Micro-module 1: Street view images (SVI) and urban analytics

Understanding the qualities of the built environment is crucial for a wide range of disciplines, including public health research, spatial social studies, real-estate and infrastructure development and urban planning. Street View Imagery (SVI) has gained a strong momentum in urban studies in the last few years, propelled by the proliferation of SVI data, advances in machine learning to extract a variety of information, and the growing computing power to facilitate processing large amounts of data.

This module will cover the whole process from street view data collecting, semantic segmentation processing, geographical visualization using google street view images as an example.

1. Street View Images and Application in Urban Analystics

- **Street view images**

Compared with tradition field observation methods and aerial data collected by remote sensing, street view images have the following advantages:

(1) large coverage thanks to omnipresent map service providers; (2) relatively homogeneous quality, sampling, and resolution; (3) free and efficient access to the data; (4) reliable and rich metadata; and (5) capture of the urban scenery from a human perspective.

(a) Aerial perspective. (b) Street-level point of view.

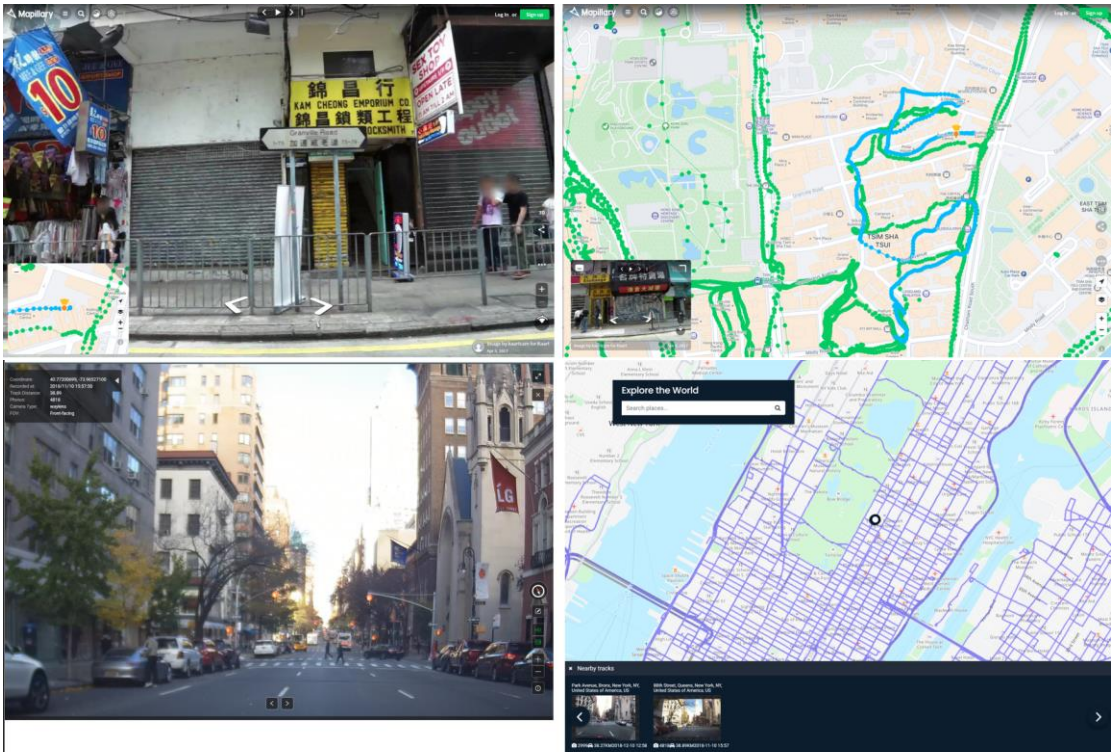
- **Platforms for collecting street view images**

Google Street View Google Street View (GSV) is arguably the most well-known and widespread service providing SVI. Barring rare exceptions such as backpack-mounted cameras to survey narrow roads, the panoramic imagery is acquired in a standardised manner: from a car mounted with multiple cameras on its roof, accompanied with various sensors including lidar.

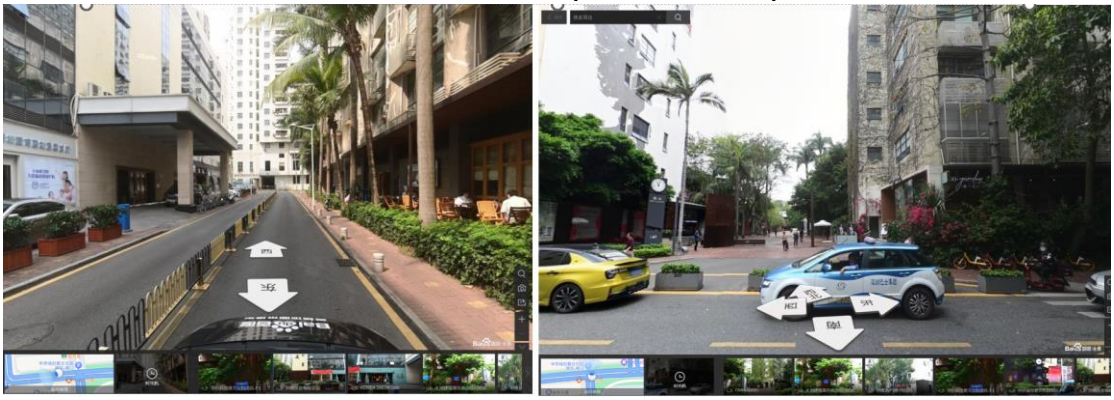
Mapillary and KartaVieware the remaining two services with a global focus. They both rely on crowdsourced imagery and are owned and operated by commercial entities. Anyone can contribute to Mapillary and KartaView.

Pros: volunteered, high temporal resolution; views from pavements, cycle tracks and walkways;

Cons: not panoramic, not standard, quality of images (resolution of pixels).



- **Tencent Street View and Baidu Total View (Mainland China)**

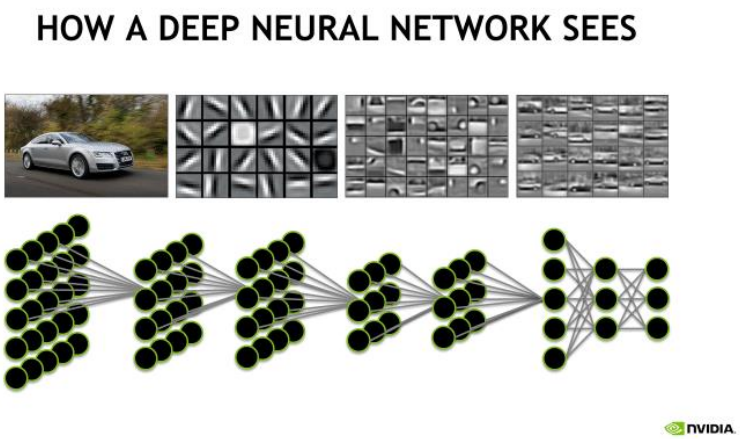


Baidu Maps is a web mapping service provided by Baidu, which can be considered as the counterpart of Google Maps for China. Since 2013 it offers a street view service — Baidu Total View. While the coverage of satellite imagery and maps in Baidu Maps spans beyond China, SVI is available only for China. Tencent Maps is a similar local service, provided by Tencent, and since 2011 it has been offering SVI under Tencent Street View.

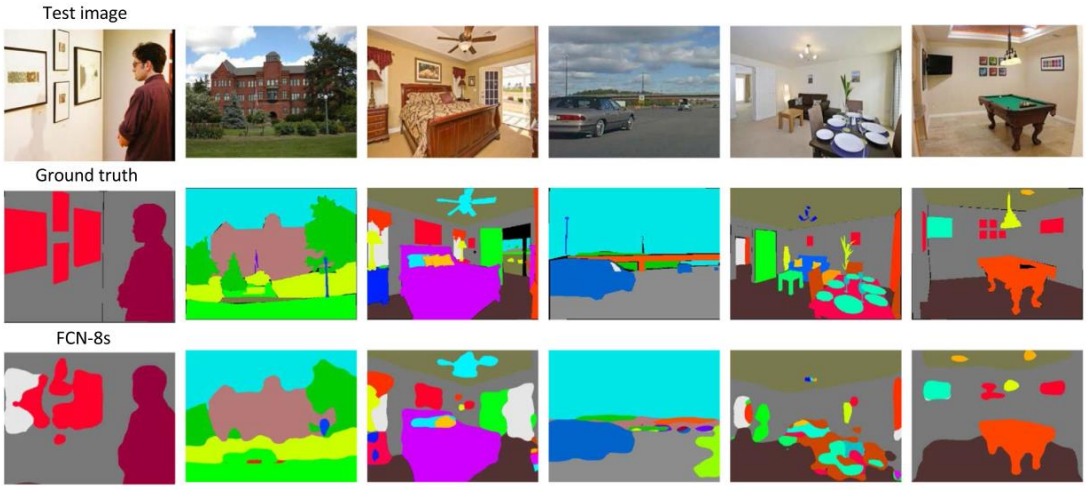
- **Application of Street View Images in Urban Analytics**

In general, street view images can be used in research about Spatial data infrastructure, Greenery, Health and well-being, Urban morphology, Transportation and mobility, Walkability, Real estate, and Urban perception.

2. Computer Vision Techniques



Computer vision analyzes images, and then creates numerical representations of what it 'sees' using a convolutional neural network (CNN). A CNN is a class of artificial neural network that uses convolutional layers to filter inputs for useful information. The convolution operation involves combining input data (feature map) with a convolution kernel (filter) to form a transformed feature map.



In this work, we present a densely annotated dataset ADE20K, which spans diverse annotations of scenes, objects, parts of objects, and in some cases even parts of parts. This dataset has both outdoor and indoor scenes, and it is typically used by CV communities as training and validation set to check their proposed DL models.

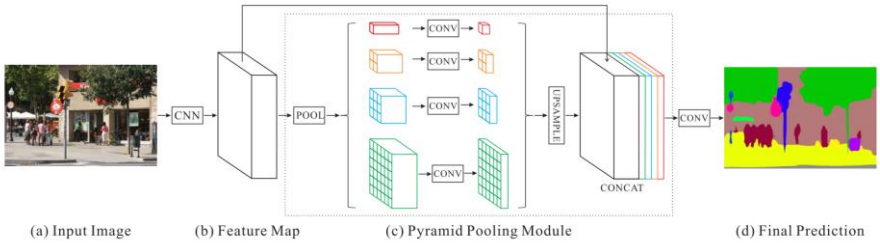


Figure 3. Overview of our proposed PSPNet. Given an input image (a), we first use CNN to get the feature map of the last convolutional layer (b), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (c). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (d).

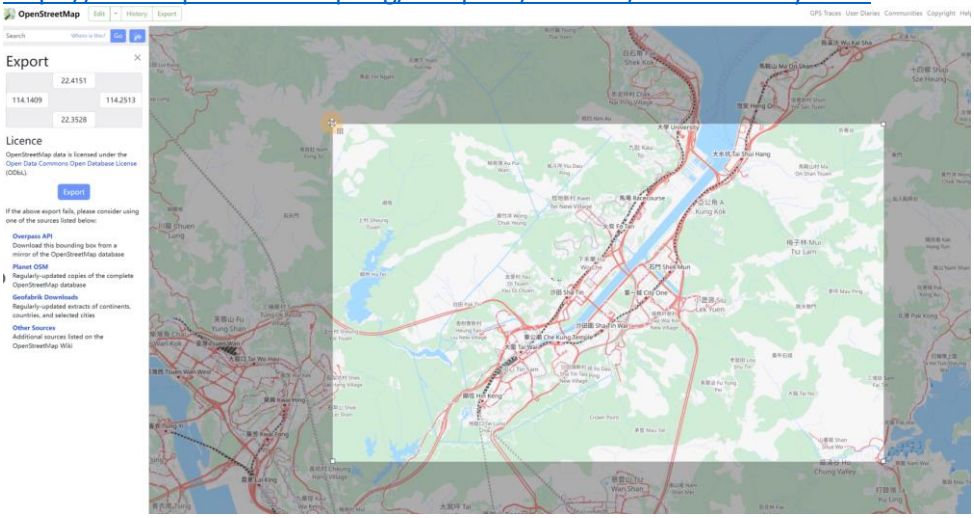
PSPNet, the Pyramid Scene Parsing Network makes great contribution to the improvement of CV. It adds a unique Pyramidal Pooling Module to the algorithm, and was able to improve the prediction accuracy compared to previous Fully Connected

Convolutional Models. And it is relatively fast, and considers the context of each pixels for better semantic segmentation. As you could see in this image above to the right, it is able to differentiate the pillows from the quilts, although it has a similar texture and color. Therefore, this algorithm has been used in many emerging studies.

3. **Data Preprocessing**

- **download OSM data**

  You can download OSM data via https://www.openstreetmap.org/#map=15/22.3311/114.1585&layers=T.



- **import OSM data**

  open QuickOSM plugin, choose import from file, choose the path of your osm file, select all features and click OK.

  For this case, we only use the polyline layers. You can remove other layers.

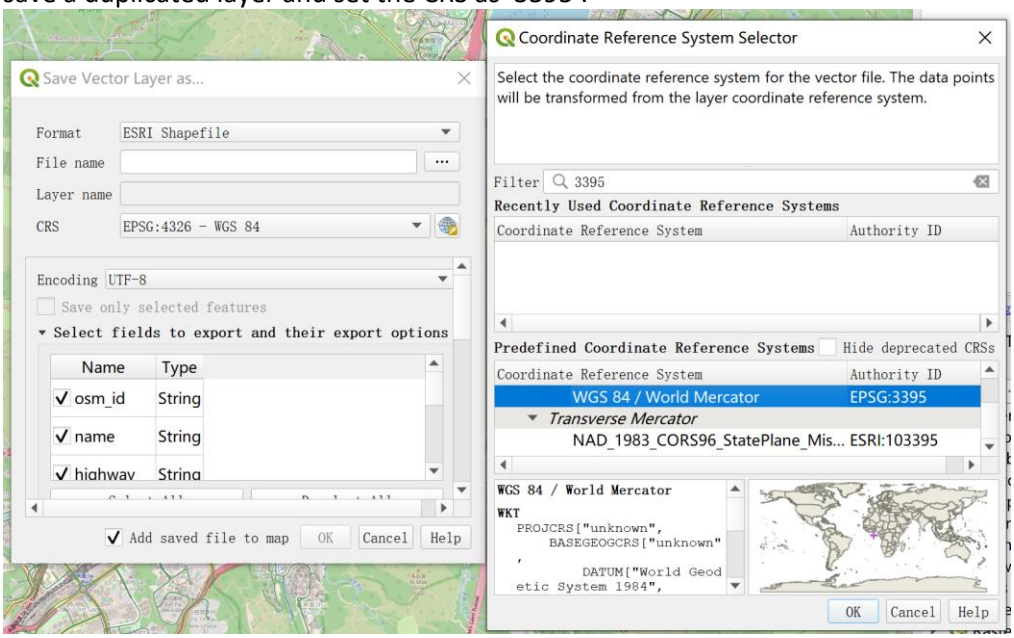
- **Clean the road network**

  Use filter (right click the layer to active the menu), and you can copy paster the following syntax:

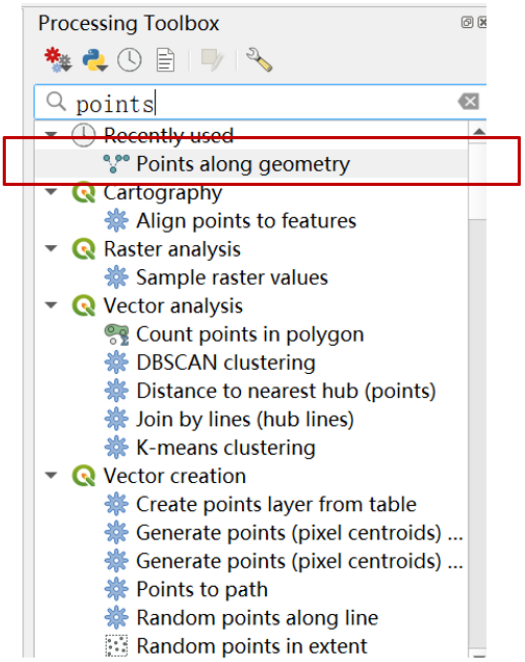
  "highway" = 'primary' OR "highway" = 'residential' OR "highway" = 'secondary' OR "highway" = 'tertiary' OR "highway" = 'tertiary_link' OR "highway" = 'trunk' OR "highway" = 'trunk_link' OR "highway" = 'unclassified'

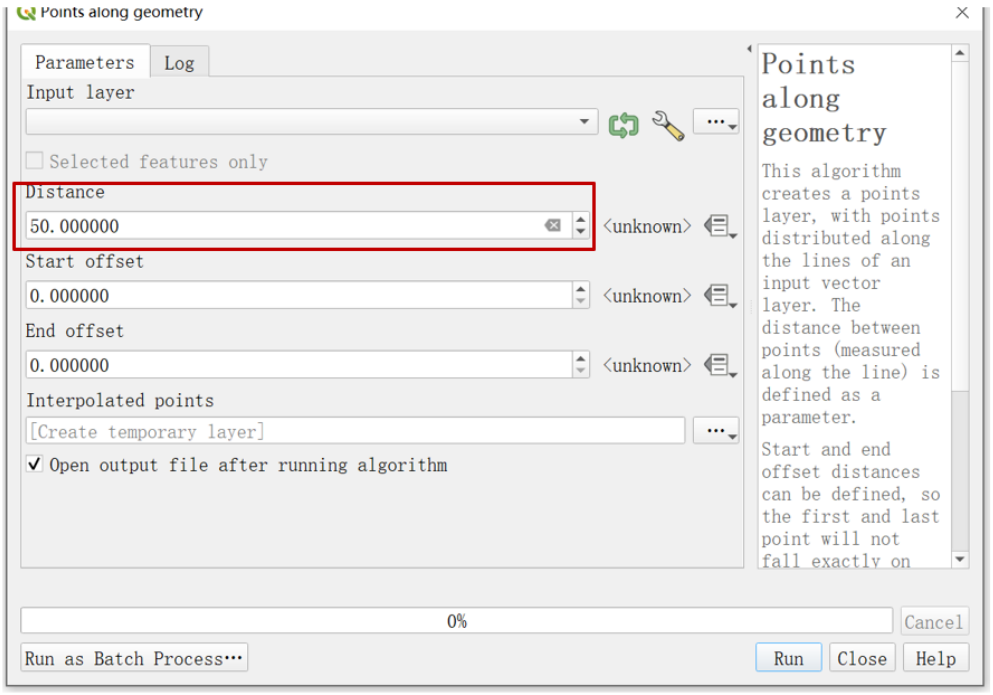
- **Create Sampling Points**

We need to make sure the unit of the layers is meter (not degree, instead). Here we save a duplicated layer and set the CRS as '3395'.

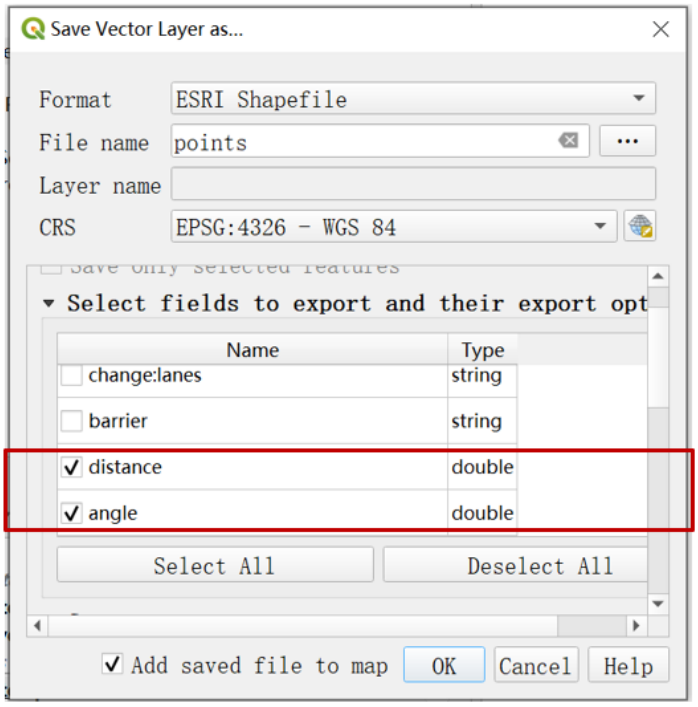


Then we use the function 'Points along geometry'. You can set your own interval. We usually use 50, 100, 200 meters as the interval.

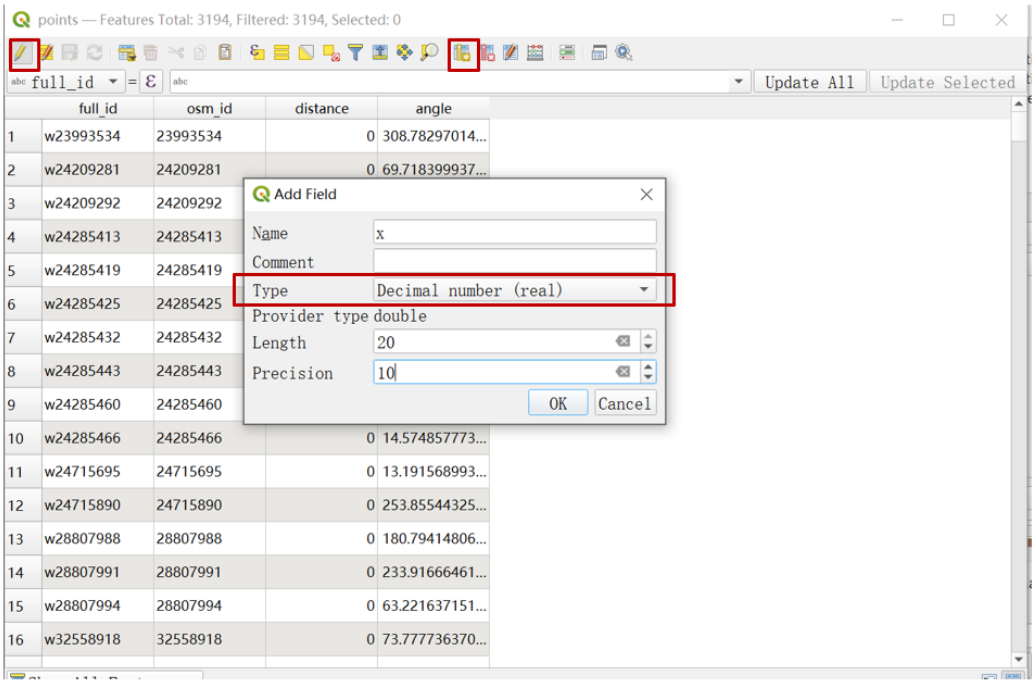




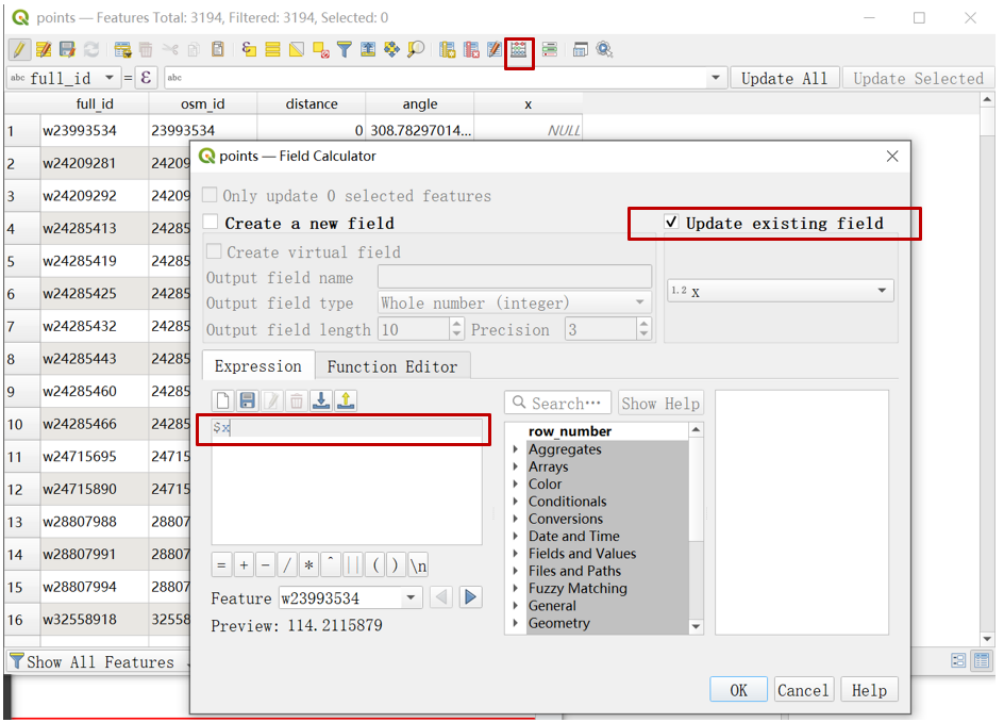
- **Relocated the point layer**
  We may need to save the point layer, and relocated it with the CRS 4326.



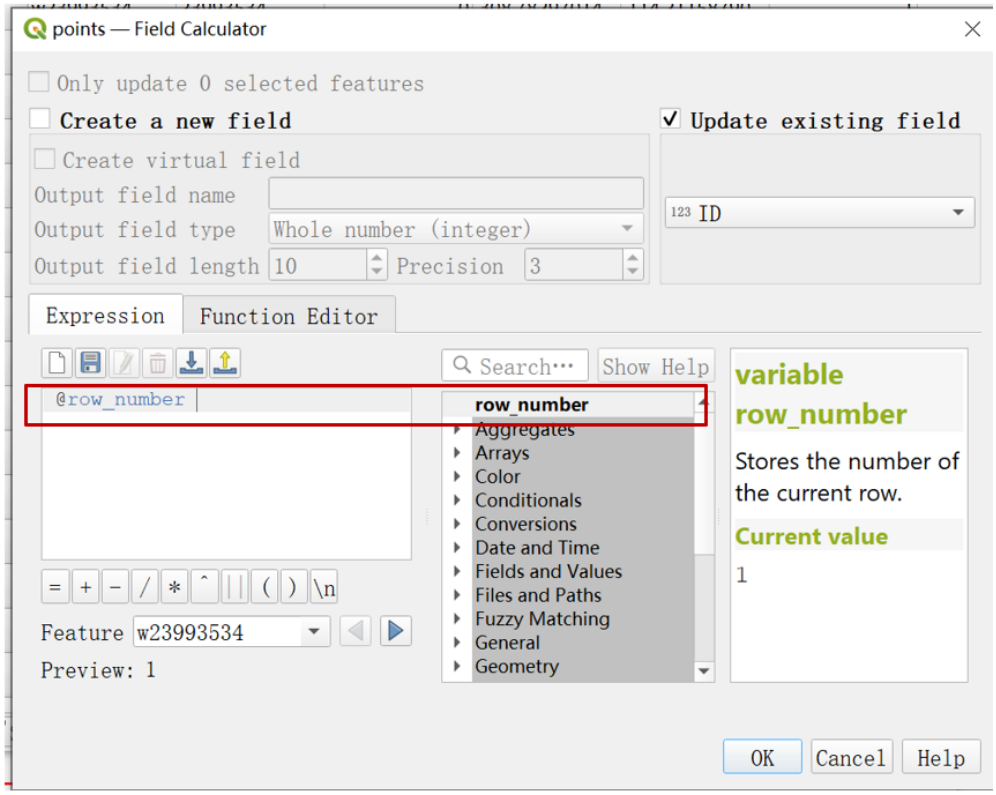
- **Add coordinate information and ID for each point**
  As this layer is used to search the google street view images at each location. But for now, we can check the attribute table, no information of the specific longitude and latitude are included.
  We can active the toggle editing, select ‘add filed’. Do the same for x and y columns.



Then open the field calculator, we use '$x' and '$y' function, and you need to select 'update existing field'.



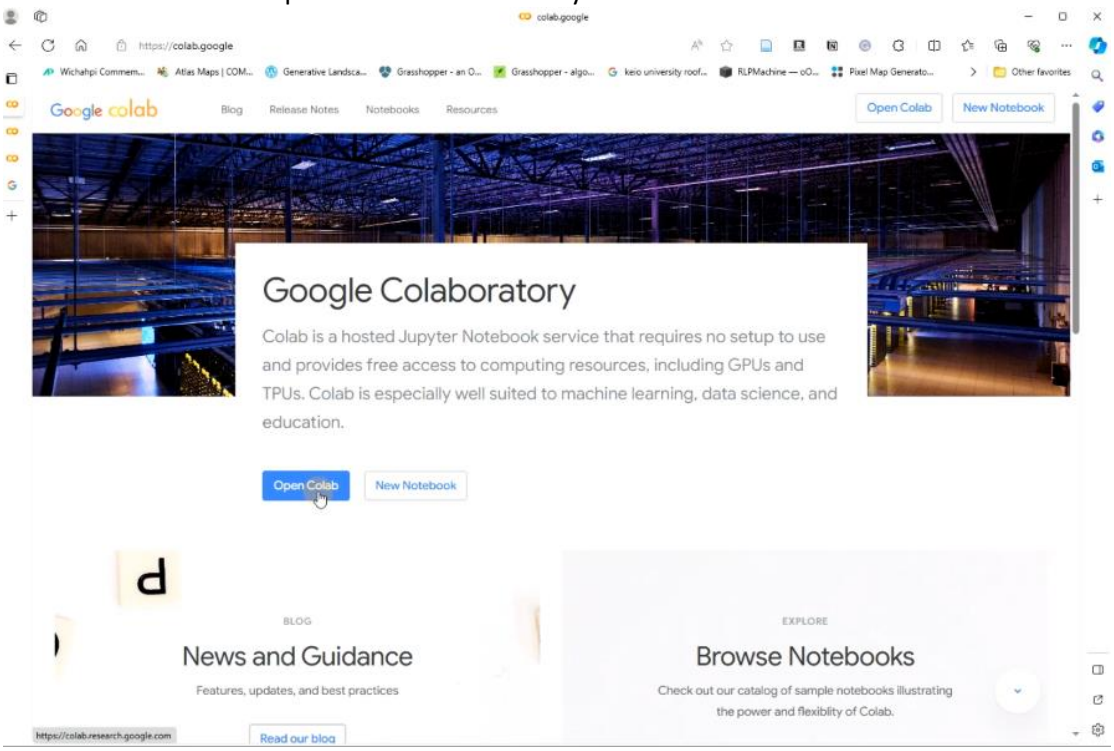
Using the same function, create a new column named 'ID', and this time, we use '@row_number' function in field calculator.



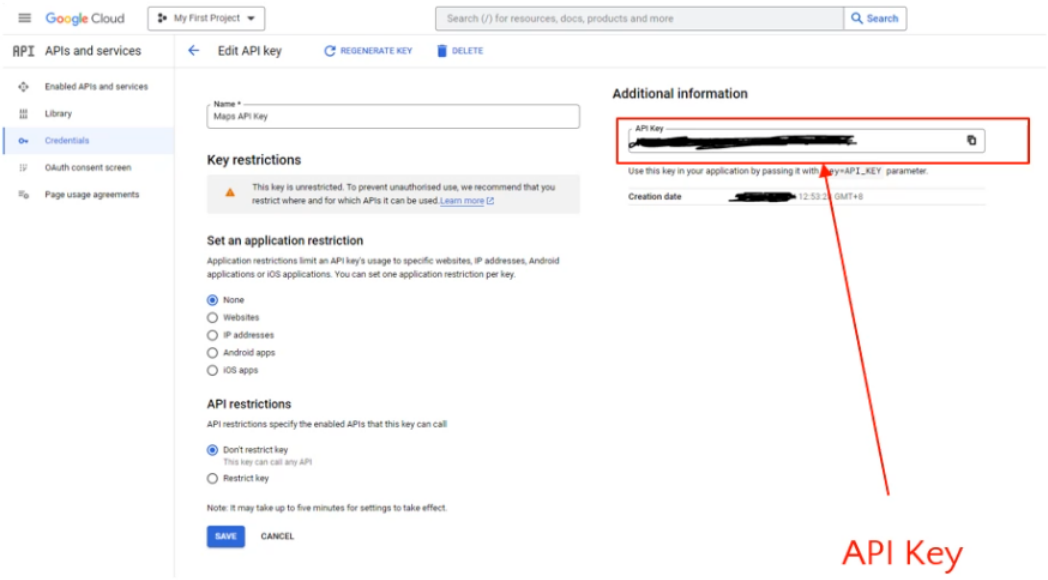
4. GSV Data Collecting

- **Loading the code into Google Colab Notebook**

  Here, we use Google Colab to open our shared code, the platform is relatively simple and convenient to set up the environment for Python code.



Then, in order to request SVI from the Google Street View API, you need to register a Google API Key (which can be found in the Google Cloud).



And we should also be aware of the monthly free quota given to the Google Account to avoid overuse by accident that could incur additional charge to your credit card account.

Pricing for the Street View Static API

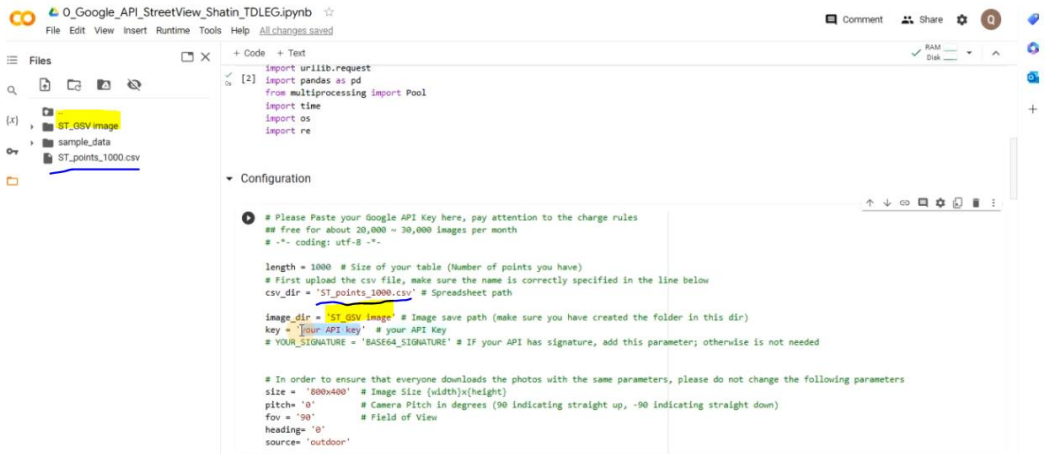
SKU: Static Street View

Street View panoramas and map loads are now charged separately. A static Street View panorama is charged for each request to the Street View Static API to embed a static (non-interactive) Street View panorama. Usage of the Street View Image Metadata endpoint is not charged.

MONTHLY VOLUME RANGE (Price per PANORAMA)		
0–100,000	100,001–500,000	500,000+
0.007 USD per each (7.00 USD per 1000)	0.0056 USD per each (5.60 USD per 1000)	Contact Sales for volume pricing

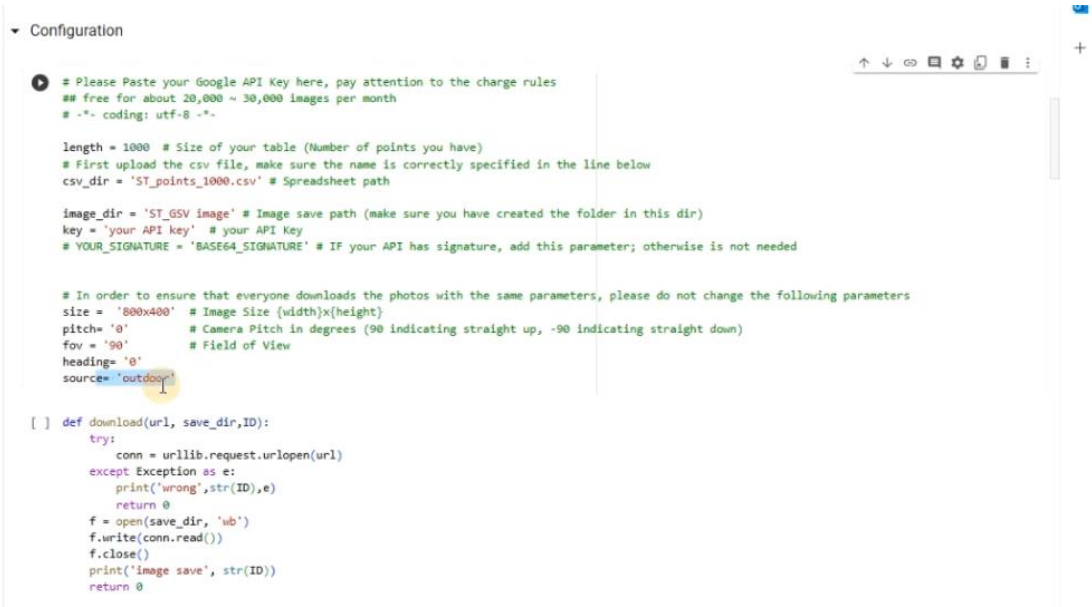
- **Configuring the basic setup and specifying parameters**

Here, we need to upload the csv file to the left panel area, and also create a folder 'ST_GSV_Image' to receive the requested street views.
Furthermore, add your API key to this line of code below.
Then run the cell.

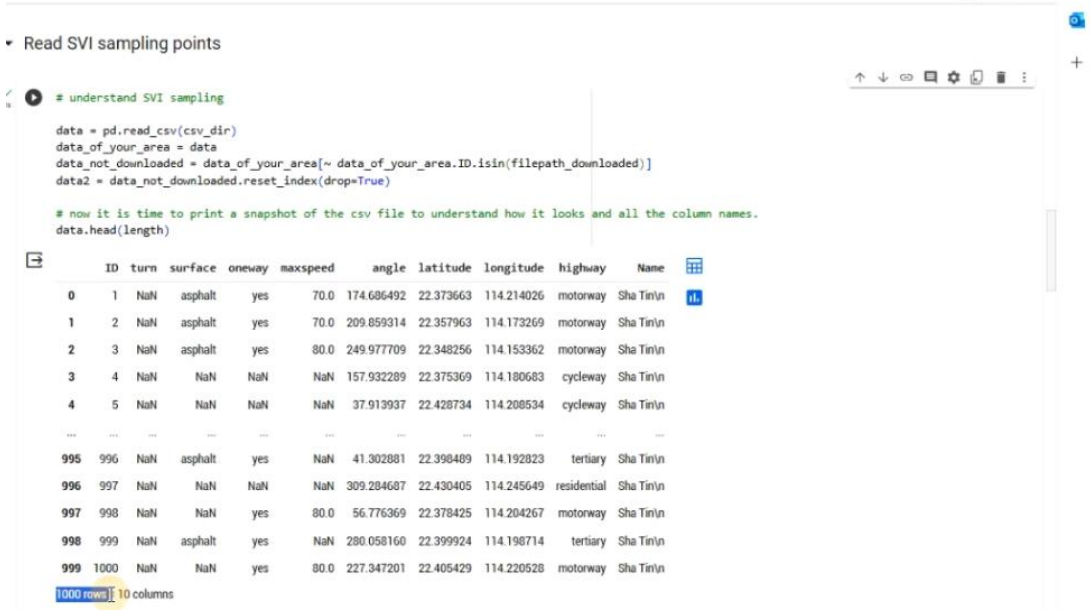


Here, we need to upload the csv file to the left panel area, and also create a folder 'ST_GSV_Image' to receive these images.
When requesting images, there are a few parameters that you could specify, for

instance, the heading indicates the compass heading of the camera. Accepted values are from 0 to 360 (both values indicating North, with 90 indicating East, and 180 South).
Size: this indicates the exact image size you want to download, for instance, 600 pixels by 400 pixels
fov: (default is 90) determines the horizontal field of view of the image expressed in degrees, with a maximum allowed value of 120.
pitch: (default is 0) specifies the up or down angle of the camera relative to the Street View vehicle.
Source: it is either default or outdoor, if it is outdoor, it will search the scenes that were taken outdoors.



Then we present the csv data to confirm the data frame it is loading.



	ID	turn	surface	oneway	maxspeed	angle	latitude	longitude	highway	Name
0	1	NaN	asphalt	yes	70.0	174.686492	22.373663	114.214026	motorway	Sha Tin\n
1	2	NaN	asphalt	yes	70.0	209.859314	22.357963	114.173269	motorway	Sha Tin\n
2	3	NaN	asphalt	yes	80.0	249.977709	22.348256	114.153362	motorway	Sha Tin\n
3	4	NaN	NaN	NaN	NaN	157.932289	22.375369	114.180683	cycleway	Sha Tin\n
4	5	NaN	NaN	NaN	NaN	37.913937	22.428734	114.208534	cycleway	Sha Tin\n
...	...	...	...	...	...	...	...	...	...	...
995	996	NaN	asphalt	yes	NaN	41.302881	22.398489	114.192823	tertiary	Sha Tin\n
996	997	NaN	NaN	NaN	NaN	309.284687	22.430405	114.245649	residential	Sha Tin\n
997	998	NaN	NaN	yes	80.0	56.776369	22.378425	114.204267	motorway	Sha Tin\n
998	999	NaN	asphalt	yes	NaN	280.058160	22.399924	114.198714	tertiary	Sha Tin\n
999	1000	NaN	NaN	yes	80.0	227.347201	22.405429	114.220528	motorway	Sha Tin\n

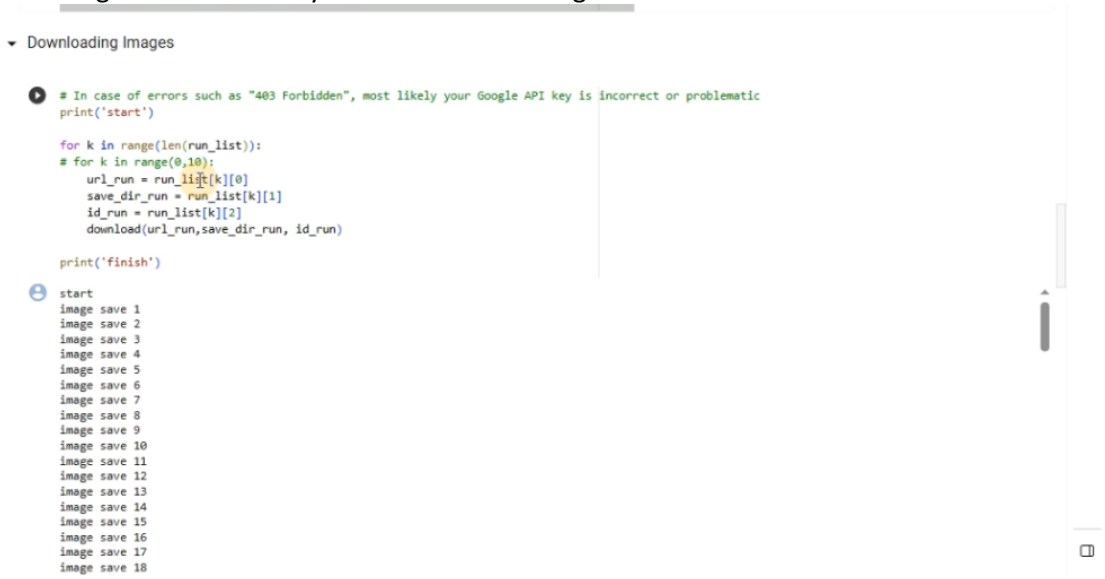
We also need to make sure the names called in these following cells could perfectly match the respective column names (e.g., latitude) read from the csv file. After making them consistent, we could successfully run these cells.

```
print('total image you need to download:',len(data_of_your_area))
print('have downloaded:', len(filepath_downloaded))
print( 'have not downloaded:',len(data2))

total image you need to download: 1000
have downloaded: 0
have not downloaded: 1000
```

Appending Data to request url

```
run_list = []
for i in range(len(data2)):
    ID = data2.loc[i, 'ID']  # Sampling point number

    latitude = data2.loc[i, 'latitude']  # Latitude ; make sure the name corresponds with the csv file.
    longitude = data2.loc[i, 'longitude']  # Longitude; make sure the name corresponds with the csv file.

    heading = data2.loc[i, 'angle']  # Road/Street Angle;need to manually change the angle number in the excel if you want to use other angle(90,2

    image_save_dir = image_dir  + '//'+ str(ID) + '.jpg'

    url = 'https://maps.googleapis.com/maps/api/streetview?size='+ size + '&location=' + str(latitude) + ',' + str(longitude) +  '&key='+  key +

    # url = "https://maps.googleapis.com/maps/api/streetview?size=" + size + "&location=" + str(latitude) + "," + str(longitude) + "&heading=" + st
    run_list.append((url,image_save_dir,ID))
```

If everything is correctly specified, then we could start downloading the images into the image folder. Patiently wait until all the images are downloaded.



Downloading Images

```
# In case of errors such as "403 Forbidden", most likely your Google API key is incorrect or problematic
print('start')

for k in range(len(run_list)):
# for k in range(0,10):
    url_run = run_list[k][0]
    save_dir_run = run_list[k][1]
    id_run = run_list[k][2]
    download(url_run,save_dir_run, id_run)

print('finish')
```

```
start
image save 1
image save 2
image save 3
image save 4
image save 5
image save 6
image save 7
image save 8
image save 9
image save 10
image save 11
image save 12
image save 13
image save 14
image save 15
image save 16
image save 17
image save 18
```

- **Saving and downloading the collected Street View Images**

  Here, we need to run the last cell in this Python notebook to mount your own Google Drive to the left side, the reason is that when you have many downloaded images saved to the temporary storage folder shown to the left panel, it is not possible to download the entire folder, you could only save each image one by one.

  Therefore, after mounting your own Drive, you could drag the image folder to the Drive folder, and you could go back to your own Google Drive to download these images there.

ST_GSV image
drive
MyDrive
sample_data
ST_points_1000.csv

```
[12] #this line of code has some issue, need to be fixed.
     print( 'have not downloaded:',len(data2))

     total image you need to download: 1000
     have downloaded: 107
     have not downloaded: 1000
```

Mount to google drive and manually move into that main folder, and then download the entire image folder from google drive. remember to refresh

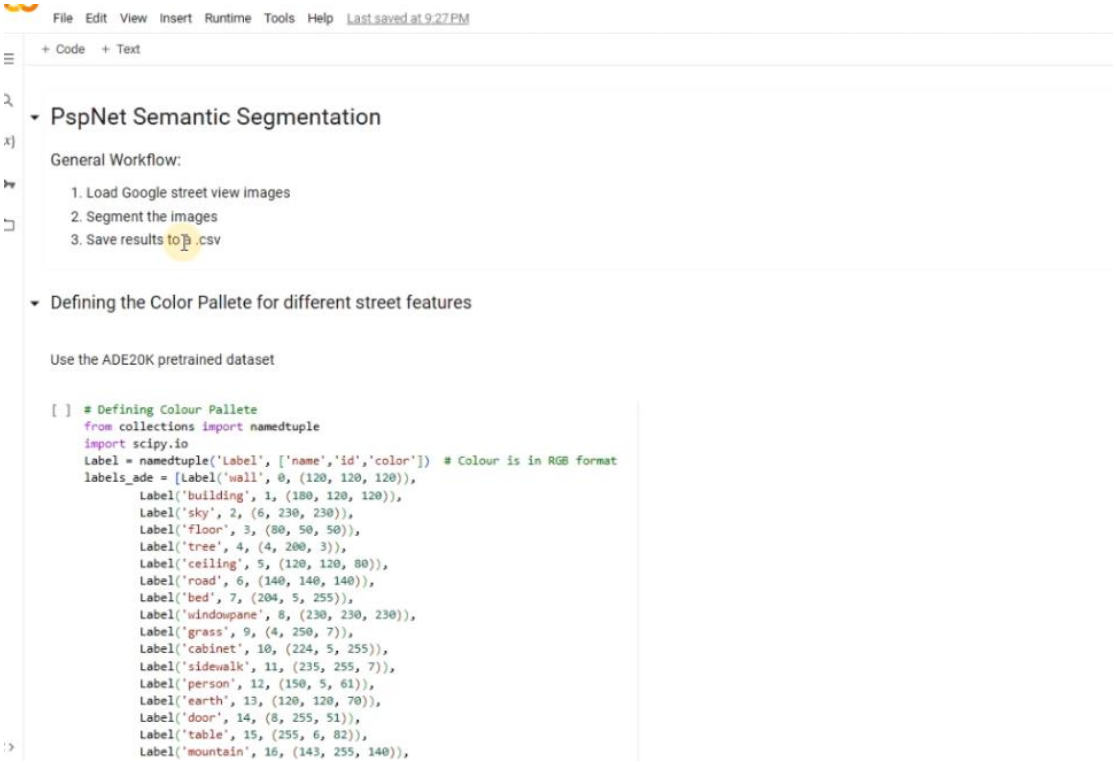
```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Please also manually check how many photos have been downloaded to avoid duplicates

5. Semantic Segmentation Using ADE20K and PSPNet

- **Loading the ADE20K dataset**

  Here, we need to read the labels from the pre-determined and widely used ADE20K dataset



File Edit View Insert Runtime Tools Help Last saved at 9:27 PM

+ Code + Text

PspNet Semantic Segmentation

General Workflow:

1. Load Google street view images
2. Segment the images
3. Save results to .csv

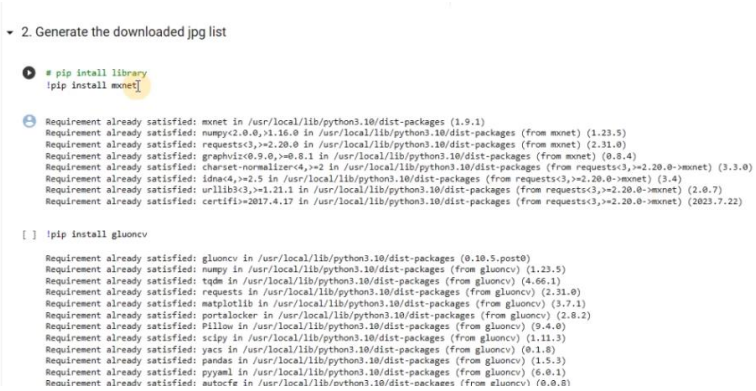
Defining the Color Pallete for different street features

Use the ADE20K pretrained dataset

```
[ ] # Defining Colour Pallete
    from collections import namedtuple
    import scipy.io
    Label = namedtuple('Label', ['name','id','color'])  # Colour is in RGB format
    labels_ade = [Label('wall', 0, (120, 120, 120)),
            Label('building', 1, (180, 120, 120)),
            Label('sky', 2, (6, 230, 230)),
            Label('floor', 3, (80, 50, 50)),
            Label('tree', 4, (4, 200, 3)),
            Label('ceiling', 5, (120, 120, 80)),
            Label('road', 6, (140, 140, 140)),
            Label('bed', 7, (204, 5, 255)),
            Label('windowpane', 8, (230, 230, 230)),
            Label('grass', 9, (4, 250, 7)),
            Label('cabinet', 10, (224, 5, 255)),
            Label('sidewalk', 11, (235, 255, 7)),
            Label('person', 12, (150, 5, 61)),
            Label('earth', 13, (120, 120, 70)),
            Label('door', 14, (8, 255, 51)),
            Label('table', 15, (255, 6, 82)),
            Label('mountain', 16, (143, 255, 140)),
```

- **Install packages and load image data**

  Here, we need to pip install related libraries and CV modules



2. Generate the downloaded jpg list

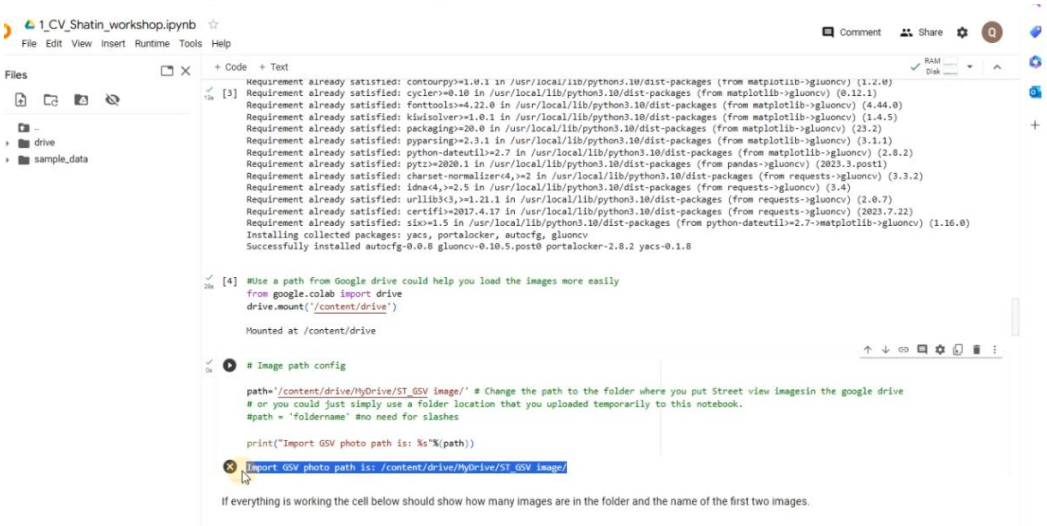
```
# pip intall library
!pip install mxnet
```

```
Requirement already satisfied: mxnet in /usr/local/lib/python3.10/dist-packages (1.9.1)
Requirement already satisfied: numpy<2.0.0,>1.16.0 in /usr/local/lib/python3.10/dist-packages (from mxnet) (1.23.5)
Requirement already satisfied: requests<3,>=2.20.0 in /usr/local/lib/python3.10/dist-packages (from mxnet) (2.31.0)
Requirement already satisfied: graphviz<0.9.0,>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from mxnet) (0.8.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.20.0->mxnet) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.20.0->mxnet) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.20.0->mxnet) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.20.0->mxnet) (2023.7.22)
```

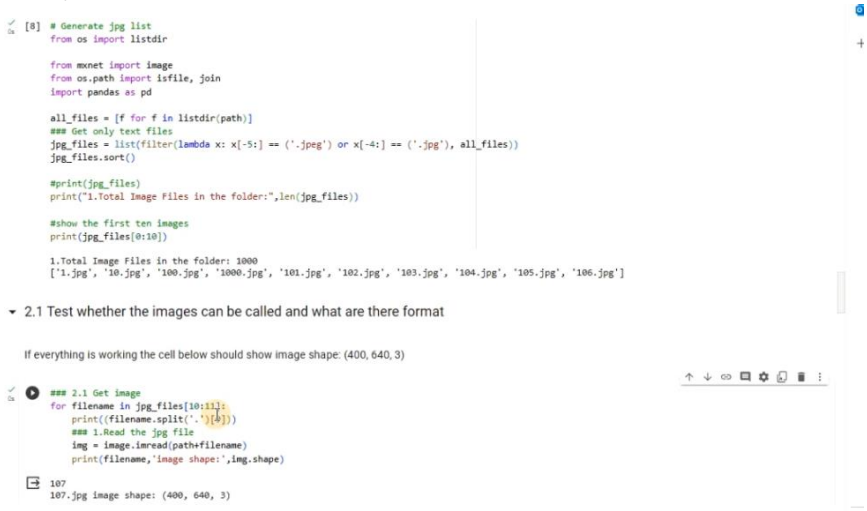
```
[ ] !pip install gluoncv
```

```
Requirement already satisfied: gluoncv in /usr/local/lib/python3.10/dist-packages (0.10.5.post0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from gluoncv) (1.23.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gluoncv) (4.66.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from gluoncv) (2.31.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from gluoncv) (3.7.1)
Requirement already satisfied: portalocker in /usr/local/lib/python3.10/dist-packages (from gluoncv) (2.8.2)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from gluoncv) (9.4.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from gluoncv) (1.11.3)
Requirement already satisfied: yacs in /usr/local/lib/python3.10/dist-packages (from gluoncv) (0.1.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from gluoncv) (1.5.3)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from gluoncv) (6.0.1)
Requirement already satisfied: autocfg in /usr/local/lib/python3.10/dist-packages (from gluoncv) (0.0.8)
```

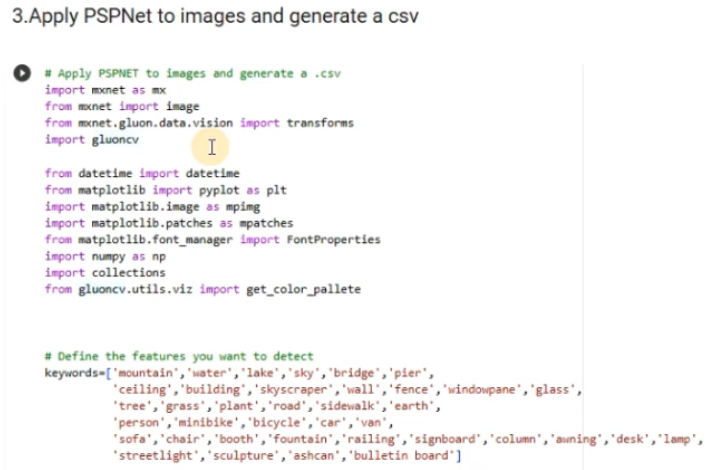
Here, we need to mount the google drive and read the image folder located in the Drive or we could simply upload the image folder directly to this temporary storage area to the left panel.



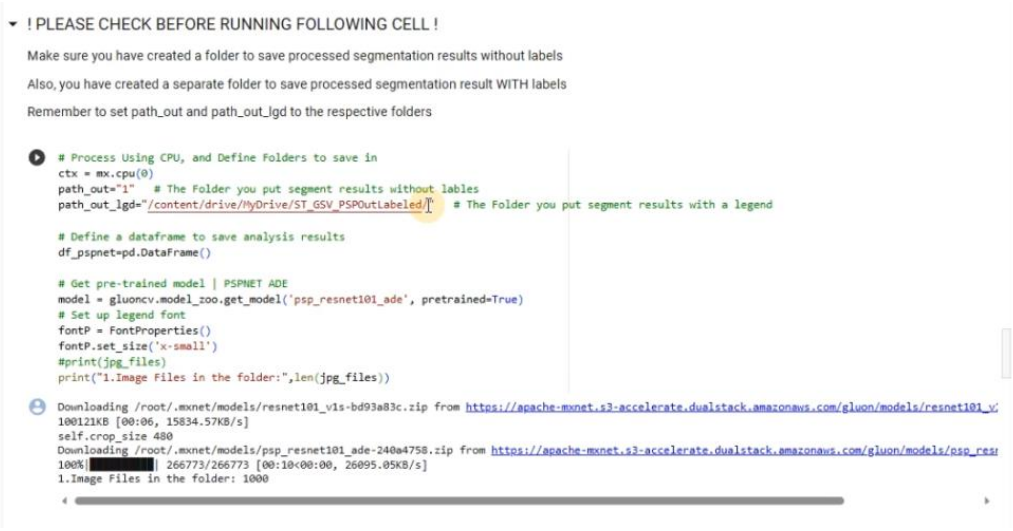
Then, we add modules and libraries that we need.



- **Applying the PSPNet and defining the features we want to extract**
  Here, we define the street elements we want to extract based on the labels from ADE20K.

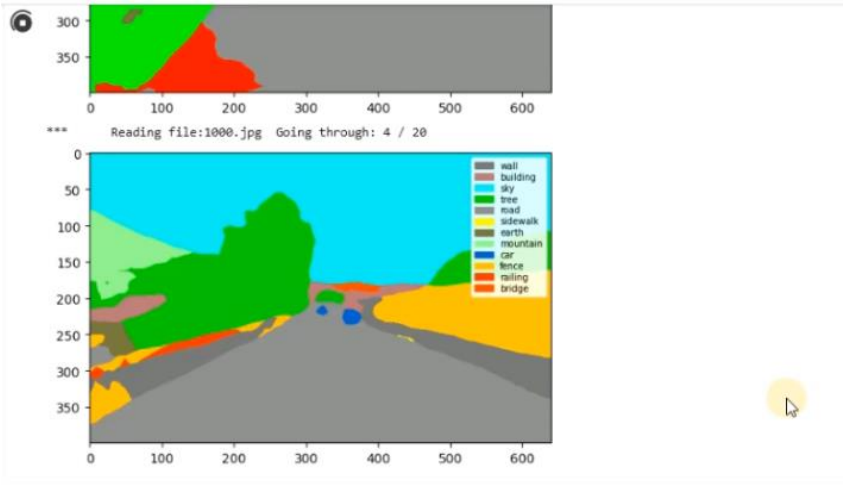


And we could then create two folders, one for the segmented images without labels, and the other containing labels. After getting the pre-trained model, we simply run this cell.



Next, define the start and end index of the images.

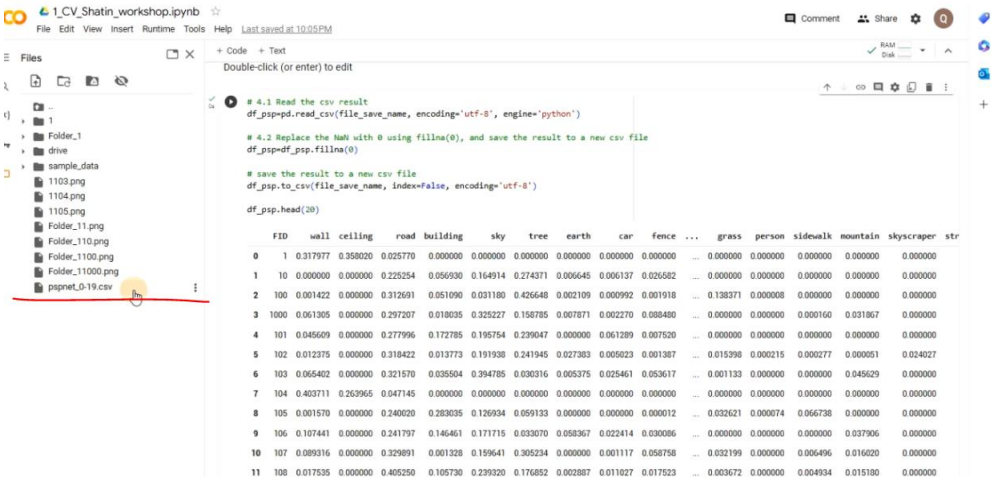
If all previous steps are run successfully, the segmented results can be obtained, and you will see the progress. Each image may need 10-20 seconds if you are using CPU, typically GPU is more powerful in terms of computing power and could accelerate this segmentation process.



Double-click (or enter) to edit

Remember to run the last piece of the code, as it saves the ratio of each street element

(also called visual index in urban studies literature) you chose previously for all the images into a combined csv file, that you could use to match back to the QGIS for future use.



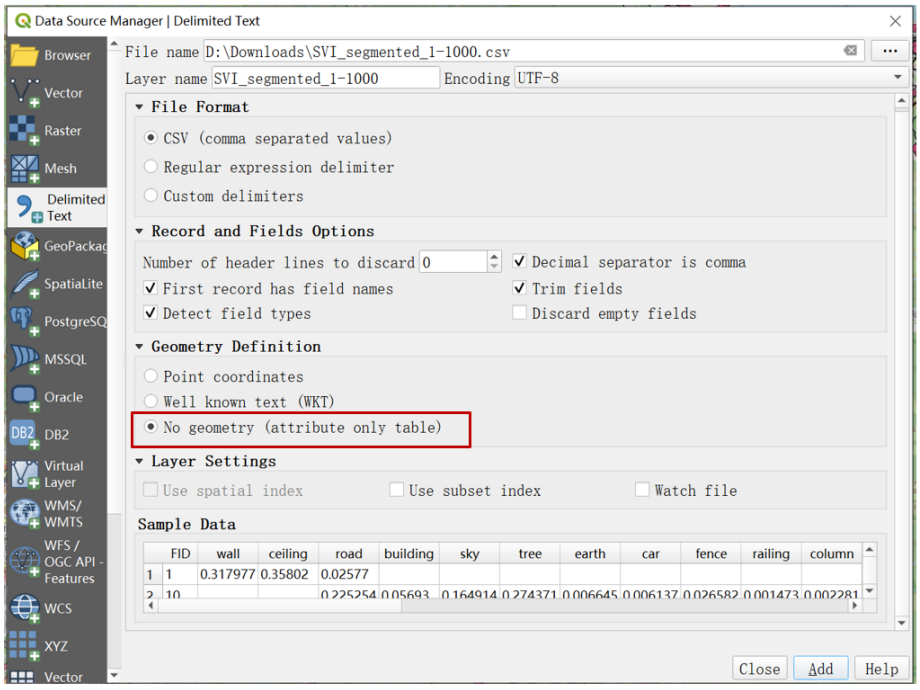
6. Data Visualization

- **Add 'delimited text layer'**

  Here we use two sample files.

  First, we import 'st_points.csv'. You can notice that this layer has coordinate information. We import it as a point layer.

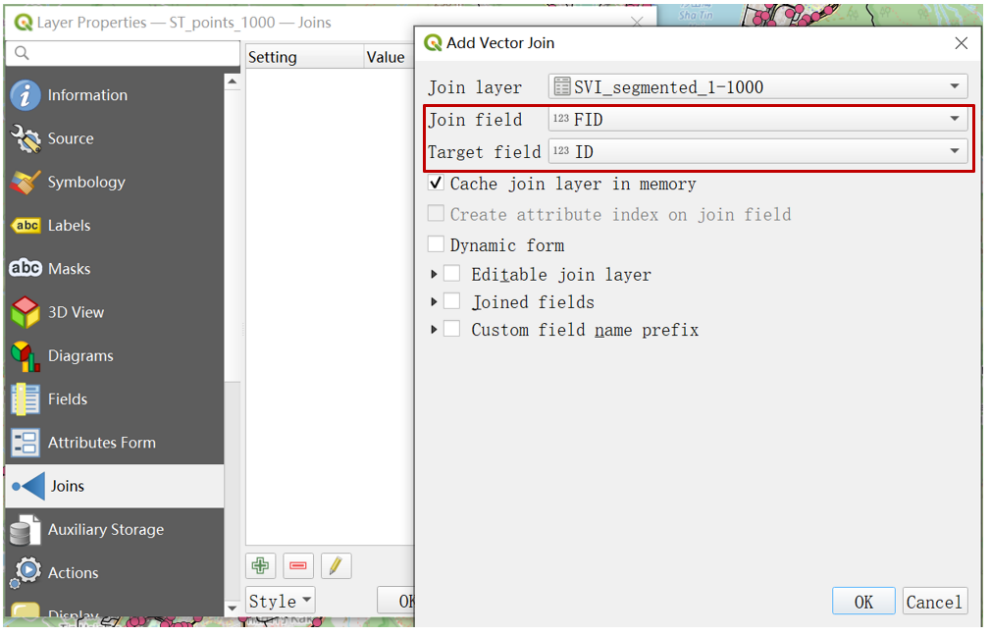
  After that, we import 'st_segmented_1-1000.csv'. This time we choose 'no geometry'.



- **Join tables**

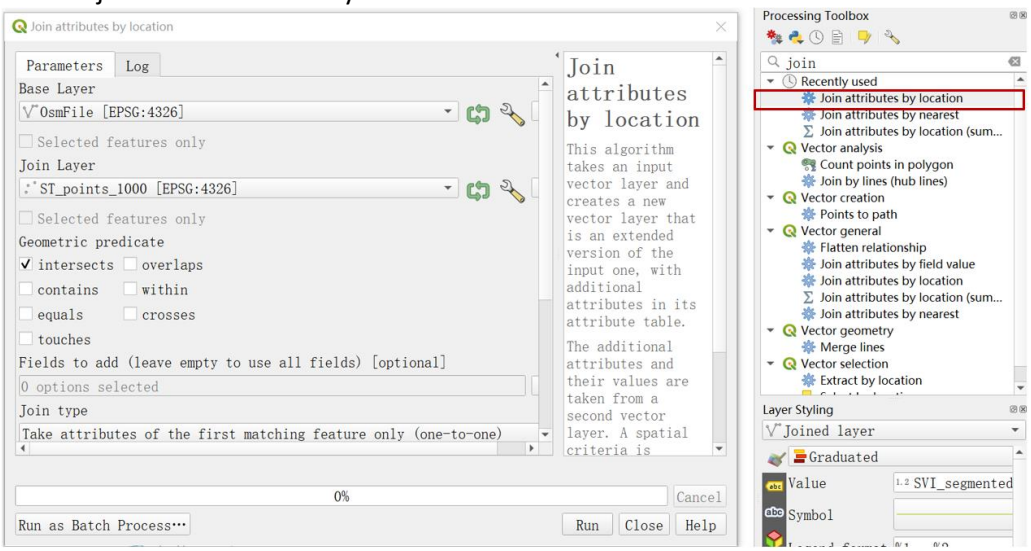
  For this case, we want to add the values of different visual indices into the point layer. For this purpose, we can notice that there are columns have equal values. 'ID' in the point layer and 'FID' in the segmented result.

  Right click the point layer, find 'properties', click 'Joins'. The join layer is the segmentation result. FID is the join field; ID is the target field. We can select joined fields, which depends on your tasks.



- **Assign value to road sections**

  We use join attribute table by location function.



- **Visualization**

  To visualize the one parameter by heatmap. You can right click the newly join layer, find ‘Properties’, select ‘Symbology’. You can either use categized or graduated. Select the parameter you take as the target, click ‘classify’. You can change the classes and mode for classification. You may need to try multiple intervals to achieved a nicer visual :).