
Deep Learning Toolkit (DLTK) for Medical Imaging

chat on [gitter](#) coverage [10%](#) chat on [gitter](#)



DLTK is a neural networks toolkit written in python, on top of TensorFlow. It is developed to enable fast prototyping with a low entry threshold and ensure reproducibility in image analysis applications, with a particular focus on medical imaging. Its goal is to provide the community with state of the art methods and models and to accelerate research in this exciting field.

Documentation

The DLTK API can be found [here](#)

Referencing and citing DLTK

If you use DLTK in your work please refer to this citation for the current version:

```
1 @article{pawlowski2017state,  
2   title={DLTK: State of the Art Reference Implementations for Deep  
   Learning on Medical Images},  
3   author={Nick Pawłowski and S. Ira Ktena, and Matthew C.H. Lee and  
   Bernhard Kainz and Daniel Rueckert and Ben Glocker and Martin  
   Rajchl},  
4   journal={arXiv preprint arXiv:1711.06853},  
5   year={2017}  
6 }
```

If you use any application from the DLTk Model Zoo, additionally refer to the respective README.md files in the applications' folder to comply with its authors' instructions on referencing.

Introduction to Biomedical Image Analysis

To ease into the subject, we wrote a quick overview blog entry (12 min read) for the new TensorFlow blog. It covers some of the speciality information required for working with medical images and we suggest to read it, if you are new to the topic. The code we refer to in the blog can be found in `examples/tutorials` and `examples/applications`.

Installation

1. Setup a virtual environment and activate it. Although DLTk<=0.2.1 supports and python 2.7, we will not support it future releases, similarly to our dependencies (i.e. SciPy, NumPy). We highly recommend using python3. If you intend to run this on machines with different system versions, use the `--always-copy` flag:

```
1 virtualenv -p python3 --always-copy venv_tf
2 source venv_tf/bin/activate
```

2. Install TensorFlow (>=1.4.0) (preferred: with GPU support) for your system as described here:

```
1 pip install "tensorflow-gpu>=1.4.0"
```

3. Install DLTk: There are two installation options available: You can simply install dlTK as is from pypi via

```
1 pip install dlTK
```

or you can clone the source and install DLTk in edit mode (preferred):

```
1 cd MY_WORKSPACE_DIRECTORY
2 git clone https://github.com/DLTk/DLTk.git
3 cd DLTk
4 pip install -e .
```

This will allow you to modify the actual DLTk source code and import that modified source whenever you need it via `import dlTK`.

Start playing

1. Downloading example data You will find download and preprocessing scripts for publicly available datasets in `data`. To download the IXI HH dataset, navigate to `data/IXI_HH` and run the

download script with `python download_IXI_HH.py`.

2. Tutorial notebooks In `examples/tutorials` you will find tutorial notebooks to better understand on how DLTK interfaces with TensorFlow, how to write custom read functions and how to write your own `model_fn`.

To run a notebook, navigate to the DLTK source root folder and open a notebook server on `MY_PORT` (default 8888):

```
1 cd MY_WORKSPACE_DIRECTORY/DLTK
2 jupyter notebook --ip=* --port MY_PORT
```


Open a browser and enter the address `http://localhost:MY_PORT` or `http://MY_DOMAIN_NAME:MY_PORT`. You can then navigate to a notebook in `examples/tutorials`, open it (c.f. extension `.ipynb`) and modify or run it.

3. Example applications There are several example applications in `examples/applications` using the data in 1. Each folder contains an experimental setup with an application. **Please note that these are not tuned to high performance, but rather to showcase how to produce functioning scripts with DLTK models.** For additional notes and expected results, refer to the notes in the individual example's README.md.

DLTK Model Zoo

We also provide a zoo with (re-)implementations of current research methodology in a separate repository DLTK/models. Each model in the zoo is maintained by the respective authors and implementations often differ to those in `examples/applications`. For instructions and information on the individual application in the zoo, please refer to the respective README.md files.

How to contribute

We appreciate any contributions to the DLTK and its Model Zoo. If you have improvements, features or patches, please send us your pull requests! You can find specific instructions on how to issue a PR on github here. Feel free to open an issue if you find a bug or directly come chat with us on our gitter channel .

Basic contribution guidelines

- Python coding style: Like TensorFlow, we loosely adhere to google coding style and google docstrings.

-
- Entirely new features should be committed to `dltk/contrib` before we can sensibly integrate it into the core.
 - Standalone problem-specific applications or (re-)implementations of published methods should be committed to the DLTk Model Zoo repo and provide a README.md file with author/coder contact information.

Running tests locally To run the tests on your machine, you can install the `tests` extras by running `pip install -e '.[tests]'` inside the DLTk root directory. This will install all necessary dependencies for testing. You can then run `pytest --cov dltk --flake8 --cov-append` to see whether your code passes.

Building docs locally To run the tests on your machine, you can install the `docs` extras by running `pip install -e '.[docs]'` inside the DLTk root directory. This will install all necessary dependencies for the documentation. You can then run `make -C docs html` to build the documentation. You can access this documentation in a web browser of your choice by pointing it at `docs/build/html/index.html`.

The team

DLTK is currently maintained by @pawni and @mrajchl with greatly appreciated contributions coming from individual researchers and engineers listed here in alphabetical order: @CarloBiffi @ericspod @ghisvail @mauinz @michaeld123 @sk1712

License

See LICENSE

Acknowledgments

We would like to thank NVIDIA GPU Computing for providing us with hardware for our research.