DD2424 Deep Learning in Data Science

Name: Sri Janani Rengarajan *Programme: Embedded Systems*

Assignment 2: Training and testing of a two layer network with multiple outputs

The aim of the assignment is to classify images into ten classes using a two layer network with a hidden layer. Input to the classifier is the vector **x,** which contains the information about the image. Output of the classifier is a probability vector **p**, for each possible class. The parameters of the classifier are **W1, W2, b1** and **b2.** These parameters are learned by minimizing the cross-entropy loss of the classifier.

1) Gradient computation check:
   Analytical gradient computation was compared against numerical gradient computation for different batch sizes and ε values. The error results are presented in table 1 and 2. The error values are considerably small. Hence, we can conclude that the analytical method works well.

```
function [grad_W, grad_b] = ComputeGradients(X, Y,
P, H, W, lambda)
W1 = W{1};
W2 = W{2};
N = size(X,2);
I = ones(N,1);
G = -(Y-P);
grad_W2 = ((G*H')/N) + (2*lambda*W2);
grad_b2 = (G*I)/N;
G = W2' * G;
H(H>0) = 1;
G = G.*H;
grad_W1 = ((G*X')/N) + (2*lambda*W1);
grad_b1 = (G*I)/N;
grad_W = {grad_W1, grad_W2};
grad_b = {grad_b1, grad_b2};
end
```

For batch size 100:

ε	$\frac{\partial J}{\partial W_1}$	$\frac{\partial J}{\partial W_2}$	$\frac{\partial J}{\partial b_1}$	$\frac{\partial J}{\partial b_2}$
0.00001	4.7799e-06	1.5090e-08	1.5190e-09	1.0877e-10
0.0001	4.7799e-07	1.5090e-08	1.5190e-09	1.0877e-10
0.001	4.7799e-08	1.5090e-08	1.5190e-09	1.0877e-10
0.01	4.7799e-09	2.19836e-09	8.5349e-10	1.0877e-10

Table 1. Gradient computation error between analytical and numerical methods for batch size 100

For batch size 50:

ε	$\frac{\partial J}{\partial W_1}$	$\frac{\partial J}{\partial W_2}$	$\frac{\partial J}{\partial b_1}$	$\frac{\partial J}{\partial b_2}$
0.00001	6.2497e-06	7.3274e-09	1.8540e-08	3.8983e-10
0.0001	6.2497e-07	7.3274e-09	1.8540e-08	3.8983e-10
0.001	6.2497e-08	7.3274e-09	1.6690e-08	3.8983e-10
0.01	6.2497e-09	3.1953e-09	1.8249e-09	3.8983e-10

Table 2. Gradient computation error between analytical and numerical methods for batch size 50

2) Cyclic Learning rate
Instead of using a vanilla implementation of the mini-batch gradient descent algorithm, in order to improve training time and to avoid unnecessary search for a good learning rate value, cyclic learning rate method is used. This method adapts the learning rate at each iteration. The result of this implementation is presented below.

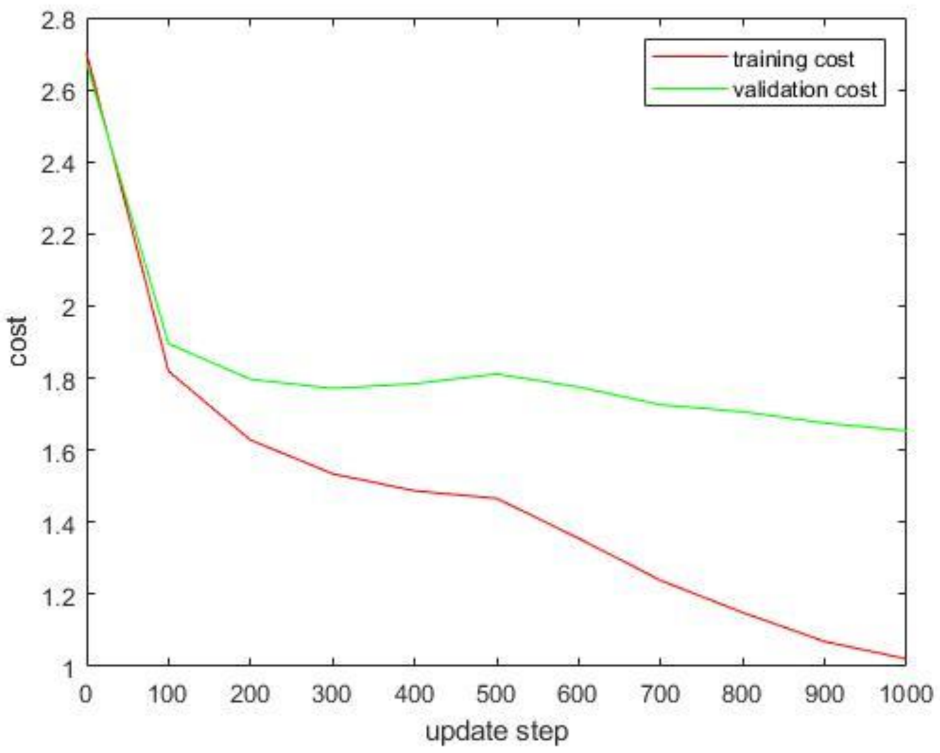

Figure 1. Training and validation cost (lambda =0, batch = 100, ns = 500, cycle = 1)

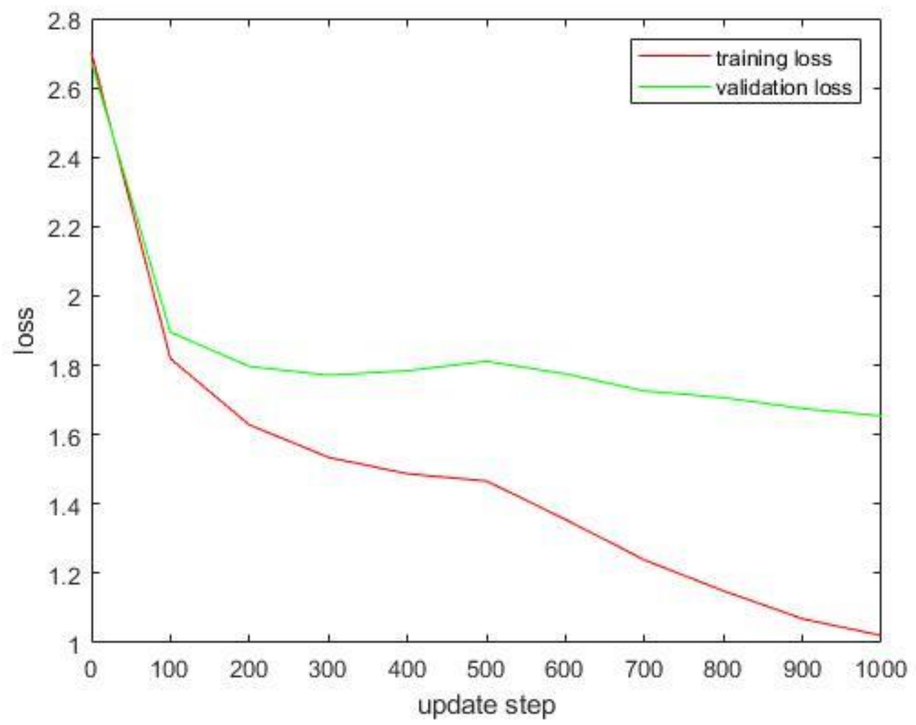

Figure 2. Training and validation loss (lambda =0, batch = 100, ns =500, cycle = 1)

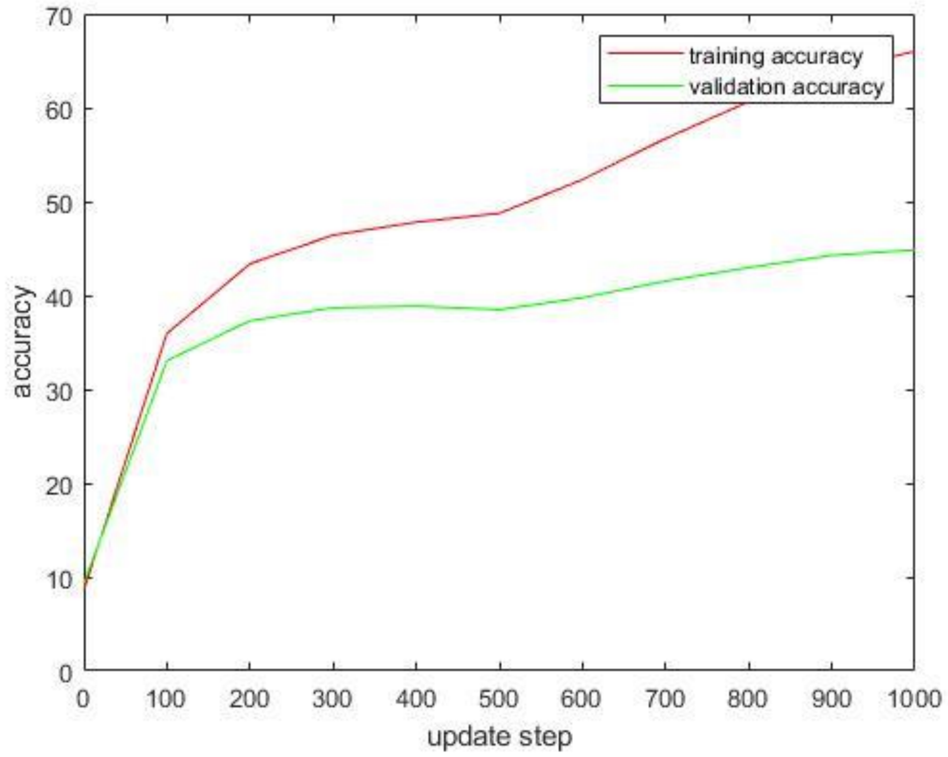

Figure 3. Training and validation accuracy (lambda =0, batch = 100, ns =500, cycle =1)

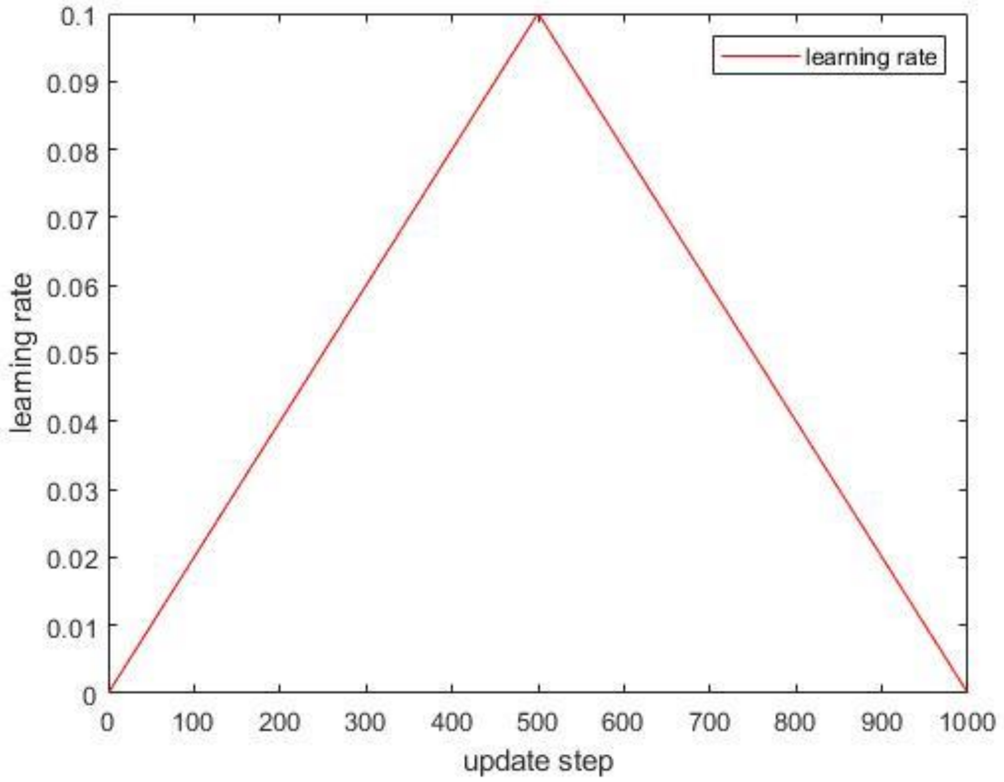

Figure 4. Learning rate (lambda =0, batch = 100, ns =500, cycle =1)

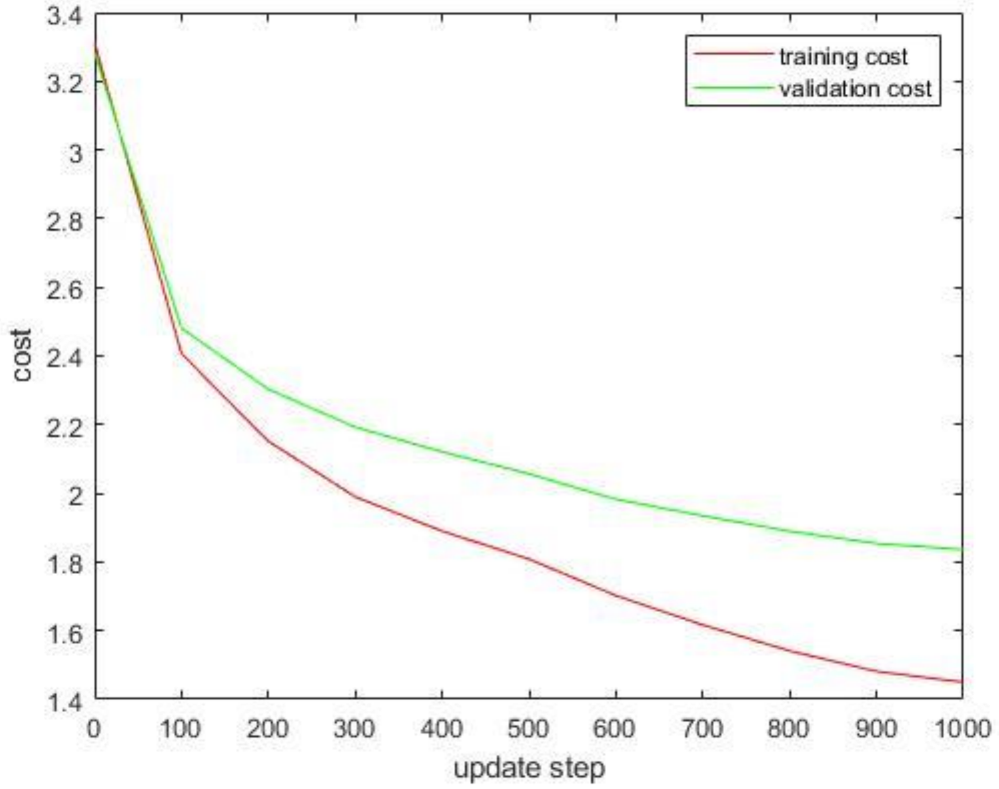

Figure 5. Training and validation cost (lambda =0.01, batch = 100, ns = 500, cycle = 1)

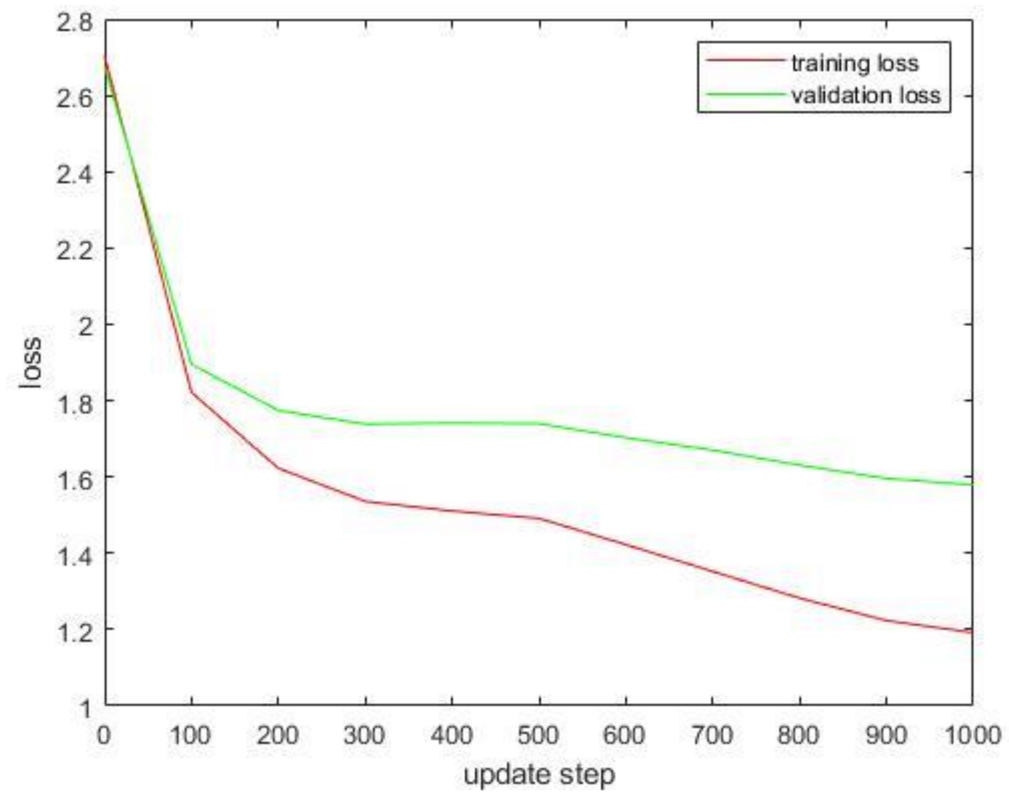

Figure 6. Training and validation loss (lambda =0.01, batch = 100, ns =500, cycle = 1)

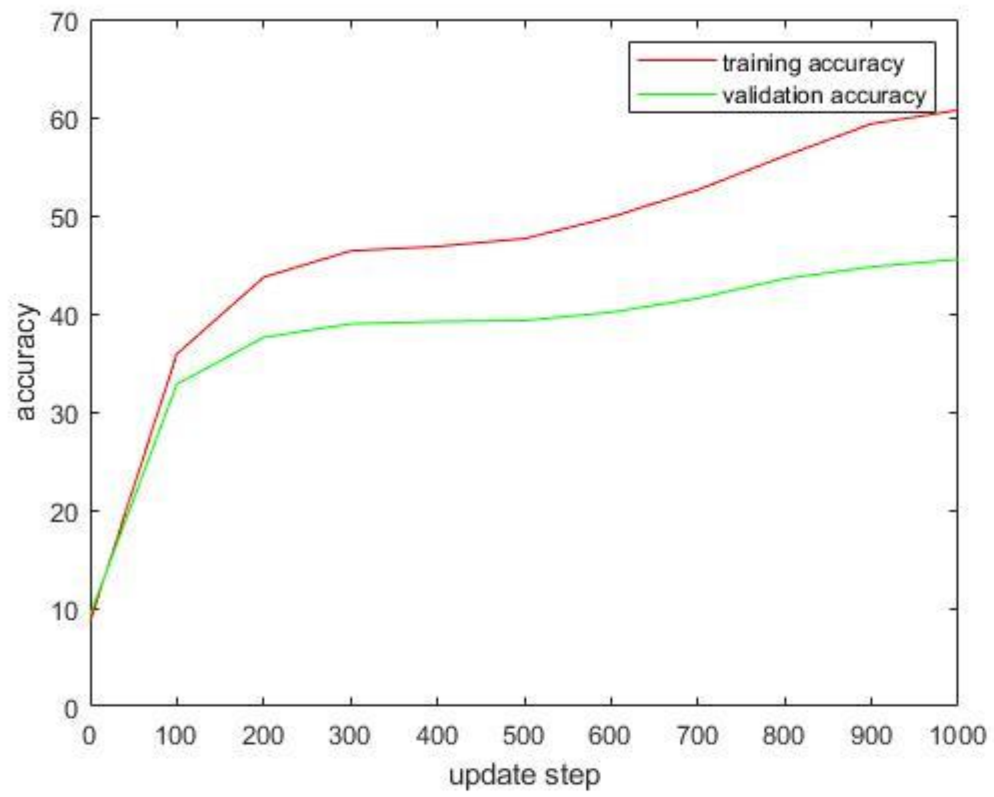

Figure 7. Training and validation accuracy (lambda =0.01, batch = 100, ns =500, cycle =1)

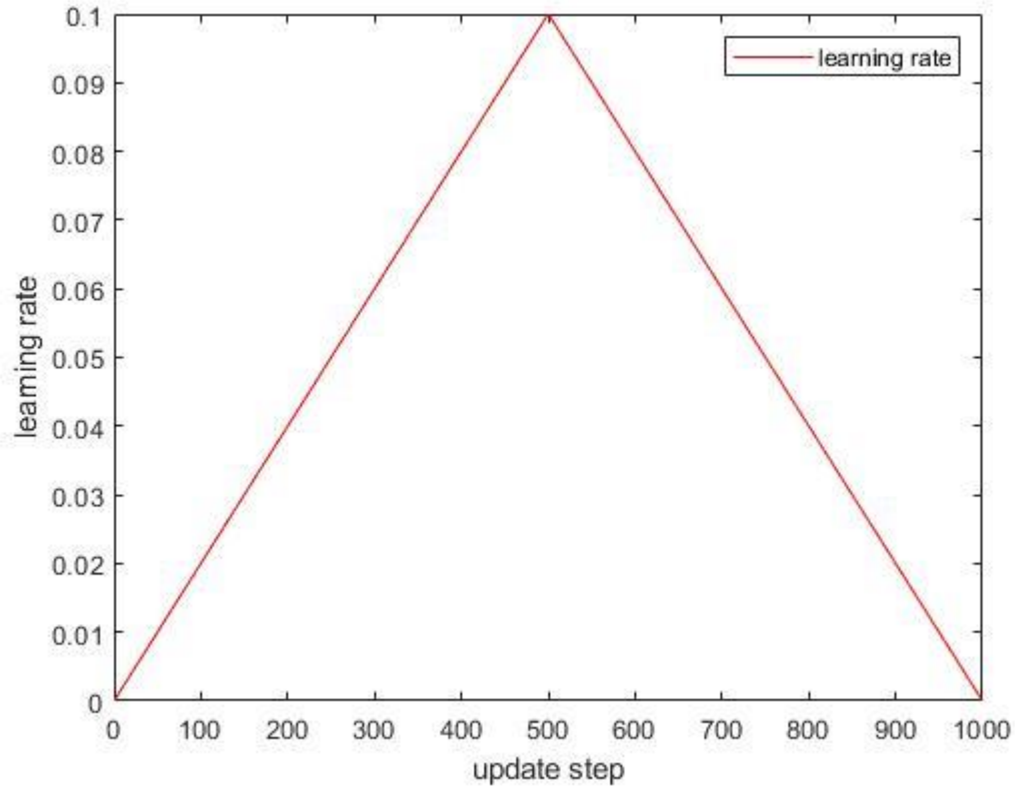

Figure 8. Learning rate (lambda =0.01, batch = 100, ns =500, cycle =1)

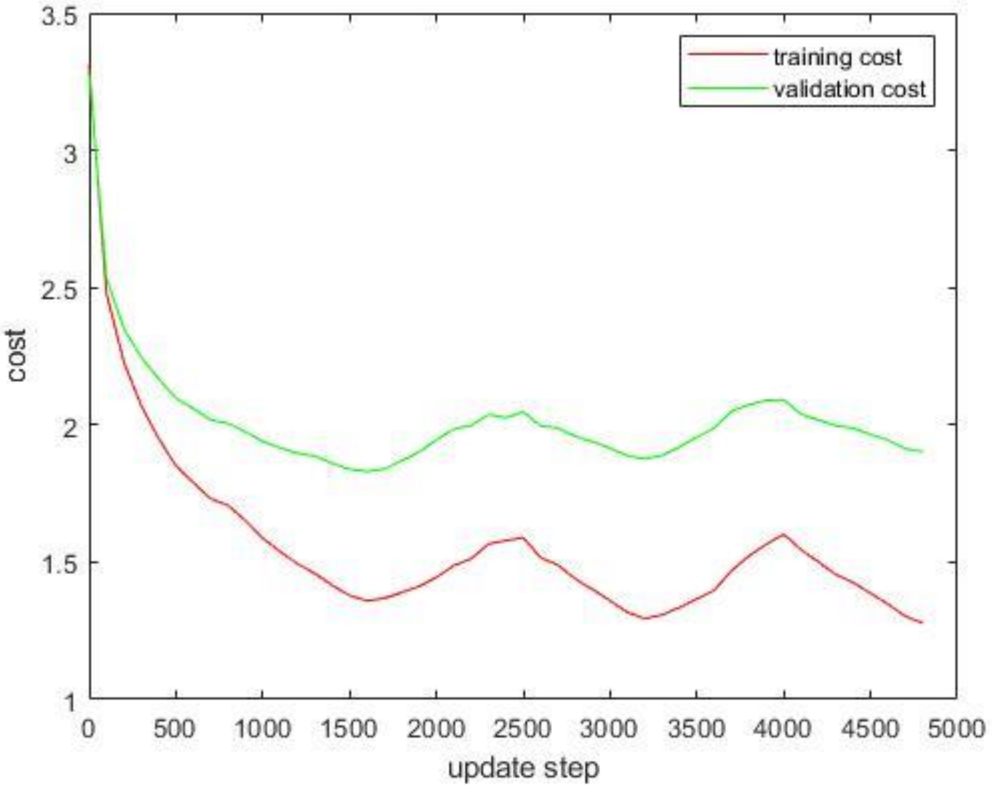

Figure 9. Training and validation cost (lambda =0.01, batch = 100, ns =800, cycle = 3)

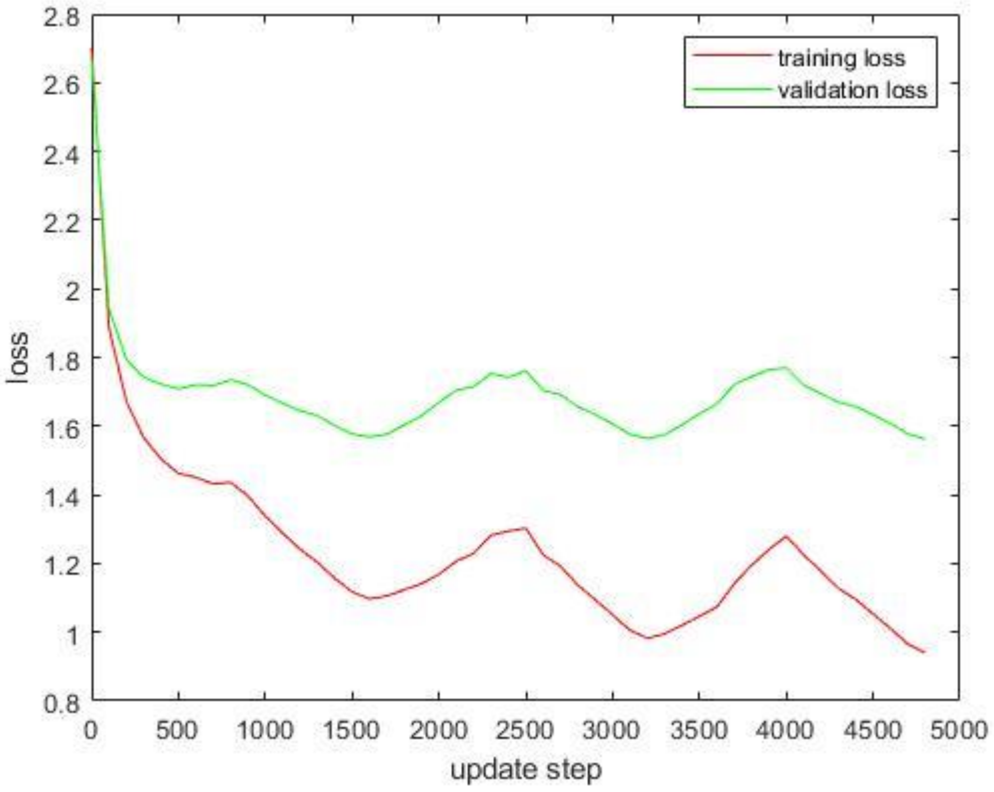

Figure 10. Training and validation loss (lambda =0.01, batch = 100, ns =800, cycle = 3)

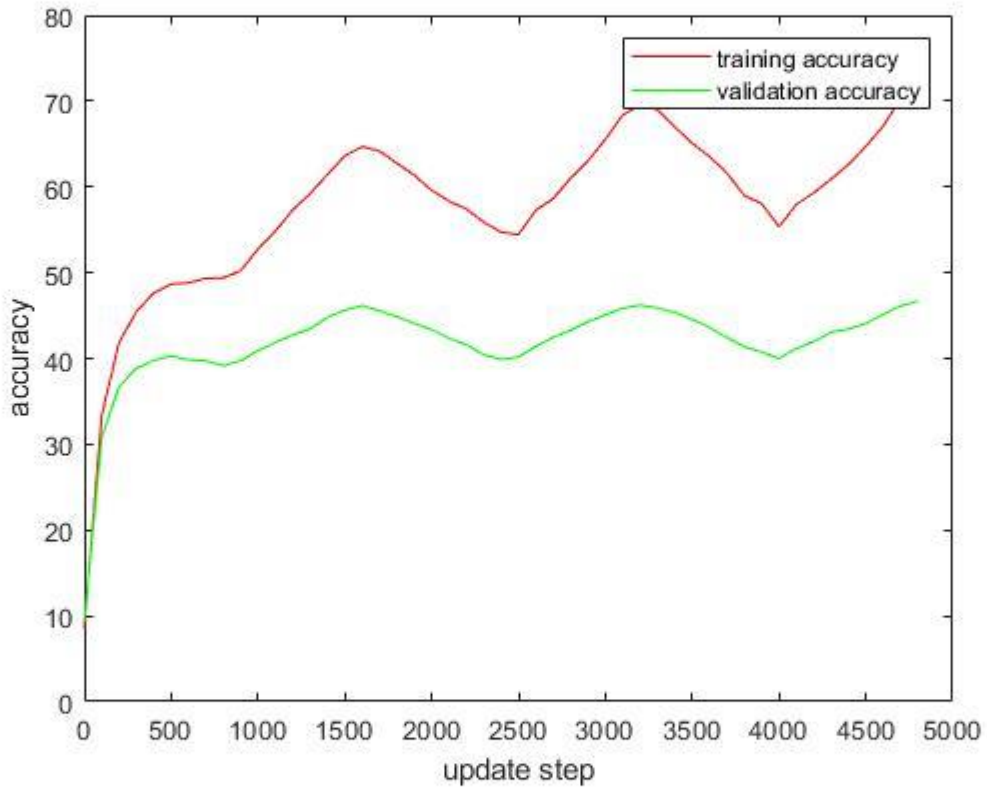

Figure 11. Training and validation accuracy (lambda =0.01, batch = 100, ns =800, cycle =3)

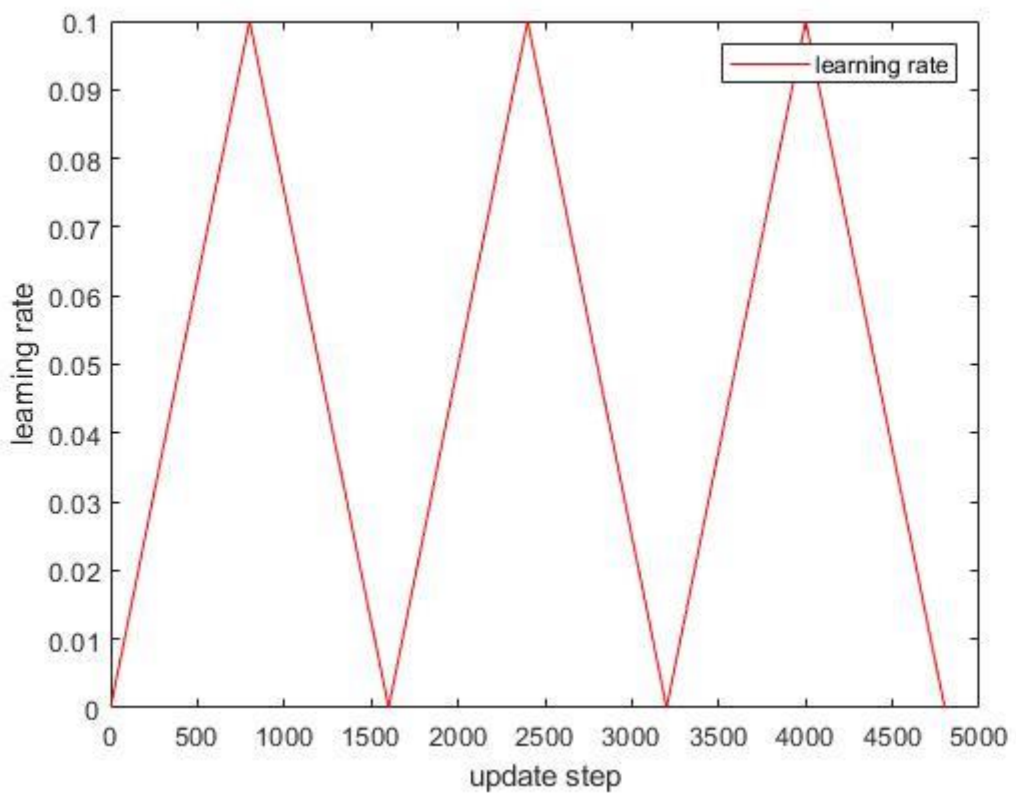

Figure 12. Learning rate (lambda =0.01, batch = 100, ns =800, cycle =3)

Regularization (λ)	Step size (n_s)	Cycles	Training Accuracy (%)		Validation Accuracy (%)		Test Accuracy (%)
			Mean	At the end of the cycles	Mean	At the end of the cycles	
0	500	1	48.2364	66.02	37.1891	44.85	44.96
0.01	500	1	46.1500	60.72	37.5155	45.53	45.88
0.01	800	3	57.6012	71.68	41.7631	46.71	46.77

Table 3. Training, validation, test accuracies for different regularization, step sizes and cycles

Observation:

From table 3 and the figures presented above, it can be understood that as the regularization decreases and the number cycles increase, the average validation accuracy and testing accuracy increases.

3. Search for lambda:
   Coarse and fine searches for lambda in *log* scale was performed using the following code and validation accuracy was used as performance parameter.

```
Grid = unigrid(-1.88,0.005,-1.8);
for i = 1:16
    for j = i+1:17
        lmin = Grid(i);
        lmax = Grid(j);
        l = lmin + (lmax-lmin)*rand(1,1);
        lambda(i,j) = 10^l;
        GDparams.lambda = lambda(i,j);
        [W, b, GDparams] = MiniBatchGD(Xt, Yt, GDparams, W, b);
        acc_valid(i,j) = ComputeAccuracy(Xv, yv, W, b);
    end
end
```

- Unigrid (-5, 0.5, -1)

λ_{min}	λ_{max}	λ	Validation accuracy (%)
$10^{-2.5}$	10^{-1}	0.0161441583339176	47.44
10^{-2}	$10^{-1.5}$	0.0114449135283787	47.42
$10^{-2.5}$	10^{-2}	0.00737885031523282	46.52

Table 4. Three best validation accuracies obtained for coarse search of lambda in the range of $[10^{-5}, 10^{-1}]$

- Unigrid (-2, 0.1, -1)

λ_{min}	λ_{max}	λ	Validation accuracy (%)
10^{-2}	$10^{-1.4}$	0.0117950979258755	48.36
$10^{-1.9}$	$10^{-1.5}$	0.0215692757202300	48.34
$10^{-1.9}$	$10^{-1.6}$	0.0143470048303172	48.10

Table 5. Three best validation accuracies obtained for fine search of lambda in the range of $[10^{-2}, 10^{-1}]$

- Unigrid (-1.9, 0.02, -1.4):

λ_{min}	λ_{max}	λ	Validation accuracy (%)
$10^{-1.88}$	$10^{-1.74}$	0.0135638396359717	48.74
$10^{-1.88}$	$10^{-1.72}$	0.0156863407035269	48.6
$10^{-1.88}$	$10^{-1.84}$	0.0142590946842586	48.6

Table 6. Three best validation accuracies obtained for fine search of lambda in the range of $[10^{-1.9}, 10^{-1.4}]$

From table 6, λ = 0.01356 is chosen as the best performing value, using this lambda value network's training and validation accuracy, cost, loss are presented in the following figures.

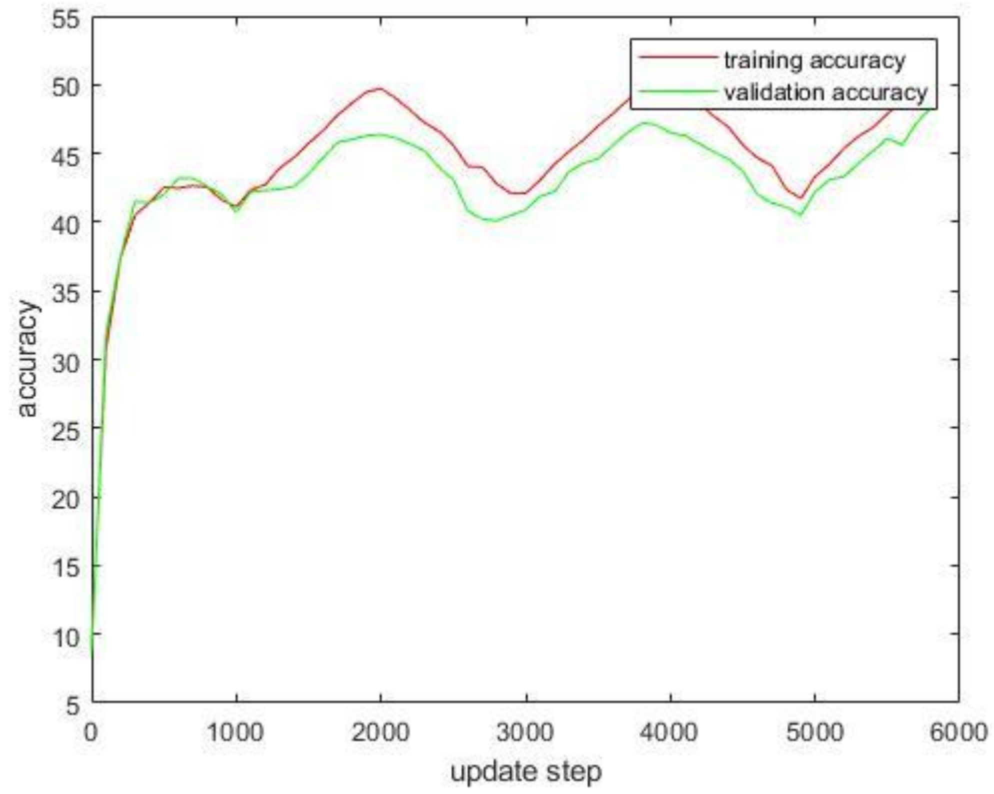

Figure 13. Training and validation accuracy (lambda = 0.01356, batch = 100, cycle = 3)

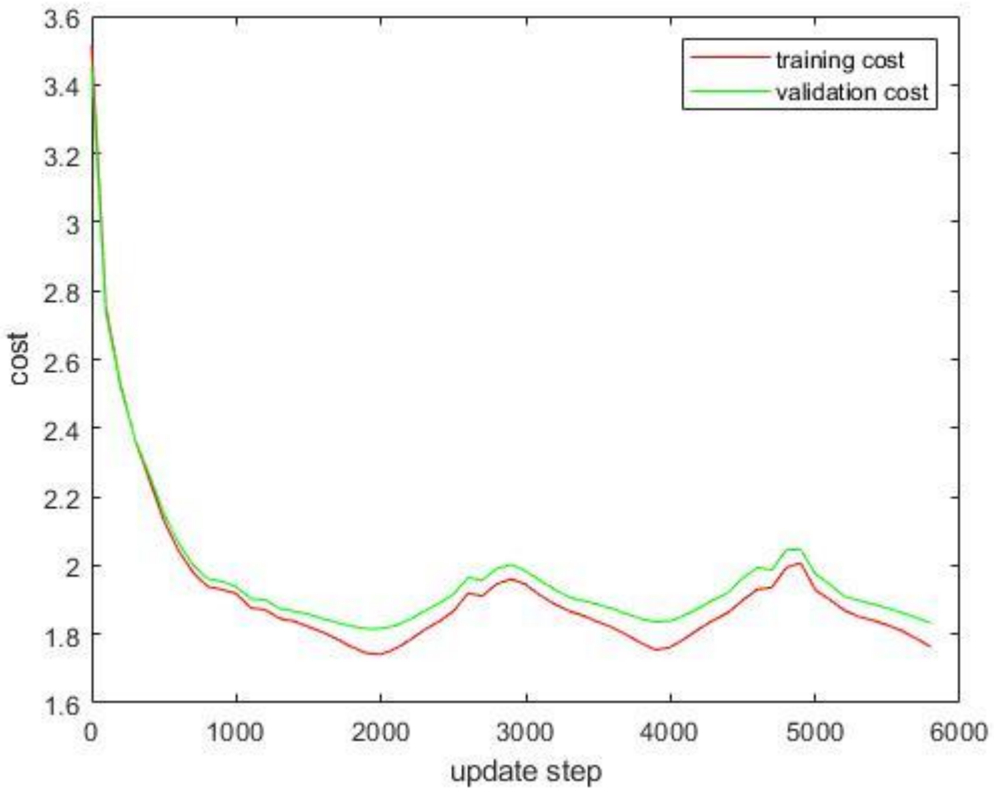

Figure 14. Training and validation cost (lambda = 0.01356, batch = 100, cycle = 3)

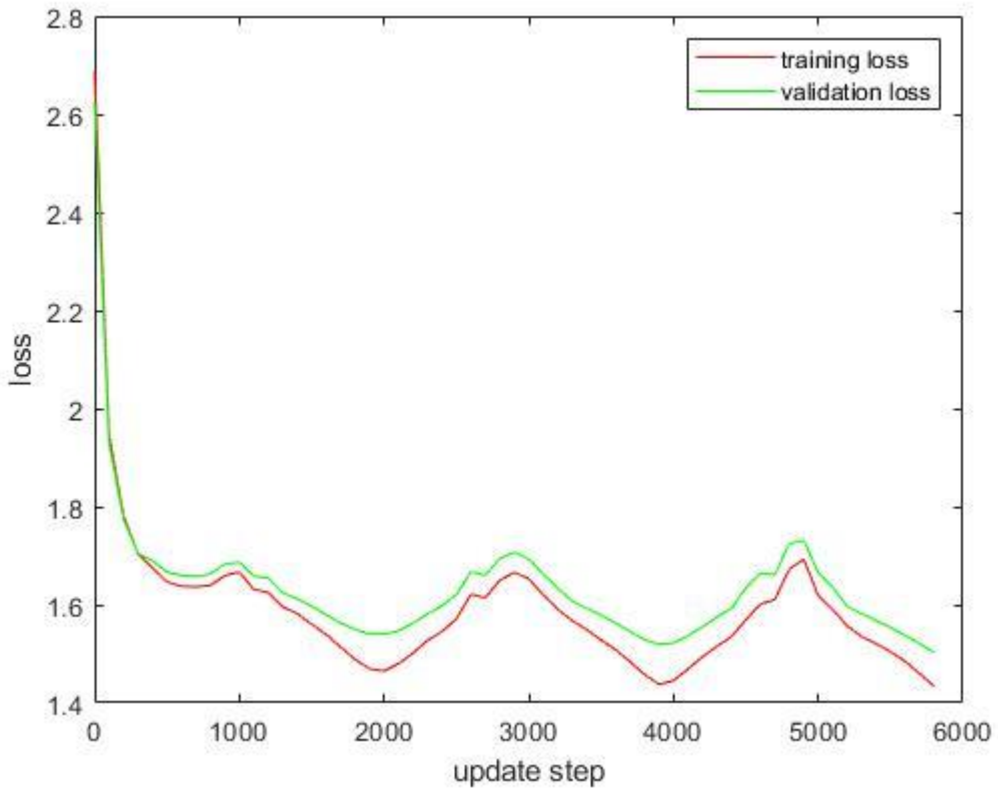

Figure 15. Training and validation loss (lambda = 0.01356, batch = 100, cycle = 3)

Final result:

Regularization (λ)	Step size (n_s)	Cycles	Batch size	Test accuracy (%)
0.01356	980	3	100	48.01