Assignement 1: Report basic

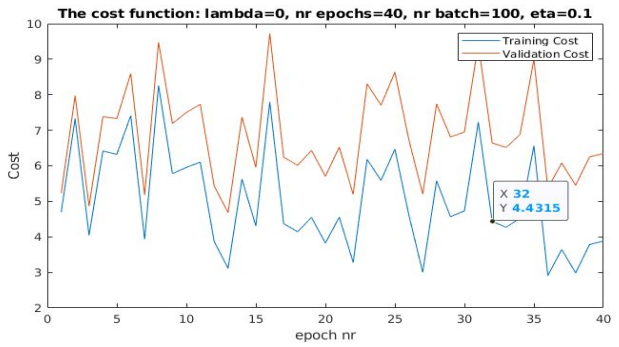
```
function [grad_W, grad_b] = ComputeGradients(X, Y, P, W, lambda)
%% Computes gradients for a given minibatch based on Cross-entropy loss function
% Input:
%  - X:         [DxN] D is the dimensionality of each image sample
%               and N is the number of samples on the minibatch
%
%  - Y:         [KxN] one hot representation of the label of each sample.
%               K is nr of possible classes.
%
%  - W:         [KxD] weights
%
%  - b:         [Kx1] bias
%
%  - lambda:    [double] regularization parameter
%
% Output:
%  - grad_W:    [KxD] gradients for the weights
%
%  - grad_b:    [KxD] gradient for the bias
%%
% initialize gradients
N = size(X,2);
[K,D] = size(W);
grad_W = zeros(K,D);
grad_b = zeros(K,1);

% gradient p for all samples in batch(gradients are columns of Grad_p)
Grad_P = -Y.* (1./sum(Y.*P,1));

% Add gradients for each sample
for i=1:N
    Pi= P(:,i);
    % Calc grad for the output of the neurons before softmax for sample i
    J = diag(Pi)-Pi*Pi';
    grad_s = J * Grad_P(:,i);

    % Calc gradients for bias and weights
    grad_b = grad_b + grad_s;
    grad_W = grad_W + grad_s * X(:,i)';
end
grad_b = grad_b./size(X,2);
grad_W = grad_W./size(X,2) + 2*lambda*W;
end
```

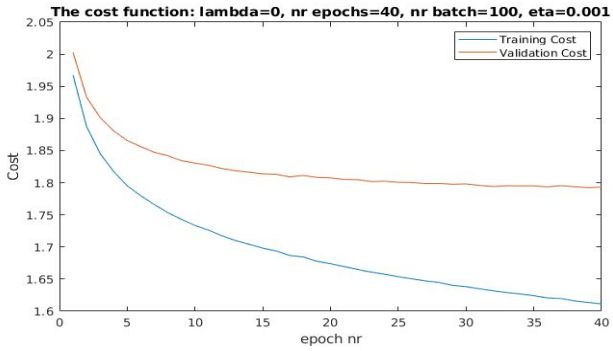
In order to check for the correctnes of the gradient I used the following formula:

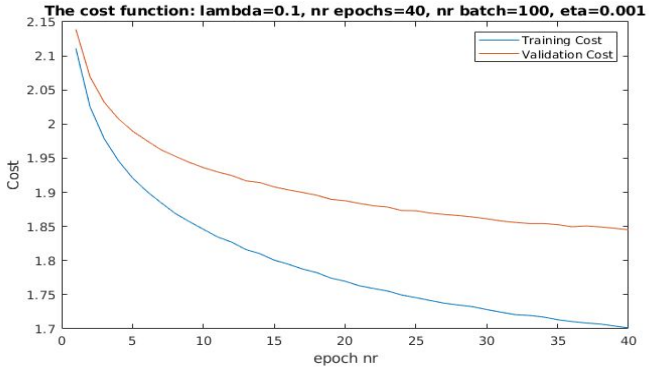
$$\frac{|g_a - g_n|}{\max(\text{eps}, |g_a| + |g_n|)} \quad \text{where eps a very small positive number}$$

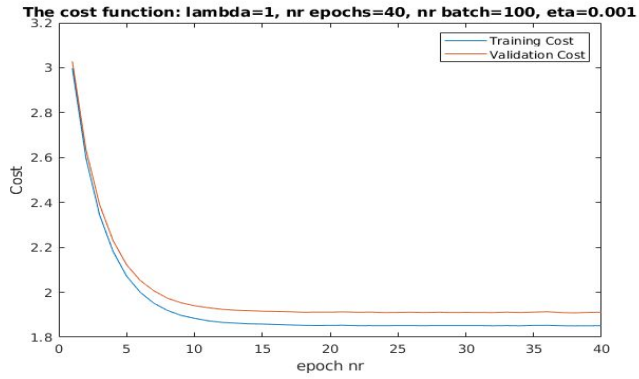
where g_a is the analytically computed gradient and g_n is the numerically computed gradient and made sure that it is small for each component of the gradient. To compute the gradients numerically I used the given function ***ComputeGradsNumSlow( ).***

For each of the following training scenarios the training samples are shuffled before each epoch.
From the plots we notice that a very big learning rate leads to oscellations of the cost function. On the other hand we notice that using regularization leads to more generalization as it reduces the variance but if the regularization is to high the bias increases and we are not able to achieve the same accuracies.









The accuracies for each case 1-4 are presented in the table below.

	lambda	Nr epochs	nr batch	eta	acc_train	acc_valid	acc_test
1)	0	40	100	0.1	0.439	0.2764	0.2735
2)	0	40	100	0.001	0.4548	0.3864	0.3895
3)	0.1	40	100	0.001	0.4478	0.3895	0.3961
4)	1	40	100	0.001	0.3974	0.3625	0.3755

Visualization of the Learned Weights