DD2424 Assignment 2 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]))}")
```

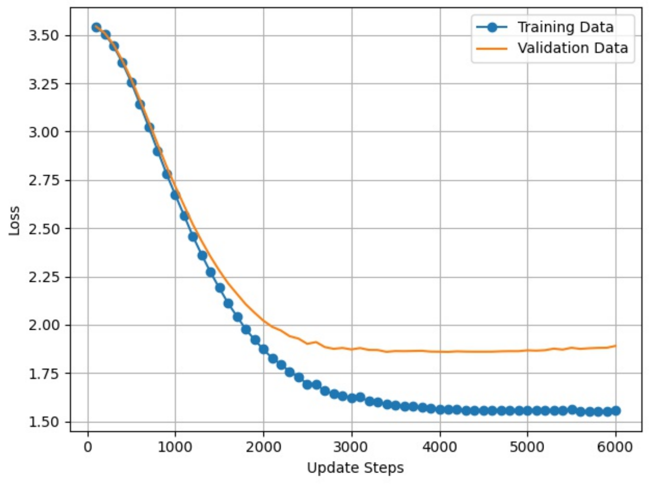
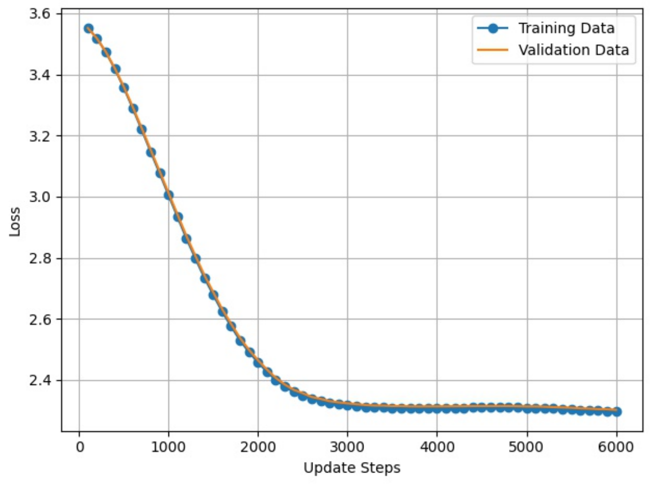
Which outputs

```
an-done
num-done
W diff:0.00020043066871582837    b diff:0.000804371882177268
W diff:4.4755865680201624e-18    b diff:1.9428902930940238e-17
```

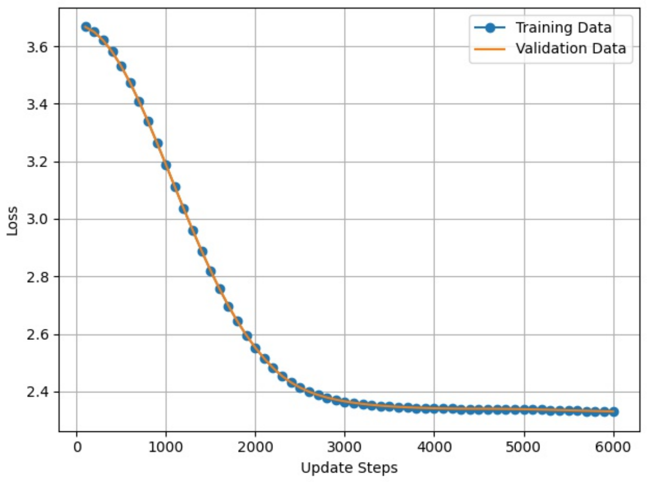
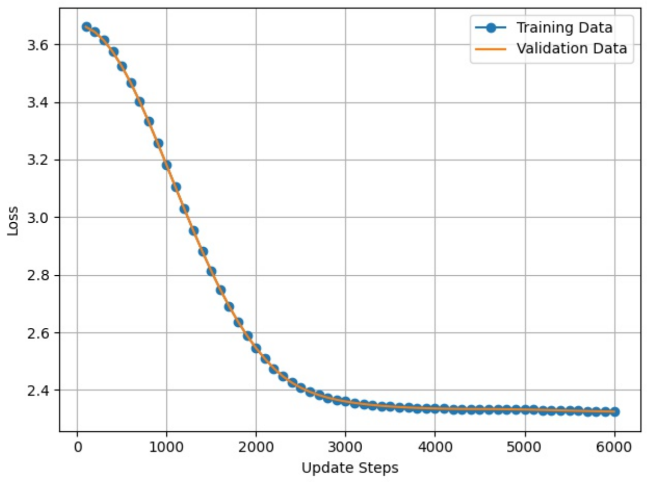
and suggests that the computations are correct.

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.

The range for the lambda in the coarse search starts from 1e-5 and ends in 1e-1. 10 random samples were taken from that interval and 10 networks were trained for 10 epochs (in order to speed up the computation).

```
lambda:0.04980332046585267 score:0.3143
lambda:0.05046101097065926 score:0.3404
lambda:0.09701967160288724 score:0.3087
lambda:0.04392942130309915 score:0.3364
lambda:0.0672233571505394 score:0.3325
lambda:0.01247904308463892 score:0.3161
lambda:0.08515533431301502 score:0.2896
lambda:0.06846646651118832 score:0.3187
lambda:0.060436091427891914 score:0.3292
lambda:0.0040241459901606234 score:0.3099
```

The two best lambdas found in this interval were `0.05046101097065926` and `0.09701967160288724`.

Another search was done between the two best values found. So 10 more random samples were taken and 10 more networks were trained with such values.

```
lambda:0.07888325183902706 score:0.3293
lambda:0.05855737292594083 score:0.2823
lambda:0.08132352700007156 score:0.3254
lambda:0.08902879485497511 score:0.3314
lambda:0.07746498180078475 score:0.346
lambda:0.09024130828007082 score:0.3241
lambda:0.0790970568716708 score:0.317
lambda:0.05091691748827633 score:0.3291
lambda:0.06960009744694348 score:0.3513
lambda:0.09106901150655683 score:0.3337
```

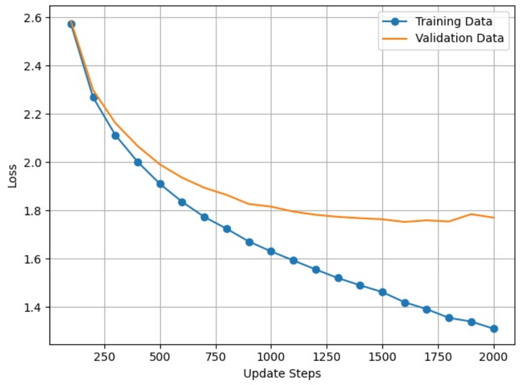
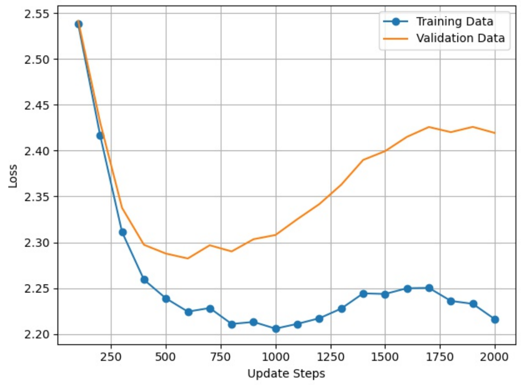
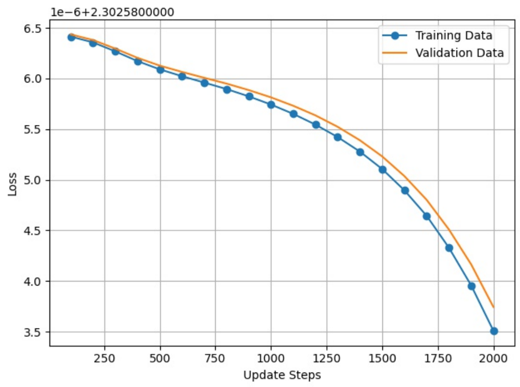
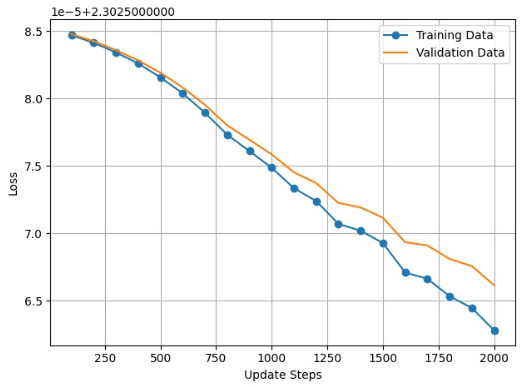
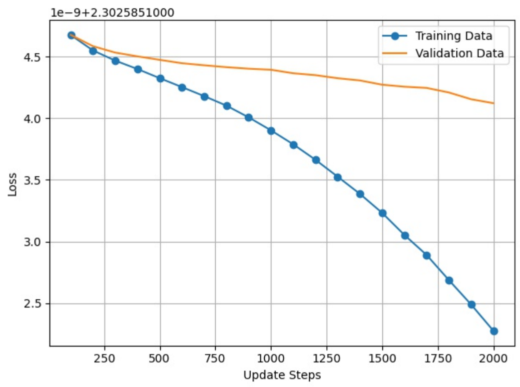
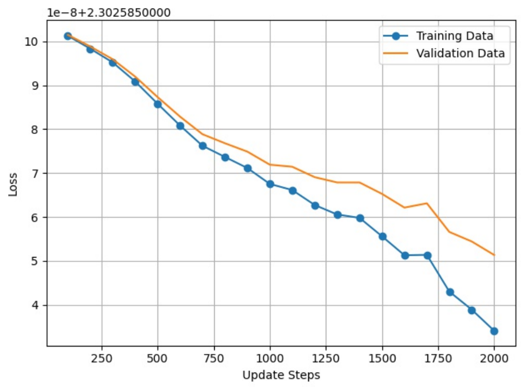
The best lambda found by this finer search was `0.06960009744694348`

And training the network for 50 epochs reports a final test score of **35.7%**, 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: 0.4118       val_acc: 0.3596
best lambda final score 0.357
```

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 even if the loss is higher when using batch normalization, batch normalization keeps the final performance (in terms of accuracy) of the network almost the same even for different initialized parameters. Whereas without batch normalization the performance is highly sensible of the initialization.