Assignment-2 Report

Sai Hemanth Kasaraneni (14EE35011)

February 10, 2019

1 Preliminary Details

Initially the image data (both training and testing) was normalized by dividing by 255 to ensure the pixel ranges between 0 and 1. By doing this,the models significantly improved in terms of convergence rate with higher learning rates.

It is expected that normalization will make the learning process easier by reducing the overhead of taking care of the distortions in the data points arising just because of the scaling factors e.g, you have two data points [0.5, 0.2, 0.3] and [500, 200, 300] belonging to the same class. They are essentially the same data points with different scaling factors.

Only weights which are required to be submitted (Task A and Task C)are submitted in weights folder. Weights in task b (not required to be submitted according to the question) can be downloaded from https://drive.google.com/open?id=1Hamue3qCrEd5Ae65056hfbxmbnXGIv94

The training process for each experiments in each task is performed with Number of epochs : 100 (can be changed in module.py file easily just at three places) and corresponding plots were obtained for 100 epochs when trained using Google colab.

Typical training time per epoch on Google Colab : 1 sec

Typical training time per epoch on CPU (I7 - 5th gen) : 5 to 15 sec

2 Task A

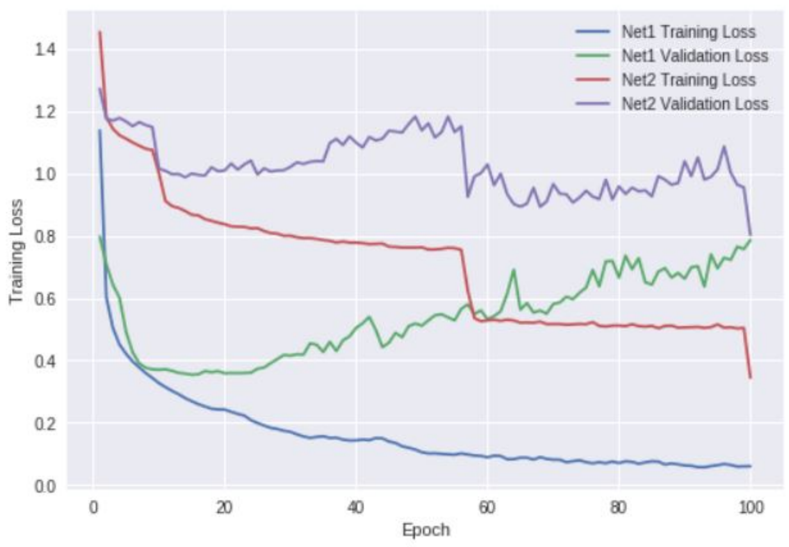

Figure 1: Training Loss comparison graph for Naive network 1 and Naive network 2

Test accuracy for Network 1 and Network2 are obtained as 86.15 and 88.02 respectively and corresponding test loss values are 0.383 and 0.861 respectively

In statistics and machine learning, one of the most common tasks is to fit a ”model” to a set of training data, so as to be able to make reliable predictions on general untrained data. In overfitting, a statistical model describes random error or noise instead of the underlying relationship

More number of Neurons doesn’t imply good performance, it all depends on data-set. If the deep and narrow network is performing poorly than wide and shallow, it means chances are that it is over-fitting on training data

3 Task B

3.1 Experiment 1

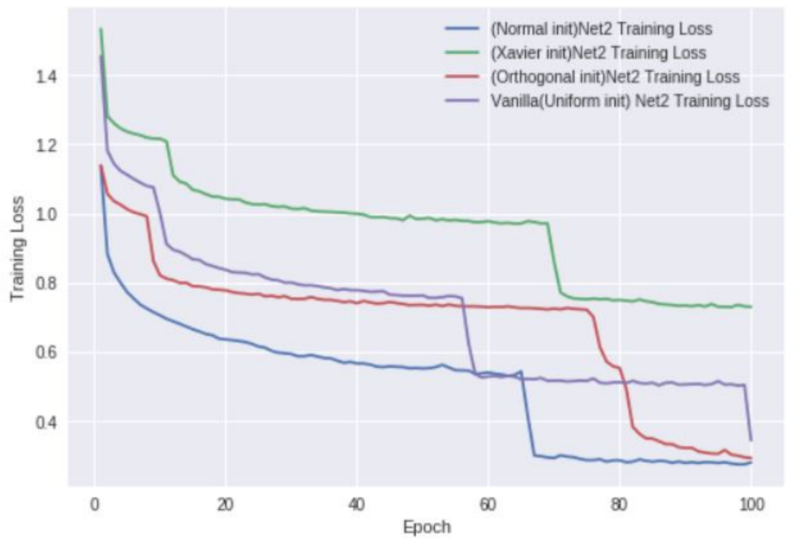

Figure 2: Training Loss comparison graph for various initializations in Network 2

The Normal initialization is found to result in the best performance during training. Xavier initialization is advantageous when it is used for initializing deeper networks. But our Network 2 is a wide and shallow network. So normal initialization is sufficient to break the symmetry in layers.

Normal initialization best validation accuracy and corresponding validation loss : 83.39, 0.816

Xavier initlialization best validation accuracy and corresponding validation loss : 63.60, 1.2

Orthogonal initlialization best validation accuracy and corresponding validation loss : 81.49, 0.872

Vanilla (Uniform) initlialization best validation accuracy and corresponding validation loss : 88.63, 0.803

so from results also, we can observe poor performance of xavier initialization as our network is wide and shallow

3.2 Experiment 2

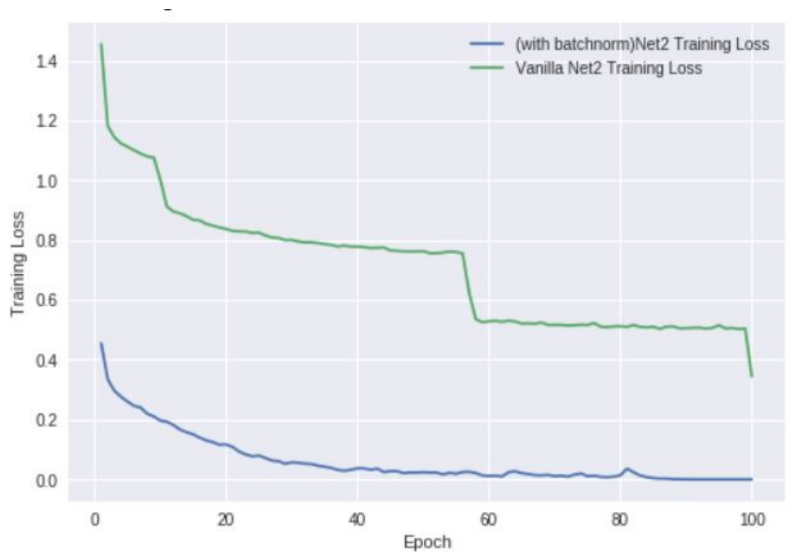

Figure 3: Training Loss comparison graph with and without batchnorm for network 2

Faster convergence is observed with Batch Normalization as expected.

Broadly speaking, BatchNorm is a mechanism that aims to stabilize the distribution (over a mini- batch) of inputs to a given network layer during training and even reduces overfitting. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift).So we can use higher learning rates because batch normalization makes sure that theres no activation thats gone really high or really low. And by that, things that previously couldnt get to train, it will start to train. And we can see from the graph also, batch normalization decreased training loss significantly

3.3 Experiment 3

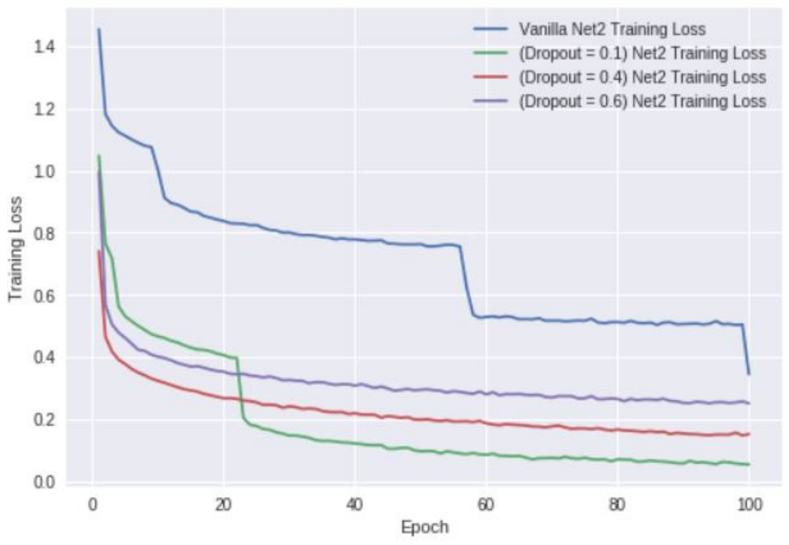

Figure 4: Training Loss comparison graph for different dropout values in Network 2

Best validation accuracy and corresponding validation loss with dropout 0.1 are 89.68 0.560

Best validation accuracy and corresponding validation loss with dropout 0.4 are 90.15 0.338

Best validation accuracy and corresponding validation loss with dropout 0.6 are 89.79 0.317

We can observe from plot that, using dropout clearly results in faster convergence and lower loss values.

Thus, it is observed that dropout prevents overfitting when compared to the vanilla model. Model with Dropout 0.6 achieves the lowest loss across all models.

In the training loss curve, it is observed that higher dropout leads to slightly higher training loss. Dropout is a regularizer and improves generalization at the expense of training set loss.

3.4 Experiment 4

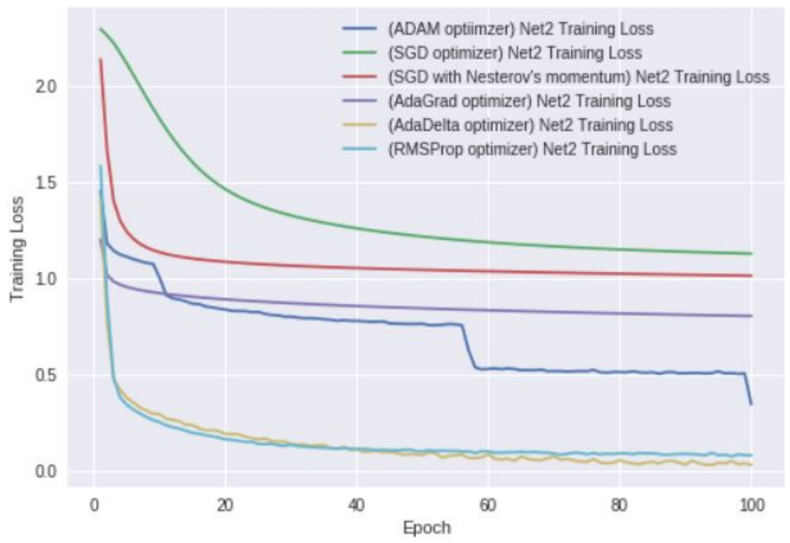

Figure 5: Training Loss comparison graph for Naive network 1 and Naive network 2

It is observed that SGD fares worse when compared to other optimizers. SGD with Nesterov momentum improves convergence but is still worse when compared to AdaGrad, Adam, AdaDelta and RMSProp. Both SGD and SGD with Nesterov momentum manipulates the learning rate globally and equally for all parameters, unlike the other methods. Unlike SGD, algorithms such as RMSprop and Adadelta All previous approaches weve discussed so far manipulated the learning rate globally and equally for all parameters. Tuning the learning rates is an expensive process, so much work has gone into devising methods that can adaptively tune the learning rates, and even do so per parameter. Many of these methods may still require other hyperparameter settings, but the argument is that they are well-behaved for a broader range of hyperparameter values than the raw learning rate.

A visualization of a saddle point in the optimization landscape, where the curvature along different dimension has different signs (one dimension curves up and another down). Notice that SGD has a very hard time breaking symmetry and gets stuck on the top. Conversely, algorithms such as RMSprop will see very low gradients in the saddle direction. Due to the denominator term in the RMSprop update, this will increase the effective learning rate along this direction, helping RMSProp proceed.

SGD with Nesterov momentum slightly performed better than SGD as it is expected to escape the saddle points for which sgd gets trapped.

In this case, Adadelta and RMSprop does the best job at optimization

4 Task C

When the activations of hidden layers of neural netowork are used as features for logistic regression, not much difference is observed between the accuracy values when compared with the vanilla network 2 values. However, the loss value is observed to increase slightly as we go for higher layers.This is maybe due to prominent increase in the number of features in input layers which seams to increase loss value for almost same accuracy

Test accuracy and test loss obtained with Vanilla net are 88.02, 0.861

Test accuracy and test loss obtained with first hidden layer activations as features are 88.49, 0.485

Test accuracy and test loss obtained with second hidden layer activations as features are 88.66, 0.881

Test accuracy and test loss obtained with third hidden layer activations as features are 88.24, 1.224