# HW2 Object Detection

Street View House Number Detection

# Outline

- Introduction
- Methodology
- Experiment
- Related Work
- Code and Reference

# Introduction

- The purpose of this project is to detect house number in street view images.

- The dataset used in this project is street view house number (SVHN)

  - Training data: 33402 images

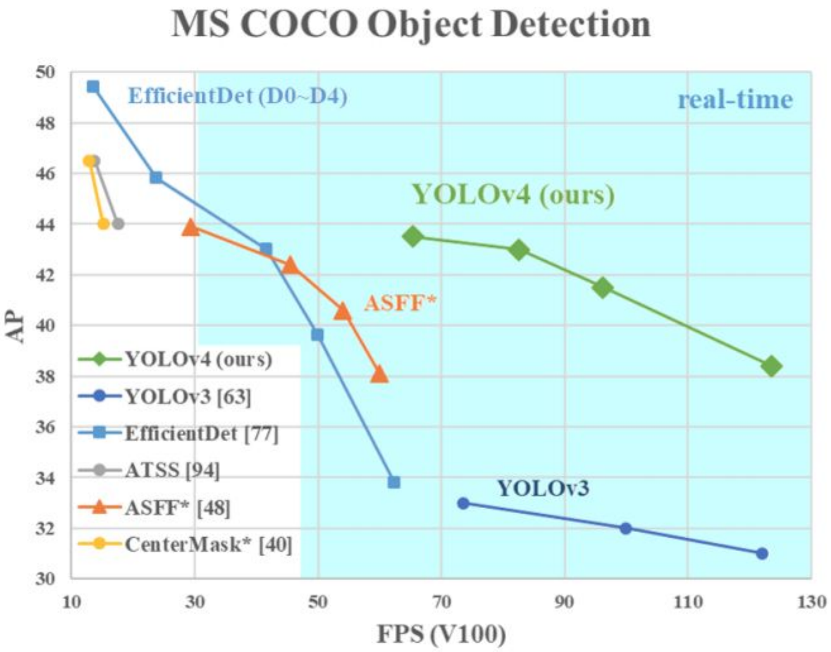
  - Testing data: 13068 images

- Not only accuracy is essential, the reference speed is important as well.

# Methodology

- There are many modern methods perform pretty well on object detection tasks, such as R-CNN family, EfficientDet, and YOLO family.

- The most widely used method nowadays is YOLOv4, which is the state-of-the-art method on real-time object detection.



# Methodology (cont.)

- Hyperparameters
  - batch: 64
  - subdivision: 64
  - momentum: 0.949
  - initial learning rate: 0.001
  - iterations: 33k

- Network Architecture
  - YOLOv4
  - NN input size: 608*608
  - NN output size: 45  (i.e., (classes+5)*3)

# Experiment

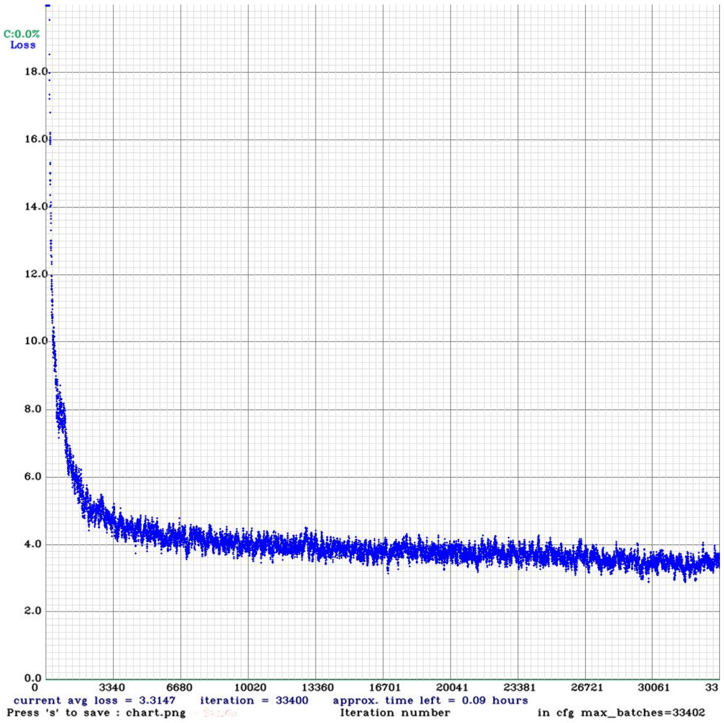
- Hardware information
  - CPU: i9-10900X
  - GPU: RTX 2080ti * 2
  - RAM: 62G

- Training time for 33k iterations: 24hr

# Experiment (cont.)

- I've tried different NN input size, such as 480*480, 608*608. The accuracy of the latter is much higher than the former, which is to be expected.

- Due to the scoring rule, I set the testing threshold to 0.005 to get higher mAP.
    - 608*608 mAP: 0.50841
    - 480*480 mAP: 0.43859



# Experiment (cont.)

- The model reference speed is tested on Google Colab (Tesla T4) by the commands in the picture below.

```python
# run darknet detection on test images
import time
start_time = time.time()
!./darknet detector test  obj.data cfg/yolo-obj.cfg cfg/yolo-obj_last.weights -ext_output -dont_show -out result.json < test.txt
cost_time = time.time() - start_time
time_per_img = cost_time/13068
print("cost time: ", cost_time)
print("average time for one image: ", time_per_img)
```

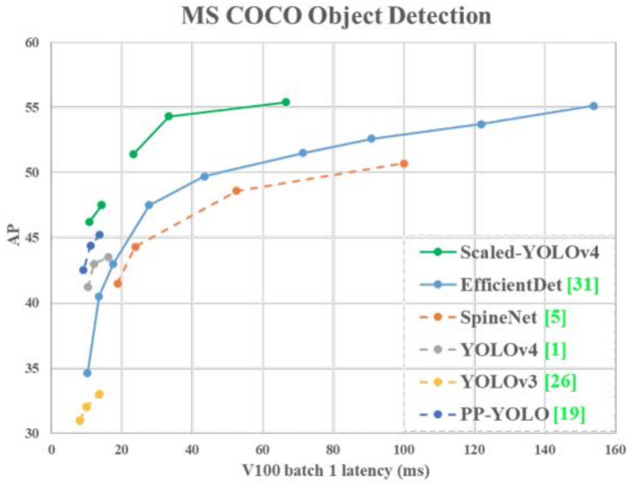
- Result: 0.03 SPF (second per frame)
  = 33.3 FPS (frame per second)

```
7: 48%  (left_x:   46   top_y:    8   width:   11
Enter Image Path: cost time:  512.1359279155731
average time for one image:  0.039190077128525645
```

- Notes: The total time cost (512.1s) includes loading YOLOv4 to GPU. So the inference time should be a little bit faster than 0.03 SPF.

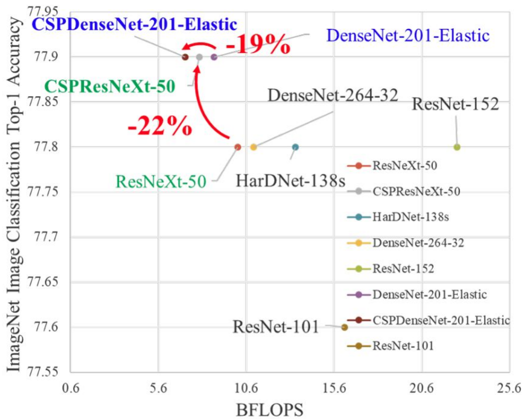
# Related Work

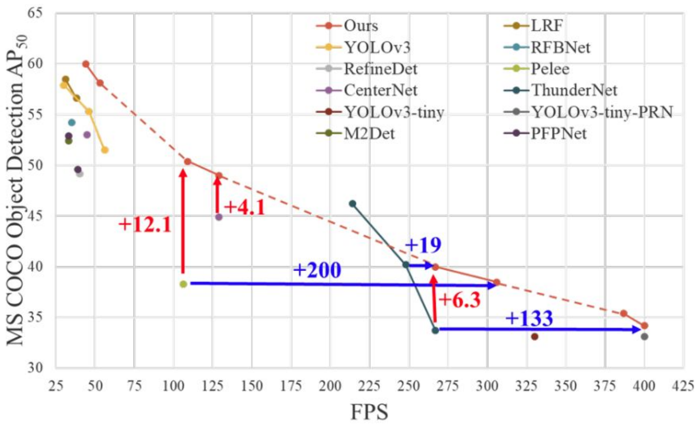
- [Scaled-YOLOv4]
  This paper proposed a scaling approach that modifies not only the depth, width, resolution, but also structure of the network. YOLOv4-large achieves state-of-the-art results: 55.4% AP for the MS COCO. YOLOv4-tiny achieves 22.0% AP  at a speed of ~443 FPS on RTX 2080Ti. With TensorRT, YOLOv4-tiny achieves 1174 FPS.



# Related Work (cont.)

- CSPNet
  Cross Stage Partial Network (CSPNet) reduces computations by 20% with equivalent or even superior accuracy on the ImageNet dataset, and significantly outperforms state-of-the-art approaches in terms of AP_50 on the MS COCO object detection dataset. CSPNet is easy to cope with architectures based on ResNet, ResNeXt and DenseNet.



# Code and Reference

- Code Github Link
- Darknet
- Data parsing
- YOLOv4