HW 2

Oscar Lin 309553018
Project repository: https://github.com/clashroyaleisgood/Course_VRDL/tree/main/HW2_Object_Detection

Competition: https://competitions.codalab.org/competitions/35888?secret_key=7e3231e6-358b-4f06-a528-0e3c8f9e328e

Colab Inference: https://drive.google.com/file/d/1RQaeVJLyXpskW6_QK5nggbJp1K8tGBL5/view?usp=sharing

Table of Contents

Introduction

This challenge is a digit characters detection task with dataset The Street View House Numbers (SVHN).
The difficulty to this challenge is the image size.
Some of the pictures are so small and even hard to detect by human.
So I use `--img 320` to enlarge the small images. It helps model to find patterns eazily.

Result

mAP:0.5:0.95: **0.41520**

19	oscar3018	18	11/25/21	309553018	0.41520 (18)

speed: **0.0948s** each image

```
[23] start_time = time.time()

     !cd yolov5; python detect.py \
                    --weights {weights} \
                    --source ../Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images \
                    --img 320 \
                    --conf-thres 0.01 \
                    --save-txt \
                    --save-conf

     end_time  = time.time()
     print("\nInference time per image: ", (end_time - start_time) / TEST_IMAGE_NUMBER)

     # Remember to screenshot!
     image 46/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2329.png: 128x320 1 1, 2 6s, 1 8, Done. (0.027s)
     image 47/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2336.png: 128x320 1 4, 1 6, Done. (0.025s)
     image 48/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2350.png: 160x320 1 1, 2 4s, Done. (0.032s)
     image 49/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2363.png: 160x320 2 2s, 1 3, 1 4, 2 5s, Done. (0.030s)
     image 50/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2407.png: 160x320 1 2, 1 3, Done. (0.030s)
     image 51/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2457.png: 160x320 1 1, 1 2, Done. (0.030s)
     image 52/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2541.png: 160x320 1 3, 1 6, Done. (0.030s)
     image 53/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2556.png: 128x320 2 2s, 1 3, 1 5, 1 6, Done. (0.028s)
     image 54/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/256.png: 128x320 1 1, 1 3, Done. (0.026s)
     image 55/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2585.png: 160x320 1 2, 1 4, 1 6, Done. (0.032s)
     image 56/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2612.png: 128x320 1 2, 1 5, Done. (0.028s)
     image 57/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2621.png: 128x320 2 2s, Done. (0.026s)
     image 58/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2642.png: 128x320 1 0, 2 1s, 1 2, 1 3, 1 5, 1 9, Done. (0.026s)
     image 59/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2647.png: 160x320 2 1s, 1 2, 1 8, Done. (0.032s)
     image 60/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2666.png: 192x320 1 5, Done. (0.034s)
     image 61/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2704.png: 160x320 1 2, 1 3, Done. (0.032s)
     image 62/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2731.png: 128x320 1 1, 2 7s, Done. (0.027s)
     image 63/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/2844.png: 160x320 1 0, 1 1, Done. (0.032s)
     image 64/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3014.png: 160x320 1 2, 1 3, Done. (0.030s)
     image 65/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3044.png: 160x320 1 6, Done. (0.030s)
     image 66/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3052.png: 160x320 1 2, Done. (0.030s)
     image 67/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3062.png: 160x320 3 1s, 1 2, Done. (0.029s)
     image 68/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3128.png: 128x320 1 0, 1 2, 1 3, Done. (0.027s)
     image 69/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3184.png: 160x320 1 2, 1 4, Done. (0.032s)
     image 70/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/319.png: 128x320 1 2, 1 3, Done. (0.028s)
     image 71/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3253.png: 128x320 2 6s, 1 7, Done. (0.025s)
     image 72/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3279.png: 128x320 1 1, 1 2, Done. (0.025s)
     image 73/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3361.png: 160x320 1 5, Done. (0.032s)
     image 74/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3391.png: 160x320 1 0, 1 1, 1 7, Done. (0.030s)
     image 75/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3402.png: 160x320 1 0, 1 4, Done. (0.030s)
     image 76/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3421.png: 160x320 1 2, 1 3, Done. (0.030s)
     image 77/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/345.png: 160x320 1 5, 1 7, Done. (0.030s)
     image 78/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3459.png: 160x320 5 1s, 3 2s, 1 6, 4 7s, Done. (0.031s)
     image 79/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3558.png: 192x320 1 1, 1 2, 1 5, Done. (0.033s)
     image 80/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3657.png: 128x320 2 2s, 1 8, Done. (0.027s)
     image 81/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3712.png: 128x320 3 0s, 1 1, 1 3, Done. (0.024s)
     image 82/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/372.png: 160x320 1 0, 1 2, Done. (0.033s)
     image 83/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3734.png: 96x320 2 1s, 1 6, 2 8s, 1 9, Done. (0.029s)
     image 84/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/3741.png: 128x320 1 3, 1 8, 1 9, Done. (0.027s)
     image 85/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/397.png: 128x320 1 3, 1 8, Done. (0.025s)
     image 86/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/401.png: 160x320 2 3s, 1 7, Done. (0.032s)
     image 87/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/499.png: 96x320 1 1, 1 2, Done. (0.028s)
     image 88/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/521.png: 96x320 2 0s, 2 2s, 1 4, 4 5s, 4 6s, Done. (0.025s)
     image 89/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/564.png: 128x320 1 3, 1 5, 1 7, Done. (0.028s)
     image 90/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/570.png: 128x320 2 4s, 2 9s, Done. (0.026s)
     image 91/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/581.png: 160x320 1 0, 1 2, Done. (0.033s)
     image 92/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/594.png: 128x320 1 1, 1 2, Done. (0.027s)
     image 93/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/621.png: 160x320 1 1, 1 4, 1 5, Done. (0.032s)
     image 94/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/757.png: 128x320 2 2s, 1 4, 1 6, Done. (0.027s)
     image 95/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/764.png: 192x320 1 3, 1 9, Done. (0.033s)
     image 96/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/776.png: 96x320 2 6s, Done. (0.027s)
     image 97/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/850.png: 160x320 1 4, 1 8, Done. (0.032s)
     image 98/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/880.png: 160x320 1 0, 1 2, 1 6, 1 9, Done. (0.030s)
     image 99/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/983.png: 96x320 2 2s, 1 4, 1 5, Done. (0.029s)
     image 100/100 /content/Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/987.png: 160x320 2 1s, 1 2, 2 6s, 1 7, 1 9, Done. (0.032s)
     Speed: 0.2ms pre-process, 29.5ms inference, 1.4ms NMS per image at shape (1, 3, 320, 320)
     Results saved to runs/detect/exp10
     100 labels saved to runs/detect/exp10/labels

     Inference time per image:  0.09480819940567016
```

Data Pre-Processing

In this challenge I only do the label format transform from .mat file to the format which yolov5 accepts.
with help of this website: https://www.vitaarca.net/post/tech/access_svhn_data_in_python/
Architechture yolov5 needs:

```
Course_VRDL/
    ├── images/
    │   ├── train/
    │   │   ├── 1.png
    │   │   └── 2.png...
    │   ├── valid/
    │   └── garbage/ <-- mentioned below
    │
    └── labels/
        ├── train/
        │   ├── 1.txt
        │   └── 2.txt...
        └── valid/
```

And pick the labels(and corresponding image) with problems out of training images, to prevent getting strange training result because of bounding box getting out of range of the image.

Model architecture

I use powerful object detection model: yolov5 and its pretrained weights.
(The model version I use is yolov5m, where the "m" means the median, the others are s for small, l for large, and x for extra-large)

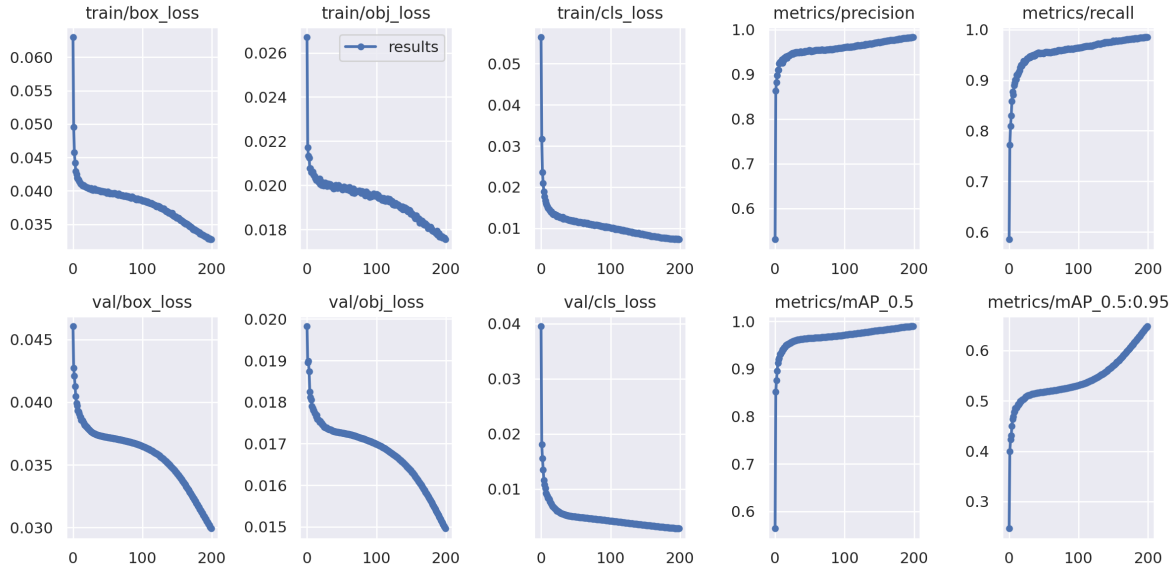
Hyperparameters

```
img_size: 320
epoch: 400
validation_percentage: 1/5
```

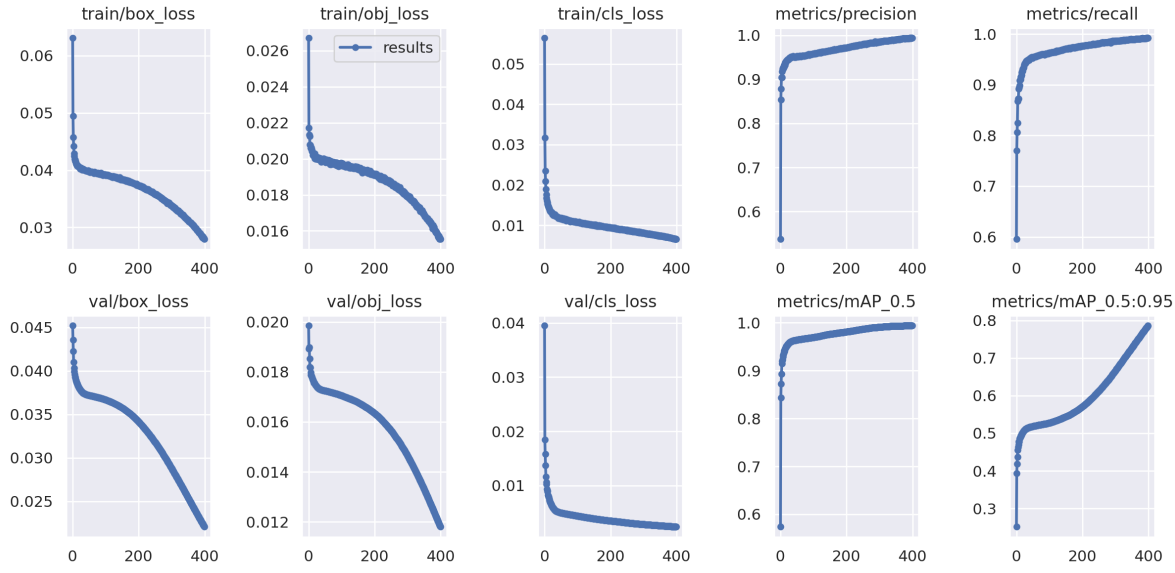
Experiments

Firstly I use `--freeze` flag to reduce the learning time and keep the well pretrained previous layers, But result are not good.
Until I gradually reduce the freeze layer number to **0**, the results are getting better and better.
I discovered that, It may take time to train the whole model rather than train the last few layers, but the result will be better because we can train the model to the detail part(first few layers).
The pretrained one is good enough, but it's not fit to this task, so we still need to modify the first few layers to get more details about our task.

Then I got struggle with the overfitting problem I guess.
I have very strange result in my training, both train/valid loss reduce in the begining, then getting smooth.
But suddenly reduce again after aound half of the epochs.
The reduction is getting more and more, and sharply droped at the very end like this:



same situation happened when I enlarge the epochs to 400



So it's not because of the not yet converge of the model

I have very high mAP in train and valid due to this, but even lower mAP in testing data.
Then I realized that this is probably an overfitting, though it confuse me a lot that, why the validation data is also getting lower...
So I finally decided to **manually** early stop by training with 100 epochs and the same hyperparameters before.
I early stoped(collect the weights/best.pt rather than directly stop the training) at 33, 40, and 67.
Stopping at **33** gives me the highest score at that moment, and **40** also gives me a higher score.
Things are getting better and better.
But the result of **67** was worse than **40**, then I known that the loss of train/valid is getting lower again and the strange overfitting problem is happening again.

Summary

In this chalenge, I use yolov5 and the pre-trained weights and finally get **mAP: 0.41520** on testing data provided by the competetion.

The strange overfitting problem was bothering me a lot.
Though I use **manually early stop** to prevent the problem, but it's not a good solution at all, I still can't figure out what is happening then.

It's not a good way but still a effective way using manually earlly stop.

About inference fairness

Because I use yolov5 model, the structure is so complicated that I can't turn command-line detection into python code detection. So, I can only use command-line detection like `python detect.py --weights weight.pt --source folder/to/images`.
I have also try to combine python for loop with command-line detection like...

```
# Test your inference time 2

test_image_folder = r'Course_VRDL/HW2_Object_Detection/HW2_dataset/test_images/'
image_list = os.listdir(test_image_folder)
image_list.sort(key = lambda x: int(x[:-4]))

start_time = time.time()
for img_name in image_list[:TEST_IMAGE_NUMBER]:
    img_path = os.path.join(test_image_folder, img_name)
    !cd yolov5; python detect.py --weights {weights} --source ../{img_path} --img 320 --conf-thres 0.01 --save-txt --save-conf

end_time  = time.time()
print("\nInference time per image: ", (end_time - start_time) / TEST_IMAGE_NUMBER)

# Remember to screenshot!
```

But there is another unfairness happened, I have to load model and weights times and times, and getting really bad speed performance.

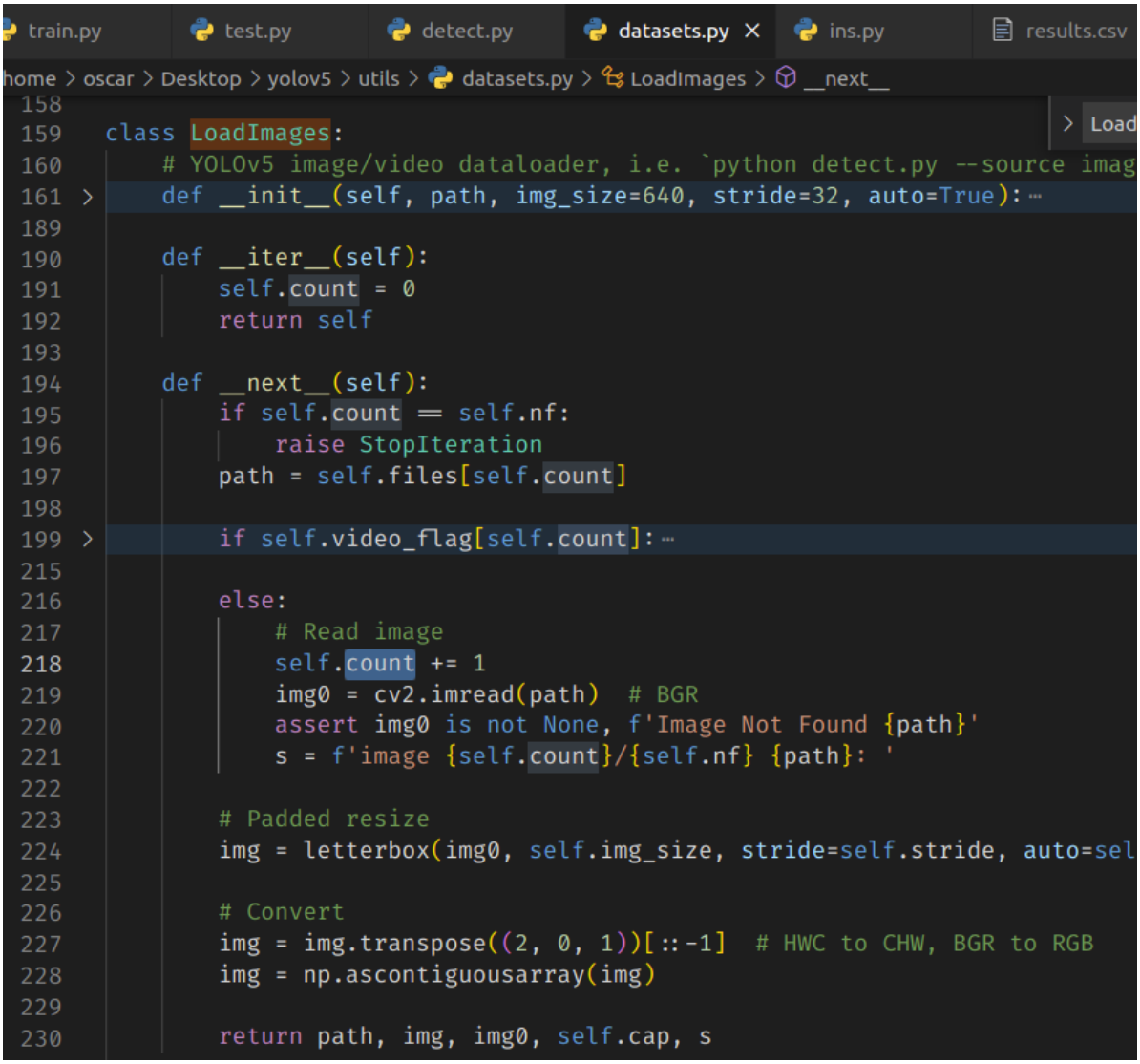
After trying this, I try to read the original `yolov5/detect.py` to see how it work inside, and I found that:

train.py | test.py | detect.py | datasets.py | ins.py

home > oscar > Desktop > yolov5 > detect.py > run

```
84      half &= pt and device.type != 'cpu'  # half precision on
85      if pt:
86          model.model.half() if half else model.model.float()
87
88      # Dataloader
89      if webcam:
90          view_img = check_imshow()
91          cudnn.benchmark = True  # set True to speed up const
92          dataset = LoadStreams(source, img_size=imgsz, stride
93          bs = len(dataset)  # batch_size
94      else:
95          dataset = LoadImages(source, img_size=imgsz, stride=
96          bs = 1  # batch_size
97      vid_path, vid_writer = [None] * bs, [None] * bs
98
99      # Run inference
100     if pt and device.type != 'cpu':
101         model(torch.zeros(1, 3, *imgsz).to(device).type_as(n
102     dt, seen = [0.0, 0.0, 0.0], 0
103     for path, im, im0s, vid_cap, s in dataset:
104         t1 = time_sync()
105         im = torch.from_numpy(im).to(device)
106         im = im.half() if half else im.float()  # uint8 to f
107         im /= 255  # 0 - 255 to 0.0 - 1.0
108         if len(im.shape) == 3:
109             im = im[None]  # expand for batch dim
110         t2 = time_sync()
111         dt[0] += t2 - t1
```

detector take a path once from `dataset`(LoadImage class)



LoadImage class is also load a image(`cv2.imread(path)`) once the **next**(self) is called by for loop

So I thought it's ok for me to use `python detect.py` with `--source path/to/img/folder` (? or I don't have any solutions at all.