cellMaskRCNN tutorial

This tutorial illustrates how to use cellMaskRCNN.py to segment cells. We will use some synthetic data generated with *generateSynthCellsImage* from *gpfunctions.py.*

Note that this tutorial merely intends to show how to use the script, not how to get the best possible segmentation results.

1. Data Generation

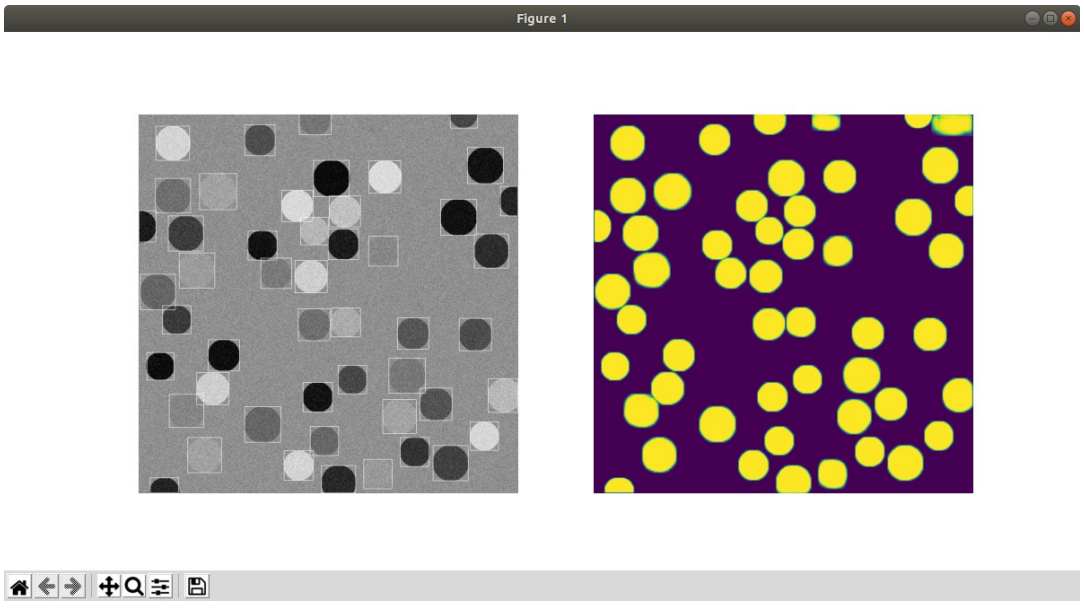
The auxiliary notebook *t05_cellMaskRCNN.ipynb* contains 3 code cells to generate training, test, and deployment data. Please refer to that file for this step.

2. Training, Testing, Deployment

The 'control panel' section of *cellMaskRCNN.py* is simpler than that of *unet2D.py* and *unet3D.py*. Here the phase choice is simply a matter of setting the variable *mode* to 'train', 'test', 'deploy' or 'deploy with PI2D'.

mode = 'train' will fine-tune a model pre-trained on the COCO instance segmentation task for as many epochs as set with *num_epochs*. A fraction of *train_subset_fraction* images from *dataset_path* will be used for training and the remaining will be used for testing. Loss on the training set, and Dice coefficient on the test set, are shown during the training process, for inspection purposes.

mode = 'test' will display a pair of outputs per image from the test set: an image containing bounding boxes overlaying the original image, and another containing cell likelihoods. One example is provided below.



mode = ‘deploy’ will save prediction in the *deploy_path_out* folder, preserving names that correspond to the input images from *deploy_path_in*. Since the underlying Mask-RCNN model has a limitation of 100 objects per image, only a maximum of 100 cell will be segmented in this mode.

mode = ‘deploy with PI2D’ uses *PartitionOfImageOM.py* to split images into tiles with overlap, segment them, then assemble the output. The output is similar to the one above. The prediction without PI2D is also given for comparison. They are saved as 16-bit .tif and 8-bit .png, respectively, in folder *deploy_path_out.* Below is a sample trio of input, output without PI2D, and output with PI2D.