Sentinel-2 MSI Data for Urban Land Cover Mapping with Superpixel and Deep Learning

Yifang Ban, Professor, yifang@kth.se
KTH Royal Institute of Technology

Introduction

The purpose of this tutorial is to evaluate an object-based classification of Sentinel-2 MSI data for urban land cover mapping using deep learning. First, you will be introduced to an image segmentation method called Simple Linear Iterative Clustering (SLIC). SLIC is a state of the art segmentation algorithm that clusters pixels in the combined five-dimensional color and image plane space to efficiently generate compact, nearly uniform superpixels (EPFL, 2010). Then you will learn how to use Tensorflow to perform image classification using convolutional neural networks (CNN), and how to visualize data and network architecture with TensorBoard. TensorBoard provides the visualization and tooling needed for machine learning experimentation. At the completion of the tutorial, you will be able to classify remotely sensed images with deep learning and to conduct accuracy assessment. Figure 1 presents the methodology overview of this tutorial.

Note: ***It is highly recommended to run "Step0_Data_Preparation.py" in advance, because the data sampling procedure may take 25min.***

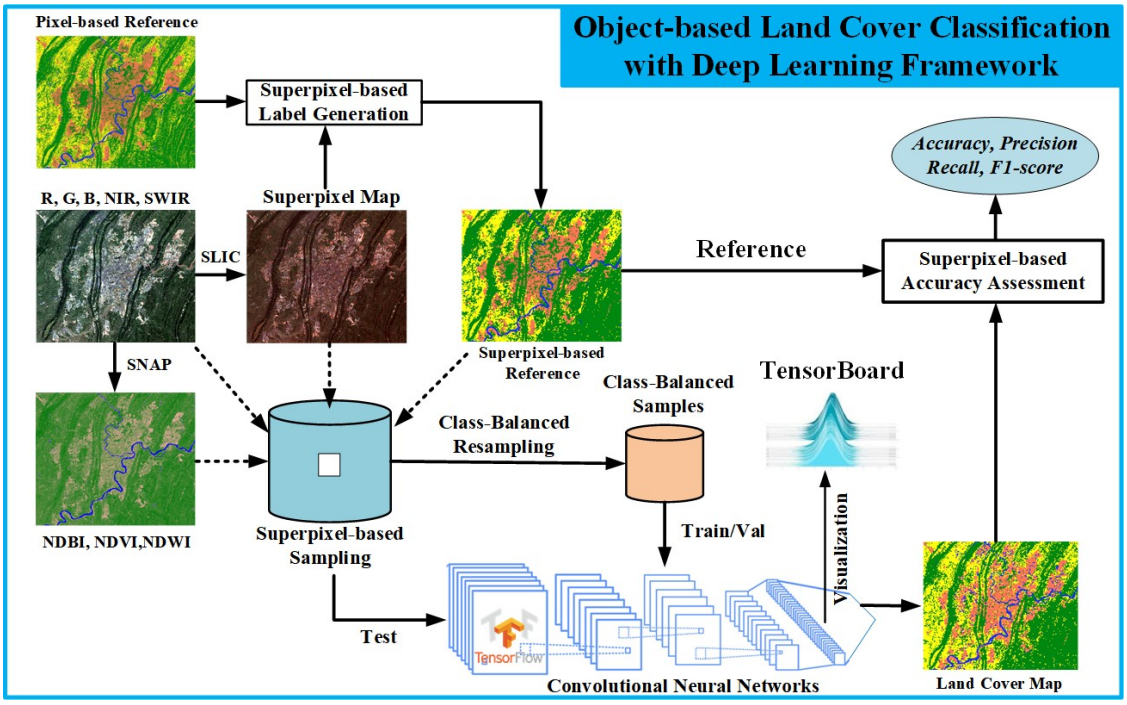


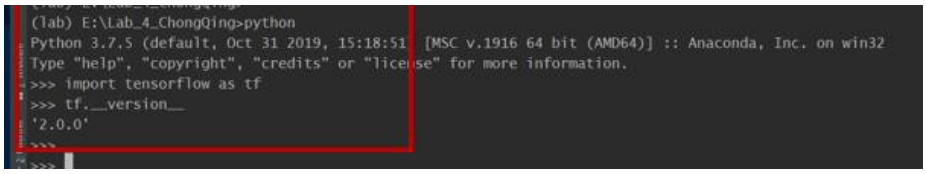
Figure 1. The methodology overview

0. Check Environment and Run code "Step0_Data_Preparation.py"

Open *pyCharm*, and click the Terminal, find and set your work path with "cd *dataPath*".

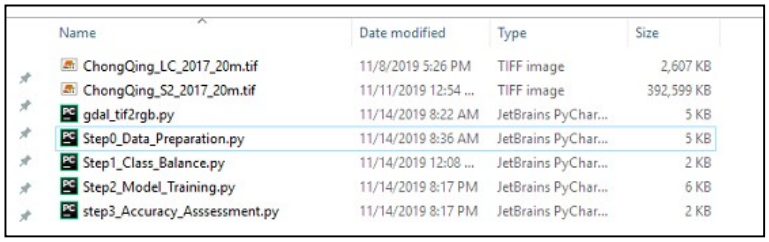
(wechat group)

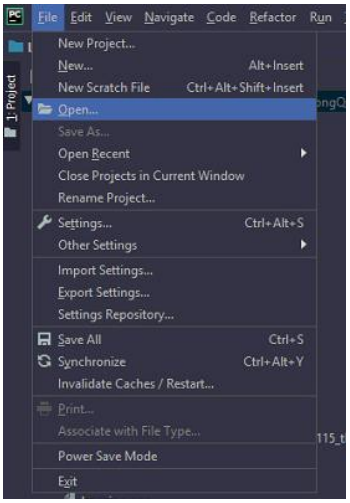
List all conda environment with "conda list env", and then select the ***lab*** environment and activate it with "conda activate ***lab***", then check ***tensorflow*** version as follows.

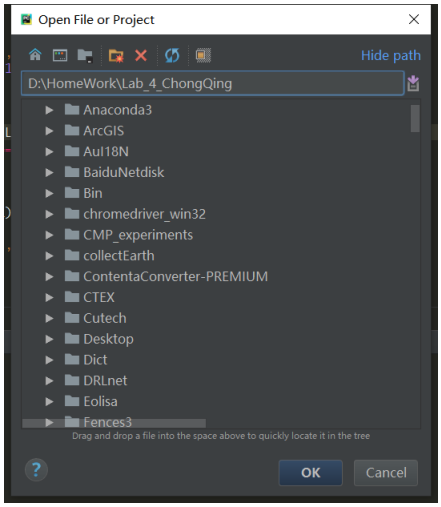


Find the path that stores data and codes, use pyCharm to open this directory. Double-click "*Step0_..._.py*", select python.exe belongs to environment *lab*, then run with

by right clicking mouse, and the code can automatically obtain its current working directory with *os.getcwd()*.







1. Sentinel-2 Multispectral Data Visualization using Spectral Indices

This are two image files in ".tif" format:

(1) "**ChongQing_S2_2017_20m.tif**" is the Sentienl-2 image mosaic containing five bands (Green, Blue, Red, NIR and SWIR) and three spectral indexes including NDVI, NDWI and NDBI (see Table 1)

Table 1: Bands contained in ChongQing_S2_2017_20m.tif

Band_1	Band_2	Band_3	Band_4	Band_5	Band_6	Band_7	Band_8
Blue	Green	Red	NIR	SWIR	NDVI	NDWI	NDBI

(2) ChongQing_LC_2017_20m.tif is the reference land cover data including urban, vegetation, crop ad water generated using FROM-GLC10 (Tsinghua, 2017).

1.1 Visualize Sentienl-2 MSI Data with SNAP

a. When you load the S2 data in SNAP, you will see 8 bands; double click one of these bands to visualize the corresponding grey scale image in the main window and its histogram in the lower left window (see Fig 1.1).

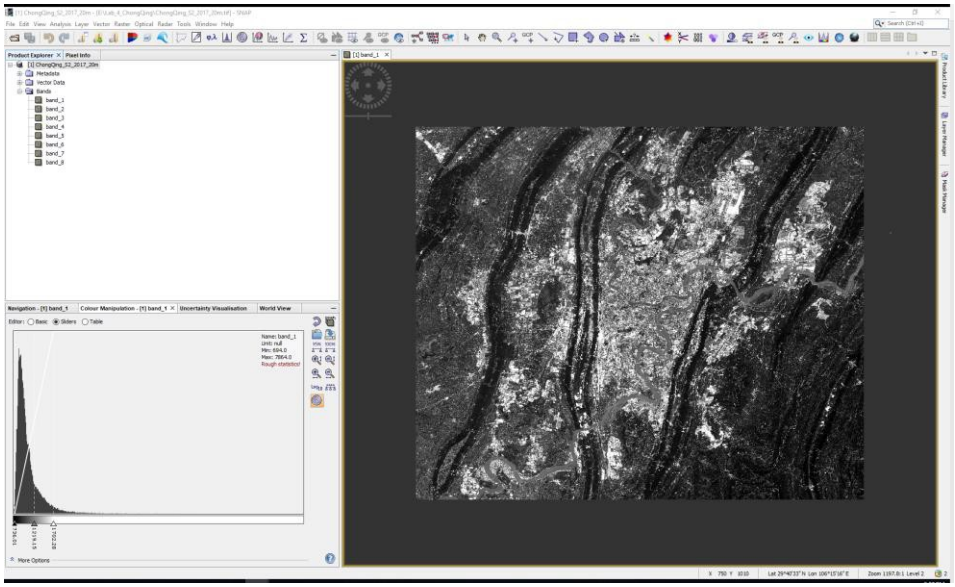
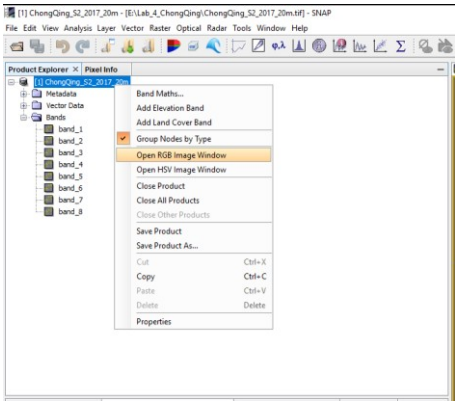


Fig 1.1 Visualize image band and histogram using SNAP

b. In the file browser window, right click the "ChongQing_S2_2017_20m" file, and select "Open RGB Image Window", then select Red = band3, Green=band2 and Blue=band1 to visualize a true color image. Try to visualize different bands composites such as [NIR, Green, Blue] and [SWIR, Green, Blue] etc.



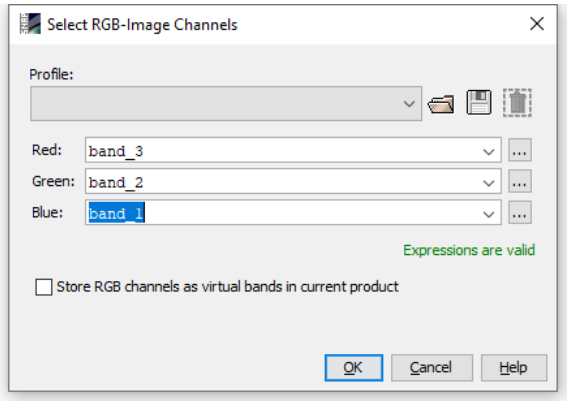


Fig 1.2 Open RGB composite

1.2 Compute a spectral index such as NDVI, NDWI and NDBI

Considering the following formulas for each of the spectral indexes:

a. NDVI (Normalized Difference Vegetation Index)

$$NDVI = (NIR - Red)/(NIR+Red)$$

b. NDWI (Normalized Difference Water Index)

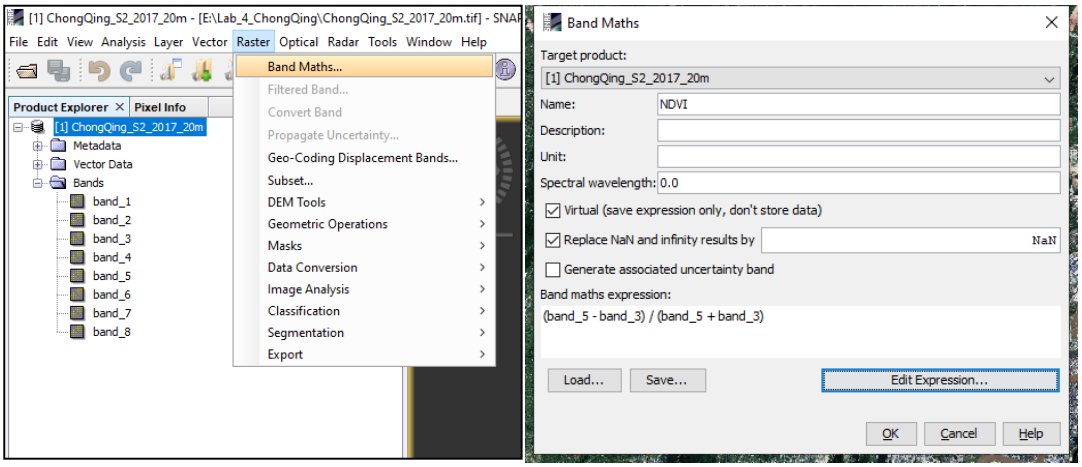
$$NDWI = (Green - SWIR)/ (Geen + SWIR)$$

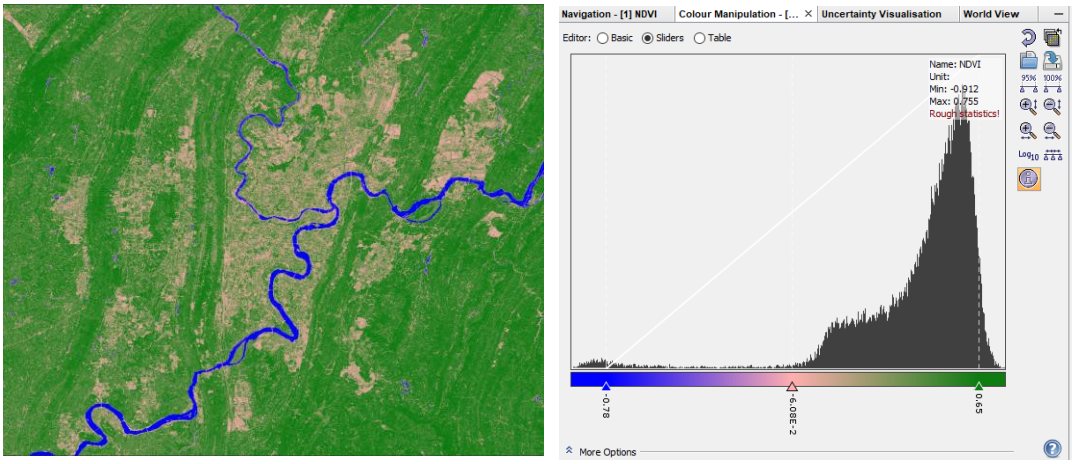
c. NDBI (Normalized Difference Built-up Index)

$$NDBI = (SWIR - NIR)/(SWIR + NIR)$$

To compute a spectral index using the Band Maths tool provided by SNAP (Note: These three spectral indexes have been contained in *ChongQing_S2_2017_20m.tif* to run code Step0, try to produce them by yourself):

Raster -> Band Maths -> Edit expression, and another NDVI band will be produced, set palette for NDVI and have a look. Similarly, NDWI and NDBI can be generated.





2. SLIC-based Superpixel Segmentation and Sampling

2.1 SLIC segmentation

In this section you will investigate different input parameters of the SLIC technique to understand the segmentation process.

Here is the code line where the slic function is used to generate the segments.

```
segments = slic(image, n_segments=30000, compactness=30)
```

- image can be 2d or 3d, and grayscale or multichannel,

- n_segments denotes the approximate number of labels in the segmented output image,

- ***compactness*** balances color proximity and space proximity: Higher values give more weight to space proximity, making superpixel shapes more square/cubic.

It is recommended to explore possible *n_segments* and *compactness* values on a log scale, e.g., 0.01, 0.1, 1, 10, 100 to find a good dimension and shape of the segments (see Fig 2.1)

(Ref:https://scikitimage.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.slic).

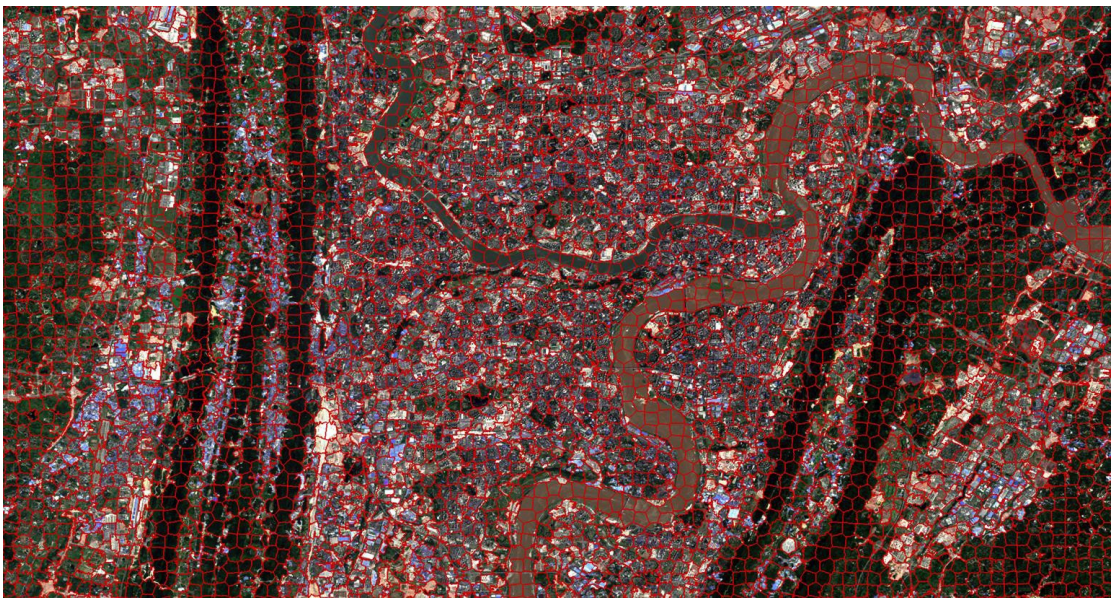


Fig. 2.1 Superpixel Segmentation Result

2.2 Superpixel-based Data Sampling

Centering at the geometric centroid of each superpixel, a cubic neighborhood of size *w*w*c* can be sampled, where *w* denotes the neighborhood window while *c* denotes the number of channels.

(**Note:** The sampling procedure may take 25min or more, please familiarize yourself with the following material while waiting.)

```
samples, labels = superpixel_based_patch_sampling(segments, Data, winSize,refMap)
```

- samples: *n*w*w*c*
- labels: *n*1*

2.3 Balance Training Data

Due to the fact that the number of training samples belongs to different classes that are so different, it is recommended to transform them into class-balanced one. Here, 2000 samples are randomly chosen for each class, and they will be taken as the training samples for deep neural networks.

3. Convolutional Neural Networks (CNN)

3.3 Data/Feature Visualization

Tensorboard Projector can be exploited to visualize the sample distribution with t-distributed stochastic neighbor embedding (t-SNE)
[TensorboardX : https://tensorboardx.readthedocs.io/en/latest/tensorboard.html].

```
''' ////////////////////// Original Data Embedding ////////////////////////////////// '''
subImg = ChongQing['images'][..., [0, 1, 2]].transpose(0, 3, 1, 2)
print("subImg shape: ", subImg.shape)
n, c, w, h = subImg.shape

# featImg = subImg.reshape([-1, c*w*h])
# print(images.max())
# writer.add_embedding(featImg, metadata=labels, label_img=subImg)
```

writer.add_embedding(featImg, metadata=labels, label_img = subImg)
-*featImg*: n*d, where n is the number of samples, d is the dimension of features
-*metadata*: labels corresponding to featImg, of size *n*1*
-*label_img*: *n*w*h*c*, where w, h and c denote width, height and number of channels of image set to visualize.

Try different bands for training and visualization via ***band4train*** and ***band4projector***.

```
''' ////////////////////////// Read Class-Balanced Sample Set ////////////////////////// '''
ChongQing = spio.loadmat(dataPath + "Samples_classBalanced_1000.mat")
band4train = [2, 1, 0]_# bands used for training, 0-8: B, G, R, NIR, SWIR, NDVI,  NDWI, NDBI
images = ChongQing['images'][..., band4train]
labels = ChongQing['labels'].transpose().squeeze()

print("images shape: {}".format(images.shape))
print("labels shape: {}".format(labels.shape))

''' ////////////////////// Original Data Embedding ////////////////////////////////// '''
band4projector = [2, 1, 0]_# bands used for projector, at most 3 bands, 0-8: B, G, R, NIR, SWIR, NDVI,  NDWI, NDBI
subImg = ChongQing['images'][..., [2, 1, 0]].transpose(0, 3, 1, 2)
print("subImg shape: ", subImg.shape)
n, c, w, h = subImg.shape
```

3.2 CNN Model

The follow codes show how to use keras layers to build a simple CNN model, taking the input data of shape 25*25*3 as example, and all samples belong to four classes, i.e., urban, crop, vegetation and water.
Please refer to TensorFlow https://tensorflow.google.cn/ if more details are needed.
Choose an optimizer and loss function for your model.

```
""" ////////////////////// CNN Model ////////////////////////////////////"""
model = tf.keras.Sequential([
    layers.Conv2D(64, (4, 4), activation='relu', input_shape=(25, 25, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dense(4, activation='softmax')
])

model.summary()

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy']
              )
```

3.3 Model Training

Configure parameters for your model fitting such as epoch and callback.

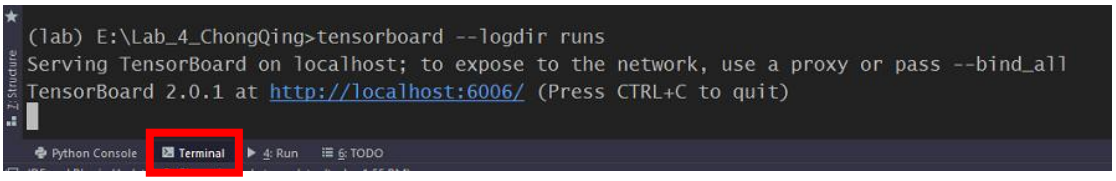
```
log_dir="runs\\" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

history = model.fit(train_ds,
                    epochs=10,
                    validation_data=val_ds,
                    callbacks=[tensorboard_callback])
```

3.4 Visualize Your Model

By using *tensorboard_callback* and *writer.add_embeddings,* try to run your tensorboard server in your local machine by inputing the following code in Terminal (in red box).

```
> tensorboard –logdir runs
```



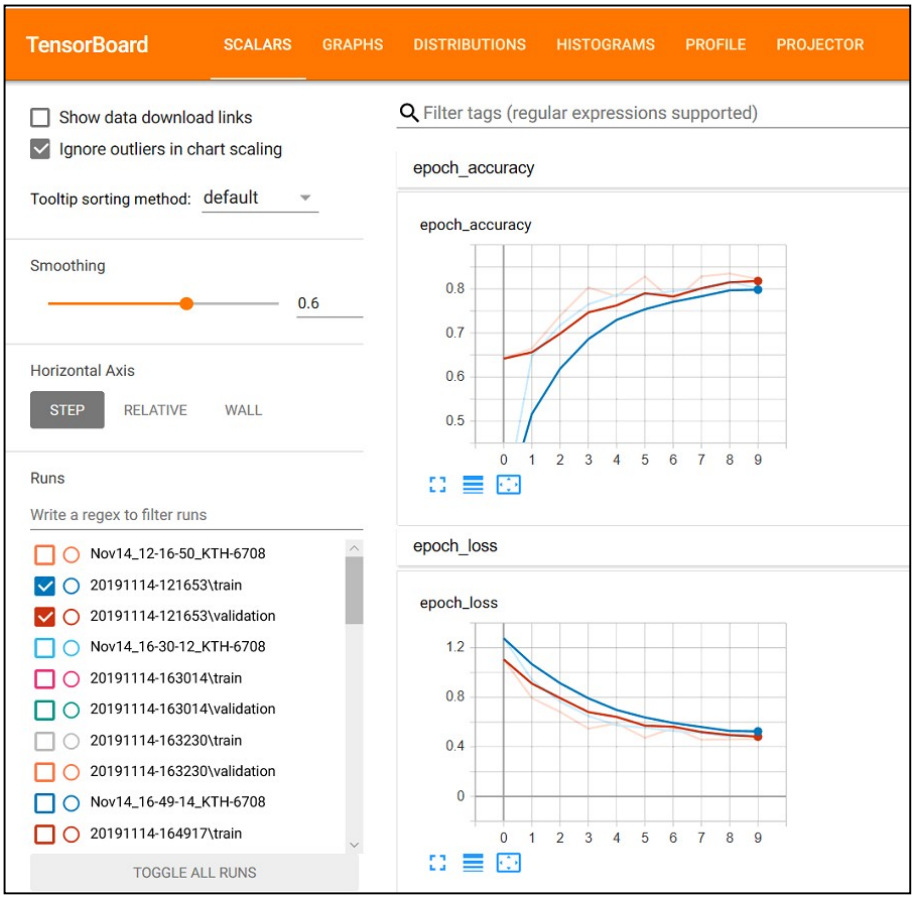


Fig 3.1 Learning Curve

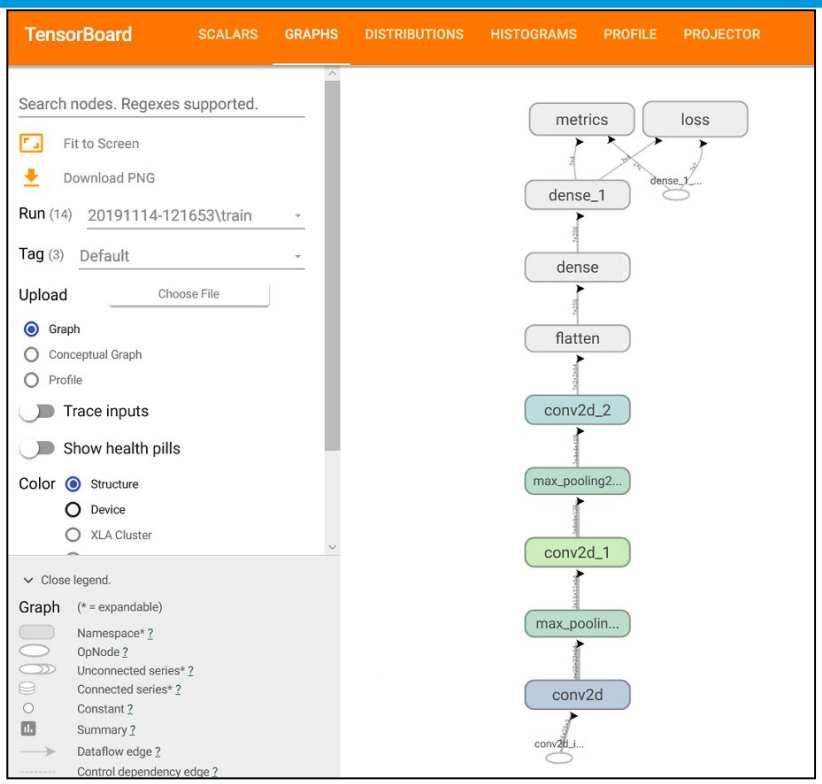


Fig 3.2 Network Archtecture

Open the url localhost://6006 via your browser (it is recommended to use **Google Chrome**). Fig 3.1 shows the learning curve with scalars, Fig 3.2 shows the network architecture with graphs, and Fig 3.3 visualize the data distribution with Projector, try different dimension reduction methods, check the difference. The Projector may not work well in other browsers except google chrome.

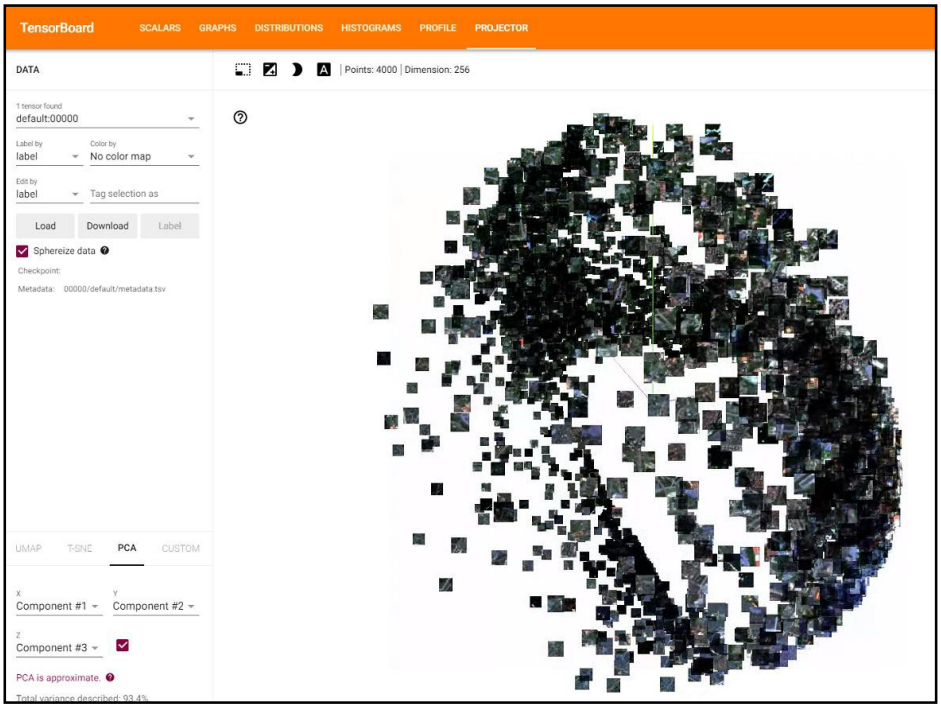


Fig 3.3 Data Visualization with Projector

4. Apply the trained model

In this section, we will apply the trained model to predict a label for each superpixel, and transform the labeled superpixels label into a land cover map.

4.1 Predict labels for superpixel with the trained model

Apply the trained model on all superpixel samples, and obtain the corresponding labels.

```
pred = model.predict(dataset0)
predLabels = np.argmax(pred, axis=1)
```

4.2 Transform labeled superpixel into a land cover map

How will you transform labeled superpixels into a land cover map?
The easiest way is to find the indexes for each superpixel, and then assign them with the same superpixel label predicted by the trained model. However, it is quite slow in practice. Another more efficient way is to find the indexes for all superpixels (segments) with the same predicted label (see the following code).

```
'''/////////////// Transform superpixel labels into a classified map ///////////////'''
start = time.perf_counter()
predMap = segments
for spLabel in np.unique(predLabels):
    a = np.where(predLabels == spLabel)[0]
    predMap[np.isin(segments, a)] = spLabel
```

Fig 4.1 shows the superpixel-based classification results predicted by CNN. Try to adjust the parameters such as epoch, network architecture and optimizer, and see how will the predicted map look like.

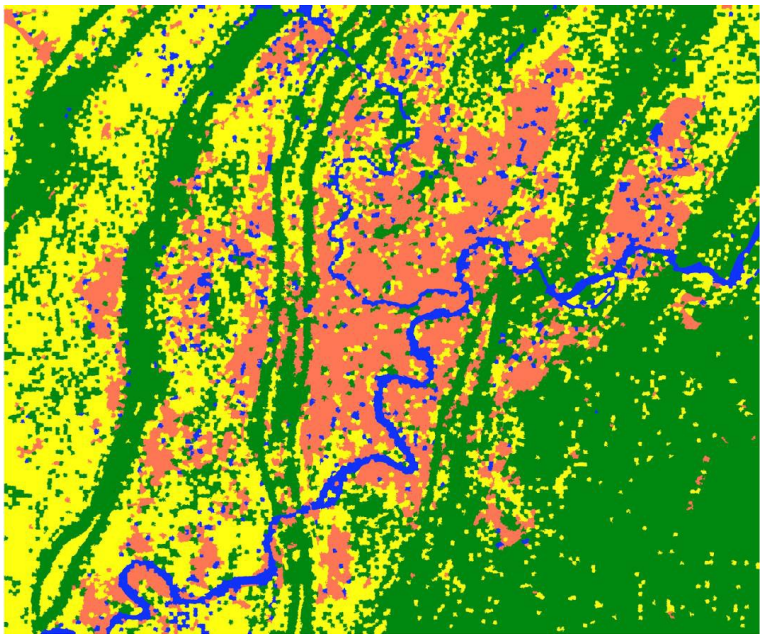


Fig 5.1 Map predicted by CNN (epoch=10)

5. Accuracy Assessment

In this section, we will conduct accuracy assessment by comparing the classified map with the reference map. There are two accuracy assessment schemes: pixel-based and superpixel-based accuracy assessment, which do you think is more reliable to assess the CNN results?

"Step3_Accuracy_Assessment.py" show how to visualize pixel-based and superpixel-based reference maps, and a superpixel-based accuracy assessment method is given. Fig 5.1 shows the pixel-based reference map, while Fig 5.2 shows the superpixel-based reference map, try to discuss their characteristic and difference between them.

Fig 5.3 presents the code for conducting accuracy assessment with *sklearn.metrics*. Please refer to the following url: http://lijiancheng0614.github.io/scikit-learn/modules/classes.html#module-sklearn.metrics

C_ij: number of superpixels belong to class-i are classified into class-j.

Table 3. Four-Class Confusion Matrix.

-	**True Label**					
Predicted Label	-	Urban	Crop	Vegetation	Water	**Recall**
	Urban	C00	C01	C02	C03	
	Crop	C10	C11	C12	C13	
	Vegetation	C20	C21	C22	C23	
	Water	C30	C31	C32	C33	
	Precision					**Acc/F1**

The accuracy, precision, recall and F1-score can be computed with the formula:

$$\text{acc} = \frac{\sum_{k=0}^{3} C\_kk}{\sum_{i=0}^{3}\sum_{j=0}^{3} C\_ij} \quad \text{precision}_k = \frac{C\_kk}{\sum_{i=0}^{3} C\_ik} \quad \text{recall}_k = \frac{C\_kk}{\sum_{j=0}^{3} C\_kj} \quad k=0,1,2,3.$$

$$[\text{class k } F_1]\text{: } F1_k = \frac{2*Precision_k*Recall_k}{Precision_k+Recall_k}$$

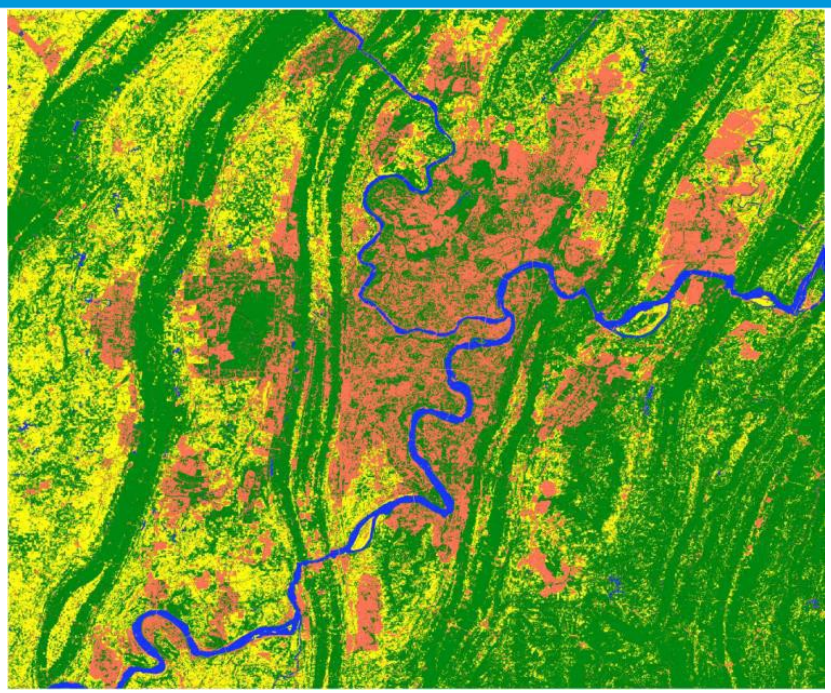


Fig 5.1 Pixel-based reference map

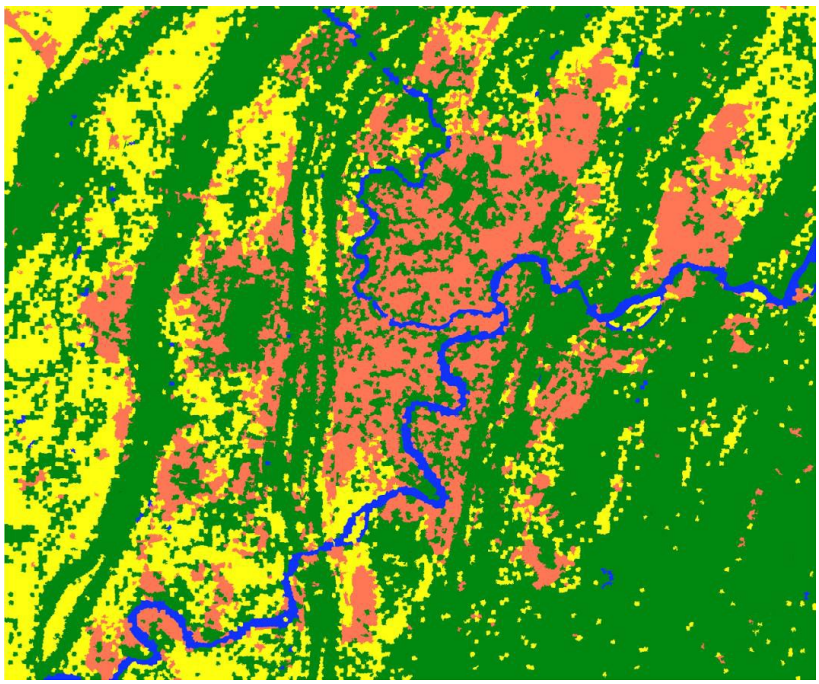


Fig 5.2 Superpixel-based reference map

```
''' Accuracy Assessment '''
confMatrix = confusion_matrix(spPredLabels, spRefLabel)
acc_score = accuracy_score(spPredLabels, spRefLabel)
F1_score = f1_score(spRefLabel, spPredLabels, average='weighted')
```

Fig 5.3 Code for accuracy assessment

```
-------- Confusion Matrix --------
  urban   crop   veg  water
[[ 4938   682  2974    56]
 [  117  3779   209    12]
 [ 1377    56 13524    53]
 [   54   240    42   606]]
---------------------------------
accuracy Score: 0.80
F1-score : 0.80
```

Fig 5.4 Accuracy Statistics

Congratulations!

==========> End <==========

Appendix: Python Environment Configuration

puzhao@kth.se

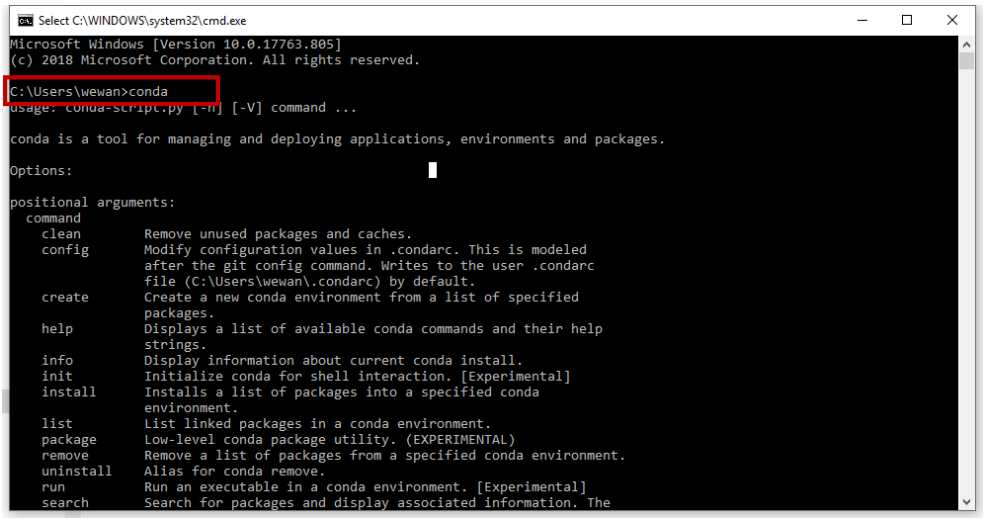
Step 0: Install Anaconda and PyCharm

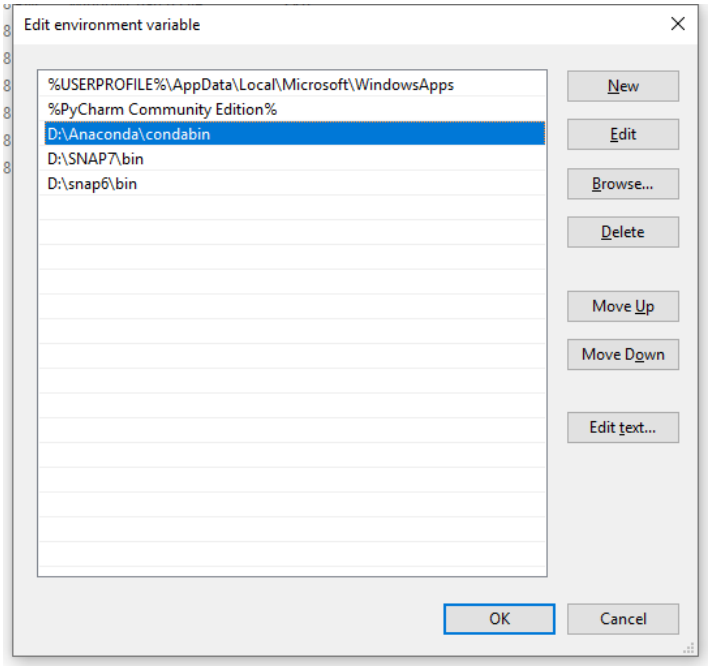
Anaconda (Python 3.7 64-bit) https://www.anaconda.com/distribution/#windows

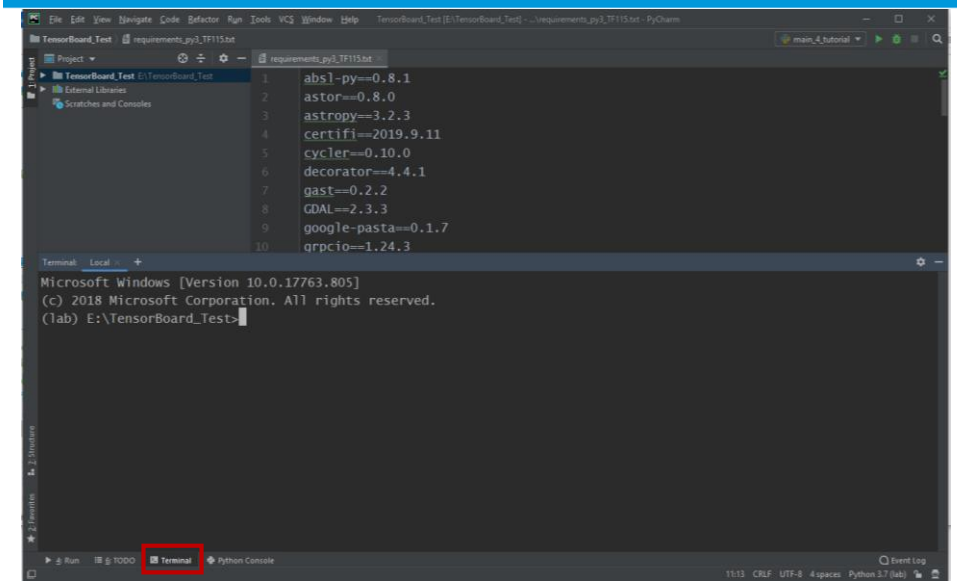
PyCharm-Community

https://www.jetbrains.com/pycharm/download/#section=windows

Test conda: It works well if it looks like the following picture, otherwise you need to add the path of "conda.bat" into the system environment variable (see the second picture).







Step 1: Create an environment named "lab" and activate it (in Pycharm Terminal)

> conda create –name lab

> activate lab

(base) E:\ Lab_4_ChongQing >activate lab

(lab) E:\ Lab_4_ChongQing >

Step 2: Install python libraries required for lab (in Pycharm Terminal)

> conda install pip==19.1.1

> conda install gdal

> conda install cytoolz

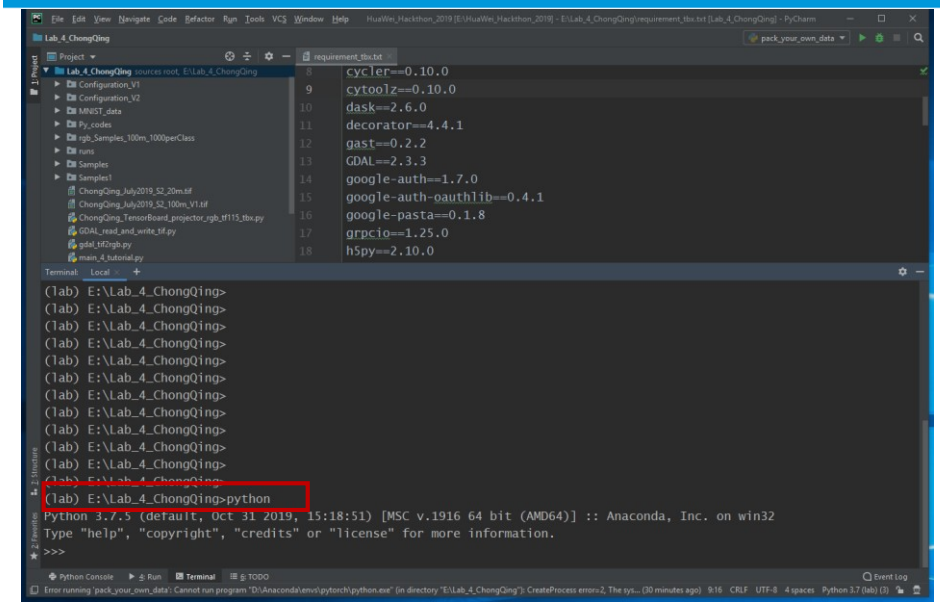
> pip install –r (Path to the file) requirements_tbx.txt

---------- Validation ----------

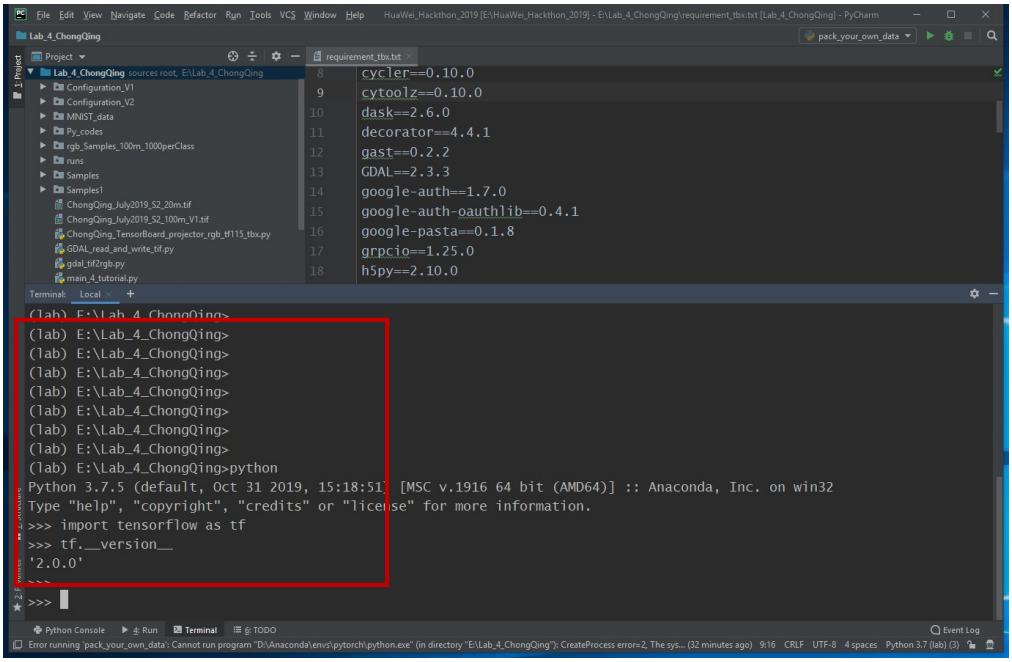
> python

(lab)

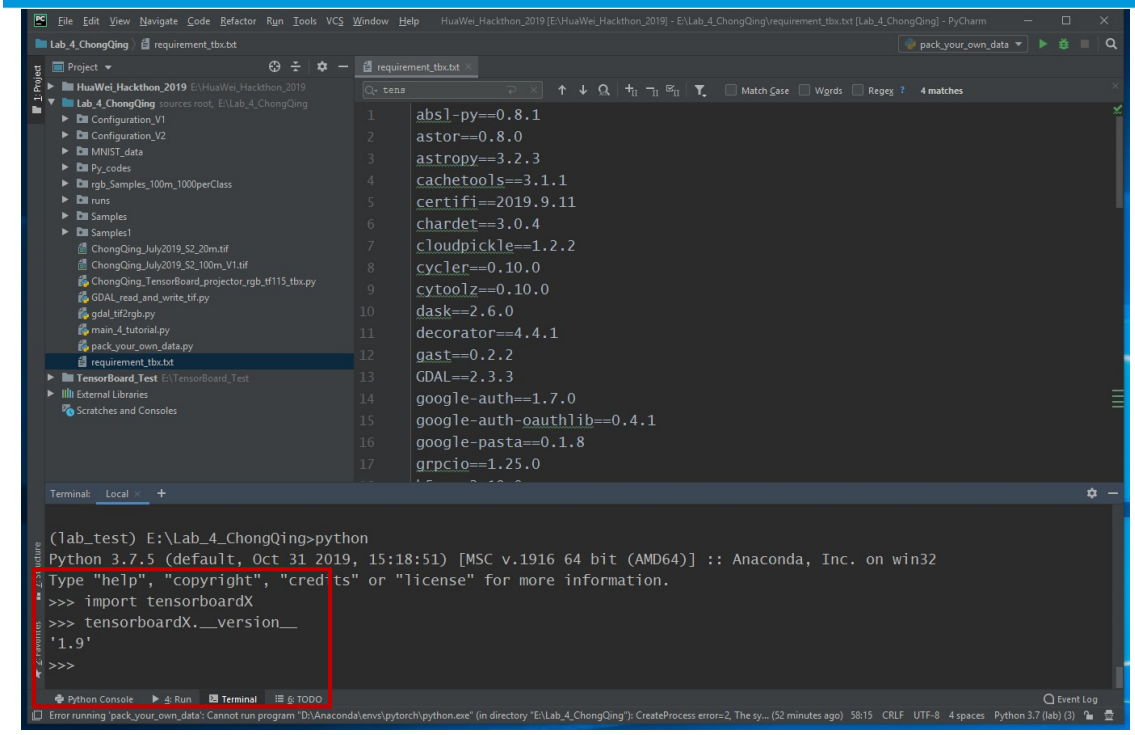
E:\Lab_4_ChongQing>python Python 3.7.5 (default, Oct 31 2019, 15:18:51) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32 Type "help", "copyright", "credits" or "license" for more information.




```
>>> import tensorflow as tf
>>> tf._version___
'2.0.0'
```




```
>>> import tensorboardX
>>> tensorboardX._version___
'1.9'
```



If there is any problem, please contact me: puzhao@kth.se