

1 Introduction

This assignment consisted of implementing a one-layer Neural Network.

2 Methods

A 1-layer neural network was implemented from scratch using Python 3.6. The neural network is trained using backprop and minibatch gradient descent. As a data set CIFAR-10 is used, which consists of 10 000 32x32 images corresponding to 10 classes of objects. Different setups are compared and the influence of regularization and the learning rate is examined.

3 Results

In Table 1 one can see the final classification accuracy of the model after training for 40 epochs, with a minibatch size of 100 examples. Figure 1 and 2 depict the evolution of respectively the loss and the accuracy of the different setups. In Figure 3 one can see a visualization of the weight matrix after the training for one of the classes, corresponding to a car. Below the results are analyzed.

3.1 Learning rate

Three setups with learning rate 0.01 were tested, and one with learning rate 0.1. In Figure 1 and Figure 2 it is clearly visible that the learning rate of 0.1 is too high. The evolution of the loss and the accuracy is very unstable. This is probably due to the fact that the model will overshoot the minima, because the learning rate is too high. A learning rate of 0.01 causes much smoother behaviour in the evolution of the loss and the accuracy, so this value is better than the higher 0.1 value. However, more values should be tested if we want to determine the optimal learning rate.

3.2 Regularization

Three different values of the regularization parameter λ were tested: $\lambda=0$, $\lambda=0.1$ and $\lambda=1$. In Figure 2 it can be seen that the accuracy of the train set is far above the accuracy of the validation set. This implies that the model is overfitting when there is no regularization, which is very likely. In Figure 1, in the evolution of the loss, this is not that clearly visible though. In Figure 2 it can also be seen that in the setup with $\lambda=1$, the train accuracy is well below the validation accuracy. Also, it can be seen in Table 1 that the accuracy for this setup is very low for both train and test set. This implies that the model is underfitting, so this value of λ is probably too high. $\lambda=0.1$ seems to be the best setup, but again it is the case that more values should be tested before the optimal value of λ can be selected.

3.3 Weight visualization

In Figure 3 one can see a visualization of the weight matrix after the training for one of the classes, corresponding to a car. In the two rightmost setups the car can be recognized clearly: the model has learned a good representation of this class. However, this does not completely match the performance reported in Table 1. For example, the setup with $\lambda=1$ provides a very good picture of a car, but has a very low accuracy. I have no answer why this is the case. If I had more time, I would do more experiments to understand this.

Setup	Train accuracy	Test accuracy
$\lambda=0$ & $\text{lr}=0.01$	0.42	0.38
$\lambda=0$ & $\text{lr}=0.1$	0.25	0.24
$\lambda=0.1$ & $\text{lr}=0.01$	0.36	0.35
$\lambda=1$ & $\text{lr}=0.01$	0.24	0.24

Table 1: Accuracy results on train and test set with different setups

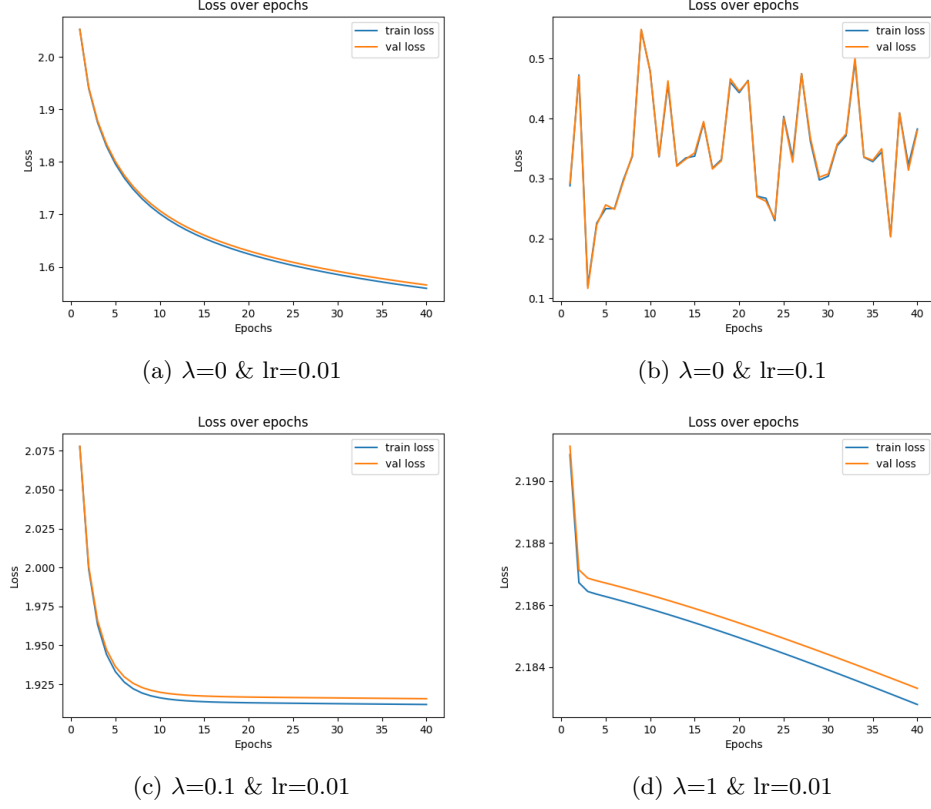


Figure 1: Evolution of the loss over epochs for the different setups (different values for λ and the learning rate (lr)).

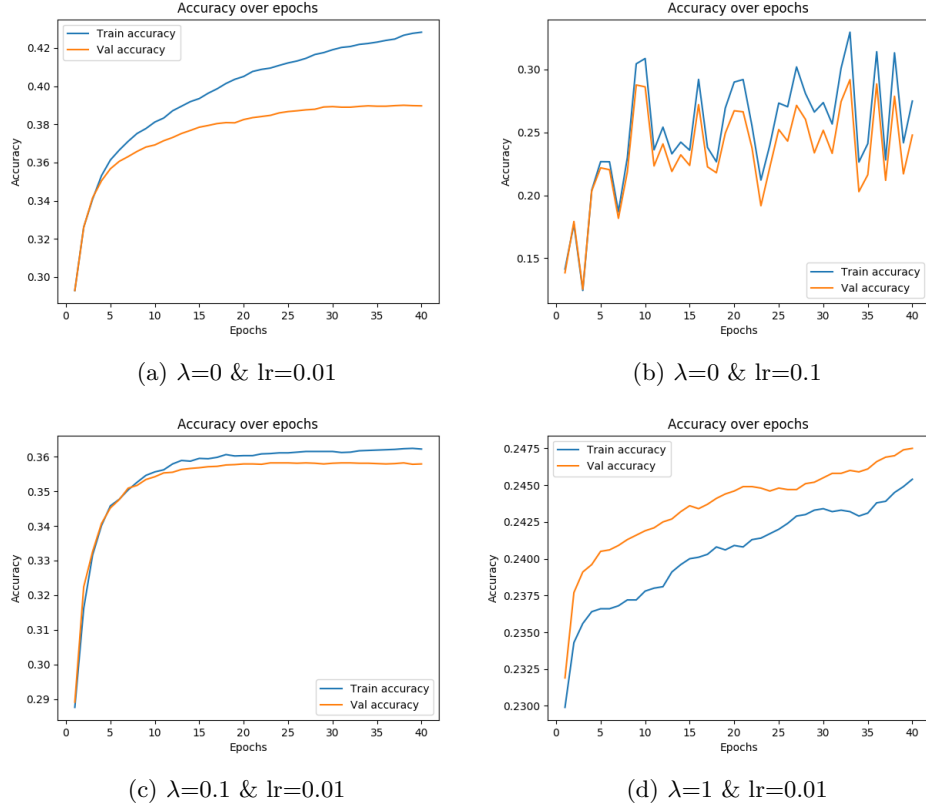


Figure 2: Evolution of the accuracy over epochs for the different setups (different values for λ and the learning rate (lr)).

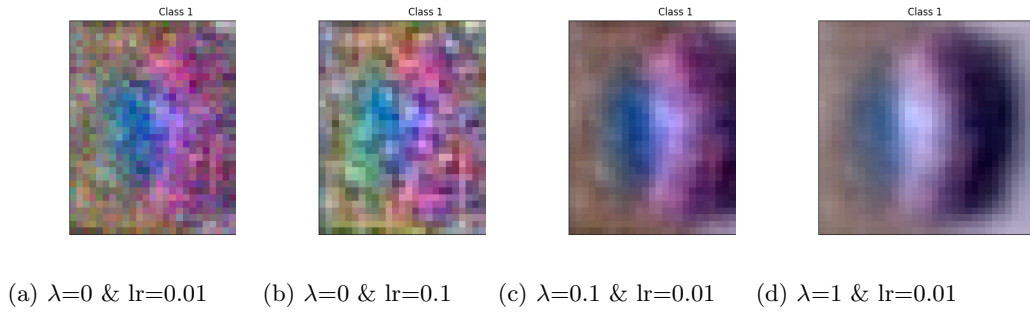


Figure 3: Visualization of the weight matrix given the different setups (different values for λ and the learning rate (lr)).