unet2D tutorial

This tutorial illustrates how to use unet2D.py to highlight cell contours. 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 *t03_unet2D.ipynb* contains 3 code cells to generate training, test, and deployment data. Please refer to that file for this step.

2. Training

From now on, we'll use the *unet2D.py* script. To configure it for training, edit the parameters of the 'control panel' session like so:

```
restoreVariables = False
(we're training from scratch, so there is no model to restore)

train = True

test = False
(we're only training now)

deploy = False
(we're only training now)

imSize = 60
(each patch in the training set has size 60x60)

nClasses = 2
(our training images are labeled with 2 classes: contour and non-contour)

nChannels = 1
(images have only 1 channel)

batchSize = 32
(how many images per training batch; this could be higher or lower depending
on your machine)

modelPathOut ='Models/unet2D_v0.ckpt'
```

```
reSplitTrainSet = True
```
(the script divides the training set in 'training' and 'validation'; since we're training for the first time, we have to record the split, so we can re-use it when we 'fine-tune' the model)

```
trainSetSplitPath = 'Models/trainSetSplit2D.data'
```

```
logDir = 'Logs/unet2D'
```
(we will look here to inspect the model during training)

```
logPath = 'Logs/unet2D_TestSample.tif'
```
(this is also useful to inspect the model during training)

```
imPath = 'tutorials/DataForUNet2D/Train_60'
```
(path to folder containing [image, label] training pairs)

```
imPathTest = 'tutorials/DataForUNet2D/Test'
```
(even though we're not 'testing' now, we will use the images here to inspect the model during training; the output is recorded in logPath set above)

```
nFeatMapsList = [16,32,64]
learningRate = 0.00001
```

```
nEpochs = 1
```
(the number of epochs depends on the case; we can train for more epochs later if needed)

```
useGPU = True
```
(this should work even if you don't have a GPU, but you can also set it to False in that case)

You can now train the model by executing unet2D.py with these settings. On Terminal, run

```
python unet2D.py
```

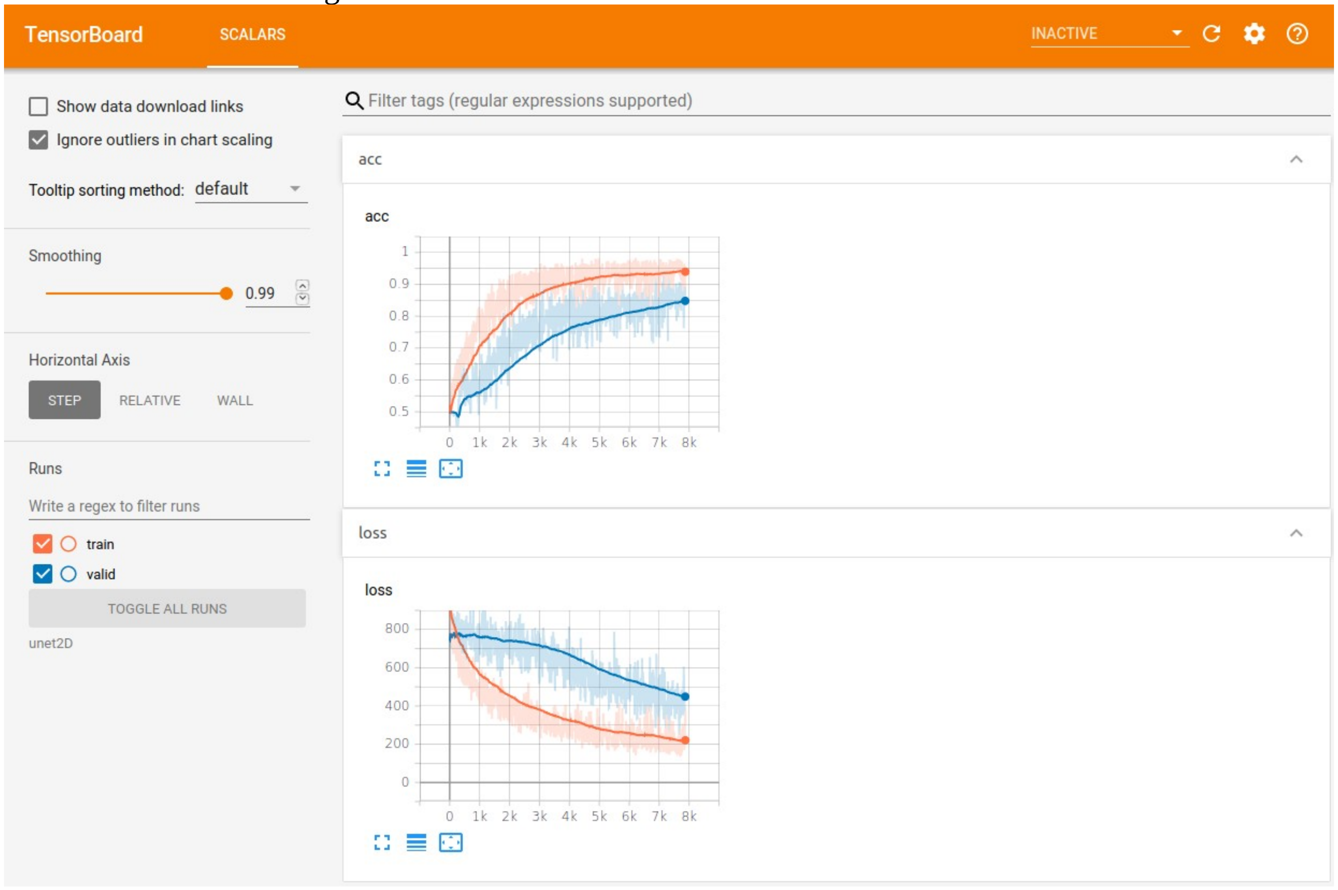
To inspect the model, open a new Terminal window and point *tensorboard* to the folder set as *logDir:*

```
tensorboard -logdir=Logs/unet2D --bind_all
```

Then point your web browser to

```
localhost:6006
```

You should see something like this:



You can also inspect prediction results on a random image from the test set by looking at the image saved in *logPath*.

3. Fine-Tuning

To ‘resume’ training, simply run unet2D.py script as above, except for the following modifications:

```
restoreVariables = True
reSplitTrainSet = False
modelPathIn = 'Models/unet2D_v0.ckpt'
modelPathOut = 'Models/unet2D_v0.ckpt'
```

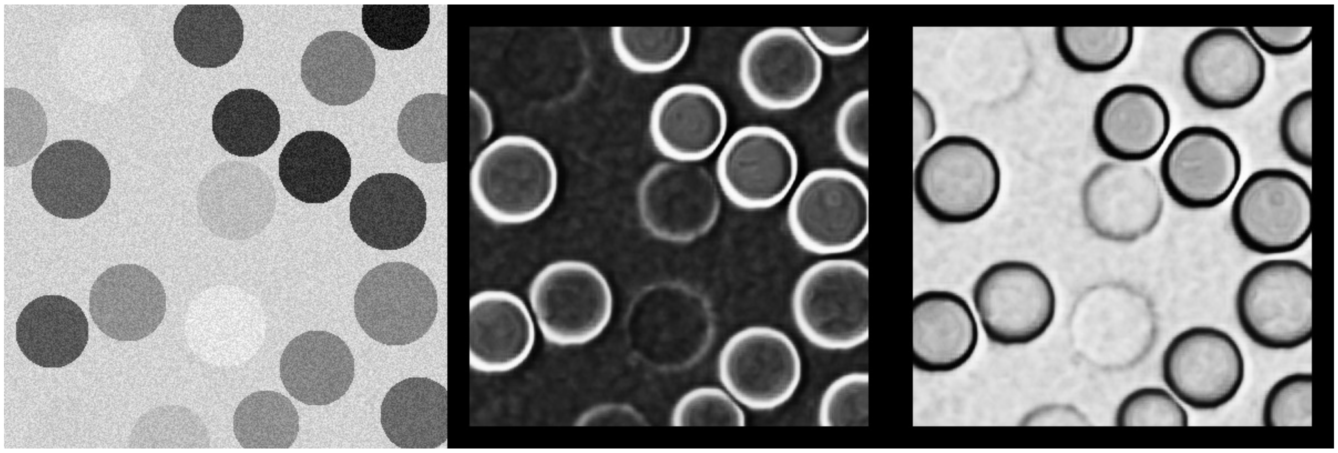
If you want to keep different versions of the model, simply give *modelPathOut* a different name from *modelPathIn*, e.g. '*Models/unet2D_v1.ckpt*'.

4. Testing

To configure unet2D.py to ‘test’ mode, modify the parameters of the ‘control panel’ session from the ‘training’ mode like so:

```
restoreVariables = True
train = False
test = True
deploy = False
reSplitTrainSet = False
```

The script will save predictions of each image in the *imPathTest* folder, under the same folder, appending ‘Pred_’ before their names. In our example one such image looks like this, where on the left is the original input, and after that the probability maps for each class are concatenated.



5. Deployment

To configure unet2D.py to ‘deploy’ mode, modify the parameters of the ‘control panel’ session from the ‘training’ mode like so:

```
restoreVariables = True
train = False
test = False
deploy = True
reSplitTrainSet = False
```

The model can be deployed either to an image, to a folder of images, or both. The following configuration deploys only to a folder, since *deployImagePathIn* is set to an empty string:

```
deployImagePathIn = ''
deployFolderPathIn = 'tutorials/DataForUNet2D/Deploy_In'
```

The following configuration deploys only to an image, since *deployFolderPathIn* is set to an empty string:

```
deployImagePathIn = 'tutorials/DataForUNet2D/Deploy_In/I00000_Img.tif'
deployFolderPathIn = ''
```

Deployment outputs are saved under *deployFolderPathOut*, with output names identical to corresponding input names. Outputs are multi-channel images where each channel is a probability map.