Instructions:

First, our environment uses:

- **numpy** – scientific computation and array manipulations.
- **matplotlib** – visualizations.
- **monai** – manipulating 3D medical images in pytorch environment.

To install these, run:

pip install numpy

pip install matplotlib

pip install monai

To run our experiments, you should follow these steps:

1. Configuration:
   Folders and paths configurations are found in "**config.py**" file. It should be changed to follow the following directory structure:

   Make sure to download the **labeled_data_meta_0000_05999.csv** file from the amos22 provided site (https://zenodo.org/records/7262581/files/labeled_data_meta_0000_0599.csv?download=1) and put it inside baseDir.

   baseDir:
   | --- labeled_data_meta_0000_05999.csv (meta file of the amos22 dataset)

   baseDirPelvic:
   | --- (here all the images of the first dataset reside)

   baseDirAmos22:
   | --- (follows the directory structure of the amos22 dataset):
   | --- imagesTr
   | --- imagesTs
   | --- imagesVa
   | --- labelsTr
   | --- labelsTs
   | --- labelsVa

   You should change the paths for the "baseDirPelvic" and "baseDirAmos22" in the "**config.py**" file (like the marker colors).

2. Datasets:
   (Note: Because we were mainly using google colab for development and testing (since we had no access to other sufficient hardware) we had some issues with using the given raw data as google colab is inefficient with loading many image files sequentially. Our solution was to use "CachedDataset" provided by the monai. So, we load all the modified images into cached datasets and then only load those.)

   To create the cached datasets, you can run the "**data.py**" file.

3. Training the base task on the 1st dataset:
   To train the base task on the 1st dataset run the "**train_pelvic.py**" script. It should save the results and the best model state dictionary in the baseDir. Edit the "params" variable to test different configurations, we changed the "channels" to test multiple U-Net architectures (the number of layers is assumed based on the number of numbers in the channels list, more details about tested architectures in the report).

4. Transfer learning using the 2nd dataset:
   To compare the results of training the model from scratch and fine-tuning it run the script "**train_amos22.py**". Edit the "params" variable to run different configurations. For the results in the report we used "channels=[64,128,256], bottleneck=512" tried different "train_size" values, and tried both "freeze_layers=True/False". For more details on the configurations please see the report.

5. Gradient reversal test:
   (Note: This wasn't a successful experiment, as it did not converge. We assume because of too big of a distribution shift. To focus on our main findings, this is not discussed in the report)

   To run this test run the file "**train_gradient_reversal.py**".