Pattern Recognition and Machine Learning (276A)

Project 0: Convolutional Neural Networks

Results of Assignment:

1. Optimization of LeNet.

(a) Complete the functions flatten(), convnet init(), convnet forward().

The output of the fifth block is sent to the flatten() function. It gets the output of the convolutional layers which is a classifier and **flattens** all its structure to create a single long feature vector to be used by the dense layer for the final classification.

In the convnet_init function, the weights and biases for all the 5 hidden convolution layers are initialised with the respective filter sizes.

In the convent_forward function, the convolutions are defined along with the additional operations that are performed in the model such as pooling, applying activation functions (relu), padding etc. The weights that are updated with each learning epoch are also applied to the layers along with the biases so that the model can learn and improve its accuracy over time.

(b) Adjust parameters learning rate, epochs such that test-accuracy > 70%.

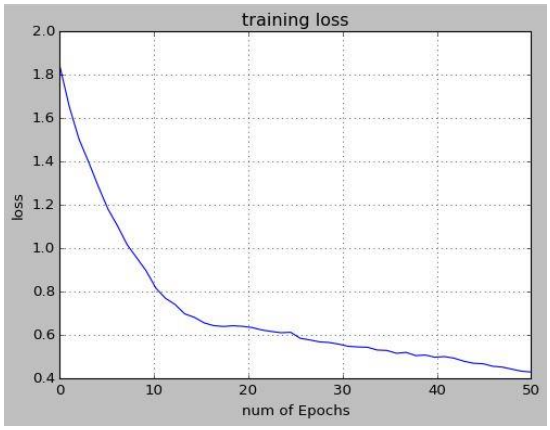
Learning rate= 0.01

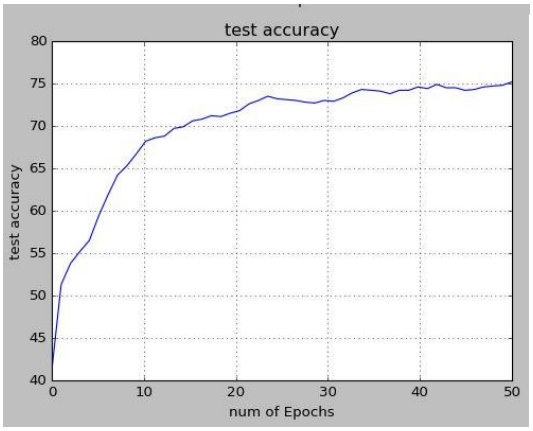
Epochs: 50 (starts stagnating after this with very little variation)

```
01:25:49 :    iteration =  300, loss =   0.5177, accuracy =    74.20%
01:25:50 :    iteration =  400, loss =   0.4196, accuracy =    73.90%
01:25:51 :    iteration =  500, loss =   0.4805, accuracy =    73.80%
01:25:52 :    iteration =  600, loss =   0.5977, accuracy =    74.70%
01:25:53 :    iteration =  700, loss =   0.4760, accuracy =    75.30%
01:25:53 : epoch   47 took 8.03s
01:25:53 : epoch   48/  50
01:25:53 :    iteration =    0, loss =   0.3602, accuracy =    74.70%
01:25:54 :    iteration =  100, loss =   0.3885, accuracy =    74.50%
01:25:56 :    iteration =  200, loss =   0.3220, accuracy =    73.30%
01:25:57 :    iteration =  300, loss =   0.5013, accuracy =    74.20%
01:25:58 :    iteration =  400, loss =   0.4024, accuracy =    73.90%
01:25:59 :    iteration =  500, loss =   0.4680, accuracy =    74.10%
01:26:00 :    iteration =  600, loss =   0.6012, accuracy =    74.60%
01:26:01 :    iteration =  700, loss =   0.4633, accuracy =    75.20%
01:26:01 : epoch   48 took 8.02s
01:26:01 : epoch   49/  50
01:26:01 :    iteration =    0, loss =   0.3433, accuracy =    74.80%
01:26:03 :    iteration =  100, loss =   0.3807, accuracy =    74.50%
01:26:04 :    iteration =  200, loss =   0.3141, accuracy =    73.80%
01:26:05 :    iteration =  300, loss =   0.4995, accuracy =    74.60%
01:26:06 :    iteration =  400, loss =   0.4053, accuracy =    73.90%
01:26:07 :    iteration =  500, loss =   0.4670, accuracy =    73.90%
01:26:08 :    iteration =  600, loss =   0.6044, accuracy =    74.70%
01:26:09 :    iteration =  700, loss =   0.4535, accuracy =    74.90%
01:26:09 : epoch   49 took 8.03s
```

Training the model with learning rate 0.01 over 50 epochs.

(c) Plot the training loss and test accuracy over epochs in two Figures.





2. Alteration of LeNet.

(a) Keep the Block5, learn a ConvNet only with:

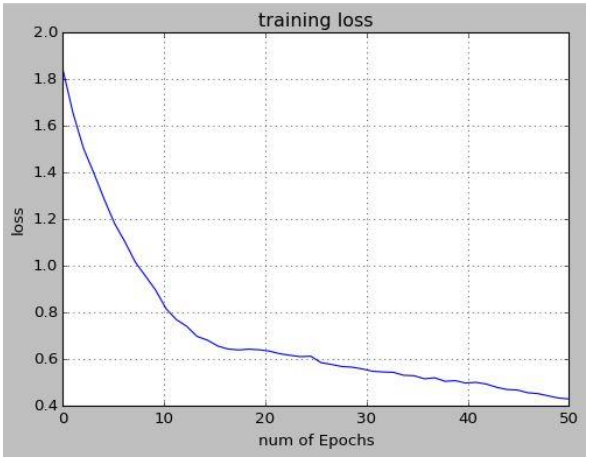
i. Block1.

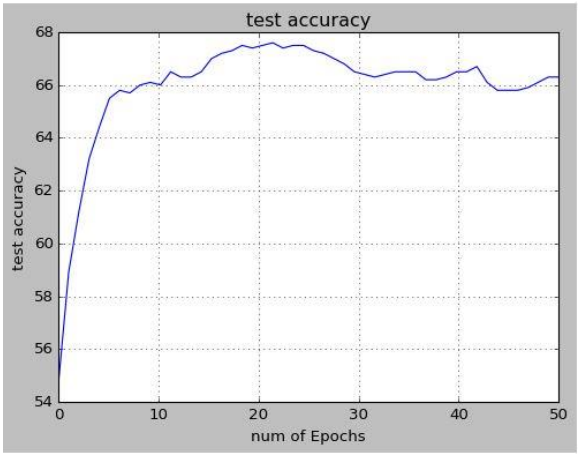
```
07:16:19 : epoch   48/  50
07:16:19 :    iteration =    0, loss =   0.2599, accuracy =    65.20%
07:16:19 :    iteration =  100, loss =   0.3365, accuracy =    66.60%
07:16:20 :    iteration =  200, loss =   0.2889, accuracy =    67.20%
07:16:20 :    iteration =  300, loss =   0.6062, accuracy =    62.10%
07:16:21 :    iteration =  400, loss =   0.2932, accuracy =    66.50%
07:16:21 :    iteration =  500, loss =   0.6454, accuracy =    62.50%
07:16:22 :    iteration =  600, loss =   0.3983, accuracy =    66.90%
07:16:23 :    iteration =  700, loss =   0.5107, accuracy =    66.30%
07:16:23 : epoch   48 took 3.90s
07:16:23 : epoch   49/  50
07:16:23 :    iteration =    0, loss =   0.2563, accuracy =    65.50%
07:16:23 :    iteration =  100, loss =   0.3319, accuracy =    66.50%
07:16:24 :    iteration =  200, loss =   0.2830, accuracy =    67.30%
07:16:24 :    iteration =  300, loss =   0.5994, accuracy =    62.20%
07:16:25 :    iteration =  400, loss =   0.2862, accuracy =    66.50%
07:16:25 :    iteration =  500, loss =   0.6405, accuracy =    62.50%
07:16:26 :    iteration =  600, loss =   0.3936, accuracy =    66.90%
07:16:26 :    iteration =  700, loss =   0.5023, accuracy =    66.30%
07:16:27 : epoch   49 took 3.92s
```

Training with only Block 1.

The filter size in block 5 was changed to 16*16*32*10 to accommodate the output of the first block.

Results:





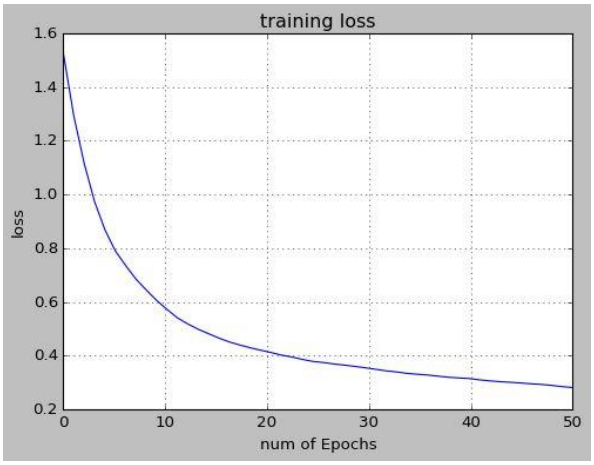
ii. Block1 and Block2.

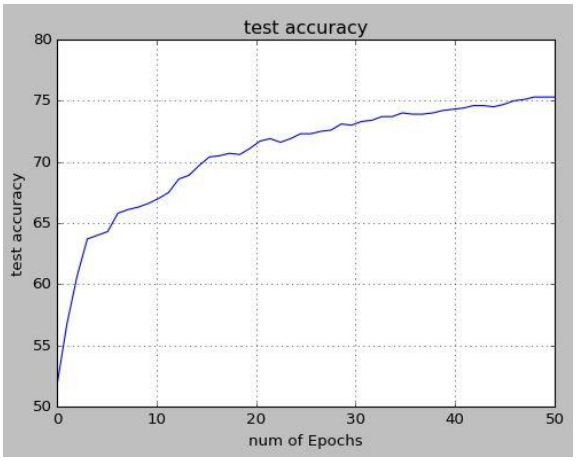
```
07:54:07 :    iteration =    0, loss =   0.3649, accuracy =    74.80%
07:54:08 :    iteration =  100, loss =   0.3943, accuracy =    75.60%
07:54:08 :    iteration =  200, loss =   0.4353, accuracy =    75.10%
07:54:09 :    iteration =  300, loss =   0.7286, accuracy =    71.80%
07:54:10 :    iteration =  400, loss =   0.4632, accuracy =    74.00%
07:54:10 :    iteration =  500, loss =   0.6284, accuracy =    74.00%
07:54:11 :    iteration =  600, loss =   0.4741, accuracy =    73.80%
07:54:12 :    iteration =  700, loss =   0.6140, accuracy =    75.30%
07:54:12 : epoch   48 took 5.64s
07:54:12 : epoch   49/  50
07:54:12 :    iteration =    0, loss =   0.3635, accuracy =    74.90%
07:54:13 :    iteration =  100, loss =   0.3887, accuracy =    75.50%
07:54:14 :    iteration =  200, loss =   0.4300, accuracy =    75.10%
07:54:15 :    iteration =  300, loss =   0.7225, accuracy =    72.00%
07:54:15 :    iteration =  400, loss =   0.4600, accuracy =    74.00%
07:54:16 :    iteration =  500, loss =   0.6217, accuracy =    74.20%
07:54:17 :    iteration =  600, loss =   0.4669, accuracy =    73.80%
07:54:18 :    iteration =  700, loss =   0.6092, accuracy =    75.30%
07:54:18 : epoch   49 took 5.64s
```

Training with Blocks 1 and 2

The filter size in block 5 was changed to 8*8*32*10 to accommodate the output of the second block.

Results:





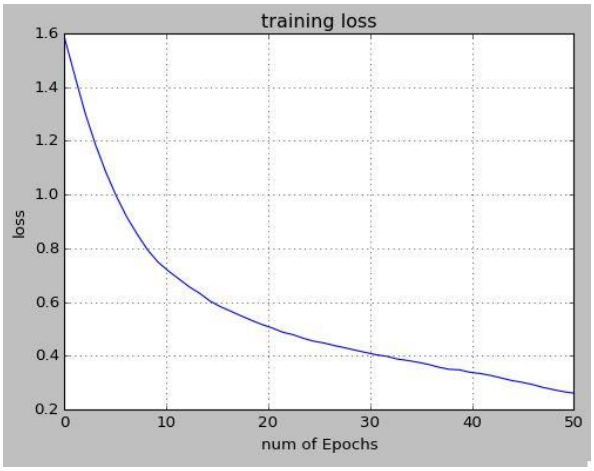
iii. Block1, Block2 and Block3.

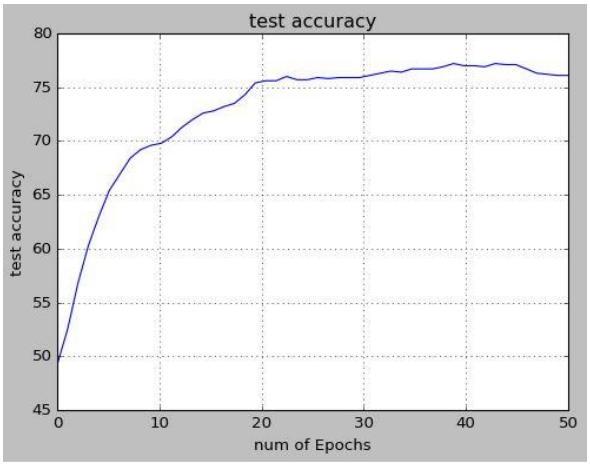
```
08:11:53 : epoch   48/  50
08:11:53 :    iteration =    0, loss =   0.3175, accuracy =    77.10%
08:11:54 :    iteration =  100, loss =   0.4825, accuracy =    76.70%
08:11:55 :    iteration =  200, loss =   0.3128, accuracy =    75.70%
08:11:56 :    iteration =  300, loss =   0.5656, accuracy =    73.60%
08:11:57 :    iteration =  400, loss =   0.3879, accuracy =    75.80%
08:11:58 :    iteration =  500, loss =   0.5731, accuracy =    74.70%
08:11:59 :    iteration =  600, loss =   0.3307, accuracy =    75.20%
08:11:59 :    iteration =  700, loss =   0.6528, accuracy =    76.10%
08:12:00 : epoch   48 took 6.81s
08:12:00 : epoch   49/  50
08:12:00 :    iteration =    0, loss =   0.3078, accuracy =    77.50%
08:12:01 :    iteration =  100, loss =   0.4778, accuracy =    76.90%
08:12:02 :    iteration =  200, loss =   0.3072, accuracy =    75.50%
08:12:03 :    iteration =  300, loss =   0.5577, accuracy =    73.50%
08:12:04 :    iteration =  400, loss =   0.3802, accuracy =    75.90%
08:12:04 :    iteration =  500, loss =   0.5704, accuracy =    74.80%
08:12:05 :    iteration =  600, loss =   0.3212, accuracy =    75.50%
08:12:06 :    iteration =  700, loss =   0.6495, accuracy =    76.10%
08:12:07 : epoch   49 took 6.78s
```

Training with blocks 1,2 and 3

The filter size in block 5 was changed to 4*4*64*10 to accommodate the output of the third block.

Results:



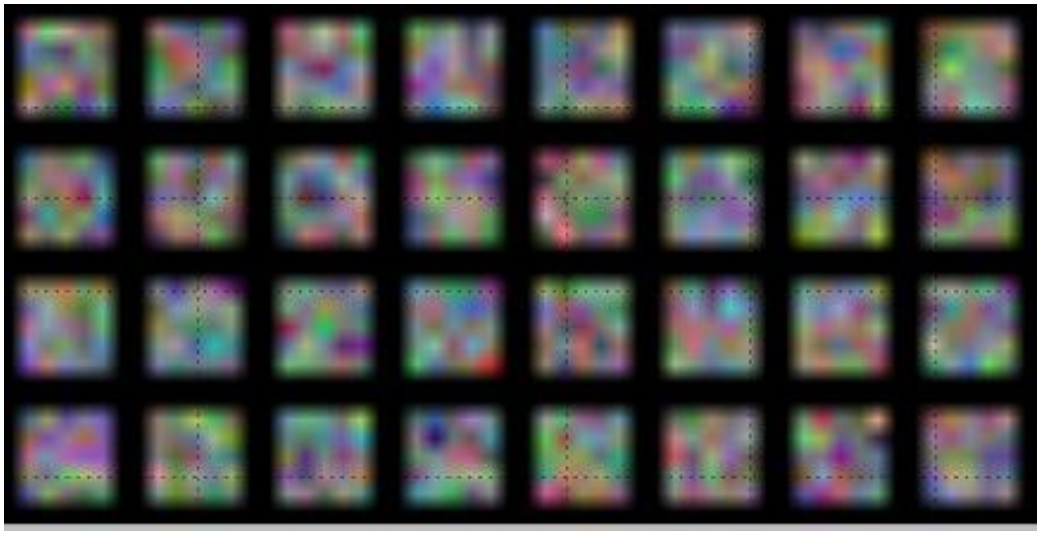


(b) Compare the final test accuracies for (i., ii., iii.) in a Table.

	Block 1	Blocks 1,2	Blocks 1,2,3
Test Accuracy	66.30%	75.30%	76.10%
Loss	0.5023	0.6092	0.6495

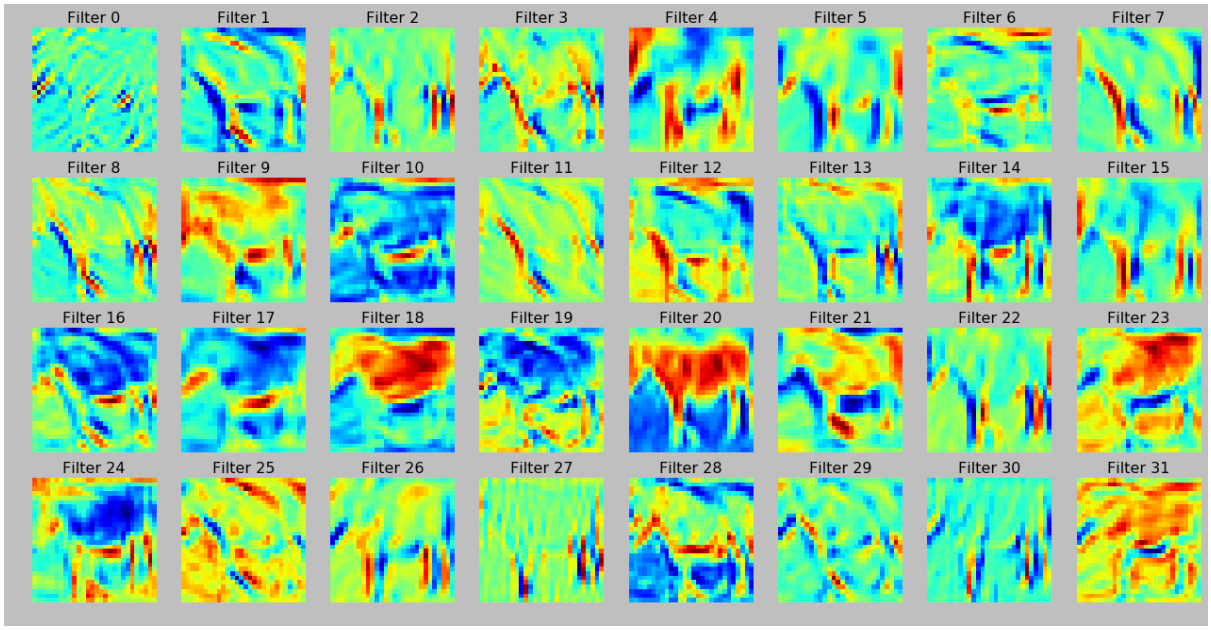
3. Visualization of filters and activations.

(a) Plot the learned 32 filters of the first convoluational layer in LeNet.



Output of the filter kernels from the first convolution layer

(b) Plot the filter response maps for a given sample image of CIFAR-10.



Response map of the 32 different filters, achieved by running a convolution on the original image(x).

Thus we have trained our model using Convolution Neural Networks achieving a 76.10% accuracy on classifying the CIFAR10 data into the correct class labels. A better accuracy can be reached (>80%) by applying optimisers on each hidden layer (such as GradientDecent, Adam etc). This helps us achieve better results, minimising the loss function and reaching the optimal solution in lesser time(epochs).