

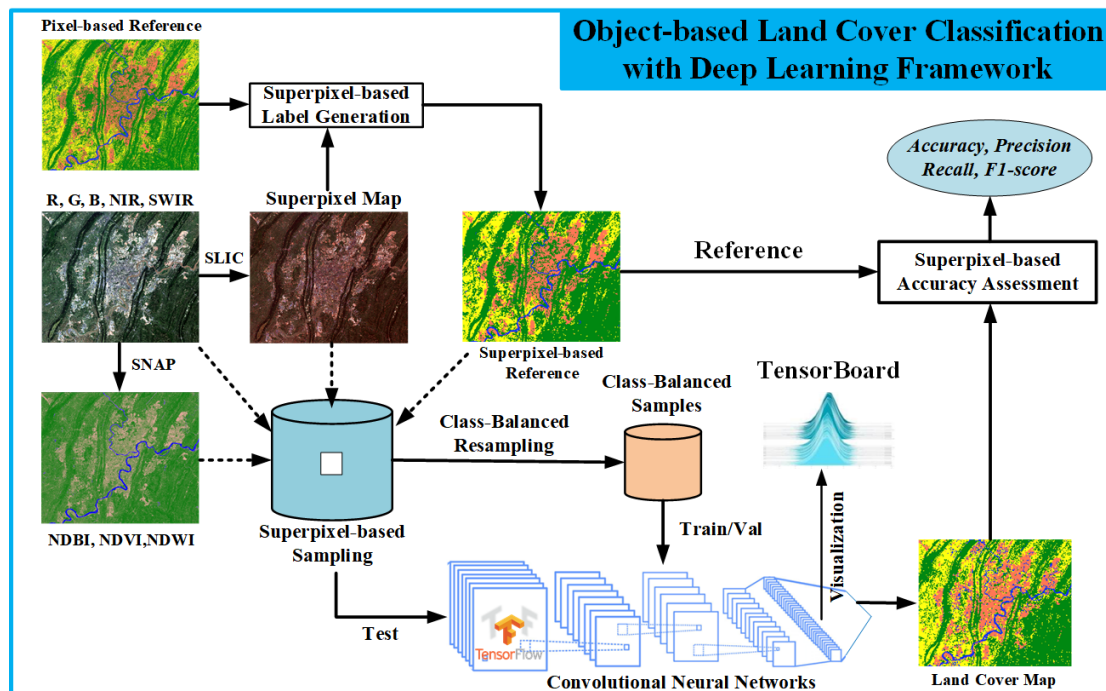
# Object-based Land Cover Classification with Deep Learning Framework

Yifang Ban, PhD, Professor  
KTH-Royal Institute of Technology

## Objective:

The purpose of this tutorial is to implement an object-based land cover classification using deep learning technique such as Convolutional Neural Networks (CNN). You will be introduced an image segmentation method named SLIC, and how to use Tensorflow to conduct CNN-based image classification and how to visualize data and network architecture with TensorBoard. At the completion of the tutorial, you will be able to classify remote sensing images with deep learning and conduct accuracy assessment.

(**Note:** It is recommended to run “Step0\_Data\_Preparation.py” in advance, because the data sampling procedure may take 25min.)

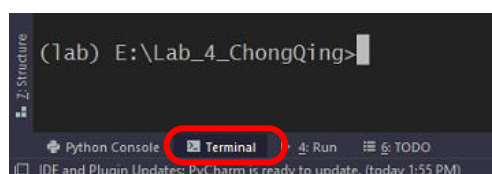


## 0. Check Environment and Run code “Step0\_Data\_Preparation.py”

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




(wechat group)



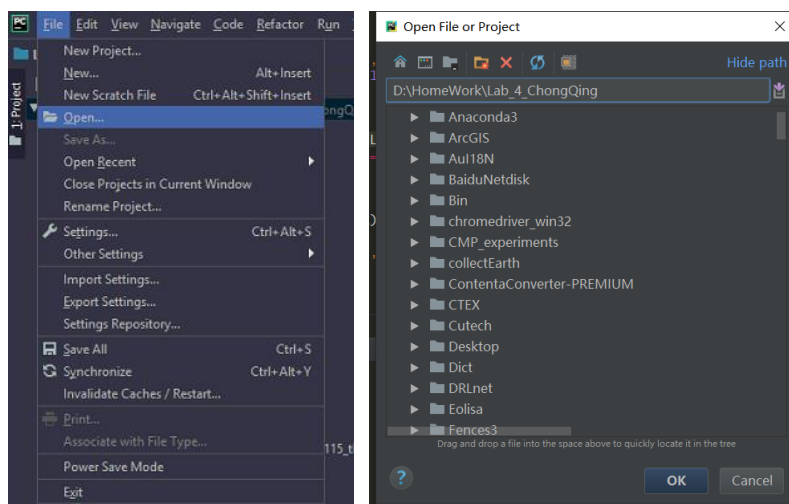
List all conda environment with “conda list env”, and then select the *lab* environment and activate it with “conda activate *lab*”, then check *tensorflow* version as follows.

```
(Tab) 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'
```

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

 Run 'Step0\_Data\_Preparati...' Ctrl+Shift+F10 by right clicking mouse, and the code can automatically obtain its current working directory with *os.getcwd()*.

Name	Date modified	Type	Size
ChongQing_LC_2017_20m.tif	11/8/2019 5:26 PM	TIFF image	2,607 KB
ChongQing_S2_2017_20m.tif	11/11/2019 12:54 ...	TIFF image	392,599 KB
gdal_tif2rgb.py	11/14/2019 8:22 AM	JetBrains PyChar...	5 KB
Step0_Data_Preparation.py	11/14/2019 8:36 AM	JetBrains PyChar...	5 KB
Step1_Class_Balance.py	11/14/2019 12:08 ...	JetBrains PyChar...	2 KB
Step2_Model_Training.py	11/14/2019 8:17 PM	JetBrains PyChar...	6 KB
step3_Accuracy_Assessment.py	11/14/2019 8:17 PM	JetBrains PyChar...	2 KB



## 1. Sentinel-2 Multispectral Data Visualization using Spectral Indexes

This are two image files in “.tif” format:

- (1) “ChongQing\_S2\_2017\_20m.tif” is the mosiaced Sentienl-2 image mosaic containing five bands (Red, Green, Blue, NIR and SWIR) and three spectral indexes including NDVI, NDWI and NDBI (see Table 1)

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

Table 1: Bands contained in “ChongQing\_S2\_2017\_20m.tif”

- (2) “ChongQing\_LC\_2017\_20m.tif” is the reference land cover types including urban, vegetation, crop ad water.

## 1.1 Check Sentinel-2 Data with SNAP

- 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).

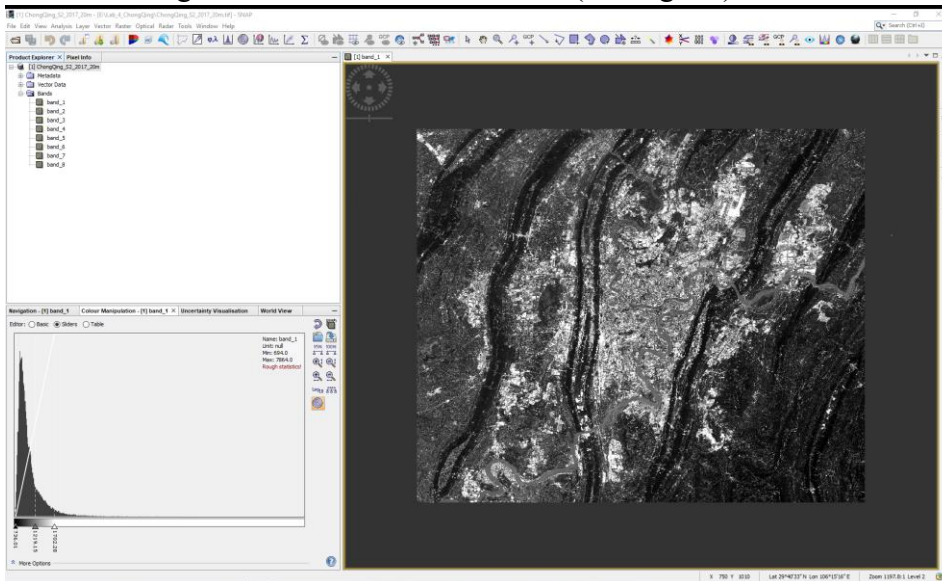


Fig 1.1 Visualize image band and histogram using SNAP

- 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, Geen, Blue] etc.

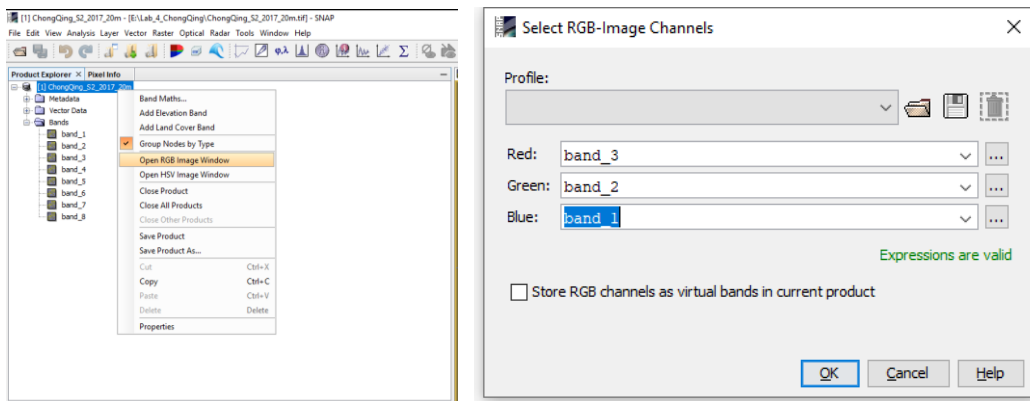


Fig 1.2 Open RGB composite

## 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)

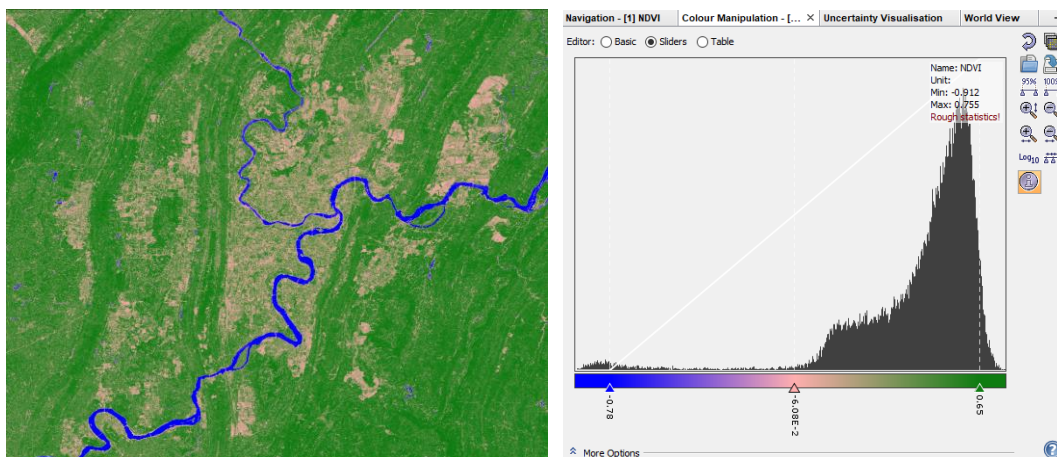
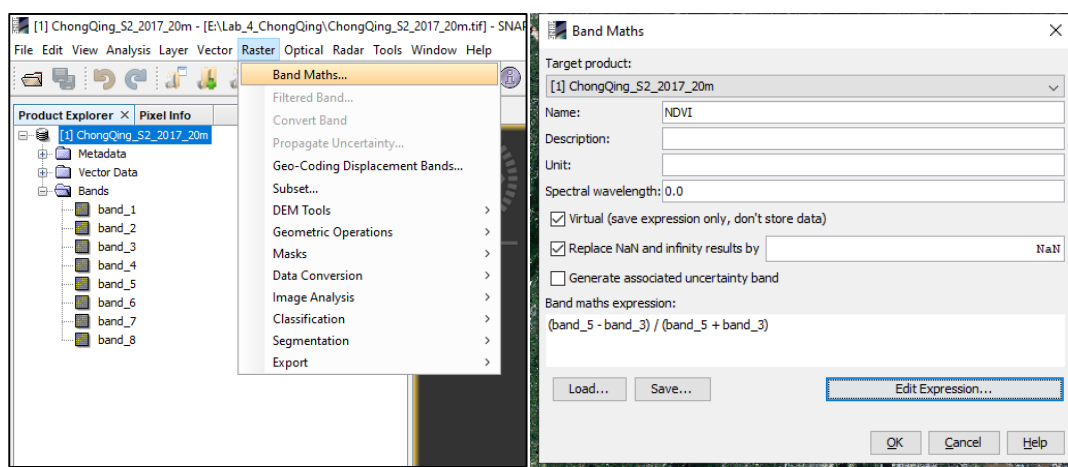
$$NDWI = (Green - SWIR)/(Green + SWIR)$$

- 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>)

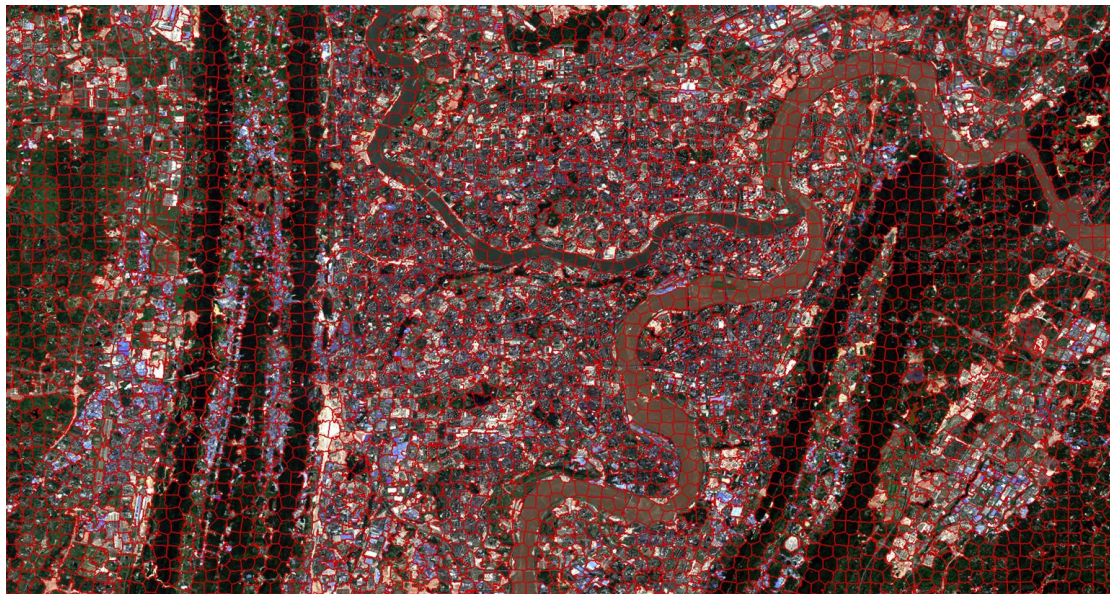


Fig. 2.1 Superpixel Segmentation Map

## 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 get familiar to the following material.)

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

- samples:  $n*w*w*c$

- labels:  $n*l$

## 2.3 Balance Training Data

Due to the fact that the number of training samples belongs to different classes is 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> ].

```

30 ''' ////////////////////////////////////////////////// Original Data Embedding //////////////////////////////////////'''
31 subImg = ChongQing['images'][..., [0, 1, 2]].transpose(0, 3, 1, 2)
32 print("subImg shape: ", subImg.shape)
33 n, c, w, h = subImg.shape
34
35 # featImg = subImg.reshape([-1, c*w*h])
36 # print(images.max())
37 # 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  $n*I$

-`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`.

```

22 ''' ////////////////////////////////////////////////// Read Class-Balanced Sample Set //////////////////////////////////////'''
23 ChongQing = spio.loadmat(dataPath + "Samples_classBalanced_1000.mat")
24 band4train = [2, 1, 0]_# bands used for training, 0-8: B, G, R, NIR, SWIR, NDVI, NDWI, NDBI
25 images = ChongQing['images'][..., band4train]
26 labels = ChongQing['labels'].transpose().squeeze()
27
28 print("images shape: {}".format(images.shape))
29 print("labels shape: {}".format(labels.shape))
30
31 ''' ////////////////////////////////////////////////// Original Data Embedding //////////////////////////////////////'''
32 band4projector = [2, 1, 0]_# bands used for projector, at most 3 bands, 0-8: B, G, R, NIR, SWIR, NDVI, NDWI, NDBI
33 subImg = ChongQing['images'][..., [2, 1, 0]].transpose(0, 3, 1, 2)
34 print("subImg shape: ", subImg.shape)
35 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.

```

57 """ ////////////////////////////////////////////////// CNN Model //////////////////////////////////////"""
58 model = tf.keras.Sequential([
59     layers.Conv2D(64, (4, 4), activation='relu', input_shape=(25, 25, 3)),
60     layers.MaxPooling2D((2, 2)),
61     layers.Conv2D(128, (3, 3), activation='relu'),
62     layers.MaxPooling2D((2, 2)),
63     layers.Conv2D(64, (3, 3), activation='relu'),
64     layers.Flatten(),
65     layers.Dense(256, activation='relu'),
66     layers.Dense(4, activation='softmax')
67 ])
68
69 model.summary()
70
71 model.compile(optimizer='adam',
72               loss=tf.keras.losses.SparseCategoricalCrossentropy(),
73               metrics=['accuracy'])
74

```

### 3.3 Model Training

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

```
78 log_dir="runs\\" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
79 tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
80
81 history = model.fit(train_ds,
82                     epochs=10,
83                     validation_data=val_ds,
84                     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 inputting the following code in Terminal (in red box).

```
> tensorboard --logdir runs
```

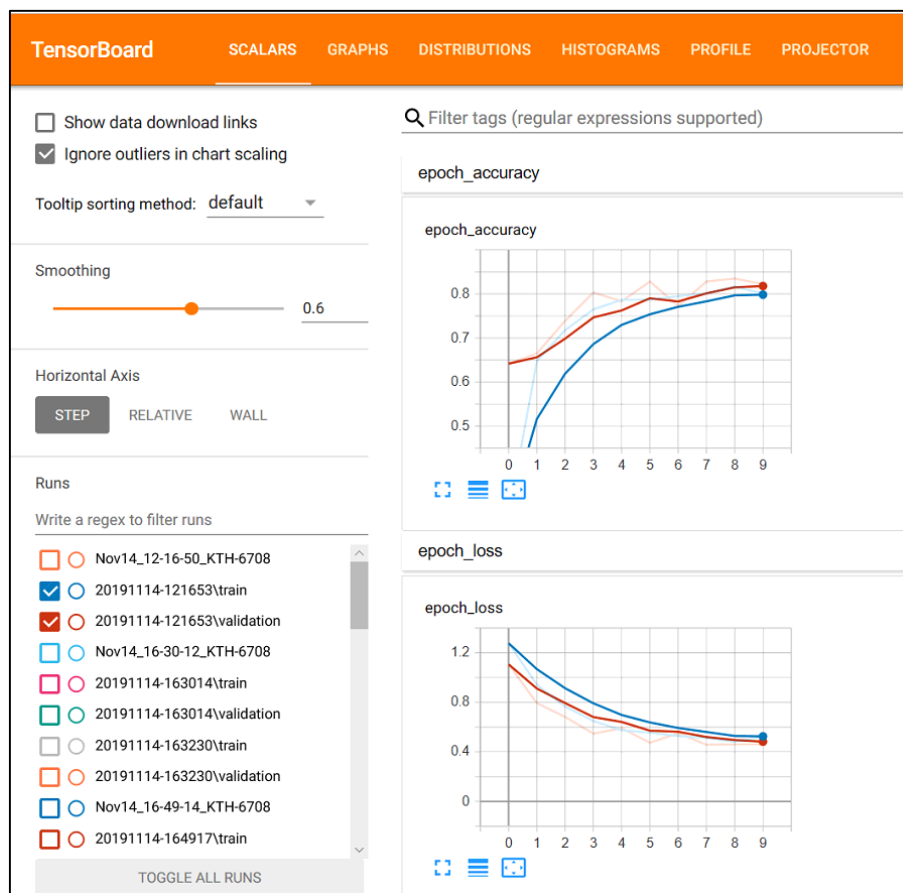
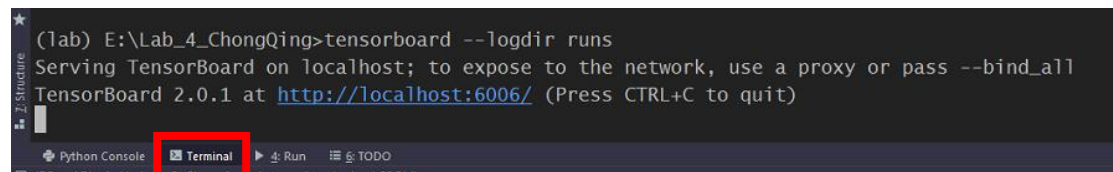


Fig 3.1 Learning Curve



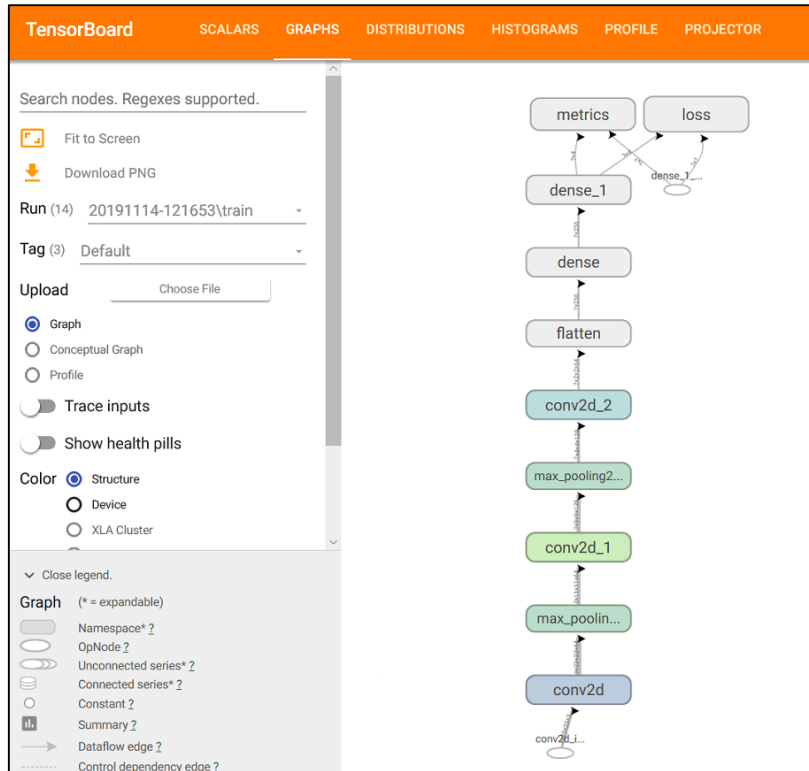


Fig 3.2 Network Architecture

Open the url [localhost://6006](http://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.

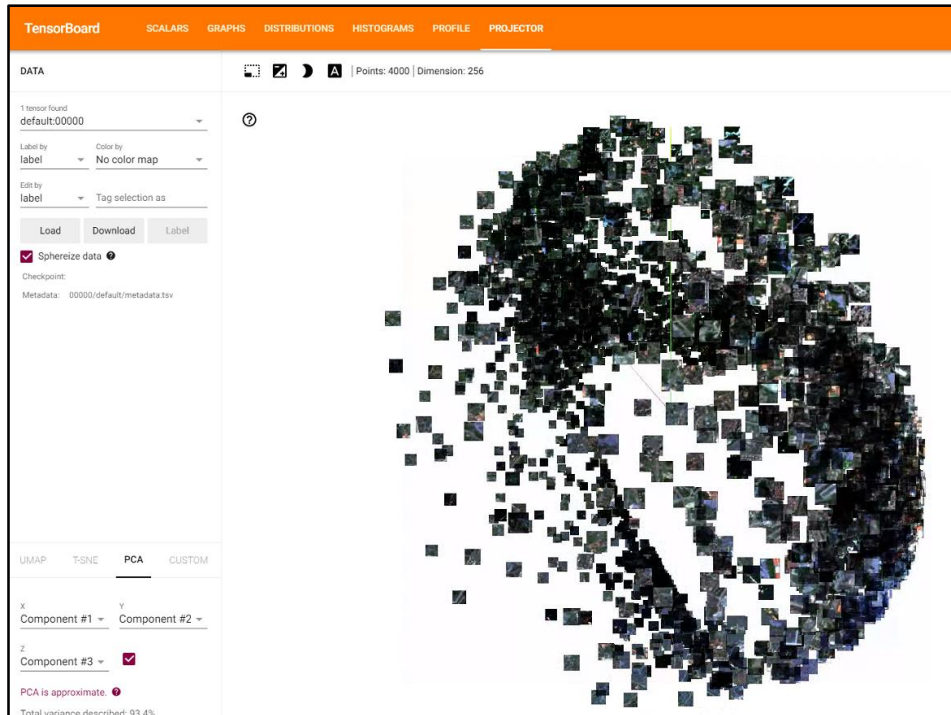


Fig 3.3 Data Visualization with Projector



## 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).

```
117     '''////////// Transform superpixel labels into a classified map //////////'''
118     start = time.perf_counter()
119     predMap = segments
120     for spLabel in np.unique(predLabels):
121         a = np.where(predLabels == spLabel)[0]
122         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 network architecture, optimizer, and epoch, how will the predicted map look?

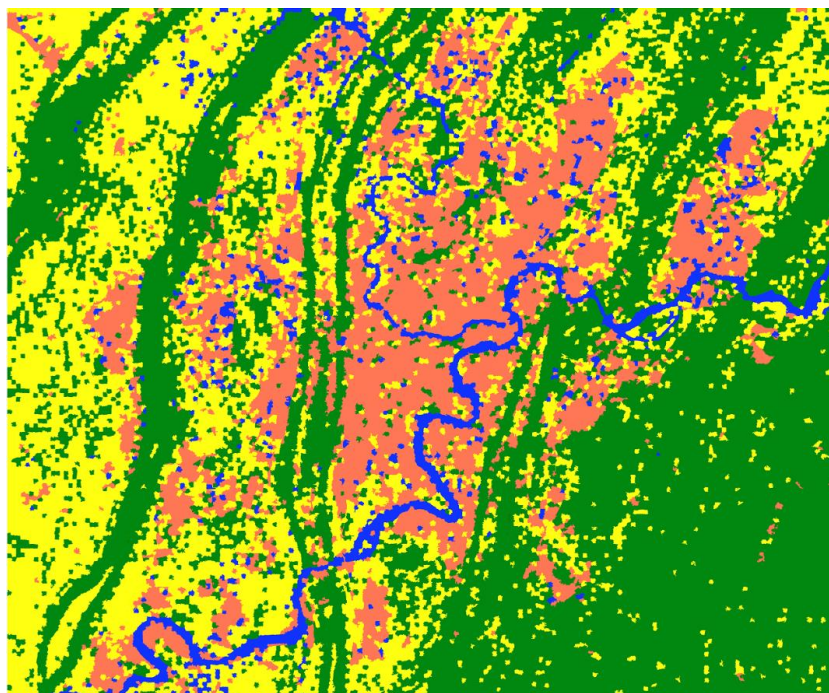


Fig 5.1 Map predicted by CNN (epoch=10)

## 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- $j$ .

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

Table 3. Four-Class Confusion Matrix.

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

$$\text{acc} = \frac{\sum_{k=0}^3 C_{kk}}{\sum_{i=0}^3 \sum_{j=0}^3 C_{ij}} \quad \text{precision}_k = \frac{C_{kk}}{\sum_{i=0}^3 C_{ik}} \quad \text{recall}_k = \frac{C_{kk}}{\sum_{j=0}^3 C_{kj}} \quad k=0,1,2,3.$$

$$[\text{class } k \text{ } F_1]: F1_k = \frac{2 * \text{Precision}_k * \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}$$

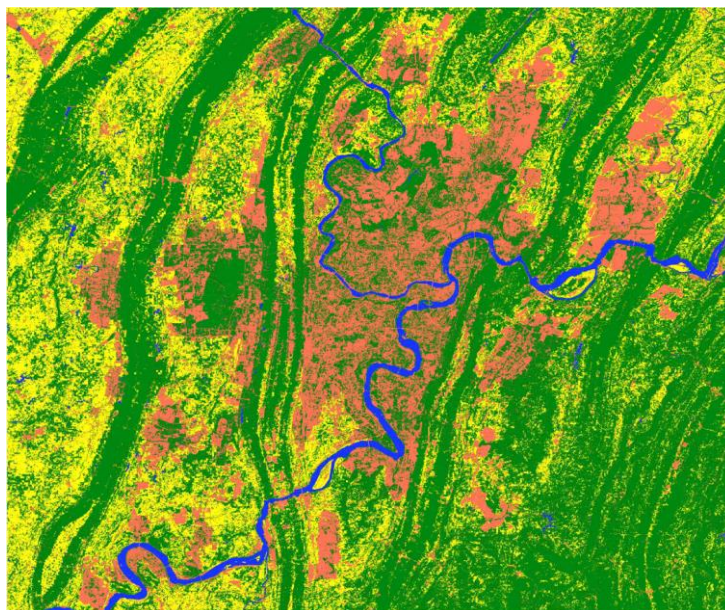


Fig 5.1 Pixel-based reference map



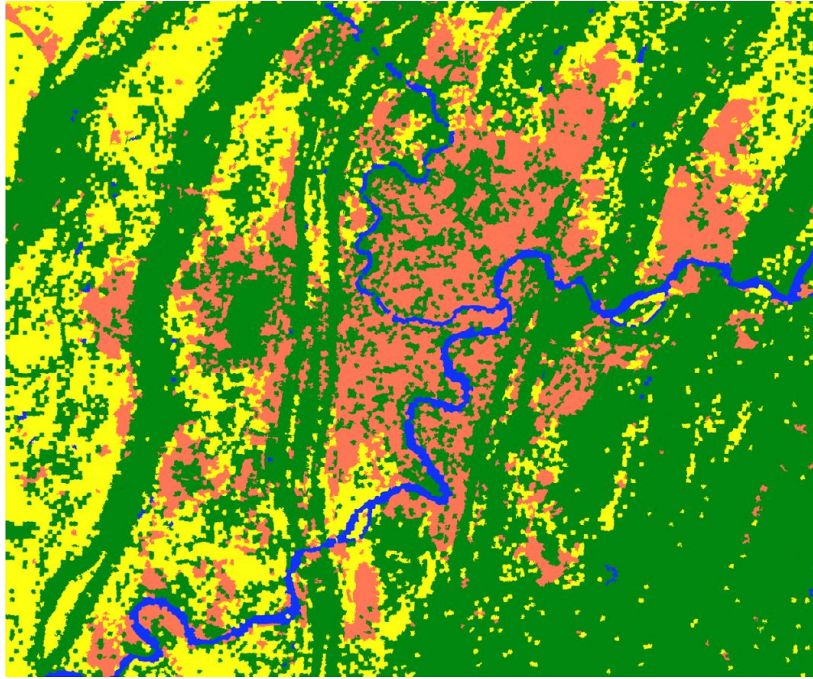


Fig 5.2 Superpixel-based reference map

```

47. ''' Accuracy Assessment '''
48. confMatrix = confusion_matrix(spPredLabels, spRefLabel)
49. acc_score = accuracy_score(spPredLabels, spRefLabel)
50. F1_score = f1_score(spRefLabel, spPredLabels, average='weighted')
51.

```

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

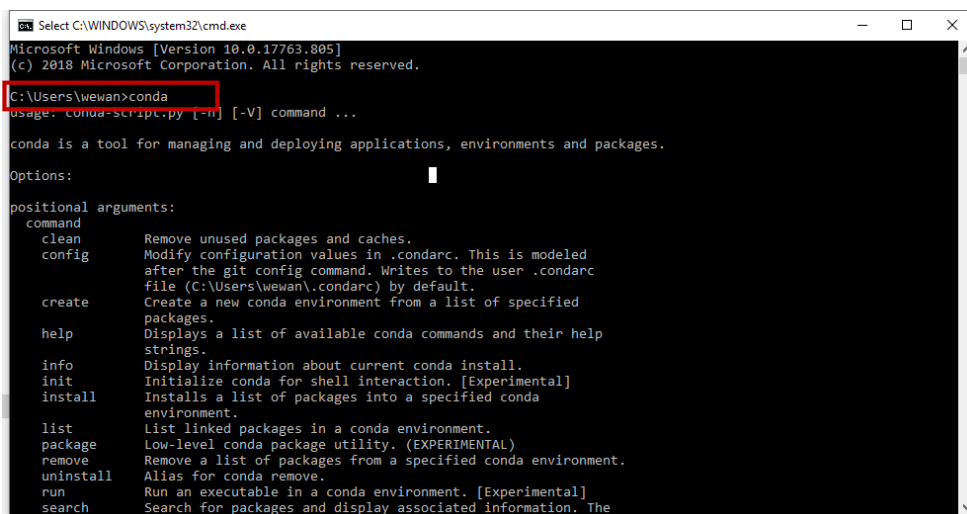
## Step 0: Install Anaconda and PyCharm

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).



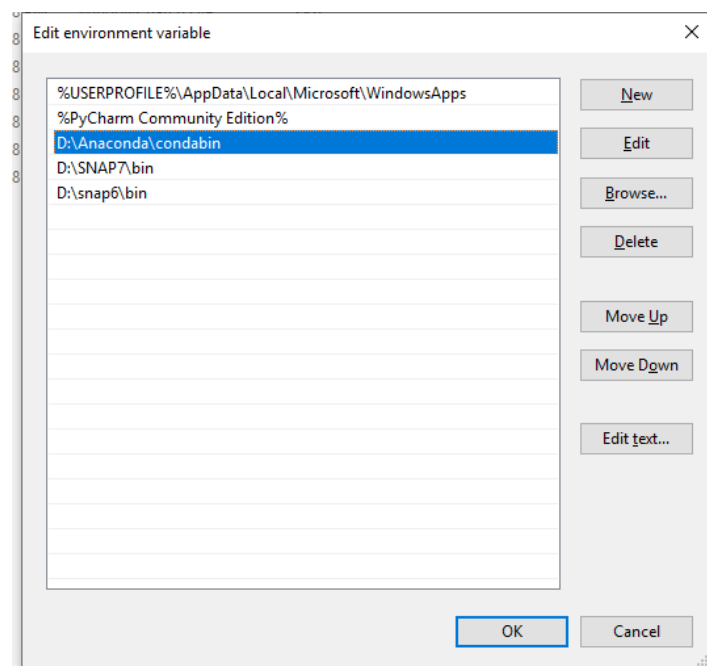
```
Select C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Wewan>conda
usage: conda-script.py [-h] [-V] command ...

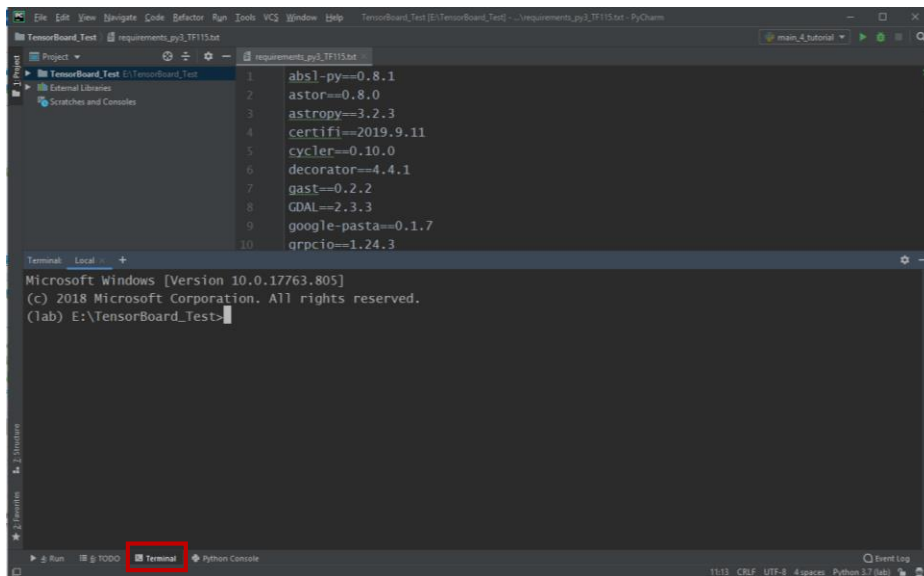
conda is a tool for managing and deploying applications, environments and packages.

Options:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit

positional arguments:
  command
  clean                Remove unused packages and caches.
  config               Modify configuration values in .condarc. This is modeled
                        after the git config command. Writes to the user .condarc
                        file (C:\Users\Wewan\.condarc) by default.
  create               Create a new conda environment from a list of specified
                        packages.
  help                 Displays a list of available conda commands and their help
                        strings.
  info                 Display information about current conda install.
  init                 Initialize conda for shell interaction. [Experimental]
  install              Installs a list of packages into a specified conda
                        environment.
  list                 List linked packages in a conda environment.
  package              Low-level conda package utility. (EXPERIMENTAL)
  remove               Remove a list of packages from a specified conda environment.
  uninstall            Alias for conda remove.
  run                  Run an executable in a conda environment. [Experimental]
  search               Search for packages and display associated information. The
```







### 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
```

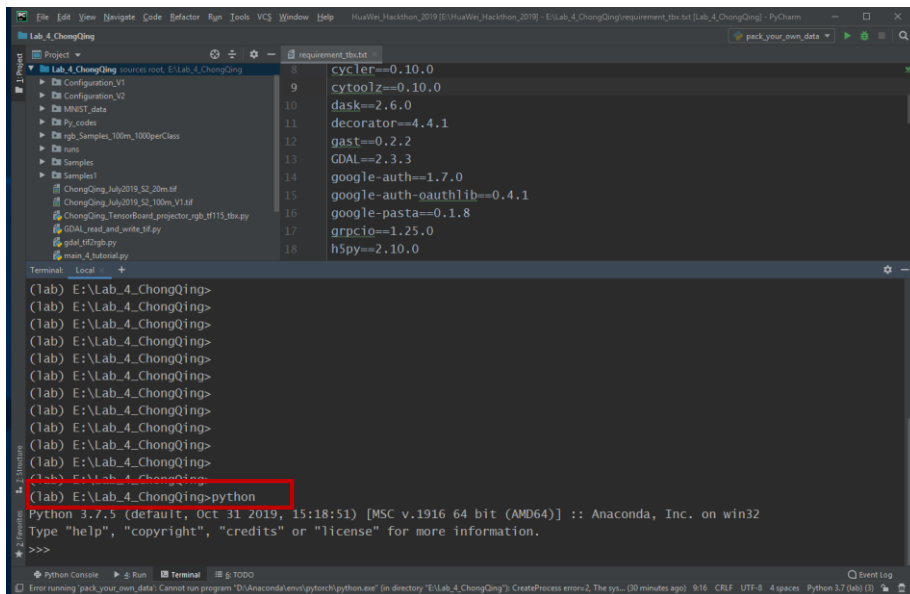
----- Validation -----

```
> 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
```

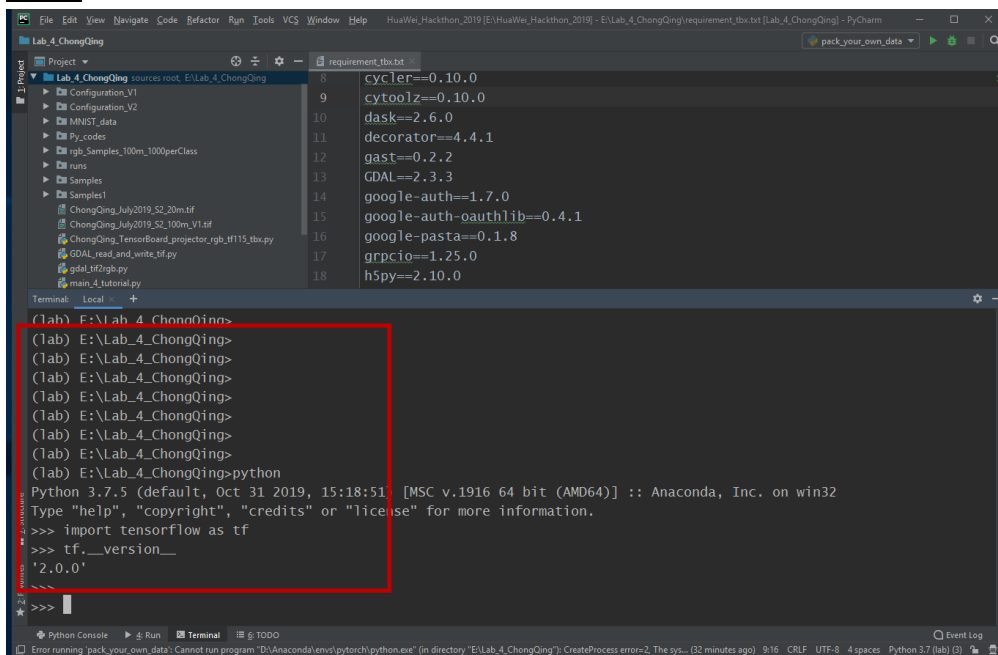
```
Type "help", "copyright", "credits" or "license" for more information.
```



```
>>> import tensorflow as tf
```

```
>>> tf.__version__
```

```
'2.0.0'
```



```
>>> import tensorboardX
```

```
>>> tensorboardX.__version__
```

```
'1.9'
```

The image shows a PyCharm IDE window with a project named 'Lab\_4\_ChongQing'. The main editor displays a file named 'requirements.txt' with the following content:

```
1  absl-py==0.8.1
2  astor==0.8.0
3  astropy==3.2.3
4  cachetools==3.1.1
5  certifi==2019.9.11
6  chardet==3.0.4
7  cloudpickle==1.2.2
8  cycloper==0.10.0
9  cytoolz==0.10.0
10 dask==2.6.0
11 decorator==4.4.1
12 gast==0.2.2
13 GDAL==2.3.3
14 google-auth==1.7.0
15 google-auth-oauthlib==0.4.1
16 google-pasta==0.1.8
17 grpcio==1.25.0
```

Below the editor is a terminal window with the following output:

```
(lab_test) 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 tensorboardX
>>> tensorboardX.__version__
'1.9'
>>>
```

An error message is visible at the bottom of the terminal window: "Error running 'pack\_your\_own\_data': Cannot run program 'D:\Anaconda\envs\pytorch\python.exe' (in directory 'E:\Lab\_4\_ChongQing'): CreateProcess error=2, The system cannot find the file specified. (52 minutes ago) 58:15 CRLF UTF-8 4 spaces Python 3.7 (lab) (3)".

If there is any problem, please contact me: [puzhao@kth.se](mailto:puzhao@kth.se)