CS7015: Deep Learning
Assignment 1B

Divya K Raman
EE15B085

29 August 2018

1 Introduction

In this assignment, we use the model that gave us the best results in assignment 1A to perform various experiments on the cifar 10 dataset. Code uses pytorch in python. It has been run using google colab's gpu. In the first part of this assignment, we perform an occlusion sensitivity experiment wherein we cover parts of the image and see how the probability of the true class varies. This tells us whether the model has really learned the location of the object in the image or if the model just classies the image based on surrounding or contextual cues. In the second part of the assignment, we do a filter analysis experiment. First, we randomly select 10 filters from our model and find images where the response to this filter is maximum. We also display the top 5 responses along with the corresponding images. We then turn off these filters and find classes and images that start to mis-classify now, but were classied correctly earlier. Details of the experiments and the results are presented below.

2 Neural Network architecture

The neural network architecture is as follows:

conv1 : Conv2d(3, 256, kernel size=(5, 5), stride=(1, 1))
pool1: MaxPool2d(kernel size=2, stride=2, padding=0, dilation=1, ceil mode=False)
conv2: Conv2d(256, 1024, kernel size=(3, 3), stride=(1, 1))
pool2: MaxPool2d(kernel size=2, stride=2, padding=0, dilation=1, ceil mode=False)
conv3: Conv2d(1024, 2048, kernel size=(3, 3), stride=(1, 1))
pool3: MaxPool2d(kernel size=2, stride=2, padding=0, dilation=1, ceil mode=False)
conv4: Conv2d(2048, 2048, kernel size=(1, 1), stride=(1, 1))
pool4: MaxPool2d(kernel size=2, stride=2, padding=0, dilation=1, ceil mode=False)
fc1: Linear(in features=2048, out features=1024, bias=True)
fc2: Linear(in features=1024, out features=256, bias=True)
fc3: Linear(in features=256, out features=10, bias=True)

The accuracy that we got for this model in assignment 1A on the CIFAR 10 testset was 77 percent. However, normalisation of data was not done in assignment 1A. Here, we normalise the data. We then retrain the network using this model and the normalised data. Testing this model on the test set gives us an accuracy of 81 percent. Accuracy on the test set increases upto epoch 13 and then starts decreasing, probably due to overtraining and hence overfitting of the network on the train set. We use the neural network weights obtained after epoch 13 which gives us a test set accuracy of 81 percent for further experimentation in this assignment.

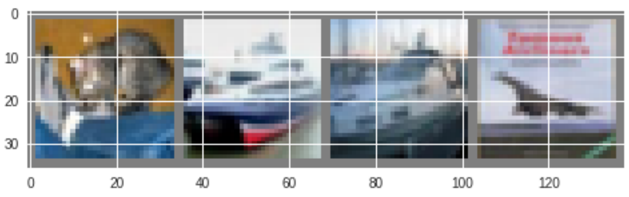
These weights are stored in a .pth file and loaded into the jupyter notebook.

3 Question 1

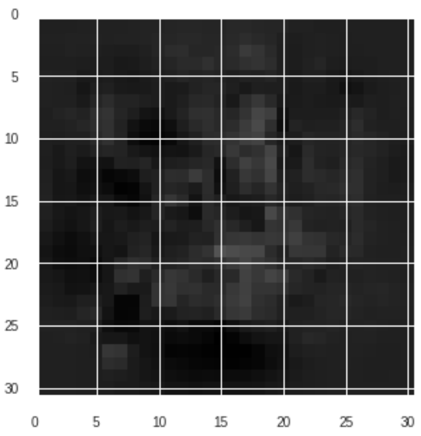
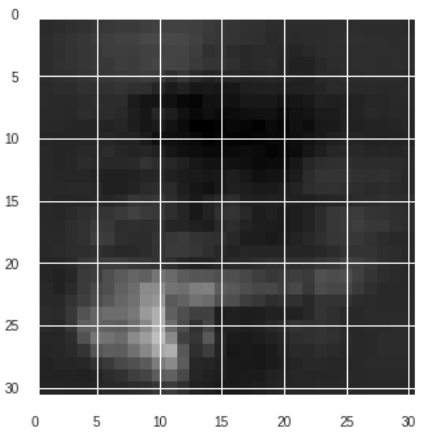
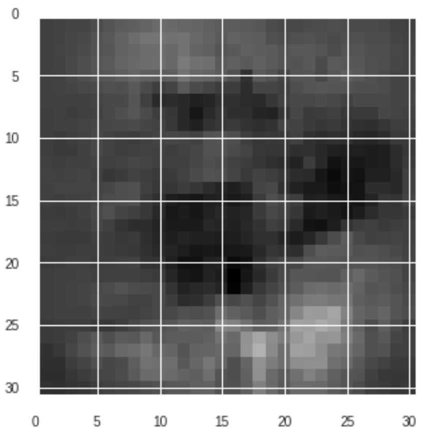
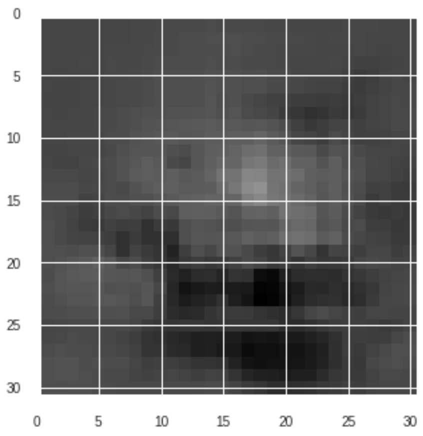
We perform the occlusion sensitivity experiment in this question. For each pixel position in the image, we consider windows of different (33,55,77,99) around (i,j) and replace the content of the window with gray pixels. We then pass the modied image (with respect to the position (i,j)) through the model and note down the probability for true class into an array. i.e., condence(i, j).The confidence array is then plotted as an image. The jupyter notebook for this question is in the file named Sem7DLAssignment1BQ1. We perform this experiment on 20 images.

The code for this question as well as the results for the 20 images are in the jupyter notebook file. We present the results for the first eight images below:

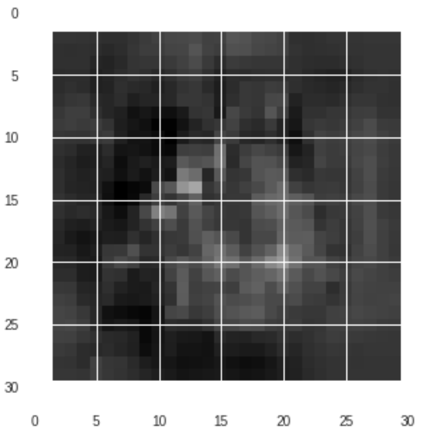
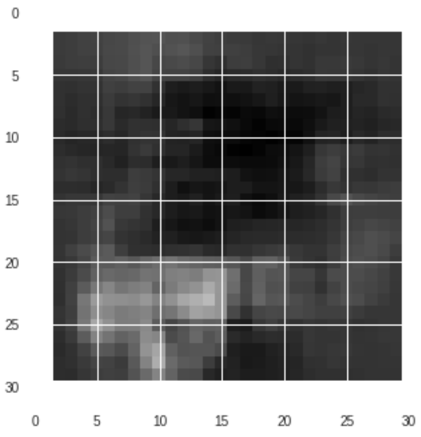
Images 1 to 4:

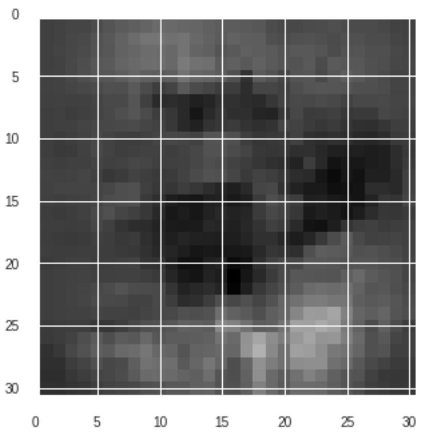
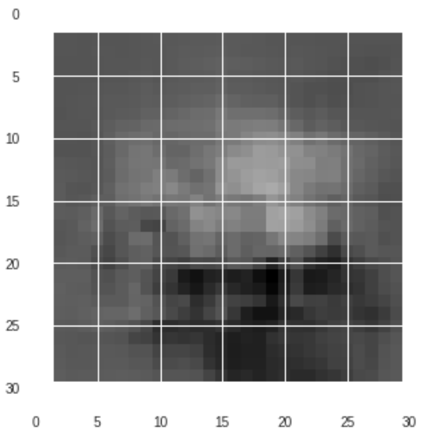


3*3 occlusion for images 1 to 4 in the same order:

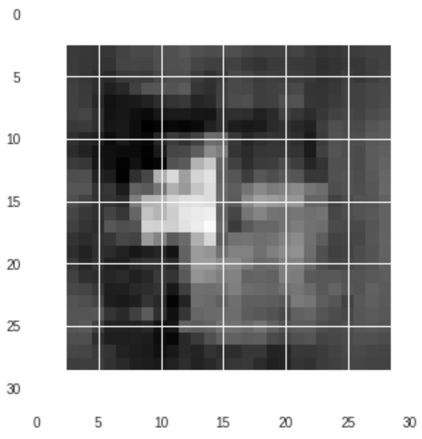
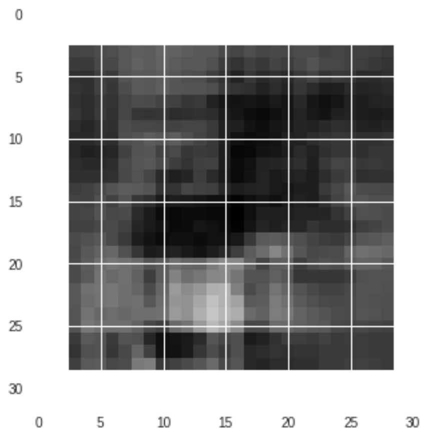
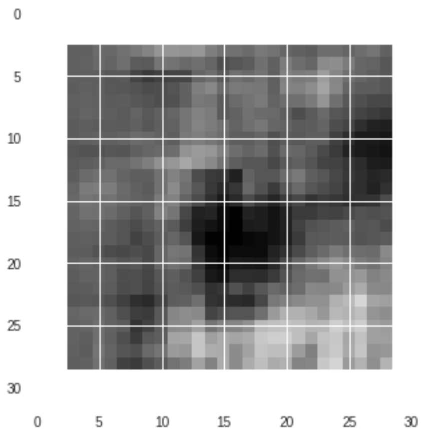
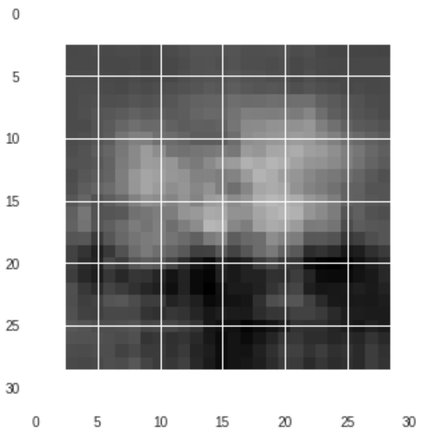





5*5 occlusion for images 1 to 4 in the same order:

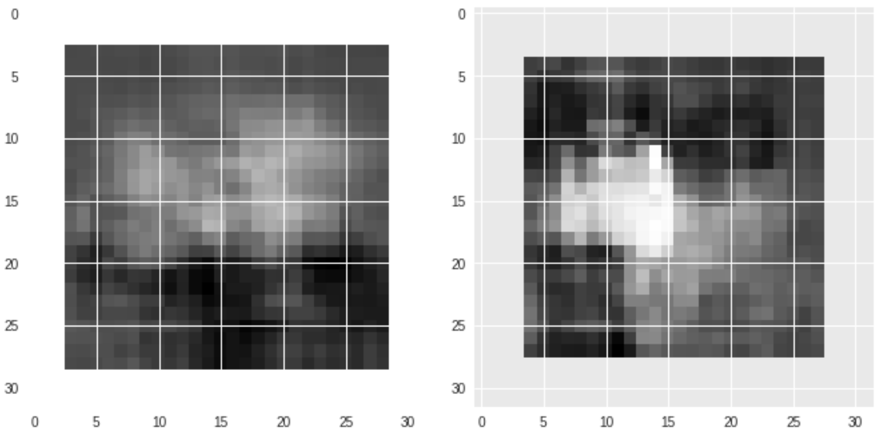



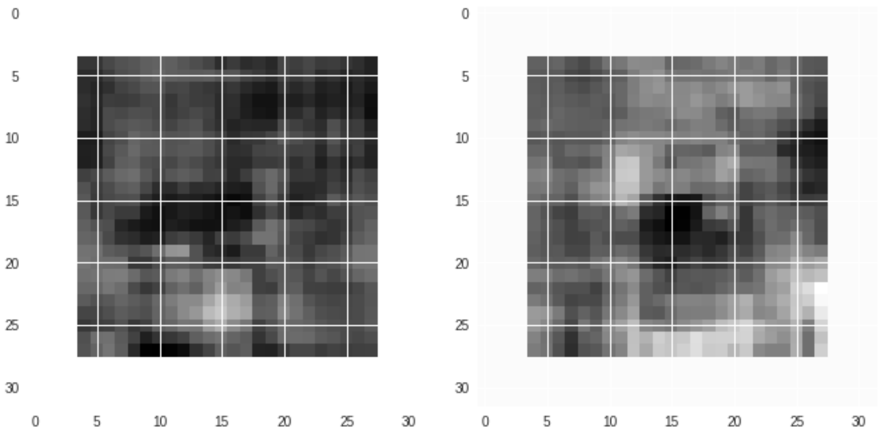



7*7 occlusion for images 1 to 4 in the same order:

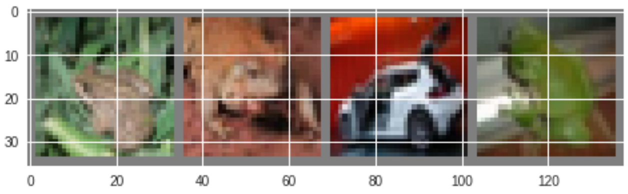





9*9 occlusion for images 1 to 4 in the same order:

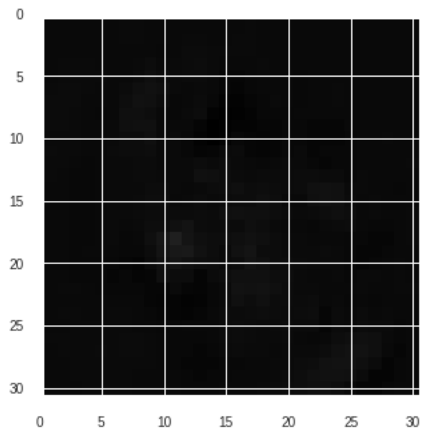
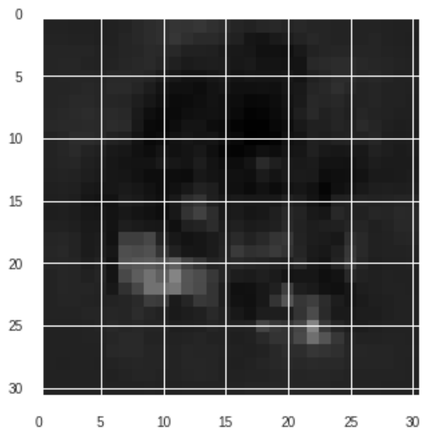
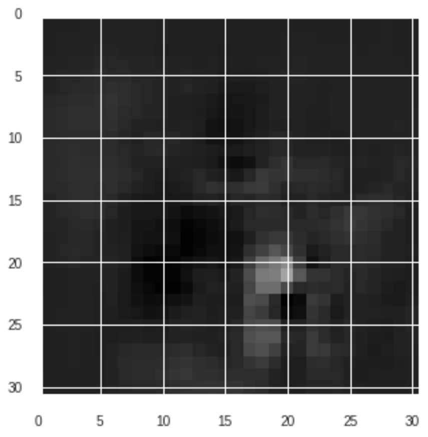
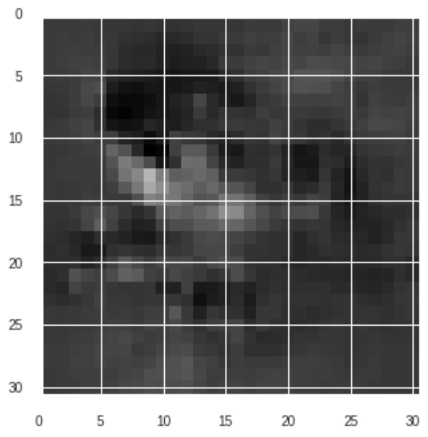




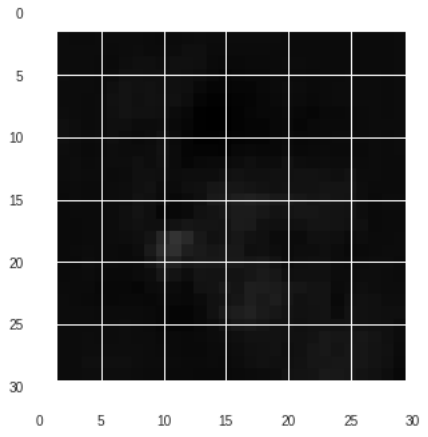
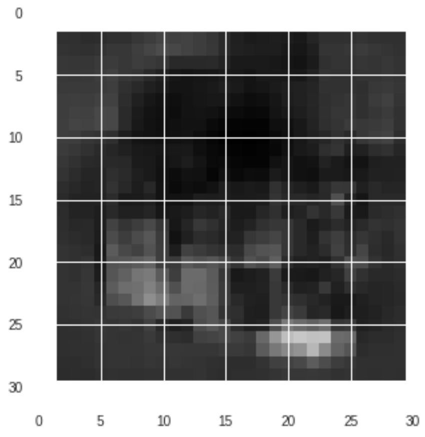
Images 5 to 8:

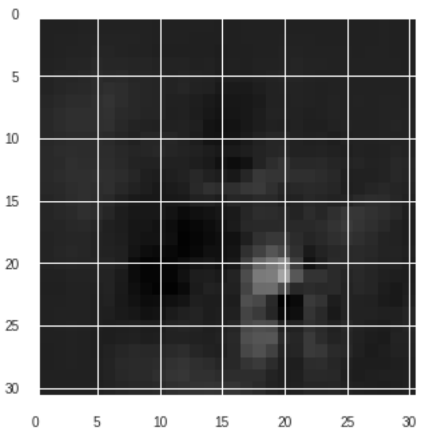
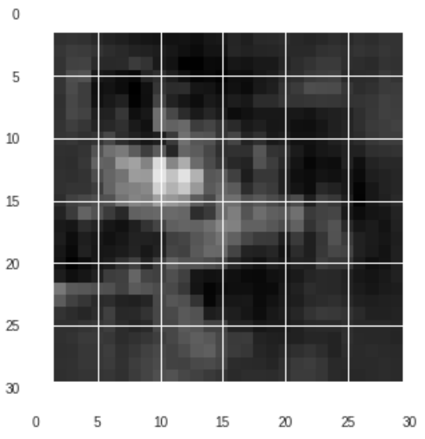


3*3 occlusion for images 5 to 8 in the same order:

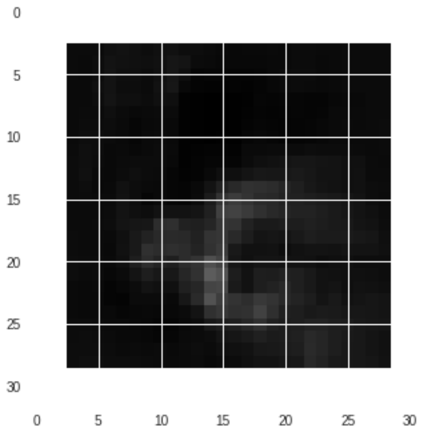
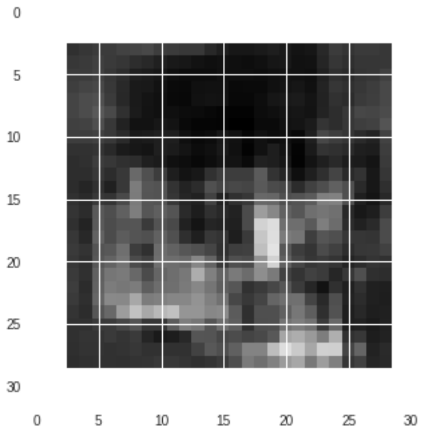
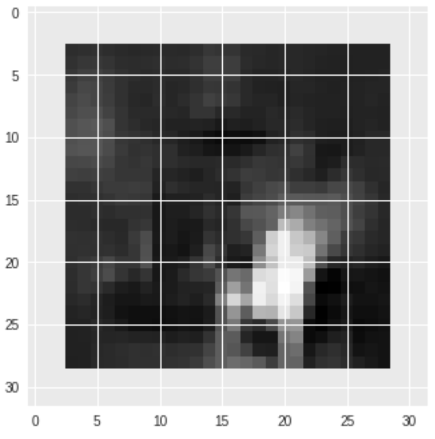
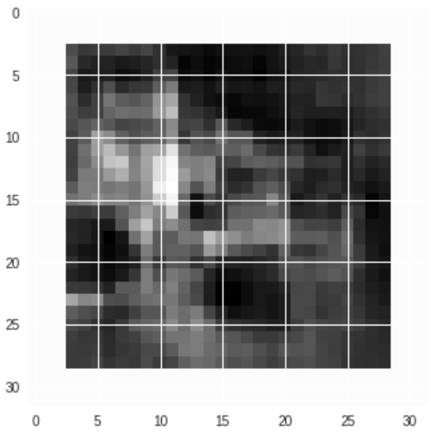





5*5 occlusion for images 5 to 8 in the same order:

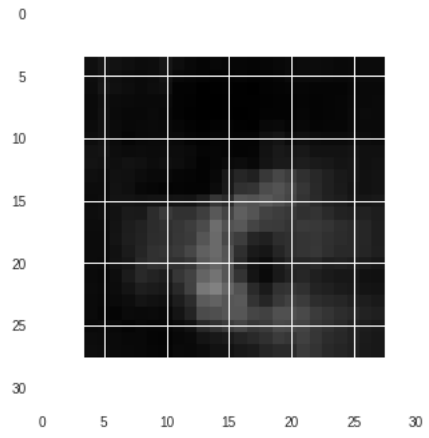
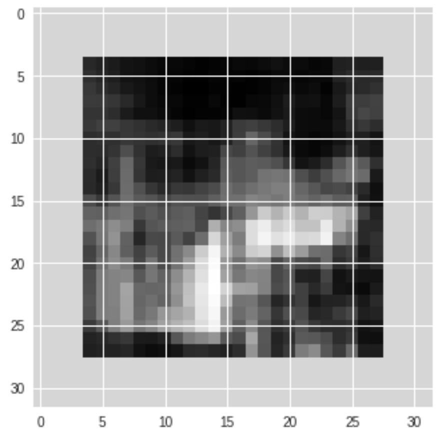
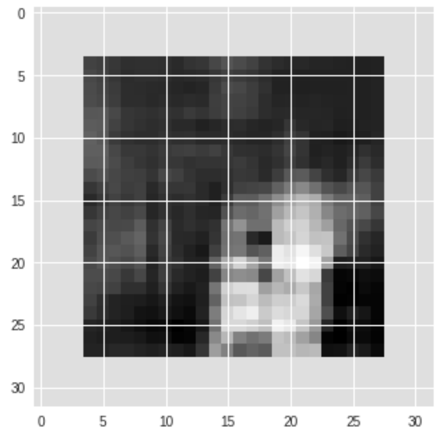
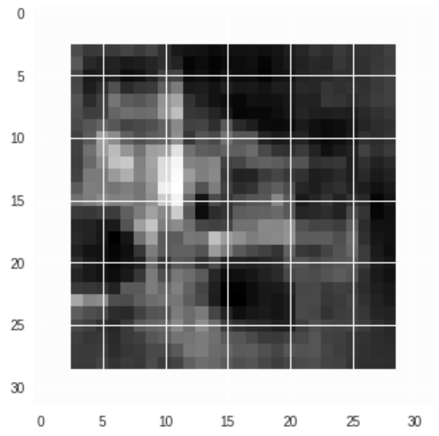



7*7 occlusion for images 5 to 8 in the same order:

9*9 occlusion for images 5 to 8 in the same order:

For images 9 to 20, the results are there(along with the results of images 1 to 8) in the output of the last cell in the jupyter notebook named Sem7 DL Assignment1B Q1. The outputs are arranged in the order images 1 to 4, followed by 3*3 window occlusion for images 1 to 4, followed by 5*5 window occlusion for images 1 to 4, followed by 7*7 window occlusion for images 1 to 4,followed by 9*9 window occlusion for images 1 to 4. The same pattern follows for images 5 to 8, images 9 to 12, images 13 to 16, and then images 17 to 20.

For most of the images, the confidence score for true class is high at most pixel locations when occluded by a 3*3 grey window. This is because, the network gives the correct category of image when the occlusion window is small. These results slowly deteriorate in the case of 5*5 occlusion windows followed by 7*7 and then 9*9. Results aren't too good when the occlusion window size is 9*9. This is because a large part of the image is covered and thus the network has much lesser features to test on. Furthermore, covering key regions of the image which tend to have the distinguish features(car- tyre, shape; cat-tail,face) of the class yield a low confidence score. However, covering regions of the image like the the background details, surroundings, etc don't seem to affect the results much. Covering the face of a cat leads to a high probability of it being misclassified as a dog. Similarly, car and truck can be confused with each other.

The overall results indicate that the model has more or less learned the location of the object in the image and doesn't just classies the image based on surrounding or contextual cues.

4 Question 2

In the second question, we perform a filter analysis on our model. The jupyter notebook for this question is named Sem7 DL Assignment1B Q2. We first choose 10 filters from our model(Since we have only 4 convolutional layers in our model, we choose 3 from the first layer, 3 from the second, 2 from the third and 2 from the fourth layer):

layer 1: output channels 100,200,240 layer 2: output channels 400,800,1000 layer 3: output channels 1000,2000 layer 4: output channels 1000,2000

Question 2.1

We then nd images where the response in this lter is the maximum across all images possible. We also display top 5 of the image patches corresponding to the maximum response position in the lter map. Cell 30's output in the jupter notebook has the results for this part of the question.

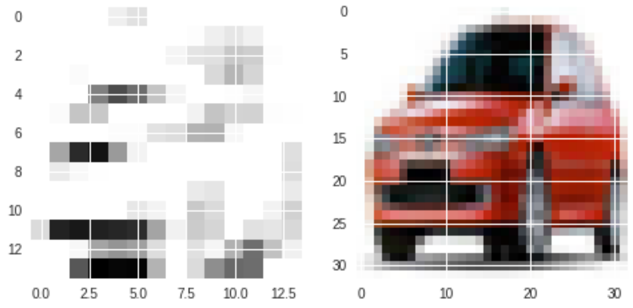
To find the maximum response across all images for a given filter, we take the matrix norm of the output of that filter. For I input feature maps(channels), O output channels, K*K kernel size, the weight values of the convolution layer is in a matrix of dimension O*I*K*K. Output of the convolution layer is of size B*O*H*W where B is the batch size and H and W are the output feature map height and width. We access the output of the ith image and oth filter using output[i,o,:,:]. Furthermore, to get the response of the various layers of the NN architecture, we store the output of each layer in a variable and return it along with the final class outputs. This is done in the 'forward' function when the class for the NN architecture is defined.

The result(image patches) are arranged in the order:
5th Highest Response
Filter 1 Filter 2 Filter 3 Filter 4 Filter 5 Filter 6 Filter 7 Filter 8 Filter 9 Filter 10
4th Highest Response
Filter 1 Filter 2 Filter 3 Filter 4 Filter 5 Filter 6 Filter 7 Filter 8 Filter 9 Filter 10
3rd Highest Response
Filter 1 Filter 2 Filter 3 Filter 4 Filter 5 Filter 6 Filter 7 Filter 8 Filter 9 Filter 10 2nd Highest Response
Filter 1 Filter 2 Filter 3 Filter 4 Filter 5 Filter 6 Filter 7 Filter 8 Filter 9 Filter

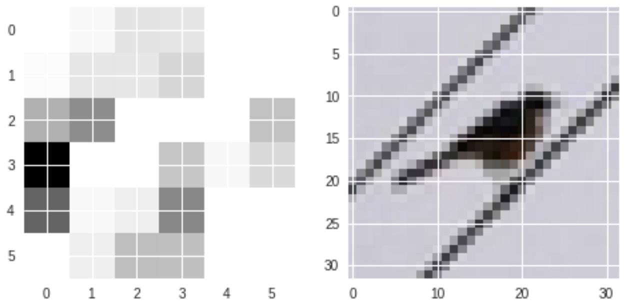
10
Highest Response
Filter 1 Filter 2 Filter 3 Filter 4 Filter 5 Filter 6 Filter 7 Filter 8 Filter 9 Filter 10

We display a few results below:

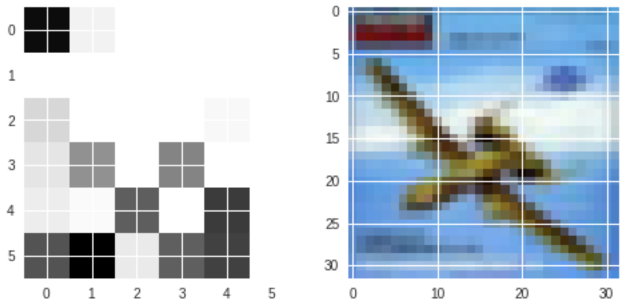
Highest response filter 2



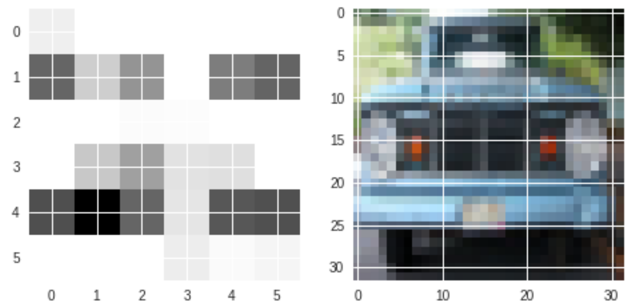
Highest response filter 4



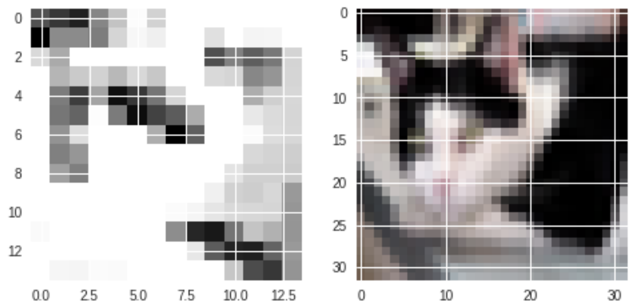
Highest response filter 5



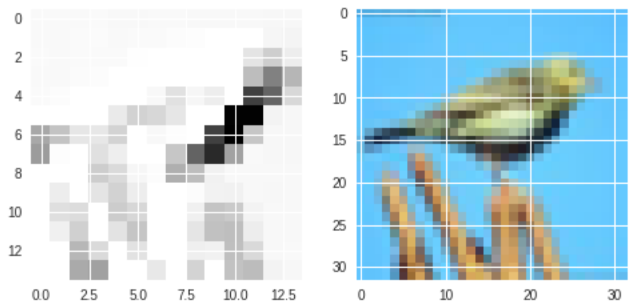
Highest response filter 6



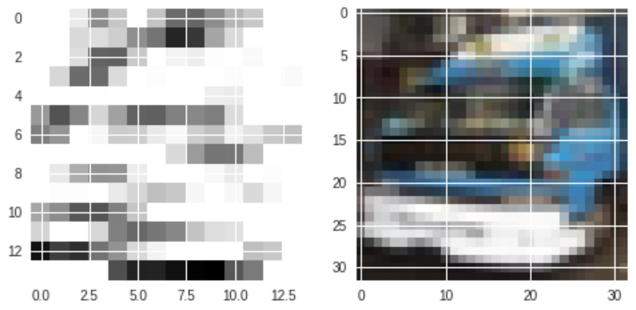
4th Highest response filter 1



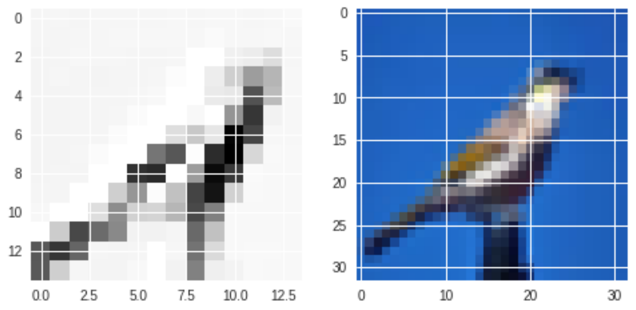
4th Highest response filter 2



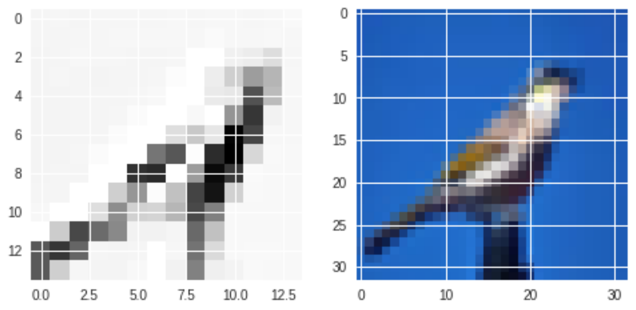
4th Highest response filter 3



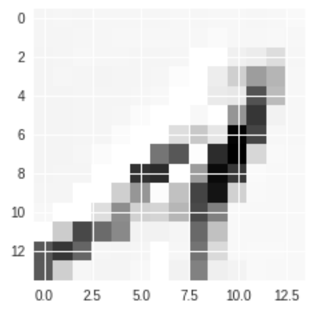
5th Highest response filter 1

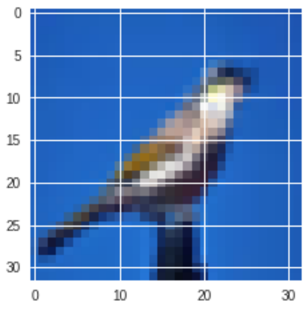


5th Highest response filter 2



5th Highest response filter 3





Observations:

Layer 1 output size: 256*14*14
Layer 2 output size: 1024*6*6
Layer 3 output size: 2048*2*2
Layer 4 output size: 2048*1*1

Filter 1,2,3 extract out the basic structure of the image. They are in layer 1. Not much is interpretable out of results of filter 7,8,9 and 10 because. This is because filter 9 and 10 are from output of layer 4 which is 1*1 dimensional and filter 7 and 8 are from output of layer 3 which is 2*2 dimensional. Filter 4,5,6 outputs are also hard to interpret, they are of size 6*6. But they too seem to be capturing features from the overall shape of the object in the image.

Question 2.2

We now switch off these 10 filters(make the weights of these 10 filters 0). Testing the model on the test set gives us an accuracy of 68 percent which is not too bad. This tells us that the network on an average has learnt pretty well. We also print out the original class that each image belongs to and the predicted class using this model for the first 1000 images. This result is there in the second last cell. One major observation is that cat and dog tend to misclassify(dog gets labelled as cat and cat as dog). Frog also gets labelled as dog in some cases. Deer and horse get labelled as cat in some cases. Truck in some cases gets miscalssifies as car and car as truck.