Course: DD2424 - Assignment 3

Sevket Melih Zenciroglu – [smzen@kth.se](mailto:smzen@kth.se)

# **Q1: Gradient Computations**

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.

# **A1:**

I calculated the gradients numerically as it was suggested in the assignment and compared these values with the ones I derived analytically. The MEAN of the differences for W and b values were smaller than 1e-7 which means that the error-rate is very small. According to the reference given from [Standford](https://cs231n.github.io/neural-networks-3/), having error values smaller than 1e-7 should make us happy.

Gradient discrepancies for W changes significantly from one layer to another. The discrepancy is higher at lower layers. There are small discrepancy changes for gradients of gammas and betas, higher layers have slightly lower values. However, discrepancy for gradients of b doesn’t follow the same pattern.

Below, you can see the parameters and their values used for these tests. Also, you can see the discrepancy values of gradients for different types and networks layer by layer.

|  |  |
| --- | --- |
| Parameter | Value |
| lambda\_cost | 0 |
| Number of images used | 3 |
| Number of dimensions used | 10 |
| h | 1e-5 |
| Mu (for W initialization) | 0 |
| Hidden nodes | [50], [50, 50], [50, 50, 30] |
| Sigma (for W initialization) | 1e-2 |
| Batch Normalization | True |

**Table-1:** Parameters used for Gradient Computations

Below tables show the mean gradient discrepancies (grad\_numerical – grad\_analytic) for different networks when Batch Normalization is used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer | grad\_W | grad\_b | grad\_gamma | grad\_beta |
| 1 | 3.4653659610592165e-07 | 2.6428512156506652e-17 | 1.1252590192370893e-11 | 1.440571455689748e-11 |
| 2 | 1.0910545735105803e-11 | 9.396558531271637e-12 |  |  |

**Table-2:** Discrepancy between numerical and analytic Gradients – 2-layer Network [50]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer | grad\_W | grad\_b | grad\_gamma | grad\_beta |
| 1 | 2.7389059050345976e-07 | 2.9178139216110936e-17 | 1.1117855059672475e-11 | 1.5014044591480967e-11 |
| 2 | 3.363769016160618e-09 | 1.0639637319324416e-17 | 1.309968732703874e-11 | 1.1983954365492256e-11 |
| 3 | 1.1286712148228518e-11 | 1.2367763757570317e-11 |  |  |

**Table-3:** Discrepancy between numerical and analytic Gradients – 3-layer Network [50, 50]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer | grad\_W | grad\_b | grad\_gamma | grad\_beta |
| 1 | 4.776375175642611e-06 | 1.3322856359798683e-12 | 6.081038442760922e-11 | 3.8791430904050114e-11 |
| 2 | 4.888949678919275e-07 | 1.3322765924688474e-12 | 1.3516458607154216e-11 | 1.1974045570524788e-11 |
| 3 | 5.8949647016560174e-08 | 7.403096316912741e-13 | 1.30570704042713e-11 | 1.0302963083380633e-11 |
| 4 | 1.0410893831053204e-11 | 1.5343132320211338e-11 |  |  |

**Table-4:** Discrepancy between numerical and analytic Gradients – 4-layer Network [50, 50, 30]

# **Q2: 3-layer Network with & without BN**

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.

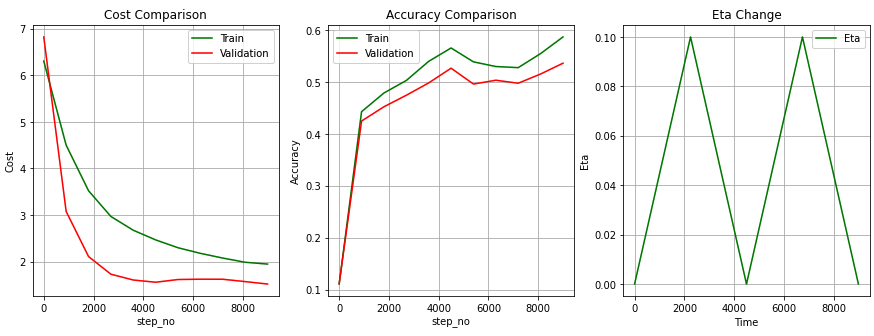
# **A2:**

|  |  |
| --- | --- |
| Parameter | Value |
| n\_batch\_size | 100 |
| n\_cycles | 2 |
| lambda\_cost | 0.005 |
| eta\_min | 1e-5 |
| eta\_max | 1e-1 |
| Hidden nodes | [50, 50] |
| k\_cyckle | 5 (n\_s = 5 \* 45000 / n\_batch) |
| h | 1e-5 |
| init\_type | ‘He’ |
| alpha | 0.9 |
| plot\_points | 10 |
| BN | False & True |

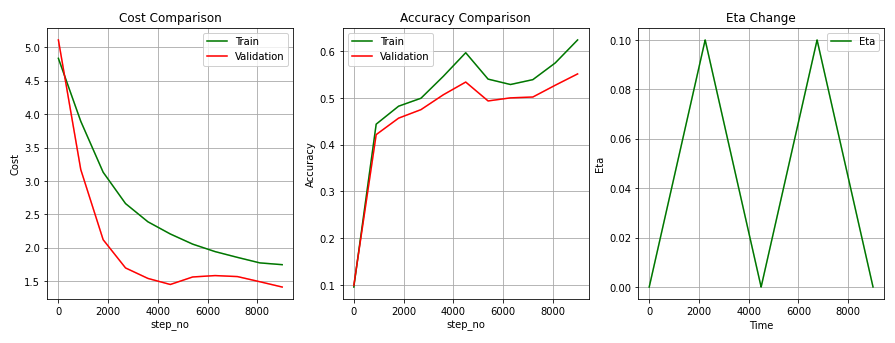
**Table-5:** Parameters used for 3-layer network accuracy computations

|  |  |  |  |
| --- | --- | --- | --- |
| Network/Accuracy (%) | Training | Validation | Test |
| Without Batch Normalization | 58.73 | 53.66 | 52.96 |
| WITH Batch Normalization | 62.4 | 55.14 | 53.71 |

**Table-6:** Accuracy results for 3-layer network



**Picture-1:** 3-layer Network – Without BN



**Picture-2:** 3-layer Network – WITH BN

**NOTE:** You can find the results for Xavier Initialization in the [App2: 3-layer network with Xavier Initialization](#_App2:_3-layer_network).

# **Q3: 9-layer Network with & without BN**

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.

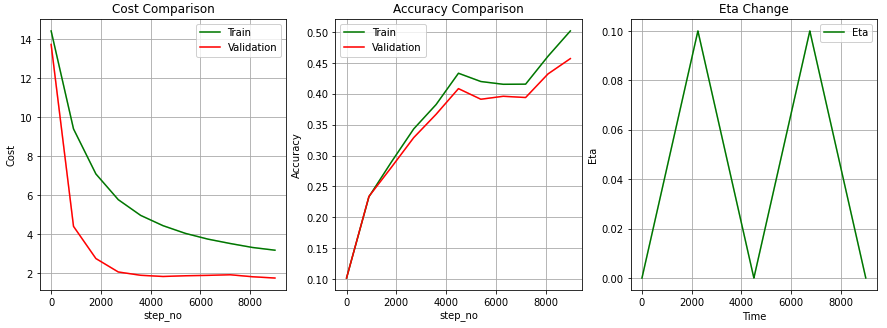
# **A3:**

|  |  |
| --- | --- |
| Parameter | Value |
| n\_batch\_size | 100 |
| n\_cycles | 2 |
| lambda\_cost | 0.005 |
| eta\_min | 1e-5 |
| eta\_max | 1e-1 |
| Hidden nodes | [50, 30, 20, 20, 10, 10, 10, 10] |
| k\_cyckle | 5 (n\_s = 5 \* 45000 / n\_batch) |
| h | 1e-5 |
| init\_type | ‘He’ |
| alpha | 0.9 |
| plot\_points | 10 |
| BN | False & True |

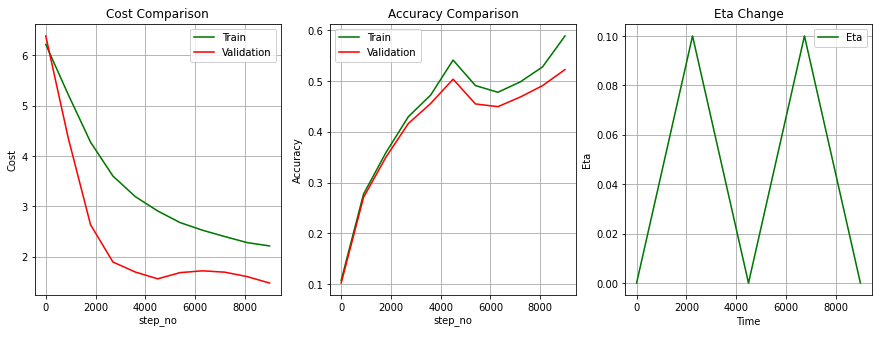
**Table-7:** Parameters used for 9-layer network accuracy computations

|  |  |  |  |
| --- | --- | --- | --- |
| Network/Accuracy (%) | Training | Validation | Test |
| Without Batch Normalization | 50.18 | 45.7 | 45.98 |
| WITH Batch Normalization | 58.91 | 52.28 | 52.02 |

**Table-8:** Accuracy results for 9-layer network



**Picture-3:** 9-layer Network – Without BN



**Picture-4:** 9-layer Network – WITH BN

**NOTE:** You can find the results for Xavier Initialization in the [App3: 9-layer network with Xavier Initialization.](#_App3:_9-layer_network)

# **Q4: Lambda Search**

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.

# **A4:**

**Note**: All above parameters are used as is, only Xavier initialization is used by mistake instead of He, but since it doesn’t affect the results much, tests were not repeated with He.

Firstly, a coarse search was applied for values between 1e-5 and 1e-1 by picking 10 random values uniformly distributed in this range.

Secondly, another coarse search was applied by narrowing down the range between 1e-2.9612 and 1e-1.9586

|  |  |
| --- | --- |
| **Coarse Search-1** | **Coarse Search-2** |
| |  |  | | --- | --- | | Lambda | Accuracy | | 0.07551817375745834 | 0.4993 | | 0.049722550077803684 | 0.5092 | | 0.00031843718683816725 | 0.5186 | | 4.084160858149218e-05 | 0.5193 | | 4.879263419455391e-05 | 0.5194 | | 0.024359159583172595 | 0.5202 | | 1.3857911624979211e-05 | 0.5233 | | 0.0009942565088993363 | 0.5242 | | 0.0010933478237383542 | 0.5274 | | 0.0035621657857409636 | 0.5347 | | |  |  | | --- | --- | | Lambda | Accuracy | | 0.0016268068655851079 | 0.5259 | | 0.00118663335141119 | 0.5262 | | 0.0015558686373988403 | 0.5268 | | 0.0034631634130629385 | 0.5291 | | 0.010252590303797503 | 0.5298 | | 0.0026033521015278852 | 0.5313 | | 0.00923296327600833 | 0.5314 | | 0.0035466209826920114 | 0.532 | | 0.007720881832204912 | 0.5335 | | 0.004768558843785876 | **0.5395** | |

**Table-9:** Coarse search results

Then a Fine Search was applied for the lambda values between 0.0038 and .0.0081, some of the result are listed below:

|  |
| --- |
| **Fine Search** |
| |  |  | | --- | --- | | Lambda | Accuracy | | 0.0045960540264639435 | 0.5308 | | 0.006158467929334429 | 0.5323 | | 0.0057678644536168075 | 0.5329 | | 0.0069396748807696714 | 0.5333 | | 0.007330278356487293 | 0.5334 | | 0.004986657502181565 | 0.5346 | | 0.005499999999999998 | 0.5348 | | 0.005699999999999998 | 0.5371 | | 0.004205450550746322 | 0.5381 | | 0.005099999999999999 | 0.5393 | |

**Table-10:** Fine search results

Overall, the best result was observed when lambda equals to **0.004768558843785876** with the accuracy of **53.95%.**

# **Q5: Sensitivity to Initialization**

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.

# **A5: Comments**

When the tests are done on a 3-layer network, it is observed that the network with Batch Normalization (BN) is not affected much from the initialization of the parameters, it is more stable. However, without BN, the network is more sensitive to the initialization of the parameters.

As an additional information, until 7-layer network, the accuracy drops gradually without BN and it is stuck at 10% when sigma 1e-1 is used. (e.g., 50, 50, 20]: 50.38%, [50, 50, 20, 20]: 48.28%, [50, 50, 20, 20, 10]: 39.45%, [50, 50, 20, 20, 10, 10]: 10.00). Yet, as you can see from the below results, even for a 3-layer network, having sigma as 1e-3 or 1e-4 results in a huge decrease in the accuracy.

When the same tests applied on a 9-layer network, this time the sensitivity of the network increased even with BN. On the other hand, if it is given enough cycles, the accuracy starts to increase, the network with BN starts to recover by the time with the increasing number of training steps. At [App4: Change for different cycles](#_App4:_Change_for) can be visited to see how cost and accuracy change once the number of cycles is increased from 2 to 3.

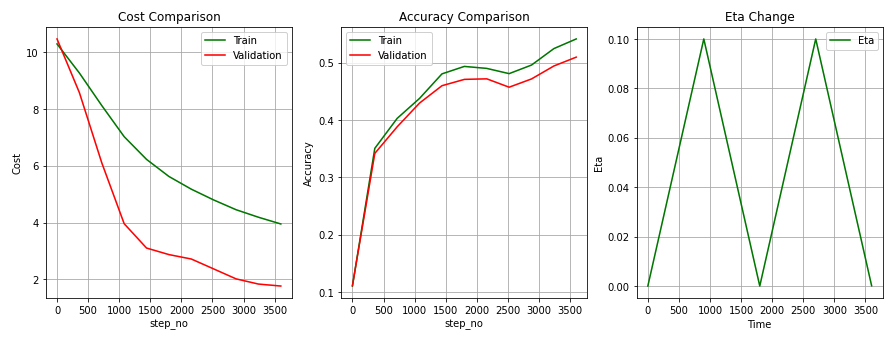
# **A5: For 3-layer network**

|  |  |
| --- | --- |
| Parameter | Value |
| n\_batch\_size | 100 |
| n\_cycles | 2 |
| lambda\_cost | 0.005 |
| eta\_min | 1e-5 |
| eta\_max | 1e-1 |
| Hidden nodes | [50, 50] |
| k\_cyckle | 5 (n\_s = 5 \* 45000 / n\_batch) |
| h | 1e-5 |
| init\_type | Fixed sigma values: 1e-1, 1e-3, 1e-4 |
| alpha | 0.9 |
| plot\_points | 10 |
| BN | False & True |

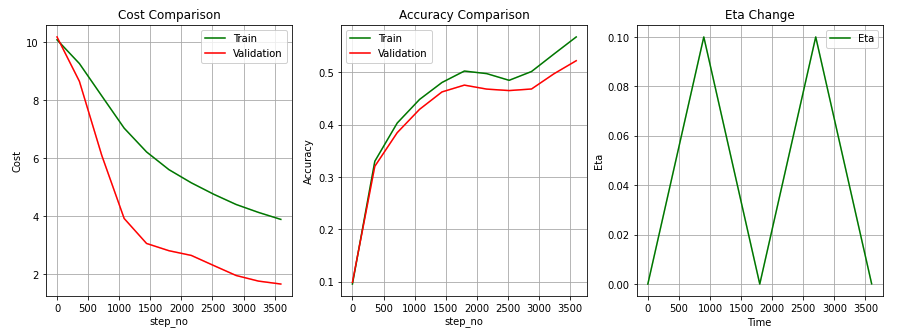
**Table-11:** Parameters used for Sensitivity Initialization network accuracy computations (3-layer Network)

|  |  |  |  |
| --- | --- | --- | --- |
| Network/Sigma | 1e-1 | 1e-3 | 1e-4 |
| Without Batch Normalization | 51.19 | 10.00 | 10.00 |
| WITH Batch Normalization | 51.4 | 55.26 | 55.32 |

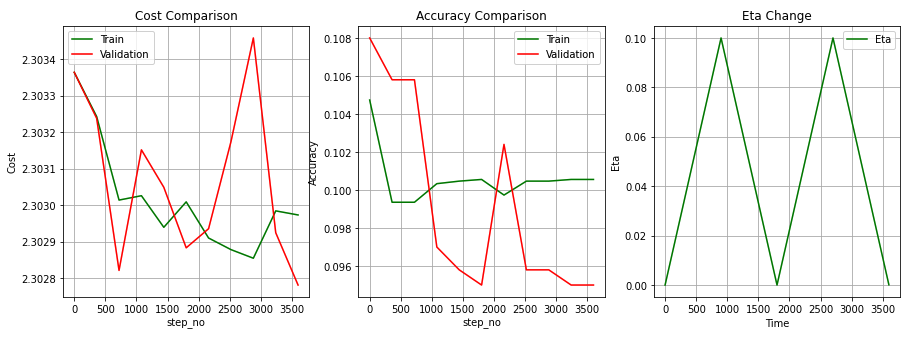
**Table-12:** Accuracy results for Sensitivity Initialization (3-layer Network)



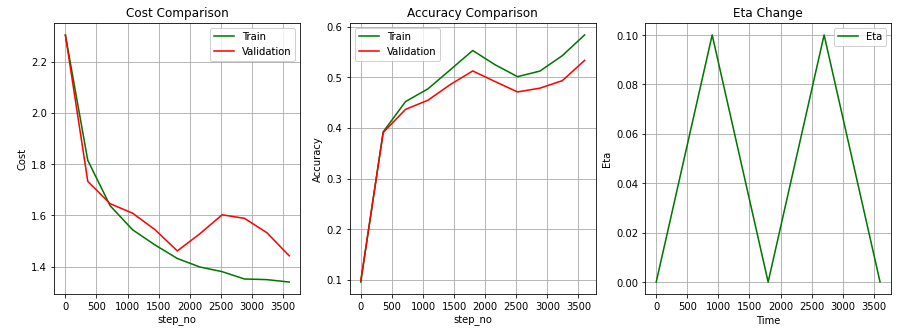
**Picture-5:** Sigma = 1e-1 – Without BN >> 3-layer



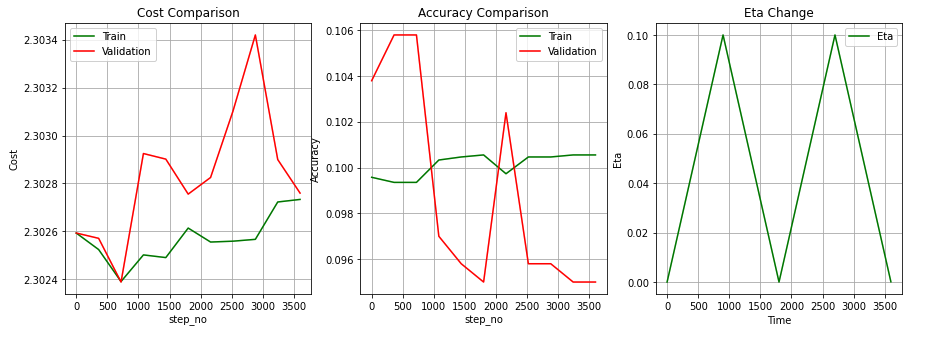
**Picture-6:** Sigma = 1e-1 – WITH BN >> 3-layer



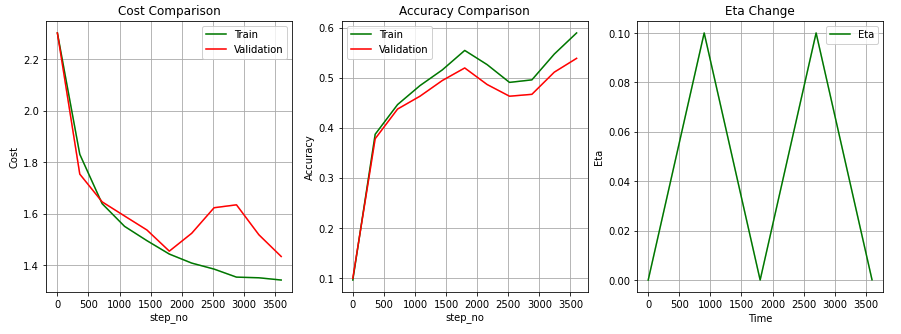
**Picture-7:** Sigma = 1e-3 – Without BN >> 3-layer



**Picture-8:** Sigma = 1e-3 – WITH BN >> 3-layer



**Picture-9:** Sigma = 1e-4 – Without BN >> 3-layer



**Picture-10:** Sigma = 1e-4 – WITH BN >> 3-layer

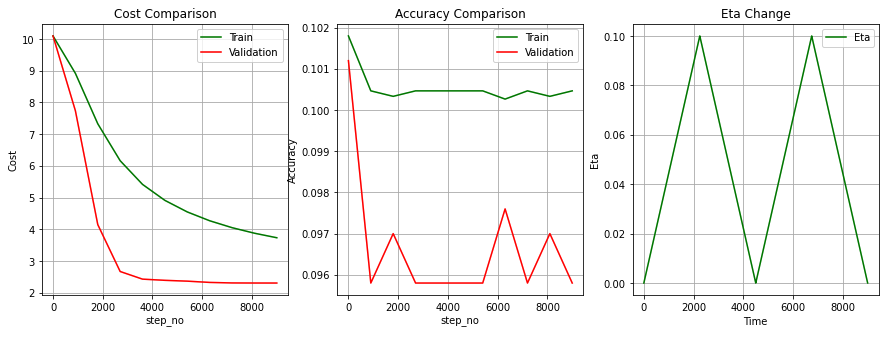
# **A5: For 9-layer network**

|  |  |
| --- | --- |
| Parameter | Value |
| n\_batch\_size | 100 |
| n\_cycles | 2 |
| lambda\_cost | 0.005 |
| eta\_min | 1e-5 |
| eta\_max | 1e-1 |
| Hidden nodes | [50, 30, 20, 20, 10, 10, 10, 10] |
| k\_cyckle | 5 (n\_s = 5 \* 45000 / n\_batch) |
| h | 1e-5 |
| init\_type | Fixed sigma values: 1e-1, 1e-3, 1e-4 |
| alpha | 0.9 |
| plot\_points | 10 |
| BN | False & True |

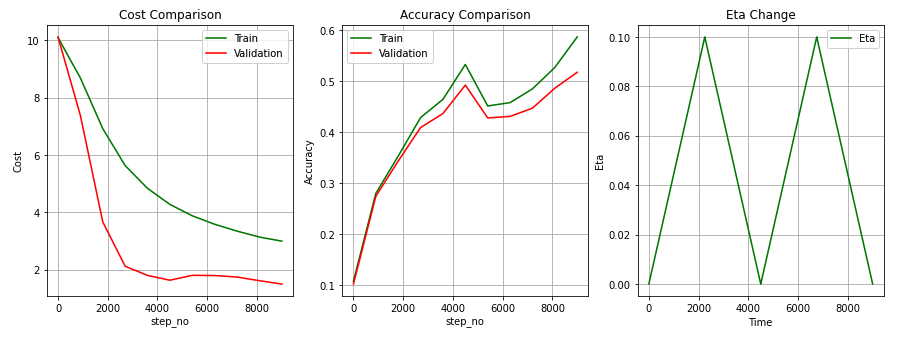
**Table-13:** Parameters used for Sensitivity Initialization network accuracy computations (9-layer Network)

|  |  |  |  |
| --- | --- | --- | --- |
| Network/Sigma | 1e-1 | 1e-3 | 1e-4 |
| Without Batch Normalization | 10.00 | 10.00 | 10.00 |
| WITH Batch Normalization | 51.98 | 17.37 | 33.75 |

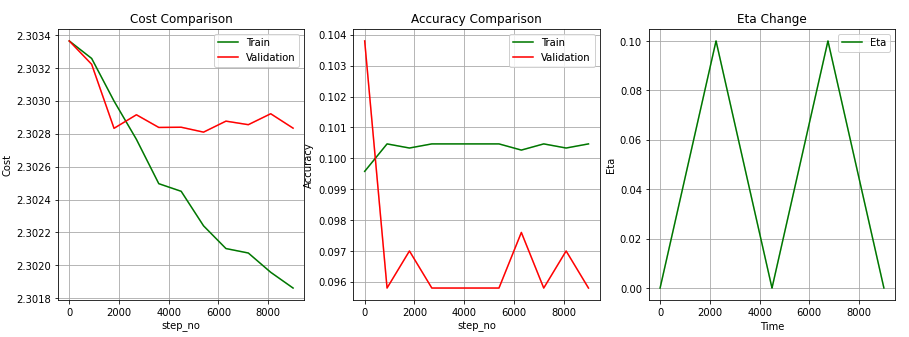
**Table-14:** Accuracy results for Sensitivity Initialization (9-layer Network)



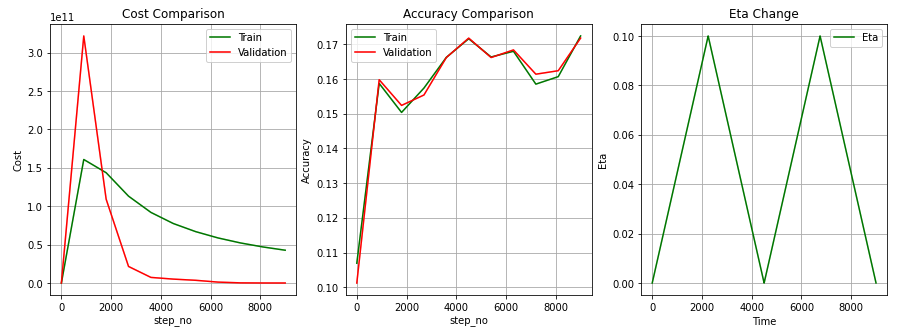
**Picture-11:** Sigma = 1e-1 – Without BN >> 9-layer



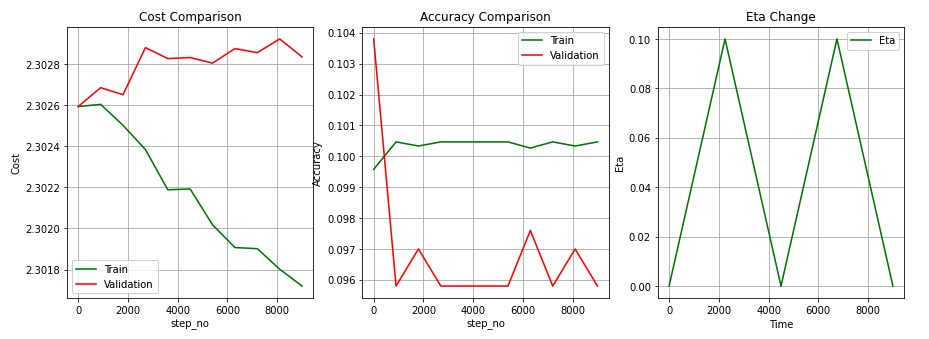
**Picture-12:** Sigma = 1e-1 – WITH BN >> 9-layer



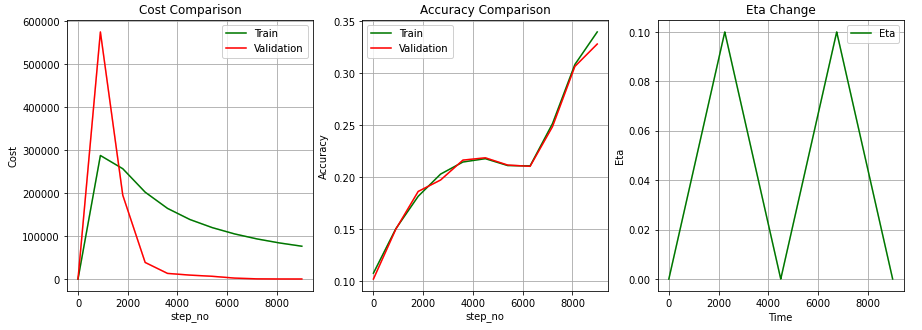
**Picture-13:** Sigma = 1e-3 – Without BN >> 9-layer



**Picture-14:** Sigma = 1e-3 – WITH BN >> 9-layer



**Picture-15:** Sigma = 1e-4 – Without BN >> 9-layer



**Picture-16:** Sigma = 1e-4 – WITH BN >> 9-layer

# **Appendix**

## App1: Parameter Descriptions

Below, you can find the parameter descriptions which are used for the assignment.

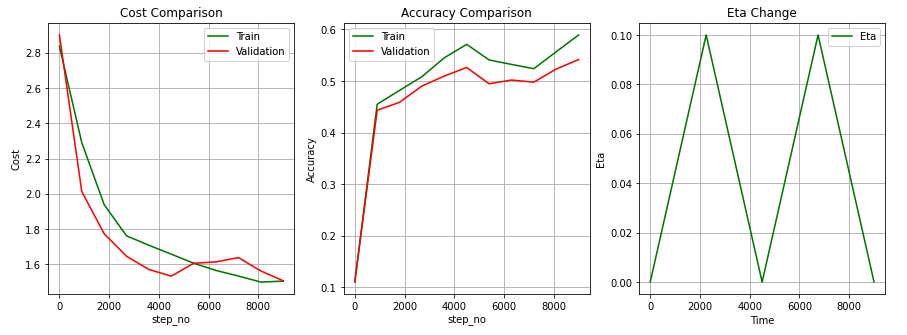
|  |  |  |
| --- | --- | --- |
| Parameter | Value (Sample) | Description |
| n\_batch\_size | 100 | Size of the mini batch. In other words, the number of images in 1 mini-batch. |
| n\_cycles | 2 | learning rate (step-size) |
| lambda\_cost | 0.005 | regularization coefficient (punishment) |
| eta | Min: 1e-5 Max: 1e-1 | learning rate (step-size) |
| Hidden nodes | [50, 50] | Hidden nodes used (hidden\_nodes) |
| k\_cyckle | 5 (n\_s = 5 \* 45000 / n\_batch) | coefficient used in cycling learning calculations |
| h | 1e-5 | Precision value |
| init\_type | ‘He’, ‘Xavier’, 1e-2 | Parameter initialization for W |
| alpha | 0.9 | Exponential moving average coefficient |
| plot\_points | 10 | Number of samples to plot the graph |
| BN | False or True | With or without Batch Normalization |

**Table-15:** Parameters used for Gradient Computations

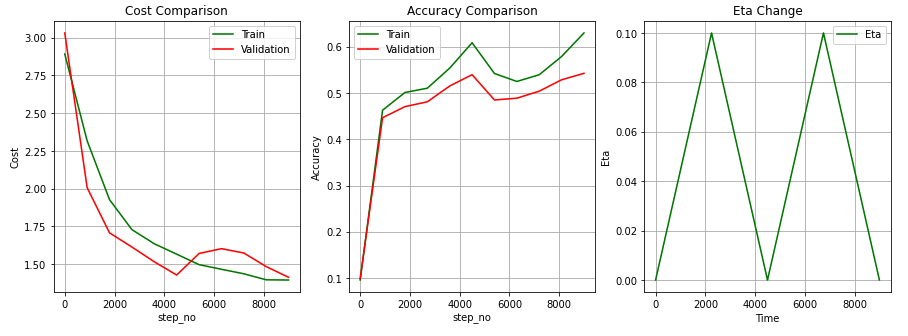
## App2: 3-layer network with Xavier Initialization

|  |  |  |  |
| --- | --- | --- | --- |
| Network/Accuracy (%) | Training | Validation | Test |
| Without Batch Normalization | 58.92 | 54.14 | 53.24 |
| WITH Batch Normalization | 62.99 | 54.26 | 53.43 |

**Table-16:** Accuracy results for 3-layer network (Xavier)



**Picture-17:** 3-layer Network – Without BN (Xavier)

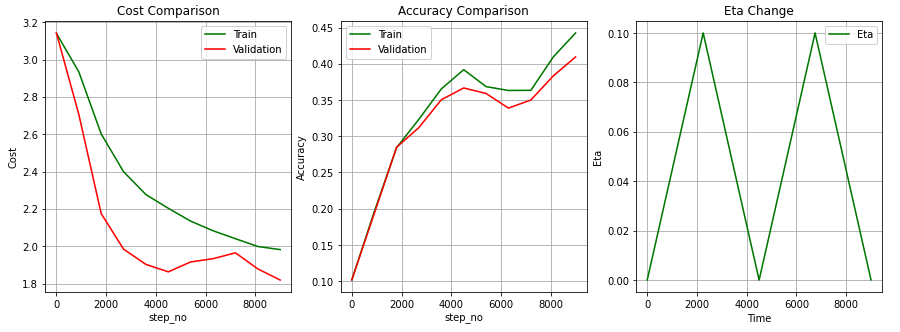


**Picture-18:** 3-layer Network – WITH BN (Xavier)

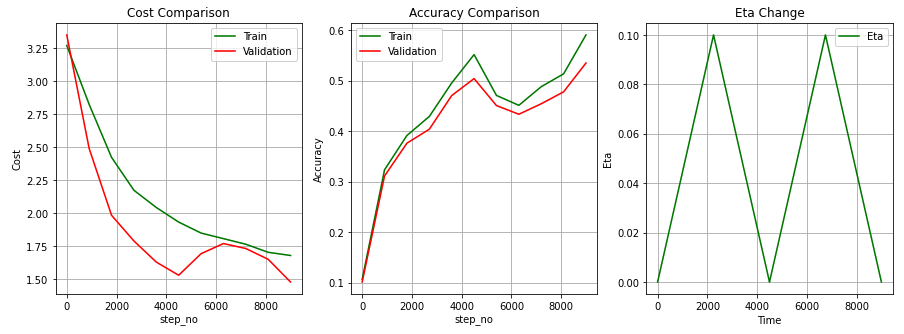
## App3: 9-layer network with Xavier Initialization

|  |  |  |  |
| --- | --- | --- | --- |
| Network/Accuracy (%) | Training | Validation | Test |
| Without Batch Normalization | 44.25 | 40.94 | 40.71 |
| WITH Batch Normalization | 59.06 | 53.5 | 51.79 |

**Table-17:** Accuracy results for 9-layer network (Xavier)



**Picture-19:** 9-layer Network – Without BN (Xavier)



**Picture-20:** 9-layer Network – WITH BN (Xavier)

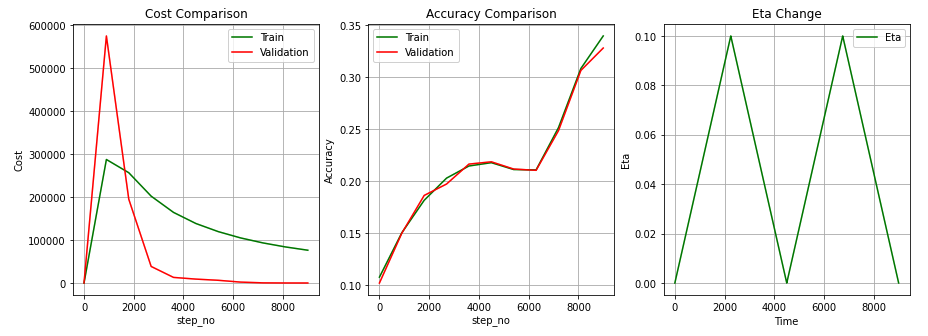
## App4: Change for different cycles

9-layer Network with BN, alpha=1e-4, using:

2-cycles (9000 training steps)

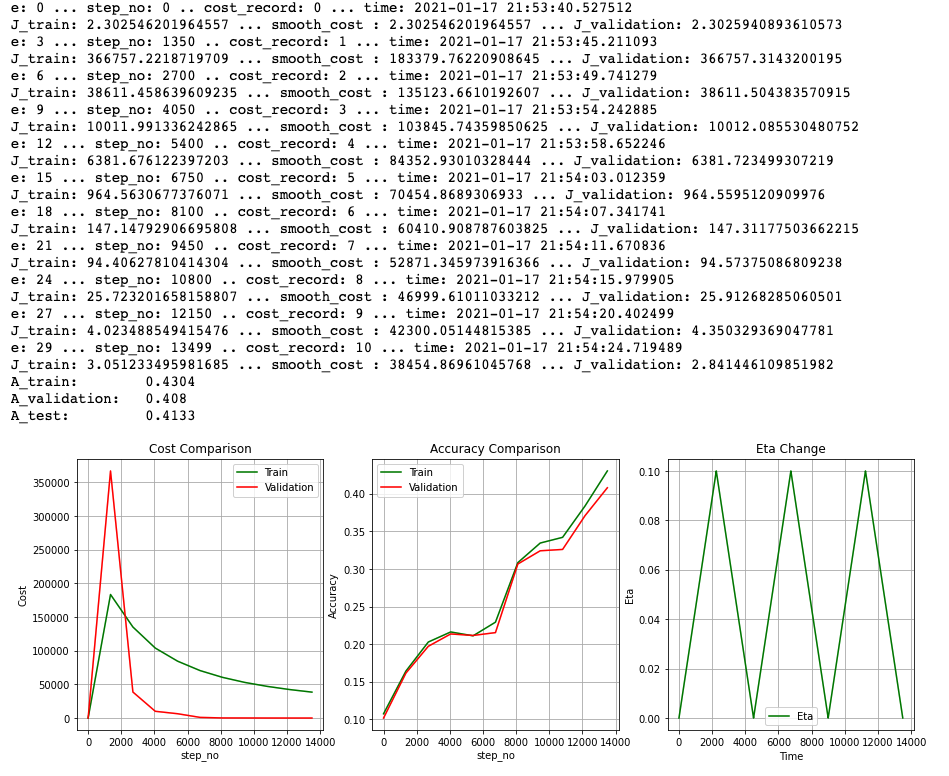
A screenshot of a computer

Description automatically generated with low confidence



**Picture-21:** 9-layer Network – WITH BN, alpha=1e-4, 2-cycle training

3-cycles (13500 training steps)



**Picture-22:** 9-layer Network – WITH BN, alpha=1e-4, 3-cycle training