Documentation of how to train and use your deep learning model

In this document it is explained how you can use a pretrained model to predict binary images for your MRI images, or how to train your own model. If you want to use a pretrained model, go to section **V** and follow the steps that are described. If you want to train your own model, go through sections **I** – **IV**. The operating system nnUNet uses is Linux (Ubuntu 16, 18 and 20; centOS, RHEL). Other operating systems are not supported. If you want to train a model, your GPU should have at least 10 GB and it is recommended that you have at least 6 CPU cores (12 threads). If you want to use a pretrained model to predict binary masks, the GPU should have 4 GB of VRAM. More information can be found here: <https://github.com/MIC-DKFZ/nnUNet>.

1. **Preparations of your initial files to be able to train a model with nnU-Net**

*Initial-situation:*

Files with extensions ‘.nii’, ‘.nii.gz’, ‘.img’ and ‘.hdr’ are acquired from MRI-scans and contain 3D MRI-images. The corresponding labels are made in MIPAV which are saved as .xml files.

*Final-situation:*

For every model that you want to train, you need a folder with training images, the corresponding binary masks and test images named: **imagesTr,** **labelsTr**, **ImagesTs**. All files must have an **‘.nii.gz’ extension**. The model will train with the images in the imagesTr folder, but also needs validation images which is defined in the **splits\_final.pkl** file. Note: validation of the model does not influence training. Therefore, for validation the images from ImagesTs are used. In this way you’ll be able to train your model with the highest possible amount of images. A **dataset.json** file is needed which contains information about the files in imagesTr, labelsTr and imagesTs. The binary masks of the images in imagesTs can later be calculated with the trained model.

You can go from *initial-situation* to *final-situation* in any operating system.

1. Install Python 3.7 (you can use Spyder as environment)
2. Download the files on the github page: main.py, XML\_to\_Nifti.py, random\_split\_patients.py and generateJSON.py.
3. In main.py, modify *your\_path* to the location where you want to put the files (Note it should start with r’.   
   Example: your\_path = r'D:\my\_folder’
4. In main.py, modify *task\_name* to TaskXXX\_MyTask.   
   Example: task\_name = ‘Task002\_Stomach’
5. Run main.py
6. Fill the created raw\_data folder with your patient data as follows (see figure 1). In the raw\_data folder, create folders for the different studies (if different studies are present). In each study folder, create two folders called ‘MRI’ and ‘Mask’. In both folders, create a folder for each patient. In ‘MRI’, these patient folders should be filled with the MRI data (with an .img, .hdr, .nii or .nii.gz extension). In the ‘Mask’ folder, the patient folders should be filled with the different time points (if multiple time points are present). Provide each time point with the corresponding binary mask(s) (with .xml extension).
7. If you want to train a 2D model, set ‘train2D = True’. Otherwise the files will be set up for a 3D model.
8. If you want to check if the binary masks correctly fit the MRI images, set ‘check\_masks = True’.
9. Now run main.py again.
10. **Setting up nnU-Net**

nnU-Net Is an open source segmentation model, and a detailed description of installation can be found there: <https://github.com/MIC-DKFZ/nnUNet#installation>. nnU-Net Is developed to train its models 1000 epochs. Because this requires a lot of computational memory, errors can occur during training. You can prevent this by training your model on less epochs. Therefore a new Trainer has to be created named **nnUNetTrainerV2E150**.

1. Install Pytorch (<https://pytorch.org/get-started/locally/>)
2. In your Command Prompt, go to the directory where you want nnUNet to be installed. You can use the *cd* command to navigate through your folders and set the root where nnUNet will be installed (for an example see step 1 in section **III**).
3. Copy these lines to your command prompt and hit enter:

git clone https://github.com/MIC-DKFZ/nnUNet.git

cd nnUNet

pip install -e .

Now you have a new folder ‘nnUNet’ with in this folder all scripts nnUNet uses.

1. Create the following new folders in this nnUNet folder:

* nnUNet/**data**
* nnUNet/data/**RESULTS\_FOLDER**
* nnUNet/data/**inference**
* nnUNet/data/**nnUNet\_preprocessed**
* nnUNet/**nnUNet\_raw\_data\_base**
* nnUNet/nnUNet\_raw\_data\_base/**nnUNet\_raw\_data**
* nnUNet/nnUNet\_raw\_data\_base/nnUNet\_raw\_data/***task\_name****Note: task\_name here is the name of your task: TaskXXX\_MyTask*

1. Insert all content of the ‘Input voor nnUNet’ folder in the new task\_name folder. An example is given in figure 2.
2. Set your environment variables by searching for your .bashrc file, and edit three new lines in this file. You can open your .bashrc file as a text file, and type the lines at the end of this file:

export nnUNet\_raw\_data\_base="/**your\_path**/nnUNet/nnUNet\_raw\_data\_base"

export nnUNet\_preprocessed="/**your\_path**/nnUNet/nnUNet\_preprocessed"

export RESULTS\_FOLDER="/**your\_path**/nnUNet/data"

Save your .bashrc file with the three new lines.

Setting up your environment variables is also as described here: <https://github.com/MIC-DKFZ/nnUNet/blob/master/documentation/setting_up_paths.md>.

1. Open *nnUNetTrainerV2.py* located in \nnUNet\nnunet\training\network\_training in Python. Change the name of the Trainer class (nnUNetTrainerV2) to **nnUNetTrainerV2E150**:

class nnUNetTrainerV2 (nnUNetTrainer):

to

class nnUNetTrainerV2E150 (nnUNetTrainer):

Now change the number of epochs to 150:

self.max\_num\_epochs = 1000

to

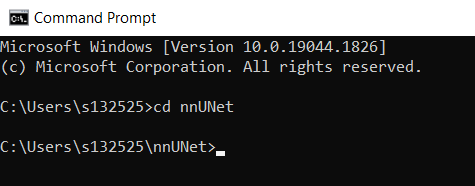
self.max\_num\_epochs = 150

Save this as a new python scipt in the same folder (\nnUNet\nnunet\training\network\_training ) named: *nnUNetTrainerE150.py*.

1. **Train your model with nnU-Net**

When executing the steps in section **I** and **II**, you can now train your model.

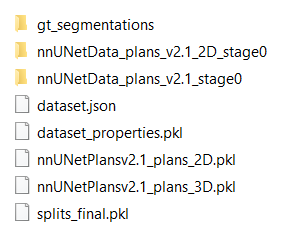
1. In your Command Prompt, go to the directory of where nnUNet was installed. You can use the *cd* command to navigate through your folders.



1. Now type in the command prompt:

nnUNet\_plan\_and\_preprocess -t 002 --verify\_dataset\_integrity

If you want to use a certain GPU on your computer (instead of all GPU’s available), type CUDA\_VISIBLE\_DEVICES=1 in front of this line.

1. Now put the **splits\_final.pkl** file (created in section **I**) in the nnUNet/data/nnUNet\_preprocessed/*task\_name* folder. The *task\_name* folder should now contain the following folders:
2. If you want to train a 2D U-Net, type in the command prompt:

CUDA\_VISIBLE\_DEVICES=1 nnUNet\_train 2d nnUNetTrainerV2E150 Task002\_Stomach 0

With ‘Task002\_Stomach’ being your own task\_name. If you want to train a 3D U-Net, replace ‘2d’ with ‘3d\_fullres’. Note that the input should be 2D when training a 2D model, and 3D when training a 3D model. The zero on the end of the line indicates the foldnumber, and should be zero if you only have 1 fold. Your trained model is now located in nnUNet/data/RESULTS\_FOLDER/nnUNet/2d or 3D\_fullres/task\_name.

1. **Predict binary masks with your trained model**

You can now use your trained model to predict binary masks from the MRI images in the imagesTs folder.

1. Create a new folder in nnUNet/data/inference/ . In this folder the binary masks predicted by the model will be saved. Example:

nnUNet/data/inference/**Model002**

1. In your command prompt, go to the folder of nnUNet (see step 1 of section **III**).
2. Now type in the command prompt:

CUDA\_VISIBLE\_DEVICES=1 nnUNet\_predict -i /your\_path/nnUNet/nnUNet\_raw\_data\_base/nnUNet\_raw\_data/Task002\_Stomach/imagesTs -o /your\_path/nnUNet/data/inference/Task002\_Stomach -t 002 -m 2d -tr nnUNetTrainerV2E150

Where behind -i the path is given to the input images folder, behind -o the path is given to the output folder, behind -t the tasknumber is given (Task**002**), behind

-m the kind of model (2d or 3d\_fullres) and behind -tr the name of the Trainer. You can find the name of the Trainer in: nnUNet/data/RESULTS\_FOLDER/2d or 3d\_fullres/*task\_name*/ . Example: **nnUNetTrainerV2E150\_\_nnUNetPlansv2.1** (the name of the Trainer is in front of ‘\_\_’).

Reminder: typing CUDA\_VISIBLE\_DEVIES=1 in front of the line is optional, it makes sure you only use one of the GPU’s of your computer, instead of all.

1. **Predict binary masks with a pretrained model**

To be able to use a pretrained model to predict the binary masks of your MRI images, you first need to follow the steps in section **II**. Your *task\_name* folder is named **Task707\_Stomach** or **Task709\_Stomach**, dependent on which model you want to use. Task707\_Stomach is a 2D U-net, so requires 2D images as input images. Note, this model has only been trained with images that contain stomach content, so it does not predict well for images without stomach content. Task709\_Stomach is a 3D U-net, so requires 3D images as input images. This model performs well on images with and without stomach contents overall.

1. Follow the steps in section **II** (except step 5 and 7).
2. Put all your MRI images in the **ImagesTs** folder, located in:

* nnUNet/nnUNet\_raw\_data\_base/nnUNet\_raw\_data/*task\_name*/**ImagesTs**

Note: your MRI images should have an .nii.gz extension!

1. Dependent on the model you chose, put the files from the google drive in the corresponding directory on your computer (so keep the same structure of directories). If a directory doesn’t exist yet, create this directory on your computer.
2. Follow the steps in section **IV**.

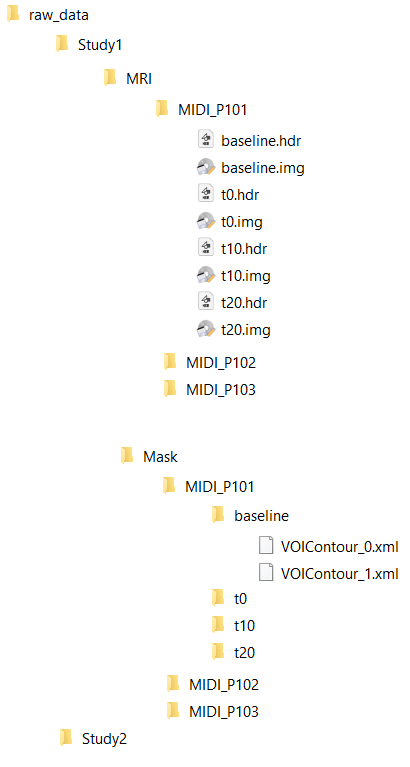
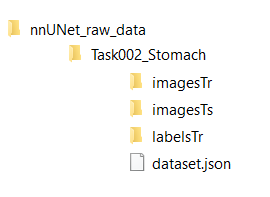


Figure Visualization of how the raw\_data folder is constructed

Figure 2 Visualization of the content of the task\_name folder