Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Sebastian Schmoll, Sabrina Friedl

Munich, 10.12.2018

Deep Learning and Artificial Intelligence

WS 2018/19

Exercise 8: Tooling

Exercise 8-1 Testing & Debugging

Download the git repository `deep_learning_and_ai_tooling_lecture` with the examples from the lecture: https://github.com/dekromp/deep_learning_and_ai_tooling_lecture. Make sure you have all the required packages installed. Open the project in your favourite IDE.

(a) Navigate to the folder `code_style` and look at the example test file. What does it do? Run the test by using the command `pytest documented_function_example_test.py`.

(b) Why are tests important in Data Science? How can we ensure reproducability of our results?

(c) Navigate to the folder `debug_tensorflow` and run the file `buggy_tensorflow_nan_output.py`. What is the problem?

(d) Have a look at the code. How is the tensorflow debugger (tfdbg) integrated? Run the file in debugger mode by executing `python buggy_tensorflow_nan_output.py --debug` in the terminal. Now you're in the command line interface of tfdbg. Follow the steps in the lecture slides p.38 – 42. To navigate back and forth use the dashed arrows at the top left corner of the screen.

(e) What caused the NaNs in the example and how was the problem fixed?

Exercise 8-2 Tensorboard

(a) Navigate to the folder `tensorboard` and look at the file `tensorflow_example.py`. How can certain tensor values be recorded over time with tf.summary?

(b) Run the file with different learning rates and save the summaries in corresponding directories. For example, for a learning rate of 0.1, execute `python tensorflow_example.py -lr=0.1 -d= ./experiments/lr_0.1`.

(c) You should have a folder 'experiments' with different subfolders for each of your runs now. To launch tensorboard use the command `tensorboard --logdir=path/to/log-directory` and click on the displayed link. Use `./experiments` as path to show all runs at once. Look at the different plots provided by tensorboard.

(d) How can the logging be achieved when using keras? *Hint*: Look at the file `/frameworks/keras.py`.

Exercise 8-3 Denoising Autoencoder

(a) Write a class `Autoencoder` for image data using tensorflow. It should have a method `fit()` which takes some data as input and trains the model. Since we want a denoising autoencoder, some noise should be added before encoding and decoding the data. The loss should however be calculated between the original and the reproduced data. You can stack several convolutional and maxpooling layers (`tf.layers.conv2d, tf.layers.max_pooling2d`) to reduce the dimensions in the encoder and use transposed convolutional layers (`tf.layers.conv2d_transpose`) and padding (`tf.pad`) in the decoder.

(b) Define meaningful scopes and integrate the tensorflow debugger. Play around with it.

(c) Generate some tensorflow summaries (e.g. for the loss) and visualize the results in Tensorboard.

(d) Alternatively (or additionally), you can define the same model by using the Keras Model class (functional) API. For a quick introduction refer to: https://keras.io/getting-started/functional-api-guide/.

Exercise 8-4 PyTorch Introduction

PyTorch is a programming library for machine learning algorithms similar to Tensorflow, Keras, etc. On the lecture web-site you find a Jupyter notebook with a small PyTorch introduction.