SOP
DCE-NET – *estimating DCE parameters with a physics informed neural network*

P. Schouten & O.J. Gurney-Champion
Department of Radiology & Nuclear Medicine, Cancer Center Amsterdam, Amsterdam UMC, University of Amsterdam, Amsterdam, The Netherlands

General remarks

- Our code is written in Python, thus an installation of Python is required (https://www.python.org/downloads/).
- Anaconda is recommended, so packages dependencies are easily installed (https://www.anaconda.com/products/individual).
- General information about anaconda environments: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html
- The data set can be downloaded from the OSIPI website (https://osf.io/u7a6f/).
- It is highly recommended to run our program on a system that allows parallelization on multiple CPU cores and has CUDA enabled on its graphical card. Failure to do so will slow down the training and interference process of the neural network severely (https://developer.nvidia.com/cuda-downloads).
- Information about the population based AIF can be found here: https://jnm.snmjournals.org/content/57/12/1843.

Disclaimer

- Because the data set is large and we use a neural network approach, we load the data set as a whole in the computer memory, thus this Python script is pretty taxing on the memory usage (+64gb). If needed we can provide a version of the code which loads in the data per patient if the independent team runs in to memory issues. Please e-mail this request to p.schouten@amsterdamumc.nl and/or o.j.gurney-champion@amsterdamumc.nl.
    - To allow quick evaluation on low resource systems, an option is included to use our per-trained neural network and pre-calculated concentration maps (see: *III.b in the guide*).
- To avoid any stochastic variations in the training of the neural networks and hence improve reproducibility, we have set a seed point for the random number modules of Pytorch and Numpy. Behavior with seed points can be version dependent, we thus advice Pytorch version 1.10.0 and Numpy version 1.19.2.
- For the ktrans maps Nifti files, an Identity affine transformation matrix is used.

Guide

I. data set & dependencies

1: Create a folder and unpack the OSIPI data sets with a folder structure similar to the names on the OSIPI website.

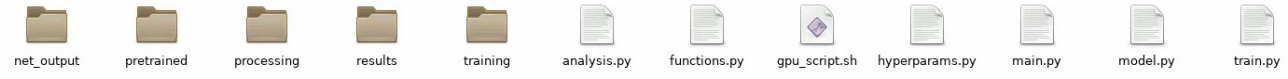
1.1: In the main data folder create two folders named: Clinical_Data & Synthetic_Data

1.2: Unpack for example ‘Clinical_P1’ in a folder called ‘Clinical_P1’ and so on. Thus the folder structure should be ‘Clinical_Px’ and ‘Synthetic_Px’ with x the data set number. Move all the ‘Clinical’ folders into the ‘Clinical_Data’ folder and so forth for the ‘Synthetic’ folders.

1.3: This should result in 8 clinical data sets each with 2 sub folders ‘Visit1’ & ‘Visit2’ and 2 synthetic data sets each with 2 sub folders ‘Visit1’ & ‘Visit2’, sorted by patient number.

Note: All Visit 1 & 2 folders should contain folders with scan data.

1: Unpack the zip file ‘DCE_NET.zip’ in a folder, the result should look something like this:



2.a: The dependencies are incorporated in the conda environment ‘dce_env.yml’ and can be found in the ‘DCE_NET’ main folder. Installation via this yml is tested on a Linux Red Hat system, but should work on Windows as well. After the installation of the Anaconda distribution:

- install the conda environment via:
  - ‘conda env create -f dce_env.yml’
- browse to the main folder of the environment and activate in the console via:
  - windows/linux: ‘conda activate dce’

2.b: The dependencies are also listed in ‘dependencies.txt’ and can be installed with ‘pip install [package name]’ as an alternative to the anaconda environment.

II. running the program and enabling the conda environment

1: To run the program open a console and follow the next steps:

- enter: ‘conda activate dce’ (or make sure python with the required packages is enabled if not running Anaconda)
- or enter: ‘conda activate <path to dce env>’
- browse to the DCE_NET folder where the python files are located.

III.a training the network and make predictions *(default)*

1: this option converts the signal maps to the concentrations maps, trains the neural network and makes predictions for ktrans.

- enter: ‘python main.py’ or ‘python3 main.py’ depending on your Python version.
  - note that two flags can be provided as input on the command line.
    - ‘--njobs x’ where x is an integer that gives the number of cpu cores to parallelize the computation on. The default setting is 10.
    - ‘--cpu’ to disable CUDA and graphic card computing. The default is CUDA enabled and recommended for speedy calculations.
    - An example for 8 cores and no CUDA is: ‘python main.py --njobs 8 --cpu’
- Alternatively open main.py in a Python IDE, like Spyder, and press run to run the program under default settings.

III.b make predictions on pretrained network and concentration data *(low system resources)*

1. This option uses pre-calculated concentration maps and our pre-trained neural network, which can be found in the ‘pretrained’ folder in the ‘DCE_NET’ main folder.

- enter: ‘python main.py --pretrained’ or ‘python3 main.py --pretrained’ depending on your Python version. This will automatically run on the CPU.
  - this mode enables the interference on the concentration maps using our pre-trained neural network if your system resources are too low to run the full Python script *(contact us if questions remain about the procedure)*.

IV. enter the OSIPI data directory

1: A prompt will appear which asks you to select the main folder of the OSIPI data set. This should be the top folder containing all the ‘Clinical_Px’ and ‘Synthetic_Px’ sub folders. Click the directory and proceed. If the prompt fails, a command line input appears asking for the directory, enter it and press [enter]. Make sure the slashes / or \ are correct in the file pathway.

V. ktrans maps location

1: The resulting ktrans maps can be found in the ‘results’ map of the DCE_NET main folder after the program has finished.

Feel free to contact us at pschouten@amsterdamumc.nl if anything is unclear.