# Tune the network exercise

**Shape of the network**

In most cases it can be said that the shape of the work, when it becomes deeper or wider, that accuracy did go up. However, it also seems that the network can be too wide and that this had a negative effect on accuracy. I saw this effect mainly when only using a small number of epochs. The effect is cancelled when the number of epochs is enlarged. At 10 epochs the larger networks were the most accurate, but also the most time consuming to train.

In several cases when the network started with more width than the actual number of input based on an estimated number on the data 28\*28 the accuracy was less than network with a lower width.

More depth did not always mean more accuracy. Again, by adding more epochs this was solved.

**The number of training steps**

In my opinion the number of training steps has the most impact on the accuracy of the network, this could be tuned by batch size, or number of epochs. The more training steps the better the overall accuracy. This could be jumping around the optimum though because of the gradient descent. This also influences the training time the most.

**Changing the learning rate**

In some cases, the final accuracy became better by changing the learning rate. But often due to early stopping the learning process was aborted early on.

**Performance difference between small and larger network**

Worth noting is that even the really small networks had a descent performance. 16 8 4 at ten epochs has an accuracy of 0.8396. When compared to the somewhat larger network like 1024, 512, 256 there is only 6% difference in performance while the computational time is a lot larger 10\* the smaller network.

**Intuition for now:**

Larger networks are in general more accurate than smaller ones

More training steps (epochs and batch size) mean more accuracy, this affected by learning step size.

Base performance can be quickly assessed by a few epochs.