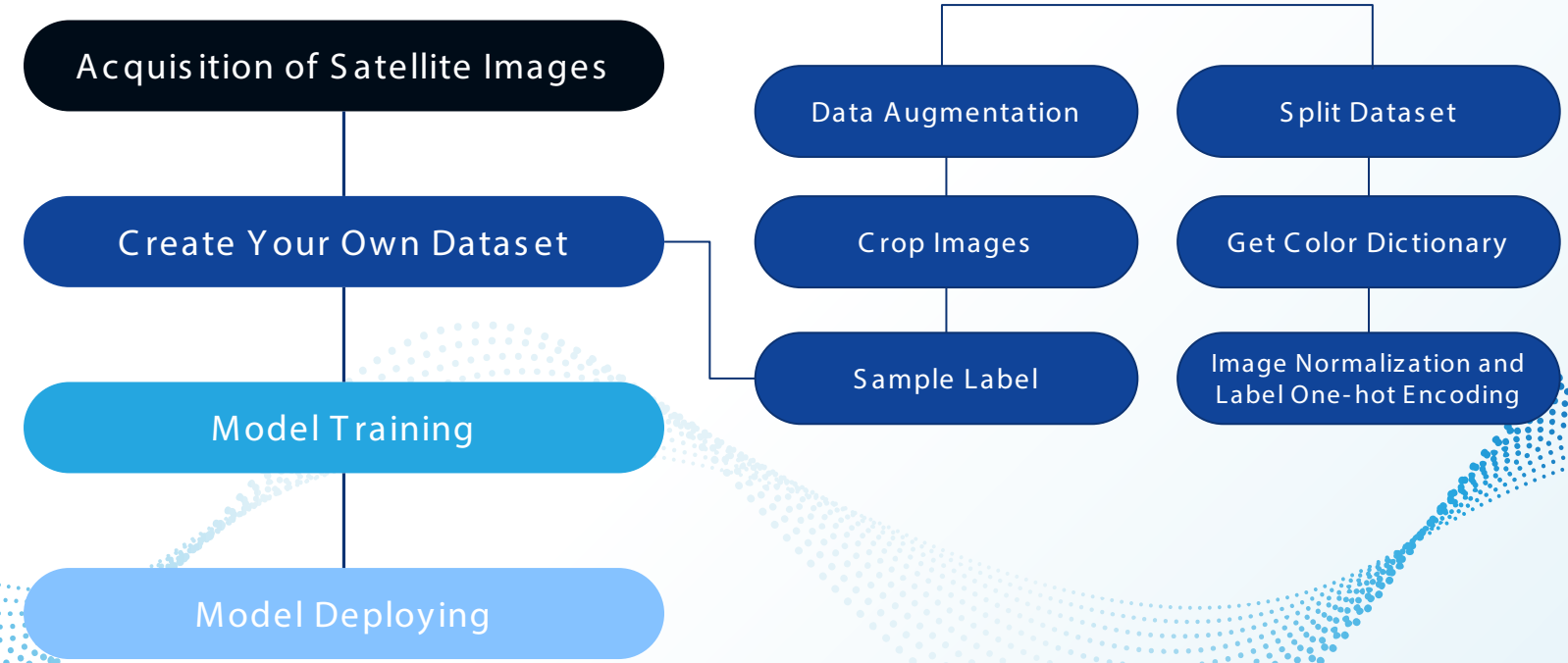
Build a Segmentation Model with Your Own Dataset

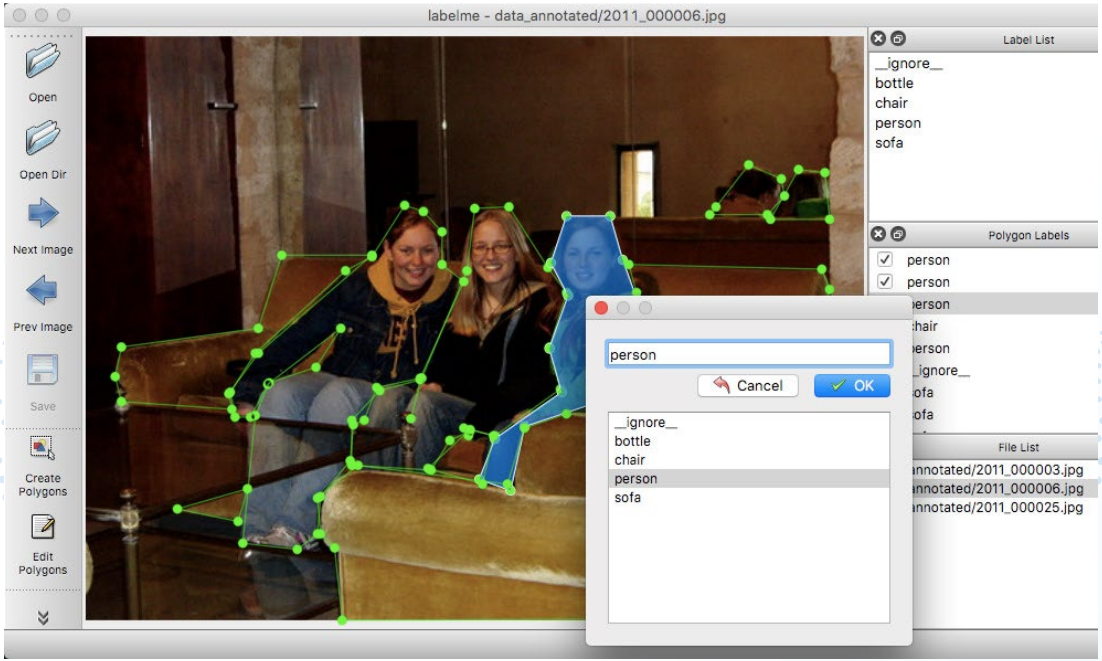
Suqi HUANG

Workflow of Building a Segmentation Model



DATA PRE-POSSESSING

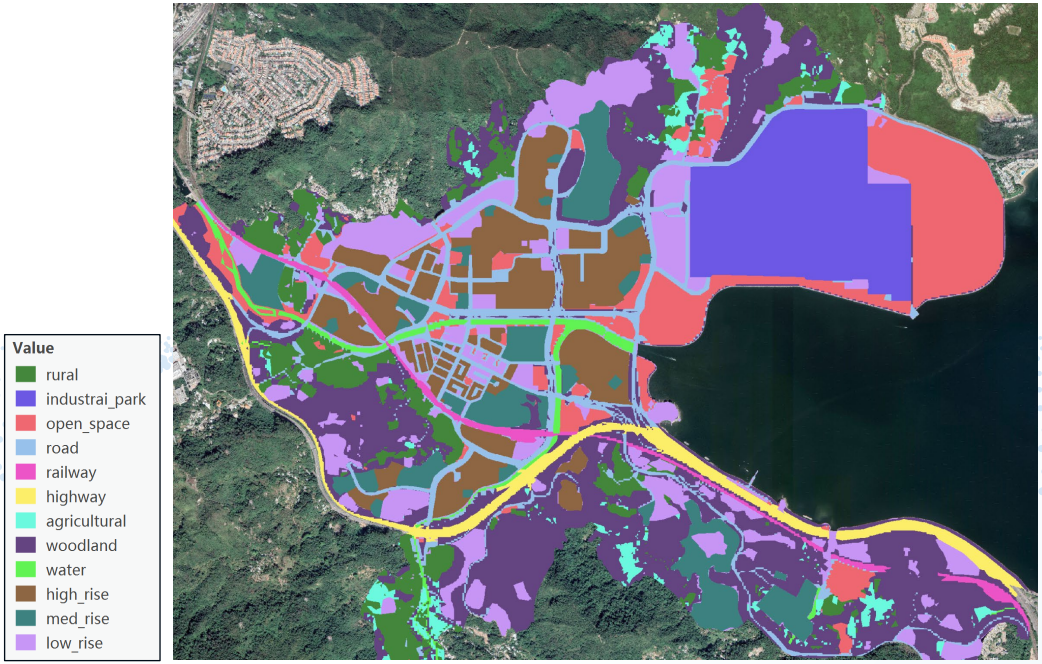
The most popular way to make masks: Labelme



- Time required to label one photo: 1 hour
- Relatively reliable rules of thumb: For each class, you need approximately 1000 representative training images

DATA PRE-POSSESSING

ArcGIS could create masks as well



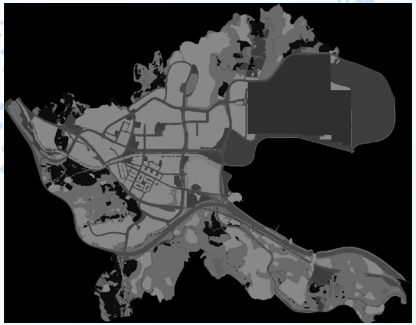
1. **Create masks**

   [Create New shapefile]

   or

   [Reclassify], Based on the land utilization raster

2. **Export raster**

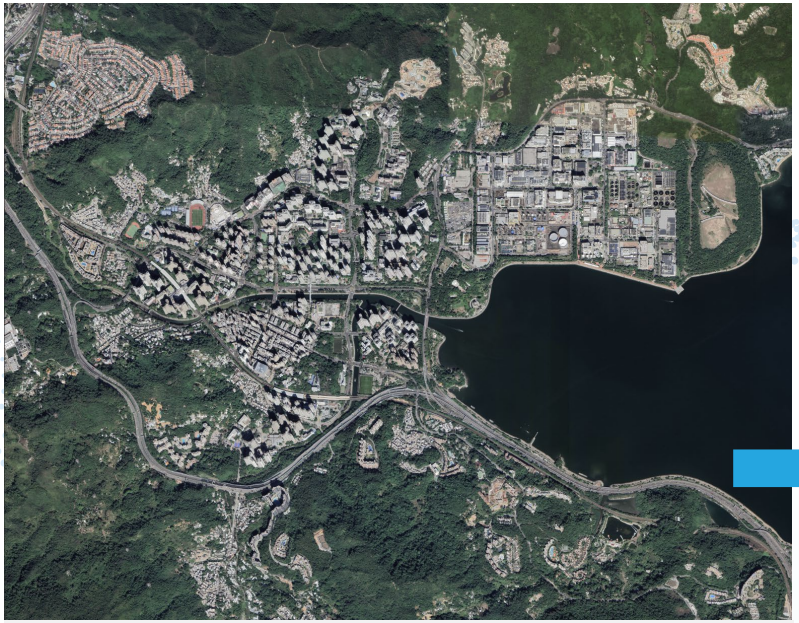


OID	Value	Count
0	3	70088818
1	22	96843288
2	32	102824718
3	41	78371678
4	42	13314064
5	43	28060578
6	61	20761070
7	71	237259757
8	92	13154283
9	101	96015701
10	102	67390684
11	103	112390515

- Save more time, compared to Labelme

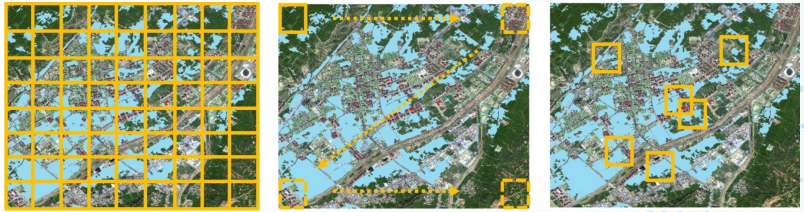
DATA PRE-POSSESSING

Crop images



```
# Regular Grid Crop
# Crop the image to a dataset of size 256×256 with a repetition rate of 0.1
TifCrop(r"Mask\LabelToRaster.tif",
        r"Mask\mask", 256, 0.1)
```

Three types of cropping



Regular Grid Crop | Sliding Window Crop | Random Crop

After cropping: 5330 images in total

DATA PRE-POSSESSING

Data augment → Split dataset to train and validation

Diagonal flip
Horizontal flip
Vertical flip

For satellite images, the sensor takes different angles of the same object and shows different positions on the image.
The geometrically transformed data tiles can enable the model to better learn the rotationally invariant features of the object and thus better adapt to different forms of images.

1. Iterate through folders to get a list of all file names

```
for root, dir, filenames in os.walk(img_path):
    for filename in filenames:
        if '.tif' in filename:
            file_path = os.path.join(root, filename)
            print(file_path)
            filelist_img_name.append(file_path)
        else:
            pass
            # print("it is not a jpg file.", filename)
```

```
Output exceeds the size limit. Open the full output data in a text editor
Data\raw_data\image\1.tif
Data\raw_data\image\10.tif
```

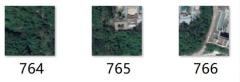
2. Randomly shuffle and get a new list

```
random.shuffle(filelist_img_name)
filelist_img_name = filelist_img_name
```

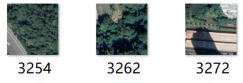
3. Copy files to the new folder according to the list

```
# split train data, train:validation=9:1
for tr in filelist_img_name[0:4797]:
    copy(tr, 'Data\\train\\image')
    tr = tr.replace('image\\', 'label\\')
    copy(tr, 'Data\\train\\label')
```

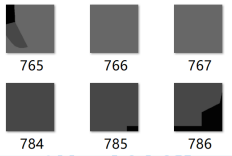
Train image



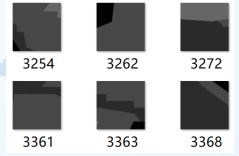
Validation image



Train label



Validation label



DATA PRE-POSSESSING

Get color dictionary

Iterate through the folder to identify and record all categories of colors and generate a color dictionary

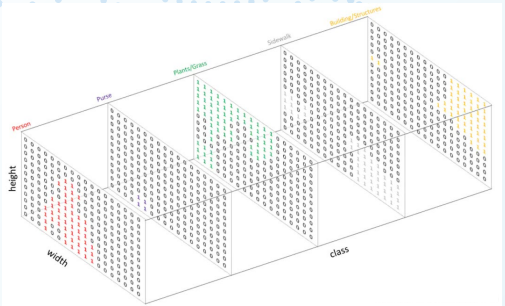
```
def color_dict(labelFolder, classNum):
    colorDict = []
    #  Get the name of the file in the folder
    ImageNameList = os.listdir(labelFolder)
    for i in range(len(ImageNameList)):
        ImagePath = labelFolder + "/" + ImageNameList[i]
        img = cv2.imread(ImagePath).astype(np.uint32)
        #  If grayscale, convert to RGB
        if(len(img.shape) == 2):
            img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB).astype(np.uint32)
        #  To extract unique values, convert RGB to a number
        img_new = img[:,:,0] * 1000000 + img[:,:,1] * 1000 + img[:,:,2]
        unique = np.unique(img_new)
        #  Add the unique value of the i-th pixel matrix to the colorDict
        for j in range(unique.shape[0]):
            colorDict.append(unique[j])
        #  Take the unique value again for the unique value in the current i-th pixel matrix
        colorDict = sorted(set(colorDict))
        #  If the number of unique values is equal to the total number of classes (ClassNum),
        #  stop iterating over the remaining images
        if(len(colorDict) == classNum):
            break

    #  Store RGB dictionary of colors for rendering results during prediction
    colorDict_RGB = []
    for k in range(len(colorDict)):
        color = str(colorDict[k]).rjust(9, '0')
        color_RGB = [int(color[0 : 3]), int(color[3 : 6]), int(color[6 : 9])]
        colorDict_RGB.append(color_RGB)
    #  Convert to numpy format
    colorDict_RGB = np.array(colorDict_RGB)
    #  Store the GRAY dictionary of colors for onehot encoding during preprocessing
    colorDict_GRAY = colorDict_RGB.reshape((colorDict_RGB.shape[0], 1 ,colorDict_RGB.shape[1])).astype(np.uint8)
    colorDict_GRAY = cv2.cvtColor(colorDict_GRAY, cv2.COLOR_BGR2GRAY)
    return colorDict_RGB, colorDict_GRAY
```

Image normalization and Label one-hot encoding

Because deep learning models are more sensitive to data value from 0 to 1, we need to normalize the data

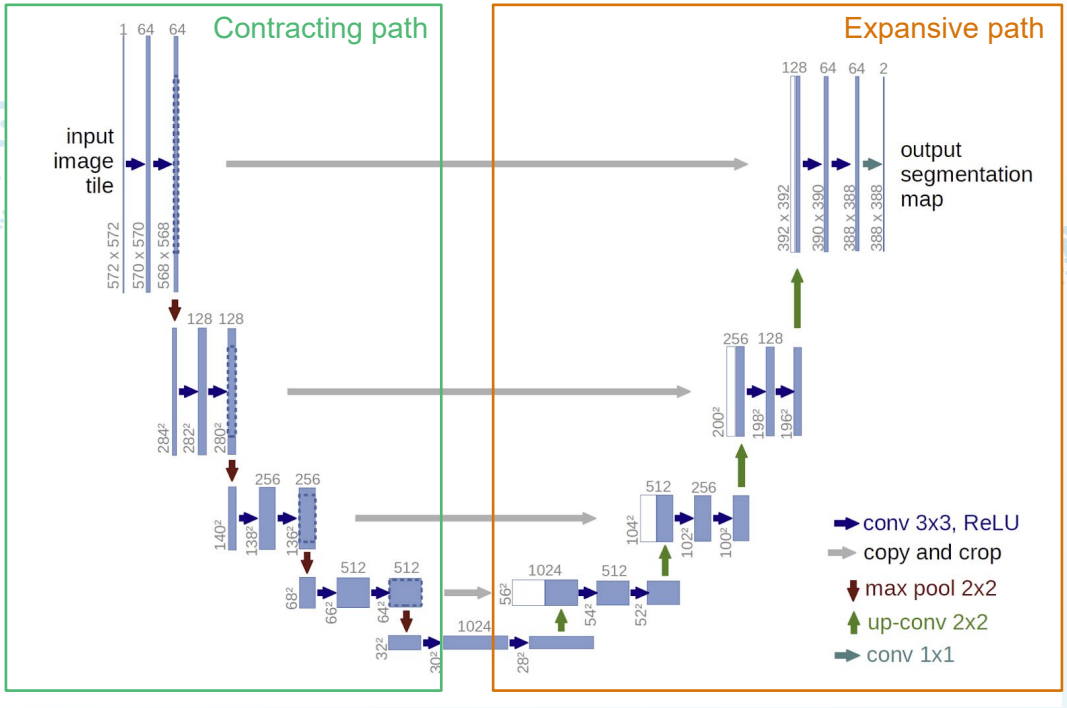
```
def dataPreprocess(img, label, classNum, colorDict_GRAY):
    #  Normalization
    img = img / 255.0
    for i in range(colorDict_GRAY.shape[0]):
        label[label == colorDict_GRAY[i][0]] = i

    new_label = np.zeros(label.shape + (classNum,))
    #  Turn each class of the flat label into a separate layer
    for i in range(classNum):
        new_label[label == i,i] = 1
    label = new_label
    return (img, label)
```

Example of label one-hot encoding



MODEL TRAINING

U- net Architecture



Running……

Thanks