Deep Learning in Data Science, DD2424

Short report on lab assignment 3
k-layer neural network with multiple outputs

Hamid Dimyati

1. Introduction

The goal of this lab assignment 3 is to train and test a *k*-layer neural network to classify multiple classes of CIFAR-10 datasets. The network applies cross-entropy loss function and L_2 regularization with parameters learning using mini-batch gradient descent.

The tool used for this assignment is primarily `python 3.7.3`, along with several packages including `numpy 1.16.4`, `pandas 0.25.3`, and `matplotlib 3.1.0`.

2. Results and discussion

2.1. Gradient checking

Before start building the network, a standardization (Z-score normalization) is performed to all of the datasets. To validate whether the self-build codes, particularly the function to compute the gradient of parameters $\boldsymbol{W} = \{W_1, W_2, \dots, W_k\}$, $\boldsymbol{b} = \{b_1, b_2, \dots b_k\}$, $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \dots, \gamma_{k-1}\}$, and $\boldsymbol{\beta} = \{\beta_1, \beta_2, \dots \beta_{k-1}\}$ could run appropriately based on the mathematical functions, comparing the results with alternative ways using the numerical approach as provided in the instruction documents is necessary. The maximum and average values of the absolute tolerance/error of each parameter are generated to investigate whether the difference between those two outputs is still tolerable, which follows the equation (1):

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)} \approx absolute\ tolerance \qquad (1)$$

Using my codes (g_a), I got results of comparing the absolute tolerance between the model without and with batch normalization, for the 2-layer networks until 4-layer networks, as summarized in Table 1. It can be seen that the difference in the earlier layers is increasing, and specifically, the $\boldsymbol{b}$ parameter in the model with batch normalization grow up significantly. Overall, my codes are able to approximate the results of numerical calculation within a tolerable gap. Training 2-layer networks until 4-layer networks on 100 samples of the training data with regularization turned off (lambda=0), η equal to $1e-1$, 200 epochs, and 50 hidden nodes for each layer, results in minimal cost value as depicted in Figure 1.

Table 1. Hyperparameter settings for two models

Architecture	Gradient	Relative error without Batch Normalization		Relative error with Batch Normalization	
		Maximum	Mean	Maximum	Mean
2-layer (50)	W_1	1.445e-5	3.547e-7	2.749e-4	1.795e-6
	b_1	6.830e-6	5.268e-7	7.500e-2	1.848e-2
	γ_1	n/a	n/a	2.394e-6	2.484e-7
	β_1	n/a	n/a	3.014e-6	2.873e-7
	W_2	8.523e-6	3.098e-7	8.864e-5	1.018e-6
	b_2	6.642e-7	3.247e-7	2.084e-6	7.043e-7
3-layer (50, 50)	W_1	1.223e-5	3.229e-7	3.929e-5	8.538e-7
	b_1	3.444e-6	3.376e-7	1.0	1.613e-1
	γ_1	n/a	n/a	2.128e-5	7.209e-7
	β_1	n/a	n/a	1.502e-5	8.573e-7
	W_2	6.517e-5	3.672e-7	6.497e-4	1.782e-6
	b_2	3.649e-6	3.345e-7	1.0	4.453e-2
	γ_2	n/a	n/a	7.919e-6	5.698e-7
	β_2	n/a	n/a	4.451e-6	3.596e-7
	W_3	4.868e-5	6.648e-7	3.531e-5	7.062e-7
	b_3	3.083e-6	1.024e-6	3.818e-6	6.213e-7
4-layer (50, 50, 50)	W_1	2.844e-5	4.387e-7	4.961e-5	1.296e-6
	b_1	3.122e-6	3.985e-7	1.0	9.600e-2
	γ_1	n/a	n/a	1.124e-6	1.939e-7
	β_1	n/a	n/a	2.531e-6	2.357e-7
	W_2	6.932e-4	6.551e-7	6.615e-5	9.452e-7
	b_2	2.286e-4	5.105e-6	1.000e-1	2.597e-2
	γ_2	n/a	n/a	1.417e-5	6.335e-7
	β_2	n/a	n/a	9.477e-7	1.331e-7
	W_3	4.499e-5	3.149e-7	5.224e-4	1.115e-6
	b_3	1.387e-6	2.214e-7	1.000e-1	1.979e-2
	γ_3	n/a	n/a	3.801e-6	4.780e-7
	β_3	n/a	n/a	9.277e-5	2.390e-6
	W_4	1.024e-4	9.208e-7	7.397e-5	7.729e-7
	b_4	4.300e-6	9.512e-7	4.487e-6	1.115e-6

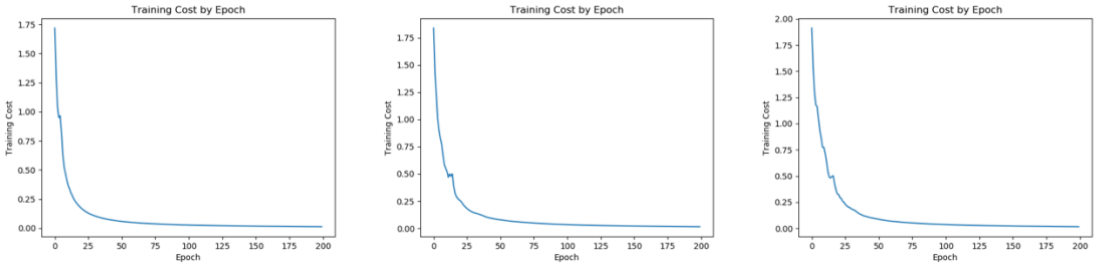

Figure 1. Cost value as a function of epoch for 2-layer (left), 3-layer (middle), and 4-layer (right) neural networks

Once the codes were proven to yield reliable results, batch normalization is implemented to 3-layer and 9-layer neural networks to see the evolution of the loss function. The datasets mainly used for this experiment are CIFAR-10 batch 1 to 5 except the last 5,000 samples for training, the last 5,000 samples in CIFAR-10 batch 5 for validation, and CIFAR-10 test batch for testing. The parameter settings for the experiment are batch size 100, η minimum 1e-5, η maximum 1e-1, total epoch 20, step size 2,250, lambda 0.005, and using He initialization.

2.2. Evolution of the loss function for 3-layer network

With the mentioned hyperparameter settings, 3-layer neural networks (hidden nodes = [50, 50]) are trained without batch normalization. I got 58.56% accuracy on training data, 52.70% accuracy on validation data, and 52.46% accuracy on testing data. When a batch normalization is implemented into the model, I got 62.66% accuracy on training data, 54.18% accuracy on validation data, and 53.78% accuracy on testing data. It turns out that batch normalization could slightly improve the accuracy of the testing data. Figure 2 and Figure 3 show the cross-entropy loss value, cost value, and the accuracy for each step of the 3-layer network without and with batch normalization, respectively.

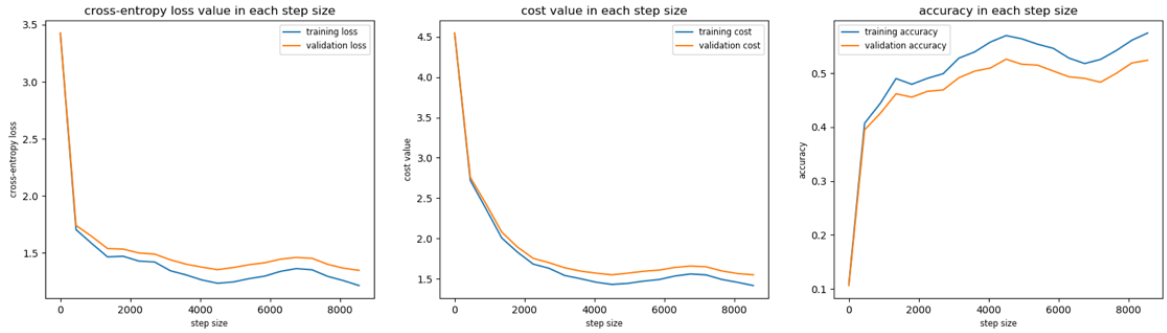

Figure 2. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network without batch normalization

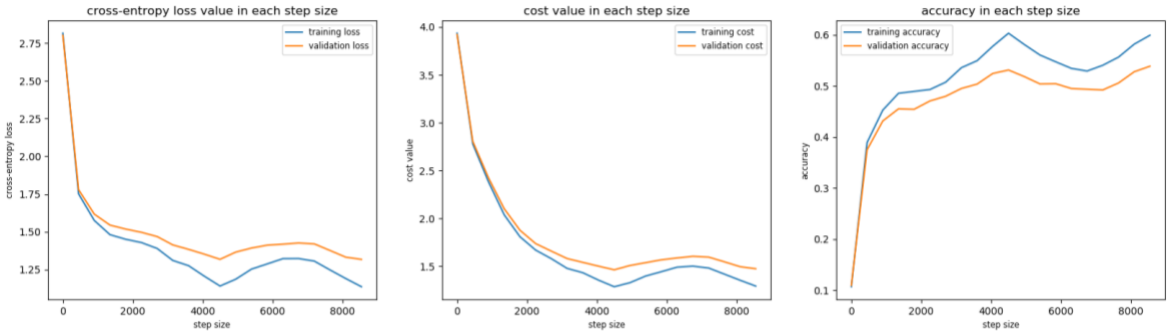

Figure 3. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network with batch normalization

2.3. Evolution of the loss function for 9-layer network

With the same hyperparameter settings, a 9-layer neural networks (hidden nodes = [50, 30, 20, 20, 10, 10, 10, 10]) are trained without batch normalization. I got 52.78% accuracy on

training data, 47.96% accuracy on validation data, and 47.08% accuracy on testing data. The accuracy tends to decline when the model complexity grows, or the more layers are added. After a batch normalization is implemented into the model, I got 59.88% accuracy on training data, 52.72% accuracy on validation data, and 51.95% accuracy on testing data. It turns out that batch normalization could significantly improve the accuracy of the testing data for the case of deeper neural networks. Figure 4 and Figure 5 show the cross-entropy loss value, cost value, and the accuracy for each step of the 9-layer network without and with batch normalization, respectively.

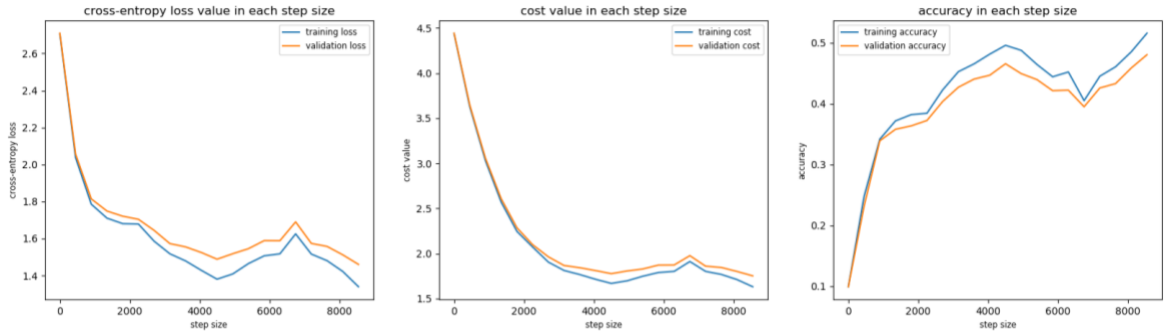

Figure 4. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 9-layer network without batch normalization

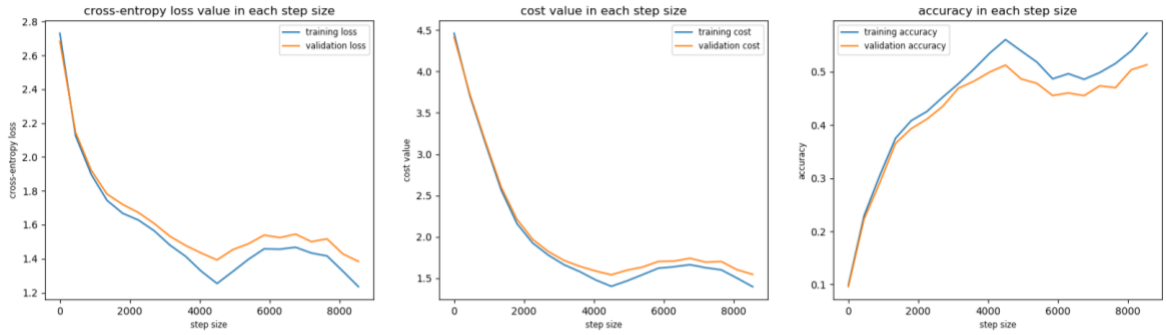

Figure 5. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 9-layer network with batch normalization

2.4. Coarse-to-fine random search

This experiment aims to find the optimal regularization parameter for the networks. This technique is beneficial to find the best hyperparameters through repetitively randomly selecting a value, which falls between the given optimal range. In the coarse search, I define a list of lambda value to be tested:

$$[10^{-5}, 10^{-4.5}, 10^{-4}, 10^{-3.5}, 10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}],$$

along with batch size 100, step size 2,250 and for 2 cycles learning. Below are the top three lambdas during the coarse search for the 3-layer neural networks with batch normalization:

Table 2. Accuracy for the top three lambdas during coarse search

Lambda	Validation Accuracy
10^{-2}	54.00%
10^{-3}	53.72%
$10^{-2.5}$	53.02%

$$l = l_{min} + (l_{max} - l_{min}) * rand(1,1); \quad (2)$$
$$lambda = 10^{l}$$

In the fine search, I also use 2 cycles learning and iteratively generate 15 random lambda values within the range of $[10^{-3}, 10^{-2}]$ using equation (2), selecting the top three lambda values based on its validation accuracy, update the range based on the best lambda until it converges, as the process could be seen in Table 3 and Figure 6 below. After three iterations, the best lambda gained from this fine search is 0.00440015 or $10^{-2.357}$, which achieves accuracy almost 55.5% on validation datasets.

Table 3. The best lambda reached from coarse-to-fine random search

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

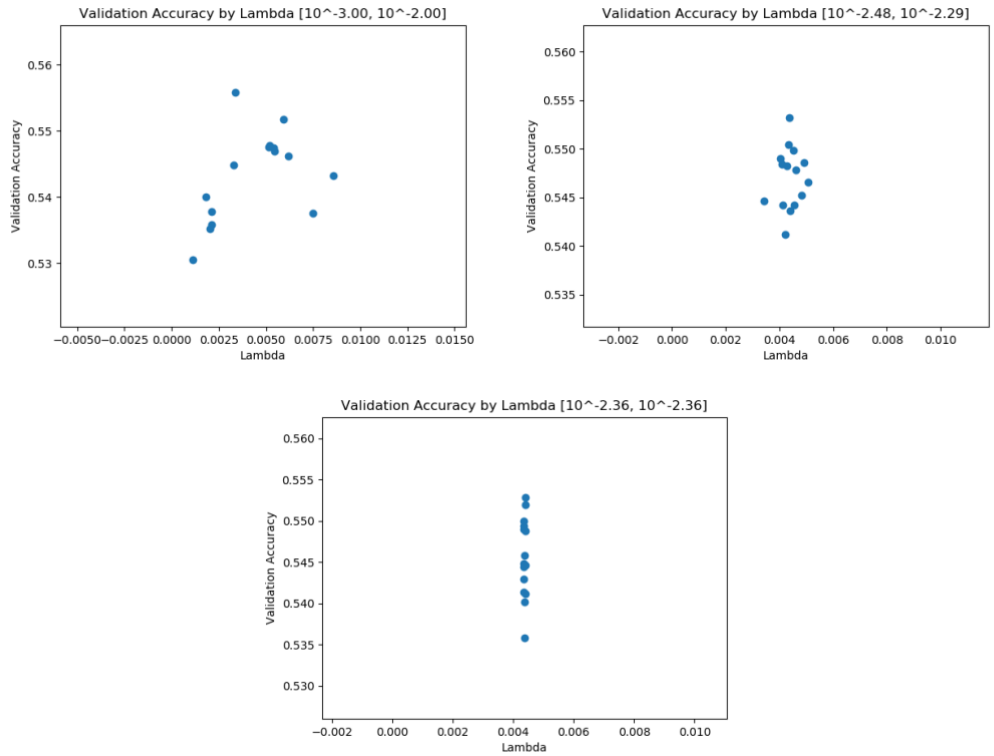

Figure 6. A fine search of the best lambda in the first iteration (top left), the second iteration (top right) and the third iteration (bottom)

Using the best lambda value mentioned above, I got 62.82% accuracy on training data, 54.46% accuracy on validation data, and 53.72% accuracy on testing data.

2.5 Sensitivity to initialization

In the last experiment, one of the advantages of implementing batch normalization to the model, which is better sensitivity to initialization, will be examined. Three different sigma values (1e-1, 1e-3, 1e-4) are applied during the weight initialization, which follows the normal distribution with zero mean and will be studied the effect of the batch normalization to the random normal initialization. Table 4 summarizes all results of the 3-layer networks with different sigma values and batch normalization treatment.

Table 4. Accuracies of batch normalization to the different values of variance in the weight initialization

Sigma	Batch Normalization	Accuracy		
		Training	Validation	Testing
1e-1	Yes	55.72%	51.34%	51.56%
	No	54.49%	50.84%	50.81%
1e-3	Yes	56.16%	51.76%	51.45%
	No	18.56%	19.20%	18.66%
1e-4	Yes	55.43%	51.42%	51.54%
	No	10.06%	9.50%	10.0%

From the table above, it can be concluded that batch normalization is able to remove the dependency on the weight initialization. Specifically, when the variance becomes smaller during the weight initialization, the model without batch normalization cannot learn properly. Figure 7 to Figure 12 depict the cross-entropy loss value, cost value, and the accuracy for each sigma and batch normalization implementation status.

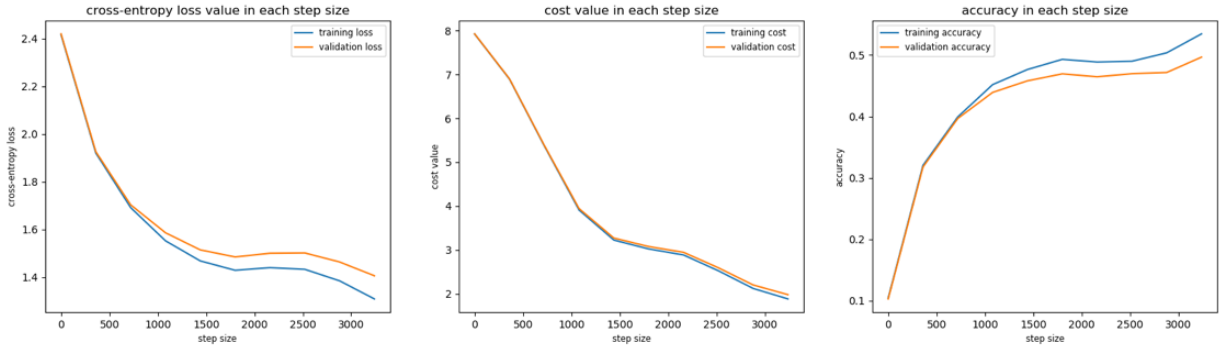

Figure 7. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network with batch normalization and sigma 1e-1

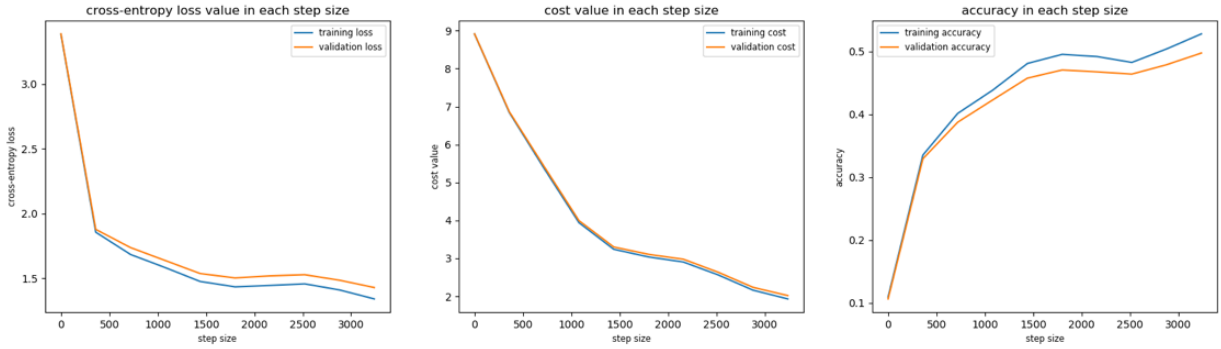

Figure 8. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network without batch normalization and sigma 1e-1

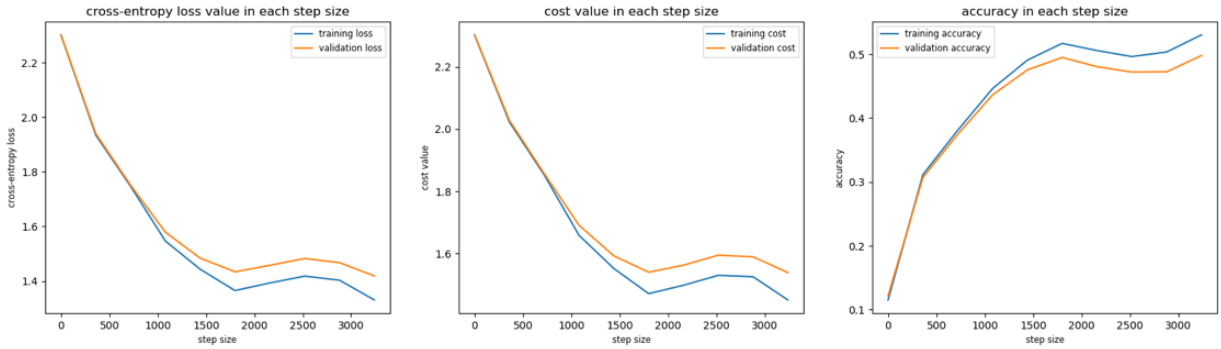

Figure 9. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network with batch normalization and sigma 1e-3

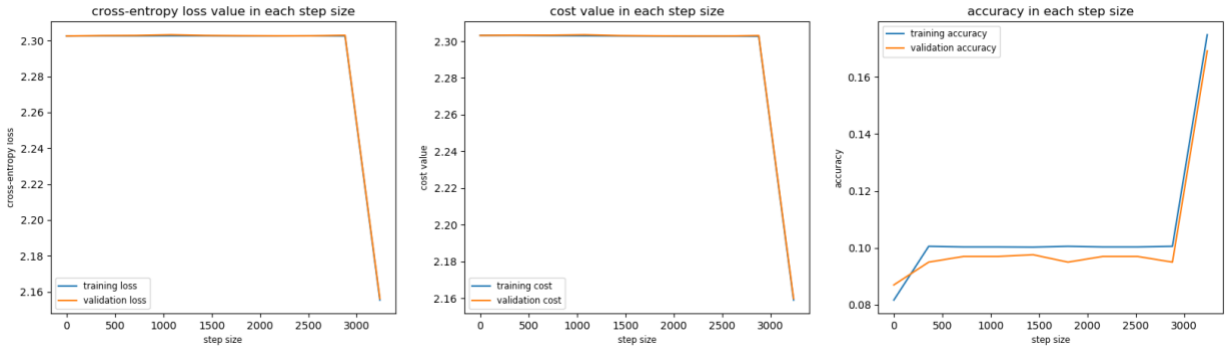

Figure 10. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network without batch normalization and sigma 1e-3

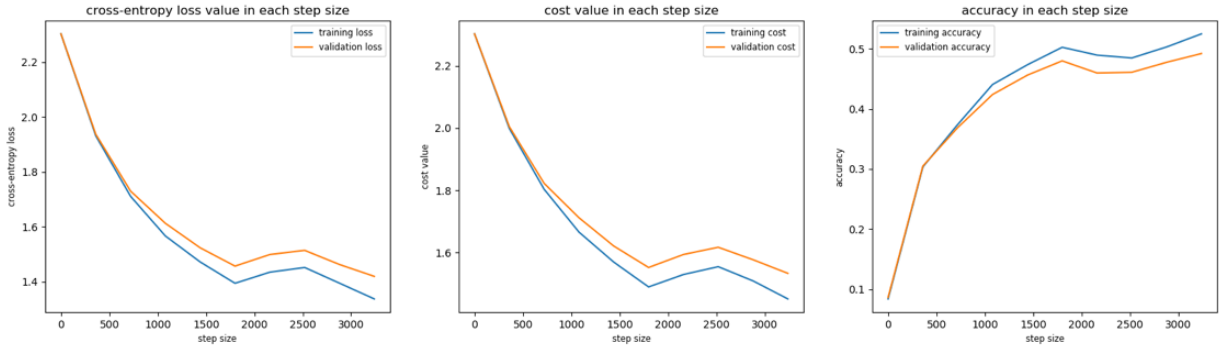

Figure 11. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network with batch normalization and sigma 1e-4

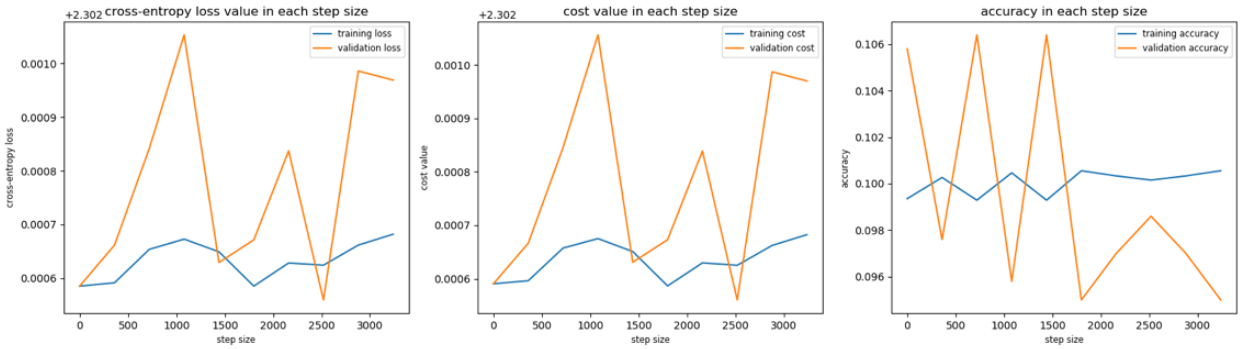

Figure 12. Cross-entropy loss (left), cost value (middle), and accuracy (right) as a function of the step size from the 3-layer network without batch normalization and sigma 1e-4

3. Final remarks

From this assignment, it could be learned that applying a batch normalization could have a significant impact when learning a deeper neural network model in terms of the testing accuracy. One also could find out the best lambda value through a coarse-to-fine random search to get the best regularization capability of the model. After getting the best lambda value, the accuracy of the testing data is 53.72% for the 3-layer networks. One of the most notable advantages of batch normalization is less sensitivity to the weight initialization.