# CADA Tutorial

# Pre-Requisites

Basic SMILE Lab program necessities

* FileZilla or other FTP program
* Putty or other SSH program to run code remotely
* PyCharm or other SSH compatible Python IDE
* Local Anaconda – Python environment
* Server Anaconda – Python Environment

Local Anaconda Environment

* Tensorboard

Server Anaconda Environment Packages

* Python 3.6
* PyTorch 1.0.0
* Albumentations
* Tensorboard

# Set-Up for Anaconda Local Environment

1. Create your anaconda environment with **conda create –name yourenvname** and activate the environment with conda activate yourenvname
2. Install tensorboard with the command **conda install -c conda-forge tensorboard**

# Set-up for Anaconda Server Environment

1. A conda environment can be created with all the necessary packages through the command **conda create -f /path/to/save/file/packages.yml**. This will create a conda environment named cada\_tutorial installing each package. You may edit the .yml file in a notepad to change the title of the environment. Alternatively, use the command cd /path/to/directory/ and use the command **conda create -f packages.yml**

# A picture containing light, dark Description automatically generatedData-Preprocessing

1. Next, you will pre-process the original images using a pretrained ROI detection algorithm. Open the python file \data\_preprocess\Generate\_ROI.py
2. Set the size of the image to pre-process. Do not change the DiscSeg\_size (this parameter cannot be changed anyway as the model is only compatible/pretrained on 640 x 640). Change the ROI\_size\_list and DiscROI\_size to be equal and a size compatible for capturing the ROI. Refer to the paper for the data sizes.

A picture containing text

Description automatically generated

1. First create an outer folder where the pre-processed data will be located (i.e., here it is called CADA\_Tutorial\_Image). Inside, create three folders corresponding to the source dataset, target training dataset, and the target testing dataset. Create subfolders in each with named image and mask.

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated

1. Here we will be editing the directories based on your paths. First, change the data type as necessary. Currently the original data is located in /DATA/charlie/AWC/data\_original/. Use these directories for the original img and mask paths and use the new directories from the previous part (6) for the pre-processed data.

Graphical user interface, text

Description automatically generated

1. Run Generate\_ROI.py. Change the flag below to False/True based on the comment. Repeat this code three times, once for the source dataset, once for the target training dataset, and once for the target testing dataset.



1. Verify that your code ran correctly by looking through the pre-processed image folders. You may download a random picture to see. It is good practice to check through all of them for ROI quality evaluation.

A picture containing text

Description automatically generated

# Model Training

1. Find the python file \dataset\refuge.py and change all directories with the pre-processed folders that you have created for each of the images and masks. Observe the target training data has only the images not the mask as it is unsupervised.

A screenshot of a computer

Description automatically generated with low confidence

1. Find the python file arguments.py. The important parameters that can be changed are shown in the upcoming figure. The input size and target size do not matter here because they are referenced nowhere else in the code, it is only for your self-reference. If you use a batch size of 4, this will probably require 2 GPU’s.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Text

   Description automatically generatedCreate a folder for saving the Model Weights during training and a tensorboard directory as well. I have created a subfolder as well called Trial1 where they will be saved for organization in the case of multiple experiments. Enter the model weights directory into “SNAPSHOT\_DIR” and the tensorboard directory into “TENSORBOARD\_DIR” of the arguments.py file.
2. This paper has three loss contributions: supervised segmentation loss, adversarial loss, and mse\_loss, each of which are accounted for five times, one for each decoder layer. The weights for each loss can be found in arguments.py and should be chosen manually or optimized through some procedure (i.e., grid search).
3. Run CADA.py. You should train at about 100 epochs worth of training (remember - epochs vs iteration number). For these datasets here, that equates to roughly 10000 iterations. You can go back to the arguments parameters to change the number of iterations to stop at if you would not like it to run infinitely.

Text

Description automatically generated

# Testing

1. Check the Model\_Weights experiment folder that you created for the saved model weights. They are saved at every 1000 iterations due to the argument settings (again, this may be changed if you would like in the arguments.py parameters). Choosing around 10000 iterations is usually safe, you may choose various options though to examine the varying effects of training.

Table

Description automatically generated

1. Create a folder for the model prediction-segmentation results and a subfolder with the experiment title for organization. I have named it Trial1 for simplicity.
2. Find predict.py and assign the weights model of your choice to “Restore\_From” and the results path folder to “Save\_Path.” Change the ROI\_size to the size of your pre-processed target testing dataset.

Text

Description automatically generated

1. In the main section, change the directory for the ground truth labels of the testing dataset.

Text

Description automatically generated with medium confidence

1. Run predict.py. You should receive an average scoring for each of the three metrics as shown in the upcoming figure (for the sake of the tutorial, I chose a very small model of 1000 iterations, hence the score is clearly poor). Lastly, you should go back to your results folder to visualize your model predictions and masks.

Graphical user interface, text

Description automatically generated

# A Small Guide to Tensorboard

1. During training, you created a tensorboard directory which contained the training loss curves. The file will be named very similar to what is shown below. Create a file locally on your own computer with a relevant name and download this tensorboard file there. You may always rename the tensorboard file if you would like.

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Create your own local anaconda environment with tensorboard. This may have been done earlier in the tutorial.
2. In anaconda prompt, activate your conda environment. Enter the command tensorboard --logdir=DIRECTORY (where DIRECTORY is the tensorboard directory that you have created and named). You might see a different message if you have Tensorflow installed but this tutorial environment that I have created only has tensorboard on it. Nonetheless, this works.

Text

Description automatically generated

1. Copy in <http://localhost:6006/> into your browser and tensorboard should show up with the training loss curves.

# Chart Description automatically generated

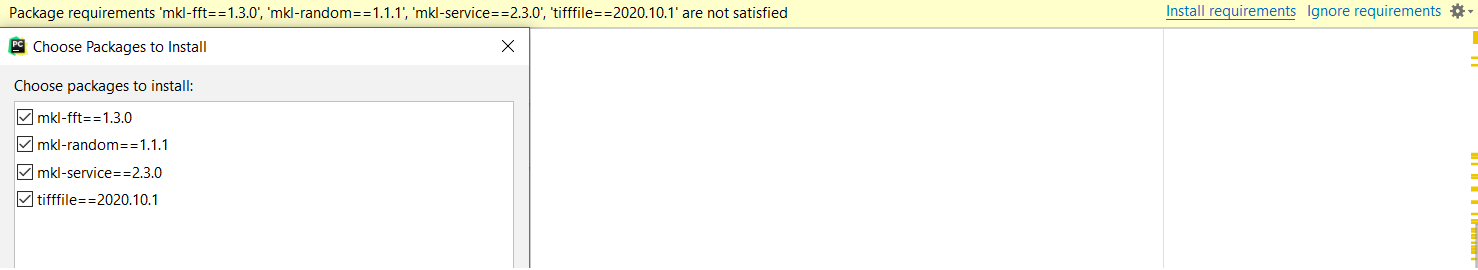
1. You can continue to download the tensorboard files during training to update the curves to monitor the progress.

# Helpful Advice

1. Be very thorough about parameter optimization. There are 15 loss function weights, a learning rate, and a discriminative learning rate, all of which are valuable to the results. There is large sensitivity to the results based on these parameters. You may use Tensorboard as a quality evaluation for training optimization.
2. Be organized. I named the folders above something simplistic (Trial1) but in practice you might want to name the folder names with tags that represent what experiment was done (i.e., REFUGE\_CADA\_lr1e-4\_lrd2.5e-5\_ema\_bsz4).

# Possible Errors

1. You may receive an error that these packages have not been satisfied due to installation errors with the yml file. The code should be safe to run regardless as they are not really used. You may install these packages manually if you would like.



1. As always, you may run into deprecation errors. This is fine in most cases as it means that you are using an outdated version of a certain package, i.e., an older SciPy.
2. It might be possible to use Tensorboard on the server rather than locally to see training updates in real-time. If you know how, that is perfectly fine, I have never looked into it in depth.

If you have any questions, you can contact me in the SMILE Lab or email me at [charlietran@ufl.edu](mailto:charlietran@ufl.edu).