Deep Learning Lab Course
Machine Learning and Computer Vision Track
Exercise 3

Artemij Amiranashvili
(send the report to amiranas@cs.uni-freiburg.de)

Due: November 28, 2017

The goal of this exercise is to train a fully convolutional autoencoder on the MNIST dataset. An autoencoder is trained to recreate the given input to the network. At the end, send us a small report (1 or 2 pages) with the below described plots and the implementation.

1 Implementation of an Autoencoder in Tensorflow

First we implement a network with the following architecture:

- Input (gray MNIST image)
- Convolution (8 filters)
- Max pooling
- Convolution (4 filters)
- Max pooling
- Convolution (2 filters)
- Transposed convolution (4 filters)
- Convolution (4 filters)
- Transposed convolution (8 filters)
- Convolution (8 filters)
- Output Convolution (1 filter)

Each convolution layer uses $[3 \times 3]$ filters with stride 1 and the output convolution uses a $[1 \times 1]$ filter with stride 1. Each max pooling or transposed convolution layer uses a pooling or filter size of $[2 \times 2]$ and a stride of 2. We use the 'same' padding of Tensorflow for each layer. Therefore the convolutions will not change the resolution of the features. Each Max pooling will reduce the feature resolution by half and each transposed convolution will double the features resolution because of their strides of 2. Altogether the output resolution should be the same as the input resolution.

2 Training of the Autoencoder

We train the network by optimizing the sum of L2 losses between each pixel of the network output tensor and the gray value of the input MNIST image of the according pixel. We divide the total loss by the total number of pixels. No softmax operation is needed. In tensorflow this can be implemented by using:

```
loss = tf.reduce_mean(
     tf.square(network_output - input_image_placeholder))
```

We scale every MNIST image gray value to values between 0 and 1. The MNIST image labels are not used for this exercise.

Use a batch size of 64 and the Adam optimizer. Try out different learning rates: 0.1, 0.01, 0.001. Plot the according learning curves. After the training, generate a sample of MNIST images that the autoencoder creates and plot them.

Optional task

Try to feed noisy images of MNIST numbers (add a small random number to each pixel) to a trained autoencoder and plot the noisy images compared to the network output images. What is the reason for the result?