

Artifact Description Appendix: Open Neural Networks for Drug Discovery

A. Abstract

This description contains the information needed to launch some experiments of the SC18 paper “OpeNNdd: Open Neural Networks for Drug Discovery”. Specifically, we explain how to prepare and run experiments detailed in Section 2.3 to reproduce the results in Section 3.1.

B. Description

1) Checklist:

- **Algorithm:** 3D Convolutional Neural Network
- **Program:** Smina, Python 3.6 libraries
- **Data set:** Molecular Mechanics Dataset (Hosting Site TBD)
- **Run-time environment:** Anaconda 5.2
- **Hardware:** Oak Ridge National Laboratory Summitdev System
- **Output:** Prediction of pKD for a given protein ligand interaction.
- **Experiment workflow:** Download ligands and protein files, clone OpeNNdd git repo, voxelize and store active site in hdf5 file, voxelize poses with cached active site and store voxelized data in hdf5 files, use OpeNNdd_Model class to build 3D convolutional neural network, train and test network.
- **Experiment customization:** Type of active-Site, voxel grid size, voxel grid resolution, neural network hyperparameters, cpu or gpu mode for network.
- **Publicly available:** Yes

2) How software can be obtained (if available):

OpeNNdd can be cloned from github via the following link: <https://github.com/Active-Site-Project/OpeNNdd>

3) Hardware dependencies: None

4) **Software dependencies:** NumPy (linear algebra and array processing), Tensorflow (neural networks), H5py (used for data storage and retrieval), Tqdm (used to monitor training/testing progress), Matplotlib (saving model metrics), Pybel (used for parsing SDF files).

C. Installation

1. Download SDF files and Active Site PDB from hosting site.
2. Use pip to install software dependencies into home or virtual environment.

3. Clone OpeNNdd and navigate to the src directory.

```
$ git clone
https://github.com/Active-Site-
Project/OpeNNdd
$ cd OpeNNdd/src
```

D. Experiment workflow

1. Run command to voxelize the active-site (will place hdf5 file wherever the script is run)

```
$ python pyActiveVox.py
path/to/ActiveSitePDB VoxRes VoxX
VoxY VoxZ
```

- VoxRes = Length of cubic voxel in Angstroms
- VoxX = Number of voxels in the X direction
- VoxY = Number of voxels in the Y direction
- VoxZ = Number of voxels in the Z direction

Run once active-site is voxelized

```
$ mpiexec -n numOfMPI python3
pyVox.py path/to/VoxelizedActiveSite
path/to/OutputFolder
path/to/DirWithPoses
path/to/ElectronClouds
numOfSdfsPerMPI VoxX VoxY VoxZ VoxRes
```

- numOfMPI = How many parallel instances of the program to run
- numOfSdfsPerMPI = Number of sdf files to be voxelized in each MPI instance
- VoxRes = Length of cubic voxel in Angstroms
- VoxX = Number of voxels in the X direction
- VoxY = Number of voxels in the Y direction
- VoxZ = Number of voxels in the Z direction\

2. If numOfMPI is specified to be greater than 1, an assembly of externally linked hdf5 files will be generated. Otherwise a single hdf5 file will be created.

3. Run command (for models used to get results in Section 3.1)*:

```
$ python opeNNdd_model
path/to/hdf5_file
path/to/storage_folder mode
db_mode
```

- `mode = 'cpu' or 'gpu'`, specifies whether training/testing should be conducted on a CPU or GPU
- `db_mode` specifies the database type that will be read from by the model. For an externally linked hdf5 database set 'l' (without quotes) for this argument and for an unlinked database set 'ru' for this argument ('ru' is also the default)
- `storage_folder` is a path to a directory where the model and model metrics will be saved.
- `hdf5_file` is a path to the voxelized dataset.

**For declaring a model class with different hyperparameters, see the team wiki at <https://github.com/Active-Site-Project/OpeNNdd/wiki>*

- For the paper, the specific model architecture was declared using the `OpeNNdd_Model` class as such (see Section 2.3 for specific architecture details)*: **All hyperparameters in this model declaration adjustable. For more specific documentation please refer to the wiki*

E. Evaluation and expected result

- Saved model can be found at `storage_folder/tmp/model_id/`
- Model metrics (error, graphs, model architecture) can be found at `storage_folder/logs/model_id/metrics/metrics.txt`
- Graphs of the training and validation error per epoch, as well as actual vs predicted values on a testing set can also be found in the metrics folder. (see wiki on more analysis)

F. Experiment customization

- In the voxelization process, the size and resolution of the voxel grid can be customized. The number of MPI processes and the number of SDF files per hdf5 file can also be altered to optimize performance.

```
model=OpeNNdd_Model(path/to/hdf5_file,
CHANNELS, [32,64], [5,5], [], [2,2],
[0.4], [1024,1],
tf.losses.mean_squared_error,
tf.train.AdamOptimizer(1e-4),
'CPCPDHH',
path/to/storage_folder, True)
```