Report for Programming Assignment-2

March 31, 2019

Checklist :

- ☑ checked.
- ☐ unchecked.
- ☒ not done.

The task you need to ensure before submission.

- ☑ We have read all the instruction carefully and followed them to our best ability.
- ☑ We have written the name, roll no in report.
- ☑ Run sanity_check.sh.
- ☑ We will be submitting only single submission on behalf of our team.
- ☑ We have not included unnecessary text, pages, logos in the assignment.
- ☑ We have not used any high level APIs(Keras, Estimators for e.g.).
- ☑ We have not copied anything for this assignment.

Team Mate 1 : Ojas Mehta
Roll No : CS18M038
Team Mate 2 : Avani Shukla
Roll No : CS18M052

Report:

- Configuration and training details of your best performing model.
  conv1 with 64 filters
  conv2 with 128 filters
  conv3 with 128 filters
  conv4 with 128 filters
  conv5 with 128 filters

conv6 with 256 filters
each filter of size 3×3
Ensemble of models with 4, 5 and 6 pooling layers with last pooling layer as average pooling rest max pooling layer.
two fully connected layers with 1100 and 400 neurons respectively and each followed by a dropout layer.
In one variant, we used dropout after 6th convolution layer.
Batch normalization is used at each layer.

- A plot of the learning curve showing iterations on the x-axis and negative log likelihood over labels on the y-axis. Make a single plot showing both the training loss and the validation loss.

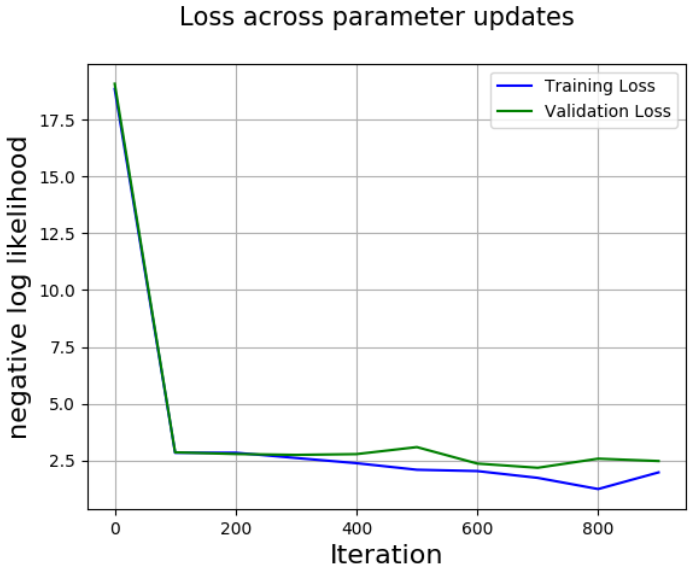

Figure 1: learning rate = 0.001 and batch size = 64

- The performance on the test data of the model that performs best on the validation data.
Our model that performs best on the validation data has 60.137 accuracy that we have achieved by ensembling of 5 models .

- The parameter setting which gave you the best results.
Ensembling of models with batch size 64 and 128.
Models with dropout 0.6 , 0.7 after FC layer and 0.2, 0.15 after last convolution layer was used.
Learning rate of 0.0001 is used.
And early stopping with patience of 12 epochs was used. Adam optimizer is used as an optimizer and leaky relu is used as activation on each layer function.

- Write down the dimensions of the input and output at each layer (for example, the input to Conv1 layer is $3 \times 64 \times 64$)
conv1 input: $3 \times 64 \times 64$

conv1 output:$32 \times 64 \times 64$
conv2 input:$32 \times 64 \times 64$
conv2 output:$64 \times 64 \times 64$
conv3 input:$64 \times 32 \times 32$
conv3 output:$64 \times 32 \times 32$
conv4 input:$64 \times 32 \times 32$
conv4 output:$64 \times 32 \times 32$
conv5 input:$64 \times 16 \times 16$
conv5 output:$64 \times 16 \times 16$
conv6 input:$64 \times 16 \times 16$
conv6 output:$128 \times 14 \times 14$
fc1 input:6272
fc1 output:256
softmax input:256
softmax output:20

- Exactly how many parameters does your network have? How many of these are in the fully connected layers and how many are in the convolutional layers?
  total parameters = 1,811,636 parameters
  parameters in convolution layers = 200,608 (including 6720 biases)
  parameters in fully connected layers = 1,611,028 (including 276 biases)
- Exactly how many "neurons" does your network have? How many of these are in the fully connected layers and how many are in the convolutional layers?
  total neurons = 1,804,640
  neurons in convolution layers = 193,888
  neurons in fully connected layers = 1,610,752
- What was the effect of using batch normalization ?

1. Without batch normalization

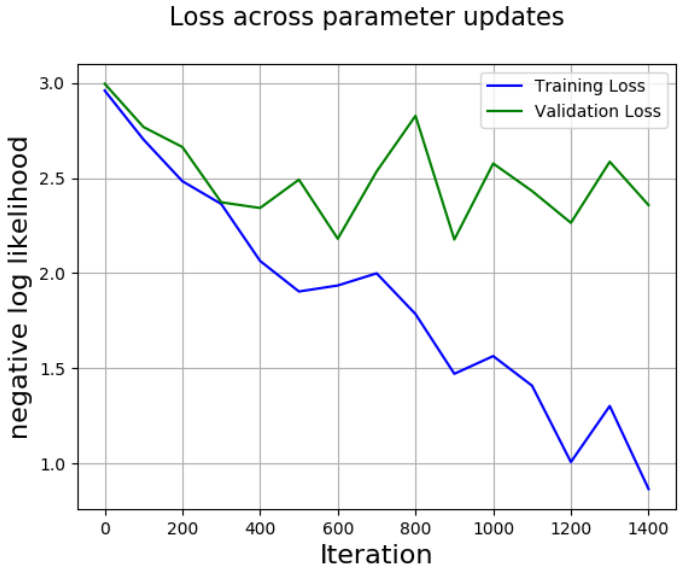

Figure 2: No batch normalization learning rate = 0.0001 and batch size = 128

2. With batch normalization

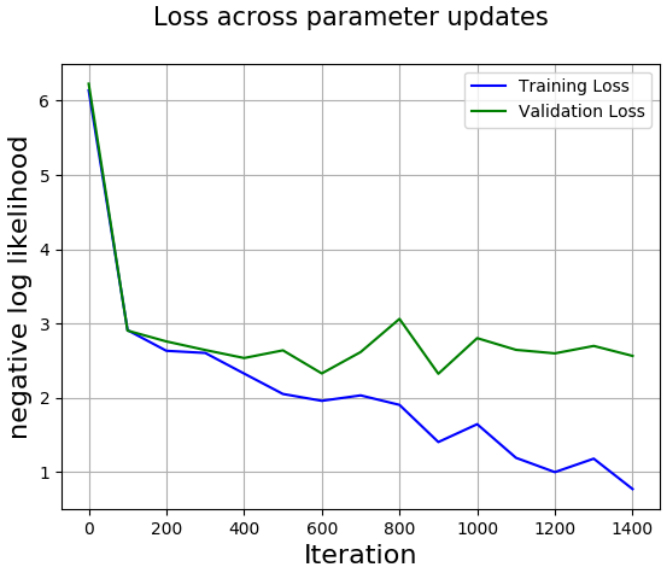

Figure 3: learning rate = 0.0001 and batch size = 128

So, with batch normalization convergence is faster especially in initial epochs.

- Plot all the 32 layer-1 (Conv1) filters in an 4×8 grid. Do you observe any interesting patterns?

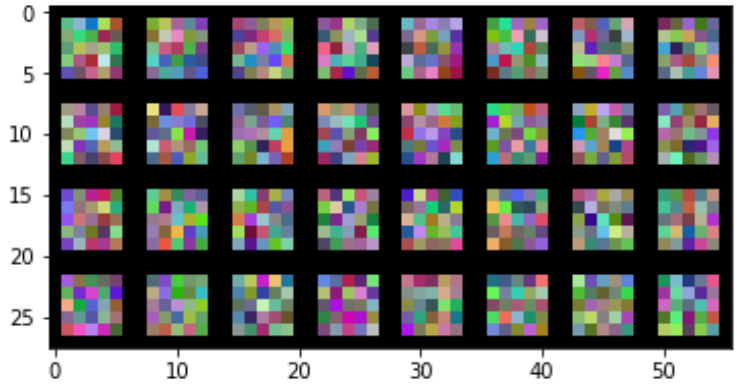

Figure 4: filters in 4×8 grid

- Apply guided back propagation on any 10 neurons in the Conv6 layer and plot the images which excite this neuron. The idea again is to discover interesting patterns which excite some neurons.
- (Extra credits) Refer to (http://www.evolvingai.org/fooling) and try to find different ways in which you can fool the network. One way is to randomly change some pixels in the image such that the class is changed. Report the plot of accuracy v/s number of pixels changed on the test set.

1 Data Augmentation

For Data Augmentation, we created images with following operations:
horizontal flip, vertical flip, different degree of rotation, random crop. Data was augmented to make the new training data 18 times of the given training data for obtaining best accuracy.
For generating plots training data was augmented to get 4 times of the given size.

2 Additional creative idea

We used ensembling of 5 models to get best accuracy.

3 Hyperparameter Tuning

- Learning rate

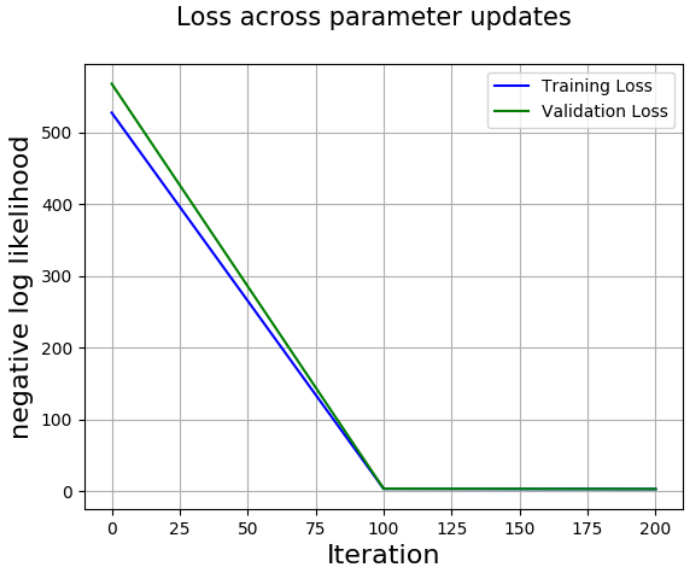

Figure 5: learning rate = 0.01 and batch size = 64

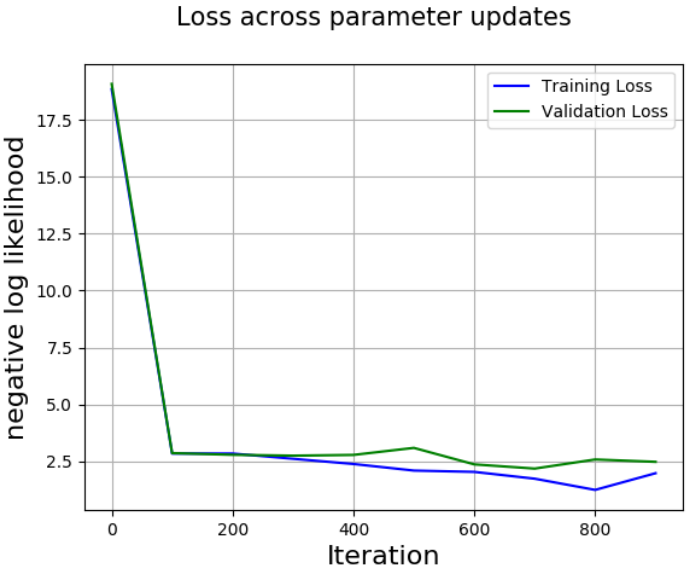

Figure 6: learning rate = 0.001 and batch size = 64

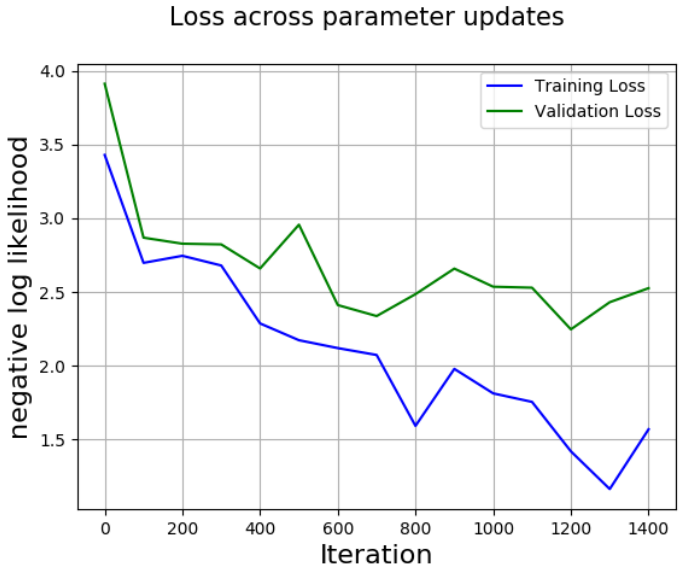

Figure 7: learning rate = 0.0001 and batch size = 64

- batch size

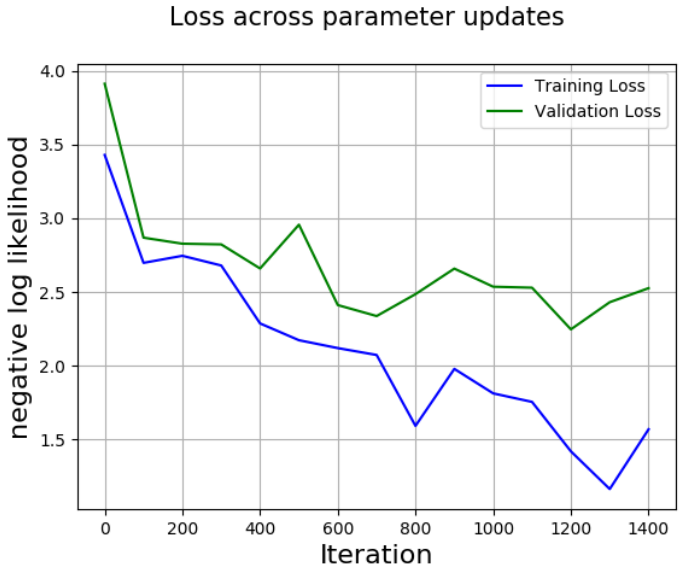

Figure 8: learning rate = 0.0001 and batch size = 64

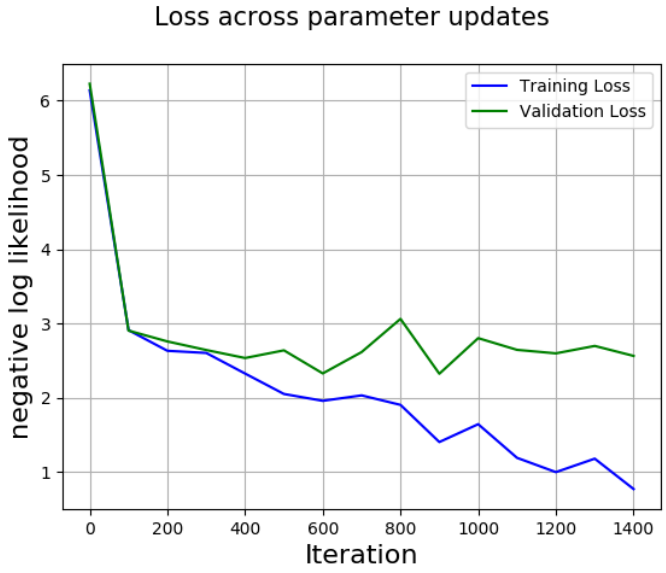

Figure 9: learning rate = 0.0001 and batch size = 128

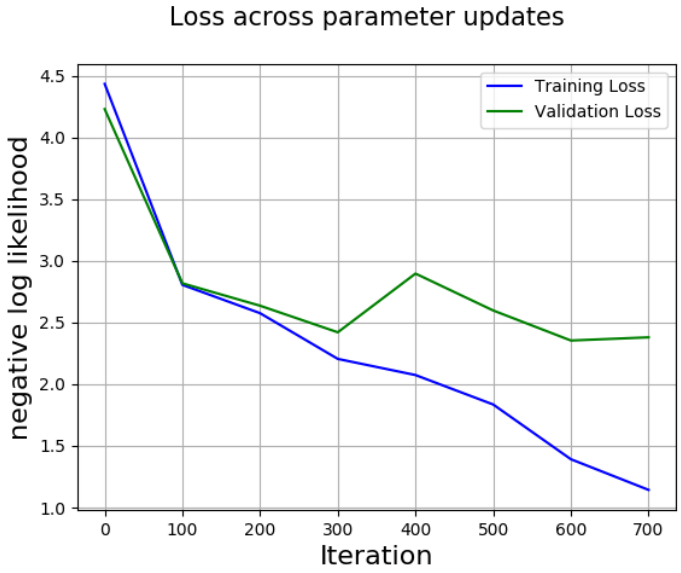

Figure 10: learning rate = 0.0001 and batch size = 256

- initialization
  xavier initialization

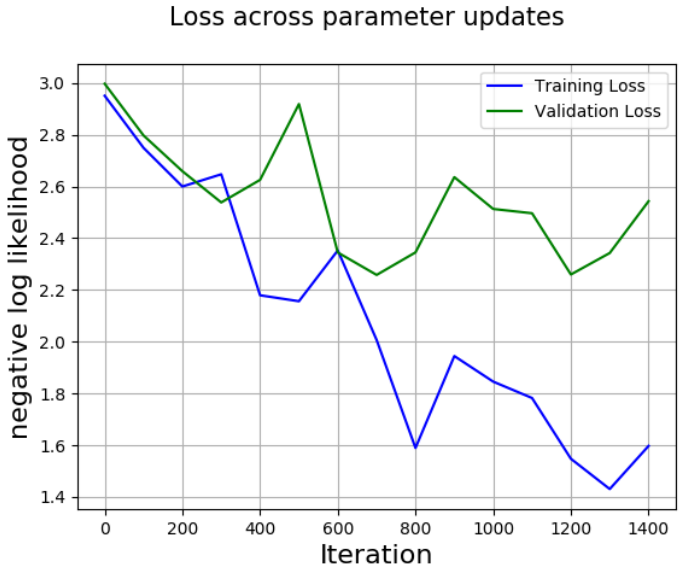

Figure 11: xavier initialization with learning rate = 0.0001 and batch size = 64

He initialization

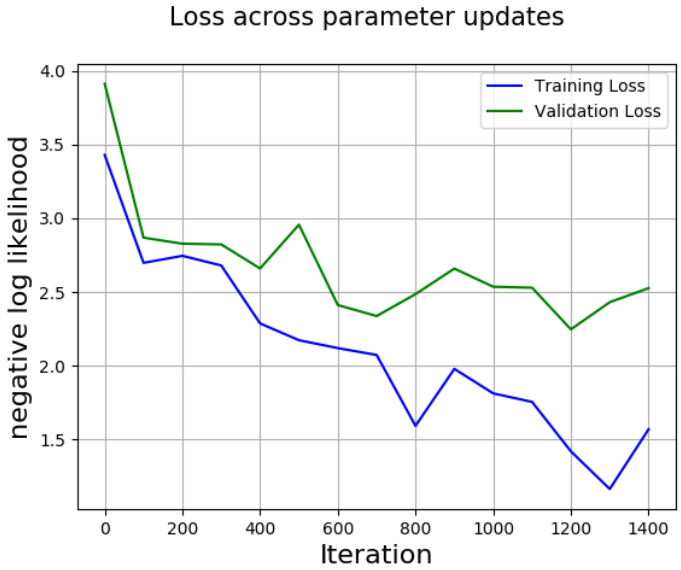

Figure 12: He initialization with learning rate = 0.0001 and batch size = 64