HW2: Street View House Numbers detection

XIU-RONG HUANG
Institute of Information Management
National Yang Ming Chiao Tung University
Jejin0327@gmail.com

Github : https://github.com/jiejin0327/object_detection

Inference.ipynb : https://drive.google.com/file/d/1IgFZgWWZmEEKyBn2KOXbzF6OoZiCUfwp/view?usp=sharing

1. Introduction

In this assignment, I implement the techniques of object detection trained in deep learning network. To detect the street view house number, I chose the YOLOv5 models for real-time object detection. Thanks to the released code from author[1], I download the pre-trained weights in order to use the model for the given data in this homework .For the high accuracy score, I utilize proper hyperparameter setting, data preprocess (create annotations) and proper weights .

2. Implementation Procedure

2.1 Data pre-process

In order to prepare data for training, I transformed the .mat format into yolo format in a .txt file. Each .txt file contains the annotations for the object class and bounding boxes. The bounding boxes can be determined by the x and y axis coordinates in the upper-left corner and the lower-right corner of the rectangle. Bounding boxes are represented by two coordinates (x, y) and width (w) and height (h). The transform formula for bounding boxes is:

$$\mathrm{x} = \left(\mathrm{xmin} + \frac{\mathrm{xmax} - \mathrm{xmin}}{2}\right) * \frac{1}{\mathrm{image}_{\mathrm{w}}}$$

$$\mathrm{y} = \left(\mathrm{ymin} + \frac{\mathrm{ymax} - \mathrm{ymin}}{2}\right) * \frac{1}{\mathrm{image}_{\mathrm{h}}}$$

$$\mathrm{w} = (\mathrm{xmax} - \mathrm{xmin}) * \frac{1}{\mathrm{image_w}}$$

$$\mathrm{h} = (\mathrm{ymax} - \mathrm{ymin}) * \frac{1}{\mathrm{image_h}}$$

2.2 Model architecture

For training a object detection model, I chose the YoloV5 model. This is one of those models which is considered one of the fastest and accurate. Since the official provided whole codes on GitHub, I utilized the pre-trained model and fine-tune the weights. Fitting the output class number of fully connected layers, I set the argument to 10 given the class numbers of number classification.

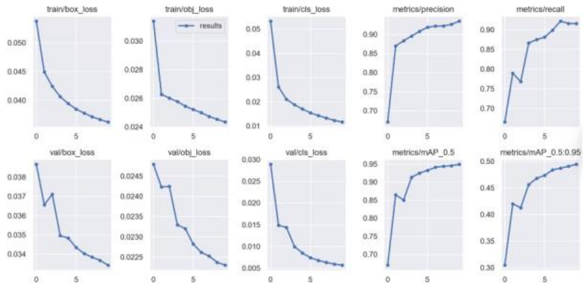
2.3 hyperparameter setting

Getting the better performance of model, I set the hyperparameter following:

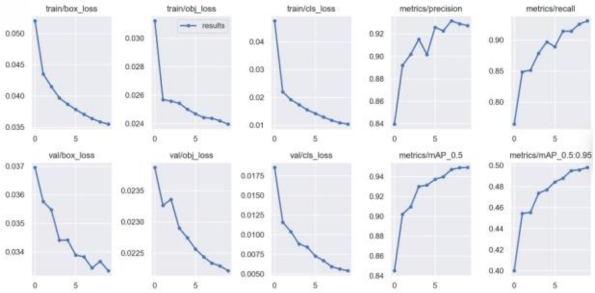
- batch size : 16
- epoch numbers:10
- image size : 640x640
- confidence threshold : 0.2
- NMS IoU threshold : 0.45
- Weight : yolov5m.pt

2.4 Experiment result

YoloV5 provides four different pre-trained weights contain YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. Due to the limitation of memory in my computer, I just trained YOLOv5s and YOLOv5m weights. The following shows the training results between two different weight :

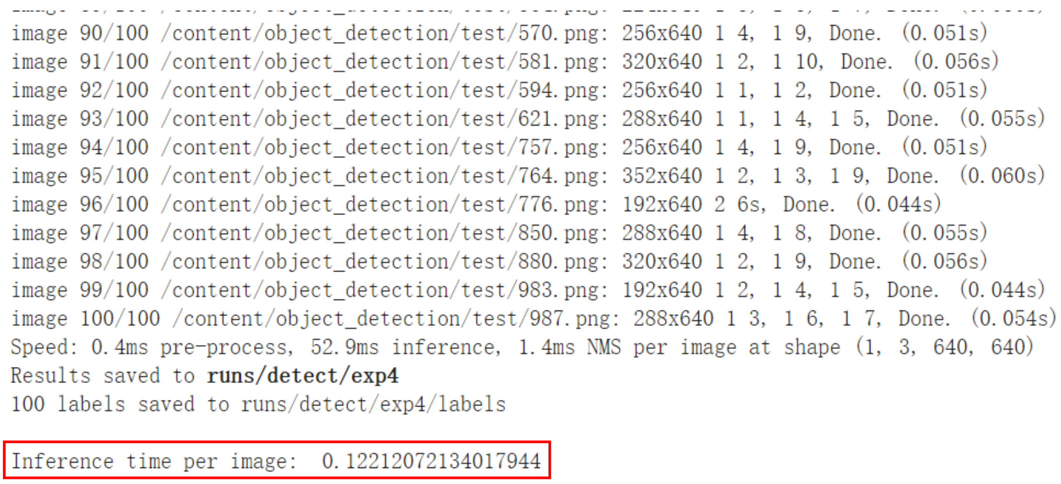


(a) YOLOv5s



(b) YOLOv5m

2.5 Inference results



```
image 90/100 /content/object_detection/test/570. png: 256x640 1 4,  1 9,  Done.  (0. 051s)
image 91/100 /content/object_detection/test/581. png: 320x640 1 2,  1 10,  Done.  (0. 056s)
image 92/100 /content/object_detection/test/594. png: 256x640 1 1,  1 2,  Done.  (0. 051s)
image 93/100 /content/object_detection/test/621. png: 288x640 1 1,  1 4,  1 5,  Done.  (0. 055s)
image 94/100 /content/object_detection/test/757. png: 256x640 1 4,  1 9,  Done.  (0. 051s)
image 95/100 /content/object_detection/test/764. png: 352x640 1 2,  1 3,  1 9,  Done.  (0. 060s)
image 96/100 /content/object_detection/test/776. png: 192x640 2 6s,  Done.  (0. 044s)
image 97/100 /content/object_detection/test/850. png: 288x640 1 4,  1 8,  Done.  (0. 055s)
image 98/100 /content/object_detection/test/880. png: 320x640 1 2,  1 9,  Done.  (0. 056s)
image 99/100 /content/object_detection/test/983. png: 192x640 1 2,  1 4,  1 5,  Done.  (0. 044s)
image 100/100 /content/object_detection/test/987. png: 288x640 1 3,  1 6,  1 7,  Done.  (0. 054s)
Speed: 0. 4ms pre-process,  52. 9ms inference,  1. 4ms NMS per image at shape (1,  3,  640,  640)
Results saved to runs/detect/exp4
100 labels saved to runs/detect/exp4/labels

Inference time per image:  0. 12212072134017944
```

3. Summary

Through the analysis of experiment results, I realize that the pre-trained model provides higher performance even feeding to another data. In this assignment, I implemented the pre-trained model YoloV5. Due to YoloV5 as one of the fastest models, my inference time per image is 0.122 which is lower than baseline. The weights choice from YOLOv5s to YOLOv5m lead to my scores increasing from 39 to 40. After a few days on tuning model, I get the 0.406789 scores on codalab in the end.

References

[1] https://github.com/ultralytics/yolov5