# VRDL HW2: Object Detection:

**Name:陳政彥**

**Student ID:310552006**

# Link of my code:

**Github repo link:**
**https://github.com/egghead2630/VRDL_HW2**

**Eval Code(inference.ipynb):**
https://colab.research.google.com/drive/1_ziiQF5g-lu8BFIUoYCgF-i9x2s8C_EB?usp=sharing

**Model:**
**https://drive.google.com/file/d/1EfmCP7WWCvxOp90zdR_7vdNfmdxq3P5A/view?usp=sharing**

**Config:**
**https://drive.google.com/file/d/1cuOqzwqRlbO7KXoyscsp33-Mt5Jk4DiH/view?usp=sharing**

**README:**
https://github.com/egghead2630/VRDL_HW2/blob/main/README.md

# Reference:

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

# Benchmark:

# Code:

```
# Test your inference time
TEST_IMAGE_NUMBER = 100 # This number is fixed.
test_img_list = []

# Read image (Be careful with the image order)
data_listdir.sort(key = lambda x: int(x[:-4]))
for img_name in data_listdir[:TEST_IMAGE_NUMBER]:
    img_src = os.path.join("/content/test/test", img_name)
    img_dst = os.path.join("/content/data/test", img_name)
    test_img_list.append(img_name)
    shutil.copy(img_src, img_dst)


start_time = time.time()

%cd /content/yolov5

!python /content/yolov5/val.py --img 320 --batch 10 --data digits.yaml --task test --device 0 --weights best.pt

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

# Remember to screenshot!
```

# Result:

```
/content/yolov5
val: data=digits.yaml, weights=['best.pt'], batch_size=10, imgsz=320, conf_thres=0.001, iou_thres=0.6, task=test, device=0, single_cls=False, augment=False, verbose=False, save_txt=False, sav
YOLOv5 🚀 v6.0-103-g7a39803 torch 1.10.0+cu111 CUDA:0 (Tesla K80, 11441MiB)

Fusing layers...
Model Summary: 213 layers, 7037095 parameters, 0 gradients, 15.9 GFLOPs
test: Scanning '../data/test.cache' images and labels... 0 found, 100 missing, 0 empty, 0 corrupted: 100% 100/100 [00:00<?, ?it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 10/10 [00:01<00:00,  9.34it/s]
                 all        100          0          0          0          0          0
Speed: 0.1ms pre-process, 5.0ms inference, 2.1ms NMS per image at shape (10, 3, 320, 320)
Results saved to runs/val/exp3

Inference time per image: 0.08265321254730225
```

# Brief introduction

My model is trained for the HW2 digit detection problem, I use mainly yolov5 to accomplish the HW2.

**Basic structure:**

I split the training data into train-part and validate-part with proportion: **9:1**. The model which performs best on validate-part would be chosen.

**Technics applied**:

**1. Some preprocess and augmentation on the train-part**

**2. Proper hyperparameters setting(Tuning)**

**3. Model selection**

In the HW2, I apply above methods to train my model better, I will explain them in detail in the following part: **Methodology**

# Methodology:

This part, I simply introduce how I apply above methods in the HW2.

**About Data Preprocess and Augmentation:**

I simply use the default augmentation and preprocess setting of yolov5, since the default settings are good enough for me to gain a satisfying predictive model

**About Hyperparameters tuning:**

**Image size:**

I first apply the (640*640) image size as default, however, I find it not suitable to do so because most of the pictures in data set are really small, even smaller than (200 * 200). Therefore, I latter choose to apply (320 * 320) image size on training process, and get a good model with about 4-times faster training

**Batch size**:

I set batch size to 16 as yolov5 github code suggested, and according to my observation, it's good to train with this batch size in this HW.

**About Model Selection:**

It's well-known that we shall take a trade-off when choosing among models, in the yolov5 structure, there are 4 type: small(s), medium(m), large(l), extra large(xl). In the aspect of complexity: xl > l > m > s, and in the aspect of speed: s > m > l > xl. In this homework, I choose to use small: **yolov5s**, for the reason that yolov5s is already complex enough to do prediction on digits according to the experimental result. Moreover, in my opinion, simply applying a more complicated model backbone is usually not useful on improving the performance, sometimes even lead to a worse result, I will specify the issue in the following part: **Comparison with another model.**

# Comparison with another model:

In this part, I will compare **yolov5s** along with another model: **detectors_cascade_rcnn_r50_1x_coco.py** in mmdetection.

In the following we use **yolo** and **detector** as abbreviation of the above two models

**Inference and training time:**

While **training**, detector need about **2 hours** per epoch, and yolo need only a few minutes. While **inferencing**, detector need 70 minutes to accomplish 13000 pictures, while yolo take less than 2 minutes to accomplish them.

**Accuracy:**

Below is the two results generated by detector and yolo respectively.

**Detector:** trained for **10** hours, 5 epochs:

| | | | | |
|---|---|---|---|---|
| 7 | 0.342286 | answer.zip | 11/15/2021 15:44:06 | Finished |

**Yolo**: trained for about **7** hours, 80 epochs:

| | | | | |
|---|---|---|---|---|
| 13 | 0.416198 | answer.zip | 11/22/2021 10:56:59 | Finished |

Apparently, yolo outperforms much than detector, in terms of not only **training and inference time, but also accuracy**. However, detector actually has a much more complex structure than yolo. This experimental result reminds me of the principals when choosing the model, that **a more-complicated model is not always a better choice.**

# Summary:

I will summarize **what I have learned in this homework.**

**What I have learned in the homework:**

Basically I learn some new criterions and file formats, and also some important aspects including:

**1.What is mAP, bbox:**

Before writing the homework, I have no idea how to evaluate an object detection model, and the mAP criterion let me know where to begin with.

**2.What is COCO-format:**

Before writing the homework, I don't know what is the popular file format in the object-detection field, COCO-format is a good start, and I believe other kind such as VOC won't be a problem after realizing COCO.

**3.How to pick proper hyperparameters:**

It is important to pick proper hyperparameters, in this homework, I learned that picking good hyperparameters can not only help us improving the accuracy of our model, it can also saves time for us when training and evaluating our model.

**4.How to choose a proper model:**

In this homework, I learned that a complicated model is not always a great model, we should always keep this in mind to choose model wisely.

Above is my report for HW2, thanks for the patience reading it all down here!

Have a Nice day!

Sincerely,
Chen.