CVPDL HW1 ntnu_40747043S 劉紹楷

1. DETR and YOLOv6

The architecture of DETR:

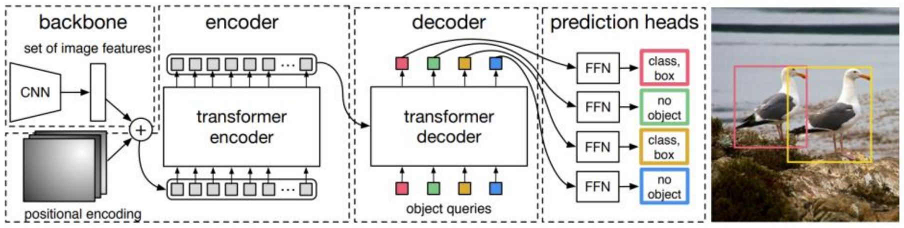


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

The architecture of YOLOv6:

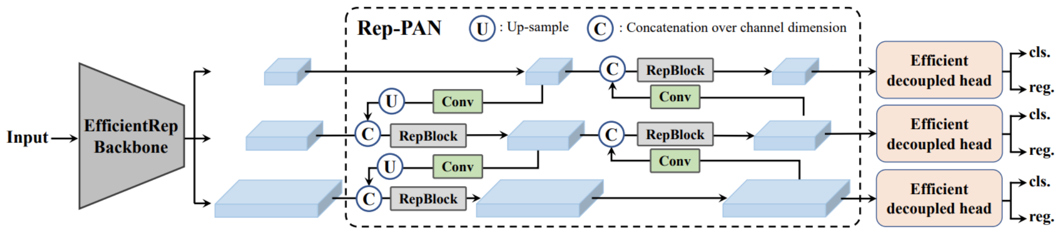
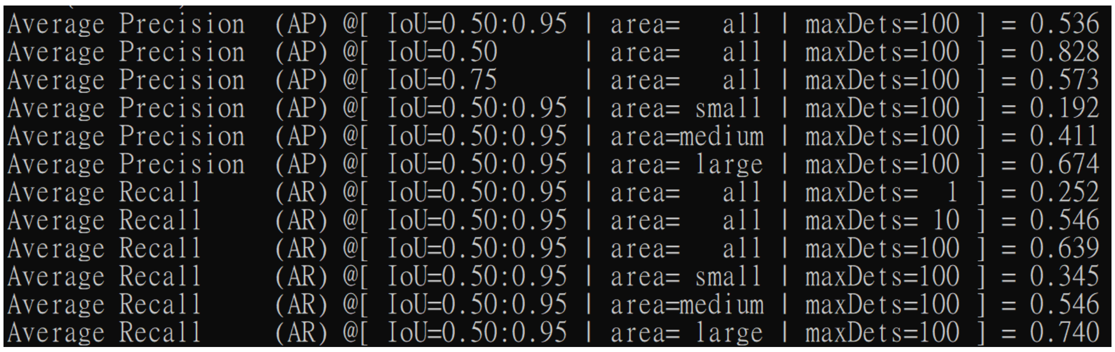


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

2. Compare the performance

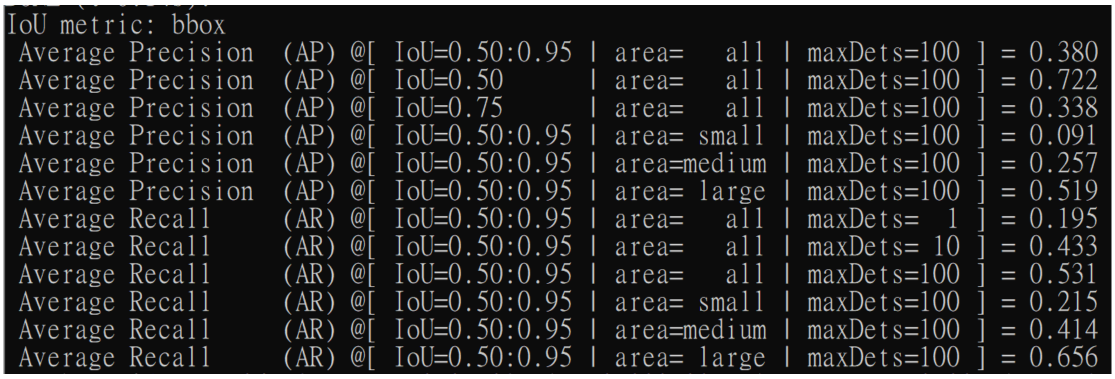
	mAP@50	mAP@75	mAP@[50:5:95]
YOLOv6	0.828	0.573	0.536
DETR	0.722	0.338	0.38

The details of YOLOv6



```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.536
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.828
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.573
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.192
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.411
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.674
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.252
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.546
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.639
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.345
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.546
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.740
```

The details of DETR



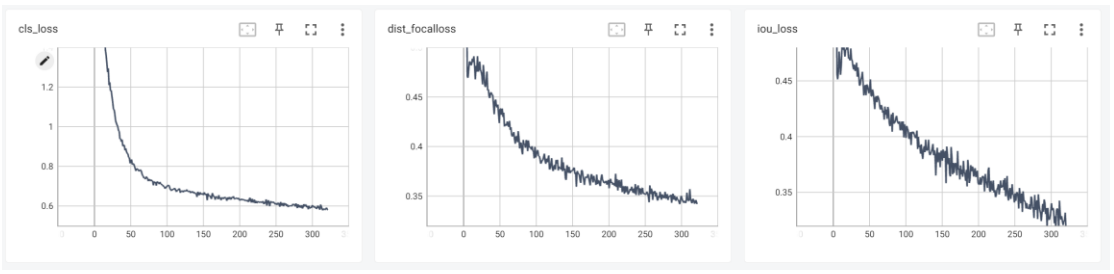
```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.380
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.722
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.338
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.091
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.257
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.519
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.195
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.433
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.531
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.215
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.414
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.656
```

3. The implementation

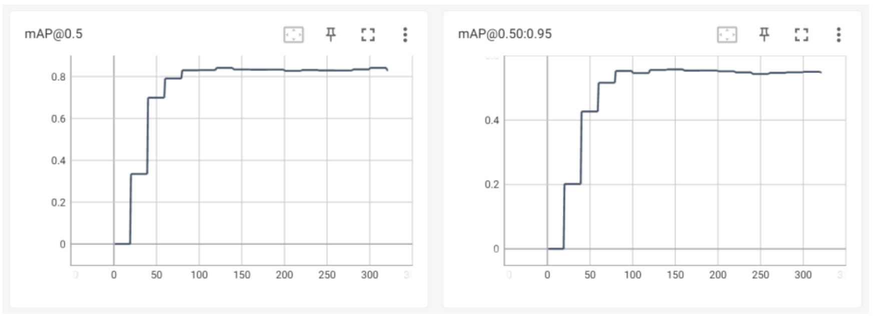
YOLOv6：

YOLOv6 有寫入 tensorboard，因此將訓練時有記錄下的數據畫圖呈現。

Train details



Valid details



Optim：SGD

lr_scheduler：cosine

loss：iou_loss(Ciou_loss or Siou_loss) + L1 loss

Augementation：Mosaic

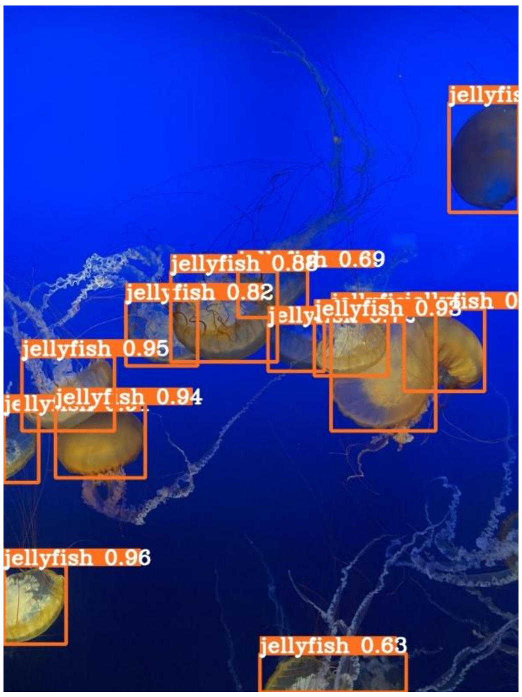
DETR

Loss：Cross entropy + L1 loss + generalized IoU loss

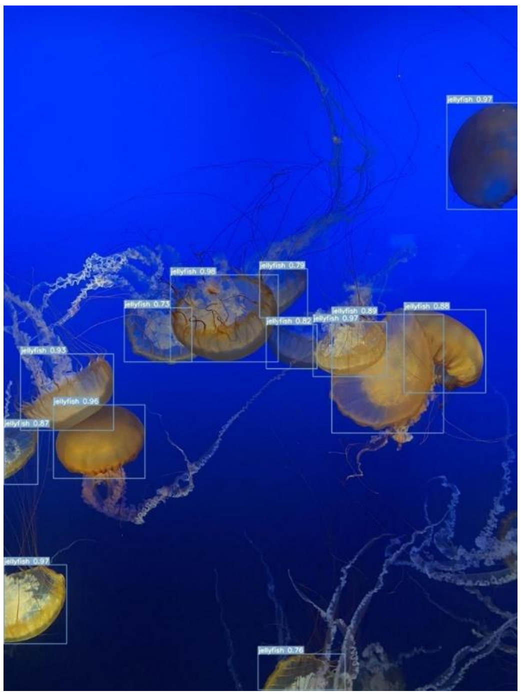
Augementation： Ramdom select + resize + Flip

4. Visualization

YOLOv6：



DETR：



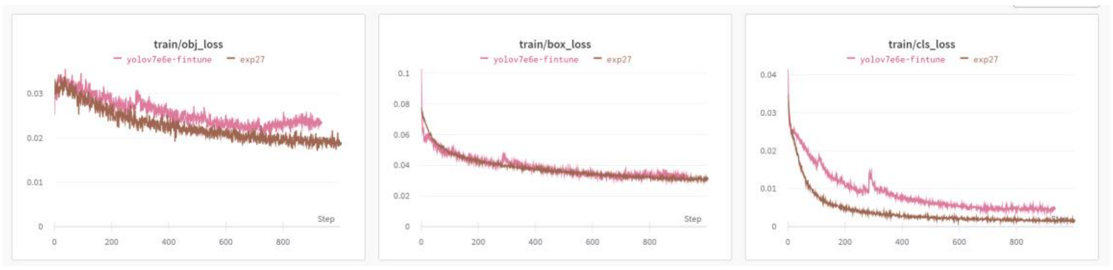
Reference

1. Draw box
   https://github.com/waittim/draw-YOLO-box
2. CoCo to yolo
   https://github.com/Weifeng-Chen/dl_scripts/blob/main/detection/coco2yolo.py
3. Numpy float errors in the version of 1.24
   https://stackoverflow.com/questions/74844262/how-can-i-solve-error-module-numpy-has-no-attribute-float-in-python
4. DETR paper
   https://arxiv.org/abs/2005.12872
5. YOLOv6 paper
   https://arxiv.org/abs/2209.02976

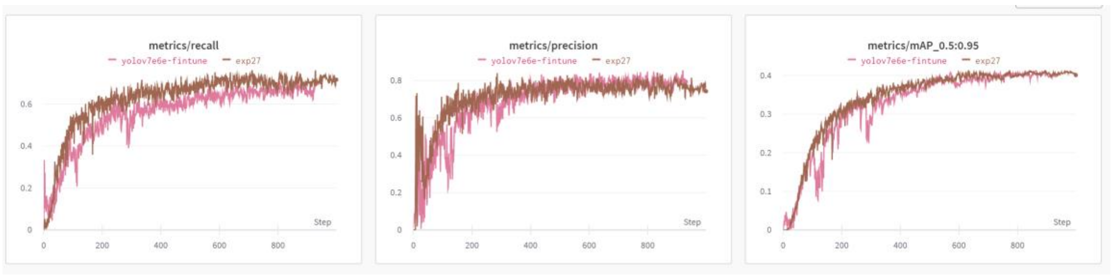
YOLOv7

我有嘗試使用 yolov7 過，official github 有使用 wandb，因此一併附上

Train details



Valid details



Performance 都比 yolov6 差很多，