Delira: A High-Level Framework for Deep Learning in Medical Image Analysis

Christoph Haarburger¹, Justus Schock¹, Michael Baumgartner¹, Oliver Rippel¹, and Dorit Merhof¹

¹ Institute of Imaging and Computer Vision, RWTH Aachen University, Germany

Summary

Medical image analysis research using deep neural networks often involves the development of problem-specific network architectures and the evaluation of models on several datasets. Contemporary deep learning frameworks such as PyTorch (Paszke et al., 2017) and Tensorflow (Abadi et al., 2016), however, operate on a low level, such that for comparing different models on several datasets, a lot of boilerplate code is necessary. So far, this boilerplate code is often copied and pasted for new projects and experiments. Reference implementations of new methods may be implemented in either PyTorch or Tensorflow, leading to a lot of friction when comparing two methods that are implemented in different low-level frameworks. Moreover, data augmentation for 3D medical images such as from computed tomography or magnetic resonance images is not natively supported by many low-level frameworks. As a result, stand-alone data augmentation solutions are often applied (MIC-DKFZ, 2019).

In order to integrate high level functionalities such as logging, data structures for image datasets, data augmentation, trainer classes and model save and load functionality in a way that is agnostic with respect to the low-level framework, we developed Delira (Deep Learning in Radiology).

Delira consists of several subpackages and modules that are structured into data_loading, io, logging, models, training and utils. This modular structure enables the reuse of datasets and data loading pipelines across different models. Moreover, reference models for classification, segmentation and data synthesis problems using generative adversarial networks (Goodfellow et al., 2014) are provided in the models subpackage.

The actual training is carried out using a NetworkTrainer class that implements the actual training routine given a dataset and model. An Experiment class runs the training using NetworkTrainer, e.g. in a cross validation scheme. A quick tutorial showing how the most important data structures interact with each other and HTML documentation is provided at https://delira.readthedocs.io/en/master/classification_pytorch.html.

Currently, PyTorch and Tensorflow backends are supported and tested. Adding more backends is easily possible if needed.

Delira is released under BSD Clause-2 license. The source code can be found at https://github.com/justusschock/delira.

References


