CS7011: Programming Assignment 2

Priyesh V(CS15S012), Vishruit Kulshreshtha(ME12B159)

March 2017

1 Introduction

This assignment illustrates the effect of different optimizers, network architecture, loss functions, etc on the training effectiveness of the resulting network. Here, in each section, error rate vs number of epochs and loss vs number of epochs is being studied for different network configurations.

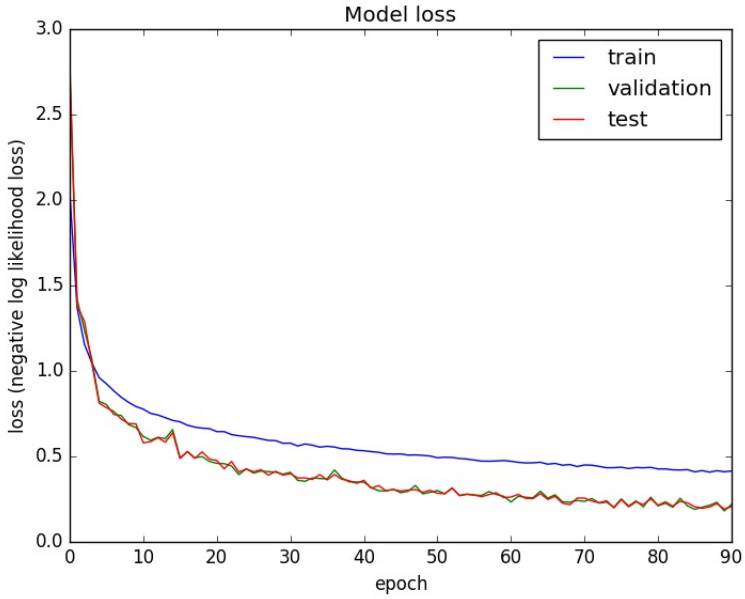
Final aim of this assignment is to gain comfort with the different back-propagation algorithms and be able to code/implement it at the very basic level. It also tries to generate some intuition to the actual working of the neural network under the hood.

Also for transparency purposes, the actual log files of these plots can be verified at this link, which also contains the network weights of the last epoch in that run.

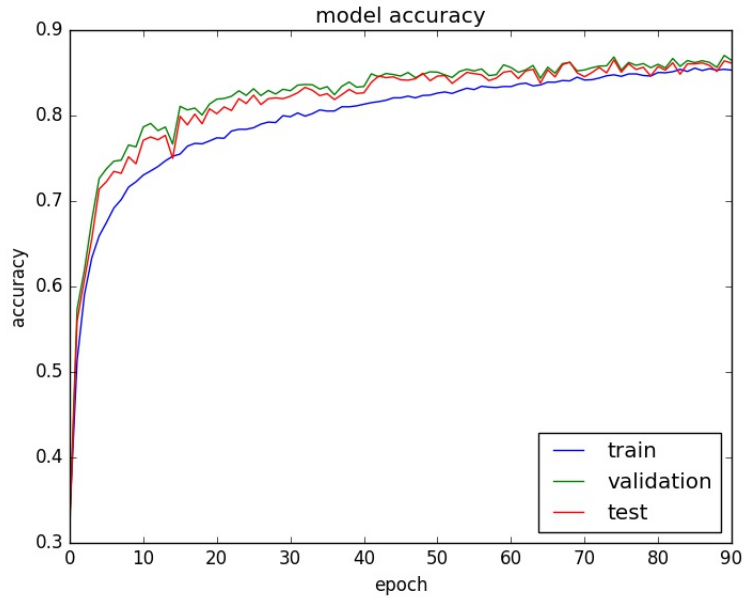
Also, to note here, the output function of the neural network

2 Task I

A plot of the learning curve showing iterations on the x-axis and negative log likelihood over labels on the y-axis is plotted. The plot shows a comparison between train loss, validation loss and test loss as the training progresses over the epochs. Here, we can see that the validation and test loss are lower than the training loss. This is a rare phonomena when the model is able to generalize well on the validation and test data than the training data. This could be possible due to the good regularizing effect of the dropout layers that we have placed after every 'convolutional + pooling' or convolutional layer. Also, as we've taken batch size as 100 and a moving mean definition for Batch Normalization layer, we observe a lot of fluctuations in the test and validation losses and accuracies which is aptly explained by the low number of samples as compared to training dataset.



We've also plotted the 'Model Accuracy' plot to show a qualitative relationship between the loss values and the corresponding accuracies achieved. Here, even though the loss values have a considerable gap, the train/valid accuracy, however, appears to converge to a single accuracy value.



3 Task II

The performance on the test data of the model that performs best on the validation data.

We obtained a categorical classification accuracy of about 88% on the CIFAR10 test dataset. We have used Batch Normalizaton layer at the output layer as mentioned along with dropouts and data augmentation with patience parameter based learning rate decay and Xavier initalization to achieve this result.

4 Task III

The parameter setting which gave you the best results.

We obtained 88% as the best testing time accuracy and the parameter settings that worked out well for us are,
patience parameter p = 5,
dropout-rate = 0.5,
batch-size = 100,
initial learning rate = 0.001,
initialization choice = Xavier initialization

5 Task IV

The table below summarizes the types of layers, their input and output shapes and the number of parameters required.

Layer (type)	**Input Shape**	**Output Shape**	**Param**
Input Data (InputLayer)	(None, 32, 32, 3)	(None, 32, 32, 3)	0
CONV1 (Conv2D)	(None, 32, 32, 3)	(None, 32, 32, 64)	1792
POOL1 (MaxPooling2D)	(None, 32, 32, 64)	(None, 16, 16, 64)	0
CONV2 (Conv2D)	(None, 16, 16, 64)	(None, 16, 16, 128)	73856
POOL2 (MaxPooling2D)	(None, 16, 16, 128)	(None, 8, 8, 128)	0
CONV3 (Conv2D)	(None, 8, 8, 128)	(None, 8, 8, 256)	295168
CONV4 (Conv2D)	(None, 8, 8, 256)	(None, 8, 8, 256)	590080
POOL3 (MaxPooling2D)	(None, 8, 8, 256)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4, 4, 256)	(None, 4096)	0
FC1 (Dense)	(None, 4096)	(None, 1024)	4195328
FC2 (Dense)	(None, 1024)	(None, 1024)	1049600
SOFTMAX (Dense)	(None, 1024)	(None, 10)	10250
BN (Batch Normalization)	(None, 10)	(None, 10)	40
$\text{SOFTMAX}_{Activation}(ReLU - Activation)$	(None, 10)	(None, 10)	0

Hence, we have a total of 6,216,114.0 parameters out of which 6,216,094.0 are trainable and remaining 20 non-trainable params. There are 960896 params

in the convolutional layers and 5255218 in the dense fully connected layers. This clearly indicates that the fully connected layers are much expensive and difficult to train as compared to the convolutional layers.

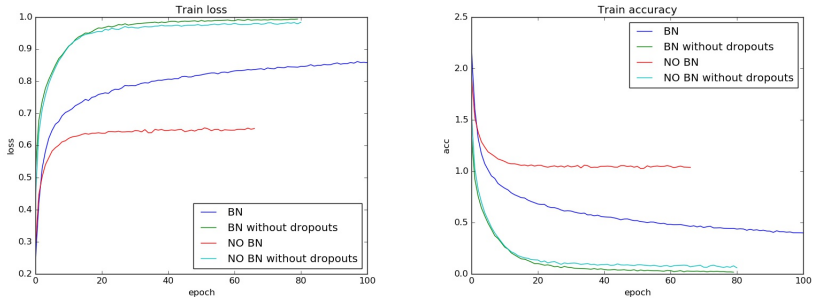
The number of filters is the number of feature maps, since each neuron perform a different convolution on the input to the layer (more precisely, the neurons' input weights form convolution kernels). Hence, the number of neurons in the fully connected layers are 1024+1024+10 = 2058. And, the number of neurons in the convolutional layers are 64*32*32+128*16*16+256*8*8+256*8*8 = 131072.

6 Task V

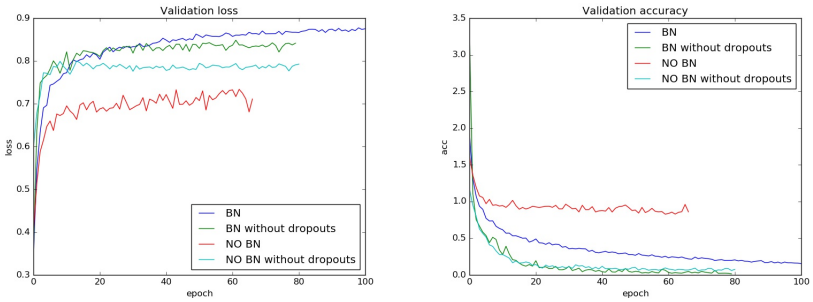
What was the effect of using batch normalization ?

Batch normalization helps the layer after it, see the same input distribution as the layer before it. Hence, if we apply batch normalization after every 'convolutional + pooling' layer and 'convolutional layer', then each layer will see the distribution of the inputs same as the input data to the network itself. This results in efficient training of the models.

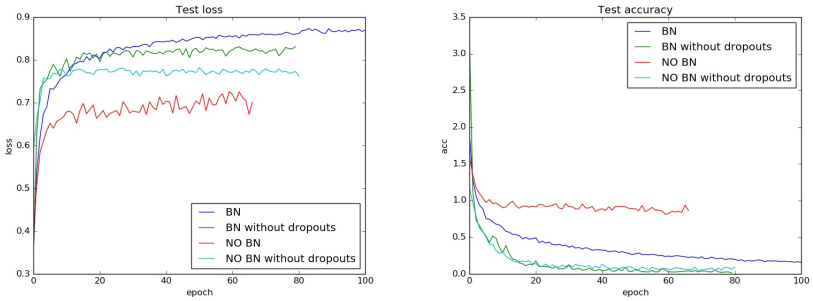
We have plotted the figures to enhance the qualitative understanding of the effect of batch normalization during training, testing, and validation via the trends in the variations of the log loss and accuracy, respectively. A legend of the figures contains combinations of batch normalization and no batch normalization, and dropouts and no dropouts and is explained below. BN = Batch Normalization is present in the network
BN without dropouts = Batch Normalization without the dropout layers
NO BN = Batch Normalization layer is absent in the network NO BN without dropouts = Network without batch normalization layers and dropout layers



Effect of Batch Normalization on training loss and accuracies.



Effect of Batch Normalization on validation loss and accuracies.

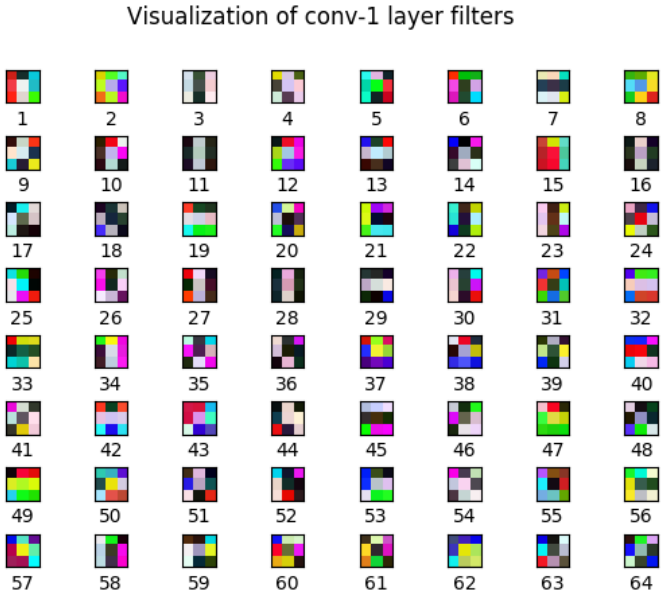


Effect of Batch Normalization on test loss and accuracies.

7 Task VI

Plot all the 64 layer 1 filters in an 8 8 grid. Do you observe any interesting patterns?

The image below shows all the 64 filters of the convolutional layer 1 (CONV1) in an 8x8 grid.
It is interesting to note the patterns of vertical strokes, patches, horizontal strokes.

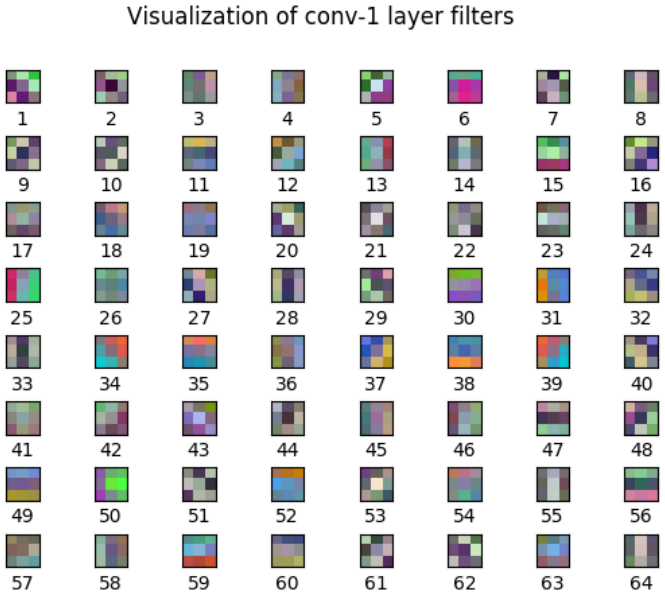


The filters 3, 11, 16 and 28 have horizontal discretizations, i.e., they are smmooth in the vertical direction. This implies that they get activated for vertical patches in the image. If we closely observe 11, it indicates that this filter detects vertical edges with sharp contrast on both the sides.

Similarly we have, filter numbers 13, 7, 19, 32, 29, 45 and 49 that gets activated when they observe a horizontal feature/stroke in the images with varying conditions on both of its sides.

Observe filter number 47, it kind of tries to detect a lower left cornered patch of probably green color.

The filtes are bright because we have'nt normalized them back. We've attached another figure where the filters are normalized and they have a similar explanation as for the one before.

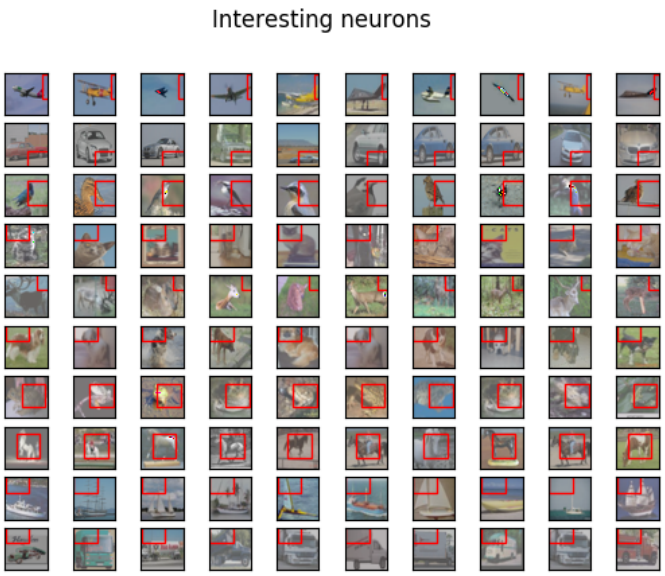


8 Task VII

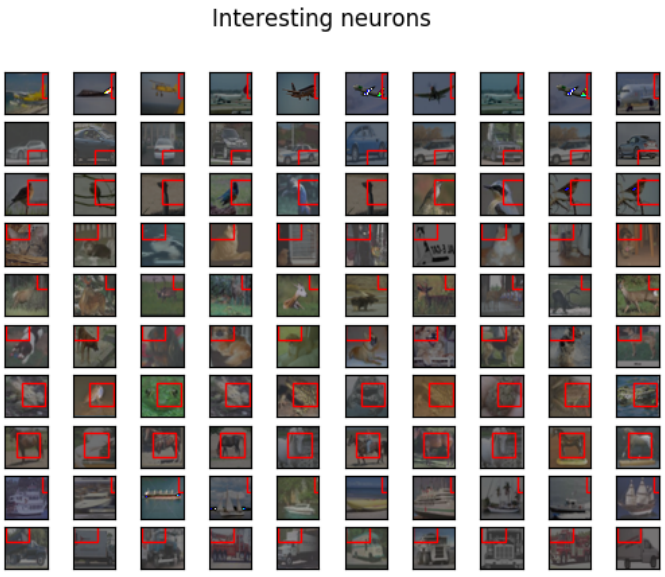
Interesting Neurons

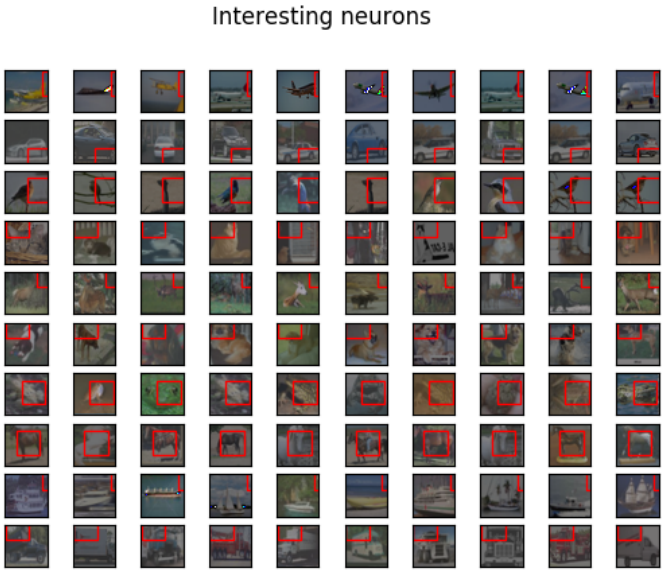
Discover some interesting neurons in CONV5. One way of doing this is to feed in a lot of images to the convnet and see if a particular neuron gets excited by similar kinds of images (say, by all car images). For such neurons trace back the patch in the original image which is responsible for exciting them and plot these patches. You can plot these patches for 10 interesting neurons. You also need to submit the code which you use for computing this trace back to the image patch.

We have plotted a 10x10 grid of images with each row corresponding to a particular class label as specified in the CIFAR10 data documentation



Filter





Since we've used the queues, we haven't picked up the best pics for all the classes. Also, we have shuffled thruogh the data in order to get the approriate filters.

9 Task VII

Guided back propagation

Apply guided back propagation on any 10 neurons in the CONV5 layer and plot the images which excite this neuron. The idea again is to discover interesting patterns which excite some neurons. We've implemented the guided back propagation, the code

10 Miscellaneous

The following are the observations and graphs on some of the other experiments.