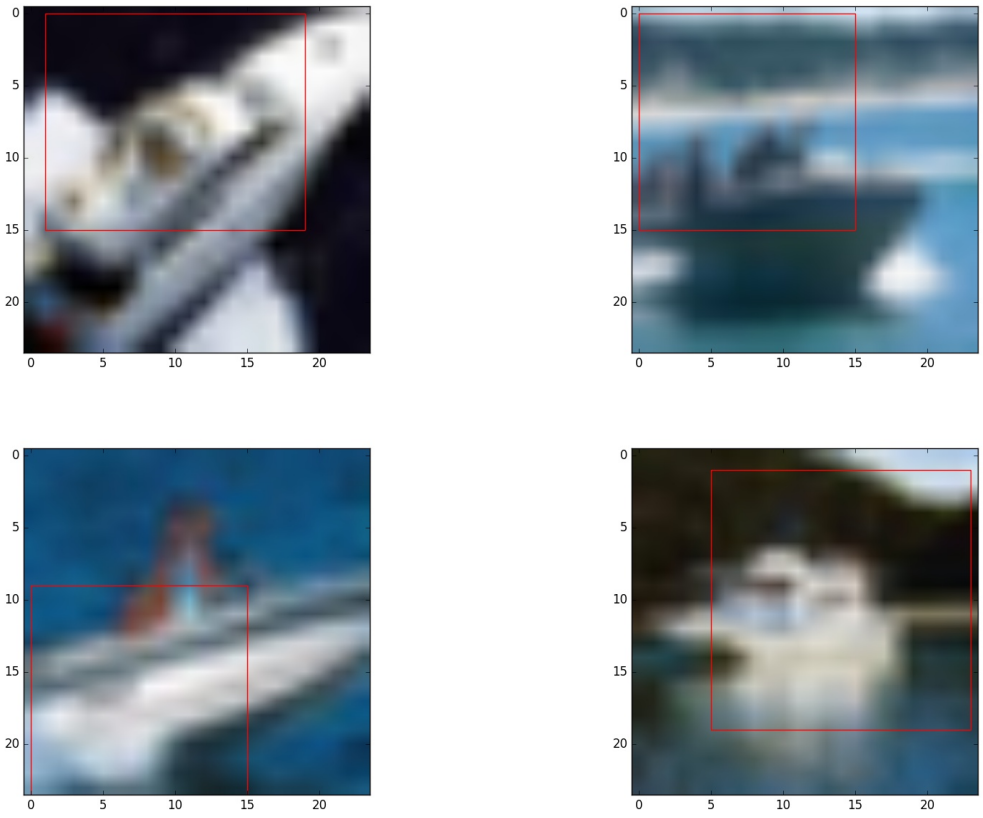
AIRPLANE : Patches obtained by backtracing neurons excited by "wing" of airplane

AUTOMOBILES : Patches obtained by backtracing neurons excited by ”front” of automobiles

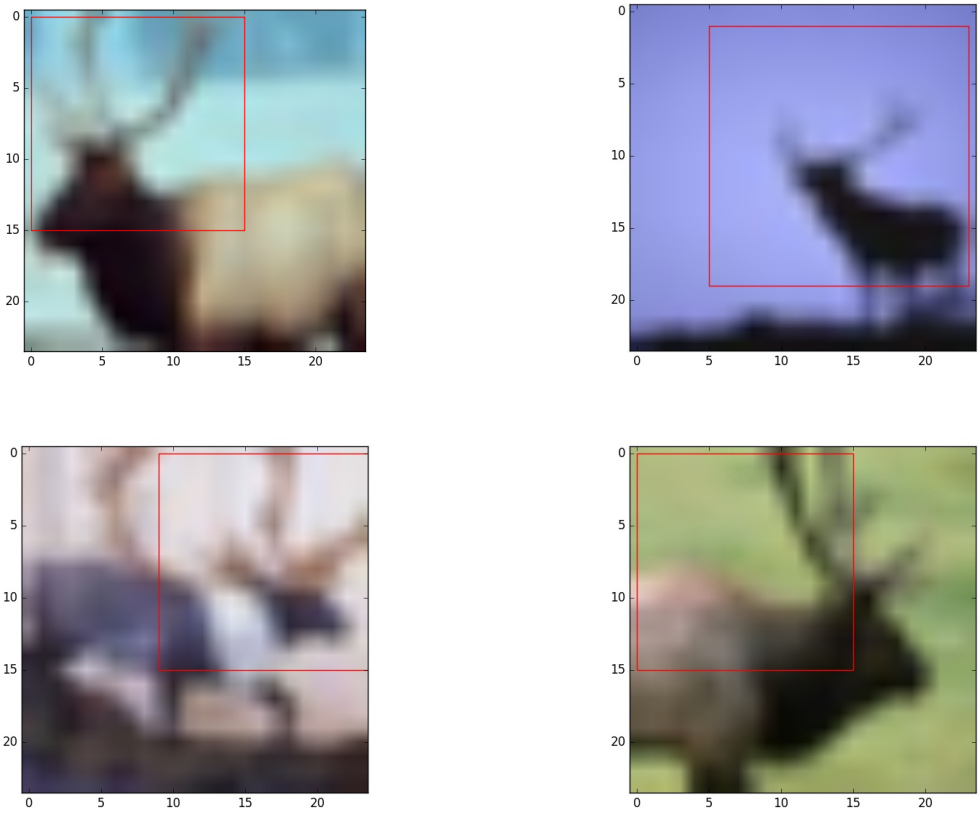
TRUCK : Patches obtained by backtracing neurons excited by "tyre" of truck



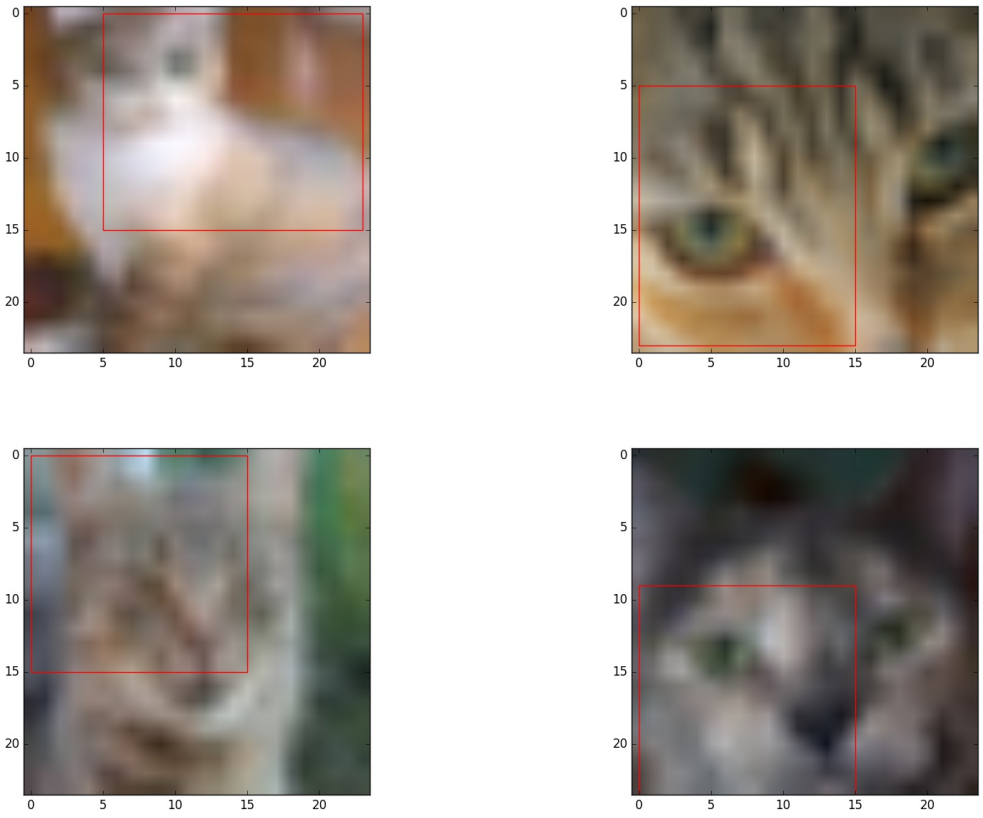
TRUCK : Patches obtained by backtracing neurons excited by ”body” of truck



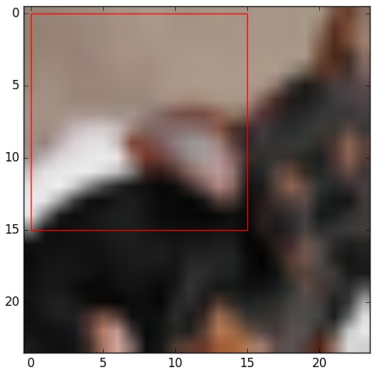
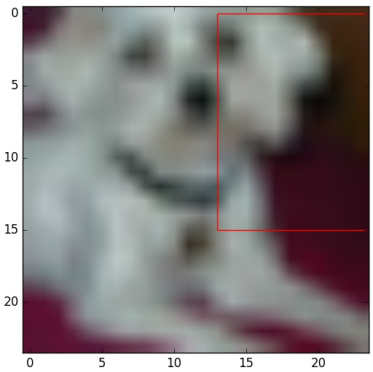
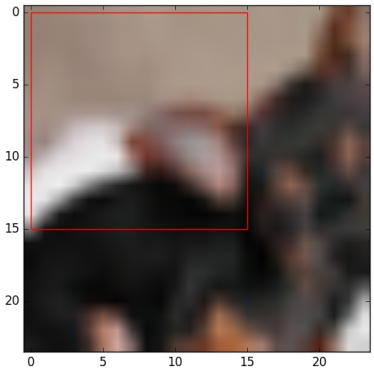
SHIP : Patches obtained by backtracing neurons excited by "top portion" of ship



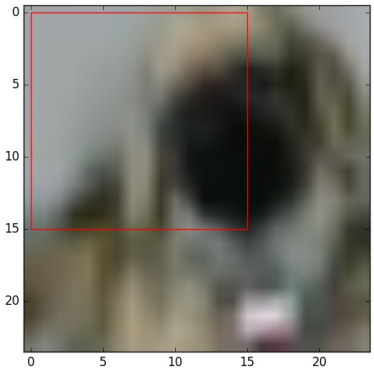
DEER : Patches obtained by backtracing neurons excited by "front face with horns" of deer

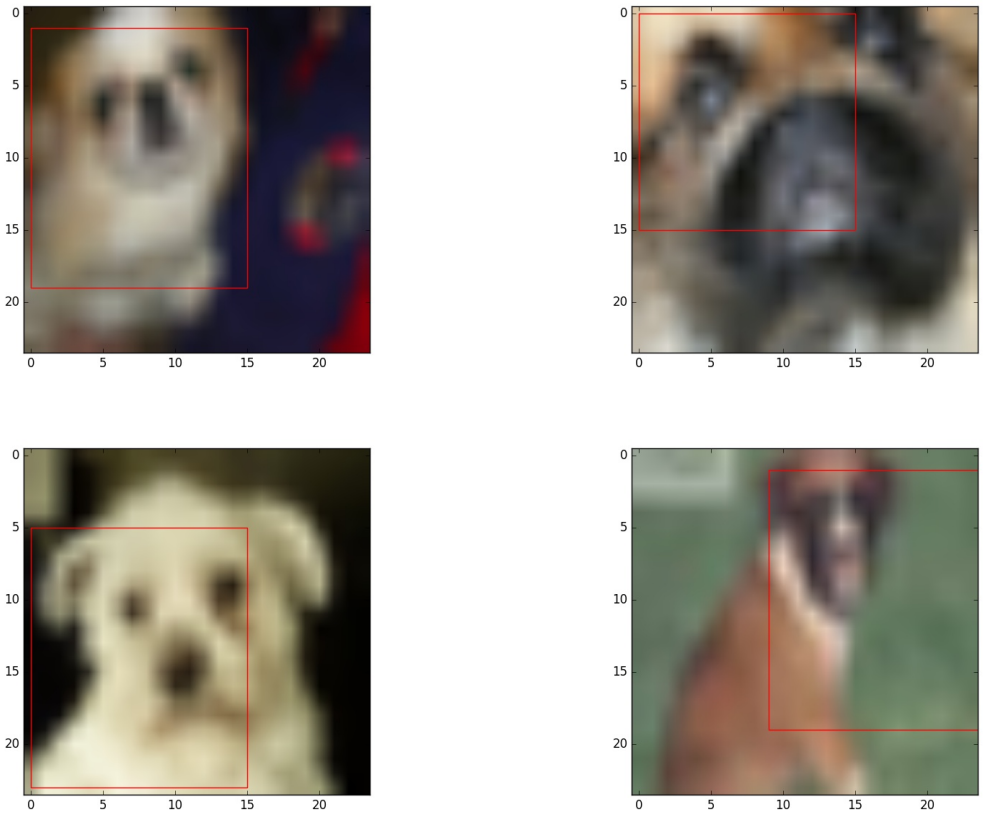


CAT : Patches obtained by backtracing neurons excited by ”eye” of cat

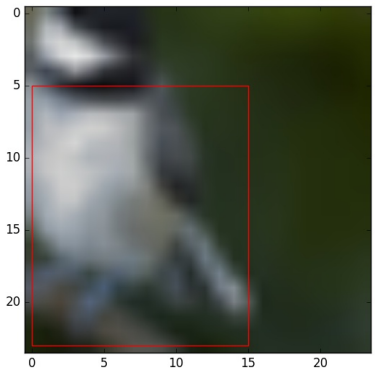
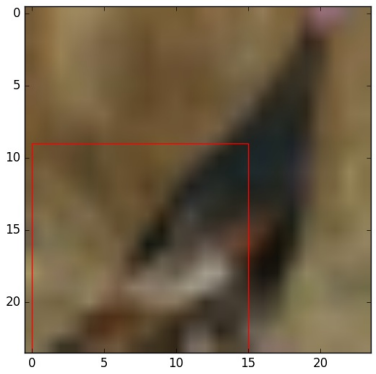
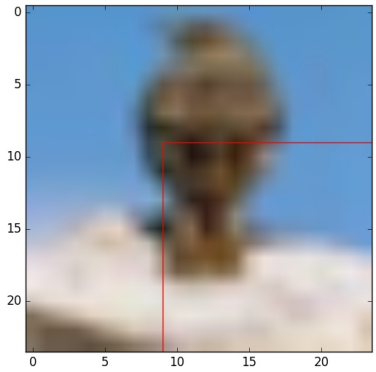




DOG : Patches obtained by backtracing neurons excited by ”ear” of dog





DOG : Patches obtained by backtracing neurons excited by "nose" of dog

BIRD : Patches obtained by backtracing neurons excited by "tail" of bird

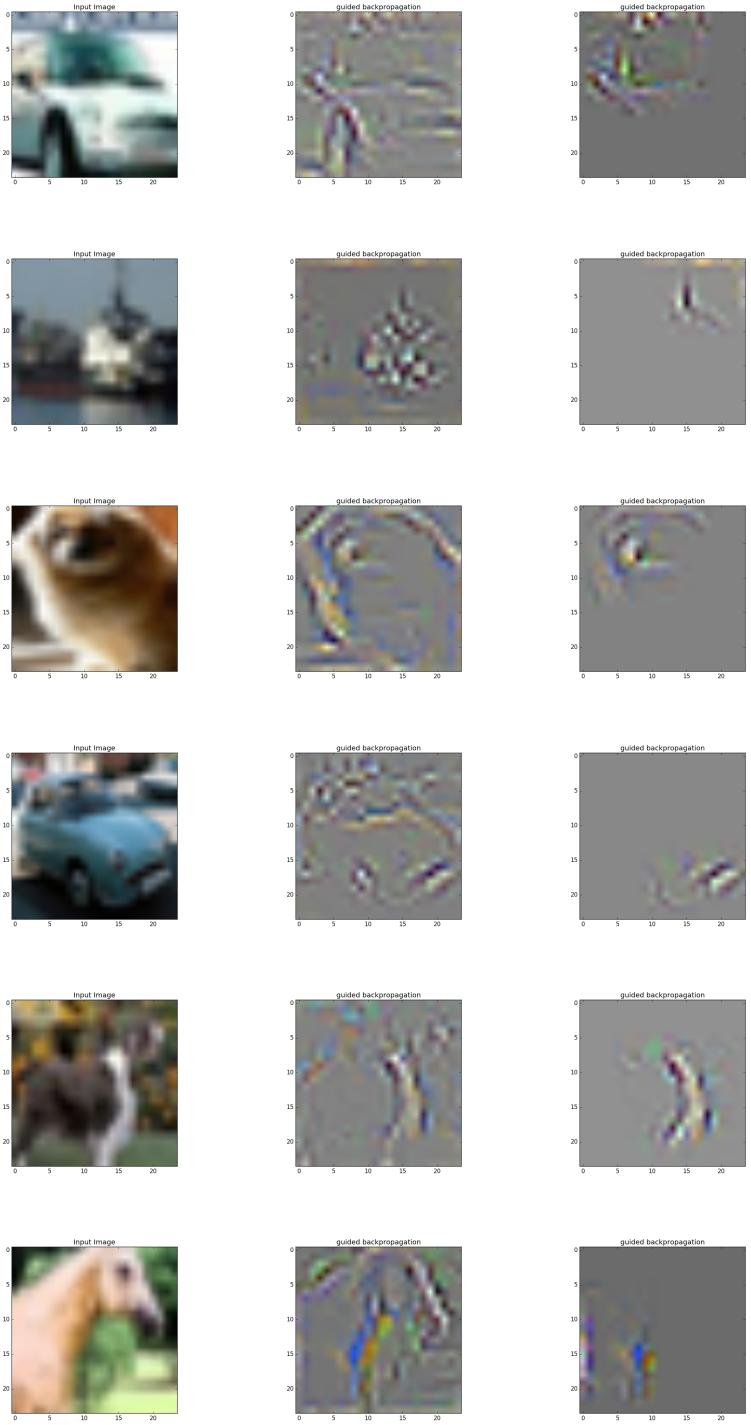
Observations :

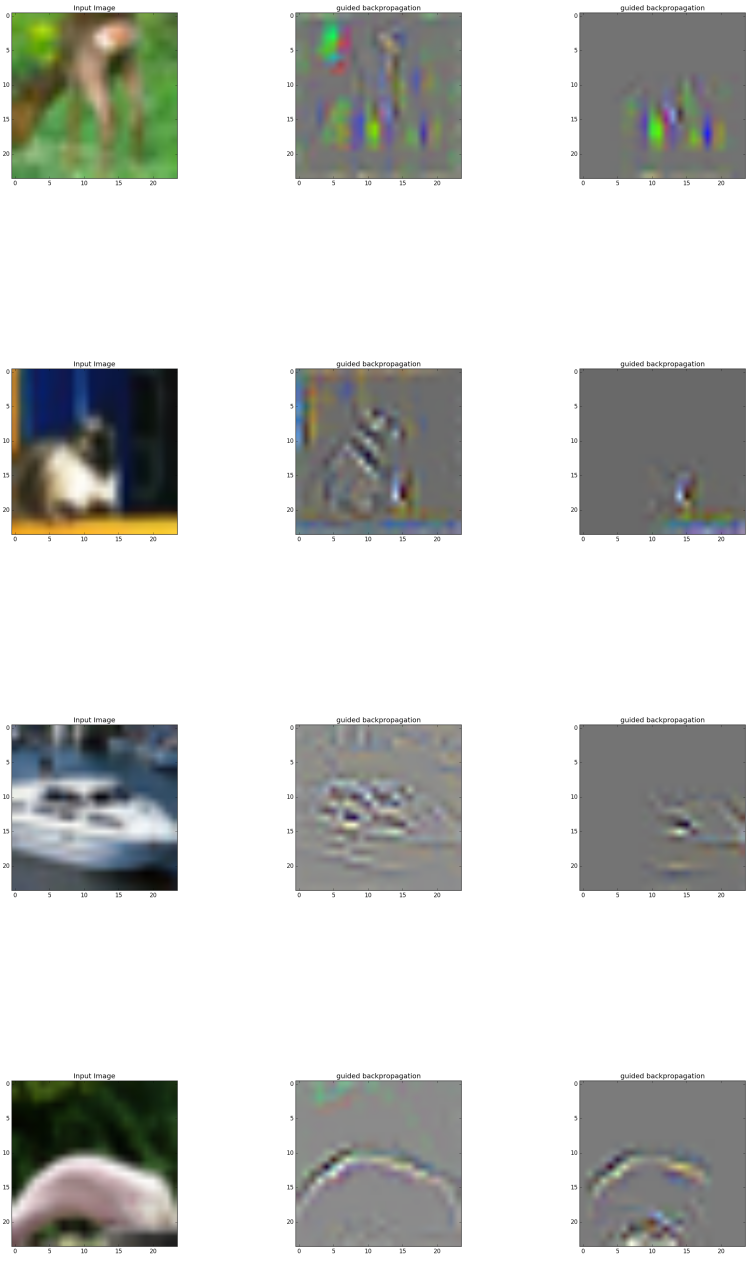
- This is naive implementation of DECONVOLUTIONS at different layers by tracing back their coordinates
- It gave decent results on CIFAR-10

Question : 10

Implementation of Guided backpropagation :

First gradient plot shows the guided backpropagation from CONV3 layer and second gradient plot shows the guided backpropagation from maximally excited neuron in CONV3 layer.


Input Image
guided backpropagation
guided backpropagation
Input Image
guided backpropagation
guided backpropagation
Input Image
guided backpropagation
guided backpropagation
Input Image
guided backpropagation
guided backpropagation
Input Image
guided backpropagation
guided backpropagation
Input Image
guided backpropagation
guided backpropagation



Images with their corresponding gradient plots.