

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".





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. Doubleclick "*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	Туре	Size
ChongQing_LC_2017_20m.tif	11/8/2019 5:26 PM	TIFF image	2,607 KE
ChongQing_S2_2017_20m.tif	11/11/2019 12:54	TIFF image	392,599 KE
gdal_tif2rgb.py	11/14/2019 8:22 AM	JetBrains PyChar	5 KE
Step0_Data_Preparation.py	11/14/2019 8:36 AM	JetBrains PyChar	5 KE
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 KE
step3_Accuracy_Asssessment.py	11/14/2019 8:17 PM	JetBrains PyChar	2 KE





1. Sentinel-2 Multispectral Data Visualization using Spectral Indices

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

 "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_	Band_7	Band_8
					6		
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

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.



oduct Explorer × Pixel II	nfo			-][
Il ChongQing_S2_201 Metadata Vector Data Bands band_1 band_2	7_20m Band Math Add Elevati Add Land C	s on Band Cover Band es by Type		
band_3	Open RGB I	image Window		
band_4	Open HSV I	image Window		
band_5	Close Prod Close All Pr Close Othe	Close Product Close All Products Close Other Products		
	Save Produ Save Produ	ct ct As		
	Cut	Ctrl+X		
	Copy	Ctrl+C		
	Paste	Ctrl+V		
	Delete	Delete		
	Properties			

🛃 Selec	t RGB-Image Channels	×
Profile:		
	~ (=	
Red:	band_3	×
Green:	band_2	×
Blue:	band_1	×
	Expres	ssions are valid
Stor	e RGB channels as virtual bands in current product	
	OK Cano	el Help



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)

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.



[1] ChongQing_S2_2017_20m - [E:\Lab	_4_ChongQing\ChongQing_S2_2017_20m.	tif] - SNAF	🚪 📕 Band Maths		×
File Edit View Analysis Layer Vector	Raster Optical Radar Tools Window H Band Maths	Help	Target product:		
	Filtered Band	3	[1] ChongQing_S2_3	2017_20m	~
Product Explorer × Pixel Info	Convert Band		Name:	NDVI	
[] ChongQing_S2_2017_20m	Propagate Uncertainty		Description:		
Metadata Vector Data	Geo-Coding Displacement Bands	11	Unit:		
Bands	Subset		Spectral wavelength:	0.0	
band_1	DEM Tools	>	Virtual (save exc	ression only, don't store dat	a)
band_2	Geometric Operations	>			-,
band_3	Masks	>	Replace NaN and	l infinity results by	NaN
band 5	Data Conversion	>	Generate associa	ated uncertainty band	
band_6	Image Analysis	>	Band maths expression	on:	
band_7	Classification	>	(band 5 - band 3) /	(band 5 + band 3)	
band_8	Segmentation	>		/	
	Export	>			
			Load Si	ave	Edit Expression
					OK Cancel Help



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,

- **c***ompactness* 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:<u>https://scikitimage.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.slic</u>).





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***l*

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 tdistributed stochastic neighbor embedding (t-SNE)

[TensorboardX : <u>https://tensorboardx.readthedocs.io/en/latest/tensorboard.html</u>].



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*.



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 <u>https://tensorflow.google.cn/</u> if more details are needed. Choose an optimizer and loss function for your model.

	""" //////////////////////////////////
	model = tf.keras.Sequential([
	<pre>layers.Conv2D(64, (4, 4), activation='relu', input_shape=(25, 25, 3)),</pre>
	layers.MaxPooling2D((2, 2)),
	<pre>layers.Conv2D(128, (3, 3), activation='relu'),</pre>
	layers.MaxPooling2D((2, 2)),
	<pre>layers.Conv2D(64, (3, 3), activation='relu'),</pre>
	layers.Flatten(),
	layers.Dense(256, activation='relu'),
	layers.Dense(4, activation='softmax')
	model.summary()
	<pre>model.compile(optimizer='adam',</pre>
	<pre>loss=tf.keras.losses.SparseCategoricalCrossentropy(),</pre>
73	metrics=['accuracy']



3.3 Model Training

Configure parameters for your model fitting such as epoch and 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



TensorBoard	SCALARS	GRAPHS	DISTRIBUTIONS	HISTOGRAMS	PROFILE	PROJECTOR
Show data download	d links art scaling		Q Filter tags (reg	ular expressions	supported)	
Tooltip sorting method:	default 👻	-	epoch_accuracy			
Smoothing	0.	.6	0.8		_	
Horizontal Axis STEP RELATIVE	WALL		0.6			
Runs Write a regex to filter runs			0 1	2345	6789	
Nov14_12-16-50_K	(TH-6708	^	epoch_loss			
 20191114-121653 20191114-121653 20191114-121653 	\train		epoch_loss			
Nov14_16-30-12_K	(TH-6708		1.2			
 20191114-163014 20191114-163014 	\train		0.8			
20191114-163230	\train		0.4			
20191114-163230	Validation		0			
20191114-164917	\train		0 1	2 3 4 5	6789	
TOGGLE AL	L RUNS	~	0 🔳 🖸			

Fig 3.1 Learning Curve



TensorBoard	SCALARS	GRAPHS	DISTRIBUTIONS	HISTOGRAMS	PROFILE	PROJECTOR
Search nodes. Regexes s	supported.	^				
Fit to Screen		_		metri	cs	loss
				214	AS NO.	30
Download Find				dense	_1	se_1
Run (14) 20191114-121	653\train			1020		
Tag (3) Default		-		dens	e	
Upload Choose	se File					
Graph				flatte	n	
O Conceptual Graph						
O Profile				0001		
Trace inputs				conv2c	1_2	
Show health pills		- 11		2 rate day		
Color Structure				(max_pooli	ng2	
O Device				2. Particular		
-		~		conv2c	J_1	
✓ Close legend.				- Antisti		
Namespace* ?				max_poc	olin	
OpNode ?	2			-		
Connected series* ?	L			conv2	2d	
Constant ?				1		
Dataflow edge ?	dao 2			conv2d_i		

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).

```
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 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 superpixelbased 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: <u>http://lijiancheng0614.github.io/scikit-learn/modules/classes.html#module-sklearn.metrics</u> *C ij: number of superpixels belong to class-i are classified into class-j.*

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

Table 5. Four-Class Confusion Matrix	Table 3.	Four-Class	Confusion	Matrix.
--------------------------------------	----------	------------	-----------	---------

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

$$acc = \frac{\sum_{k=0}^{3} C_k k}{\sum_{i=0}^{3} \sum_{j=0}^{3} C_i i j} \text{ precision}_k = \frac{C_k k}{\sum_{i=0}^{3} C_i i k} \text{ recall}_k = \frac{C_k k}{\sum_{j=0}^{3} C_k j} \text{ k=0,1,2,3.}$$

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





Fig 5.1 Pixel-based reference map



Fig 5.2 Superpixel-based reference map





		Contr	ISTON MA	Cr1x
L	ırban	crot	o veg	water
11	4938	682	2974	56]
Ι	117	3779	209	12]
Γ	1377	56	13524	53]
Γ	54	240	42	606]]
	uracy	Score		
act	uracy	30016	. 0.00	
F1-	score	: 0.8	30	

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) <u>https://www.anaconda.com/distribution/#windows</u> PyCharm-Community <u>https://www.jetbrains.com/pycharm/download/#section=windows</u>

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).



dit environment variable	×
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps	New
D:\Anaconda\condabin	Edit
D:\SNAP7\bin	
D:\snap6\bin	<u>B</u> rowse
	Delete
	Move <u>U</u> p
	Move D <u>o</u> wn
	Edit <u>t</u> ext
OF	Cancel

Point Dig Control Protection Statestand

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.



📓 Eile Edit View Navigate Code Befactor Ryn Tools VCS				
Lab_4_ChongQing				
g ■ Roject + C → C → C → C → C → C → C → C → C → C	3 cycler=0.10.0 9 cyclolz=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==1.7.0 16 google-pasta=-0.1.8 17 grpcig=-1.25.0			
<pre>B main_Listenday Terminal Local + (lab) E:\Lab_4_ChongQing> (lab)</pre>), 15:18:51) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32 " or "license" for more information.			
Python Console		4 spaces Python 3.7 (lab)	svent Log) (3) 🔒	

>>> import tensorflow as tf



>>> import tensorboardX







Eile Edit View Navigate Code Refactor Run Tools VCS			
🖿 Lab_4_ChongQing 👌 🛃 requirement_tbx.txt			
g ■ Project ▼ 😌 🛧 🗢 –	☐ requirement_tbx.bt ×		
 HuwWei, Hackthon, 2019 E (MarWei, Hackthon, 2019 HuwWei, Hackthon, 2019 E (MarWei, Hackthon, 2019 Di Configuration, V1 Di Configuration, V2 Di MVIS data Di Py, codes Di Ry, smples, 100m, 1000 perClass Di runs Di Samples Consg\u00f3ing, UM2019, 52, 20m.tf Chong\u00f3ing, TensorBoard, projector, rgb, ff115, tbx.py GoAL, read, and, write, tripy GoAL, read, and, tripy GoAL, read, and tripy GoAL, read, tripy GoAL, read, tripy	Q • tens P ★ Q + 1 □ □ ∞ 1 T. Match ⊆sic Wgrds Regrg ? 4 matches 1 abs]-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 cycler=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-pasta==0.1.8 17 grpcio==1.25.0		× ×
		\$	* -
<pre>(lab_test) E:\Lab_4_ChongQing>pytl Python 3.7.5 (default, Oct 31 2019 Type "help", "copyright", "cred t: >>> import tensorboardX >>> tensorboardXversion '1.9' >>></pre>	ion 9, 15:18:51) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32 " or "license" for more information.		
Python Console ► 4: Run 🛛 Terminal 🗮 6: TODO	oda)envs\pytorch\python.exe" (in directory "E\Lab 4 ChonaQina"): CreateProcess error=2. The sy (52 minutes ago) 58:15 CRLF UTF-8 4 spaces Python 3.7 (II)Event Loi ab) (3)) =

If there is any problem, please contact me: <u>puzhao@kth.se</u>