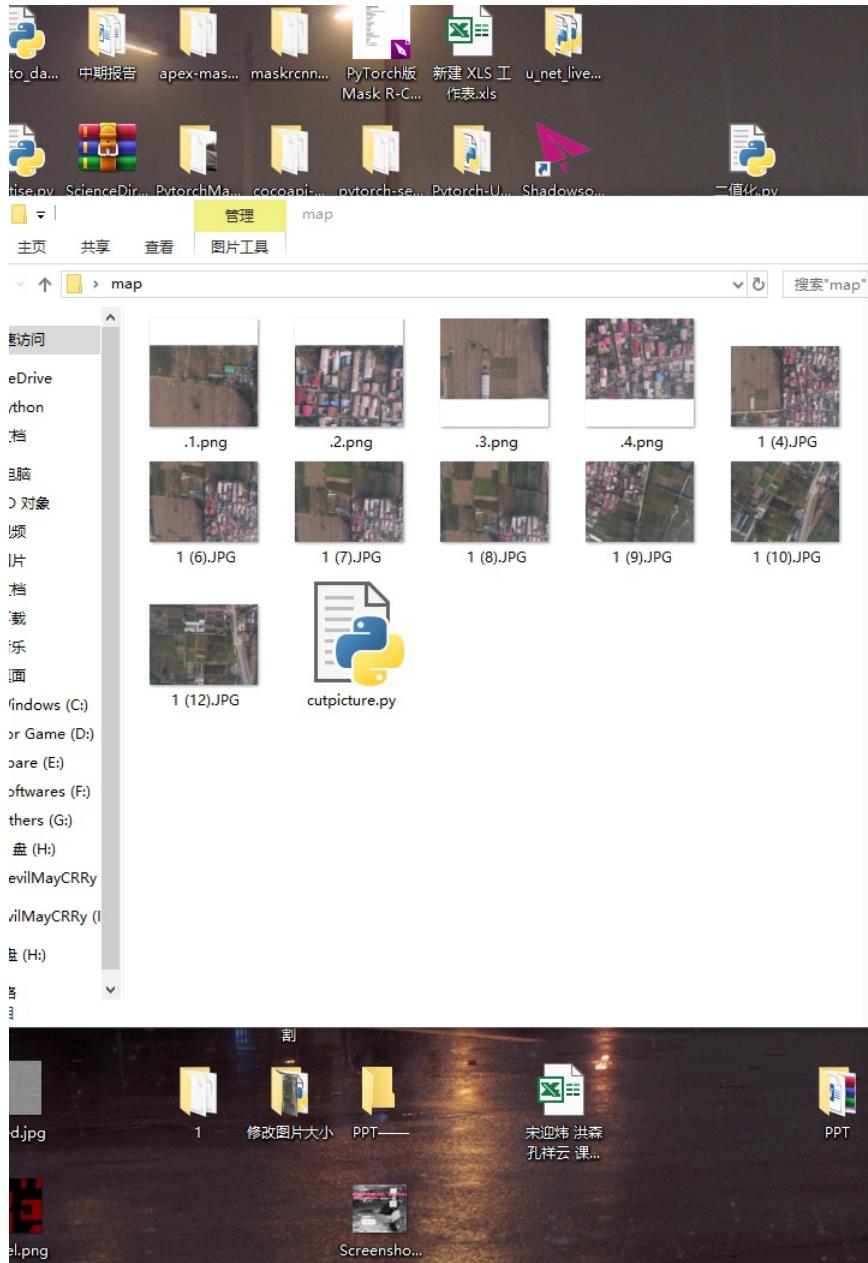


# U-net Segmentation(Pytorch)

Roofs in villages

Yingwei Song  
2019.12.17

# Divide the picture and divide it into nine equal parts for two times. Preliminary processing



```
File Edit Selection View Go Debug Terminal Help
cutpicture.py - map - Visual Studio Code - Insiders

cutpicture.py
22     # (left, upper, right, lower)
23     for i in range(0,2):#两重循环,生成9张图片基于原图的位置
24         for j in range(0,2):
25             #print((i*item_width,j*item_width,(i+1)*item_width,(j+1)*item_width))
26             box = (j*item_width,i*item_width,(j+1)*item_width,(i+1)*item_width)
27             box_list.append(box)
28
29     image_list = [image.crop(box) for box in box_list]
30     return image_list
31 #保存
32 def save_images(image_list):
33     index = 1
34     for image in image_list:
35         image.save('.+'+str(index) + '.png', 'PNG')
36         index += 1
37
38 if __name__ == '__main__':
39     file_path = "python.jpeg"
40     image = Image.open("1 (5).jpg")
41
42     image = fill_image(image)
43     image_list = cut_image(image)
44     save_images(image_list)

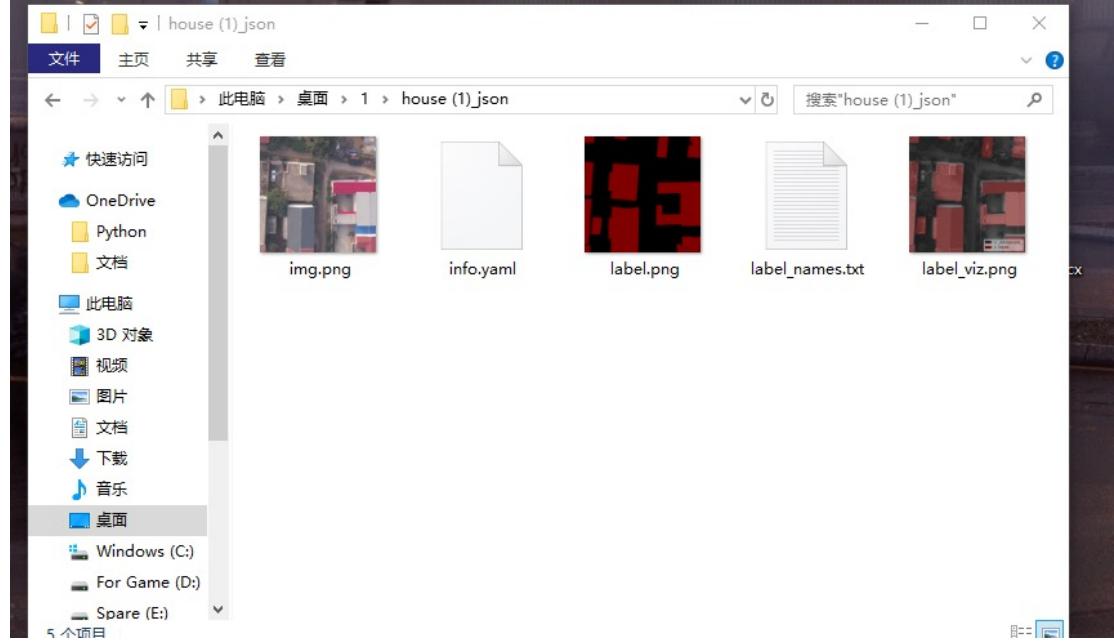
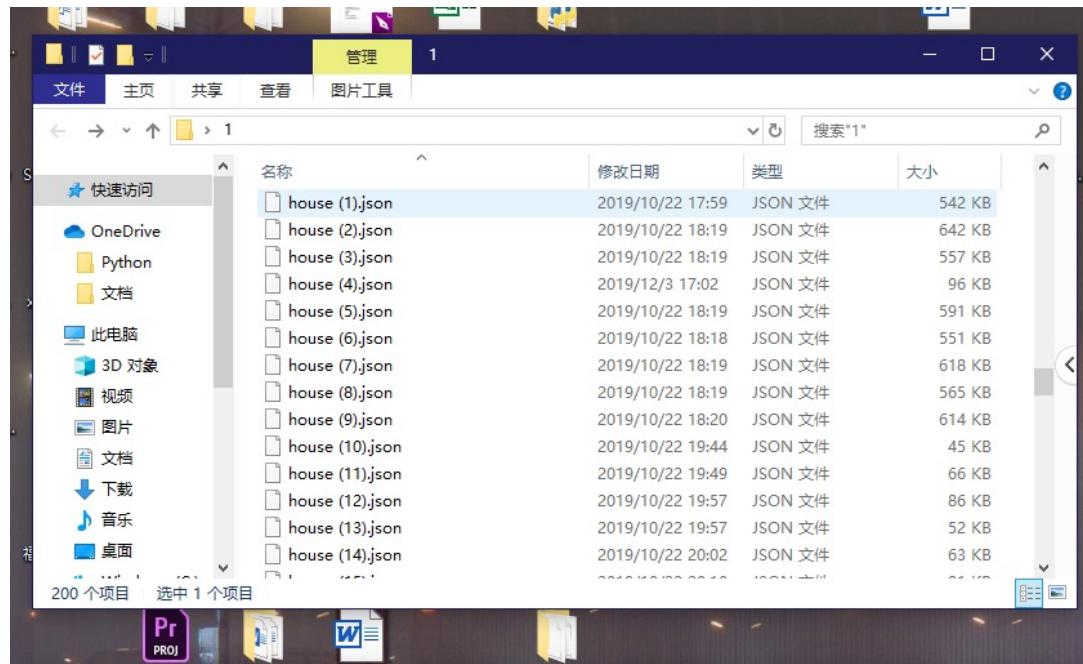
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Users\muyin\Desktop\map> conda activate base
PS C:\Users\muyin\Desktop\map> & E:/Work/Anaconda/file/python.exe c:/Users/muyin/Desktop/map/cutpicture.py
Traceback (most recent call last):
  File "c:/Users/muyin/Desktop/map/cutpicture.py", line 40, in <module>
    image = Image.open("1 (5).png")
  File "E:/Work/Anaconda/file/lib/site-packages/PIL\Image.py", line 2609, in open
    fp = builtins.open(filename, "rb")
FileNotFoundError: [Errno 2] No such file or directory: '1 (5).png'
PS C:\Users\muyin\Desktop\map> & E:/Work/Anaconda/file/python.exe c:/Users/muyin/Desktop/map/cutpicture.py
PS C:\Users\muyin\Desktop\map>
```



## Using “labelme” to label datasets manually

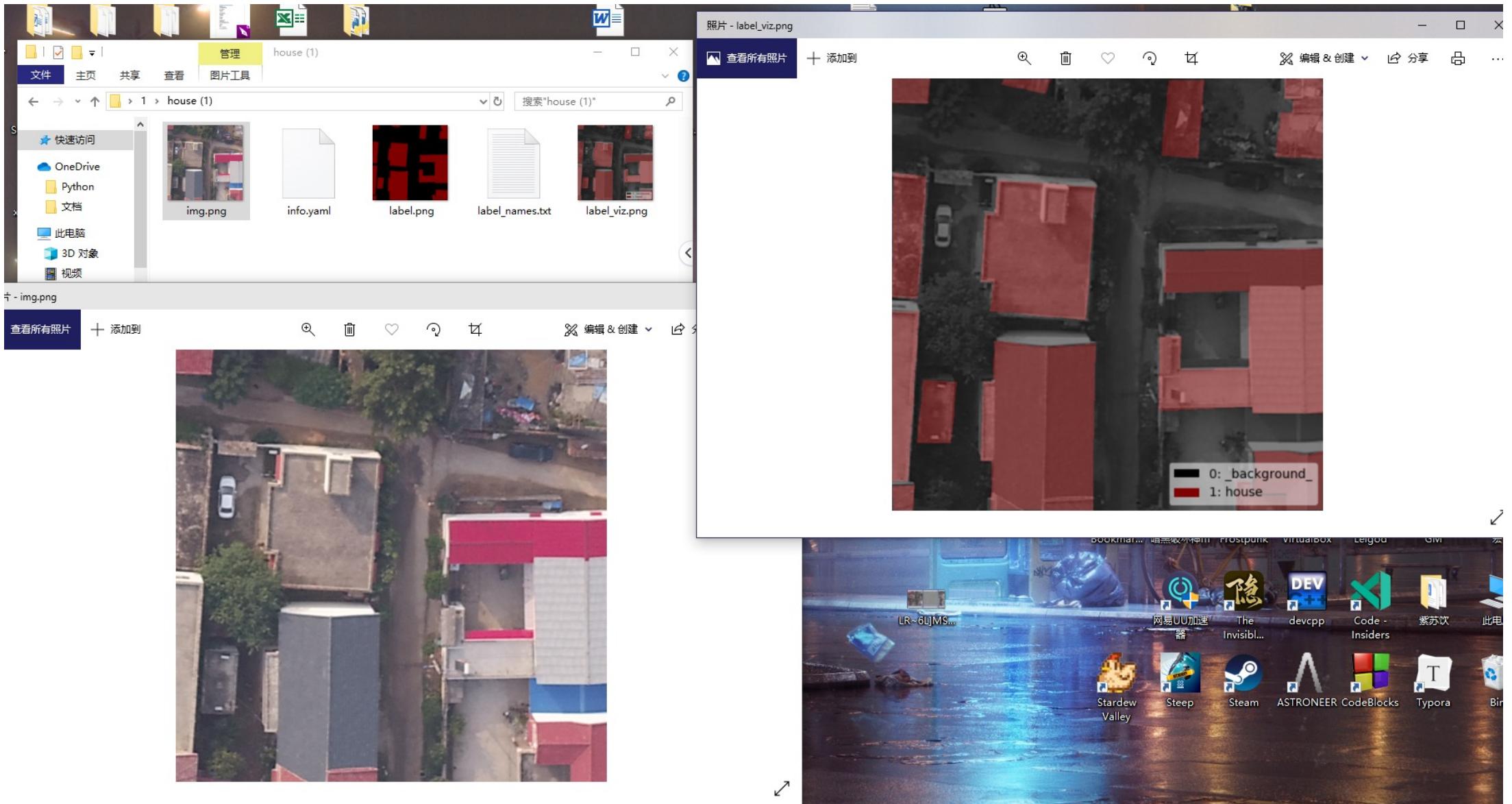


## Parse the .json file



```
PS C:\Users\muyin\Desktop> labelme_json_to_dataset "C:\Users\muyin\Desktop\1\house (1).json"
[33m[WARNING] [0m This script is aimed to demonstrate how to convert the JSON file to a single image dataset, and not to handle multiple JSON files to generate a real-use dataset. ([1]json_to_dataset.py[0m:15)
[33m[WARNING] [0m info.yaml is being replaced by label_names.txt ([1]json_to_dataset.py[0m:67)
[37m[INFO] [0m Saved to: C:\Users\muyin\Desktop\1\house (1)_json ([1]json_to_dataset.py[0m:72)
PS C:\Users\muyin\Desktop>
PS C:\Users\muyin\Desktop>
```

# Outcomes



# Set up the environment

The image shows a Windows desktop environment. In the top-left corner, there is an 'Anaconda Prompt - python -m visdom.server' window. The command `python -m visdom.server` is being run, and the output shows a warning about the deprecation of `zmq.eventloop.ioloop.install()` and the application starting at port 8097. In the top-right corner, there is a Jupyter Notebook interface titled 'jupyter Semantic Segmentation-HOUSE-train'. The notebook has a toolbar with various icons for file operations, cell execution, and help. Below the toolbar, the code cell 'In [9]:' contains Python imports for various libraries including os, cv2, time, numpy, torch, torch.nn, torch.optim, torch.backends.cudnn, torch.utils.data, matplotlib, albumenations, sklearn.model\_selection, PIL, glob, segmentation\_models\_pytorch, and visdom. The code also initializes a Visdom server with `vis = visdom.Visdom(use\_incoming\_socket=False)`.

```
(base) C:\Users\myuin>python -m visdom.server
E:\Work\Anaconda\file\lib\site-packages\visdom-0.1.8.9-py3.7.egg\visdom\server.py:36: DeprecationWarning: zmq.eventloop.ioloop is deprecated in pyzmq 17. pyzmq now works with default tornado and asyncio eventloops.
    ioloop.install() # Needs to happen before any tornado imports!
It's Alive!
INFO:root:Application Started
You can navigate to http://localhost:8097
```

jupyter Semantic Segmentation-HOUSE-train 最后检查: 29分钟前 (自动保存)

In [9]:

```
import os
import cv2
import time
import numpy as np
from torch.optim.lr_scheduler import ReduceLROnPlateau
import torch
import torch.nn as nn
from torch.nn import functional as F
import torch.optim as optim
import torch.backends.cudnn as cudnn
from torch.utils.data import DataLoader, Dataset
from matplotlib import pyplot as plt
from albumentations import (Resize, RandomCrop, VerticalFlip, HorizontalFlip, Normalize, Compose)
from albumentations.pytorch import ToTensor
import torch.nn.functional as F
from torch.autograd import Variable
from sklearn.model_selection import train_test_split
from PIL import Image
from glob import glob
import segmentation_models_pytorch as smp
import visdom
```

```
vis = visdom.Visdom(use_incoming_socket=False)
```

Data enhancement:  
Flip horizontally and vertically

```
[11]: def get_transforms(phase, mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)):  
    list_transforms = []  
    if phase == "train":  
        list_transforms.extend(  
            [  
                HorizontalFlip(),  
                VerticalFlip()  
            ]  
        )  
    else:  
        list_transforms.append(CenterCrop(256))  
    return list_transforms
```

- batch size=8
- lr=1e-3
- optimizer=Adam
- loss=BCEWithLogitsLoss
- epoch=50

```
39]: class Trainer(object):  
    '''This class takes care of training and validation of our model'''  
    def __init__(self, model):  
        self.num_workers = 0  
        self.batch_size = {"train": 8, "val": 1}  
        self.accumulation_steps = 32 // self.batch_size['train']  
        self.lr = 1e-3  
        self.num_epochs = 50  
        self.best_loss = float("inf")  
        self.best_dice = float(0)  
        self.phases = ["train", "val"]  
        self.device = torch.device("cuda:0")  
        torch.set_default_tensor_type("torch.cuda.FloatTensor")  
        self.net = model  
        self.criterion = nn.BCEWithLogitsLoss()  
        self.optimizer = optim.Adam(self.net.parameters(), lr=self.lr)  
        self.scheduler = ReduceLROnPlateau(self.optimizer, mode="min", patience=4, verbose=True)  
        self.net = self.net.to(self.device)  
        cudnn.benchmark = True
```

## Train:

The screenshot shows a Jupyter Notebook interface with the following elements:

- Toolbar:** Includes icons for file operations (New, Open, Save, etc.), cell selection (Run Cell, Run Cell and Next, Run Cell and Previous), and code execution (Cell Kernel).
- Code Cell:** Contains Python code to initialize a trainer and start the training loop.
- Output Stream:** Displays the training logs for five epochs (0 to 4) in a monospaced font. Each epoch includes metrics for both train and validation phases, and a note indicating new optimal found and state saving.

```
[*]: model_trainer = Trainer(model)
model_trainer.start()

Starting epoch: 0 | phase: train | : 20:41:26
Loss: 0.638610 | IoU: 0.2651 | dice: 0.3997 | dice_neg: 0.0000 | dice_pos: 0.3997
Starting epoch: 0 | phase: val | : 20:41:31
Loss: 2.471894 | IoU: 0.2841 | dice: 0.4293 | dice_neg: 0.0000 | dice_pos: 0.4293
***** New optimal found, saving state *****

Starting epoch: 1 | phase: train | : 20:41:33
Loss: 0.483489 | IoU: 0.4683 | dice: 0.6192 | dice_neg: 0.0000 | dice_pos: 0.6192
Starting epoch: 1 | phase: val | : 20:41:39
Loss: 0.721443 | IoU: 0.3587 | dice: 0.5139 | dice_neg: 0.0000 | dice_pos: 0.5139
***** New optimal found, saving state *****

Starting epoch: 2 | phase: train | : 20:41:41
Loss: 0.426659 | IoU: 0.4948 | dice: 0.6348 | dice_neg: 0.0000 | dice_pos: 0.6348
Starting epoch: 2 | phase: val | : 20:41:47
Loss: 0.454439 | IoU: 0.4610 | dice: 0.5923 | dice_neg: 0.0000 | dice_pos: 0.5923
***** New optimal found, saving state *****

Starting epoch: 3 | phase: train | : 20:41:49
Loss: 0.385652 | IoU: 0.5588 | dice: 0.6938 | dice_neg: 0.0000 | dice_pos: 0.6938
Starting epoch: 3 | phase: val | : 20:41:55
Loss: 0.501673 | IoU: 0.4748 | dice: 0.6104 | dice_neg: 0.0000 | dice_pos: 0.6104
```

After 50 iterations:

```
Loss: 0.155410 | IoU: 0.8101 | dice: 0.8897 | dice_neg: 0.0000 | dice_pos: 0.8897
Starting epoch: 44 | phase: val | : 20:46:49
Loss: 0.204534 | IoU: 0.7118 | dice: 0.8259 | dice_neg: 0.0000 | dice_pos: 0.8259
```

```
Starting epoch: 45 | phase: train | : 20:46:50
Loss: 0.156369 | IoU: 0.7942 | dice: 0.8802 | dice_neg: 0.0000 | dice_pos: 0.8802
Starting epoch: 45 | phase: val | : 20:46:56
Loss: 0.209885 | IoU: 0.7027 | dice: 0.8153 | dice_neg: 0.0000 | dice_pos: 0.8153
```

```
Starting epoch: 46 | phase: train | : 20:46:57
Loss: 0.148625 | IoU: 0.8246 | dice: 0.9007 | dice_neg: 0.0000 | dice_pos: 0.9007
Starting epoch: 46 | phase: val | : 20:47:03
Loss: 0.217943 | IoU: 0.6911 | dice: 0.8049 | dice_neg: 0.0000 | dice_pos: 0.8049
```

```
Starting epoch: 47 | phase: train | : 20:47:04
Loss: 0.147558 | IoU: 0.8140 | dice: 0.8932 | dice_neg: 0.0000 | dice_pos: 0.8932
Starting epoch: 47 | phase: val | : 20:47:10
Loss: 0.212451 | IoU: 0.6969 | dice: 0.8111 | dice_neg: 0.0000 | dice_pos: 0.8111
```

```
Starting epoch: 48 | phase: train | : 20:47:11
Loss: 0.149801 | IoU: 0.8203 | dice: 0.8978 | dice_neg: 0.0000 | dice_pos: 0.8978
Starting epoch: 48 | phase: val | : 20:47:16
Loss: 0.198635 | IoU: 0.7146 | dice: 0.8272 | dice_neg: 0.0000 | dice_pos: 0.8272
Epoch      48: reducing learning rate of group 0 to 1.0000e-04.
```

```
Starting epoch: 49 | phase: train | : 20:47:17
Loss: 0.136240 | IoU: 0.8406 | dice: 0.9106 | dice_neg: 0.0000 | dice_pos: 0.9106
Starting epoch: 49 | phase: val | : 20:47:23
Loss: 0.194541 | IoU: 0.7197 | dice: 0.8305 | dice_neg: 0.0000 | dice_pos: 0.8305
```

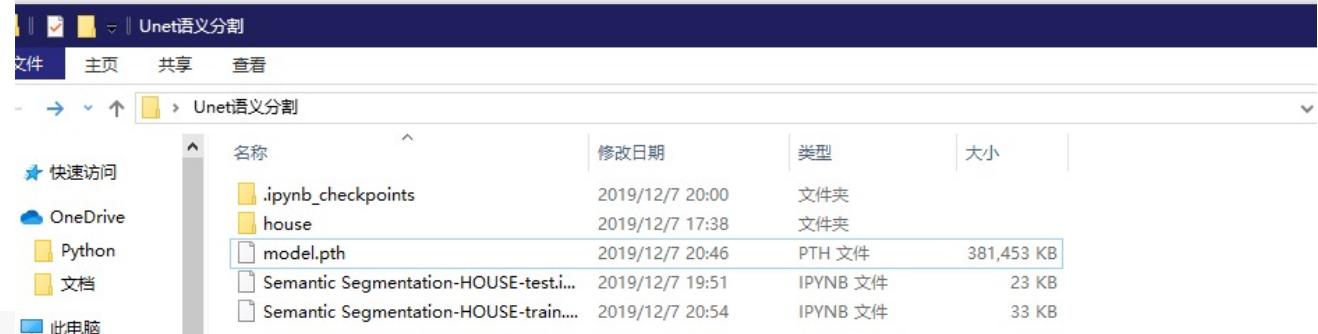
## 50+20 iterations (overfitting)

```
Loss: 0.192592 | IoU: 0.7088 | dice: 0.8182 | dice_neg: 0.0000 | dice_
Starting epoch: 17 | phase: train | ⏱: 21:34:13
Loss: 0.133003 | IoU: 0.8201 | dice: 0.8983 | dice_neg: 0.0000 | dice_
Starting epoch: 17 | phase: val | ⏱: 21:34:18
Loss: 0.189890 | IoU: 0.7115 | dice: 0.8207 | dice_neg: 0.0000 | dice_
***** New optimal found, saving state *****

Starting epoch: 18 | phase: train | ⏱: 21:34:20
Loss: 0.134696 | IoU: 0.8054 | dice: 0.8877 | dice_neg: 0.0000 | dice_
Starting epoch: 18 | phase: val | ⏱: 21:34:25
Loss: 0.187460 | IoU: 0.7170 | dice: 0.8252 | dice_neg: 0.0000 | dice_
***** New optimal found, saving state *****

Starting epoch: 19 | phase: train | ⏱: 21:34:27
Loss: 0.141069 | IoU: 0.8253 | dice: 0.9014 | dice_neg: 0.0000 | dice_
Starting epoch: 19 | phase: val | ⏱: 21:34:32
Loss: 0.186447 | IoU: 0.7184 | dice: 0.8273 | dice_neg: 0.0000 | dice_
***** New optimal found, saving state *****
```

## Read the trained model



```
iou = np.nanmean(ious)
return iou
```

[38]:

```
ckpt_path = "./model.pth"
device = torch.device("cuda")
model = smp.Unet("resnet50", encoder_weights=None, classes=1, activation=None)
model.to(device)
state = torch.load(ckpt_path, map_location=lambda storage, loc: storage)
model.load_state_dict(state["state_dict"])
#
#model = smp.Unet('resnet50', classes=1, activation=None)
```

[39]: `class Trainer(object):`

# Inspect using visdom

Apps Normal TV Essay\_tool CS ML google Bilibili YouTube Live Wikipedia MOOC Washington Post... Google Earth JupyterLab GitHub

visdom | Environment main

x o target pred x o target pred x o target pred

x o target pred x o target pred

This figure displays four pairs of aerial images and their corresponding binary masks, arranged in a 2x2 grid. Each pair consists of a 'target' image (color aerial view) and a 'pred' image (binary mask). The 'pred' images show white rectangular regions on a black background, indicating detected buildings or structures. The first two pairs show relatively simple urban areas with few buildings, while the last two pairs show more complex, densely built-up areas with many buildings.



## Test new data using trained network

```
    self.iterate("test")
```

```
In [12]: MODELPATH = ".\model.pth" # 模型路径  
IMAGEPATH = r".\test" # 图片路径
```

```
In [13]: if os.path.exists(MODELPATH):  
  
    model = smp.Unet('resnet50', classes=1, activation=None)  
    state = torch.load(MODELPATH, map_location=lambda storage, loc: storage)
```

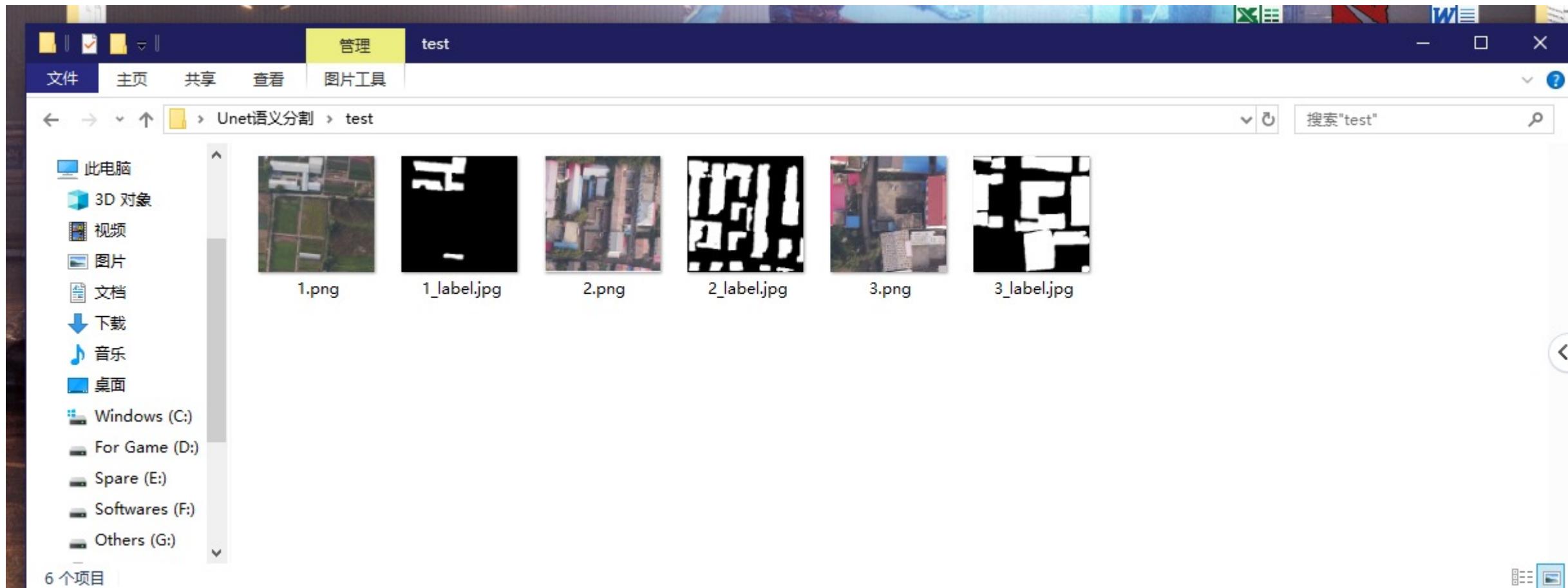
```
    model.load_state_dict(state["state_dict"])  
else:  
    model = smp.Unet('resnet50', classes=1, activation=None)  
  
    device = torch.device("cuda")  
    model.to(device)  
    model_trainer = Trainer(model)  
    model_trainer.start()
```

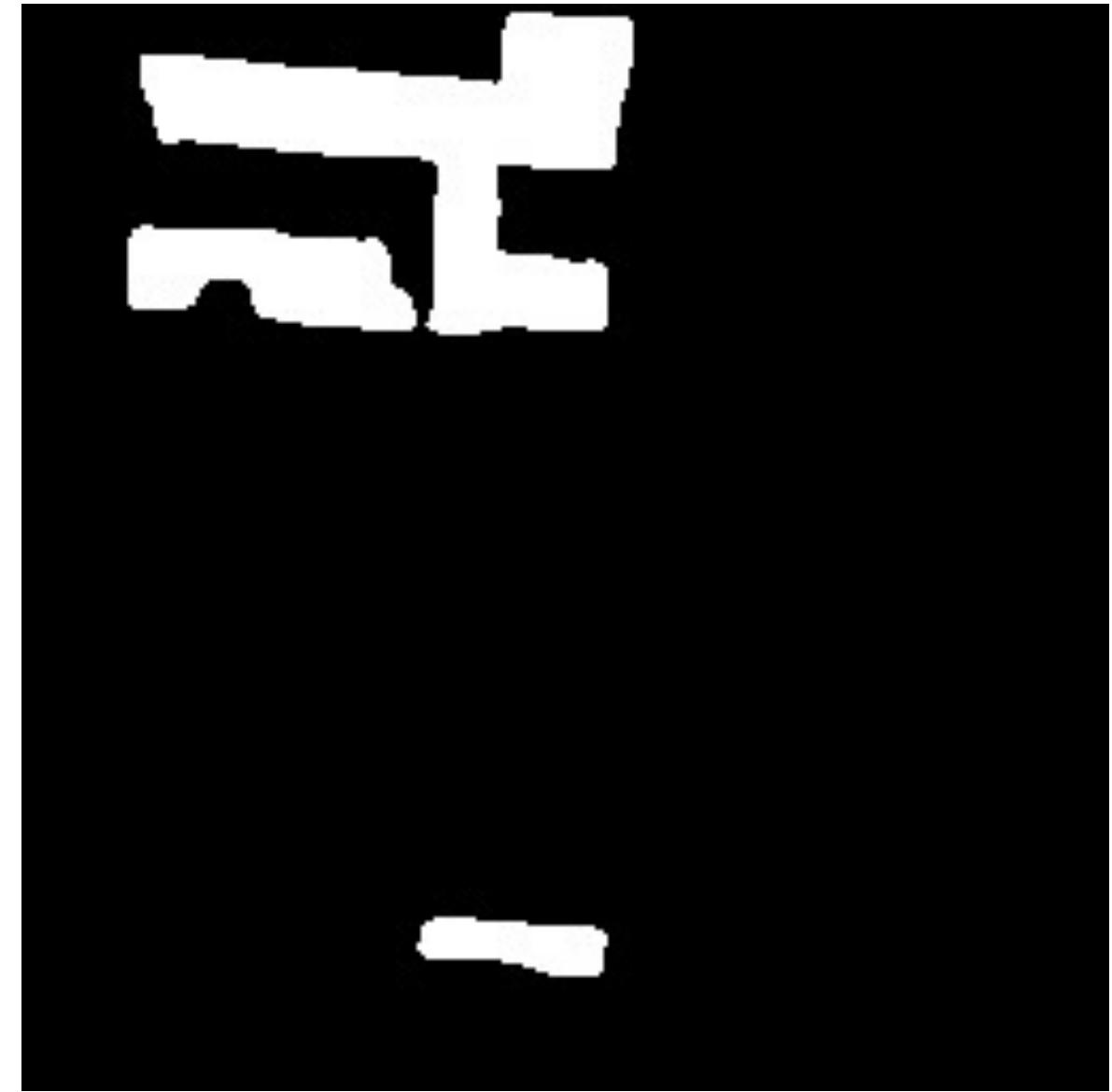
```
total images: 1  
Starting epoch: 0 | phase: test | ⏱: 21:23:43
```

```
100%|██████████| 1/1 [00:00<00:00, 19.28it/s]
```

```
In [1]:
```

## Get the labeled image





Sample 1

Sample 2





Sample 3

Learning from :

- Andrew NgDeep Learning
- Udacity – AI Programming with Python (Nanodegree Program)
- Udacity – Deep Learning course
- Yudi Tang's Tensorflow series
- Pytorch tutorials
- Resources on github

### **Acknowledgment:**

- All blogs & codes in Github

Thanks for your time