**How to use AxonDeepSeg (ADS)**

Hi, greetings from Baldvin! Here is a very rough guide to approaching ADS, this is written as a supplement to the guide that Yiming has already written, so for all the basic stuff, look at her pdf file.

**When using ADS for the first time**

If you are new to all this here is how I would probably go about starting if I were to start again:

First of all, open Anaconda. When in anaconda you should get a line like this

(base) PS C:\Users\(your user account)>

I would advise you to use the ADS website to help set everything up as it is very detailed and quite well set up <https://axondeepseg.readthedocs.io/en/latest/index.html>. Most of the following is on the website:

First you need to download AxonDeepSeg, this you do by writing the following in the promt:

git clone https://github.com/neuropoly/axondeepseg.git

cd axondeepseg

After you have finished downloading ADS, you need to create a ‘virtual environment‘ (this is where all the files you use to run your programs will be stored). This you do by typing:

conda env create

This migh take a while, but after it is finished, you should have a file in your used folder (in my case in C:\Users\bfg21) named **.conda**, in this folder should be another folder named **envs** (where your virtual environments are stored) and in there should be your virtual environment, named **ads\_venv**.

After you have created your virtual environment you need to activate it (tell the prompt you want to use it now), This you do by typing in:

conda activate ads\_venv

Now you need to install the ADS software inside your virtual environment, you do this with:

pip install -e . plugins/

Now ADS should be ready to use, almost...

There is a bug/something in the code where when you start running models and testing ADS, you might get an error that says something about trying to create/duplicate a file called libiomp5md.dll [OMP: Error#15: Initializing libiomp5md.dll, but found libiomp5md.dll already initialized] (or something like that). There is a (sort of) simple fix!

You need to go into the following file [.conda/envs/ads\_venv/Library/bin] and find the file named libiomp5md.dll, and move it (don‘t delete it), keep it somewhere where you can find it, because you might have to move it back to the same place later. (Make sure you have closed Anaconda when you do this, cause otherwise it might say you can‘t move it because it is open in another program...)

Now you should be able to test if ADS is working and installed, and you do that with

axondeepseg\_test

Does it work? Excellent! You‘re already well on your way to doing some computer magic.

Does it not work? Well, that‘s frustrating... It might be a problem I have already encountered, in which case the solution should be in here somewhere...

It could also be that I have gotten the timing of the moving of the libiomp5md.dll file wrong. If it doesn‘t do anything, try putting the libiomp5md.dll file back and see if it‘s working now.

**What you need to do every time you open Anaconda again to use ADS (after your first time)**

There are two things you always need to do when you open Anaconda, the first one is opening your virtual environment, you do that with:

conda activate ads\_venv

And the second is activating ADS, and you do that with

cd axondeepseg

**Downloading Cuda (so that you can use the GPU on the computer)**

In order to run faster calculations you need to tell the computer to use the GPU instead of the CPU. This will make the training go many many times faster than it would were you using the CPU. In order to do this you need to install Cuda. On the Ivadomed website under ‚Train model‘ (We are using ‚Two-class microscopy segmentation with 2d U-Net, it‘s under Tutorials on the ivadomed website) you will see a blue note where it says „If a compatible GPU is available, it will be used by default. Otherwise, training will use the CPU, which will take a prohibitively long computational time (several hours)“ In there is a link to the website where you can download Cuda (if you cant find it, here it is: <https://pytorch.org/get-started/locally/>). On there, under ‚start locally‘ you should find a boxy thing where you can select an operating system, language, packages and so on. These are the settings I selected: PyTorch Build – Stable (2.2.2); Your OS – Windows; Package – Pip; Language – Python; Compute Platform – CUDA (I had to use version 11.8, 12.1 doesn‘t work with the NVIDIA driver on the analysis computer). Below you will get a command which you copy and paste into your promt (Anaconda). This should download Cuda for you. After you finish this the computer should run all commands on the GPU ☺.

Example code for downloading Cuda:

Pip3 install torch torchvision torchaudio –index-url https://download.pytorch.org/whl/cu118

**Creating your own dataset**

Now... You need to create your dataset so you can train your own models? In my data folders you should hopefully be able to get a semi-clear view of how I approached this and what you need to do. I tried my best to organise them as well as I could ☺.

Here is what you do, if you do it the same way I did it. Let‘s start from the beginning. You have an overview EM image you want to use. The first thing you do is open it in FIJI. First I decided how big I wanted the images I will use to train ADS to be. In my case they were 4099x4099 pixels (the pixel size of the images were 0.0018625um/pixel). I then created a square of that size and saved it as a ROI. So lets say I want to have four 4099x4099 images from the overview image I would create and save four 4099x4099 ROIs. I then traced manually the axons and the myelin around them for every axon and myelinsheath inside those squares (I always started with the myelin and then did the axon inside (I would recommend doing that, both because the myelin is tougher to draw, and because the FIJI macro we used assumes that the first of every pair of ROIs is around the myelin and the second around the axon)). This process is one of the most time-consuming of all of them.

Let‘s look at my ‚Original protocol‘ folder. In that folder I have one folder corresponding to each overview image, as well as a folder containing the FIJI macros (while I remember, the one I used is ‘labelling\_ADS\_test2.ijm‘). Now, lets look at ‚Image 2336‘. In there we have a few files: First of all we have the overview image itself as well as the settings for the image (that among others include the pixel size). Then we have the ROIs, I had a ROI set containing all of the tracings, and then the ROIs for the squares separate, you will see why in just a bit. Then I had two folders with morphometrics, one for morphometrics gotten from the manual tracings, and one from the images segmented by ADS (this you get in the very end after training your model). Then I have two folders with cropped images. The first is ‚Cropped images for Napari‘ and the second is ‚Cropped images for ADS‘.

**Step 1: Tracing in FIJI**

Now, back to the time-consuming task of manually tracing all the axons and myelin sheaths‘. When you have traced everything it is time to save them all. Now, the order here isn‘t crucial but I would recommend starting in this order and then when you get a better grasp of how everything works you can change this and implement your own system. The first thing I did was open the ROIs (Without the ‚image squares‘, so just the ROIs containing the axons and myelin tracings (in this case ‚RoiSet\_2336\_more‘)). Now, you want to open the FIJI macro Sebastian and I wrote called ‚labelling\_ADS\_test2.ijm‘ and run it. When you do, you will get a promt asking you where you want the image to go, in this case I wanted it to go in the ‚Image 2336‘ file. When you run it, you should see the image become black and then fill with the axons and myelin sheaths in grey and white, this is your thresholded image, and it is important for when you want to import your image into Napari. Now, you have two images, your original overview image, and the thresholded black, grey and white overview image. Now it is time to crop your images to the size you want for ADS. First, open your original overview image in FIJI and open the ROI with the square representing your image. Then press shift+D and it should open a cropped image. Save it into your ‚Cropped images for Napari‘ file (try to save it as a .png file), if you have more squares, do the same to those (remember to rename them to what you want so you know which one is which). Now open your thresholded overview image and do the same. Now in your ‚Cropped images for Napari file you should have two images for each square you wanted.

**Step 2: Tracing in Napari**

Now it‘s time to open Napari, you do that in the Anaconda promt by simply typing in:

napari

When you have opened napari go to ‚plugins‘ and add ‚ADS plugin (napari-ads)‘.

First thing you do is you import your cropped image (original, not the thresholded one). Then once it‘s in napari you press ‚load mask‘ and you select the corresponding thresholded image‘. This should give you an image with your tracings overlaid on top. Here I did some fine-tuning as well as separating all of the myelin-sheaths from each other (I think they all need to be separated), this can also be very time-consuming. When you have finished doing that you can press ‚save segmentation‘. This is what I put into the ‚Cropped images for ADS‘ folder. I would also recommend you do the ‚computer morphometrics‘ now, as you might need them later. These I saved as ‚morphometrics – manual´.

When you have done this to all of your images you should be ready to start setting all of your data/images up in a BIDS format! (this was extremely frustrating for me to figure out, so I will try to be as detailed as I can!)

**Setting your files up in a BIDS format**

I created a folder called ‚data\_axondeepseg\_tem‘ that included my training dataset, and a folder named ‚log\_microscopy\_tem‘ which included all of my models (can be found in ‚Original\_protocol\_ADS‘ and ‚New\_protocol\_ADS‘). Let‘s look at the data\_axondeepseg\_tem folder, this is the most crucial part of getting ready to use ADS, the files contained in this folder must be arranged in a very specific way and include very specific things.

When you first open the data\_axondeepseg\_tem folder you see a folder named ‚derivatives‘, some folders named ‚sub-sample1‘ trough 10 and then 5 other files.

The first thing you need to do is copy the .gitignore file into your corresponding folder. The ‚dataset\_description.json‘ file you can also copy in there, but remember to change the contents of the file to what you want it to be (i.e. change names and so on...). I think you can remove the README file, but I kept it in there just in case. Now, you will have two ‚samples‘ files, one TSV file and one JSON file. Let‘s start with the samples.json file. Copy it into your corresponding folder, it is important to have it in there as it says what you call your sample and participant IDs.

The next step is to open the samples.tsv file. In there you will see three columns; sample\_id, participant\_id and sample\_type. The things you write in here are very important when it comes to naming the image and text files later. When you write in the columns you start by writing in the name of your sample. I had a system where I first wrote ‚sample-‚ and then gave each sample/image an identifier, so for example for the new protocol the first image was ‚sample-411‘ where 4 meant FIX4, 1 meant the first overview image and the second 1 meant the number of the cropped image from that first overview image. The next column is the participant\_id column. You can play around with what you need to write there, but I found that naming them ‚sub-sampleX‘ (where X is the number of the sample) seemed to work. The sample\_type is just ‚tissue‘ in my case.

Now that you have named your sample\_id and participant\_id, it is time to start creating the folders and naming your images. For every image you have you have to have the three thresholded images that Napari gives you (each with their own endings, \_seg-axon, \_seg-axonmyelin and \_seg-myelin). All of your ‚raw‘ (not-thresholded) images will have their own separate folder in your version of the ‚data\_axondeepseg\_tem‘, and the thresholded images will all be in another folder called ‚derivatives‘. Let‘s start with your non-thresholded images. Since I labelled my images in the participant\_id as ‚sub-sampleX‘ I will name my folders the same, with a different folder for every image. Inside every one of those folders you will have to create another folder called ‚micr‘ (don‘t ask me why, I don‘t know...) (so data\_axondeepseg\_tem/sub-sample1/micr/); it‘s inside this micr folder you will put your image, as well as a .json file with information about your image (this is very important, mostly because it defines the pixelsize of the image you‘re using). This will look something like this:

{

„PixelSize“: [0.0018625, 0.0018625],

„PixelSizeUnits“: „um“,

„BodyPart“: „ON“,

„SampleFixation“: 4% PFA, 2% Glutald in 0.15 M Cacodyolate buffer + 1.5% mannitol“,

„SampleEnvironment“: „ex vivo“

}

You can also copy this from my file, just remember to edit it‘s contents to match your images. You should also import your image as a .png into this folder. Okay, now in your first sub-sampleX folder you should have an image (as a .png) and a .json file; now it‘s time to name them (if you look into my folders you will sometimes find a file with the .nii ending, don‘t worry about them, they are not necessary and are made automatically when you run ADS). The beginning of the name should be the name and/or number of the image and after that you should have the corresponding ‚sample\_id‘ (I always follow that with a \_TEM, but I don‘t know if that‘s necessary, you can maybe play around with that and see what works and what doesn‘t). So an example of a name would be ‚sub-sample1\_sample-411\_TEM‘ (important, both the image and the .json file should have the same name). For all your other images you can do the same, so the next image goes in another folder (so data\_axondeepseg\_tem/sub sub-sample2/micr/) and in that micr folder the image and .json file will be called ‚sub-sample2\_sample-412\_TEM‘, and so on and so forth.

Now it‘s time to import your thresholded images. For this you need to begin by making a folder in data\_axondeepseg\_tem called ‚derivatives‘ and another one in there called ‚labels‘ (so data\_axondeepseg\_tem/derivatives/labels/). In there you need a .json file called ‚dataset\_description‘ (you can copy this from my files), but if you can‘t these are it‘s contents:

{

„Name“: „data\_axondeepseg\_tem labels“,

„BIDSVersion“: „1.7.0“,

„PipelineDescription“: {„Name“: „Axon and myelin manual segmentation labels“},

„GeneratedBy“: [{„Name“: „Axon and myelin manual segmentation labels“}]

}

Now, in this ‚labels‘ folder you will again also acreate folders for every image (so sub-sample1, sub-sample2...) and everyone of them should contain a folder named ‚micr‘. In this ‚micr‘ folder you will put your 3 thresholded images. The name of these images should follow the same system as the original image but with the addition of the correct \_seg-axon, \_seg-axonmyelin or \_seg-myelin. So an example of three thresholded images would be: **s**ub-sample1\_sample-411\_TEM\_seg-axon, sub-sample1\_sample-411\_TEM\_seg-axonmyelin and sub-sample1\_sample-411\_TEM\_seg-myelin.

Now, you should have everything sorted, and if you have done everything correctly (and more importantly that I have written everything correctly), you should have a BIDS system of folders and files that ADS can use. You can check this online with a BIDS validator where you upload your data folder and it tells you if you‘ve done everything correctly:

<https://bids-standard.github.io/bids-validator/>

Figure 1: A diagram showing the BIDS file arrangement as I did it

**Segmenting images in Napari and calculating morphometrics**

I know I have mentioned this above, but it doesn‘t hurt to give this it‘s own special chapter. Once you have finished tracing and editing your images in Napari, it‘s time to save them. This you do by pressing ‚segment‘ (or something like that) and save them wherever you want (I did it in my ‚cropped images for ADS‘ folder). Let‘s say you forgot to compute the morphometrics, here‘s how you can from an image and it‘s thresholded mask. First you move your image into Napari. Then you press ‚load mask‘ and import the mask you want, it can be any mask you want that corresponds to your image (so any of the segmented images from your trained models, or your manual one). Then you press ‚compute morphometrics‘ (the pixel size you give is the pixel size of the original image in um (so in my case 0.0018625).

**Problems I‘ve faced and how I‘ve (sort of) fixed them**

After I use the promt/ADS to segment images, some things seem to stop working, for example the commands ‚axondeepseg\_test‘, ‚napari‘ and the axondeepseg\_morphometrics function. It runs the line and then just gives me a new line as if nothing happened. I think this might be a bug, and that when you segment images some files are changed. So far the only workaround I‘ve found is to delete the virtual environment (I would recommend saving the config files first (.conda/envs/ads\_venv/Lib/site-packages/ivadomed/config)) and reinstall everything. So first you delete the ads\_venv folder, then you open Anaconda, do ‚cd axondeepseg‘ and then conda env create and then just follow the instructions as if you are installing it for the first time. This seems to work, I can then use Napari again. If this is still a problem when you are using this, either Sebastian has already contacted the ADS people and found out what‘s wrong, or you might have to ask him to, it is a little bit annoying. (Also, don‘t forget to delete the libiomp5md.dll file (see somewhere above)).

**Important!**

The text above might be wrong in some places or not apply to you, you will just have to try yourself. I found out that maybe this bug has something to do with the libiomp5md.dll file being missing. So before you try to uninstall and reinstall the virtual environment, try either putting the libiomp5md.dll file back or taking it out (depending on if it is in already or not), it might fix your problem, it did for me once, and I haven‘t had the chance to test it again, so you just have to try yourself.