ML Exercise

Task

Implement and train a Fully Convolutional Neural Network (FCNN) model for segmenting an image into two classes in PyTorch or Tensorflow. The network architecture is the following:

Input -> conv1 -> ReLu -> conv2 -> ReLu -> MaxPool 2x2 -> conv3 -> relu -> Upsample 2x2 -> conv4 -> Output

The input patch is of size WxHx3 (RGB), and the corresponding output patch is of size WxHx2.

The layer details follow:
conv1: 3x3, 16 channels, padding 1 conv2: 3x3, 32 channels, padding 1 conv3: 3x3, 16 channels, padding 1 conv4: 5x5, 2 channels, padding 2

Requirements

- Implement the code in Python 3 with PyTorch or Tensorflow. You are allowed to use image loading modules such as skimage or PIL.
- Save the predictions image with your best results to a file called **predictions.jpg**.
- For training, write a script **train.py** that trains the network on the training image. Add the necessary code to plot the training loss as a function of the number of iterations. Assume that at some point we will train this algorithm on a GPU and that the network cannot fit in the GPU memory for an input image larger than 256x256x3.
- Write a script **predict.py** that predicts a trained network on the whole training image. The restriction on the GPU memory from the previous item applies here as well.
- Provide instructions on how we can execute your program. Ideally specify a “requirements.txt” file with all the dependencies.

Notes

- We are aware the training set is very small and that the model may overfit. Our focus for this exercise is on the implementation and understanding of the network. There is no need to do cross-validation or early stopping.
- There is no need to run this on a GPU, we are satisfied with the code running on the CPU. This way you will be able to try it on a laptop or your home computer.
- You may need to make some assumptions in order to implement the training and prediction code. If you have any questions don’t hesitate to contact us.