Experiments

- Normalization:
  - Pixel values linearly normalized to range of [0, 1]
- Training scheme/ data augmentation:
  Pre-processing:
  - Random horizontal and vertical flip on the training set (20 images → 60 images)

  On-the-fly:
  - Train on random patches (i.e. 64*64, 128*128, 256*256, 512*512) selected from the images (motivated by https://towardsdatascience.com/vessel-segmentation-with-python-and-keras-722f9fb71b21)
- Architecture
  - Unet (https://pytorch.org/hub/mateuszbuda_brain-segmentation-pytorch_unet/)
- Loss Function:
  - Pixel-wise binary cross-entropy loss
  - Soft dice loss (i.e. to address class imbalance)
- Optimizers:
  - Tested SGD, Adam, and RAdam; ended up using RAdam for best performance
- Prediction reconstruction
  - Original image resized to 512*512
  - Predicted segmentation reconstructed from patches of corresponding size used in training (i.e. 64*64, 128*128, 256*256, 512*512)
- Hyperparamters:
  - lr = 0.001, batch_size = 4, epoch = 250, default beta and weight decay values

Model trained on GCP with GPU.
Some training graphs/hyperparameter details:
https://app.wandb.ai/chloewangxq/vessel_segmentation

For testing and reconstructions, original images and corresponding masks are resized to (512, 512) with nearest neighbor interpolation (yet to explore bilinear + thresholding). Predicted segmentation is reconstructed from patches obtained from the corresponding model (i.e. for model trained on patches of size 256*256, the original image is split into 4 sub-images each being fed into the NN, and the predicted patches are recombined to reconstruct the full predicted segmentation; some changes made to the eval code).

	Soft dice loss				Pixel-wise cross entropy			
Patch size	Class 0 IoU	Class 1 IoU	Mean IoU	Accuracy	Class 0 IoU	Class 1 IoU	Mean IoU	Accuracy
512*512	0.9688	0.6564	0.8126	82.49	To be tested			
256*256	0.9704	0.6701	0.8203	83.03	To be tested			
128*128	0.9685	0.6623	0.8154	85.31	0.9725	0.6652	0.8189	75.84
64*64	0.9640	0.6280	0.7960	83.98	0.9699	0.6391	0.8045	74.21