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 were bug free. Give evidence for these conclusions.

Even if the code was almost completely restructured since assignment 1, the technique to verify the accuracy of the gradient did not change.

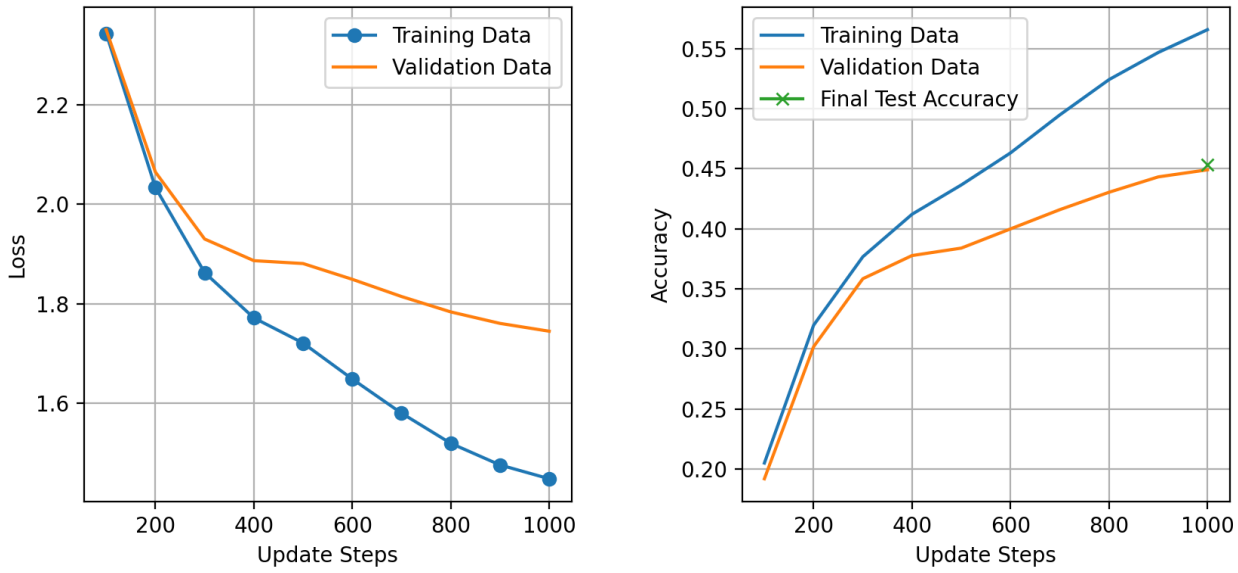
The absolute difference between the mean of the weights and the biases.

	Absolute difference between mean
w1	5.464830700232145e-18
w2	9.934456791901010e-12
b1	2.00264830732145e-8
b2	1.72485503754268e-10

The low values obtained seem to confirm that the analytical computation is correct.

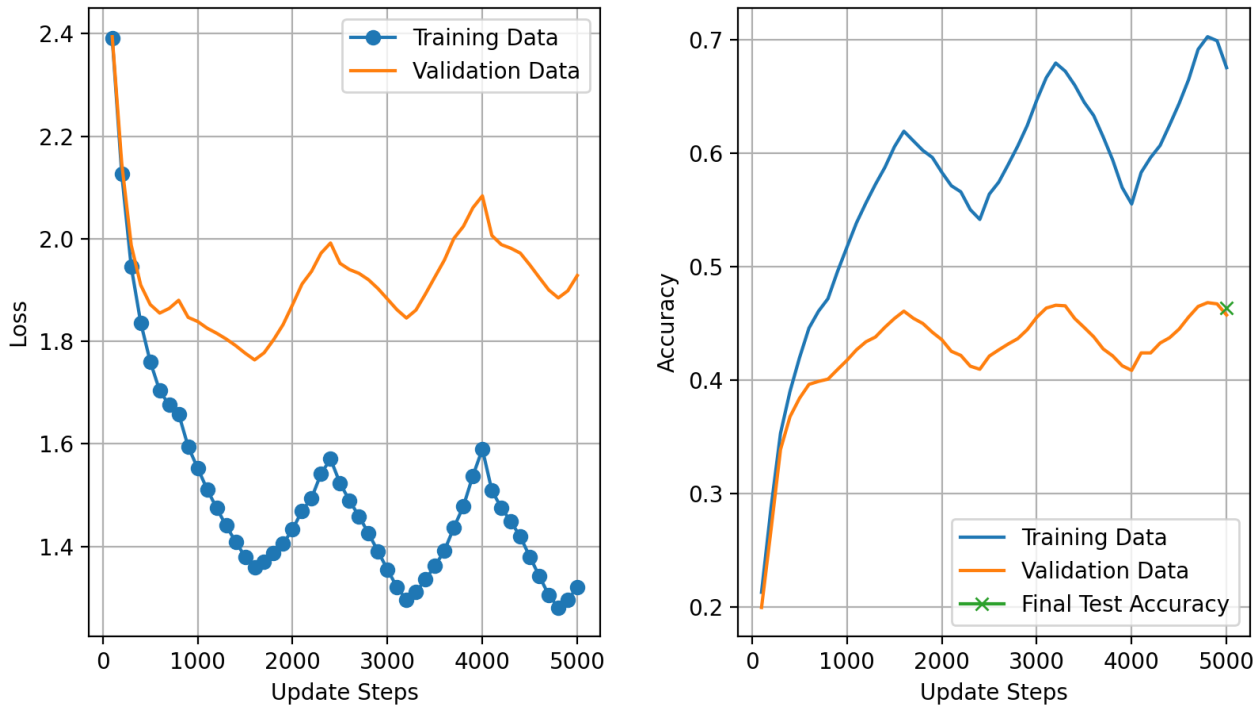
The curves for the training and validation loss/cost when using the cyclical learning rates with the default values, that is replicate figures 3 and 4. Also comment on the curves.

Figure 3



The figure looks similar to figure 3 of the Assignment 2 PDF. But I do not understad why there is a different plot for the loss. I assumed loss is the same as cost.

Figure 4



The figure looks similar to figure 4 of the Assignment 2 PDF.

State the range of the values you searched for lambda, the number of cycles used for training during the coarse search and the hyper-parameter settings for the 3 best performing networks you trained.

The range of values searched ranged from

l_min = -5 l_max = -1

10 networks were trained with the following hyperparameters:

epochs=20, n_batch=100, eta={ 'min': 1e-5, 'max': 1e-1, 'step_size': 800, }

```
l: -1.8947090150056005 lambda: 0.0127435663655190 9    score: 0.4656
l: -2.164243963715208  lambda: 0.0068510326348737135  score: 0.4673
l: -2.9341172686161867 lambda: 0.001163811733408894   score: 0.4561
l: -1.1141916982418971 lambda: 0.07687910198615475    score: 0.3977
l: -2.7332280070573325 lambda: 0.0018482979973155457  score: 0.4604
l: -3.9348257015805768 lambda: 0.00011619148396262135 score: 0.4495
l: -1.9064842048684607 lambda: 0.012402687324190004   score: 0.4713
l: -2.1019485333286476 lambda: 0.007907723340221424   score: 0.4698
l: -2.0408834142522307 lambda: 0.009101575703700678   score: 0.4727
l: -1.6196449492702847 lambda: 0.024007948495962123   score: 0.4527
```

Out of those, the three best performing networks were

```
l: -2.0408834142522307 lambda: 0.009101575703700678   score: 0.4727
l: -1.9064842048684607 lambda: 0.012402687324190004   score: 0.4713
l: -2.1019485333286476 lambda: 0.007907723340221424   score: 0.4698
```

State the range of the values you searched for lambda, the number of cycles used for training during the fine search, and the hyper-parameter settings for the 3 best performing networks you trained.

Following the results obtained in the previous step the search was restricted to the range of the three best performing networks:

l_min = -2.1 l_max = -1.9

10 more networks were trained with the same hyperparameters as before.

```
l: -2.0757288478556335 lambda: 0.008399842677560622   score: 0.4776
l: -2.021510833000273 lambda: 0.009516761092984709    score: 0.4774
l: -1.9525949158935823 lambda: 0.011153343662429453   score: 0.4755
l: -2.0232425894564736 lambda: 0.009478888409300773   score: 0.473
l: -2.02293884881013 lambda: 0.009485520155408765     score: 0.4713
l: -1.9365410798507732 lambda: 0.011573345548103298   score: 0.4704
l: -2.031527029450491 lambda: 0.00929978633305501     score: 0.47
l: -1.9600301468773702 lambda: 0.010964020859280207   score: 0.4696
l: -2.0359721428917643 lambda: 0.009205086143769758   score: 0.4673
l: -1.900981625802063 lambda: 0.012560831051284907    score: 0.4671
```

The best performing network was

```
l: -2.0757288478556335 lambda: 0.008399842677560622   score: 0.4776
```

We got a 5% improvement in the accuracy.

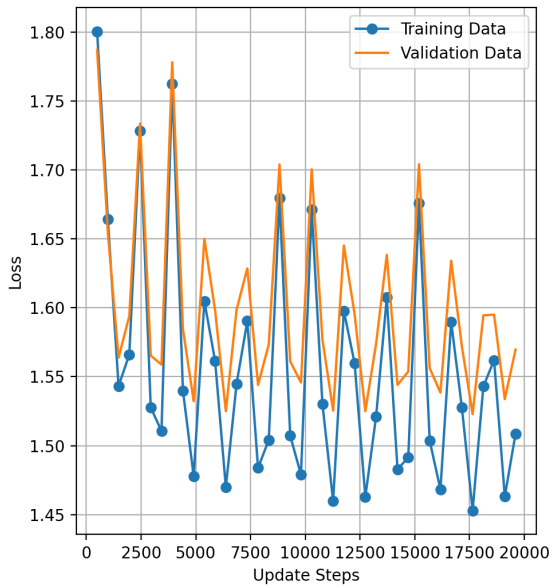
For your best found lambda setting (according to performance on the validation set), train the network on all the training data (all the batch data), except for 1000 examples in a validation set, for ~3 cycles. Plot the training and validation loss plots and then report the learnt network's performance on the test data.

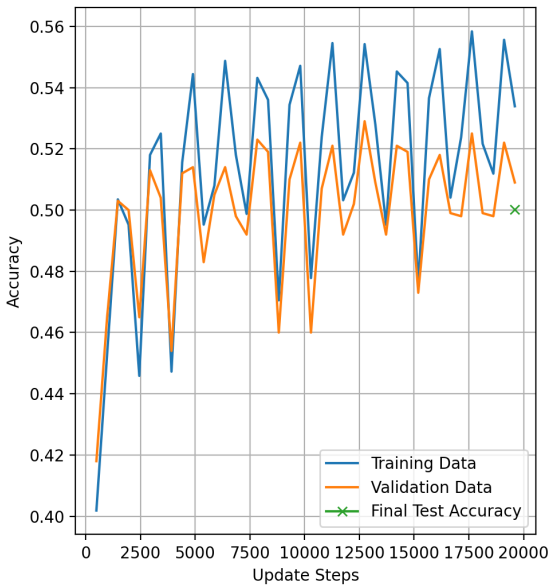
The final network was trained with the following parameters:

```
nnt = NeuralNet((train_X, train_Y, train_labels), mu, sigma, hidden_nodes=50)
nnt.MiniBatchGD((val_X, val_Y, val_labels), (test_X, test_Y, test_labels),
    _lambda=0.008399842677560622,
    epochs=40,
    n_batch=100,
    eta={
        'min': 1e-5,
        'max': 1e-1,
        'step_size': 800,
        'l': -2.0757288478556335
})
```

In the last epoch the network achieved the following Loss, train accuracy and validation accuracy:

```
Epoch: 39        cost: 1.5086114097675993        train_acc: 0.5338979591836734   val_acc: 0.509
```





In the accuracy graph in the left we can also see the final test accuracy that is ~50%.