# Deep Learning, Assignment 1

Carlo Saccardi, saccardi@kth.se

April 2022

In this assignment the data in the CIFAR 10 file will be used, in particular data batch 1 will be used for training, the file data batch 2 for validation and the file test batch for testing. Each batch contains 10000 images (columns) of 3072 pixels (rows) and 10 possible labels (targets).
As pre-process the raw input data helps training, I will normalize the training, validation and test data with respect to the mean and standard deviation values computed from the training data. A neural network with one input layer and one output layer with the loss function cross entropy (and an L2 regularization term on the weight matrix will be used for training. The predicted class will correspond to the label with the highest predicted probability. As for the network parameters, the number of rows of the weights matrix is equal to the number of output nodes and, and the number of columns is equal to the number of input nodes. The bias consists in a column vector with as many rows as output nodes.

Then, for computing the gradients of the cross entropy loss function plus the L2 regularization term in mini-batch mode of gradient descent (back propagation) I have used the following equations:

$$\frac{\partial J}{\partial W} = \frac{\partial L}{\partial W} + \frac{\partial \lambda \|W\|^2}{\partial W} = gX^T + 2\lambda W \quad \text{where} \quad g = -(Y_{C\times N} - P_{C\times N})$$

$$\text{and} \quad \frac{\partial J}{\partial b} = \frac{\partial L}{\partial b} = \frac{1}{N}(g \times 1_{N\times 1})$$

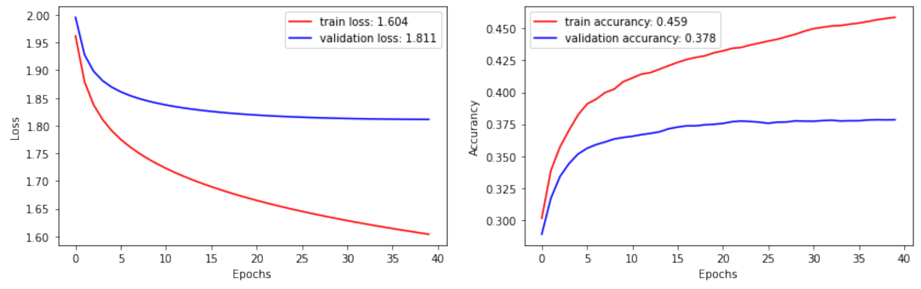
To compare the gradient obtained by the network (analytical method) with the gradient computed with the finite difference method (numerical method), I computed the relative error, and chcked that this value was small.

$$RelativeError = \frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)}$$

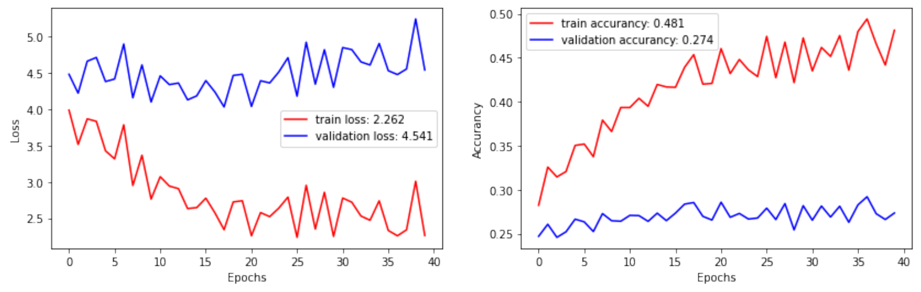
Every time I evaluated this measure, I obtained small values in the order of $e - 06, e - 07$

To evaluate the learning process I used the loss and accuracy metrics after each epoch of the vanilla mini-batch gradient descent algorithm for the 4 parametrization cases proposed. 8) A large learning rate ($\eta = 1$) leads to instability in the training process and the loss does not seem to converge. Thus, finding a good value for this parameter is fundamental for convergence. Regularization increases the training error and simultaneously reduces the gap between training and validation error. In this case, the net effect is always a higher validation error if $\lambda$ is increased, which is definitely not a desired outcome, but it's something that we can expect given that ridge regression usually only makes sense if we train models with high capacity on relatively small data-sets

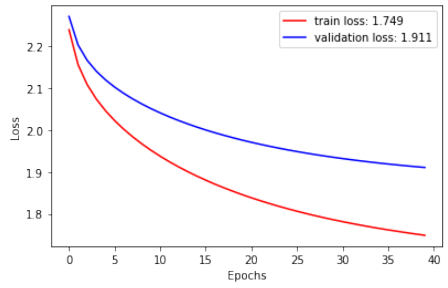
**lambda=0, n epochs=40, n batch=100, eta=.001**

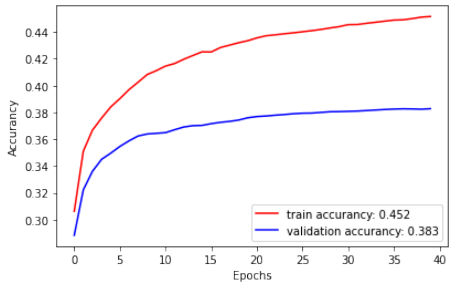


**lambda=0, n epochs=40, n batch=100, eta=.1**

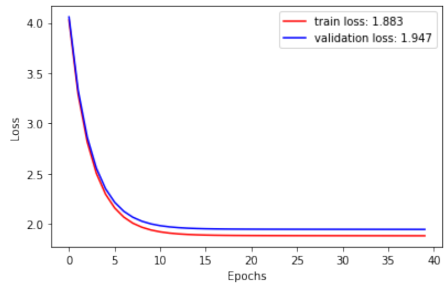


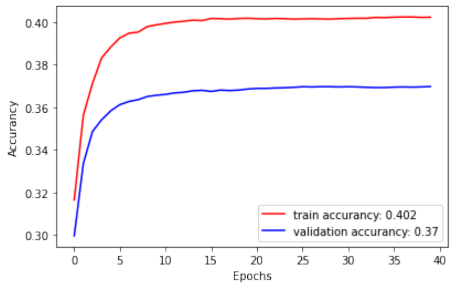
**lambda=0.1, n epochs=40, n batch=100, eta=.001**





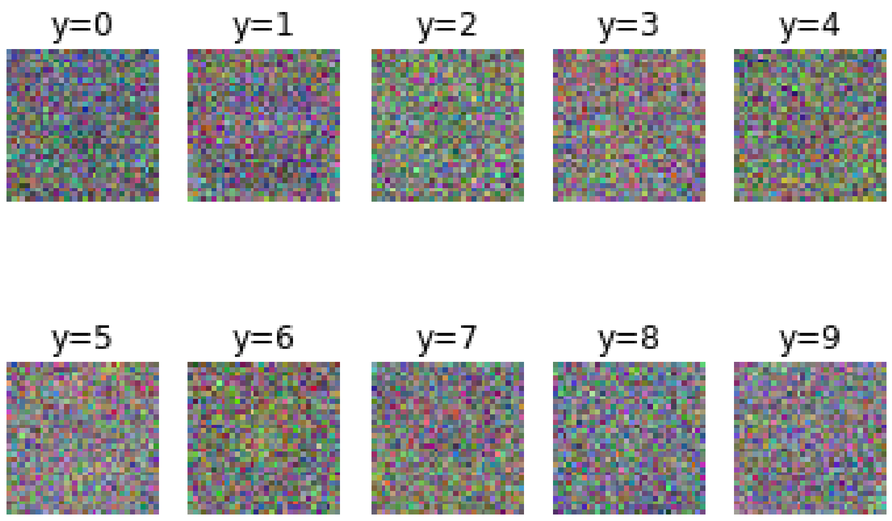
**lambda=1, n epochs=40, n batch=100, eta=.001**



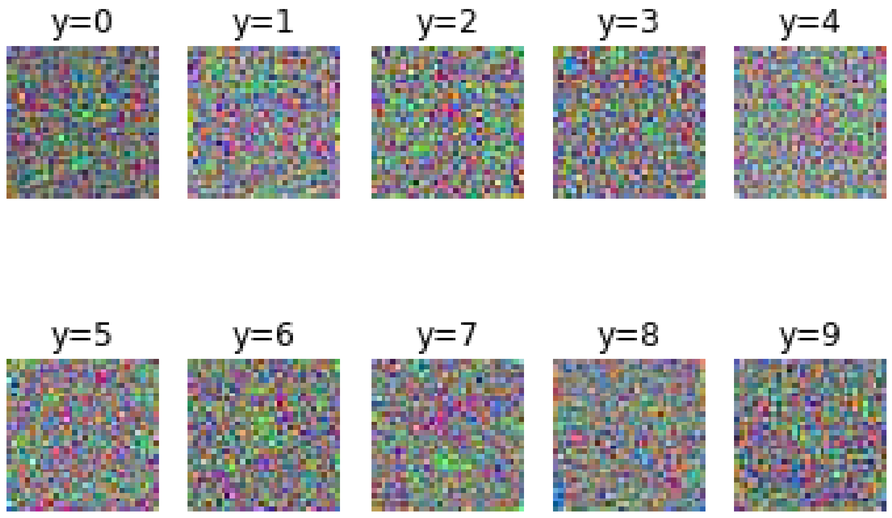


The next figures will show what class templates the network has learnt for each output node for the different parametrization cases. Since regularization shrinks the weights towards zero and forces them to be closer to each other, we notice smoother fugues for these cases. Also, for some of the classes, we can see the shape of the image or contour of objects described such as for car or horse.

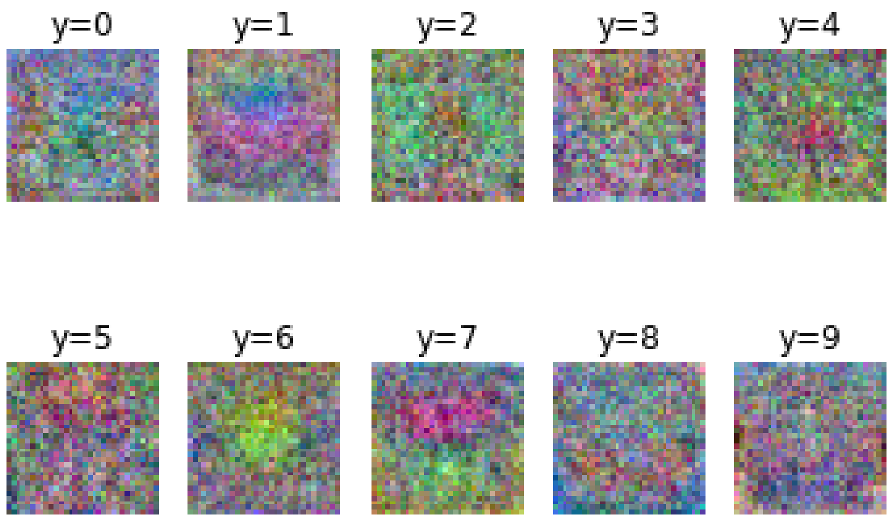
**lambda=0, n epochs=40, n batch=100, eta=.001**



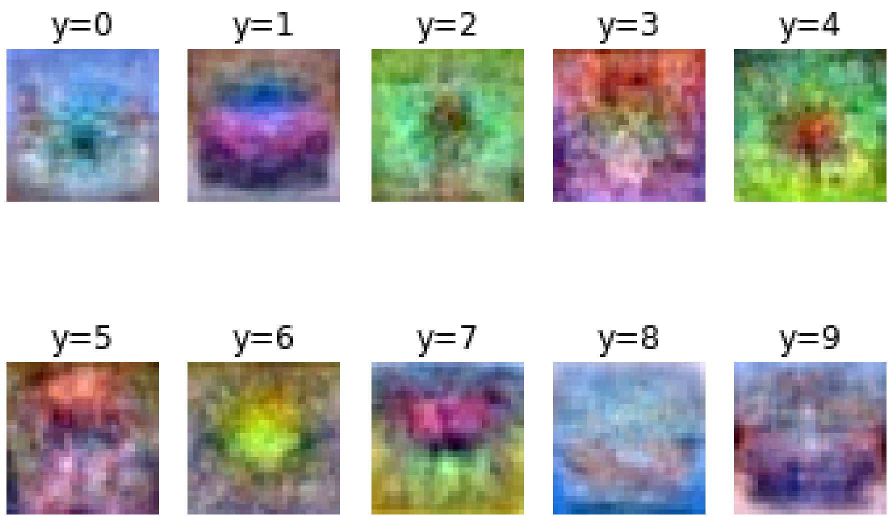
**lambda=0, n epochs=40, n batch=100, eta=.1**



**lambda=0.1, n epochs=40, n batch=100, eta=.001**



**lambda=1, n epochs=40, n batch=100, eta=.001**


y=0
y=1
y=2
y=3
y=4
y=5
y=6
y=7
y=8
y=9