**Progress Report**

1. **Environment Building**

The runtime environment of mine for YOLO v5 is windows + anaconda3 + YOLO v5-3.1, which requires the related software include python 3.7, torch 1.6.0, torchvision 0.7.0, and cuda 10.1. As different versions of YOLO v5 require a different version of torch, so please download the WHL files of corresponding versions of torch and torch vision on the [website](https://download.pytorch.org/whl/torch_stable.html). You may follow this [blog](https://blog.csdn.net/handsomeandge/article/details/116238172?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163160109916780255234771%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163160109916780255234771&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-3-116238172.first_rank_v2_pc_rank_v29&utm_term=yolov5+anaconda&spm=1018.2226.3001.4187) to finish the installation in the win10 environment.

1. **Data preparation**

1.1 Data information

[FLIR](https://www.flir.com/oem/adas/adas-dataset-form/) provides annotated thermal imaging dataset and corresponding non annotated RGB images for training and verifying neural networks. The data is obtained by an RGB camera and thermal imaging camera installed on the vehicle.

The data set contains 14452 infrared images in total, of which 10228 are from multiple short videos; 4224 from a 144s video. All videos come from streets and highways. The sampling rate of most pictures is two frames per second, and the frame rate of the video is 30 frames per second; When there are few targets in a few environments, the sampling rate is 1 second and 1 frame.

Label with MSCOCO label vector. The details are as follows:

* Category 1: People
* Category 2: Bicycle - bicycles and motorcycles (inconsistent with coco)
* Category 3: Cars - private cars or other small commercial vehicles.
* Category 18: Dogs
* Category 91: other vehicles - large trucks, boats, trailers

The folder structure is as follows, including 3 folders. Each folder contains five sub-files.

* train: contains 8862 sampled images;
* val: including 1366 pictures;
* video: contains a 144s video;

The contents of each subfile are as follows:

* Annotated\_ thermal\_ 8\_ Bit: contains the marked image data;
* thermal\_ Annotations.json: annotation data in MSCOCO format; Additional data is included in extra info;
* thermal\_ 16\_ Bit: image;
* thermal\_ 8\_ Bit: image;
* RGB: visible image;

1.2 Data preparing

Before using the dataset to train YOLO v5, some preparations need to be made for the downloaded data files:

* The dataset was split and compressed into 15 xx.zip.0\* files. Extracting one file cannot extract the dataset’s contents. Therefore, all 15 files need to be downloaded and placed in the same directory. Then open CMD under this directory and enter “copy /B xx.zip.01+xx.zip.02+… xxx.zip”, you’ll see the combined compressed file named xxx.
* The data format of the annotation is based on the coco dataset, that is, it is saved in the. JSON file. To train it on YOLO v5, we need to convert the .JSON format of the YOLO format first. You may follow this [blog](https://blog.csdn.net/hello_levy/article/details/105212876?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163160307816780269866847%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163160307816780269866847&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-6-105212876.first_rank_v2_pc_rank_v29&utm_term=flir%E6%95%B0%E6%8D%AE%E9%9B%86&spm=1018.2226.3001.4187) to finish the conversion. You may click [ipynp](https://github.com/chenbinluo/Learning_target_detection) to download the notebook version of the code.
* Because YOLO v5 obtains the path of the annotations(labels) according to the path of the images, we need to change the folder’s structure to follows:
  + - FILR\_dataset
      * images
        + train
        + val
      * labels
        + train
        + val

1. **Training YOLO v5 with FLIR dataset**

2.1 Steps

* Step 1. Download [Yolo V5 code](https://github.com/ultralytics/yolov5/), unzip the file;
* Step 2. Open CMD under the folder yolov5\_master, activate the environment, and then enter “pip install - r requirements.txt”;
* Step 3. Modify models / yolov5s.yaml and change nc = 80 to nc = 4 (number of categories)
* Step 4. Create a new [FLIR.yaml](https://github.com/chenbinluo/Learning_target_detection/blob/main/FLIR.yaml) file under the data folder
* Step 5. Train from scratch: Python train.py -- img 640 -- epochs 3 -- data / FLIR.yaml -- CFG models / yolov5s.yaml -- weight "";
* Step 6. Fine tune from the pre training model: Python train.py -- img 640 -- epochs 300 -- data / FLIR.yaml -- CFG models / yolov5s.ya Python train.py -- img 640 -- epochs 3 -- data / FLIR.yaml -- CFG models / yolov5s.yaml -- weight "";ml -- weight weights / yolov5s.pt;

2.2 some results



Figure 1. Counting

|  |  |  |
| --- | --- | --- |
| precision | recall | mAP\_0.5 |
| 0.77732 | 0.55753 | 0.54298 |

Table 1. Metrics

1. **Training YOLO v5 with own dataset**

3.1 Steps

* Go to Google to download the corresponding image or write a simple image crawler;
* Open CMD in the folder yolov5\_master, activate the environment, and then enter “pip install - R requirements.txt”;
* Then enter “pip install labelimg”;
* Enter labelimg, and manually label the pictures and save them as XML files according to the steps of “open dir > create box > save”;
* [Split](https://github.com/chenbinluo/Learning_target_detection/blob/main/split.py) the XML file into training tests and [convert it into Yolo label data](https://github.com/chenbinluo/Learning_target_detection/blob/main/txt2yolo_label.py);
* Add a new [.yaml](https://github.com/chenbinluo/Learning_target_detection/blob/main/FLIR.yaml) file under the data folder to describe the images and labels of own dataset’s path and the number of categories;
* Before training the model, create a new weights folder under yolov5\_master, go to the [web page](https://github.com/ultralytics/yolov5/releases) to download the model’s pt file, and put it under the weight folder;
* Change the “nc” in the yaml file under the model folder to the number of categories just marked by labelimg;
* Then enter Python train.py -- img 640 -- batch 4 -- epoch 300 -- data. / data / myvoc.yaml -- CFG. / Models / yolov5m.yaml -- weights / yolov5m.pt -- workers 0 in CMD to start training;
  1. some results



Figure 2. Classification

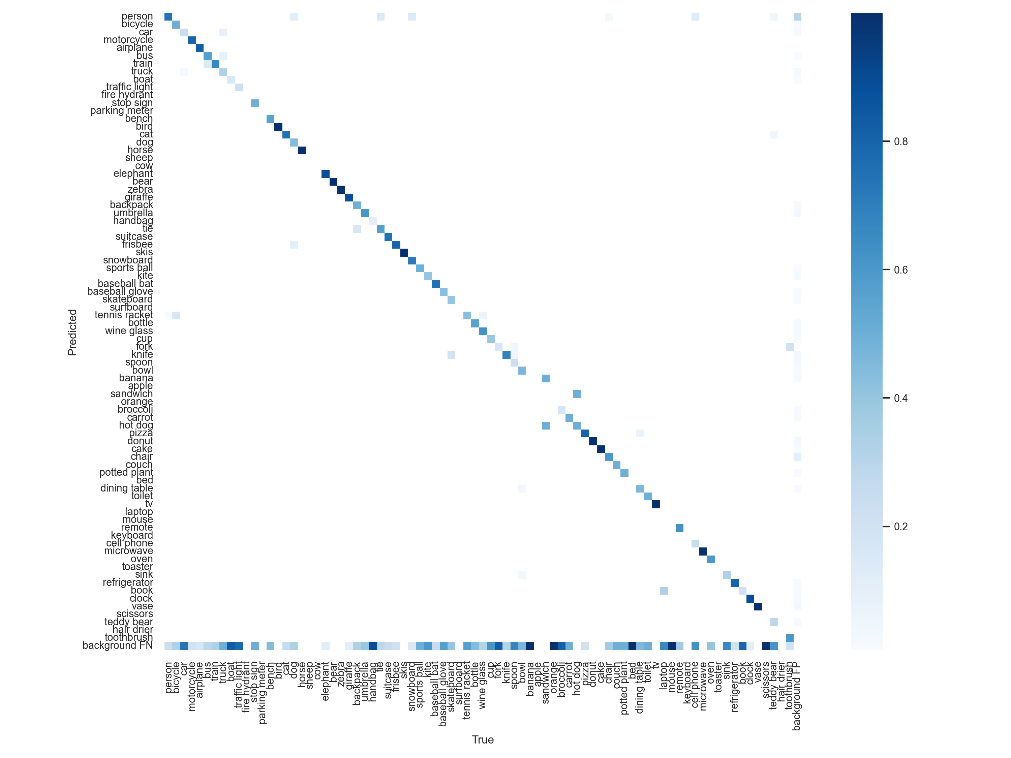


Figure 3. Confusion matrix

|  |  |  |
| --- | --- | --- |
| Precision | recall | mAP\_0.5 |
| 0.68602 | 0.57195 | 0.64445 |
| 0.69043 | 0.56874 | 0.64931 |

Table 2. Metrics

1. **Training Y****OLO v5 + Deepsort with own dataset**

Result:

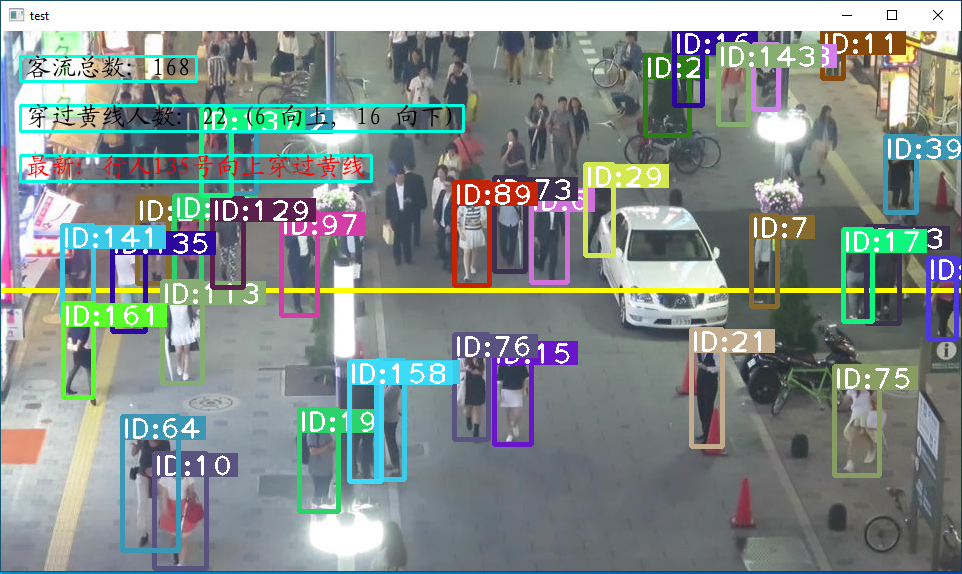


Figure 4. Calculate the pedestrian flow

1. **Next plan**

After finished the environment building and run some experiments with YOLO v5 and Deepsort, I think some of the problems need to be discussed and working plans are as follows:

* After completing the deployment and preliminary test of YOLO v5 and Deepsort, the next work is to encapsulate the two models into classes, and transform the video data in FLIR dataset into MP4 file, and try to use the MP4 data to train the encapsulated classes to complete the target detection and tracking of video data;
* For the model parameters and weights, there is still a lot of room for improvement. Therefore, next, we can design improved strategies and methods to improve the prediction MAP/MAPE;
* At present, the biggest obstacle of running the code is that it takes a long time to complete an epoch, which obviously cannot meet the needs of the industry. Therefore, we can also take improving the running efficiency of the code as another starting point;
* After discussed with Jessie about this project, I'm still quite confused about the division of our responsibilities between us. So, I'll be very grateful to spend some time to help me understand this project better, about what we want to do, and after finish the experiment what I need to do next.