Interface Layout for BRATS Docker Images

This document should serve as a guideline for the common interface for BRATS Docker containers. You should provide a container that can be used as described in this document and provides a standard interface. The aim is to have a uniform standard for containers such that running them in an automated manner and using input files in the BRATS-challenge format is as easy as possible. The user should only have to provide a script or a call to run the container with an input folder containing the raw files and receive the results in an output folder.

0.1 Interface definition

Your Docker image will be called with the following command:

```
docker run -v <user_directory>:/data -it <your_image>
 <your_script_call>
```

We have a master script that calls all available containers and passes one patient set of volumes per call. Your container must accept all data through the mounted folder in /data. This folder contains a folder for your results and the input files (see below). Your script (called via your_script_call) should take those files, run the segmentation algorithm on them and return the output to the results folder. All input files are in compressed or uncompressed Nifti format (.nii), both will be provided. File structure:

- /data (contains:)
  - /results (folder: your results should be placed here)
  - flair.nii
  - t1.nii
  - t1c.nii
  - t2.nii

Note: We will use fla.nii as file name for the FLAIR modality, if you have test sets with flair.nii, please rename them accordingly.
Output format: Same volume as the input volume; the space of one of the MNI atlases. Please use the labels of the BRATS paper (0 is non tumorous, 1 is NCR and NET, 2 is edema, 4 is enhancing). More info on labels and annotation can be found here. The output file name should be "tumor_<your_image>_class.nii.gz".
You should provide us with a Docker image running your code and a command to call your script inside the container which performs all the tasks mentioned above.
Please also note that Docker currently does not support GPU-accelerated code on all platforms (as of Sept. 2017); please provide a CPU-only version (performance reduction is not of concern for our comparison).
Questions to : c.berger[at]tum.de