

The background features three decorative elements: a large blue circle with concentric rings in the top right, a medium-sized similar circle in the middle right, and a large blue circle with concentric rings in the bottom right. Two thin blue lines originate from the top left and extend diagonally towards the middle-right circles.

# OpenR8

Image-Segmentation-MaskRCNN-Keras

Version 20190808

LeaderG

*Contents*

I. Image-Segmentation-MaskRCNN-Keras Introduction .....4

II. Image-Segmentation-MaskRCNN-Keras Folder Introduction .....5

III. Prepare training sample image and tag categories .....7

IV. Perform 1\_train.py start training.....14

V. Perform 2\_inference.py to see the results of the training .....17

VI. Parameter introduction.....20

## Figures

Fig. 1.	Process for marking, training, and testing samples.....	4
Fig. 2.	Image-Segmentation-MaskRCNN-Keras location .....	5
Fig. 3.	Tagging web site interfaces.....	7
Fig. 4.	Press Add Files to add a new file .....	8
Fig. 5.	Select and open the profile you want to train .....	8
Fig. 6.	Use the Polygon box to select pills .....	9
Fig. 7.	Use the polygon box to select pills .....	9
Fig. 8.	Use the polygon box to select pills .....	10
Fig. 9.	New name .....	11
Fig. 10.	Enter a category name.....	11
Fig. 11.	Output tag JSON file.....	12
Fig. 12.	Place the output tag JSON file in the data\pill\train folder.....	13
Fig. 13.	Open the OpenR8 program .....	14
Fig. 14.	Select 1_train.py .....	15
Fig. 15.	Open 1_train.py.....	15
Fig. 16.	Set dataset_path .....	16
Fig. 17.	Select 2_inference.py .....	17
Fig. 18.	Open 2_inference.py .....	17
Fig. 19.	Fill in the sample path to test .....	18
Fig. 20.	Fill in the h5 sample path to be trained.....	18
Fig. 21.	Test results for 2_inference.py .....	19
Fig. 22.	Change the read h5 file name in surgery.py.....	20
Fig. 23.	Set the category name in "data\predefined_classes.txt".....	21
Fig. 24.	Change the read category JSON name in surgery.py.....	21
Fig. 25.	Increase GPU number settings in surgery.py .....	22

## *Tables*

Table 1. Image-Segmentation-MaskRCNN-Keras folder introduction .....	6
----------------------------------------------------------------------	---

## I. Image-Segmentation-MaskRCNN-Keras Introduction

Mask R-CNN is an extended application of Faster R-CNN, adding a branch more than Faster r-cnn. The target pixels are segmented while the target is being detected.

This solution uses Mask r-cnn to determine the type and location of the pill. When marking, the target uses polygons to depict the outline of the object, labeled its category.

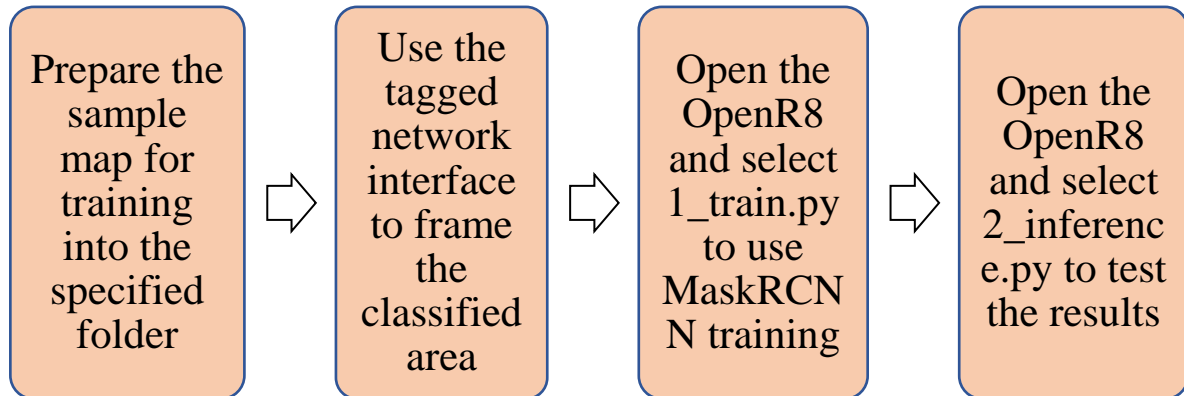


Fig. 1. Process for marking, training, and testing samples

## II. Image-Segmentation-MaskRCNN-Keras Folder Introduction

Image-Segmentation-MaskRCNN-Keras is located in the solution folder of OpenR8, which contains:

1. Folders : 【data folder】 、 【src folder】 、 【tool folder】 。
2. Py files : 【1\_train.py】 、 【2\_inference.py】 。

※For the first time, it is recommended that you change only the file content of the pill folder in data, and then make your own changes to the desired location.

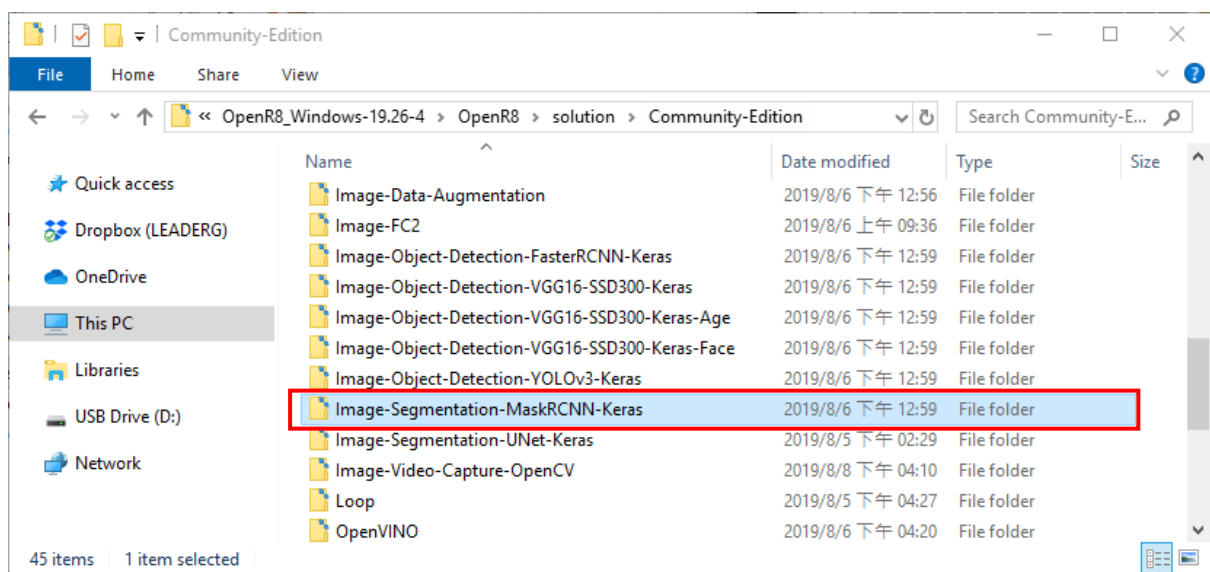


Fig. 2. Image-Segmentation-MaskRCNN-Keras location

Name	Use and function	Content
data folder	Store training sample images, categories, test images, categories, and model files. Store the model files that have been trained to complete.	pill\test 、 pill\train 、 pill\val 、 mask_rcnn_coco.h5 (Used as a sample model training (Use do not delete for the first time)) mask20181217T0933\ events.out.tfevents.1545010450.S3

		mask20181217T0933\ mask_rcnn_mask_0020.h5(Well-trained model files)
src folder	Python files used in training and inference, of which surgery.py are mainly for training and testing purposes.	real_time_detection.py split_dataset.py setup.py surgery.py
tool folder	Use a web page to mark a profile.	via-2.0.2\via.html
1_train.py	A solution for training samples.	
2_inference.py	A solution for testing samples.	

Table 1. Image-Segmentation-MaskRCNN-Keras folder introduction

### III. Prepare training sample image and tag categories

When we have to train, we have to decide the right direction first.

In this file, for example, we want to test the type of pill and where it is located, so we will sample picture one by one of their category (pill).

Step 1 : Open the tag web site interface

Open the via.html page in the via-2.0.2 folder in the tool folder to mark the sample categories we want to train.

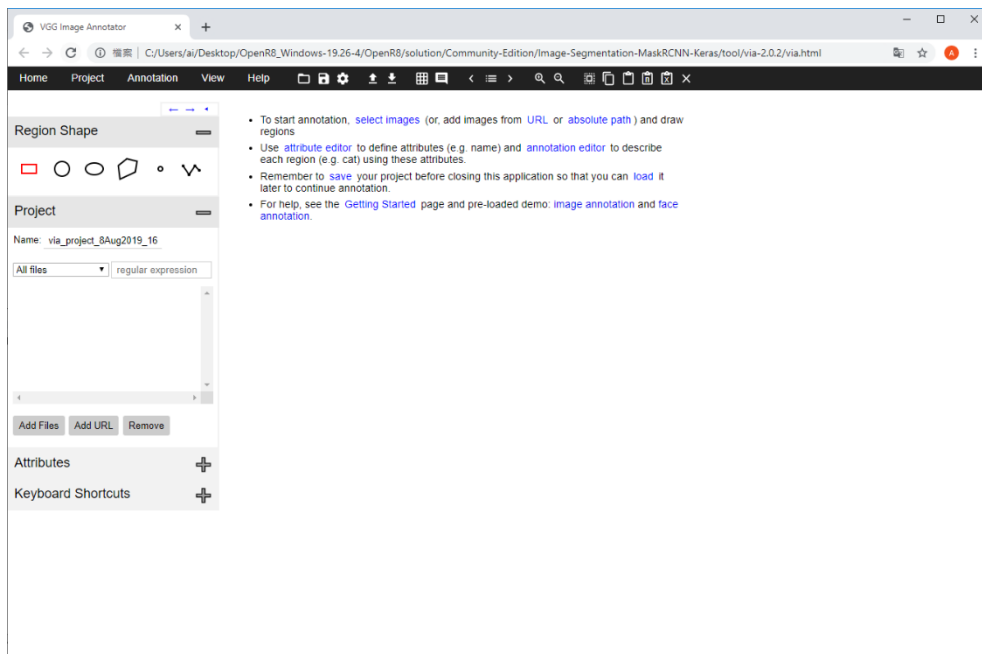


Fig. 3. Tagging web site interfaces

Step 2 : Select sample picture storage folder

Click open dir to open the folder location where the picture sample is placed.

Take the solution here, for example, to train the picture to be placed in data\pill\train, so press "Add Files" to prepare to mark the picture, as shown in Fig. 4, Fig. 5.



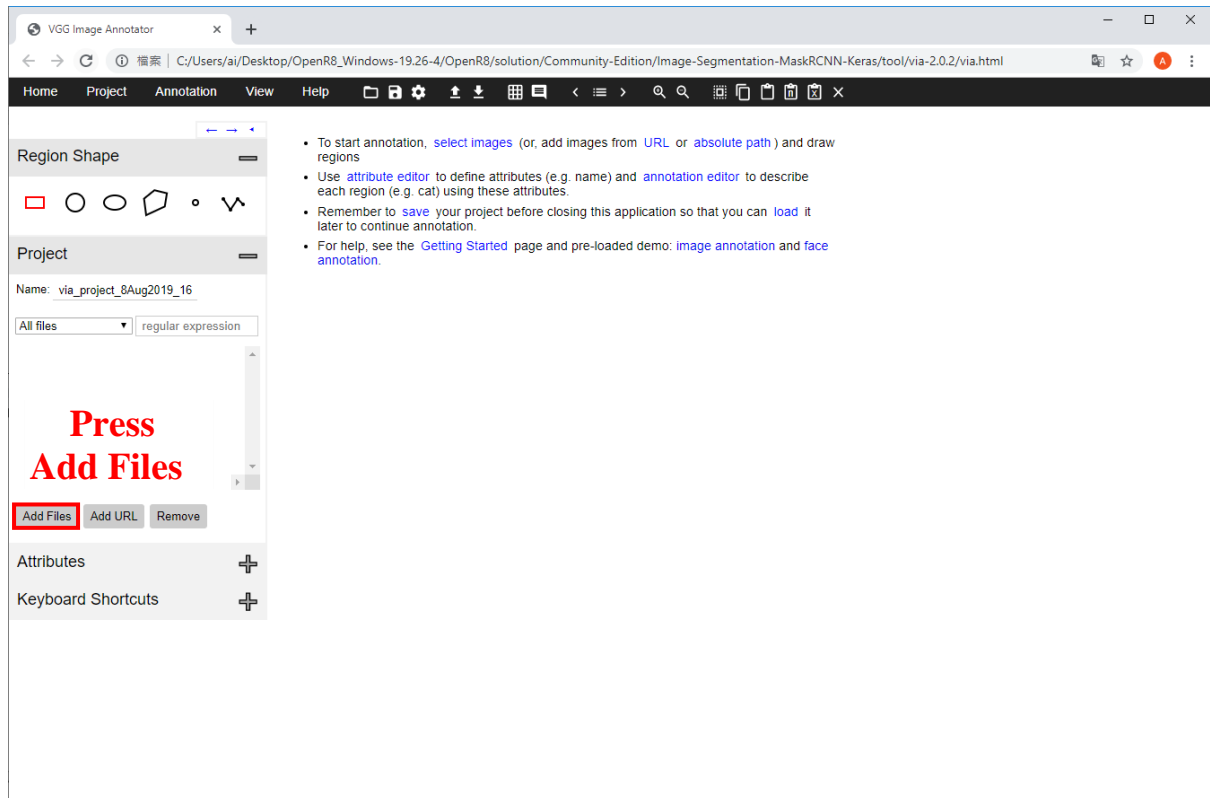


Fig. 4. Press Add Files to add a new file

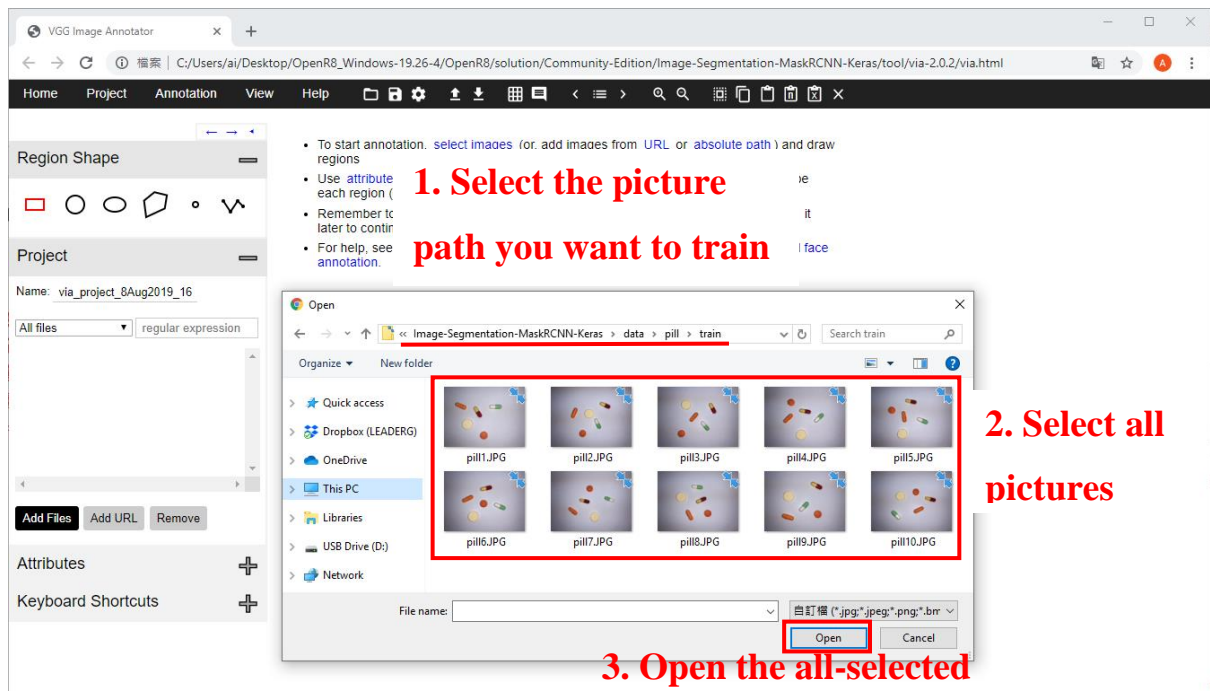


Fig. 5. Select and open the profile you want to train

### Step 3 : Box selection categories

Use polygons to depict the area you want to identify.

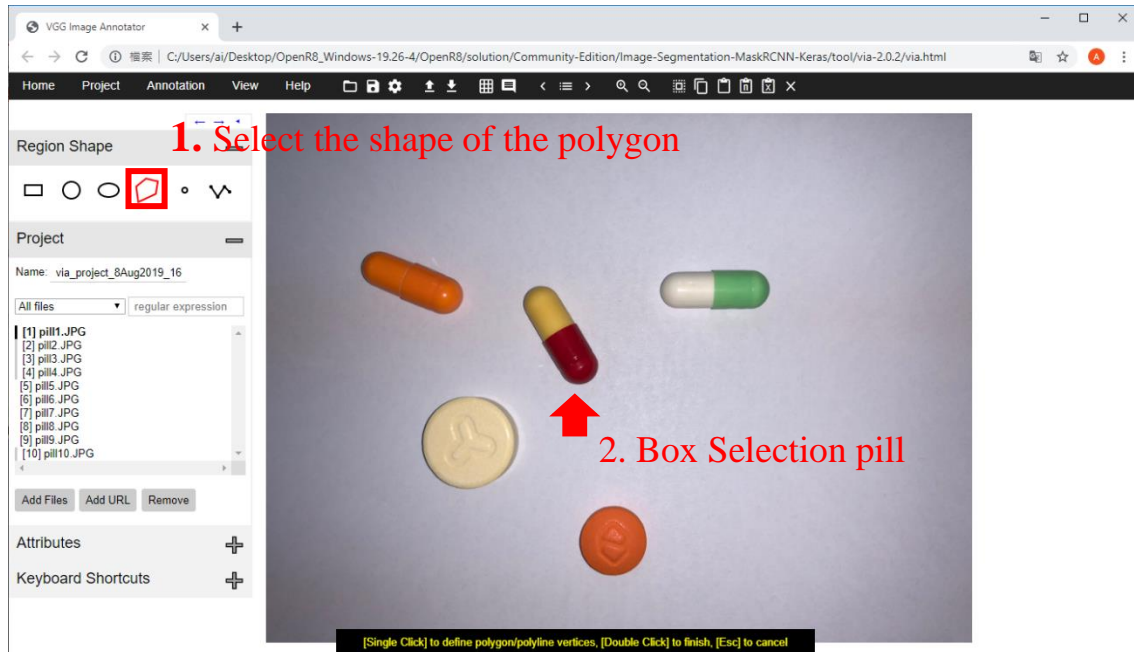


Fig. 6. Use the Polygon box to select pills

If you use the polygon box to finish, double-click the left button to end the polygon box selection, as shown in Fig. 7 、 Fig. 8.

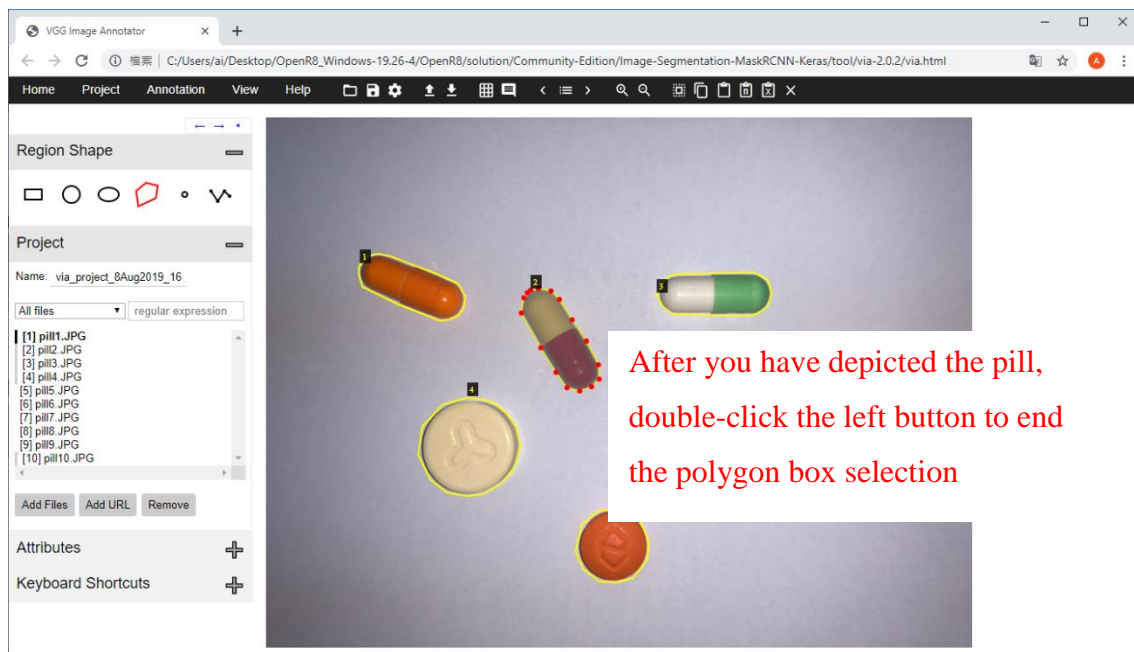


Fig. 7. Use the polygon box to select pills

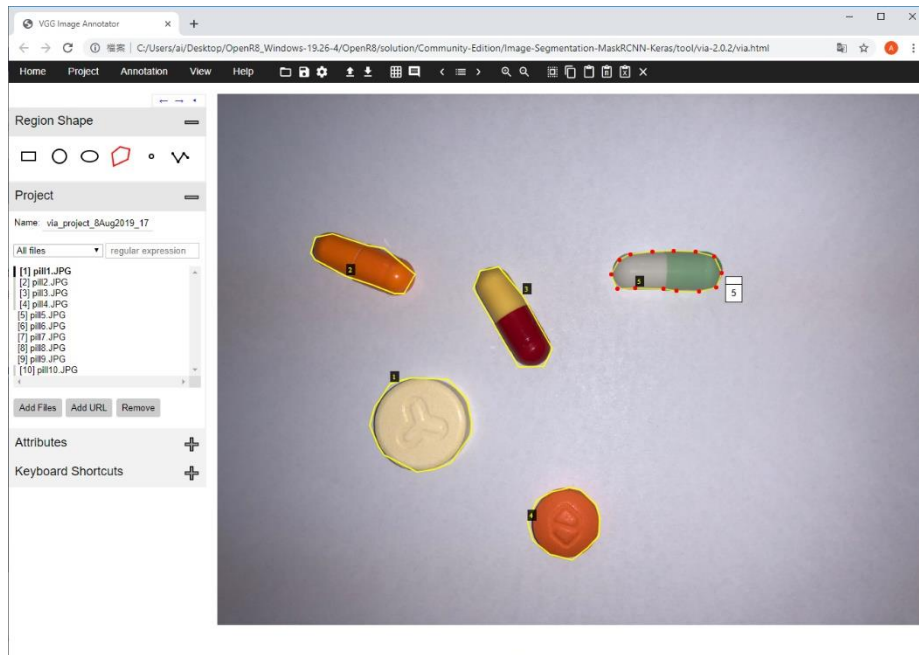


Fig. 8. Use the polygon box to select pills

Step 4 : Box to select a sample picture and mark a category

As shown in Fig. 9, enter the name in the Attributes of attribute name field. As shown in Fig. 10, fill in the category name according to the number of the box. Take this document as an example, it is to judge the pill, because there are different kinds of pills, so fill in "pill1", "pill2", "pill3" and so on, fill out can press X to close.

Continue to select the next sample picture until all sample pictures are marked with a category.

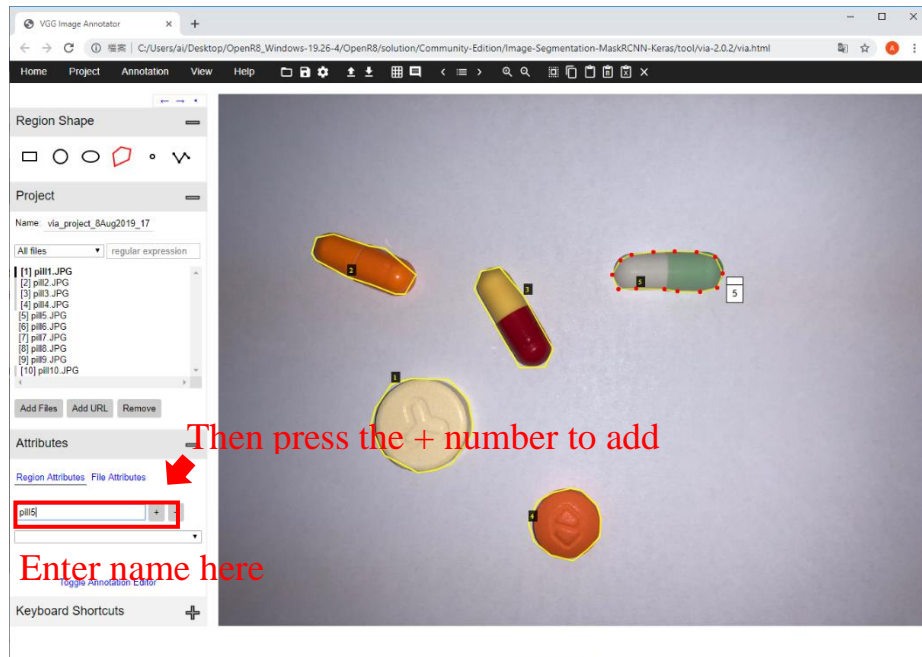


Fig. 9. New name

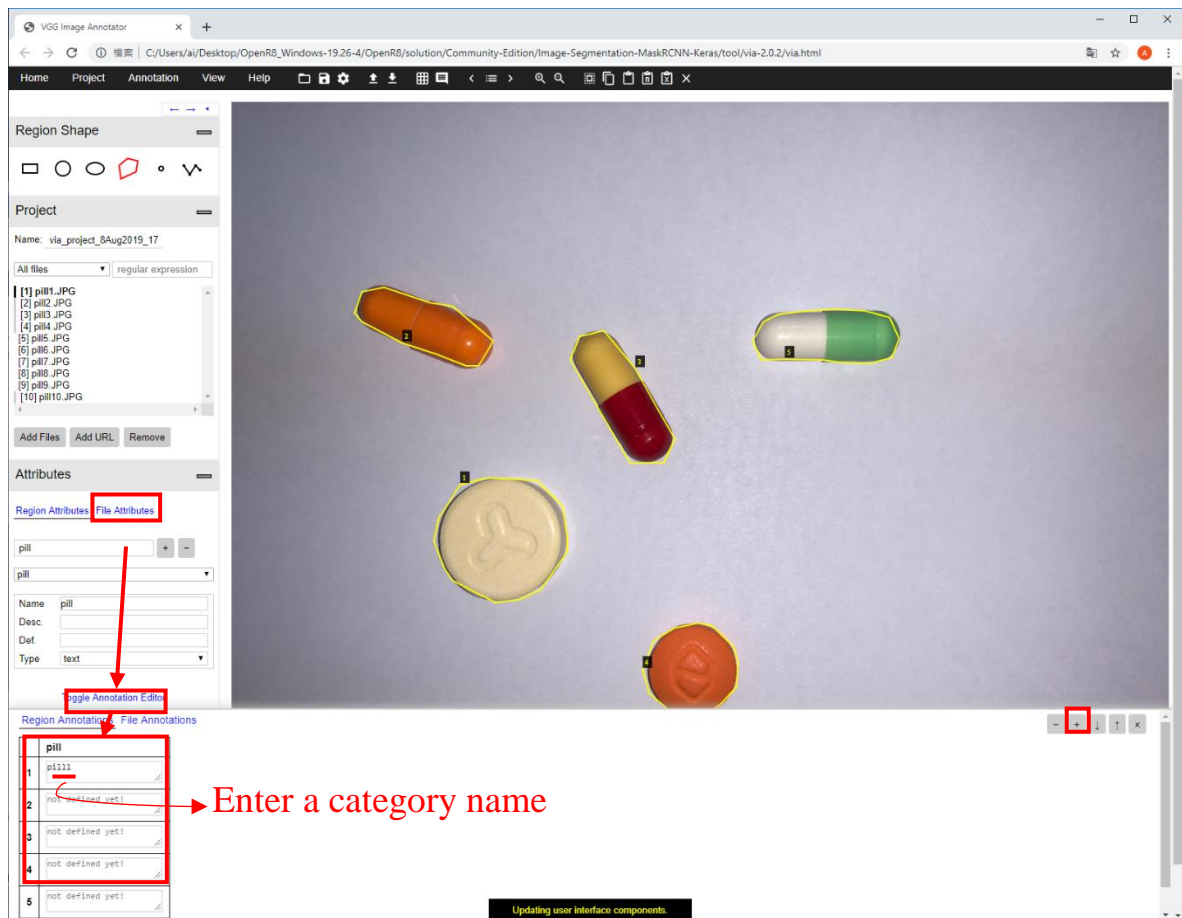


Fig. 10. Enter a category name

### Step 5 : Output tag category file

To output the tagged file after all the tags have been completed, as shown in Fig. 11. Press the export annotations (as JSON) output in the upper annotation to mark the JSON file for all picture categories.

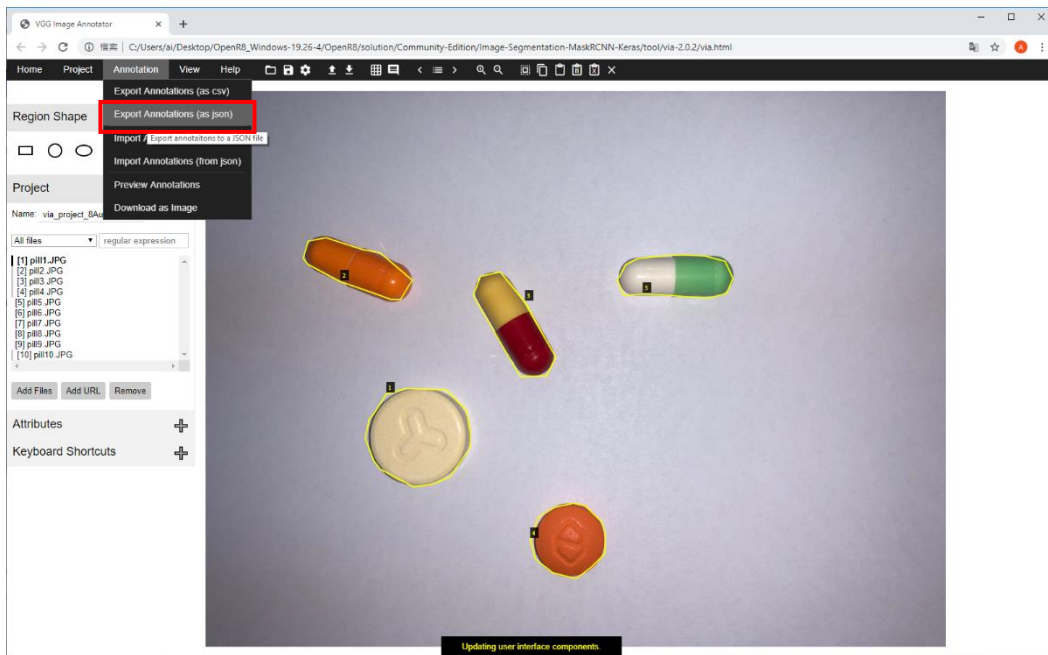


Fig. 11. Output tag JSON file

### Step 6 : Place the output JSON file in the data\pill\train folder

Place the JSON file you just output in the data\pill\train folder and confirm that the file name is "via\_region\_data.json" and, if not, rename it "via\_region\_data.json", as shown in Fig. 12.



Fig. 12. Place the output tag JSON file in the data\pill\train folder

※ To test the sample in the data\pill\val folder, do the same as the first step to sixth step.



## IV. Perform 1\_train.py start training

At the beginning, please open the "OpenR8 program", if the computer has installed display adapter, please click "R8\_Python3.6\_GPU.bat" to execute the file. If not, click "R8\_Python3.6\_CPU.bat" to execute the file, as shown in Fig. 13. After you start the "OpenR8 program", please click "File" => "Open" => "go to OpenR8 under the solution folder" => "select Image-Segmentation-MaskRCNN-Keras folder" => "select 1\_train.py open", as shown in Fig. 14 and Fig. 15.

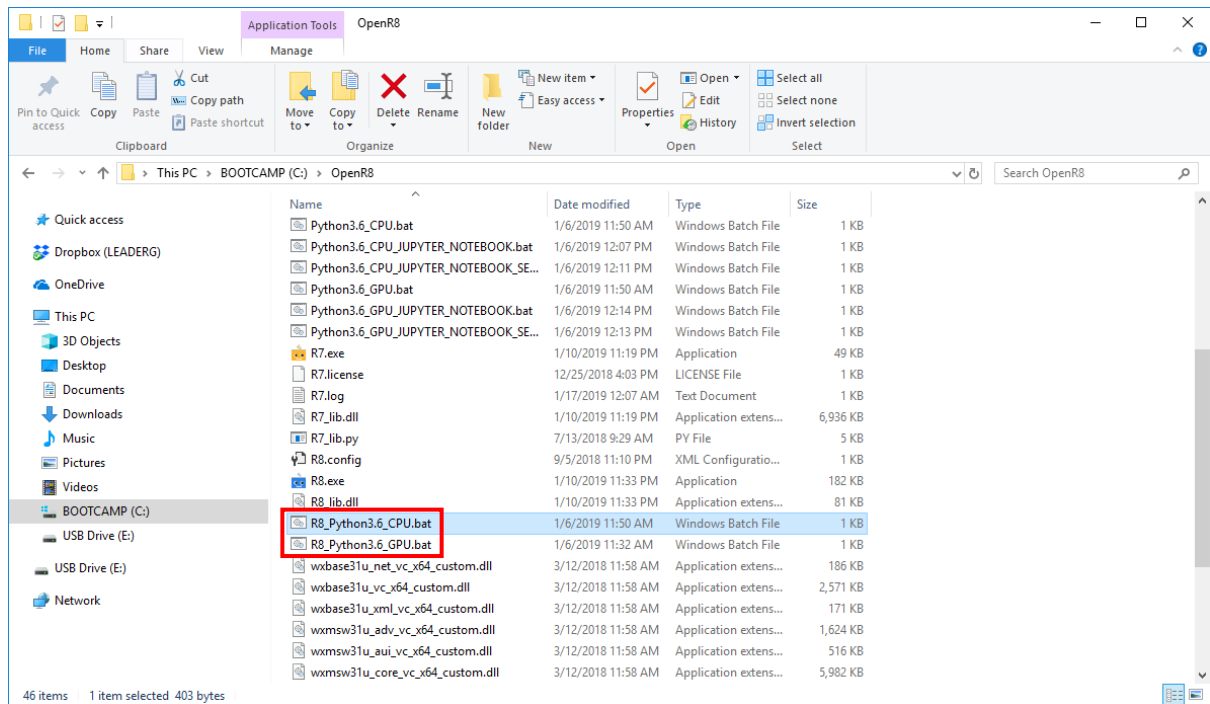


Fig. 13. Open the OpenR8 program

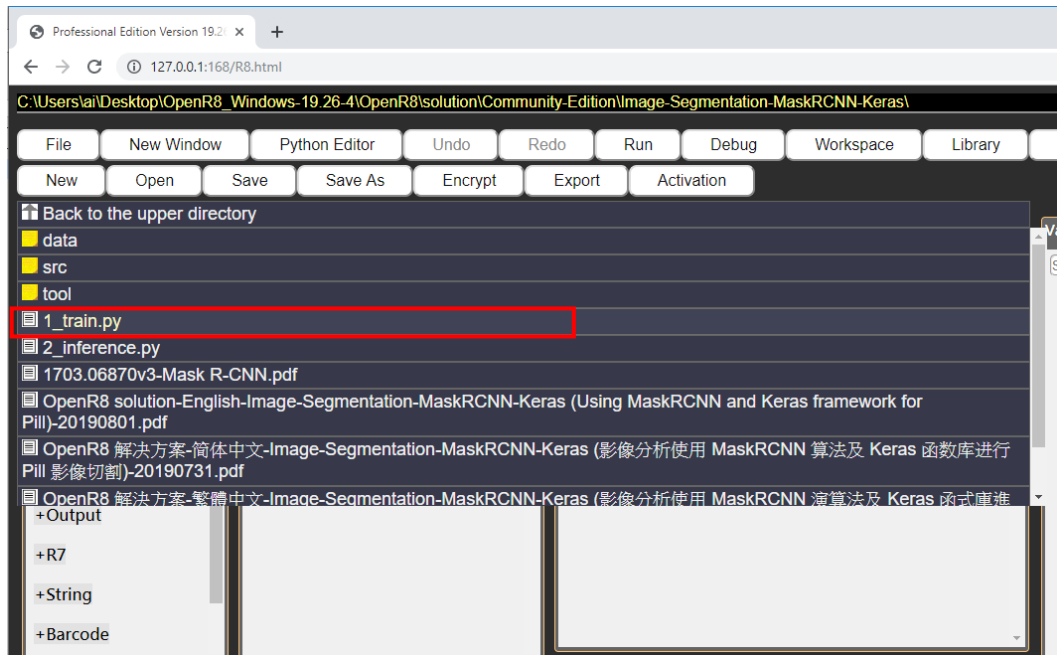


Fig. 14. Select 1\_train.py

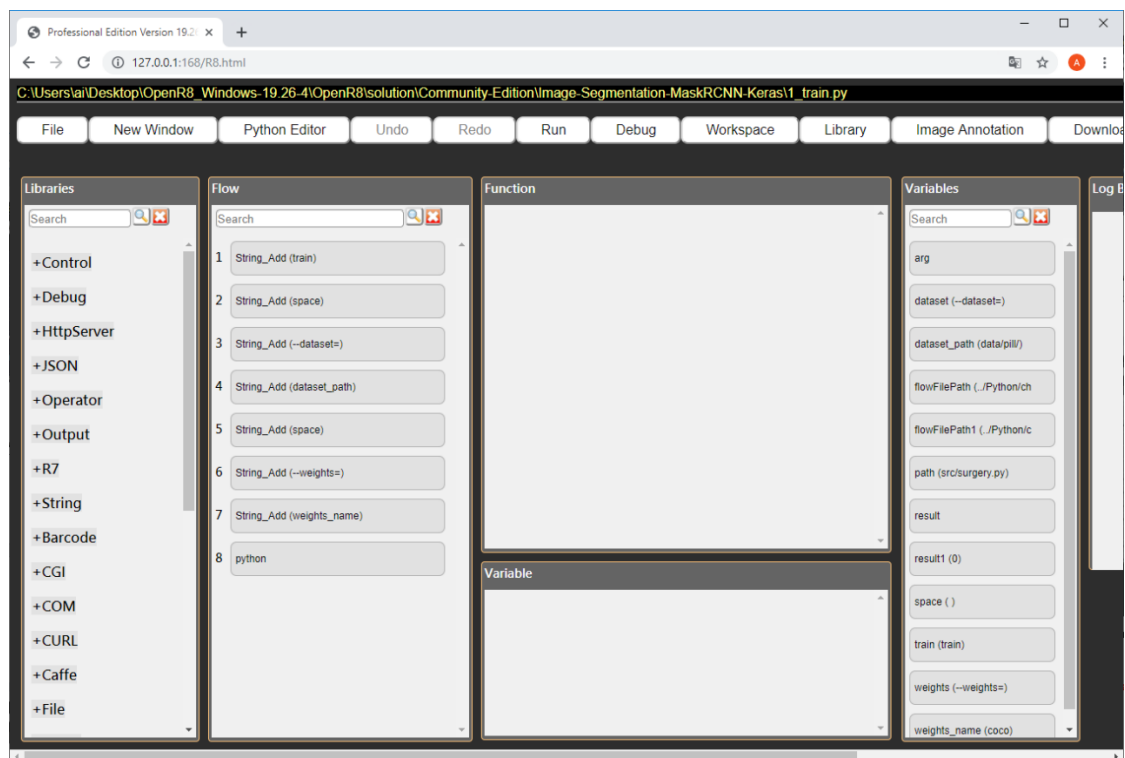


Fig. 15. Open 1\_train.py

※ If the sample image is not placed in "data\pill", an additional dataset\_path is required, as shown in Fig. 16.



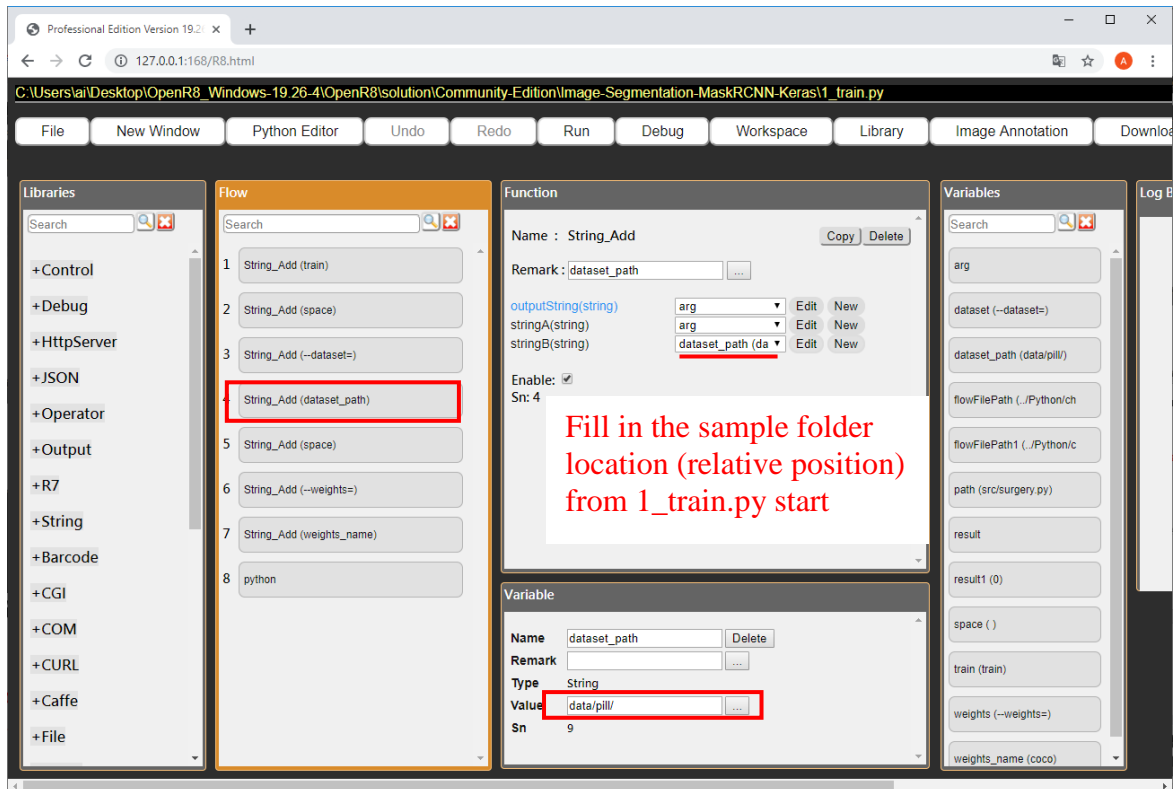


Fig. 16. Set dataset\_path

※ Before running, if you do not want to follow the previous model, please delete all h5 files (but keep mask\_rcnn\_coco.h5), unfamiliar suggestions are not deleted.

※ Before executing, if you want to change the "training model name", "training number", "classification category" ... such as parameter setting, please see chapter 6 introduction.

Press perform to start the training sample until you jump out of "press any key to continue...".

## V. Perform 2\_inference.py to see the results of the training

At the end of 1\_train.py training, open 2\_inference.py to test the picture, as shown in Fig. 17 and Fig. 18.

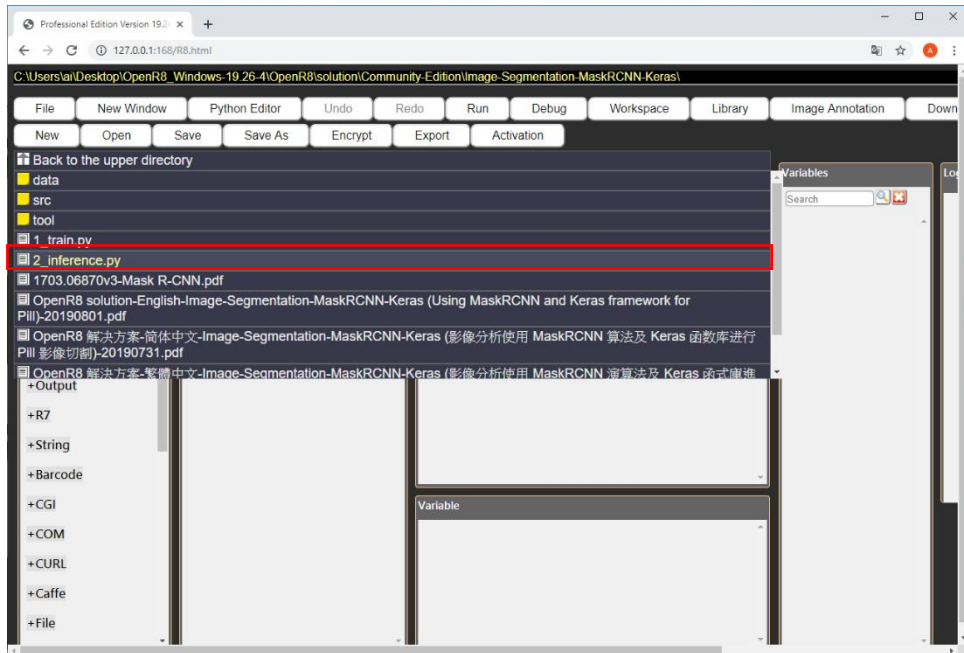


Fig. 17. Select 2\_inference.py

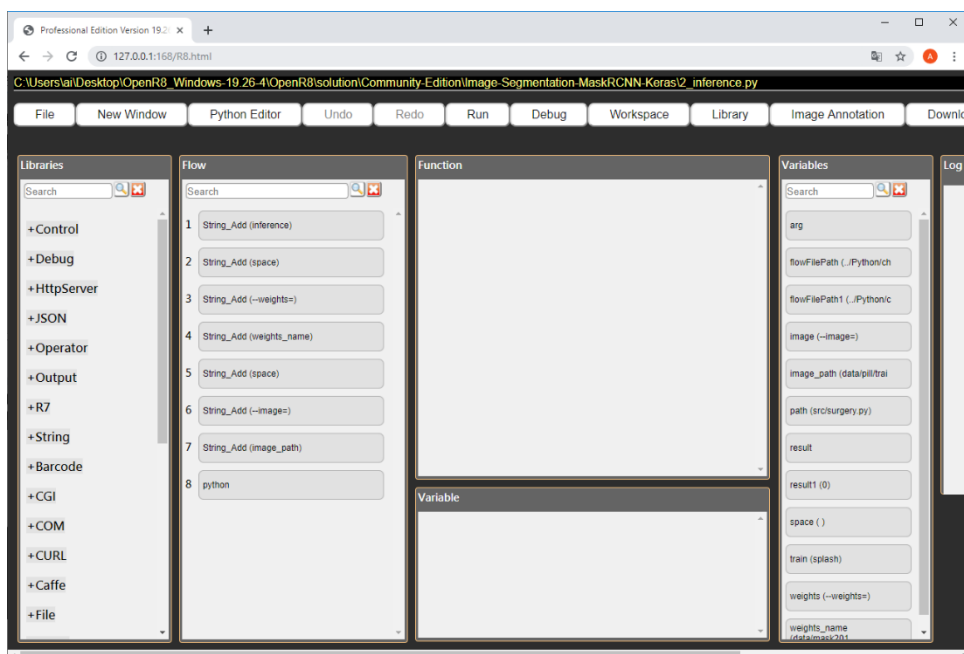


Fig. 18. Open 2\_inference.py

Fill in the sample path to be tested with the trained h5 file path, as shown in Fig. 19 and Fig. 20.

※If you have performed a 1\_train.py and successfully trained the model, be sure to confirm the Fig. 20 of the h5 file names are consistent.

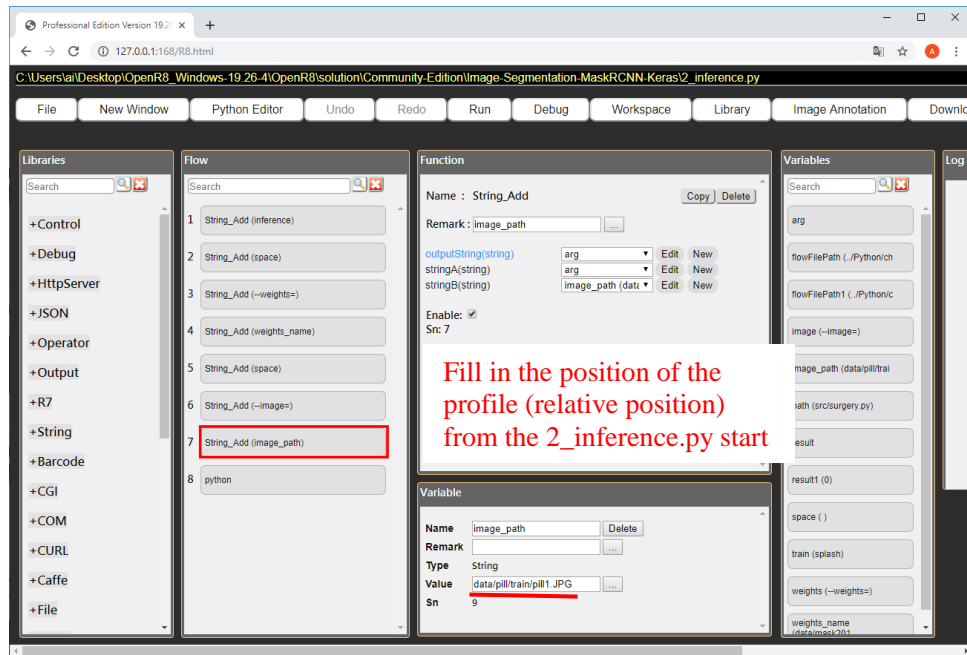


Fig. 19. Fill in the sample path to test

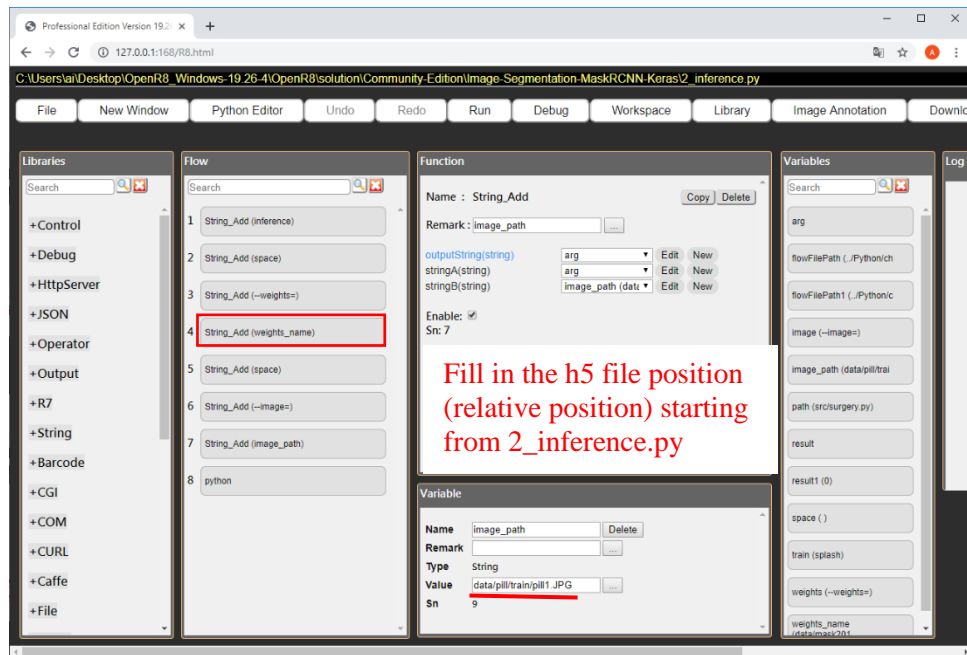


Fig. 20. Fill in the h5 sample path to be trained

Press execution to see the results, MASK\_R\_CNN and other ways to display the results are not the same, if there is a judgment to the category, that area will be marked as a color and frame up to show the category and similarity, on the contrary, if nothing is caught, there will be no mark color, as shown in Fig. 21. The position of the pill is marked separately in different colors, representing the being caught.

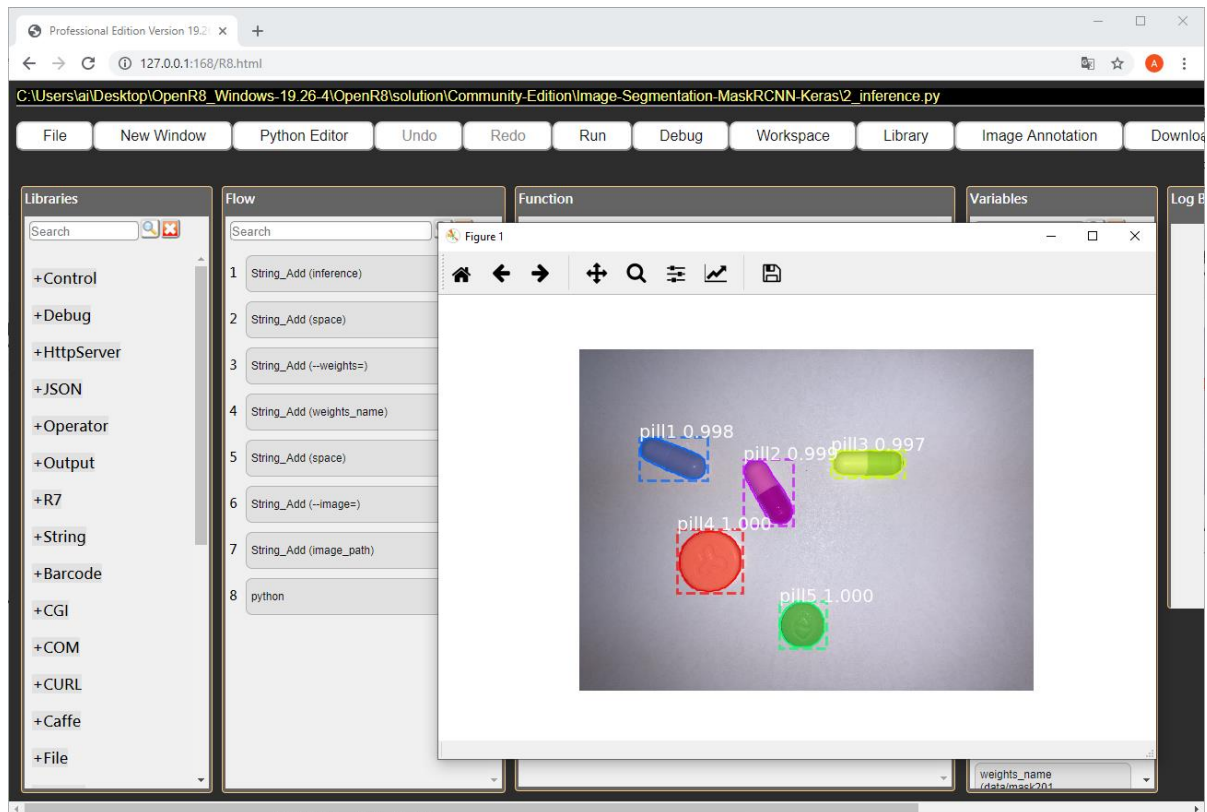
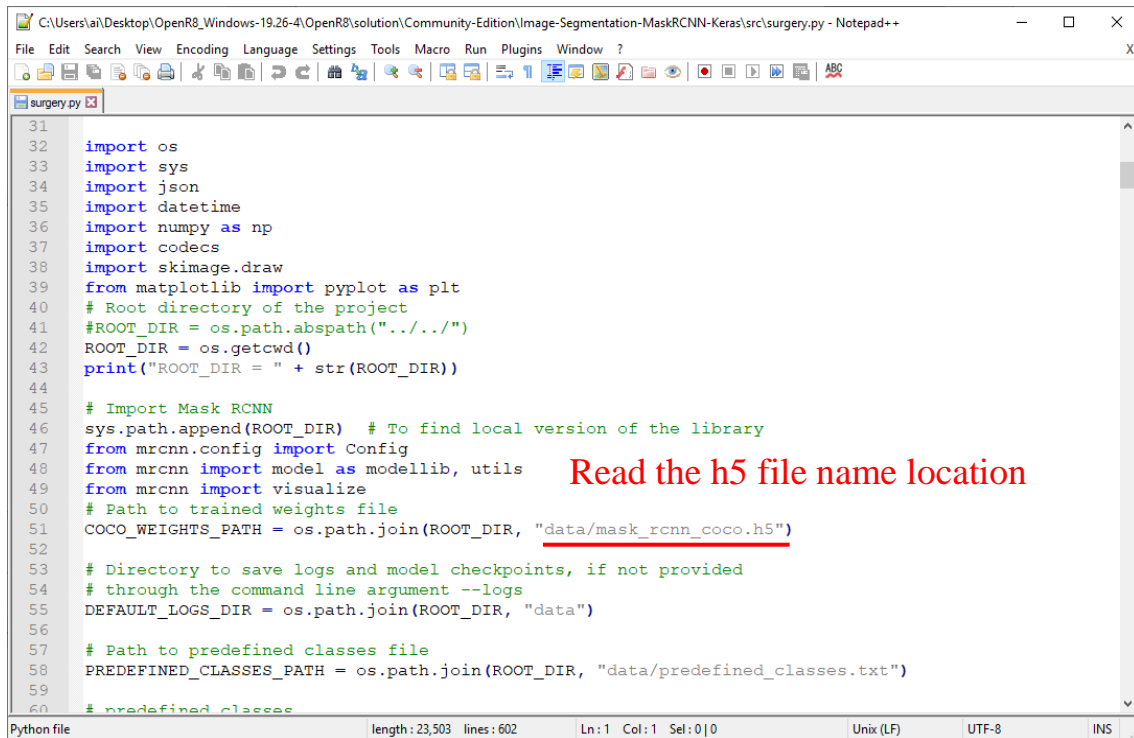


Fig. 21. Test results for 2\_inference.py

## VI. Parameter introduction

- ※ Change read h5 file name : Fig. 22 ◦
- ※ Set the category name in "data\predefined\_classes.txt" : Fig. 23 ◦
- ※ Change the JSON name : Fig. 24 ◦
- ※ Set the number of GPUs : Fig. 25 ◦



```

31
32 import os
33 import sys
34 import json
35 import datetime
36 import numpy as np
37 import codecs
38 import skimage.draw
39 from matplotlib import pyplot as plt
40 # Root directory of the project
41 #ROOT_DIR = os.path.abspath("../..")
42 ROOT_DIR = os.getcwd()
43 print("ROOT_DIR = " + str(ROOT_DIR))
44
45 # Import Mask RCNN
46 sys.path.append(ROOT_DIR) # To find local version of the library
47 from mrcnn.config import Config
48 from mrcnn import model as modellib, utils
49 from mrcnn import visualize
50 # Path to trained weights file
51 COCO_WEIGHTS_PATH = os.path.join(ROOT_DIR, "data/mask_rcnn_coco.h5")
52
53 # Directory to save logs and model checkpoints, if not provided
54 # through the command line argument --logs
55 DEFAULT_LOGS_DIR = os.path.join(ROOT_DIR, "data")
56
57 # Path to predefined classes file
58 PREDEFINED_CLASSES_PATH = os.path.join(ROOT_DIR, "data/predefined_classes.txt")
59 # predefined_classes
60

```

Read the h5 file name location

Fig. 22. Change the read h5 file name in surgery.py

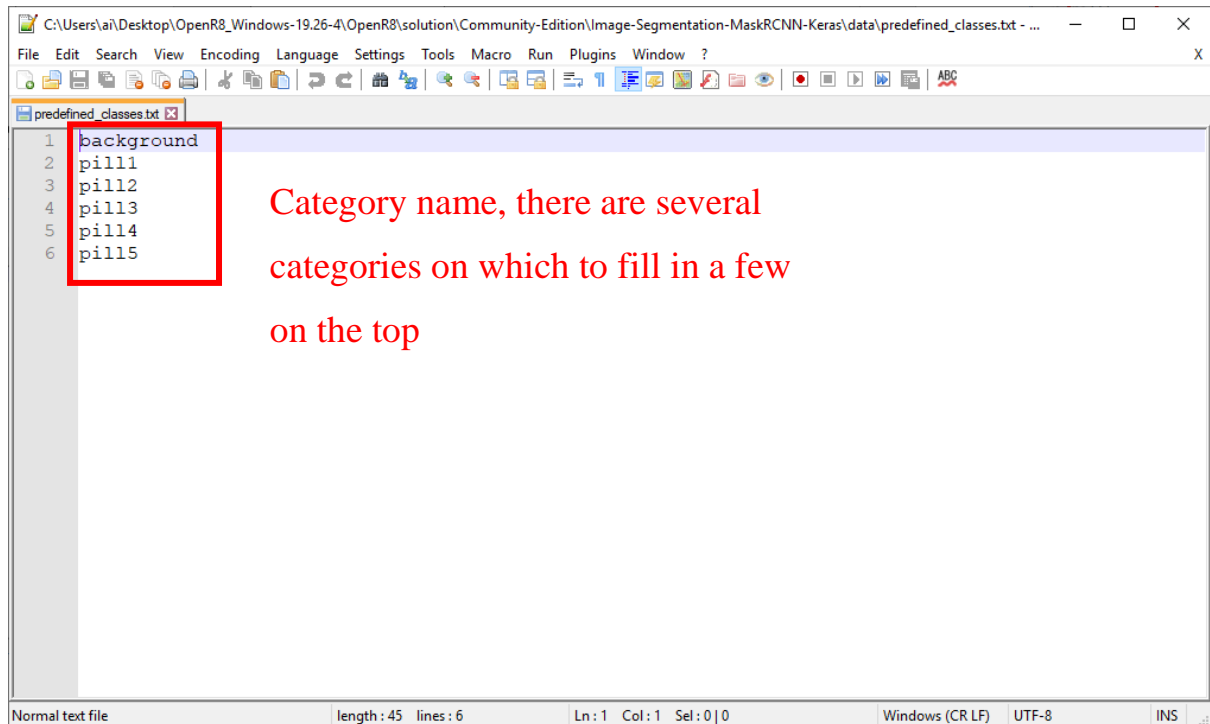


Fig. 23. Set the category name in "data\predefined\_classes.txt"

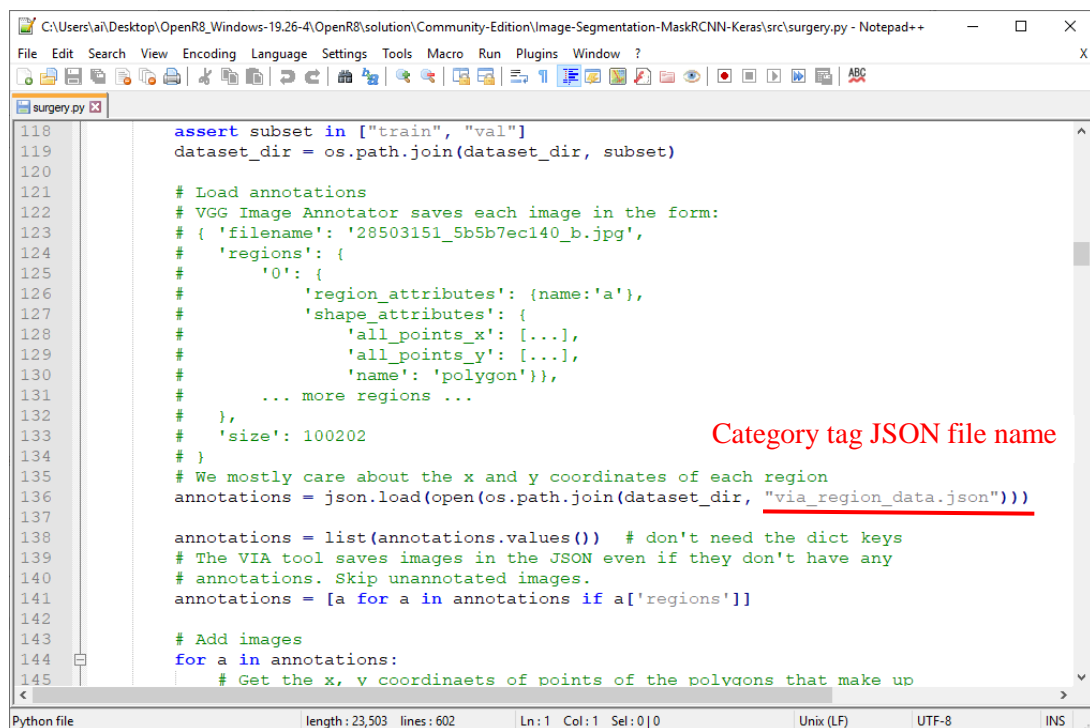


Fig. 24. Change the read category JSON name in surgery.py

```

67 #####
68
69
70 class SurgeryConfig(Config):
71     """Configuration for training on the toy dataset.
72     Derives from the base Config class and overrides some values.
73     """
74     # Give the configuration a recognizable name
75     NAME = "mask"
76
77     # We use a GPU with 12GB memory, which can fit two images.
78     # Adjust down if you use a smaller GPU.
79     IMAGES_PER_GPU = 2
80
81     GPU_COUNT = 1
82
83     # Number of classes (including background)
84     # NUM_CLASSES = 1 + 3 # Background + objects
85     NUM_CLASSES = 1
86
87     # Number of training steps per epoch
88     STEPS_PER_EPOCH = 100
89
90     # Skip detections with < 90% confidence
91     DETECTION_MIN_CONFIDENCE = 0.9
92
93
94 #####

```

GPU number setting, set to 2 when using two GPUs

Python file    length: 23,503    lines: 602    Ln: 1    Col: 1    Sel: 0 | 0    Unix (LF)    UTF-8    INS

Fig. 25. Increase GPU number settings in surgery.py