DD2424 Assignment 3 Report

Student: Alberto Xamin xamin@kth.se

State how you checked your analytic gradient computations and whether you think that your gradient computations are bug free for your k-layer network with batch normalization.

The gradients were checked with a subset of the training data and with a network with 10 hidden nodes to speed up the comparison.

```
nnt = NeuralNet((train_X[:, 0:100], train_Y[:, 0:100], train_labels[:, 0:100]),
         mu, sigma, hidden_nodes=[10])
nnt.SanityCheck()

# where sanitycheck is defined as
def SanityCheck(self):
    x, y = self.data[0][:, 0:1], self.data[1][:, 0:1]
    anw, anb  = self.ComputeGradients(x, y, 0)
    print("an-done")
    numw, numb = self.ComputeGradsNum(x, y, 0, 1e-5)
    print("num-done")

    for i in range(len(anw)):
            print(f"W diff:{abs(np.mean(anw[i]) - np.mean(numw[i]))}\t
             b diff:{abs(np.mean(anb[i]) - np.mean(numb[i]))}")
```

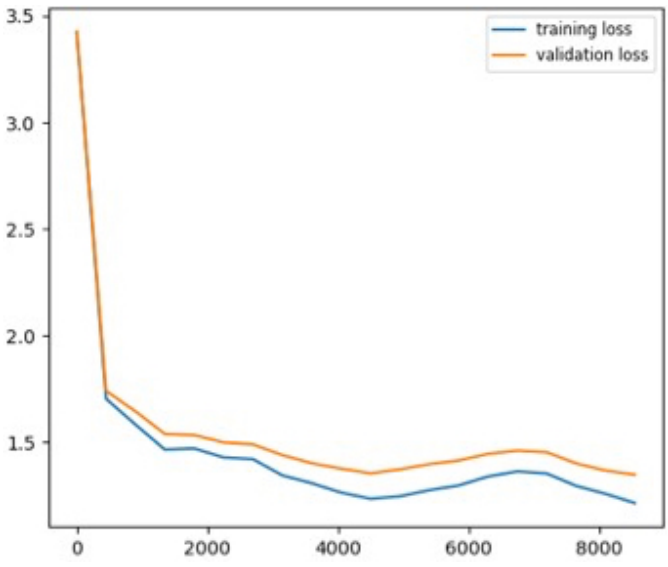
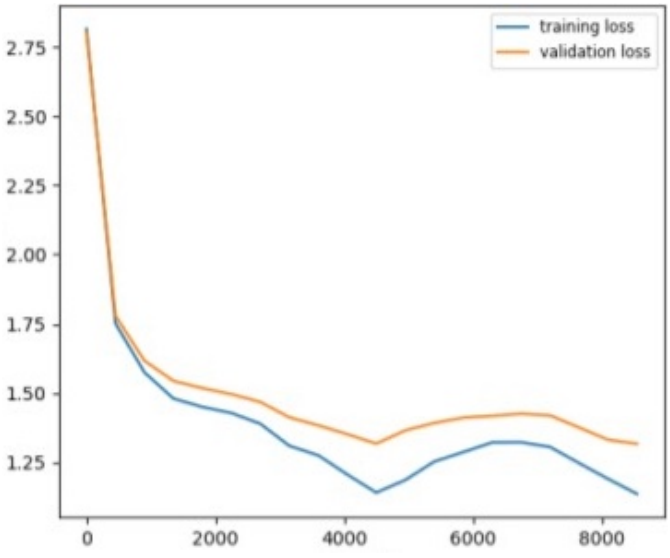
Which outputs

```
an-done
num-done
W diff:0.0021004306687158283e-17    b diff:0.000094371882177268
W diff:2.7545865680201624e-17    b diff:1.5428902930940238e-16
```

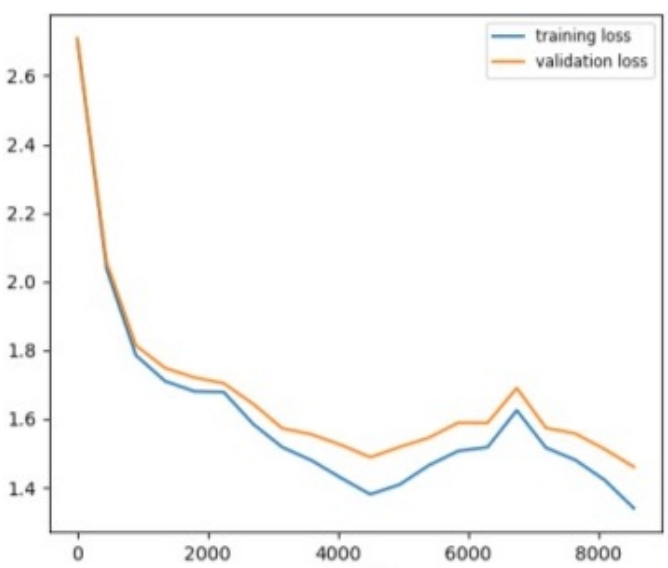
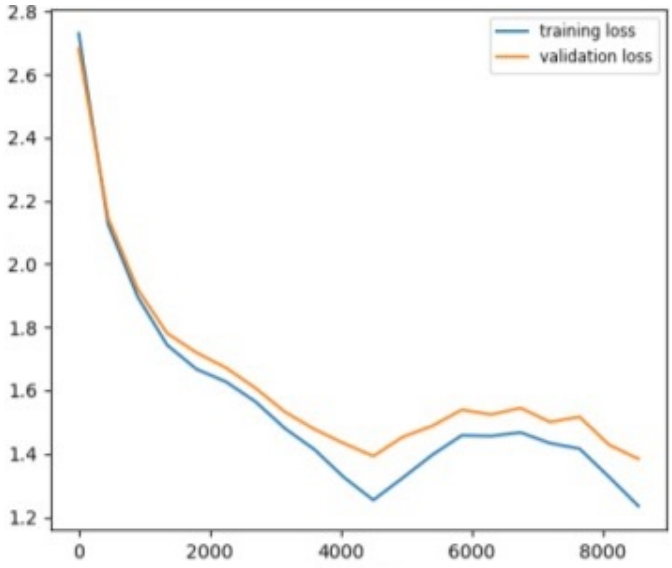
and suggests that the computations are correct.

Include graphs of the evolution of the loss function when you train the 3.layer network with and without batch normalization with the given default parameter setting.

lambda=0, n epochs=40, n batch=100, eta=.1

Without	With
	

Include graphs of the evolution of the loss function when you train the 9.layer network with and without batch normalization with the given default parameter setting.

Without	With
	

State the range of the values you searched for lambda when you tried to optimize the performance of the 3.layer network trained with batch normalization, and the lambda settings for your best performing 3.layer network. Also state the test accuracy achieved by this network.

I define a list of lambda value to be tested:

`[10e-5, 10e-4.5, 10e-4, 10e-3.5, 10e-3, 10e-2.5, 10e-2, 10e-1.5, 10e-1]`

along with batch size 100, step size 2250 and for 2 cycles learning. Below are the top three lambdas during the coarse search for the 3-layer neural networks with batch normalization:

```
lambda: 10e-2   score:0.54
lambda: 10e-3   score:0.5372
lambda: 10e-2.5 score:0.5302
```

Then a series of finer searches was run.

Search	iteration	Range of lambda	Top three lambda
Coarse	0	[10e-5, 10e-1]	[0.01, 0.001, 0.00316]
Fine	1	[10e-3, 10e-2]	[0.00335, 0.00595, 0.00517]
Fine	2	[10e-2.47, 10e-2.22]	[0.00353, 0.00443, 0.00505]
Fine	3	[10e-2.36, 10e-2.34]	[0.00440, 0.00439, 0.00435]

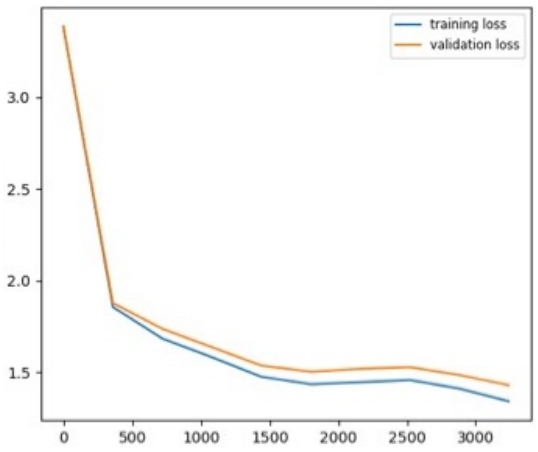
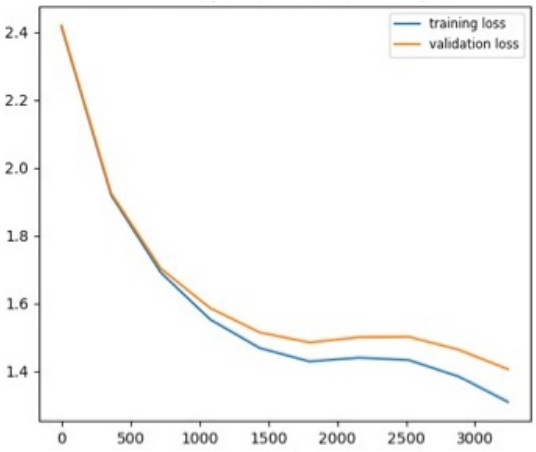
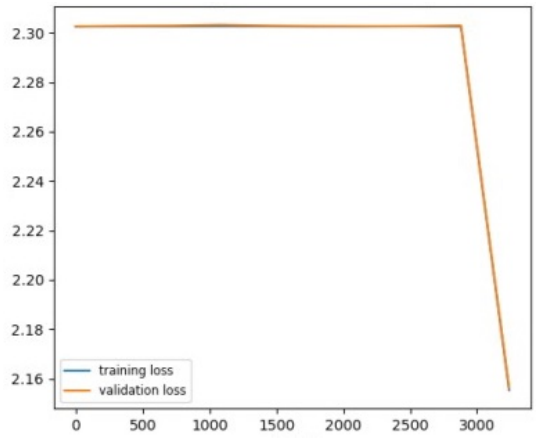
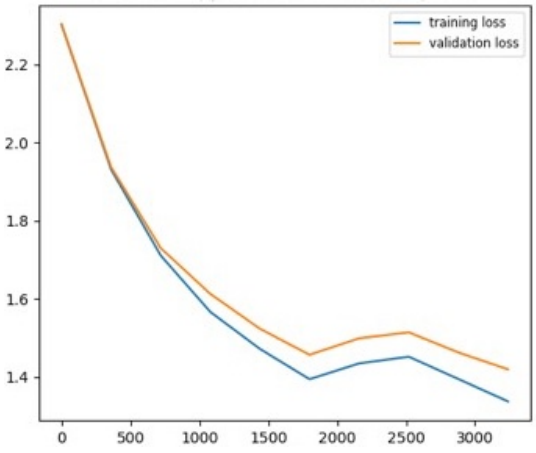
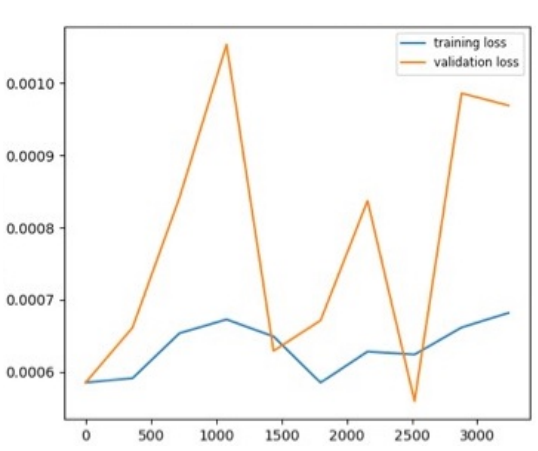
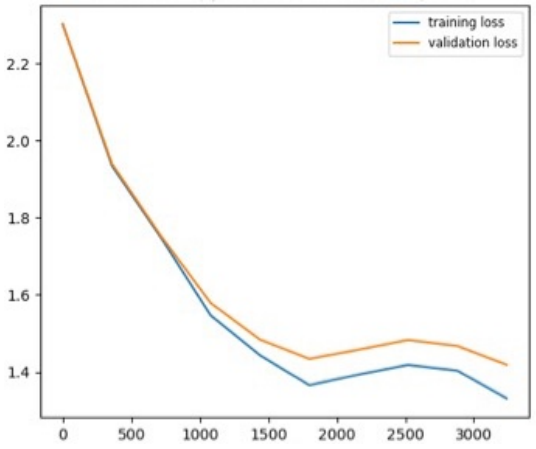
The best lambda gained from this fine search is `0.00440015` or `10e-2.357`, which achieves accuracy almost `55.5%` on validation datasets.

And training the network for 50 epochs reports a final test score of **53.72%**, which is less than expected, which leads to question wether that lambda was good enough or if the search required more epochs to provide a clearer idea of how good each lambda was.

```
Epoch: 49        cost: 2.30829186711376  train_acc: 0.6282       val_acc: 0.5446
best lambda final score 0.5372
```

Include the loss plots for the training with Batch Norm Vs no Batch Norm for the experiment related to Sensitivity to initialization and comment on your experimental findings.

In the following table are reported the graphs of the loss function of the same network with 2 hidden layers of size 50 and the following parameters, but with different sigmas. `lambda=0.0000075, epochs=20, n_batch=100, eta={ 'min': 1e-5, 'max': 1e-1, 'step_size': 2250, 'l': 0 }`

Sigma	Without Batch norm	With Batch Norm
1e-1		
1e-3		
1e-4		

The results seem to indicate that batch normalization could have a significant impact when learning a deeper neural network model in terms of the testing accuracy. Batch normalization also renders the netw. Whereas without batch normalization the performance is highly sensible of the initialization.