icord

Automated Segmentation of Medical Images Using a Convolutional Neural Network

User Manual

Vancouver CoastalHealth Research Institute

Document Purpose

To describe how to run the platform developed to automatically segment regions of interest (ROIs) from medical imaging data.

Table of Contents

Data Management

Training Data

Data Storage

In order to allow the code to run successfully, it is recommended to organize the training data using the following structure:

- project directory
    - data
        - train
            - dicoms
                - contains all the raw images available in the training dataset (each scan axial slice represents a sample)
            - labels
                - contains all the segmentation files for the corresponding images available in the training dataset

Data Format

Here are the supported data formats:

- **images:** DICOM images (.dcm)
- **labels:** numpy arrays (.npy), JPEG images (.jpg), PNG images (.png)

Data Labeling

The images and corresponding segmentation files should have the same filenames. Otherwise, the code will not be able to run successfully, as it will be looking for matching pairs of images and labels based on filenames during training.

For example, if a DICOM image is named 'slice_1.dcm', the corresponding segmentation file should be named 'slice_1.npy' or 'slice_1.jpg' or 'slice_1.png'.

Data Segmentation

Segmentation files can be generated using the software of choice (3D Slicer, ImageJ, Photoshop, etc.), as long as:

- The resulting files are in one of the supported formats (.npy, .jpg or .png).
- The filenames match with the corresponding segmented images.
- One segmentation file is generated per slice.

In addition, the pixels of the segmentation file should define what specific region of interest (ROI) each pixel belongs to. For example, if a total of 4 ROIs are being segmented, every pixel that belongs to ROI #1 should have a value of 1, every pixel that belongs to ROI #2 should have a value of 2, etc. and every pixel that is not labeled (i.e. belongs to the background or other regions) should have a value of 0.

WARNING: Make sure that the labels are consistent across samples! In other word, if the pixels of ROI #1 on image 1 have a value of 1, the pixels that belong to the same ROI on every other image in the training set should all have a value of 1.

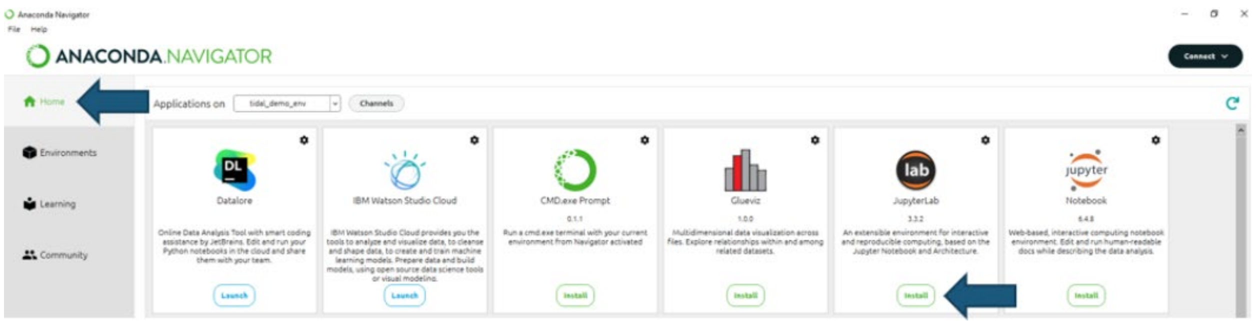
Testing Data

Combine all the raw axial DICOM images that you wish to segment in a single directory then select that directory when running the testing notebook.

Jupyter Lab

Session Launch

In the Anaconda Navigator, click on the '*Home*' tab in the top left corner, then look for the Jupyter Lab icon. If Jupyter Lab has not been installed yet, click on the '*Install*' button.



Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

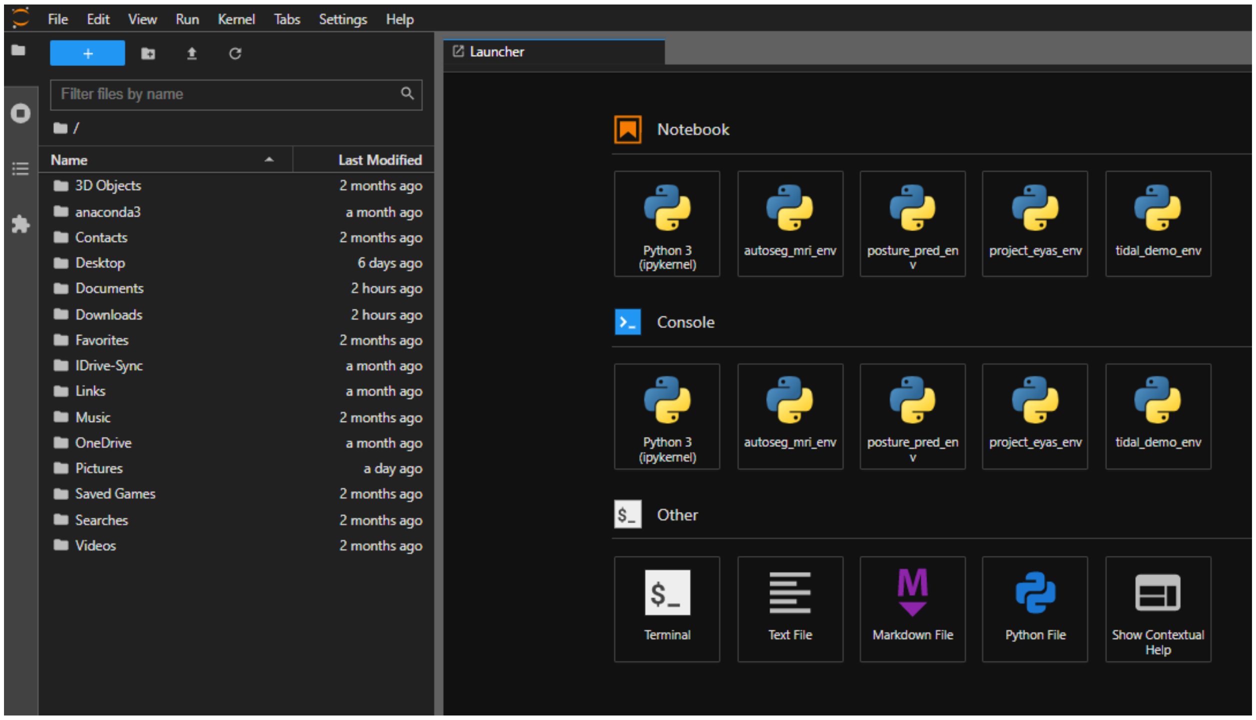
Run a cmd.exe terminal with your current environment from Navigator activated

Multidimensional data visualization across files. Explore relationships within and among related datasets.

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Once installed, click on the '*Launch*' button, which will automatically open a tab in your default internet browser and launch a Jupyter Lab session.



Name	Last Modified
3D Objects	2 months ago
anaconda3	a month ago
Contacts	2 months ago
Desktop	6 days ago
Documents	2 hours ago
Downloads	2 hours ago
Favorites	2 months ago
IDrive-Sync	a month ago
Links	a month ago
Music	2 months ago
OneDrive	a month ago
Pictures	a day ago
Saved Games	2 months ago
Searches	2 months ago
Videos	2 months ago

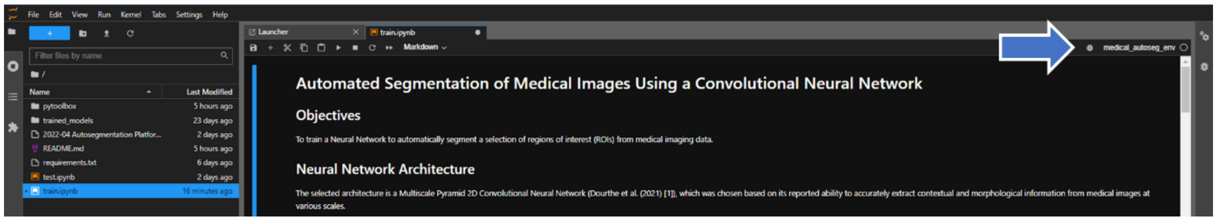
Notebook Launch

On the top left corner, you will see a tab called '*File Browser*' (indicated by a folder icon). By default, the File Browser will open at the location of your main User directory (i.e. '*C:\Users\username*'). Use the File Browser to navigate to the location of your local copy of the project GitHub repository (should be named '*medical-autosegmentation-cnn-main*').

Then, open the jupyter notebook named '*train.ipynb*' or '*test.ipynb*'.

Python Environment Activation

The environment within which the notebook is running is indicated in the top right corner of the screen.



Name	Last Modified
pytoolbox	5 hours ago
trained_models	23 days ago
2022-04 Autosegmentation Platfor...	2 days ago
README.md	5 hours ago
requirements.txt	6 days ago
test.ipynb	2 days ago
train.ipynb	16 minutes ago

Automated Segmentation of Medical Images Using a Convolutional Neural Network

Objectives

To train a Neural Network to automatically segment a selection of regions of interest (ROIs) from medical imaging data.

Neural Network Architecture

The selected architecture is a Multiscale Pyramid 2D Convolutional Neural Network (Dourthe et al. (2021) [1]), which was chosen based on its reported ability to accurately extract contextual and morphological information from medical images at various scales.

If the environment is different from the one you created for this project, click on the current environment name, then use the drop-down menu to select the right environment. If the right environment is not showing, go back to the section called '*Adding the Python Environment to Jupyter Lab*' to add the environment to Jupyter Lab.

Notebook Tips

An IPython Notebook is composed of a series of cells, which can contain text (Markdown) or code.

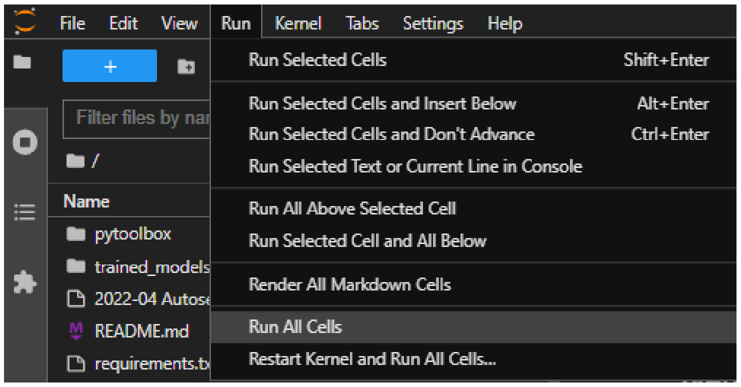
Collapse/Expand Cells

By default, all cells should appear expanded (i.e. showing what they contain). To facilitate navigation, any code cell can be collapsed by clicking on the blue line on the left of the selected cell.

Run a Cell

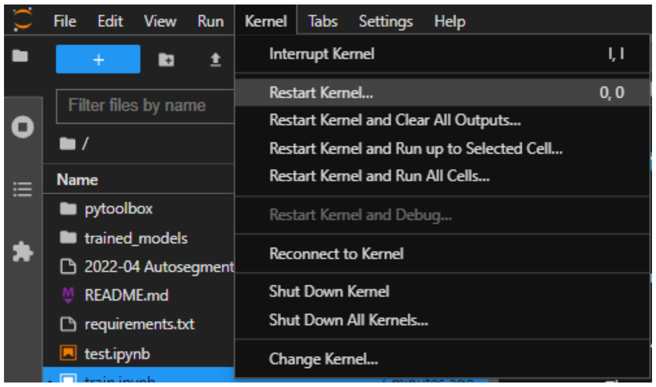
To run a single cell, select the corresponding cell by clicking on it, then press '*Shift + Enter*'.

You can also click on Run > Run All Cells in the top menu:


File
Edit
View
Run
Kernel
Tabs
Settings
Help
Run Selected Cells Shift+Enter
Run Selected Cells and Insert Below Alt+Enter
Run Selected Cells and Don't Advance Ctrl+Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
Filter files by nar
Name
pytoolbox
trained_models
2022-04 Autose
README.md
requirements.tx

Restarting a Kernel

If you wish to re-run the code, or if any issue happens that requires a reset, click on Kernel > Restart Kernel... in the top menu:



Training Notebook Walkthrough

The purpose of the training is to train a deep learning model to learn how to perform automated image segmentation. This code uses manually labeled/segmented data as input.

Imports

Go to the '*Imports*' section and run the corresponding cell (Shift + Enter) to import all the required packages.

Systems Characteristics

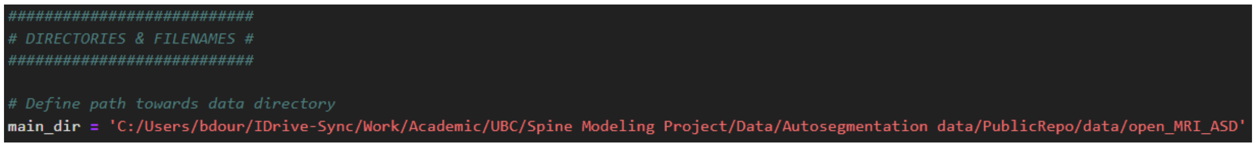
In this section, run the corresponding cell to check whether your GPU has been enabled. If your computer is equipped with an NVIDIA GPU and if all the requirements for GPU computing described in the installation manual were following correctly, you should see a similar print statement when running that cell:

```
RAM Memory:        21.9 GB (Available)

GPU enabled:       True

Device Name:       NVIDIA GeForce RTX 3080 Ti Laptop GPU
Memory Details:    16.4 GB (Total)         0.3 GB (2% Used)      15.9 GB (98% Free)
```

Settings

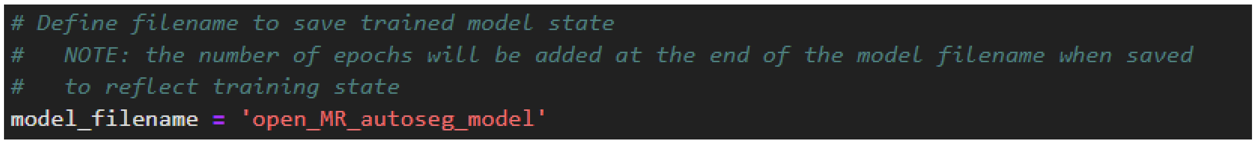
Go to the '*Settings*' section and edit the following variables:

- Under the DIRECTORIES & FILENAMES section:
  - **main_dir:** type the path to the training data.



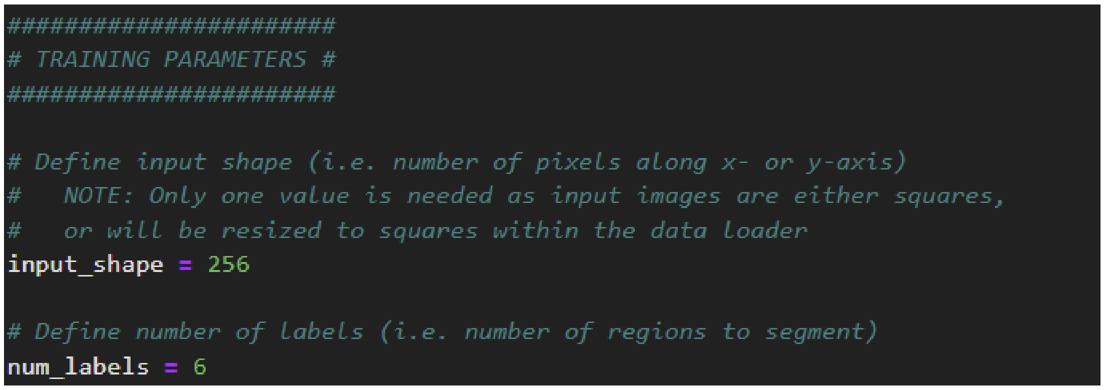
```
###########################
# DIRECTORIES & FILENAMES #
###########################

# Define path towards data directory
main_dir = 'C:/Users/bdour/IDrive-Sync/Work/Academic/UBC/Spine Modeling Project/Data/Autosegmentation data/PublicRepo/data/open_MRI_ASD'
```

  - **model_filename:** choose how the trained model will be saved as. Note that the number of epochs at which the corresponding model was saved will be automatically added to the filename (i.e. if the model is saved at epoch #100, the corresponding filename will be '*model_filename_100.pt*':



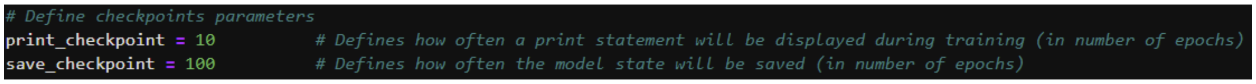
```
# Define filename to save trained model state
#   NOTE: the number of epochs will be added at the end of the model filename when saved
#   to reflect training state
model_filename = 'open_MR_autoseg_model'
```

- Under the TRAINING PARAMETERS section:
  - **input_shape:** specify the shape of the training images (e.g. if training images are 256x256, then the input shape should be 256).
  - **num_labels:** specify the number of regions of interest (ROIs) that were segmented in the selected dataset.



```
#######################
# TRAINING PARAMETERS #
#######################

# Define input shape (i.e. number of pixels along x- or y-axis)
#   NOTE: Only one value is needed as input images are either squares,
#   or will be resized to squares within the data loader
input_shape = 256

# Define number of labels (i.e. number of regions to segment)
num_labels = 6
```

- **print_checkpoint (Optional):** defines how often a print statement will be displayed during training (in number of epochs).
- **save_checkpoint (Optional):** defines how often the model state will be saved (in number of epochs).

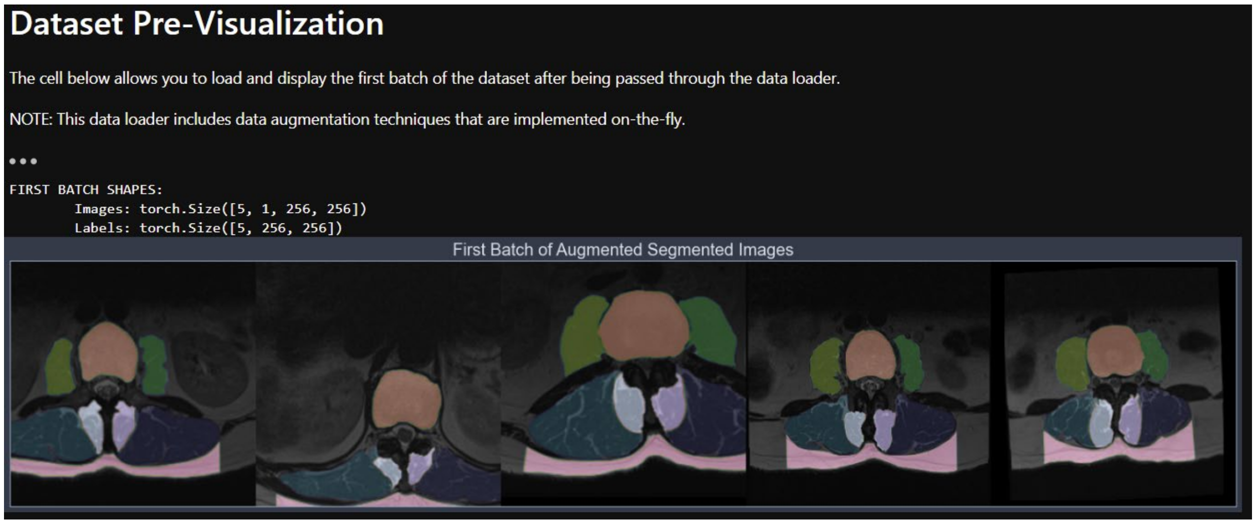


```
# Define checkpoints parameters
print_checkpoint = 10          # Defines how often a print statement will be displayed during training (in number of epochs)
save_checkpoint = 100          # Defines how often the model state will be saved (in number of epochs)
```

- Every other parameter can be left as their original value.

Data Pre-Visualization

Run the cell in this section to check that the data is imported and loaded correctly. This will also allow you to check that the ROIs match with each image (i.e. ROI of sample #1 is displayed on sample #1). Also note that the color of each ROI should be consistent from one sample to another (i.e. region of interest #1 in sample #1 should be the same color than region of interest #1 in sample #2, etc.). If the colors of the ROIs appear inconsistent, check the corresponding segmentation files, as this could lead to errors when training the model.



Dataset Pre-Visualization

The cell below allows you to load and display the first batch of the dataset after being passed through the data loader.

NOTE: This data loader includes data augmentation techniques that are implemented on-the-fly.

```
•••
FIRST BATCH SHAPES:
        Images: torch.Size([5, 1, 256, 256])
        Labels: torch.Size([5, 256, 256])
```

Model Training

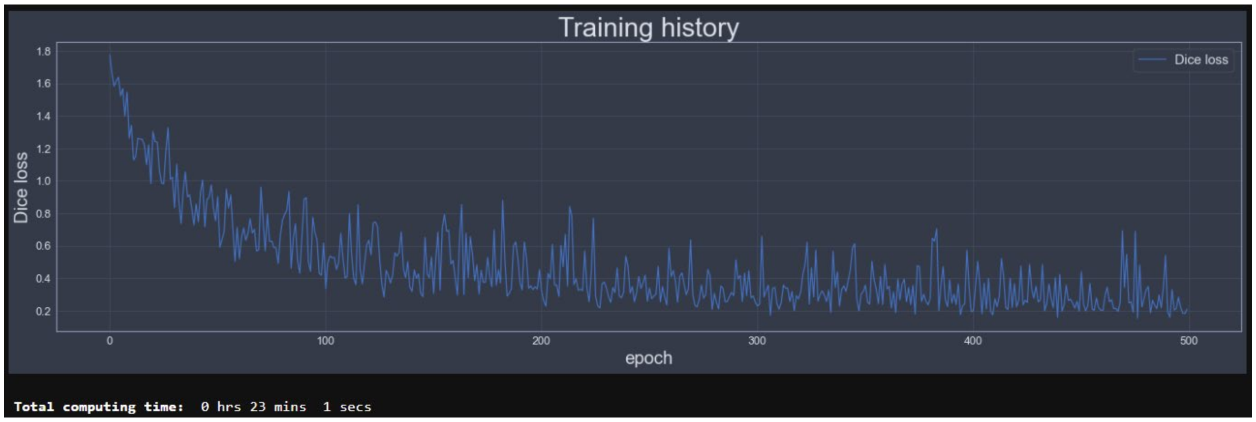
Run the corresponding cell to start the training process. Print statements will be displayed every 10 epochs (this can be changed by changing the '*print_checkpoint*' variable in the '*Settings*' section) to show the training status:

```
Completed epochs:   1/500 | Dice loss: 1.774 | Mean dice score: 0.113 | Time elapsed:  0 hrs  0 mins 10 secs
Completed epochs:  10/500 | Dice loss: 1.262 | Mean dice score: 0.369 | Time elapsed:  0 hrs  0 mins 35 secs
Completed epochs:  20/500 | Dice loss: 0.982 | Mean dice score: 0.509 | Time elapsed:  0 hrs  1 mins  4 secs
Completed epochs:  30/500 | Dice loss: 1.021 | Mean dice score: 0.489 | Time elapsed:  0 hrs  1 mins 34 secs
Completed epochs:  40/500 | Dice loss: 0.728 | Mean dice score: 0.636 | Time elapsed:  0 hrs  2 mins  3 secs
Completed epochs:  50/500 | Dice loss: 0.753 | Mean dice score: 0.623 | Time elapsed:  0 hrs  2 mins 31 secs
Completed epochs:  60/500 | Dice loss: 0.712 | Mean dice score: 0.644 | Time elapsed:  0 hrs  2 mins 60 secs
Completed epochs:  70/500 | Dice loss: 0.575 | Mean dice score: 0.712 | Time elapsed:  0 hrs  3 mins 28 secs
Completed epochs:  80/500 | Dice loss: 0.664 | Mean dice score: 0.668 | Time elapsed:  0 hrs  3 mins 56 secs
Completed epochs:  90/500 | Dice loss: 0.662 | Mean dice score: 0.669 | Time elapsed:  0 hrs  4 mins 25 secs
Completed epochs: 100/500 | Dice loss: 0.617 | Mean dice score: 0.691 | Time elapsed:  0 hrs  4 mins 53 secs
```

Additionally, the trained model will be saved every 100 epochs (this can be changed by changing the '*save_checkpoint*' variable in the '*Settings*' section):

```
            Checkpoint -> Model saved at  100/500 epochs
```

When all 500 epochs (this can be changed by changing the '*epochs*' variable in the '*Settings*' section) have been completed, the evolution of the training loss will be displayed as a function of epochs:



```
Total computing time:  0 hrs 23 mins  1 secs
```

Additionally, the model (in pt format) and training history (in CSV format) will be saved under the '*trained_model*' directory.

Training Performance Evaluation

What is a good training performance?

In order to evaluate whether the model was able to successfully learn how to perform the requested segmentation, you can look at the training history, as well as the final value of the Dice loss and mean Dice score.

- **Good learning performance:** Dice loss is close to 0, and mean Dice score is close to 1.
- **Bad learning performance:** Dice loss is close to 2, and the mean Dice score is close to 0.

Reasons why training performance may be low

Here is a non-exhaustive list of reasons why training performance might be low:

- **Too high variability between training samples.** This can happen either when:
  - o There are not enough samples in the training dataset (hard for the model to detect patterns).
  - o Large field of views are used in the training data. More specifically, if multiple scans are used, and if each scan is covering a large portion of the body, we can expect a lot of variability across slices (especially for

the ones at the opposite edges of the scan). Additionally, since every slice is considered a distinct sample in this 2D implementation, it may be hard for the model to detect consistent patterns across samples.

- **Large field of view used during scanning.** This can result in some ROIs being present in some slices, but not in others, which can add confusion to the model.
- **Inconsistent labeling.** This can happen when:
    - The manual segmentation was done poorly:
        - Poor attention was given by the operator who defined the edges of each ROI in an inconsistent manner.
        - The ROIs were segmented in a different order between scans or samples, resulting in one organ being labeled as ROI #1 for some samples, and as ROI #2 for others, which will confuse the model.
    - The segmentation files are not correctly loaded: When reading the segmentation file, the pixel values are inconsistent across samples. For instance, if 5 ROIs were labeled, we could expect the following pixel values: 0 for every background pixel (not labeled), 1 for ROI #1, 2 for ROI #3, etc. But if the loaded file result in inconsistent pixel values across ROIs between samples, this will confuse the model.

Testing Notebook Walkthrough

The purpose of the testing notebook is to apply the model trained using the training notebook to unseen data (which have never been segmented), and automatically generate segmentation files for the corresponding images.

Imports

Go to the '*Imports*' section and run the corresponding cell (Shift + Enter) to import all the required packages.

Settings

Go to the '*Settings*' section and edit the following variables:

- Under the DATASET INFORMATION section:
    - **labels_list:** specify the list of labels, i.e. the name of each ROI (including background), in the order that they were labeled.

WARNING: To find out the right order in which to specify each label, check the value of a random pixel located in each ROI in the segmentation image. Background should usually be the first element in that list (as it is usually labeled with a 0). From there, look at what ROI has a pixel value of 1, which should then appear second in the labels list, etc.

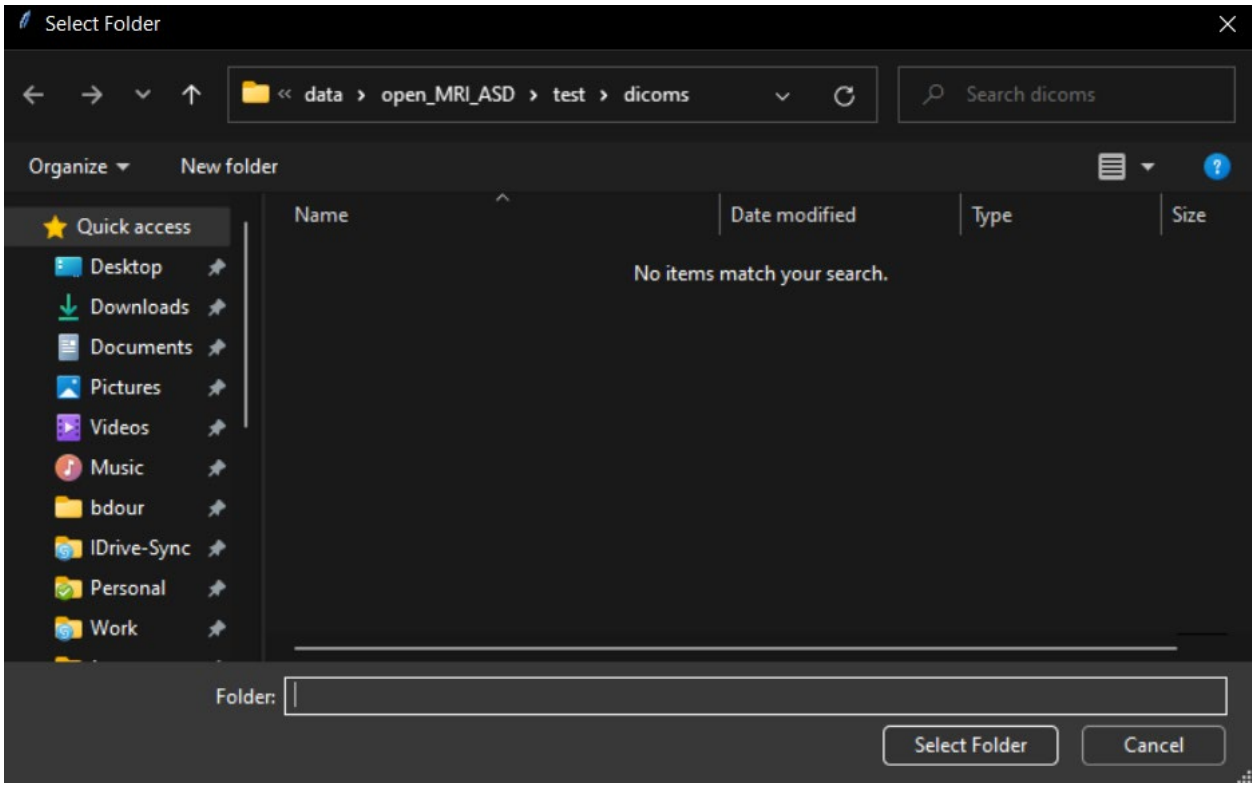
- Under the DIRECTORIES & FILENAMES section:
    - **model_filename:** specify the full name (including extension) of the trained model that you wish to use:

```
# Define model filename
model_filename = 'autoseg_model_500.pt'
```

- Under the MODEL PARAMETERS section:
    - **input_shape:** specify the input shape used during model training (if input_shape is different from the one used during training, this will result in an error).
- Under the SCANNING INFORMATION section:
    - **fov:** define field of view (in mm) used during scanning (used to estimate cross-sectional areas).
    - **matrix_size:** define matrix size used during scanning (used to estimate cross-sectional areas).

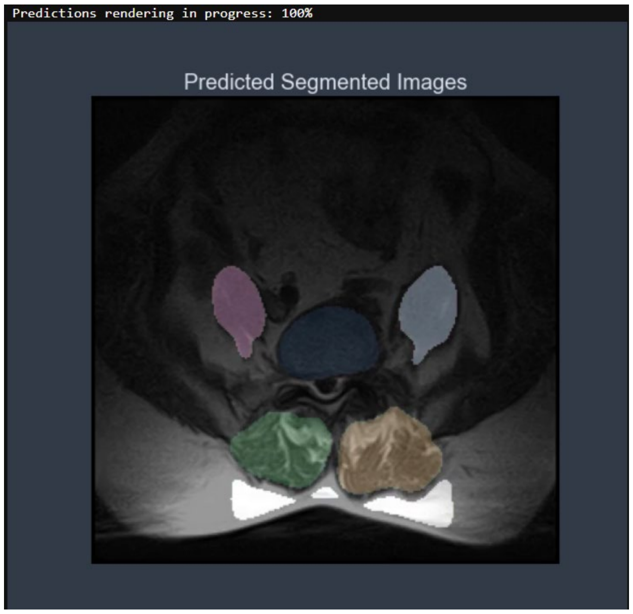
Generate Segmentation Predictions

Run the corresponding cell, which will automatically open a window in your explorer and ask you to select the folder containing the raw DICOM images that you wish to segment. Navigate to that folder, then click '*Select Folder*'.



No items match your search.

Once selected, all the DICOM images present in the selected folder will be segmented. An animated rendering of the resulting segmentation will be displayed in the notebook:

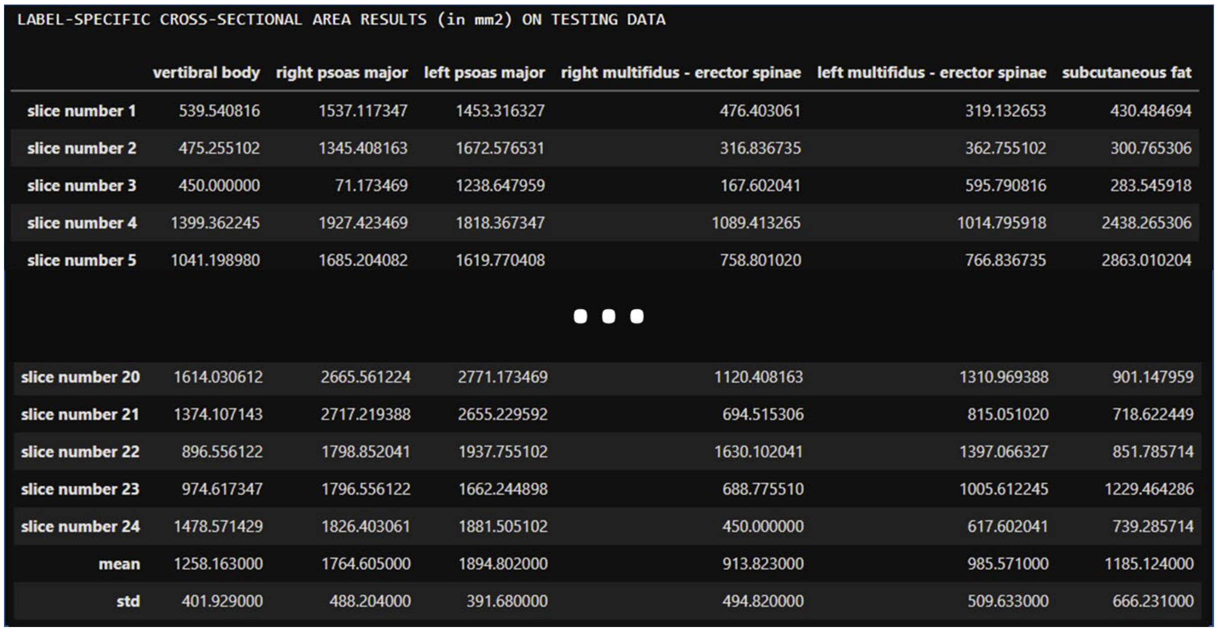


```
Predictions rendering in progress: 100%
```

The resulting segmentations will also be saved as JPG images in a folder named '*predictions*' located in the repository.

Calculate Cross-Sectional Area

Run this cell to calculate the cross-sectional area of each ROI on each image present in the selected folder. The results will be both displayed on screen, and saved in a CSV file located in the '*predictions*' folder.



```
LABEL-SPECIFIC CROSS-SECTIONAL AREA RESULTS (in mm2) ON TESTING DATA
```

	vertibral body	right psoas major	left psoas major	right multifidus - erector spinae	left multifidus - erector spinae	subcutaneous fat
slice number 1	539.540816	1537.117347	1453.316327	476.403061	319.132653	430.484694
slice number 2	475.255102	1345.408163	1672.576531	316.836735	362.755102	300.765306
slice number 3	450.000000	71.173469	1238.647959	167.602041	595.790816	283.545918
slice number 4	1399.362245	1927.423469	1818.367347	1089.413265	1014.795918	2438.265306
slice number 5	1041.198980	1685.204082	1619.770408	758.801020	766.836735	2863.010204
			• • •			
slice number 20	1614.030612	2665.561224	2771.173469	1120.408163	1310.969388	901.147959
slice number 21	1374.107143	2717.219388	2655.229592	694.515306	815.051020	718.622449
slice number 22	896.556122	1798.852041	1937.755102	1630.102041	1397.066327	851.785714
slice number 23	974.617347	1796.556122	1662.244898	688.775510	1005.612245	1229.464286
slice number 24	1478.571429	1826.403061	1881.505102	450.000000	617.602041	739.285714
mean	1258.163000	1764.605000	1894.802000	913.823000	985.571000	1185.124000
std	401.929000	488.204000	391.680000	494.820000	509.633000	666.231000