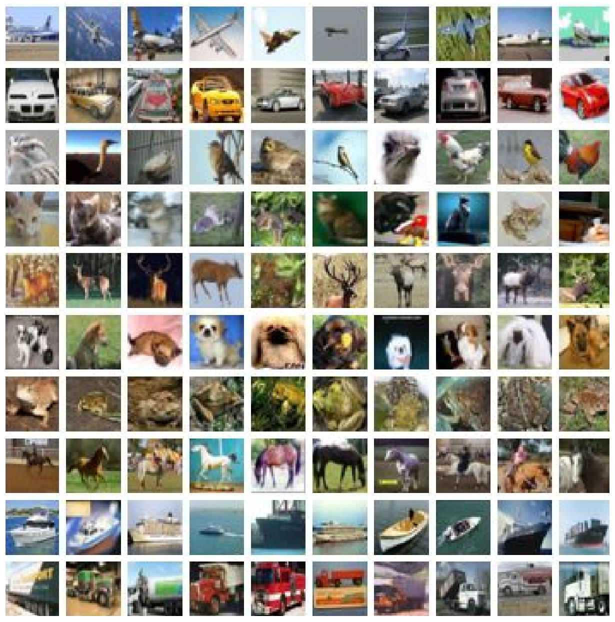
CS7015: Deep Learning (for Computer Vision)

Name: **Jose Moti**
Roll no: **CE17B118**
Date: **28/08/2019**

Programming Assignment 1

Introduction

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. In this programming assignment, we trained on the training batches using different parameter arrangements in order to find out how the test accuracy varies with these parameter changes. We also did an occlusion experiment as well as filter analysis on the best model that gave us maximum test accuracy.



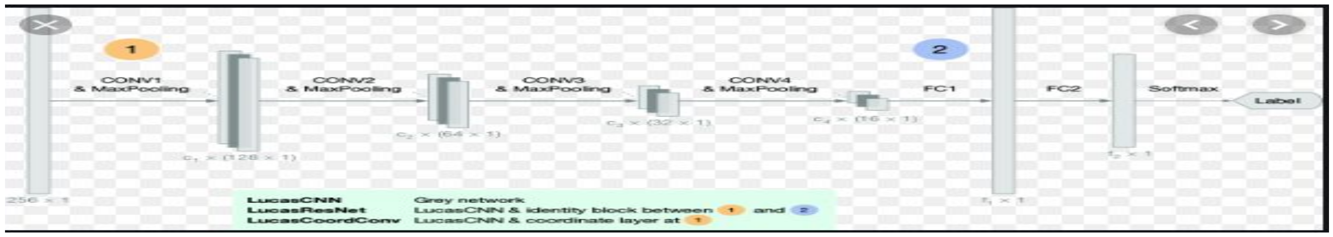
1 Part-A

To come up with a study of the effect of some of the following network parameters on the classification performance. I used the boilerplate code and varied parameters from it. The basic block form of operations done on the image is shown below.

image-->(conv_layer_1)-->(conv_layer_2)-->........-->(fc_layer_1)-->(fc_layer_2)-->....

The number of layers is varied for testing. Each conv_layer consists of

Conv2d-->MaxPool2d-->relu



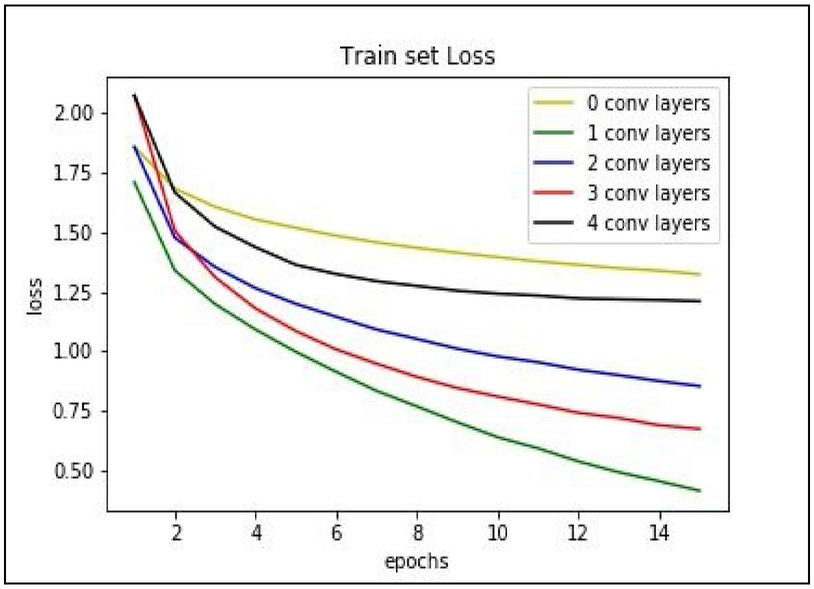
A typical convolutional network

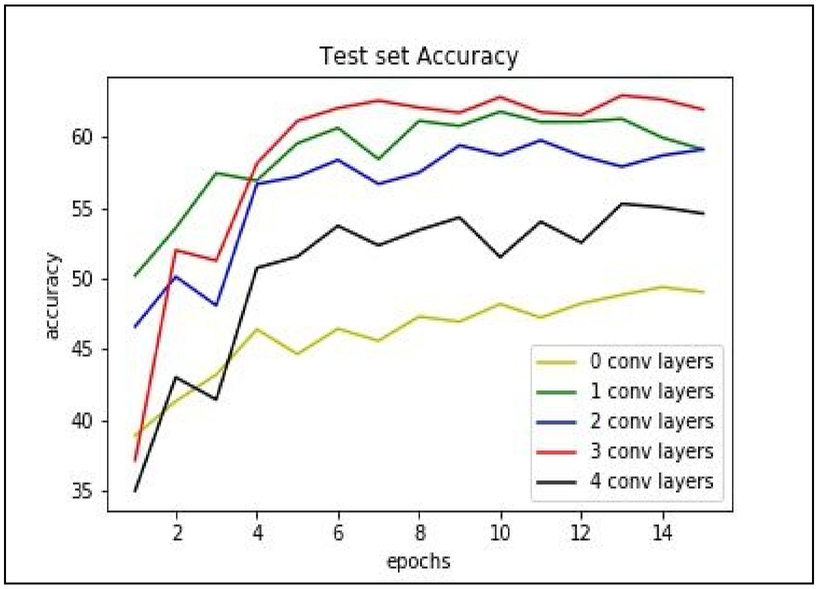
*Codes for Part 1 are given in assignment folder. The parameters are changed in the code itself in each section.

1.1) Number of Convolutional (Conv) layers

From the basic model given I varied the number of conv layers from 0-4 in order to find its effect in training loss and test accuracy. The number of FC layers used is 3 (same as in boilerplate code). The results are plotted below:

Plots





Plots Inference

- A common trend is not visible for training loss. Using 1,2,3 gives fewer loss values when compared to 0 and 4 conv layers. Least loss while using 1 conv layer.
- Similarly, for test processes too no common trend is visible. Using 1,2,3 conv layers gives better test accuracy than 0,4. Highest accuracy while using 3 conv layers.
- So using conv layers actually helps in increasing test accuracy as well as decreasing training loss. But a higher number of conv layers can cause a negative impact on loss as well as test accuracy. This is because while using more conv layers the model fails to identify the identity function thus making them fall into local minima. This results in lower test accuracy. Res-Nets can be used to avoid this.

Table

Models	Final loss	Best test accuracy	Class wise accuracy									
			Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Model 1.0 (0 conv layers)	**1.32**	**49%** epoch 15	**61**	**60**	**31**	**13**	**43**	**44**	**61**	**47**	**65**	**62**
Model 1.1 (1 conv layer)	**0.418**	**61%** epoch 10	**70**	**76**	**54**	**48**	**46**	**46**	**49**	**61**	**75**	**63**
Model 1.2 (2 conv layers)	**0.855**	**59%** epoch 15	**57**	**63**	**49**	**41**	**52**	**55**	**60**	**64**	**76**	**70**
Model 1.3 (3 conv layers)	**0.676**	**64%** epoch 14	**63**	**67**	**48**	**50**	**63**	**46**	**62**	**69**	**75**	**72**
Model 1.4 (4 conv layers)	**1.21**	**55%** epoch 10	**64**	**82**	**45**	**40**	**36**	**52**	**43**	**57**	**67**	**57**

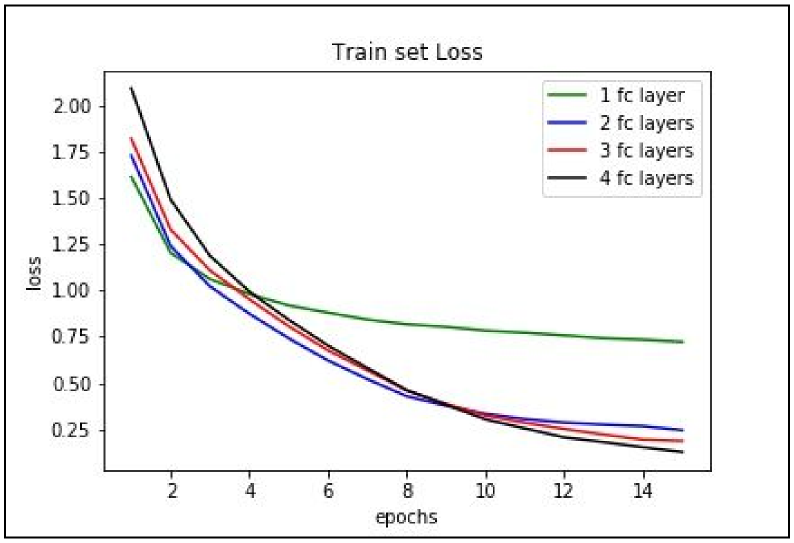
Table Inference

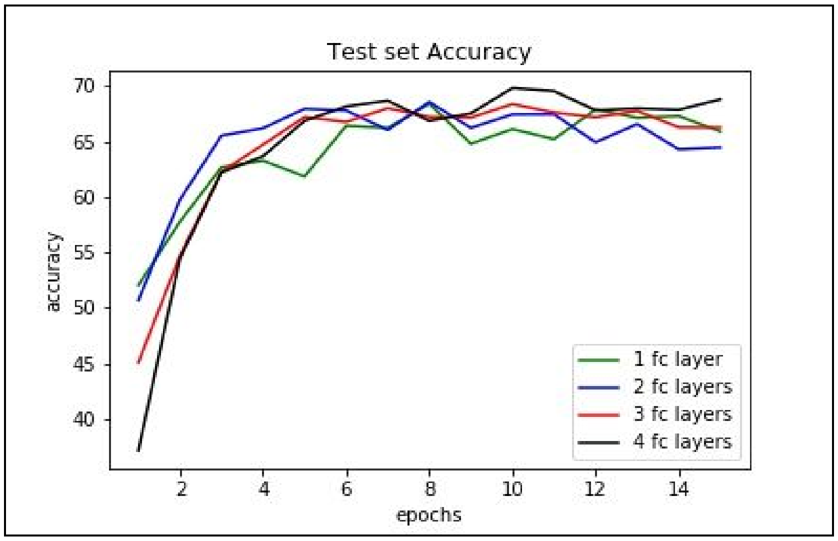
- Minimum loss comes for Model 1.1 -0.418. But maximum accuracy comes for model 1.3- 64%. This might be happening because Model 1.1 maybe overfitting on the train set data.
- Model 1.4 is giving high loss and low accuracy which makes us conclude that having higher conv layers can have a negative impact as well.
- No common trends for class wise accuracies can be seen. But for model 1.4 class car has way higher accuracy than the rest. This shows that the model may be overfitting on the car class predicting several cars and non - cars as cars leading to lower overall accuracy. So it is not good to add more conv layers.
- I would be using **Model 1.3** for subsequent sections because it gives a higher test accuracy as well as a decent training loss.

1.2) Number of Fully-connected layers

The following plots show the variation of loss, as well as test accuracy with rest to epochs for different models with a different number of FC layers, used ranging from 1-4. I have used 3 Conv layers for all these models as it gave the best performance on out of the ones I trained in the previous section.

Plots





Plot Inferences

- Minimum final loss while using 4 FC layers and maximum while using 1 FC layer. We are able to see a common trend that the final loss decreases when the number of FC layers used increases. This is because as the number of layers increases the number of parameters in the model also increases which helps in better fitting the training data and thus reducing the loss.
- The variation of loss with respect to epochs is a decreasing function with the slope negative in all cases while for test accuracy it is increasing as well as decreasing with the number of epochs.
- We are getting a maximum test accuracy of 70% in the 10th epoch of the model with 4 FC layers. Even if the loss decreases we are getting a subsequent decrease in test accuracy. This is because the model may be overfitting on the training data with epochs which will decrease the loss but may not give better results on test data. It can even worsen the test accuracy.
- We can also see that from epoch 6 all these models will be having almost the same test accuracy varying from 65% to 70% without any visible trend. This show decrease in loss does not guarantee an increase in test accuracy.

Observation Table

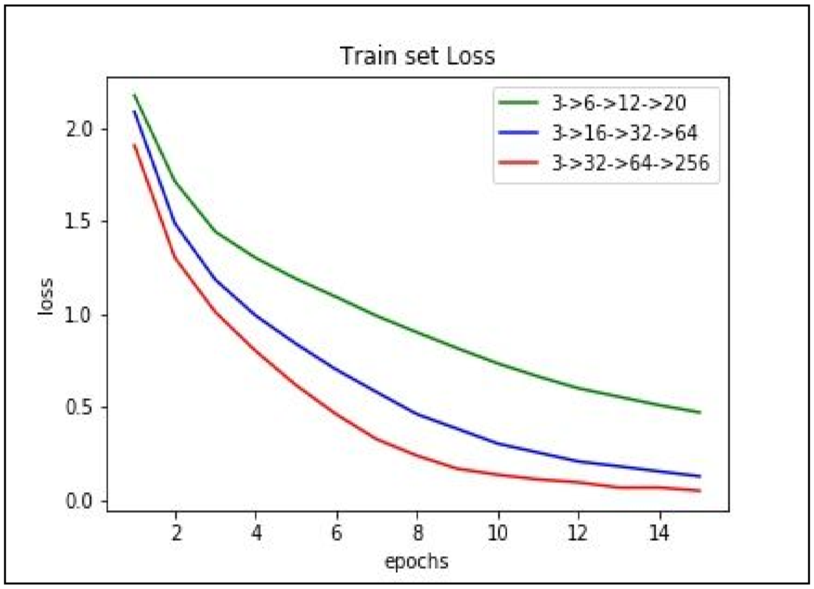
Models	Final loss	Best test accuracy	Class wise accuracy									
			Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Model 2.1 (1 FC layer)	**0.721**	**68%** epoch 8	**69**	**73**	**61**	**42**	**53**	**51**	**83**	**75**	**73**	**74**
Model 2.2 (2 FC layers)	**0.245**	**68%** epoch 8	**70**	**86**	**55**	**38**	**57**	**54**	**63**	**71**	**80**	**66**
Model 2.3 (3 FC layers)	**0.186**	**68%** epoch 10	**71**	**82**	**48**	**57**	**59**	**60**	**68**	**61**	**80**	**73**
Model 2.4 (4 FC layers)	**0.152**	**70%** epoch 10	**72**	**82**	**56**	**51**	**64**	**60**	**72**	**73**	**78**	**77**

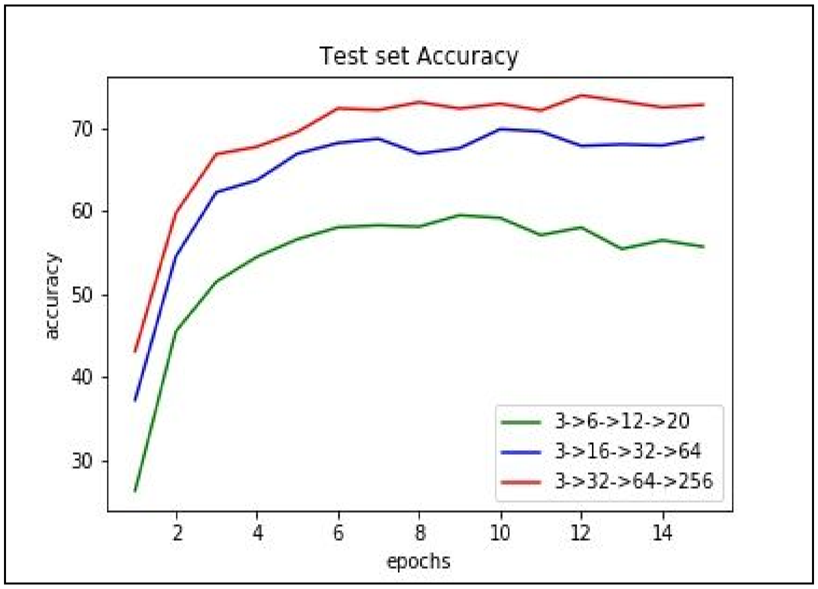
Table Inference

- Model 2.1 with 1 FC layer gives us 68% training accuracy on epoch 8 while training Model 2.4 with 4 FC layers and 10 epochs gives us 70% accuracy. But comparing models 2.1 and 2.4 in training results we can see that the latter gives way less loss (0.152) than the earlier one (0.721). So we would not be able to conclude this as the best model. What we need actually is a model that reduces the training loss and also brings about a subsequent higher increase in test accuracy. Model 2.4 may be more overfitting to the training set that Model 2.1.
- Classwise accuracy: Class wise accuracy data gives us an idea of which class is difficult to predict and which ones are easier. From the table, we can see that car, ship, truck classes are easier to predict while bird, dog, cat, deer classes are difficult to predict. This is because the latter involves more detailing than the previous ones.
- Considering test accuracy I think ***Model 2.4*** is the best and would be using it for further sections.

1.3) Number of Filters in each Layer

For this, I would be using Model 2.4 3 Conv layers and 4 fc layers and changing the out_channels in each of the conv layers without changing any other parameters. The plot given below shows variation in loss and accuracy with respect to epochs for different models. In the plot the depth of the channel after each layer is given as 3->(channel depth after conv 1)->....





Plot Inference

- We are able to see a common trend here that as the number of channels in each layer increases the loss decreases. This is because of the increase in network parameters that helps in the better fitting of the train data than one with fewer channels. Also, the losses decrease monotonically with epochs.
- The test set accuracy can also be found out to be increasing as channels increase. This is because adding more channels enables more features to be learned by the model which helps in better identification of the class of the objects. Not always increasing -better accuracy at intermediate epochs.
- But one of the major drawbacks in increasing the channel size is the time it takes for the training process. Adding more channels increases the number of parameters and may increase the training time by several factors.

Observation table

Models	Final loss	Best test accuracy	Class wise accuracy									
			Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Model 3.1 (3->6->12->20)	**0.470**	**56%** epoch 9	**66**	**82**	**51**	**42**	**47**	**41**	**63**	**57**	**59**	**45**
Model 3.2 (3->16->32->64)	**0.152**	**69%** epoch 10	**72**	**82**	**56**	**51**	**64**	**60**	**72**	**73**	**78**	**77**
Model 3.3 (3->32->64->256)	**0.048**	**74%** epoch 12	**74**	**88**	**65**	**55**	**76***	**53**	**80**	**68**	**83**	**81**

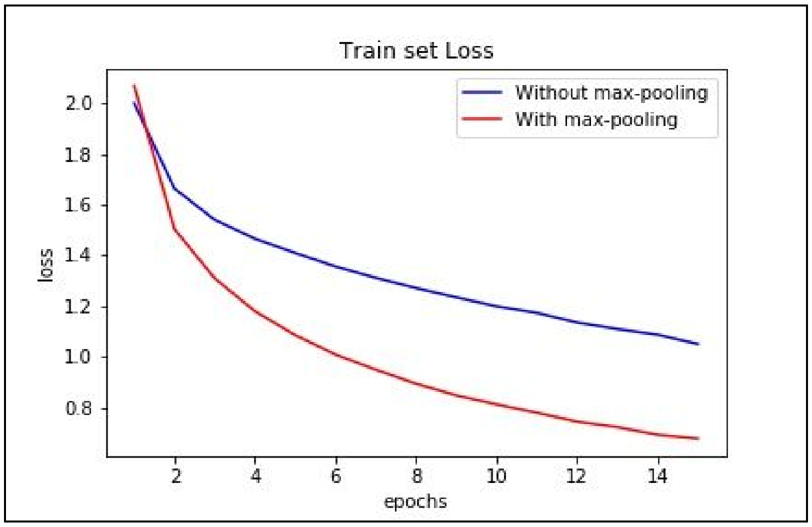
Table Inference

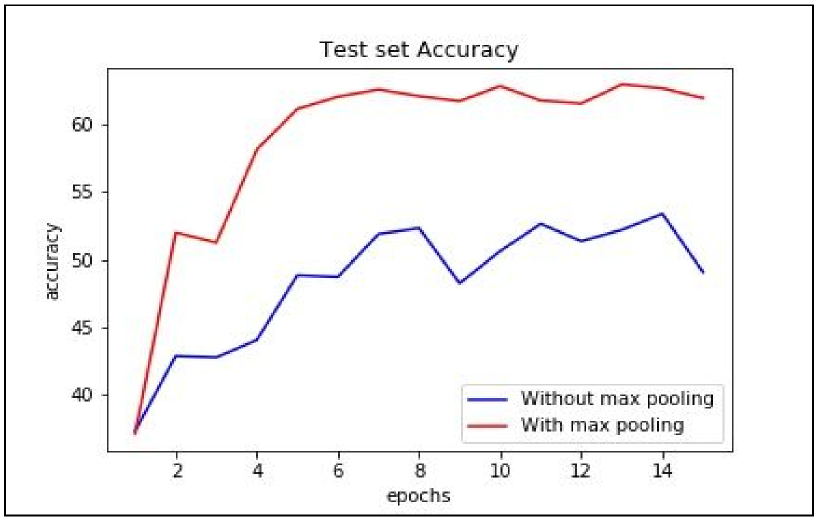
- Minimum test loss for Model 3.3 - 0.048 which is almost one-third of Model 3.2. We can see a clear decrease in final losses with the number of channels used.
- A clear trend in the best test accuracy can also be seen with a maximum test accuracy of 74% for model 3.3 at epoch 12. The subsequent decrease in test accuracy for epoch 13, 14, 15 maybe because of the overfitting of the model in the training set.
- Determining the best model out of these depends on several factors. Model 3.3 takes a huge time to train and if we want to train a model fast I would suggest going with Model 3.2 which gives a descent loss as well as accuracy. But if we want to give a higher focus on accuracy on test set go with Model 3.3 which gives the max test accuracy.
- Class wise accuracy: We can see a clear increase in accuracy for the classes with models except for dog and horse where it decreases from second to the third model. The anomaly is highlighted. I think this might be because of the similarity in features of dogs and horses which can lead to the wrong classification. In order to solve this, we should focus on the features that are exclusive for each of them and train models with these features.

1.4) With or Without MaxPooling

I took Model 1.3 and retrained it, this time without the max-pooling layers.

Plots





Observation table

Models	Final loss	Best test accuracy	Class wise accuracy									
			Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Model 4.1 (With max-pool)	**0.676**	**64%** epoch 14	**63**	**67**	**48**	**50**	**63**	**46**	**62**	**69**	**75**	**72**
Model 3.2 (Without max_pool)	**1.05**	**53%** epoch 14	**44**	**66**	**34**	**52**	**28**	**26**	**49**	**62**	**76**	**49**

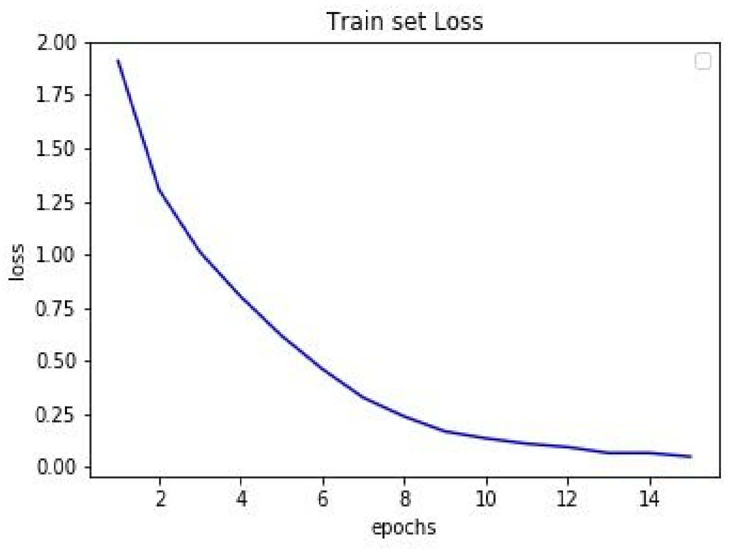
Plot and Table inference

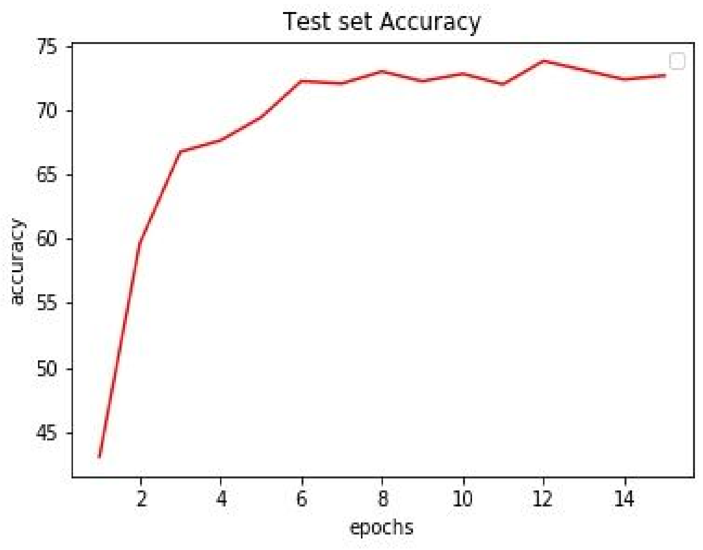
- Better training set performance- with max-pooling(loss=0.676).
- Better test set performance-with max-pooling.(acc=64%)
- Loss in decreasing with epoch while accuracy doesn't increase monotonically.
- In class wise test results for classes car, cat, ship is almost the same for both models. For car and ship classes maybe sufficient parameters are provided by moth models and for cat class insufficient parameters may be provided by max-pooling for further training.
- Max-pooling provides an additional layer and extra parameters that can improve the results in the train as well as test data.

1.5) Variation with epochs

The best model I have trained so far is Model 3.3 with 3 conv layers and 4 fc layers and (3->32->64->256) out channels in each conv layer. I would be using this model to show the relation between epochs and loss/accuracy.

Plot





Inferences

- Loss vs epoch is a monotonically decreasing function.
- Accuracy vs epoch is not monotonically increasing. Here the function is decreasing from 12 to 14. This is due to the overfitting of the model in train data.
- Minimum loss= 0.0487 at epoch 15.
- Maximum accuracy- 74% at epoch 12.
- The decrease in loss is very high in the first epochs and it decreases gradually in the final epochs-moves to a saturation value like a logarithmic graph. Initial slope highly negative and it increases to 0 with epochs.
- Similarly, the increase in loss is very high in the first epochs and it decreases gradually in the final epochs-moves to a saturation value like a logarithmic graph. The initial slope highly positive and it decreases to 0 with epochs.
- Here it is better to use the model at epoch 12 as it gives the highest test accuracy. It also gives a decent training loss.

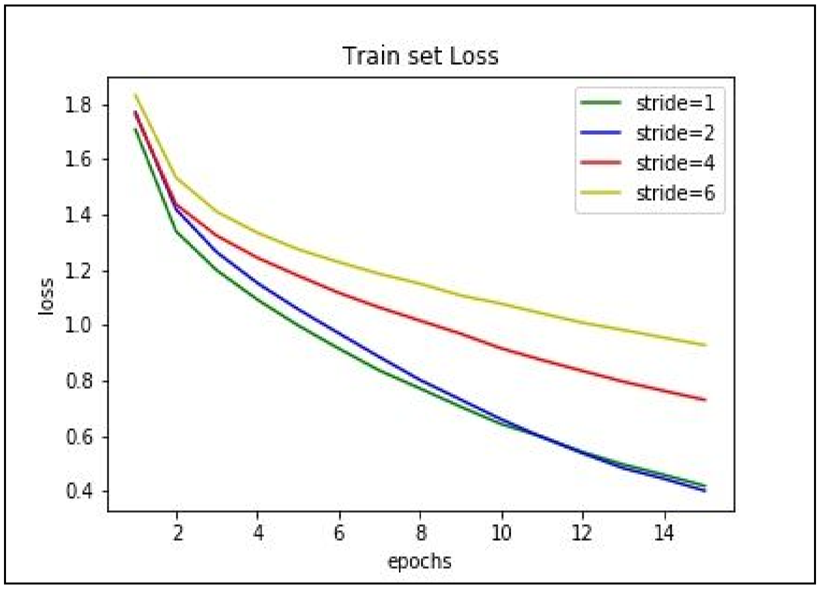
1.6) Variation with strides

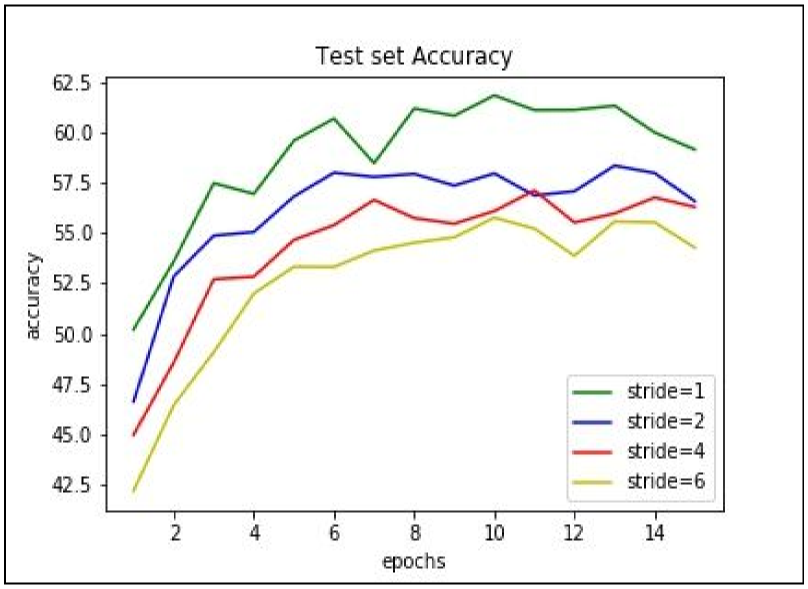
For testing the effect of strides on training loss and test accuracy a basic network is trained.

image--->(1conv layer without max pool)---->(4 FC layers)

Relu is applied to each layer except the last one. Since only one conv layer is used, we could change its number of strides in order to study its effect on the training and test dataset.

Plots





Plots inference

- It can be easily seen that the loss increases with an increase in the number of strides. This is because as the number of strides increases the filters are skipping over several pixel values of images thus failing to find some functions between the input and output. This skipping decreases the parameters which decrease the extent to which the model can be fit on the training set, thus increasing the final loss.
- Similar to the loss function, the model performs best on the test set too when the number of strides is kept minimum. The same reason applies here too.
- A common trend can be seen for both the test as well as the train set.

Observation table

Models	Final loss	Best test accuracy	Class wise accuracy										Iterations per second
			Plan e	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	
Model 6.1 (strides=1)	**0.419**	**62%** epoch 10	**70**	**62**	**47**	**33**	**50**	**36**	**69**	**61**	**73**	**64**	**75**
Model 6.2 (strides=2)	**0.421**	**57%** epoch 13	**61**	**67**	**35**	**26**	**53**	**55**	**71**	**66**	**68**	**61**	**150**
Model 6.3 (strides=4)	**0.72**	**56%** epoch 11	**72**	**65**	**37**	**36**	**44**	**53**	**64**	**69**	**56**	**63**	**180**
Model 6.4 (strides=6)	**0.927**	**55%** epoch 13	**67**	**61**	**45**	**28**	**46**	**54**	**57**	**55**	**65**	**61**	**200**

Table inference

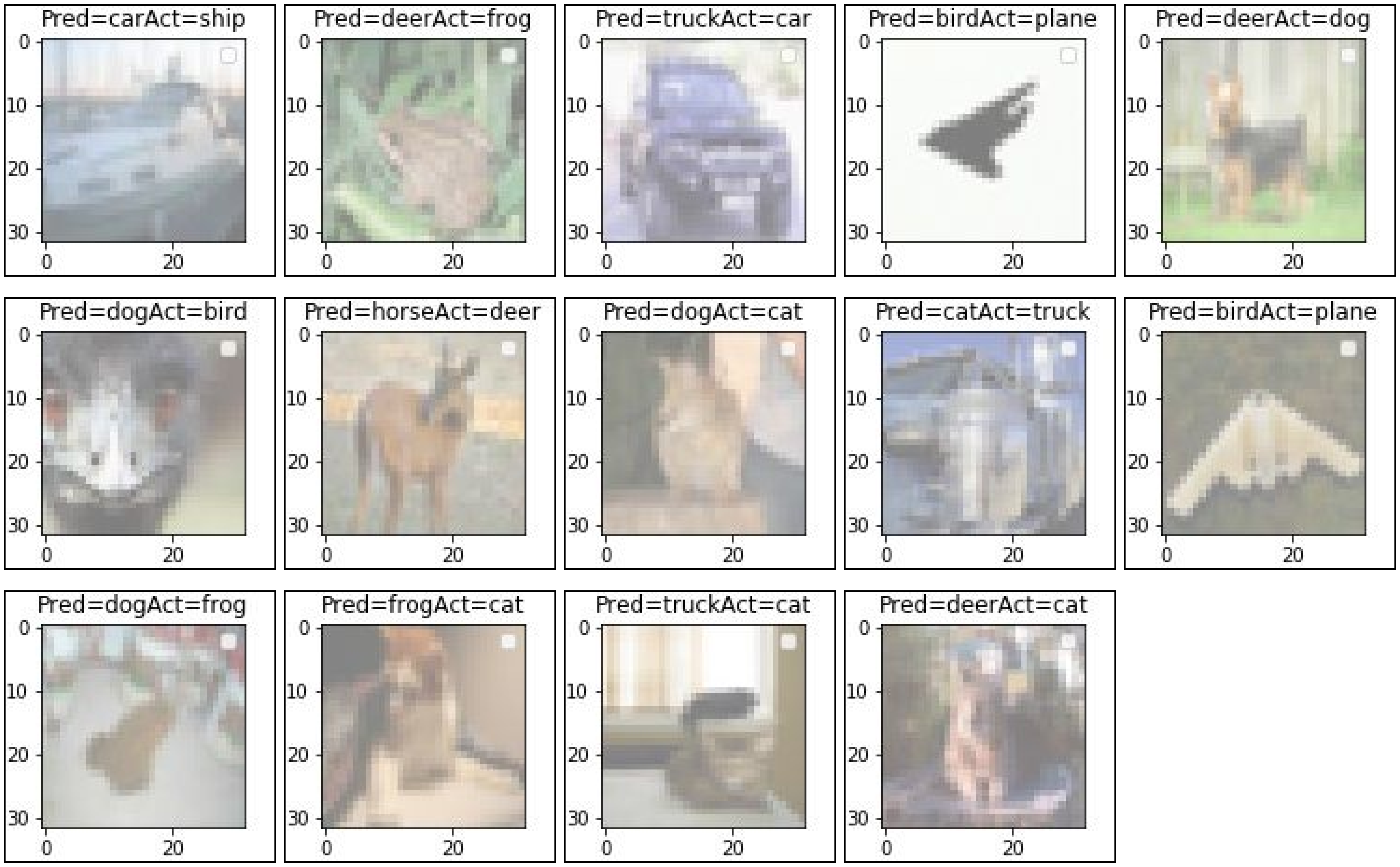
- Minimum loss - for model 6.1 -0.419
- Maximum accuracy - for model 6.1 - 62%
- Class wise accuracy- showing inconsistent behavior. The maximum of each class accuracies is highlighted. We can see for different classes max accuracies coming for different strides used- 3 for stride=1, 4 for stride=2 and 3 for stride=4 while 0 for stride=6. This is determined by the features trained by a model in which the features of which class can be better expressed by a model.
- The main benefits of stride come in saving time for training a model. The iterations per second increase with great fold with an increase in strides thus making the time for training less. This is because strides ignore several parameters by making the filters jump between points in an image. Reducing parameters result in reducing time. Strides come in handy for large pixel-sized images which would be having greater amounts of detail which are not required and so some values can be skipped while raining with the help of strides. This makes the training of these models faster without much drop in test accuracy.

1.7) Wrongly Labeled images

I trained Model 3.3(3 conv layers and 4 fc layers) again and this time I got a maximum test accuracy of 73.04 at the final epoch. I would be using this model for the upcoming analysis. Class-wise accuracy as follows:

plane	car	bird	cat	deer	dog	frog	horse	ship	truck
74%	87%	55%	48%	71%	67%	80%	82%	78%	82%

Accuracy of 73.04 means out of 10000 test images 7304 are correctly labeled while the remaining 2696 are wrongly labeled. Some of the wrongly labeled images are shown below:

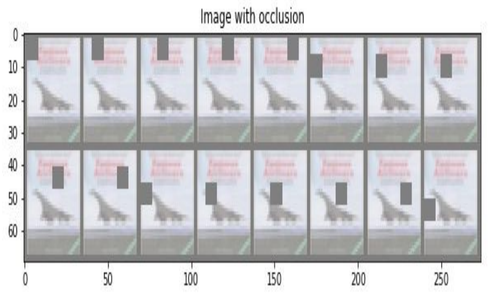


- One common misclassification that can be observed is a different animal is predicted for a given animal picture such as, deer predicted instead of a dog, deer instead of cat, dog instead of bird, dog instead of a cat, horse for deer and so on. This error can be minimized by training with some features that are exclusive for each animal.
- Another common misclassification is between birds and planes. The sky actually plays an important role in detecting bird/plane which we can see from the occlusion test below. Since the sky would be common to most pictures of planes and birds and both having similar shapes can create a classification problem.
- Even I find it difficult in recognizing some pictures because of the low pixel size. So having more clearer images could help in providing a better accuracy rate.

Part-B

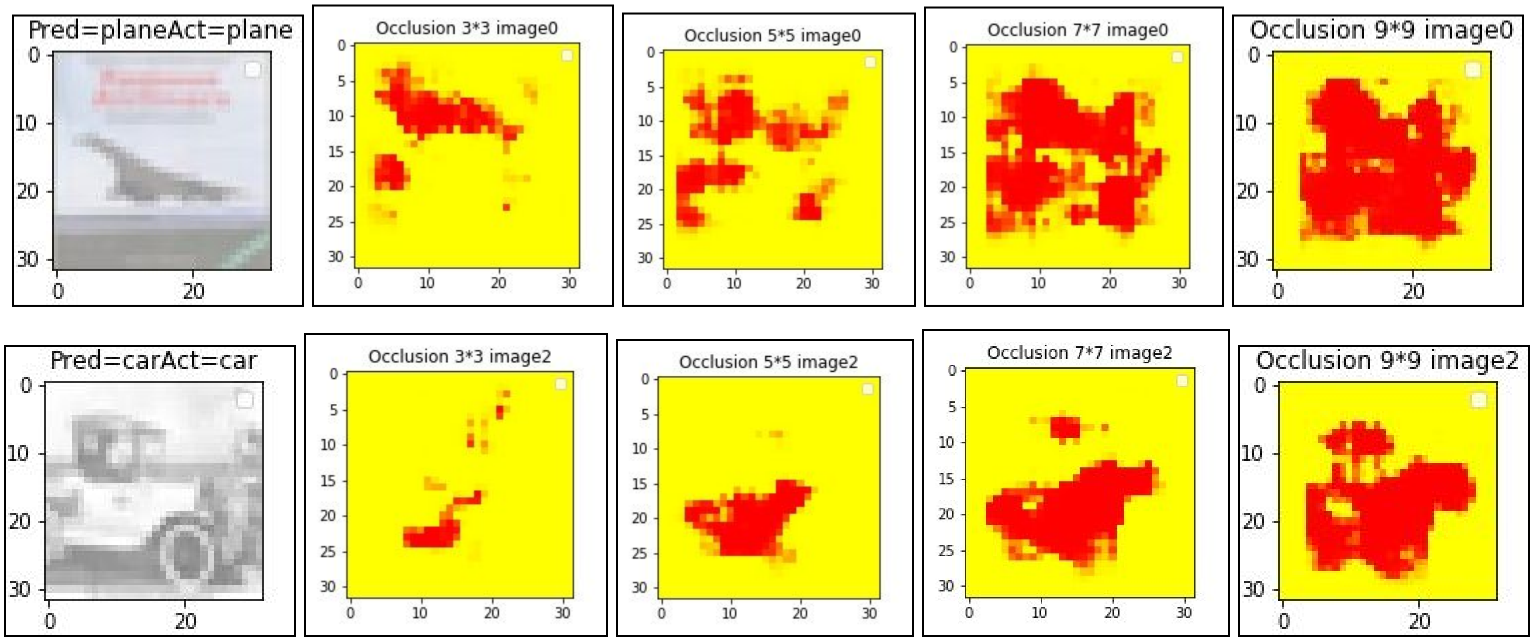
2.1) Occlusion sensititvity experiment

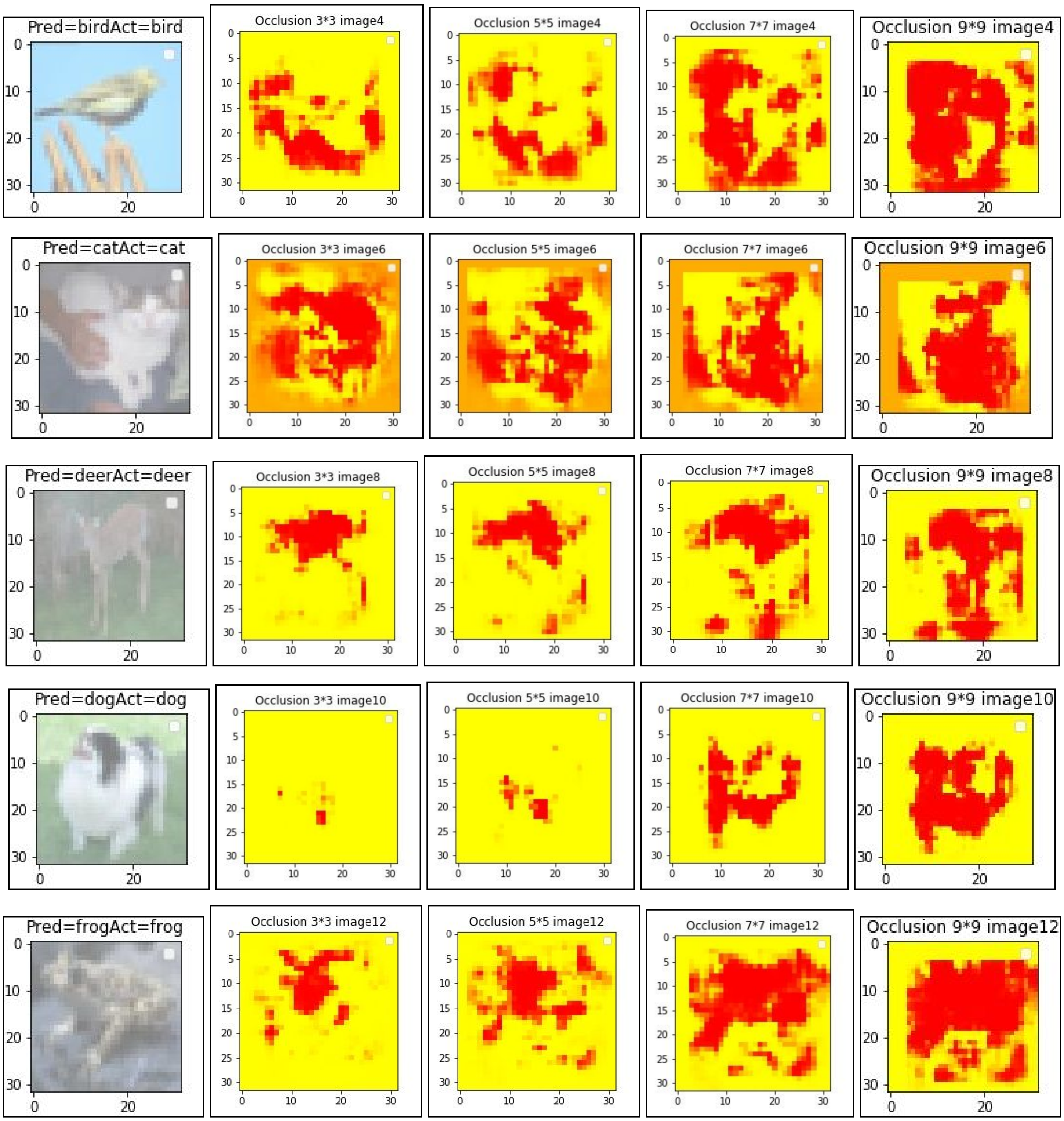
- 20 correctly labeled images, 2 from each class are taken. These are tested using the above model and passed to a softmax output layer.
- Now windows of sizes 3*3,5*5,7*7,9*9 pixels are taken. These are greyscaled (ie pixel values made 0) and each of these windows is passed throughout the image. See image---> The grey color square boxes are the windows.

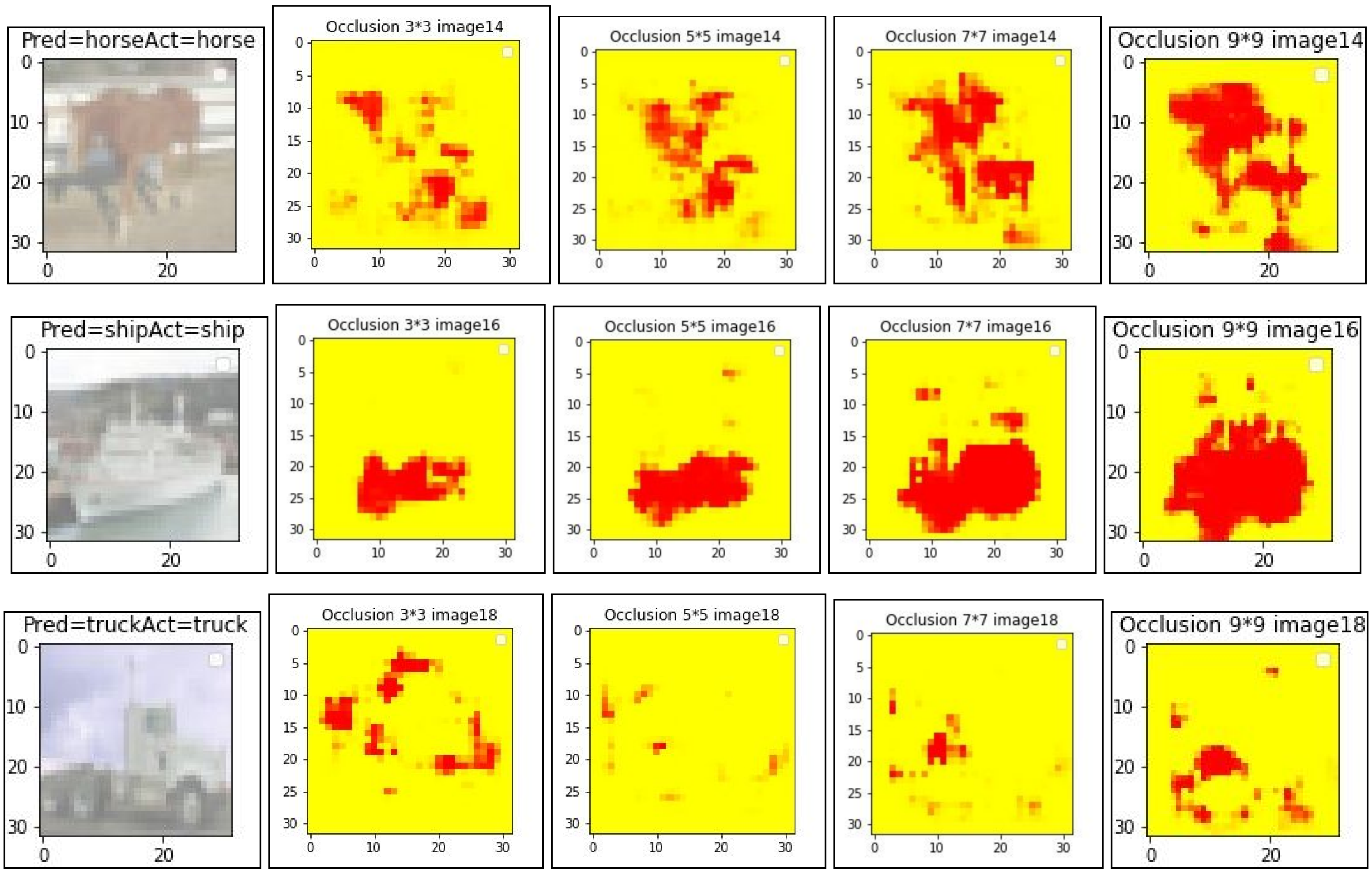


- This changes the softmax prediction of each class and it is plotted as a function of space as shown below. Colormap used here is 'autumn' where 'yellow' represents values close to 1 and 'red' represents values close to 0. It means the regions where red color comes are giving low softmax output values. This means these are the responsible regions of why the image is predicting the correct class. The color bar for autumn from 0 to 1 is shown here.

- These occlusion images also help us in understanding whether the model has really learned the location of the object in the image or if the model just classifies the image based on surrounding or contextual cues.
- Out of the 20 images tested 10 (1 from each class) images with their occlusion plots are given below.




Pred=birdAct=bird
Occlusion 3*3 image4
Occlusion 5*5 image4
Occlusion 7*7 image4
Occlusion 9*9 image4
Pred=catAct=cat
Occlusion 3*3 image6
Occlusion 5*5 image6
Occlusion 7*7 image6
Occlusion 9*9 image6
Pred=deerAct=deer
Occlusion 3*3 image8
Occlusion 5*5 image8
Occlusion 7*7 image8
Occlusion 9*9 image8
Pred=dogAct=dog
Occlusion 3*3 image10
Occlusion 5*5 image10
Occlusion 7*7 image10
Occlusion 9*9 image10
Pred=frogAct=frog
Occlusion 3*3 image12
Occlusion 5*5 image12
Occlusion 7*7 image12
Occlusion 9*9 image12



Occlusion Inferences

- The region of red patches increases as the window size increases. This is because as the window covering the image becomes larger more part of the picture is unknown and becomes difficult to predict, thus leading to lower softmax outputs which cause an increase in red patches.
- Comparing the red patches with the image we can see that not for all the cases the output depends only on the object. In some cases, it depends on the environment or background too, eg: in the case of image 0, it depends on the sky above the plane too.
- For car, deer, dog, frog, horse, ship - red patches in area of the object which shows the location of the object is understood.
- For truck, plane, bird, cat some background are also coming as red patches which shows that the model hasn't really learned the location of these objects and some background factors are also taken into consideration for classification.

*Codes for occlusion test given in the assignment folder.

2.2) Filter analysis

2.2.1) *Filter identification*

This test is for determining what each filter does when the image passed through it. This test would be performed on 10 filters. First three filters from layer 1, first 3 filters from Layer 2, first 4 filters from Layer3.

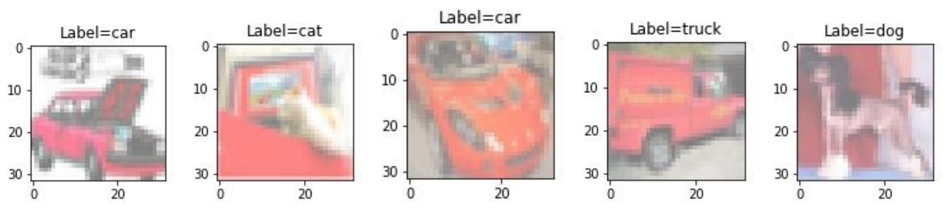
Procedure

- Initially, for testing with the filter of a given layer, all other filters of that layer are switched off by making their corresponding weights 0. The filters in the remaining layers are left unchanged. For example to make weights of all filters except filter 1 on 1st layer the following line of code is required:

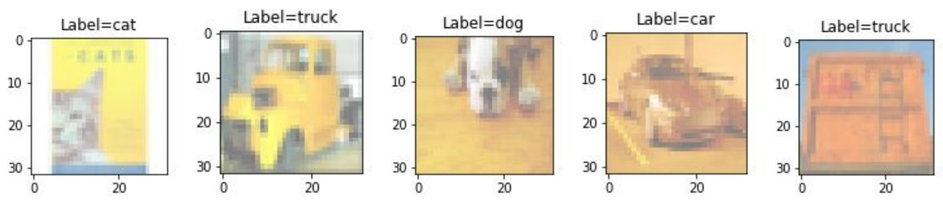
```
net.conv1.weight.data[1:32,:,:,:]=0
```

- The model class is changed in such a way that an intermediate output is given after the first conv layer which is a tensor. The overall sum of this tensor is taken.
- Each image is passed through the changed model and the sum of the elements of intermediate output is recorded. From these 100 images giving the maximum sum is separated.
- Images giving the maximum sum for each filter are saved.
- Classwise analysis of the 100 images separated is done and given in the table below.
- This is done for all the 10 filters separately. A combined test for 3,3,4 filters is done for layer 1,2,3 respectively. That is, for layer 1 first 3 filters are kept open and remaining is closed and the same processes as given above are done.

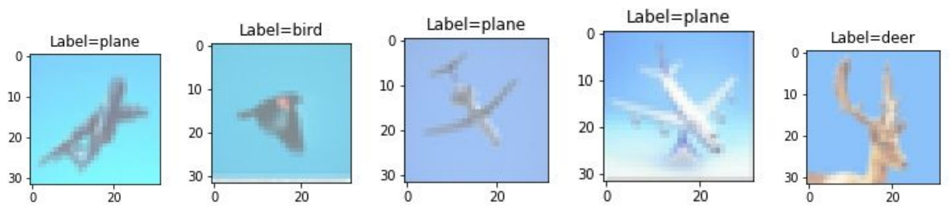
Images giving maximum response to each of the 10 filters are given in the assignment folder. For the three filters of layer 1 these images are given below:



Layer 1 Filter 1 max response images



Layer 1 Filter 2 max response images



Layer 1 Filter3 max response images

Classwise analysis of 100 images giving the maximum response to each of the 10 filters is shown below (F corresponds to filter number, L corresponds to layer number, ALL corresponds to taking all filters in a layer together-combined). The class showing the high response for each filter is highlighted.

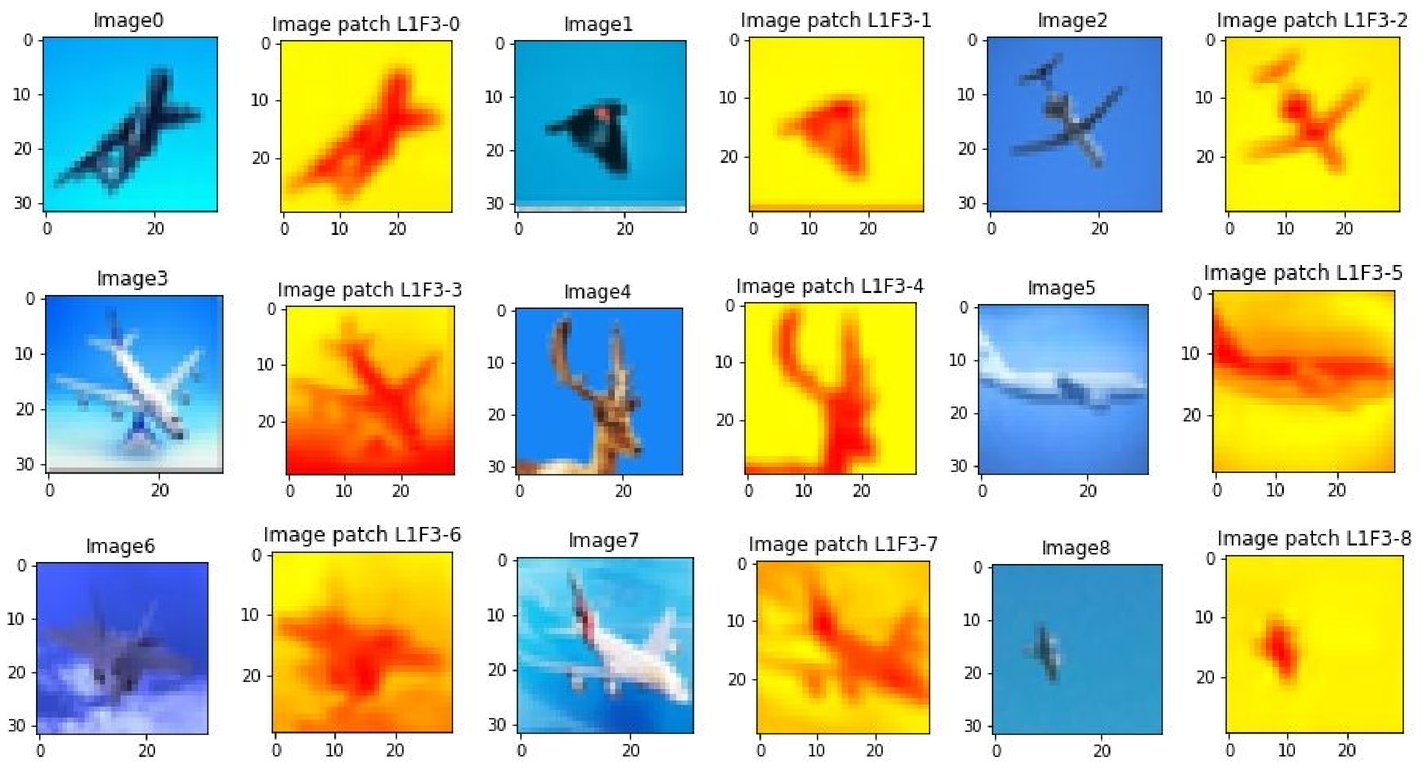
Class	***plane***	***car***	***bird***	***cat***	***deer***	***dog***	***frog***	***horse***	***ship***	***truck***
L1F1	3	35	1	13	7	16	9	2	1	13
L1F2	4	6	12	9	23	12	23	6	0	5
L1F3	70	1	12	0	1	2	0	0	14	0
L1ALL	77	1	10	0	1	0	0	0	10	1
L2F1	2	61	0	3	0	5	1	0	8	20
L2F2	8	14	4	6	4	14	0	21	2	27
L2F3	9	0	32	6	25	10	8	8	1	1
L2ALL	8	32	10	5	1	12	1	9	8	13
L3F1	26	2	19	9	8	4	3	24	1	4
L3F2	8	16	14	11	2	13	14	4	9	9
L3F3	43	7	12	1	6	2	0	3	21	5
L3F4	21	12	11	14	1	15	0	7	2	17
L3ALL	45	2	19	5	7	0	4	11	3	4

Inference:

- Some filters like L1F1 and L2F2 doesn't have a clear favorite. It most of its classes showing value from 15 to 25 percent. What this means is that these filters would be a feature that is common to many of the classes and thus multiple classes are showing higher responses.
- Some filters on the other hand like L1F3 and L2F1 has clear favorites, planes, and car respectively. This means the feature represented by this L1F3 is one that can be seen in planes and none of the other classes leading to maximum response for most of the plane images in this case.
- If we check the overall table we can see that the **plane** is the class showing better responses for most of these filters and the class **frog/ship** shows the least response. This implies that these filters are very important in identifying a plane as a plane but not at all important for identifying a frog as a frog and ship as a ship.

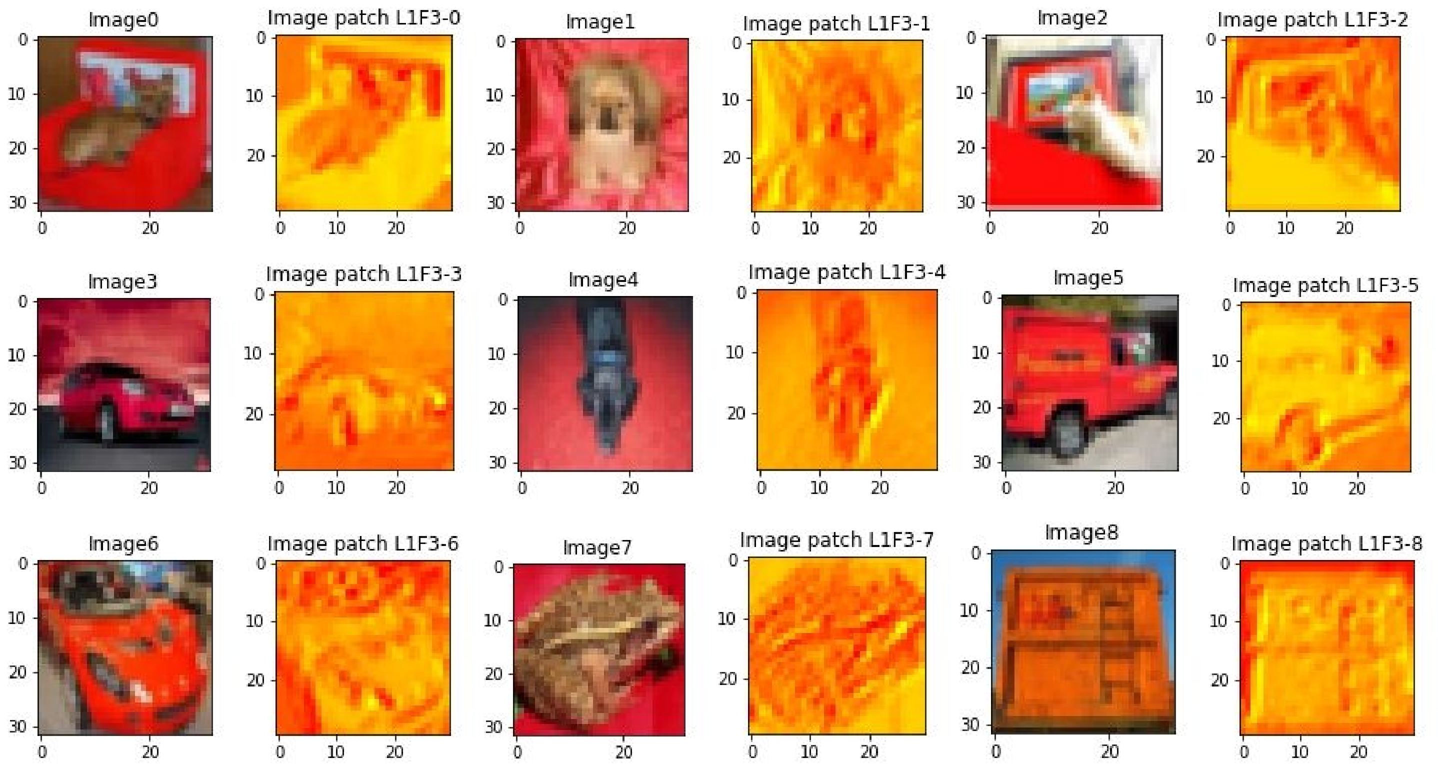
Finding image patches

To find image patches corresponding to the maximum response position in the filter map I am gonna take the filter L1F3 as it gives a better classwise response and apply the filter across the image just like occlusion was did. The 10 images giving the maximum response to this filter are taken and are passed through the model. The intermediate tensor after 1st convolution is taken. It would be having dimensions of (1,32,30,30) since we used out_channel_size=32 and filter height and width to be 3. This tensor is sliced as tensor[0,2] as we need the values of the second filter, converted to numpy. Now it is a numpy array of dimensions (30,30). This array is plotted using plot.imshow which gives the image patches of each image just like we did in the occlusion test. (of these images and their image patches for filter 3 of conv1 are given below.



Inferences

- From the above patches, we can find out which parts of the image are giving low values when the filter is passed above it. For most of the images, the image patches are coming as the shape and location of the object. If we check image 3 and its corresponding patch we can see an extra part where the response is different. This is actually a white part of the sky. Thus from examining all these image patches we can conclude that this filter is detecting a **blue background** for which it gives the maximum response which is shown in yellow color.
- Similarly, the Layer 1 filter 1 patches are given below for images showing max outputs after the 1st conv layer. By clear inspection, we can see that this filter is showing the maximum response for a red background.
- Similarly, each filter would be showing max response to a certain feature. Higher the number of filters more the number of features which can be represented.



*Codes for filter identification are given in assignment folder.

2.2.2) **Filter Modification**

In our Model 3.3, we had 3 conv layers. Now I switched of these 10 filters by making their corresponding weights 0. The filter numbers from each layer are:

1st conv layer- filters 1,2,3

2nd conv layer- filters 1,2,3

3rd conv layer- filters 1,2,3,4

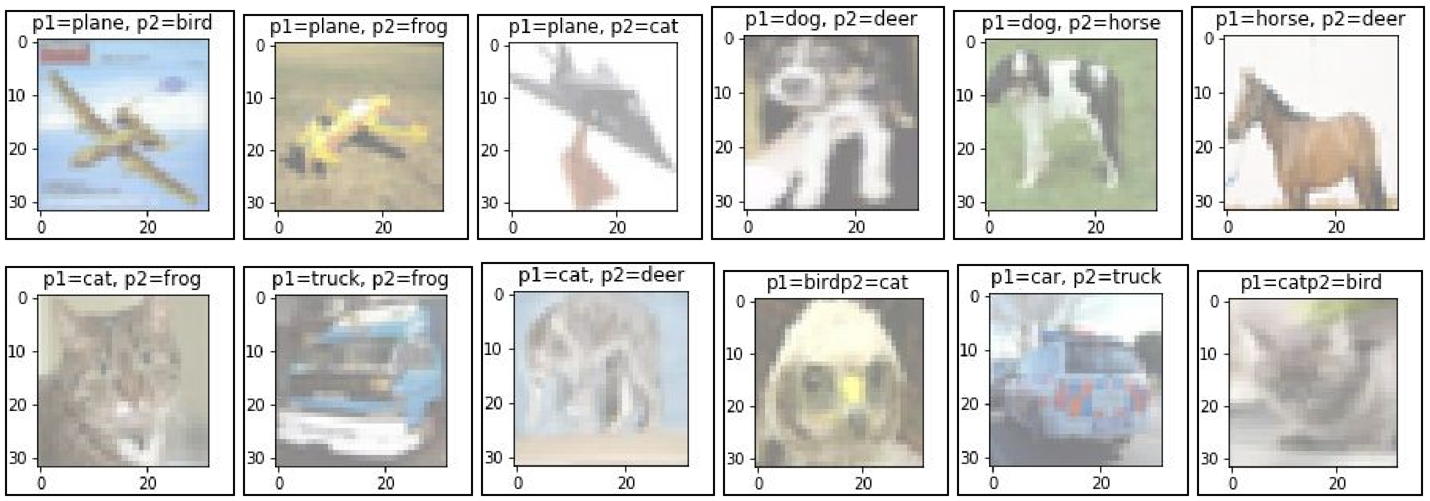
Now I tested the model on the test data with the modified weights. Initial accuracy 73.04. New accuracy with modified weights 72.30. Let these tests be Test1 and Test2 respectively.

Test 1 - 7304 /10000 images correctly predicted

Test 2 -7230 /10000 images correctly predicted (74 images less)

1	Correct both Test 1 and Test 2	7046
2	Wrong both Test 1 and Test 2	2512
3	Correct in Test 1 Wrong in Test 2	258
4	Wrong in test 1 Correct in Test 2	184
	Total	**10000**

Some of the images that are correctly predicted in Test 1 but wrong in Test 2 are given below: (p1 is prediction in test 1, p2 is prediction in test 2). Rest is given in the assignment folder



Class-wise differences in Test 1 and Test 2 are given in the table below:

Class	***Test 1 correct***	***1-correct 2-wrong***	***1-wrong 2- correct***	***Test 2 correct***	***Difference***	***Percentage change***
plane	748	45	10	713	-35	-3.5
car	873	24	8	857	-16	-1.6
bird	554	17	33	570	16	1.6
cat	488	31	31	488	0	0
deer	716	13	28	731	15	1.5
dog	676	48	17	645	-31	-3.1
frog	808	14	21	815	7	0.7
horse	825	19	10	816	-9	-0.9
ship	788	25	11	774	-14	-1.4
truck	828	22	15	821	-7	-0.7
Total	**7304**	**258**	**184**	**7230**	**-74**	**-0.74**

Inferences

- Switching off filters has an overall negative impact while testing. For 6 of the 10 classes, we see the testing accuracy decreasing, for 3 classes increasing and for 1 class it remains the same. The overall difference of -0.74% (74 images).
- The class-wise test accuracy decreases most for the **plane, dog** by 3.5% and 3.1 percent respectively (red in the table). It means that it is these classes were the ones who maximum depended on the removed filters. When these filters were removed 45 and 48 images of these respectively which gave correct predictions in the first test are giving wrong predictions now.
- On the other hand, the class-wise accuracy increases for the classes **bird, deer, frog** (blue in the table). These would be the classes least depending on the removed filters. And thus removal of those filters have benefitted them. So it might have been a feature favoring the former rather than the latter.
- For example, if there is an object in the sky, the feature would be favoring plane over birds. But now since the feature is gone planes would not be getting the same preference and so its accuracy goes down and the accuracy of birds, on the other hand, goes up. The above pictures give a rough idea of this- the first picture above is a plane correctly predicted in Test 1 but is being predicted bird in Test 2.
- Also in the identification test, we did above we can see that the maximum response of the filters in a whole was produced by planes which are now the most affected class when I removed those filters. This verifies the inference from the identification test that these filters are very much responsible for identifying a plane as a plane.

Results And Conclusions

- Was able to design a good model which gave a test accuracy of up to 74 % in the CIFAR 10 dataset.
- Was able to figure out how to design an efficient neural network.
- Did the occlusion test to find out our model is actually predicting the object by seeing them and not by the background.
- Did the filter tests which made me understand how each filter stands for a unique feature and having more filters means having more features that can be learned which can really boost the learning process.

Codes

Codes for doing all the sections are given in the programming assignment folder which I would submit with this report. Each codes are given as .py files.

- For Part A, **definitions.py---->realcode1.py---->realcode2.py.** Changes should be made in realcode1 and realcode2 depending on the parameters we are testing and we want to plot a graph of.
- For Part B, Occlusion **definitions.py---->occlusiontest1.py----->occlusiontest2.py.** Changes is made in occlustiontest2 depending on the window size we would be using.
- For filter identification, **definitions.py-->filteridentify1.py-->filteridentify2.py-->filteridentify3.py-->filteridentify4.py--->filteridentify5.py** . 1,2,3,4 for getting class analysis and 5 for getting image patches. Changes in the weights to be made 0 and where the output should be given from the model should be done in filteridentify2.
- For filter modification, **definitions.py-->filtertest1.py-->filtertest2.py-->filtertest3.py**

References

- Pytorch documentation
- Matplotlib documentation
- Visualizing and Understanding Convolutional Networks-Matthew D. Zeiler
- Used Pytorch boilerplate code given with the assignment
- Stackoverflow website for coding doubts

THANK YOU!!!!