Assignment 1

1. Gradient Computation

In this assignment, an analytical gradient computation for a mini-batch is implemented to train the neural network based on generic backward pass. The Matlab code is listed below.

```
function [grad_W, grad_b] = ComputeGradients(X, Y, P, W, lambda)
% DENOTE d as the dimensionality of each image, N as the number of images
%        K as the number of label kinds
% INPUT     - X:                    d*N
%           - Y:                    K*N
%           - P:                    K*N
%           - W:                    K*d
%           - lambda:               1*1
% OUTPUT    - grad_W:               K*d
%           - grad_b:               K*1

grad_W = zeros(size(W));
grad_b = zeros(size(W, 1), 1);

for i = 1 : size(X, 2)
    Pi = P(:, i);
    Yi = Y(:, i);
    Xi = X(:, i);
    g = -Yi'*(diag(Pi) - Pi*Pi')/(Yi'*Pi);
    grad_b = grad_b + g';
    grad_W = grad_W + g'*Xi';
end

grad_b = grad_b/size(X, 2);
grad_W = grad_W/size(X, 2);

% add regularization term
grad_W = grad_W + 2*lambda*W;

end
```

Here, to test whether this analytic gradient code is correct, the function `ComputeGradsNumSlow` is used for comparison. The numerically and analytically computed gradient matrices are compared by computing the relative error between these two matrices and check if this error is small. We denote the relative error ε, the numerically computed gradient matrix $\boldsymbol{g_n}$, and the analytically computed gradient matrix $\boldsymbol{g_a}$. Then, the relative error can be expressed as:

$$\varepsilon = \max\left(|\boldsymbol{g_a} - \boldsymbol{g_n}|./\max\left(0, |\boldsymbol{g_a}| + |\boldsymbol{g_n}|\right)\right)$$

According to the equation above, gradient checks on mini-batch are performed with different batch size and regularization (lambda). The results are listed in Tab. 1.

Tab. 1: Relative errors with respect to batch size and regularization (lambda).

Batch size	1	50	200	1	50	200
lambda	0	0	0	0.1	0.1	0.1
ε_b	9.2168e-10	7.1639e-08	5.2492e-08	8.0413e-10	1.7182e-07	3.7798e-07
ε_W	6.4088e-08	1.1671e-04	9.2398e-04	4.0654e-06	3.0648e-04	6.9472e-04

From the results in the above table, we can see that all the relative errors are small enough. Thus, we can draw the conclusion that the analytical gradient computation is correct.

2. Results and Analysis

In this section, results of plots and figures for different parameters settings are shown below:

- **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.1**

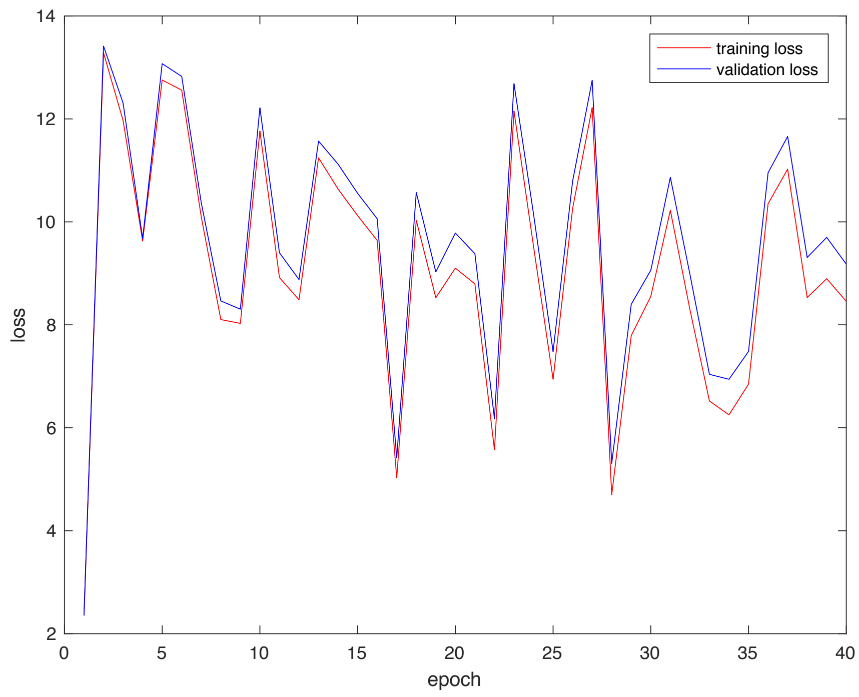

Fig. 1: Total loss function on the training data and validation data after each epoch of the mini-batch gradient descent algorithm (lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.1)

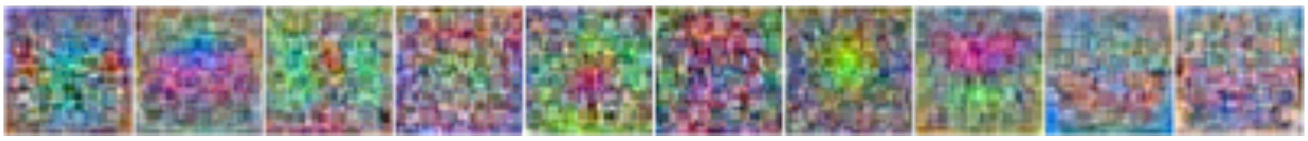

Fig. 2: The learnt W matrix visualized as class template images (lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.1)

- **lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.01**

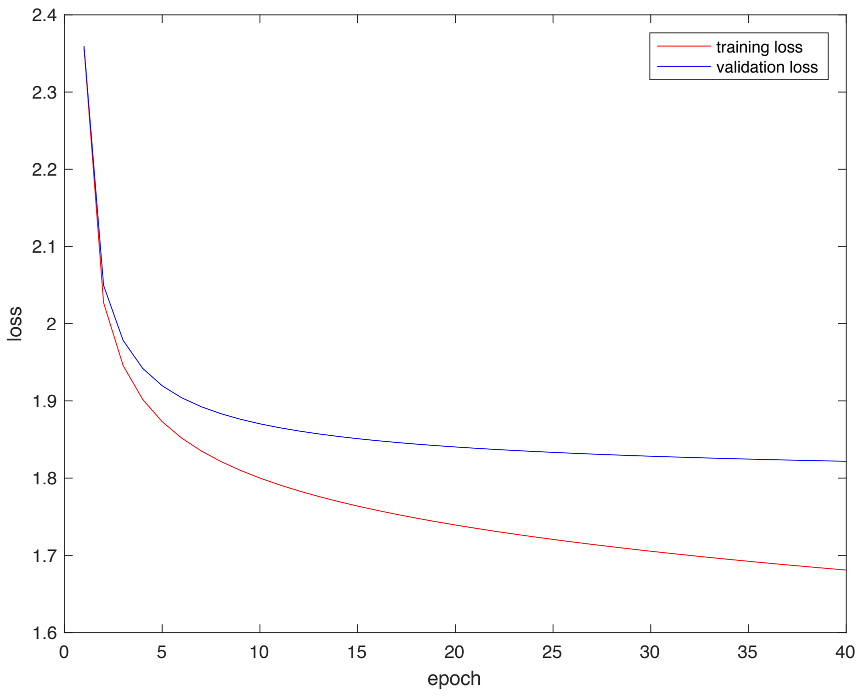

Fig. 3: Total loss function on the training data and validation data after each epoch of the mini-batch gradient descent algorithm (lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.01)

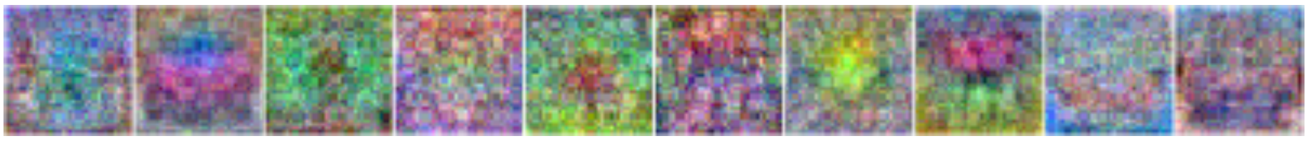

Fig. 4: The learnt W matrix visualized as class template images (lambda = 0, n_epochs = 40, n_batch = 100, eta = 0.01)

- **lambda = 0.1, n_epochs = 40, n_batch = 100, eta = 0.01**

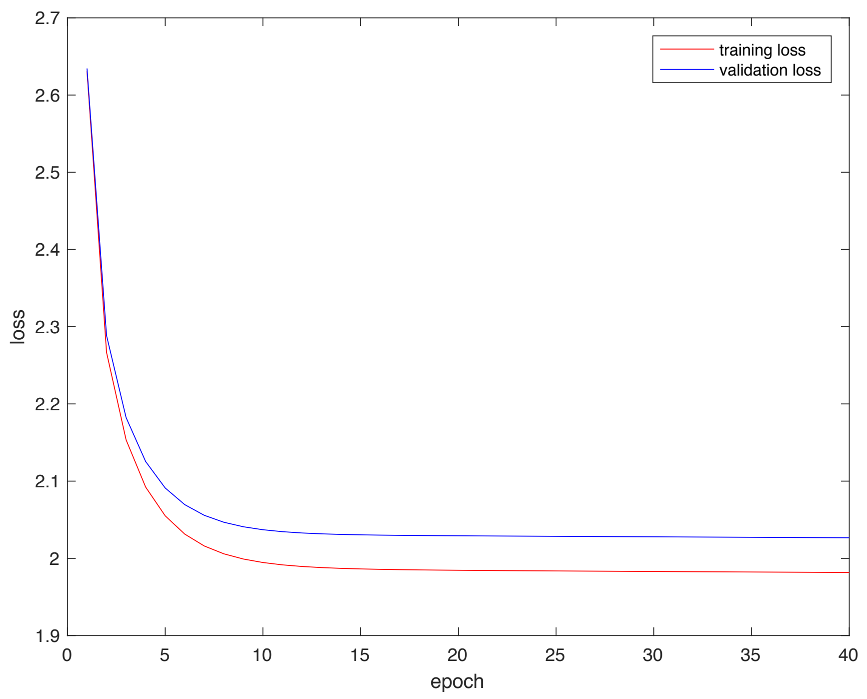

Fig. 5: Total loss function on the training data and validation data after each epoch of the mini-batch gradient descent algorithm (lambda = 0.1, n_epochs = 40, n_batch = 100, eta = 0.01)

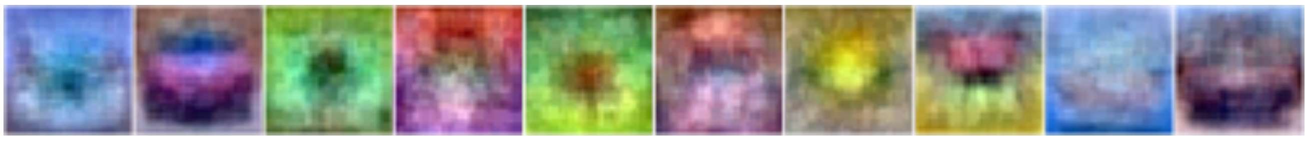

Fig. 6: The learnt W matrix visualized as class template images (lambda = 0.1, n_epochs = 40, n_batch = 100, eta = 0.01)

- **lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.01**

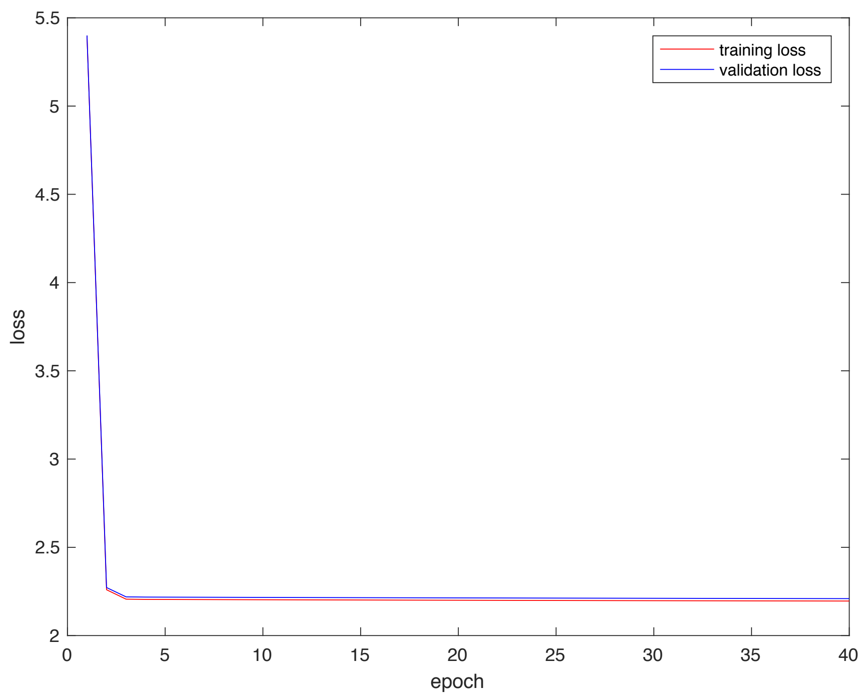

Fig. 7: Total loss function on the training data and validation data after each epoch of the mini-batch gradient descent algorithm (lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.01)

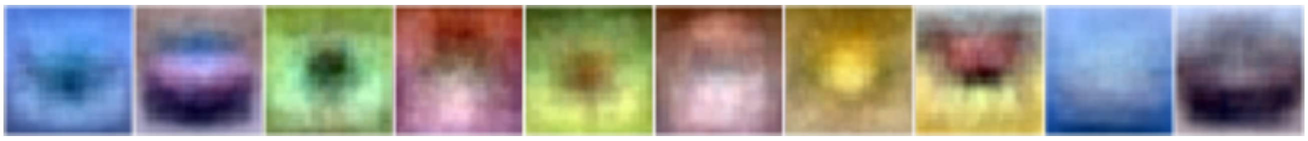

Fig. 8: The learnt W matrix visualized as class template images (lambda = 1, n_epochs = 40, n_batch = 100, eta = 0.01)

Tab. 2: Training and test accuracy for different parameter settings

lambda	n_epochs	n_batch	eta	training accuracy (%)	test accuracy (%)
0	40	100	0.1	31.20	26.84
0	40	100	0.01	41.65	36.91
0.1	40	100	0.01	34.15	33.38
1	40	100	0.01	22.31	21.93

Besides, training accuracy and test accuracy of the network after each of the training runs above is listed in Tab. 2. From the results, we can get some conclusions:

1. When increasing the learning rate, the loss function will accelerate its convergence, but may at the same time cause significant oscillations after some epochs. Thus, the test accuracy will decrease; when decreasing the learning rate, the convergence of the loss function will slow down, but the plot will be much smoother and the test accuracy will increase.

2. When increasing the amount of regularization, the network complexity will be restricted to some degree. In theory, the generalization of the network will be enhanced and the test accuracy will increase. But if we increase regularization too much, the network may not be able to represent the data, and therefore cause a drop of its test accuracy.