Karlsruher Institut für Technologie

Due Date: Computerpraktikum am 27.07. und 28.07.2023

**Verantwortlich für Vorlesung bzw. Computerpraktikum:**
Dr. P. Goldenzweig, P.-D. Dr. R. Wolf, Dr. S. Stefkova
**Tutoren:** Dr. G. De Pietro, Dr. R. Quishpe, Dr. S. Mitra, G. Heine, A. Brusamolino, T. Voigtländer

# Computerpraktikum zur Vorlesung Moderne Methoden der Datenanalyse - Blatt 10

## Exercise 10: Deep Learning

The classification of handwritten digits is a standard problem in the field of image classification. In this exercise, we will process labeled images of handwritten digits from the MNIST[1] dataset in order to train and test (deep) neural networks on this task. The goal of this exercise is for you to dive into a state-of-the-art software package for large-scale deep learning, to experiment with your own neural-network designs and to compare your results with modern setups.

`TensorFlow` is one of the most popular and powerful tools in the machine learning community and can be used to build, train and execute large-scale machine-learning models. The core concept of `TensorFlow` is the representation of the information flow as tensors in a graph. In this exercise the wrapper `Keras` is used, which hides this concept to a large extent and makes the library much easier to use.

The exercise is shipped with a script for the download of relevant data and one notebook. All needed software such as TensorFlow (`www.tensorflow.org`) and Keras (www.keras.io) is already installed on the Jupyter Machine.

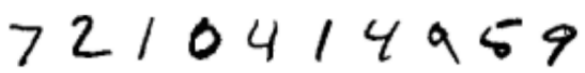

Figure 1: Example images from the MNIST dataset

The MNIST dataset (`http://yann.lecun.com/exdb/mnist/`) contains a total of 70,000 images of handwritten digits. The images are in greyscale with $28 \times 28$ pixels each (see Figure 1 for some examples). Execute the script `download_dataset.py` to download and extract the dataset as binary. The script also converts some example images from the binary dataset as `png` files (greyscale-inverted images of Figure 1). Have a look at the example images and at the code.

In preparation of the training (Exercise 10.1), read the code in the function `train()` in the Jupyter notebook and identify the part, where the machine learning model is defined. The model contains many components of modern architectures, e.g., convolutional, dense, and maximum pooling layers. Also specified in the code are the loss function, the optimizer algorithm, and the validation metrics used for the training. The full documentation is available on the `Keras` webpage (www.keras.io).

[1] (Modified) National Institute of Standards and Technology

- **Exercise 10.1: Training and Testing** **obligatory**

  Explain the meaning and function of the various parts of the example model and understand how the total number of trainable parameters, 1765, comes about. Hint: Look at the output of the code line `model.summary()` and keep in mind that there are bias terms. What would be the number of parameters for a conventional (dense) neural network with one hidden layer of n nodes and 10 outputs?

  Now, modify the model defined in `train()` and try to achieve the best global accuracy. Hint: You will need to increase the model capacity, e.g., larger number of convolution filters or additional dense layers, and the number of epochs. With increasing model capacity, you will quickly understand why GPUs play such a big role in machine learning. What would be a good method to evaluate the optimal number of training epochs? Hint: Plot the accuracy of the model on training and validation sets as a function of training epochs. Can you explain the worse training accuracy compared to the validation accuracy?

  For your trained model, produce an estimate of your achieved accuracy by running the function `apply()` manually on about twenty images, i.e. `example_input_*.png`. How does your estimate compare with the result on the test dataset computed with the function `test()`? Do the results match?

- **Exercise 10.2: Application on other data** **voluntary**

  Use `GIMP` (or any other graphics editor) to create images of your own handwritten digits and evaluate whether the performance you experience with your own example images matches the performance achieved during training on the MNIST dataset. You need to create greyscale `png` images with $28 \times 28$ pixels, the background of the image has to be black and the digits have to be written in white color. The file `your_own_digit.xcf` can be used as template. Does the model classify your images correctly? If not, what can be possible reasons that it does not work as expected?

  Have a look at the website http://yann.lecun.com/exdb/mnist/, which holds a leaderboard for the test dataset with the performance of modern (deep) machine learning models. You might want to compare your model with popular computer vision models such as `LeNet-5`, `VGG-16`, `AlexNet` or `Inception-v4` to get an idea of the complexity of modern neural network architectures.