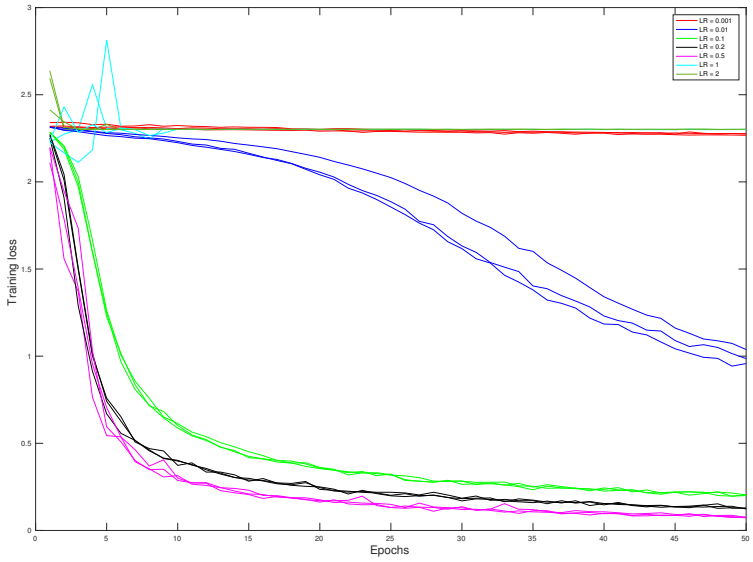
Exercise 3

Deep Learning lab

Silvio Galesso

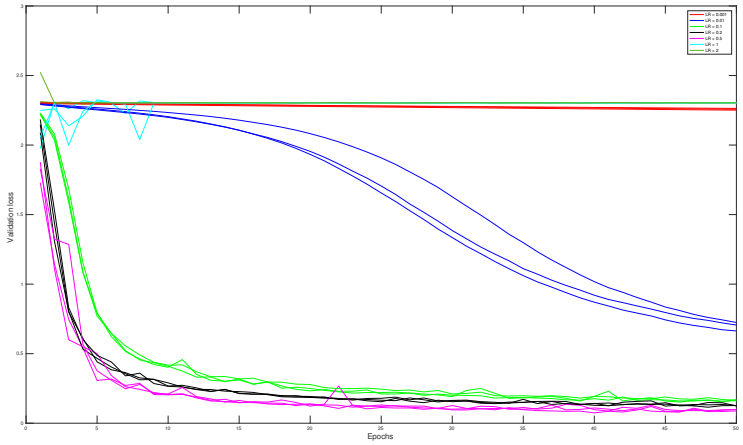
Scenario 1

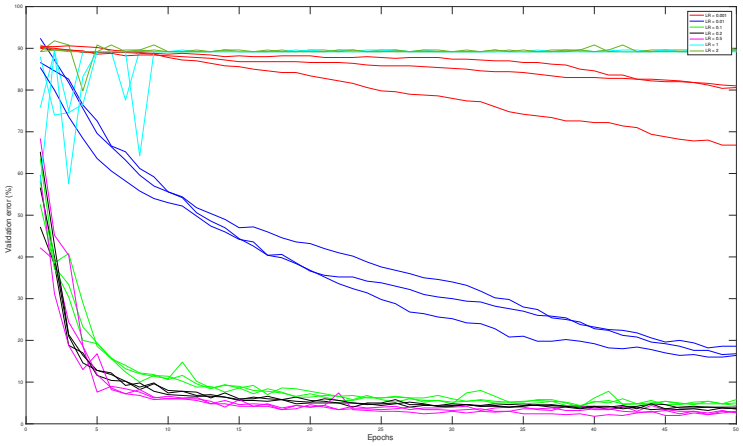


The learning rate sets the size of the steps taken in gradient descent, so the learning reates that work best are the ones that do steps small enough not to miss the minumum by overshooting it, while still providing a reasonably fast and effective descent. The overshooting phenomenon can be seen in the plot for the biggest rates $LR = \{1, 2\}$: for $LR = 2$ it's immediate, then the network sets to a constant loss not being able to hit a "valley" to descend. For $LR = 0.001$ (too small) the training loss decreases extremely slowly, therefore the network risks to get stuck in suboptimal minima.

Sources of stochasticity

- random weight initialization

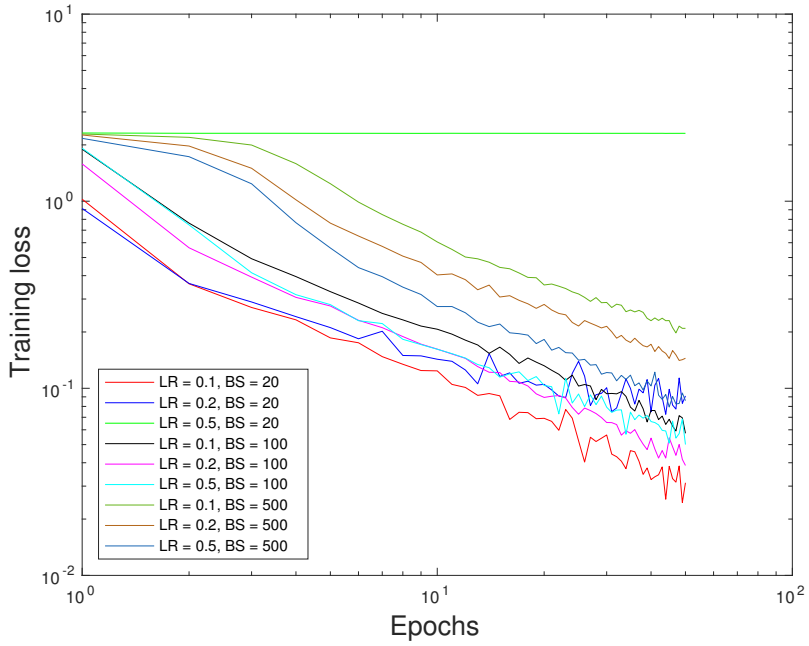


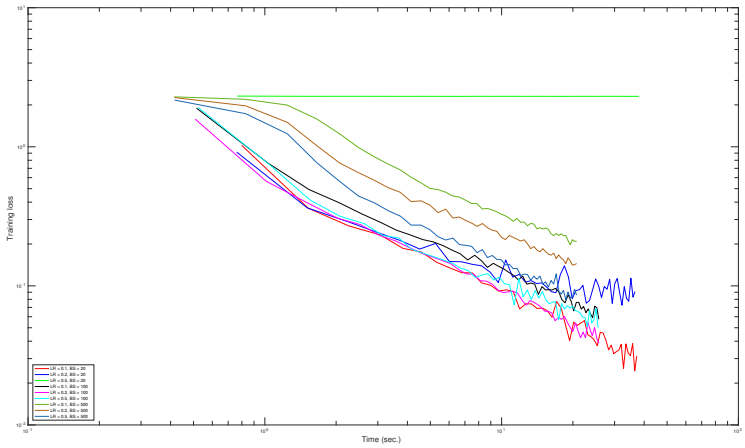


- SGD: randomly selected minibatches
- . . . ?

Scenario 2

With the smallest batch size of 20, the best rate is $LR = 0.1$, and this configuration also gives the best overall result. With $BS = 100$ the best learning rate is 0.2, and for $BS = 500$ is $LR = 0.5$. The best results therefore are obtained with directly proportional hyperparameters.



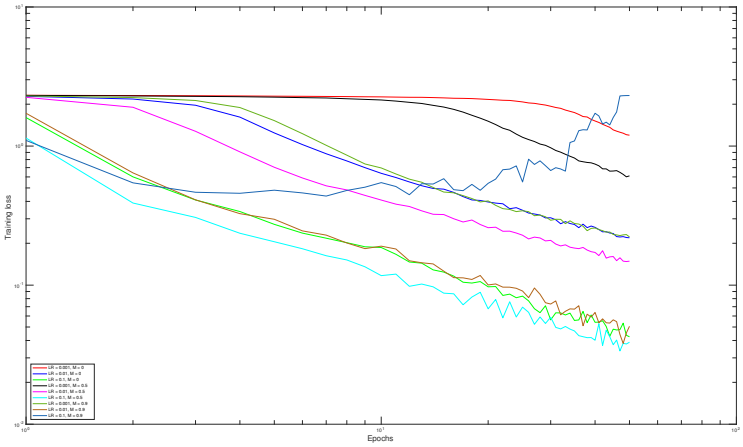


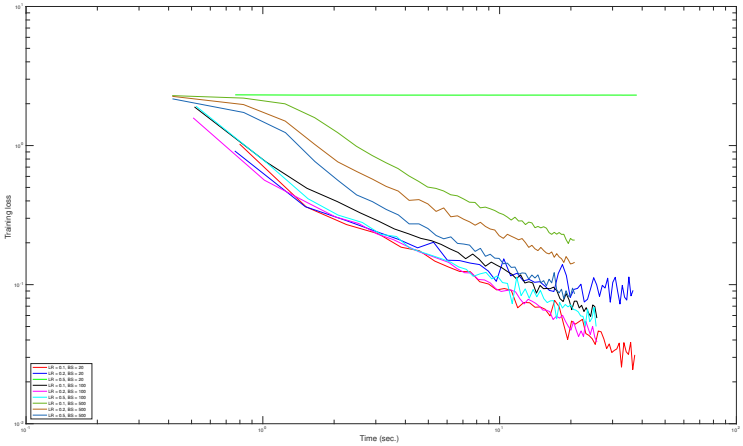
The smaller the batch size, the less accurate the gradient estimate on which the network trains (higher variance between different gradients), this means that it makes sense to use a smaller learning rate and not make overconfident descent steps on noisy gradients. The noisy descent can be already seen in the learning curve of $BS = 20, LR = 0.1$; with $BS = 20, LR = 0.2$ there is a

worse descent on average; with $BS = 20, LR = 0.5$ the network is completely unable to perform descent. Larger batch sizes have smoother curves but slower descents, because the network does less steps per epoch.

The setups have different training times, and they are not different by a factor of five, because the operations done on the batches have nonlinear time complexity w.r.t. the batch size.

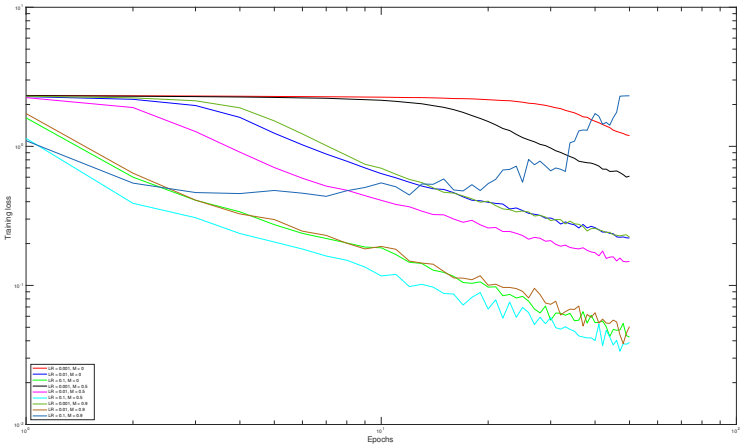
Scenario 3





The plots show that a balanced combination between learning rate and momentum works best. The best hyperparameter configurations are, in decreasing order, $LR = 0.1, M = 0.5$, $LR = 0.01, M = 0.9$ and $LR = 0.1, M = 0$, showing that the momentum can help slower learning rates to get a better performance. This is evident watching the red, black, and dark green curves, which are relative to the smallest learning rate (0.001): with this rate the network still learns very slowly, but we can see the positive effect of the momentum. This effect is to help the descent algorithm to overcome local minima and to speed up the training accumulating velocity when possible. The only case where momentum is counterproductive is with $LR = 0.1$: in this case the combination between (relatively) high LR and high momentum make the network diverge by "climbing" uphill the loss function.

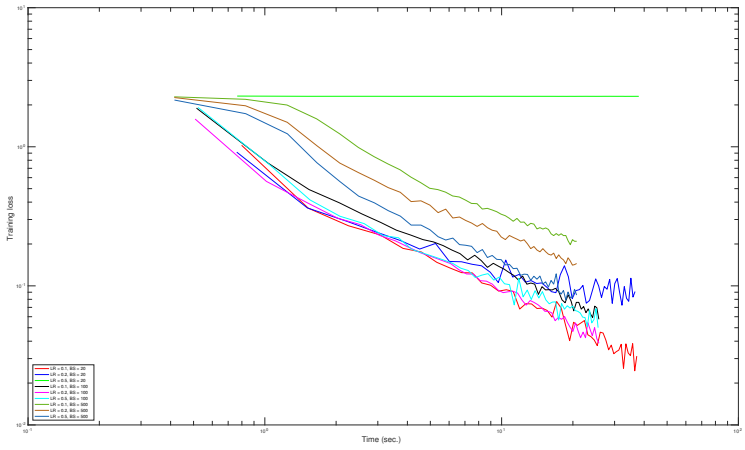
Scenario 4



Scenario 5

Best configurations The best results (validation accuracy $> 99.20\%$) for a run of the 5th scenario were:

nfilters = 47, batch size= 97, M = 0.265, LR = 0.25: validation accuracy = 99.21


Training loss
Time (sec.)
LR = 0.1, BS = 20
LR = 0.2, BS = 20
LR = 0.5, BS = 20
LR = 0.1, BS = 100
LR = 0.2, BS = 100
LR = 0.5, BS = 100
LR = 0.1, BS = 500
LR = 0.2, BS = 500
LR = 0.5, BS = 500