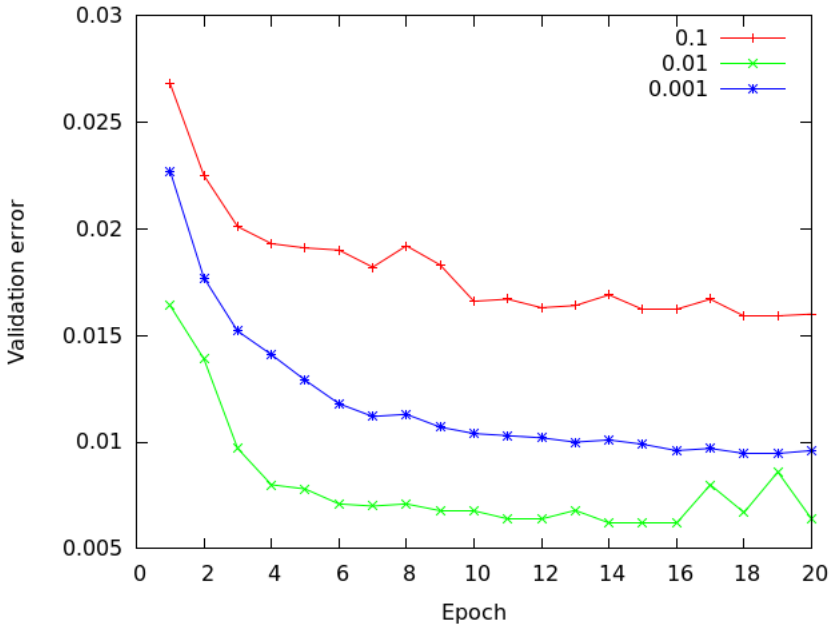
Deep Learning Lab Course – Report for assignment 3

The specification for the neural network used for this assignment can be found in `autoencoder_nn.json` and passed to `neural_network.py` as command line argument to train the network on the MNIST dataset. The learning curves for the three different learning rates are shown in figure 1.

Figure 1: Learning curves



One can observe that the best result is achieved with a medium learning rate, in this case 0.01. If the learning rate gets too large or too small, the performance of the autoencoder starts to degrade. It might also be worth mentioning that I observed similar effects when adjusting the standard deviation of the normal distribution used for the weight initialization: for a too low or too high value, the learning procedure would quickly get stuck, and performance would not improve at all (instead fluctuate around a rather high value). Although I didn't try out a wide range of values, it appears that the combination of both the learning rate and weight initialization parameters together has a high impact on the performance of the network.

The autoencoder outputs are visualized in figure 2, which contains randomly sampled images from the validation set (on the left side of each pair) and the output of the network for the corresponding image (on the right side of each pair). One can observe that most examples are reconstructed almost perfectly, although the result is sometimes slightly smoother and blurred (which can be seen very well in the first example, where the output of the pixelated 0 is very smooth).
When adding noise to the input images, the noise is almost completely gone in the output, as can be seen in figure 3. However, the borders of the numbers in the output images are now often uneven. This is probably the case because the network has learned that multiple high intensities in the same small area make up part of a pen stroke. The noise at the stroke borders makes it difficult for the network to figure out the exact borders of such an area, leading to uneven strokes. This guess is further reinforced by the observation that the network sometimes falsely makes a stroke in the output image if there is a random cluster of high intensities in the input (see 2nd example of the 2nd row).
Anyhow, this experiment shows that autoencoders could be used as a kind of "specialized denoiser". General denoisers which just smooth over the input image do not know whether a change in intensity is caused by noise or by an edge in the image, thus being limited in their performance. By having a neural network learn the underlying structure of a certain kind of images, it could differentiate between these two cases more easily and therefore provide a much cleaner denoised output image.

Figure 2: Random samples and their corresponding output from the autoencoder

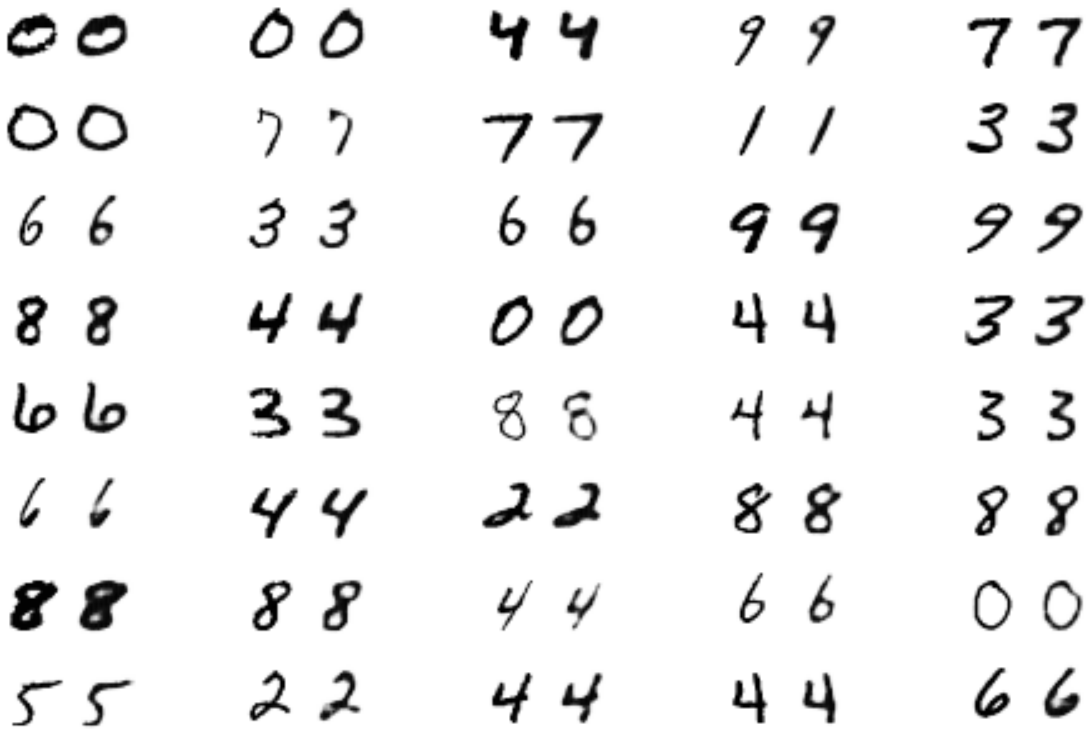


Figure 3: Random noisy samples and their corresponding output from the autoencoder