PA 3: Object Detection

20222421 GSAI SimJaeYoon

0. Overview

This project is about object detection which is one of famous computer vision task. The goal of this project is to train and test famous object detectors such as Faster RCNN, Mask RCNN, YOLOv3 and YOLOv5 on Pascal VOC 2007 dataset.

1. [Problem #1] Training and Testing Faster RCNN with ResNet-50 on Pascal VOC 2007 Dataset

1.1. Fill out the following blanks in terms of mean average precision (mAP) and inference times (FPS) where mAP@# means that a prediction is positive if IoU ≥ # and discuss your experimental results

	mAP@0.5	mAP@0.6	mAP@0.7	mAP@0.8	mAP@0.9	FPS
Faster RCNN	0.738	0.688	0.561	0.352	0.072	31.541

```
load checkpoint from local path: work_dirs/Config_Faster_RCNN/epoch_12.pth
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 32.2 task/s, elapsed: 154s, ETA:     0s
---------------iou_thr: 0.5---------------

+-------------+------+------+--------+-------+
| class       | gts  | dets | recall | ap    |
+-------------+------+------+--------+-------+
| aeroplane   | 285  | 451  | 0.863  | 0.793 |
| bicycle     | 337  | 498  | 0.852  | 0.796 |
| bird        | 459  | 680  | 0.813  | 0.780 |
| boat        | 263  | 521  | 0.719  | 0.618 |
| bottle      | 469  | 662  | 0.672  | 0.592 |
| bus         | 213  | 427  | 0.892  | 0.774 |
| car         | 1201 | 1711 | 0.867  | 0.805 |
| cat         | 358  | 594  | 0.913  | 0.858 |
| chair       | 756  | 1446 | 0.680  | 0.541 |
| cow         | 244  | 537  | 0.898  | 0.784 |
| diningtable | 206  | 566  | 0.811  | 0.689 |
| dog         | 489  | 914  | 0.920  | 0.856 |
| horse       | 348  | 659  | 0.925  | 0.859 |
| motorbike   | 325  | 562  | 0.865  | 0.786 |
| person      | 4528 | 6554 | 0.876  | 0.794 |
| pottedplant | 480  | 702  | 0.598  | 0.441 |
| sheep       | 242  | 438  | 0.814  | 0.730 |
| sofa        | 239  | 682  | 0.895  | 0.711 |
| train       | 282  | 547  | 0.876  | 0.780 |
| tvmonitor   | 308  | 443  | 0.828  | 0.769 |
+-------------+------+------+--------+-------+
| mAP         |      |      |        | 0.738 |
+-------------+------+------+--------+-------+
OrderedDict([('mAP', 0.7377896904945374), ('AP50', 0.738)])
```

Faster RCNN mAP@0.5

```
load checkpoint from local path: work_dirs/Config_Faster_RCNN/epoch_12.pth
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 31.3 task/s, elapsed: 158s, ETA:     0s
---------------iou_thr: 0.6---------------

+-------------+------+------+--------+-------+
| class       | gts  | dets | recall | ap    |
+-------------+------+------+--------+-------+
| aeroplane   | 285  | 451  | 0.811  | 0.766 |
| bicycle     | 337  | 498  | 0.807  | 0.759 |
| bird        | 459  | 680  | 0.758  | 0.677 |
| boat        | 263  | 521  | 0.643  | 0.517 |
| bottle      | 469  | 662  | 0.644  | 0.580 |
| bus         | 213  | 427  | 0.878  | 0.761 |
| car         | 1201 | 1711 | 0.831  | 0.787 |
| cat         | 358  | 594  | 0.866  | 0.771 |
| chair       | 756  | 1446 | 0.616  | 0.504 |
| cow         | 244  | 537  | 0.861  | 0.749 |
| diningtable | 206  | 566  | 0.733  | 0.600 |
| dog         | 489  | 914  | 0.865  | 0.752 |
| horse       | 348  | 659  | 0.899  | 0.794 |
| motorbike   | 325  | 562  | 0.822  | 0.753 |
| person      | 4528 | 6554 | 0.814  | 0.752 |
| pottedplant | 480  | 702  | 0.529  | 0.400 |
| sheep       | 242  | 438  | 0.793  | 0.675 |
| sofa        | 239  | 682  | 0.858  | 0.693 |
| train       | 282  | 547  | 0.805  | 0.724 |
| tvmonitor   | 308  | 443  | 0.802  | 0.754 |
+-------------+------+------+--------+-------+
| mAP         |      |      |        | 0.688 |
+-------------+------+------+--------+-------+
OrderedDict([('mAP', 0.6882988214492798), ('AP60', 0.688)])
```

Faster RCNN mAP@0.6

```
load checkpoint from local path: work_dirs/Config_Faster_RCNN/epoch_12.pth
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 31.5 task/s, elapsed: 157s, ETA:     0s
---------------iou_thr: 0.7---------------

+-------------+------+------+--------+-------+
| class       | gts  | dets | recall | ap    |
+-------------+------+------+--------+-------+
| aeroplane   | 285  | 451  | 0.688  | 0.565 |
| bicycle     | 337  | 498  | 0.718  | 0.659 |
| bird        | 459  | 680  | 0.634  | 0.529 |
| boat        | 263  | 521  | 0.532  | 0.389 |
| bottle      | 469  | 662  | 0.539  | 0.466 |
| bus         | 213  | 427  | 0.798  | 0.672 |
| car         | 1201 | 1711 | 0.759  | 0.696 |
| cat         | 358  | 594  | 0.768  | 0.649 |
| chair       | 756  | 1446 | 0.519  | 0.412 |
| cow         | 244  | 537  | 0.754  | 0.628 |
| diningtable | 206  | 566  | 0.612  | 0.453 |
| dog         | 489  | 914  | 0.753  | 0.633 |
| horse       | 348  | 659  | 0.793  | 0.662 |
| motorbike   | 325  | 562  | 0.692  | 0.567 |
| person      | 4528 | 6554 | 0.684  | 0.571 |
| pottedplant | 480  | 702  | 0.375  | 0.249 |
| sheep       | 242  | 438  | 0.698  | 0.555 |
| sofa        | 239  | 682  | 0.770  | 0.607 |
| train       | 282  | 547  | 0.695  | 0.580 |
| tvmonitor   | 308  | 443  | 0.744  | 0.671 |
+-------------+------+------+--------+-------+
| mAP         |      |      |        | 0.561 |
+-------------+------+------+--------+-------+
OrderedDict([('mAP', 0.5606789588928223), ('AP70', 0.561)])
```

Faster RCNN mAP@0.7

```
load checkpoint from local path: work_dirs/Config_Faster_RCNN/epoch_12.pth
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 31.1 task/s, elapsed: 159s, ETA:     0s
---------------iou_thr: 0.8---------------

+-------------+------+------+--------+-------+
| class       | gts  | dets | recall | ap    |
+-------------+------+------+--------+-------+
| aeroplane   | 285  | 451  | 0.446  | 0.299 |
| bicycle     | 337  | 498  | 0.499  | 0.377 |
| bird        | 459  | 680  | 0.420  | 0.310 |
| boat        | 263  | 521  | 0.297  | 0.169 |
| bottle      | 469  | 662  | 0.365  | 0.283 |
| bus         | 213  | 427  | 0.723  | 0.625 |
| car         | 1201 | 1711 | 0.620  | 0.561 |
| cat         | 358  | 594  | 0.539  | 0.412 |
| chair       | 756  | 1446 | 0.319  | 0.230 |
| cow         | 244  | 537  | 0.590  | 0.365 |
| diningtable | 206  | 566  | 0.369  | 0.233 |
| dog         | 489  | 914  | 0.503  | 0.340 |
| horse       | 348  | 659  | 0.580  | 0.426 |
| motorbike   | 325  | 562  | 0.465  | 0.338 |
| person      | 4528 | 6554 | 0.461  | 0.353 |
| pottedplant | 480  | 702  | 0.192  | 0.122 |
| sheep       | 242  | 438  | 0.566  | 0.435 |
| sofa        | 239  | 682  | 0.531  | 0.371 |
| train       | 282  | 547  | 0.479  | 0.359 |
| tvmonitor   | 308  | 443  | 0.542  | 0.427 |
+-------------+------+------+--------+-------+
| mAP         |      |      |        | 0.352 |
+-------------+------+------+--------+-------+
OrderedDict([('mAP', 0.3517701029777527), ('AP80', 0.352)])
```

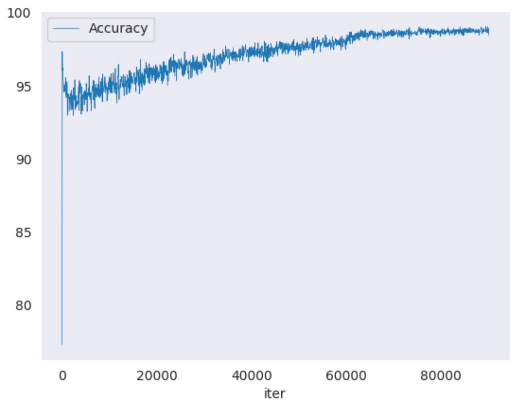
Faster RCNN mAP@0.8

```
load checkpoint from local path: work_dirs/Config_Faster_RCNN/epoch_12.pth
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 31.7 task/s, elapsed: 156s, ETA:     0s
---------------iou_thr: 0.9---------------

+-------------+------+------+--------+-------+
| class       | gts  | dets | recall | ap    |
+-------------+------+------+--------+-------+
| aeroplane   | 285  | 451  | 0.098  | 0.019 |
| bicycle     | 337  | 498  | 0.139  | 0.076 |
| bird        | 459  | 680  | 0.094  | 0.091 |
| boat        | 263  | 521  | 0.049  | 0.027 |
| bottle      | 469  | 662  | 0.090  | 0.036 |
| bus         | 213  | 427  | 0.296  | 0.173 |
| car         | 1201 | 1711 | 0.293  | 0.179 |
| cat         | 358  | 594  | 0.173  | 0.055 |
| chair       | 756  | 1446 | 0.069  | 0.027 |
| cow         | 244  | 537  | 0.193  | 0.066 |
| diningtable | 206  | 566  | 0.136  | 0.114 |
| dog         | 489  | 914  | 0.153  | 0.109 |
| horse       | 348  | 659  | 0.210  | 0.109 |
| motorbike   | 325  | 562  | 0.135  | 0.036 |
| person      | 4528 | 6554 | 0.146  | 0.056 |
| pottedplant | 480  | 702  | 0.038  | 0.008 |
| sheep       | 242  | 438  | 0.153  | 0.113 |
| sofa        | 239  | 682  | 0.134  | 0.038 |
| train       | 282  | 547  | 0.156  | 0.043 |
| tvmonitor   | 308  | 443  | 0.195  | 0.070 |
+-------------+------+------+--------+-------+
| mAP         |      |      |        | 0.072 |
+-------------+------+------+--------+-------+
OrderedDict([('mAP', 0.07233119010925293), ('AP90', 0.072)])
```

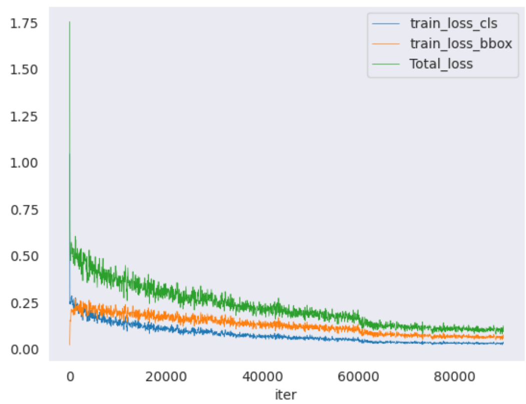
Faster RCNN mAP@0.9

1.2. Try the efforts to improve the performances on your network models, such as your learning techniques or your network improvements that are not provided by basic codes

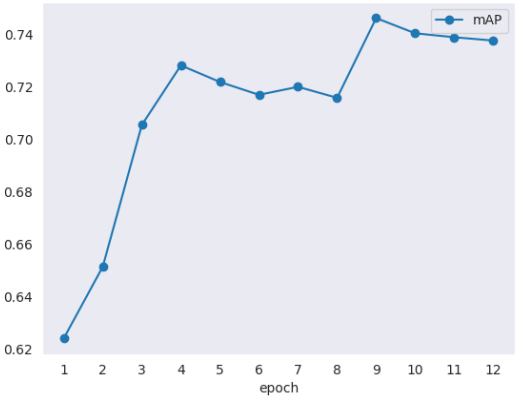
A. Show learning curves for training and validation



Faster RCNN Accuracy



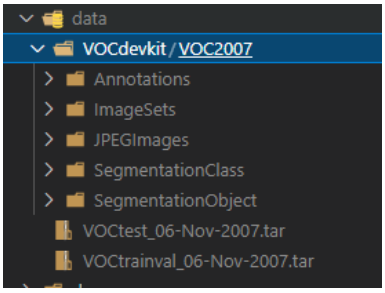
Faster RCNN Loss



Faster RCNN mAP

B. Show your source codes and trained model parameters

I downloaded mmdetection from github (GitHub - open-mmlab/mmdetection: OpenMMLab Detection Toolbox and Benchmark) according to the reference. And I downloaded Pascal VOC 2007 dataset like the previous assignment.



Pascal VOC 2007 dataset

To use mmdetection, I had to make a config file. So I made a file called Config_Faster_RCNN.py, and I could select and write various information such as data information, model information, and parameter information.

I made my own configuration file "Config_Faster_RCNN.py" based on

a. configurations/_base_/default_runtime.py,

b. configurations/_base_/datasets/voc0712.py,

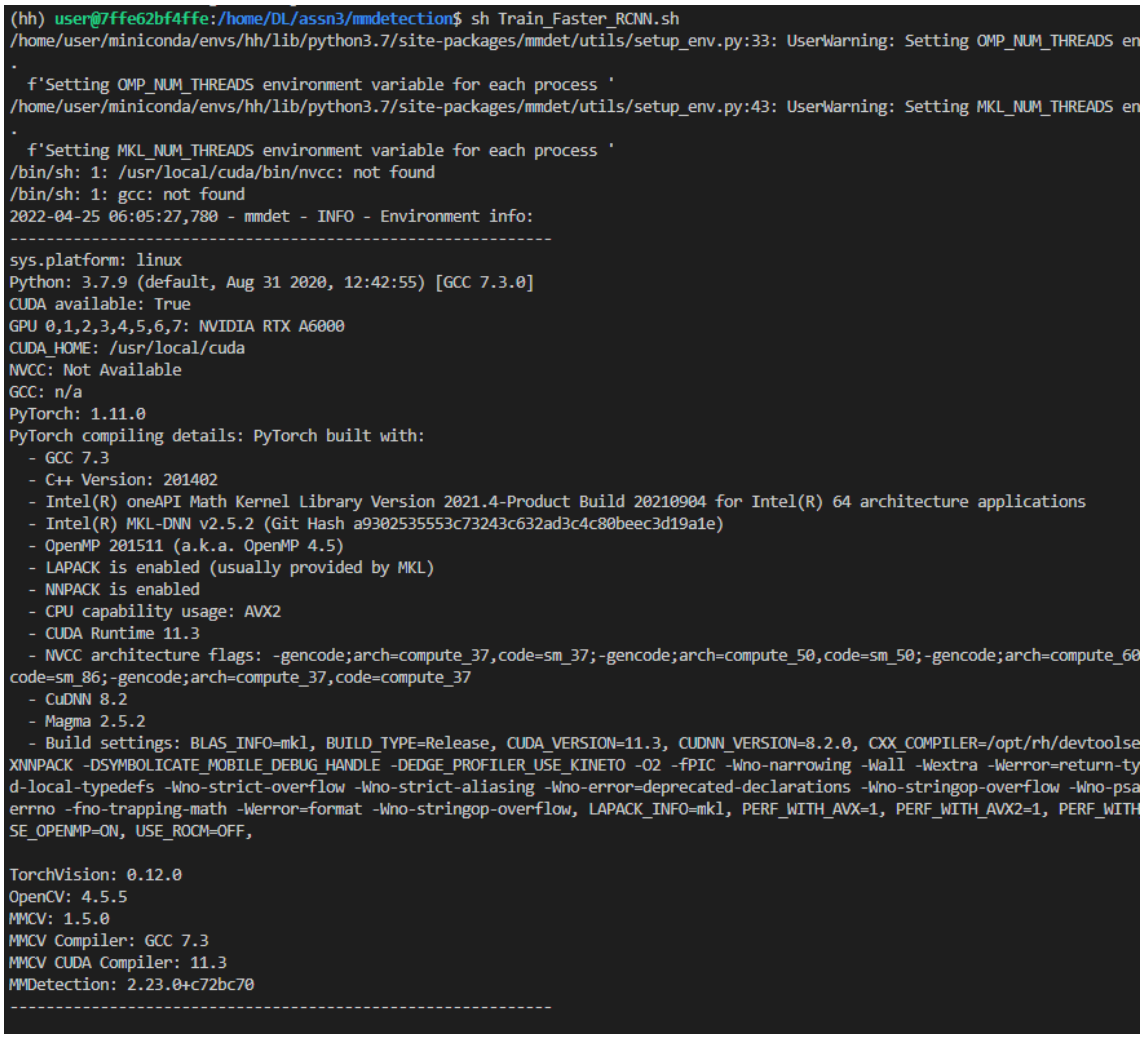
c. configurations/_base_/models/faster_rcnn_r50_fpn.py,

d. configurations/_base_/schedules/schedule_1py.

Based on this, training can be performed through the following command.

[train] python tools/train.py Config_Faster_RCNN.py

Then I can check my system environment in the terminal window as follows. I used the latest version with python 3.7.9, pytorch 1.11.0, CUDA 11.3, CuDNN 8.2, MMCV 1.5.0, and MMDetection 2.23.0. An important part of this was the compatibility issue between mmcv and mmdetection. In the beginning, it took a lot of time to install due to compatibility issues. Then, after reading the instruction again from the beginning and selecting and installing the mmcv-full version well, the compatibility problem was solved.



Training Environment

In more detail, in my config file, I can decide the path of data as follows or change the parameters to continue to change the learning conditions. While reading the paper, I thought about finding the best hyperparameter. I tried to change the scale of the data size and modify the normalization value and other values. In learning, the hyperparameter made a choice by referring to the thesis, and it took longer than I thought to learn, so I couldn't do many tests. Still, since the config file was configured well, it was easy to change and experiment.

```python
### 2022 Fall Deep Learning(AIGS538) ###

# Dataset settings
dataset_type = 'VOCDataset'
data_root = 'data/VOCdevkit/'
CLASSES = ('aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car',
           'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse',
           'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train',
           'tvmonitor')
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(1000, 600), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),
]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(1000, 600),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ])
]
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type='RepeatDataset',
        times=3,
        dataset=dict(
            type=dataset_type,
            ann_file=[
                data_root + 'VOC2007/ImageSets/Main/trainval.txt'
            ],
            img_prefix=[data_root + 'VOC2007/'],
            pipeline=train_pipeline)),
    val=dict(
        type=dataset_type,
        ann_file=data_root + 'VOC2007/ImageSets/Main/test.txt',
        img_prefix=data_root + 'VOC2007/',
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root + 'VOC2007/ImageSets/Main/test.txt',
        img_prefix=data_root + 'VOC2007/',
        pipeline=test_pipeline))
evaluation = dict(interval=1, metric='mAP')
```

Config_Train_RCNN.py data information

```python
load_from = "http://download.openmmlab.com/mmdetection/v2.0/faster_rcnn/faster_

# Set Schedule
# optimizer
optimizer = dict(type='SGD', lr=0.0025, momentum=0.9, weight_decay=0.0001)
optimizer_config = dict(grad_clip=None)
# learning policy
lr_config = dict(
    policy='step',
    warmup='linear',
    warmup_iters=500,
    warmup_ratio=0.001,
    step=[8, 11])
runner = dict(type='EpochBasedRunner', max_epochs=12)
```

Config_Train_RCNN.py parameter information

In order to load and use the pre-trained model, the pre-trained model was loaded and used in the config file. When learned, the results are stored as follows for each epoch.

```
work_dirs / Config_Faster_RCNN
    20220425_060434.log
    20220425_060434.log.json
    20220425_060527.log
    20220425_060527.log.json
    Config_Faster_RCNN.py
    epoch_1.pth
    epoch_2.pth
```

Training model information

After learning, I can see the learning results as follows.

```
2022-04-25 10:03:06,978 - mmdet - INFO - Saving checkpoint at 12 epochs
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 36.9 task/s, elapsed: 134s, ETA:     0s
---------------iou_thr: 0.5---------------
2022-04-25 10:05:26,025 - mmdet - INFO -
+-------------+------+------+--------+-------+
| class       | gts  | dets | recall | ap    |
+-------------+------+------+--------+-------+
| aeroplane   | 285  | 451  | 0.863  | 0.793 |
| bicycle     | 337  | 498  | 0.852  | 0.796 |
| bird        | 459  | 680  | 0.813  | 0.780 |
| boat        | 263  | 521  | 0.719  | 0.618 |
| bottle      | 469  | 662  | 0.672  | 0.592 |
| bus         | 213  | 427  | 0.892  | 0.774 |
| car         | 1201 | 1711 | 0.867  | 0.805 |
| cat         | 358  | 594  | 0.913  | 0.858 |
| chair       | 756  | 1446 | 0.680  | 0.541 |
| cow         | 244  | 537  | 0.898  | 0.784 |
| diningtable | 206  | 566  | 0.811  | 0.689 |
| dog         | 489  | 914  | 0.920  | 0.856 |
| horse       | 348  | 659  | 0.925  | 0.859 |
| motorbike   | 325  | 562  | 0.865  | 0.786 |
| person      | 4528 | 6554 | 0.876  | 0.794 |
| pottedplant | 480  | 702  | 0.598  | 0.441 |
| sheep       | 242  | 438  | 0.814  | 0.730 |
| sofa        | 239  | 682  | 0.895  | 0.711 |
| train       | 282  | 547  | 0.876  | 0.780 |
| tvmonitor   | 308  | 443  | 0.828  | 0.769 |
+-------------+------+------+--------+-------+
| mAP         |      |      |        | 0.738 |
+-------------+------+------+--------+-------+
2022-04-25 10:05:26,050 - mmdet - INFO - Exp name: Config_Faster_RCNN.py
2022-04-25 10:05:26,056 - mmdet - INFO - Epoch(val) [12][4952]  mAP: 0.7378, AP50: 0.7380
```

Train Result Epochs with 12

And in relation to the learning process, the following commands were given according to the graph I want to draw. From each command, graphs such as training loss, acurracy, and validation mAP were drawn and checked.

[Train Loss] python tools/analysis_tools/analyze_logs.py plot_curve work_dirs/Config_Faster_RCNN/FasterRCNN.log.json --out ./results/loss.png --keys loss_cls loss_bbox loss --legend train_loss_cls train_loss_bbox Total_loss

[Train Accuracy] python tools/analysis_tools/analyze_logs.py plot_curve work_dirs/Config_Faster_RCNN/FasterRCNN.log.json --out ./results/accuracy.png --keys acc --legend Accuracy

[Validation mAP] python tools/analysis_tools/analyze_logs.py plot_curve work_dirs/Config_Faster_RCNN/FasterRCNN.log.json --out ./results/map.png --keys mAP --legend mAP

Finally, the mAP was checked according to each IoU threshold value through a test based on the model that was trained. Here, "epoch_12.pth" is the model information that was previously used for learning.

[test] python tools/test.py Config_Faster_RCNN.py work_dirs/Config_Faster_RCNN/epoch_12.pth --eval mAP --eval-options iou_thr=0.5(choose 0.5/0.6/0.7/0.8/0.9)

2. [Problem #2] Training and Testing Mask RCNN with ResNet-50 on Pascal VOC 2007 dataset

2.1. Fill out the following blanks in terms of mean average precision (mAP) and inference times (FPS) where mAP@# means that a prediction is positive if IoU ≥ # and discuss your experimental results

	mAP@0.5	mAP@0.6	mAP@0.7	mAP@0.8	mAP@0.9	FPS
Mask RCNN		0.553	0.397	0.190	0.023	22.351

It was confirmed that mAP or FPS was relatively inferior to Fast RCNN in Mask RCNN.

```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 22.6 task/s, elapsed: 219s, ETA:     0s
Evaluating bbox...
Loading and preparing results...
DONE (t=0.23s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.55s).
Accumulating evaluation results...
DONE (t=0.83s).

 Average Precision  (AP) @[ IoU=0.50:0.50 | area=   all | maxDets=100 ] = 0.628
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.50 | area= small | maxDets=1000 ] = 0.122
 Average Precision  (AP) @[ IoU=0.50:0.50 | area=medium | maxDets=1000 ] = 0.505
 Average Precision  (AP) @[ IoU=0.50:0.50 | area= large | maxDets=1000 ] = 0.737
 Average Recall     (AR) @[ IoU=0.50:0.50 | area=   all | maxDets=100 ] = 0.772
 Average Recall     (AR) @[ IoU=0.50:0.50 | area=   all | maxDets=300 ] = 0.772
 Average Recall     (AR) @[ IoU=0.50:0.50 | area=   all | maxDets=1000 ] = 0.772
 Average Recall     (AR) @[ IoU=0.50:0.50 | area= small | maxDets=1000 ] = 0.197
 Average Recall     (AR) @[ IoU=0.50:0.50 | area=medium | maxDets=1000 ] = 0.667
 Average Recall     (AR) @[ IoU=0.50:0.50 | area= large | maxDets=1000 ] = 0.898

OrderedDict([('bbox_mAP', 0.628), ('bbox_mAP_50', -1.0), ('bbox_mAP_75', -1.0), ('bbox_mAP_s', 0.122),
```

Mask RCNN mAP@0.5

```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 22.4 task/s, elapsed: 222s, ETA:     0s
Evaluating bbox...
Loading and preparing results...
DONE (t=0.24s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.64s).
Accumulating evaluation results...
DONE (t=0.80s).

 Average Precision  (AP) @[ IoU=0.60:0.60 | area=   all | maxDets=100 ] = 0.553
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.60:0.60 | area= small | maxDets=1000 ] = 0.092
 Average Precision  (AP) @[ IoU=0.60:0.60 | area=medium | maxDets=1000 ] = 0.421
 Average Precision  (AP) @[ IoU=0.60:0.60 | area= large | maxDets=1000 ] = 0.660
 Average Recall     (AR) @[ IoU=0.60:0.60 | area=   all | maxDets=100 ] = 0.706
 Average Recall     (AR) @[ IoU=0.60:0.60 | area=   all | maxDets=300 ] = 0.706
 Average Recall     (AR) @[ IoU=0.60:0.60 | area=   all | maxDets=1000 ] = 0.706
 Average Recall     (AR) @[ IoU=0.60:0.60 | area= small | maxDets=1000 ] = 0.160
 Average Recall     (AR) @[ IoU=0.60:0.60 | area=medium | maxDets=1000 ] = 0.579
 Average Recall     (AR) @[ IoU=0.60:0.60 | area= large | maxDets=1000 ] = 0.833

OrderedDict([('bbox_mAP', 0.553), ('bbox_mAP_50', -1.0), ('bbox_mAP_75', -1.0), ('bbox_mAP_s', 0.092),
```

Mask RCNN mAP@0.6

```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 22.4 task/s, elapsed: 221s, ETA:     0s
Evaluating bbox...
Loading and preparing results...
DONE (t=0.22s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.58s).
Accumulating evaluation results...
DONE (t=0.80s).

 Average Precision  (AP) @[ IoU=0.70:0.70 | area=   all | maxDets=100 ] = 0.397
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.70:0.70 | area= small | maxDets=1000 ] = 0.061
 Average Precision  (AP) @[ IoU=0.70:0.70 | area=medium | maxDets=1000 ] = 0.293
 Average Precision  (AP) @[ IoU=0.70:0.70 | area= large | maxDets=1000 ] = 0.482
 Average Recall     (AR) @[ IoU=0.70:0.70 | area=   all | maxDets=100 ] = 0.557
 Average Recall     (AR) @[ IoU=0.70:0.70 | area=   all | maxDets=300 ] = 0.557
 Average Recall     (AR) @[ IoU=0.70:0.70 | area=   all | maxDets=1000 ] = 0.557
 Average Recall     (AR) @[ IoU=0.70:0.70 | area= small | maxDets=1000 ] = 0.110
 Average Recall     (AR) @[ IoU=0.70:0.70 | area=medium | maxDets=1000 ] = 0.446
 Average Recall     (AR) @[ IoU=0.70:0.70 | area= large | maxDets=1000 ] = 0.660

OrderedDict([('bbox_mAP', 0.397), ('bbox_mAP_50', -1.0), ('bbox_mAP_75', -1.0), ('bbox_mAP_s', 0.061),
```

Mask RCNN mAP@0.7

```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 22.3 task/s, elapsed: 222s, ETA:     0s
Evaluating bbox...
Loading and preparing results...
DONE (t=0.23s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.48s).
Accumulating evaluation results...
DONE (t=0.80s).

 Average Precision  (AP) @[ IoU=0.80:0.80 | area=   all | maxDets=100 ] = 0.190
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.80:0.80 | area= small | maxDets=1000 ] = 0.018
 Average Precision  (AP) @[ IoU=0.80:0.80 | area=medium | maxDets=1000 ] = 0.124
 Average Precision  (AP) @[ IoU=0.80:0.80 | area= large | maxDets=1000 ] = 0.248
 Average Recall     (AR) @[ IoU=0.80:0.80 | area=   all | maxDets=100 ] = 0.336
 Average Recall     (AR) @[ IoU=0.80:0.80 | area=   all | maxDets=300 ] = 0.336
 Average Recall     (AR) @[ IoU=0.80:0.80 | area=   all | maxDets=1000 ] = 0.336
 Average Recall     (AR) @[ IoU=0.80:0.80 | area= small | maxDets=1000 ] = 0.045
 Average Recall     (AR) @[ IoU=0.80:0.80 | area=medium | maxDets=1000 ] = 0.249
 Average Recall     (AR) @[ IoU=0.80:0.80 | area= large | maxDets=1000 ] = 0.409

OrderedDict([('bbox_mAP', 0.19), ('bbox_mAP_50', -1.0), ('bbox_mAP_75', -1.0), ('bbox_mAP_s', 0.018), (
```

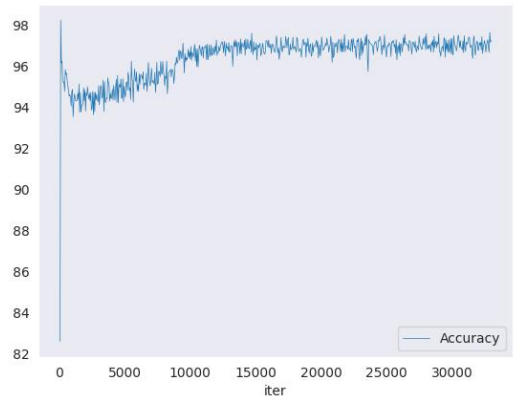
Mask RCNN mAP@0.8

```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 4952/4952, 22.4 task/s, elapsed: 221s, ETA:     0s
Evaluating bbox...
Loading and preparing results...
DONE (t=0.24s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=3.55s).
Accumulating evaluation results...
DONE (t=0.79s).

 Average Precision  (AP) @[ IoU=0.90:0.90 | area=   all | maxDets=100 ] = 0.023
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = -1.000
 Average Precision  (AP) @[ IoU=0.90:0.90 | area= small | maxDets=1000 ] = 0.000
 Average Precision  (AP) @[ IoU=0.90:0.90 | area=medium | maxDets=1000 ] = 0.011
 Average Precision  (AP) @[ IoU=0.90:0.90 | area= large | maxDets=1000 ] = 0.034
 Average Recall     (AR) @[ IoU=0.90:0.90 | area=   all | maxDets=100 ] = 0.080
 Average Recall     (AR) @[ IoU=0.90:0.90 | area=   all | maxDets=300 ] = 0.080
 Average Recall     (AR) @[ IoU=0.90:0.90 | area=   all | maxDets=1000 ] = 0.080
 Average Recall     (AR) @[ IoU=0.90:0.90 | area= small | maxDets=1000 ] = 0.002
 Average Recall     (AR) @[ IoU=0.90:0.90 | area=medium | maxDets=1000 ] = 0.045
 Average Recall     (AR) @[ IoU=0.90:0.90 | area= large | maxDets=1000 ] = 0.108

OrderedDict([('bbox_mAP', 0.023), ('bbox_mAP_50', -1.0), ('bbox_mAP_75', -1.0), ('bbox_mAP_s', 0.0), ('b
```

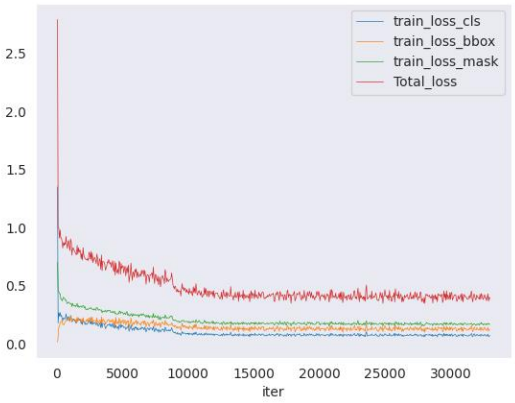
Mask RCNN mAP@0.9

2.2. Try the efforts to improve the performances on your network models, such as your learning techniques or your network improvements that are not provided by basic codes

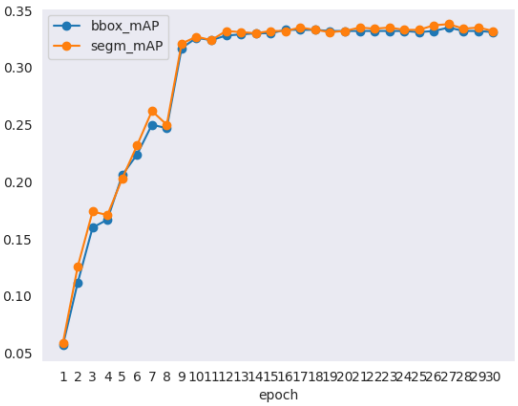
A. Show learning curves for training and validation



Mask RCNN Accuracy



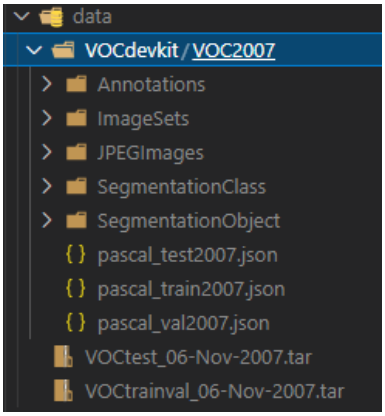
Mask RCNN Loss



Mask RCNN mAP

B. Show your source codes and trained model parameters

The Pascal VOC 2007 COCO dataset conversion file was used to use Mask RCNN. The .json files were used under the mmdetection/data/ folder as follows.



Data with .json files

To use mmdetection, I had to make a config file. So I made a file called Config_Mask_RCNN.py, and I could select and write various information such as data information, model information, and parameter information.

I made my own configuration file "Config_Mask_RCNN.py" based on

a. configurations/_base_/default_runtime.py,

b. configurations/_base_/datasets/coco_instance.py,

c. configurations/_base_/models/faster_rcnn_r50_fpn.py,

d. configurations/_base_/schedules/schedule_1py.

Based on this, training can be performed through the following command.

[train] python tools/train.py Config_Mask_RCNN.py

Like the first question, I tried to set the path of data in the config file and load the pretrained model. While reading the paper and lots of reference, I thought about finding the best hyperparameter.

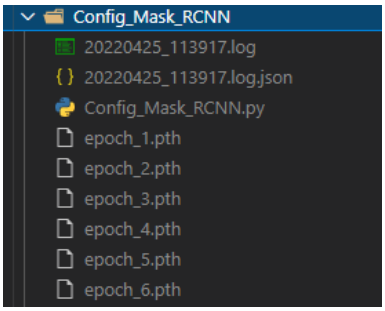
```python
### 2022 Fall Deep Learning(AIGS538) ###

# Dataset settings
dataset_type = 'CocoDataset'
data_root = 'data/VOCdevkit/'
CLASSES = ('aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car',
           'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse',
           'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train',
           'tvmonitor')
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True, with_mask=True),
    dict(type='Resize', img_scale=(1333, 800), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels', 'gt_masks']),
]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(1333, 800),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ])
]
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type=dataset_type,
        ann_file=data_root + 'VOC2007/pascal_train2007.json',
        img_prefix=data_root + 'VOC2007/JPEGImages/',
        pipeline=train_pipeline),
    val=dict(
        type=dataset_type,
        ann_file=data_root + 'VOC2007/pascal_val2007.json',
        img_prefix=data_root + 'VOC2007/JPEGImages/',
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root + 'VOC2007/pascal_test2007.json',
        img_prefix=data_root + 'VOC2007/JPEGImages/',
        pipeline=test_pipeline))
evaluation = dict(metric=['bbox', 'segm'])
```

Config_Mask_RCNN.py data information

```
load_from = "http://download.openmmlab.com/mmdetection/v2.0/mask_rcnn/mask_rcnn_r50_fpn_1x_coco/mask_r

# Set Schedule
# optimizer
optimizer = dict(type='SGD', lr=0.0025, momentum=0.9, weight_decay=0.0001)
optimizer_config = dict(grad_clip=None)
# learning policy
lr_config = dict(
    policy='step',
    warmup='linear',
    warmup_iters=500,
    warmup_ratio=0.001,
    step=[8, 11])
runner = dict(type='EpochBasedRunner', max_epochs=30)
```

Config_Mask_RCNN.py parameter information

When learned, the results are stored as follows for each epoch.



Training model information

After learning, I can see the learning results as follows.

```
/home/user/miniconda/envs/hh/lib/python3.7/site-packages/pycocotools/cocoeval.py:378: DeprecationWa
rning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `floa
t` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the n
umpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-n
otes.html#deprecations
  tp_sum = np.cumsum(tps, axis=1).astype(dtype=np.float)
DONE (t=1.16s).
2022-04-25 15:07:18,756 - mmdet - INFO -
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.332
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=1000 ] = 0.624
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=1000 ] = 0.308
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.100
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.246
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.412
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.445
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=300 ] = 0.445
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=1000 ] = 0.445
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=1000 ] = 0.130
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=1000 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=1000 ] = 0.536

2022-04-25 15:07:18,890 - mmdet - INFO - Exp name: Config_Mask_RCNN.py
2022-04-25 15:07:18,891 - mmdet - INFO - Epoch(val) [30][2510]  bbox_mAP: 0.3310, bbox_mAP_50: 0.62
70, bbox_mAP_75: 0.3170, bbox_mAP_s: 0.0900, bbox_mAP_m: 0.2460, bbox_mAP_l: 0.4100, bbox_mAP_copyp
aste: 0.331 0.627 0.317 0.090 0.246 0.410, segm_mAP: 0.3320, segm_mAP_50: 0.6240, segm_mAP_75: 0.30
80, segm_mAP_s: 0.1000, segm_mAP_m: 0.2460, segm_mAP_l: 0.4120, segm_mAP_copypaste: 0.332 0.624 0.3
08 0.100 0.246 0.412
```

Learning Results

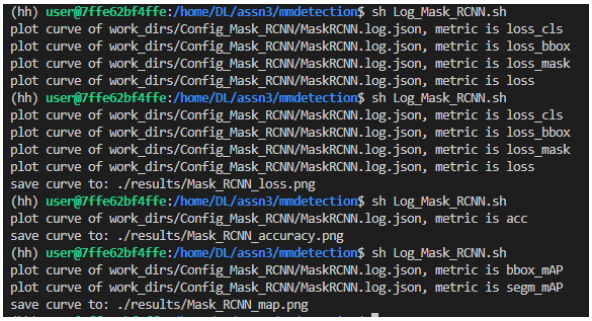
And in relation to the learning process, the following commands were given according to the graph I want to draw. From each command, graphs such as training loss, accurracy, and validation mAP were drawn and checked.

[Train Loss] python tools /analysis_tools/analyze_logs.py plot_curve

work_dirs/Config_Mask_RCNN/MaskRCNN.log.json --out ./results/Mask_RCNN_loss.png --keys loss_cls loss_bbox loss_mask loss --legend train_loss_cls train_loss_bbox train_loss_mask Total_loss

[Train Accuracy] python tools /analysis_tools/analyze_logs.py plot_curve work_dirs/Config_Mask_RCNN/MaskRCNN.log.json --out ./results/Mask_RCNN_accuracy.png --keys acc --legend Accuracy

[Validation mAP] python tools /analysis_tools/analyze_logs.py plot_curve work_dirs/Config_Mask_RCNN/MaskRCNN.log.json --out ./results/Faster_RCNN_map.png --keys bbox_mAP segm_mAP --legend bbox_mAP segm_mAP



Make result images

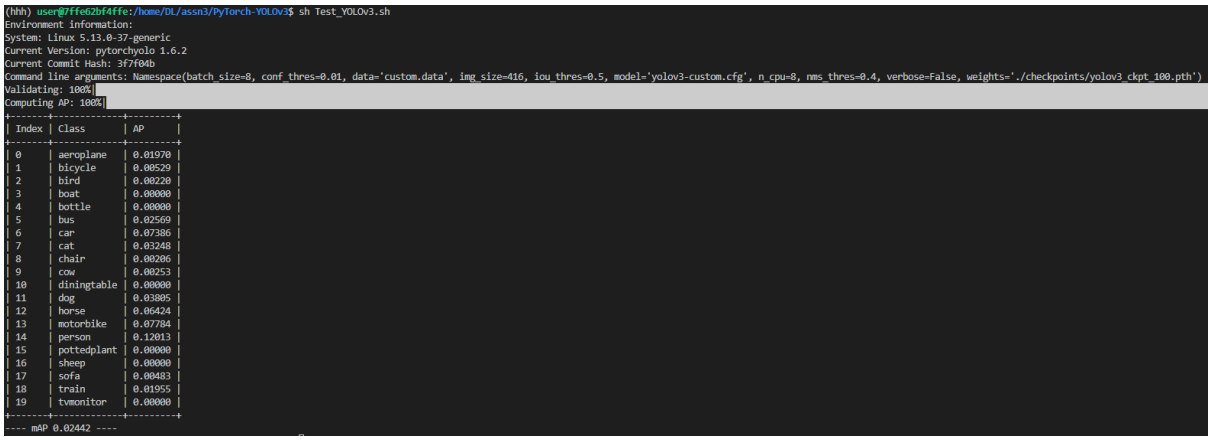
Finally, the mAP was checked according to each IoU threshold value through a test based on the model that was trained. Here, "epoch_30.pth" is the model information that was previously used for learning.

[test] python tools/test.py Config_Mask_RCNN.py work_dirs/Config_Mask_RCNN/epoch_30.pth --eval bbox --eval-options iou_thrs=[0.5](choose 0.5/0.6/0.7/0.8/0.9)

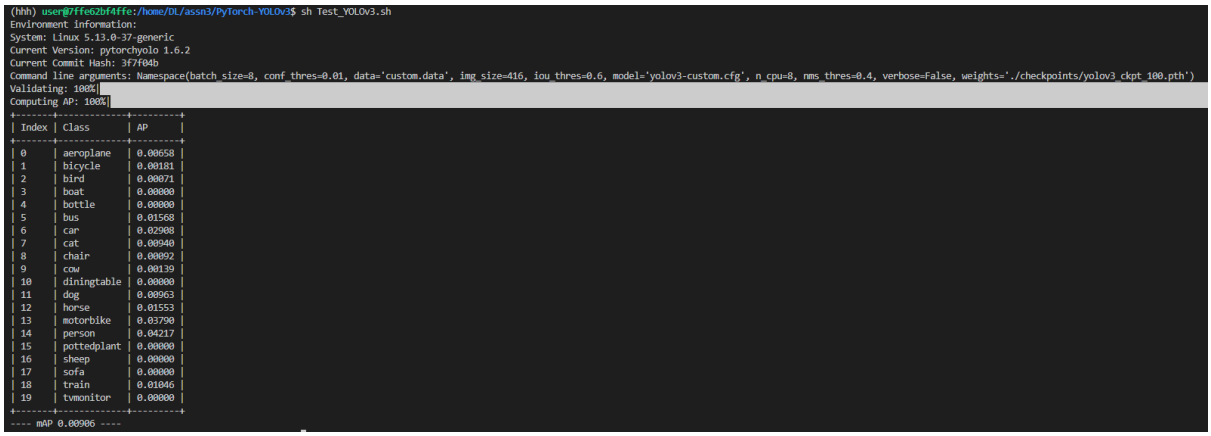
3. [Problem #3] Training and Testing YOLOv3 with darknet53 on Pascal VOC 2007 dataset

3.1. Fill out the following blanks in terms of mean average precision (mAP) and inference times (FPS) where mAP@# means that a prediction is positive if IoU ≥ # and discuss your experimental results

	mAP@0.5	mAP@0.6	mAP@0.7	mAP@0.8	mAP@0.9	FPS
YOLOv3(old)	0.02442	0.00906	0.00183	0.00013	0.00002	260.631



YOLOv3 mAP@0.5



YOLOv3 mAP@0.6

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn3/PyTorch-YOLOv3$ sh Test_YOLOv3.sh
Environment information:
System: Linux 5.13.0-37-generic
Current Version: pytorchyolo 1.6.2
Current Commit Hash: 3f7f04b
Command line arguments: Namespace(batch_size=8, conf_thres=0.01, data='custom.data', img_size=416, iou_thres=0.7, model='yolov3-custom.cfg', n_cpu=8, nms_thres=0.4, verbose=False, weights='./checkpoints/yolov3_ckpt_100.pth')
Validating: 100%|
Computing AP: 100%|
+-------+-------------+---------+
| Index | Class       | AP      |
+-------+-------------+---------+
| 0     | aeroplane   | 0.00059 |
| 1     | bicycle     | 0.00011 |
| 2     | bird        | 0.00011 |
| 3     | boat        | 0.00000 |
| 4     | bottle      | 0.00000 |
| 5     | bus         | 0.00000 |
| 6     | car         | 0.00337 |
| 7     | cat         | 0.00514 |
| 8     | chair       | 0.00014 |
| 9     | cow         | 0.00019 |
| 10    | diningtable | 0.00000 |
| 11    | dog         | 0.00348 |
| 12    | horse       | 0.00124 |
| 13    | motorbike   | 0.01313 |
| 14    | person      | 0.00817 |
| 15    | pottedplant | 0.00000 |
| 16    | sheep       | 0.00000 |
| 17    | sofa        | 0.00000 |
| 18    | train       | 0.00100 |
| 19    | tvmonitor   | 0.00000 |
+-------+-------------+---------+
---- mAP 0.00183 ----
```

YOLOv3 mAP@0.7

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn3/PyTorch-YOLOv3$ sh Test_YOLOv3.sh
Environment information:
System: Linux 5.13.0-37-generic
Current Version: pytorchyolo 1.6.2
Current Commit Hash: 3f7f04b
Command line arguments: Namespace(batch_size=8, conf_thres=0.01, data='custom.data', img_size=416, iou_thres=0.8, model='yolov3-custom.cfg', n_cpu=8, nms_thres=0.4, verbose=False, weights='./checkpoints/yolov3_ckpt_100.pth')
Validating: 100%|
Computing AP: 100%|
+-------+-------------+---------+
| Index | Class       | AP      |
+-------+-------------+---------+
| 0     | aeroplane   | 0.00000 |
| 1     | bicycle     | 0.00000 |
| 2     | bird        | 0.00000 |
| 3     | boat        | 0.00000 |
| 4     | bottle      | 0.00000 |
| 5     | bus         | 0.00000 |
| 6     | car         | 0.00044 |
| 7     | cat         | 0.00065 |
| 8     | chair       | 0.00001 |
| 9     | cow         | 0.00000 |
| 10    | diningtable | 0.00000 |
| 11    | dog         | 0.00006 |
| 12    | horse       | 0.00013 |
| 13    | motorbike   | 0.00047 |
| 14    | person      | 0.00051 |
| 15    | pottedplant | 0.00000 |
| 16    | sheep       | 0.00000 |
| 17    | sofa        | 0.00000 |
| 18    | train       | 0.00037 |
| 19    | tvmonitor   | 0.00000 |
+-------+-------------+---------+
---- mAP 0.00013 ----
```

YOLOv3 mAP@0.8

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn3/PyTorch-YOLOv3$ sh Test_YOLOv3.sh
Environment information:
System: Linux 5.13.0-37-generic
Current Version: pytorchyolo 1.6.2
Current Commit Hash: 3f7f04b
Command line arguments: Namespace(batch_size=8, conf_thres=0.01, data='custom.data', img_size=416, iou_thres=0.9, model='yolov3-custom.cfg', n_cpu=8, nms_thres=0.4, verbose=False, weights='./checkpoints/yolov3_ckpt_100.pth')
Validating: 100%|
Computing AP: 100%|
+-------+-------------+---------+
| Index | Class       | AP      |
+-------+-------------+---------+
| 0     | aeroplane   | 0.00000 |
| 1     | bicycle     | 0.00000 |
| 2     | bird        | 0.00000 |
| 3     | boat        | 0.00000 |
| 4     | bottle      | 0.00000 |
| 5     | bus         | 0.00000 |
| 6     | car         | 0.00002 |
| 7     | cat         | 0.00019 |
| 8     | chair       | 0.00000 |
| 9     | cow         | 0.00000 |
| 10    | diningtable | 0.00000 |
| 11    | dog         | 0.00000 |
| 12    | horse       | 0.00000 |
| 13    | motorbike   | 0.00009 |
| 14    | person      | 0.00001 |
| 15    | pottedplant | 0.00000 |
| 16    | sheep       | 0.00000 |
| 17    | sofa        | 0.00000 |
| 18    | train       | 0.00002 |
| 19    | tvmonitor   | 0.00000 |
+-------+-------------+---------+
---- mAP 0.00002 ----
```

YOLOv3 mAP@0.9

	mAP@0.5	mAP@0.6	mAP@0.7	mAP@0.8	mAP@0.9	FPS
YOLOv3(new)	0.673	0.632	0.557	0.421	0.184	99.04

```
val: Scanning 'data/custom/yolotest' images and labels...4952 found, 0 missing, 0 empty, 0 corrupted: 100%|
val: New cache created: data/custom/yolotest.cache
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|          | 155/155 [00:50<00:00,  3.08it/s]
                 all       4952      14976      0.802      0.617      0.673      0.452
           aeroplane       4952        311      0.917      0.704       0.78      0.517
             bicycle       4952        389      0.857      0.722      0.772      0.534
                bird       4952        576      0.811      0.535      0.602       0.36
                boat       4952        393      0.708      0.461      0.494      0.262
              bottle       4952        657      0.822      0.435      0.497        0.3
                 bus       4952        254      0.858      0.664      0.753      0.604
                 car       4952       1541      0.859      0.746      0.816      0.591
                 cat       4952        370      0.794      0.731      0.744      0.514
               chair       4952       1374      0.767      0.401      0.489      0.288
                 cow       4952        329      0.797      0.578      0.678      0.459
         diningtable       4952        299      0.845      0.502      0.577      0.409
                 dog       4952        530      0.826      0.626      0.744      0.511
               horse       4952        395      0.871      0.762      0.814      0.599
           motorbike       4952        369      0.819      0.729      0.777      0.509
              person       4952       5227      0.846      0.757      0.817      0.518
         pottedplant       4952        592      0.656      0.343      0.391      0.201
               sheep       4952        311      0.675      0.656      0.645      0.426
                sofa       4952        396      0.688      0.579      0.614      0.431
               train       4952        302      0.837      0.748      0.787      0.547
           tvmonitor       4952        361      0.779      0.659      0.672      0.466
Speed: 0.1ms pre-process, 3.9ms inference, 1.4ms NMS per image at shape (32, 3, 416, 416)
```

YOLOv3(new) mAP@0.5

```
val: Scanning 'data/custom/yolotest.cache' images and labels... 4952 found, 0 missing, 0 empty, 0 corrupted: 100%|
               Class     Images     Labels          P          R     mAP@.6 mAP@.5:.95: 100%|          | 155/155 [00:50<00:00,  3.07it/s]
                 all       4952      14976      0.769      0.593      0.632      0.407
           aeroplane       4952        311      0.892      0.685      0.742      0.462
             bicycle       4952        389      0.835      0.704      0.744      0.482
                bird       4952        576      0.771      0.509      0.543      0.305
                boat       4952        393      0.637      0.415      0.424      0.211
              bottle       4952        657      0.781      0.414      0.459      0.254
                 bus       4952        254      0.858      0.664      0.743      0.579
                 car       4952       1541       0.83       0.72      0.777      0.543
                 cat       4952        370      0.755      0.695      0.691       0.47
               chair       4952       1374      0.718      0.376      0.436      0.244
                 cow       4952        329      0.784      0.568       0.65      0.412
         diningtable       4952        299      0.817      0.485       0.55      0.377
                 dog       4952        530      0.799      0.606      0.707      0.465
               horse       4952        395      0.848      0.742      0.786      0.557
           motorbike       4952        369      0.783      0.696      0.724      0.451
              person       4952       5227      0.797      0.712      0.755      0.456
         pottedplant       4952        592      0.594      0.311       0.33       0.16
               sheep       4952        311      0.658       0.64      0.608      0.381
                sofa       4952        396      0.654      0.551      0.576      0.401
               train       4952        302      0.818      0.732      0.757        0.5
           tvmonitor       4952        361      0.759      0.643      0.643      0.422
 Speed: 0.1ms pre-process, 3.9ms inference, 1.4ms NMS per image at shape (32, 3, 416, 416)
```

YOLOv3(new) mAP@0.6

```
val: Scanning 'data/custom/yolotest.cache' images and labels... 4952 found, 0 missing, 0 empty, 0 corrupted: 100%|
               Class     Images     Labels          P          R     mAP@.7 mAP@.5:.95: 100%|          | 155/155 [00:50<00:00,  3.05it/s]
                 all       4952      14976      0.724      0.538      0.557      0.339
           aeroplane       4952        311      0.832      0.617      0.656      0.377
             bicycle       4952        389      0.791      0.635      0.659      0.405
                bird       4952        576      0.684      0.436       0.44      0.234
                boat       4952        393      0.534      0.331      0.307      0.149
              bottle       4952        657      0.715      0.358      0.381      0.193
                 bus       4952        254      0.835      0.626      0.698      0.535
                 car       4952       1541      0.794      0.672      0.702      0.476
                 cat       4952        370       0.74      0.653      0.637      0.404
               chair       4952       1374      0.681      0.336      0.373      0.188
                 cow       4952        329      0.736      0.517      0.578      0.343
         diningtable       4952        299      0.818      0.458      0.505      0.323
                 dog       4952        530      0.733      0.528      0.607      0.399
               horse       4952        395      0.822      0.709      0.744      0.481
           motorbike       4952        369      0.751      0.634      0.644      0.368
              person       4952       5227      0.718      0.623      0.638      0.366
         pottedplant       4952        592      0.523      0.259      0.249       0.11
               sheep       4952        311      0.609       0.58      0.543      0.311
                sofa       4952        396      0.643      0.508      0.511      0.347
               train       4952        302       0.77      0.666      0.674      0.419
           tvmonitor       4952        361      0.746      0.609      0.603      0.347
Speed: 0.1ms pre-process, 3.9ms inference, 1.4ms NMS per image at shape (32, 3, 416, 416)
```

YOLOv3(new) mAP@0.7

```
val: Scanning 'data/custom/yolotest.cache' images and labels... 4952 found, 0 missing, 0 empty, 0 corrupted: 100%|
               Class     Images     Labels          P          R     mAP@.8 mAP@.5:.95: 100%|          | 155/155 [00:50<00:00,  3.07it/s]
                 all       4952      14976      0.632      0.432      0.421      0.242
           aeroplane       4952        311      0.689      0.473      0.466      0.253
             bicycle       4952        389      0.694      0.514      0.506      0.283
                bird       4952        576      0.573      0.326      0.292      0.149
                boat       4952        393      0.423      0.232      0.181      0.084
              bottle       4952        657      0.566      0.259      0.255      0.112
                 bus       4952        254      0.848      0.594      0.654       0.46
                 car       4952       1541      0.719      0.569      0.577      0.373
                 cat       4952        370      0.657      0.522      0.481      0.303
               chair       4952       1374       0.56      0.247      0.236      0.111
                 cow       4952        329      0.631      0.416      0.421      0.229
         diningtable       4952        299      0.752      0.375      0.412      0.238
                 dog       4952        530       0.69      0.455      0.496      0.301
               horse       4952        395      0.758      0.613      0.622      0.362
           motorbike       4952        369      0.639      0.501      0.469      0.244
              person       4952       5227      0.595       0.48      0.452      0.248
         pottedplant       4952        592       0.37      0.157       0.13      0.057
               sheep       4952        311       0.49      0.428      0.364      0.213
                sofa       4952        396      0.623      0.412      0.421      0.275
               train       4952        302      0.707      0.568      0.525      0.311
           tvmonitor       4952        361      0.657      0.501      0.456      0.232
Speed: 0.1ms pre-process, 3.9ms inference, 1.4ms NMS per image at shape (32, 3, 416, 416)
```

YOLOv3(new) mAP@0.8

```
val: Scanning 'data/custom/yolotest.cache' images and labels... 4952 found, 0 missing, 0 empty, 0 corrupted: 100%|
               Class     Images     Labels          P          R      mAP@.9 mAP@.5:.95: 100%|          | 155/155 [00:49<00:00,  3.11it/s]
                 all       4952      14976       0.37       0.24      0.184      0.105
           aeroplane       4952        311      0.358      0.235      0.178     0.0826
             bicycle       4952        389        0.4       0.28      0.209       0.11
                bird       4952        576       0.33      0.175      0.108     0.0579
                boat       4952        393      0.187     0.0891     0.0501     0.0258
              bottle       4952        657      0.224     0.0959     0.0523     0.0239
                 bus       4952        254       0.71      0.464      0.435      0.281
                 car       4952       1541      0.475      0.361       0.33      0.212
                 cat       4952        370      0.469      0.349      0.249       0.15
               chair       4952       1374      0.278      0.115     0.0682     0.0324
                 cow       4952        329      0.332       0.21      0.162     0.0812
         diningtable       4952        299      0.446      0.211      0.174      0.112
                 dog       4952        530      0.488      0.292      0.241       0.13
               horse       4952        395      0.445      0.342      0.271      0.156
           motorbike       4952        369      0.361      0.262      0.187     0.0943
              person       4952       5227      0.311      0.238      0.176      0.104
         pottedplant       4952        592      0.164     0.0591     0.0308     0.0138
               sheep       4952        311      0.263      0.215      0.148     0.0787
                sofa       4952        396      0.436      0.263      0.233      0.144
               train       4952        302       0.41      0.311      0.224      0.126
           tvmonitor       4952        361      0.307      0.227      0.146     0.0841
Speed: 0.1ms pre-process, 3.9ms inference, 1.3ms NMS per image at shape (32, 3, 416, 416)
```

YOLOv3(new) mAP@0.9

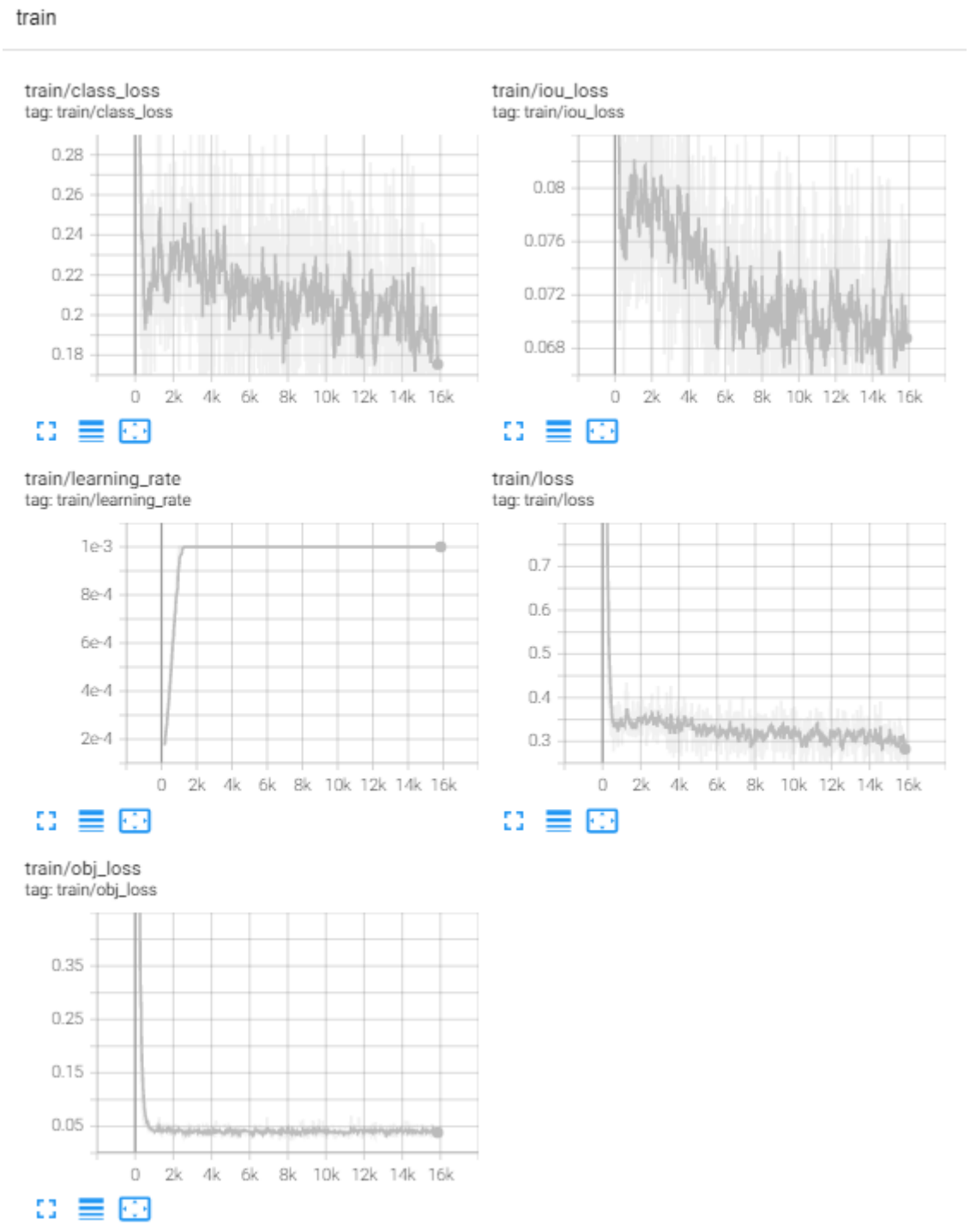
3.2. Try the efforts to improve the performances on your network models, such as your learning techniques or your network improvements that are not provided by basic codes

A. Show learning curves for training and validation

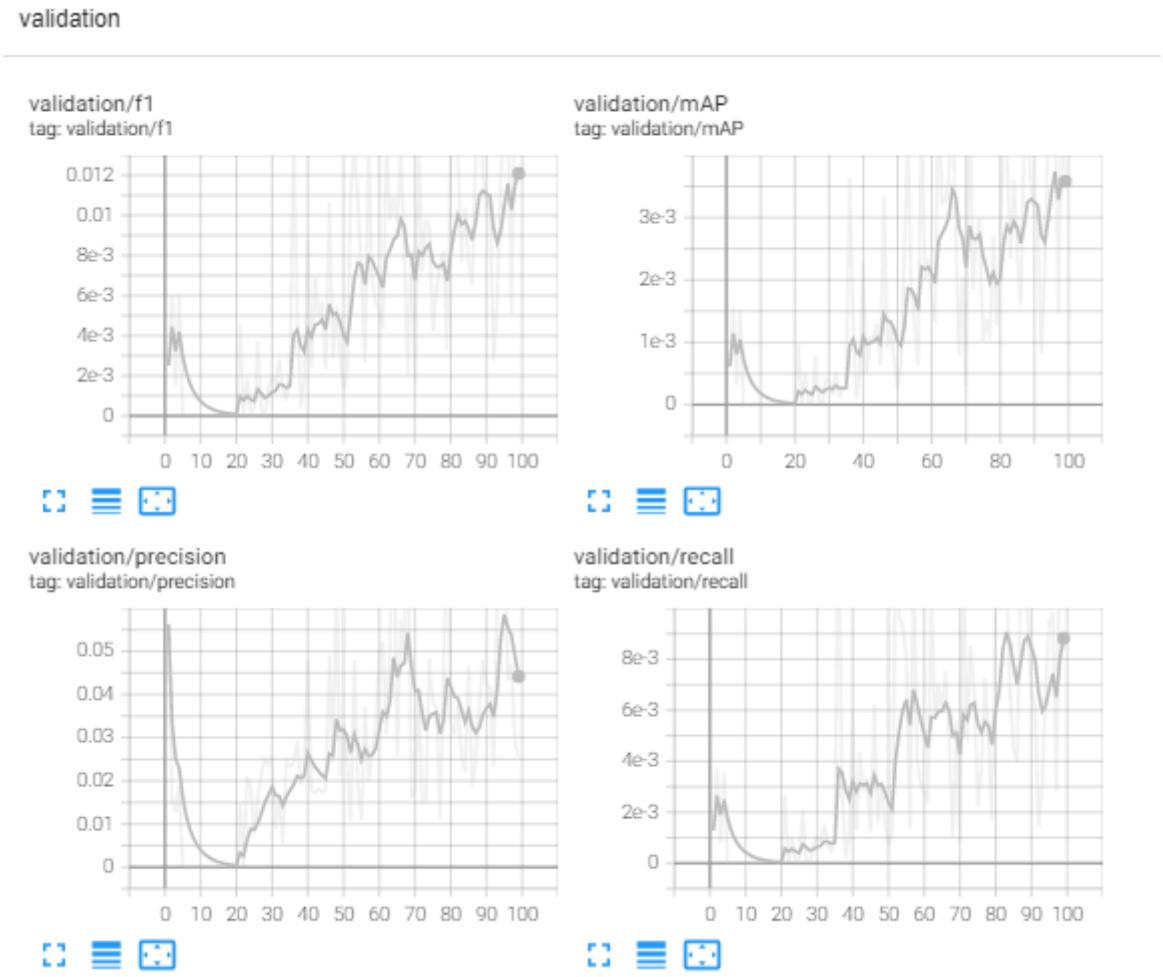
< YOLOv3(old) >

```
(hhh) user@7ffe62bf4ffe:/home/DL/assn3/PyTorch-YOLOv3$ poetry run tensorboard --logdir='logs' --port=6006
TensorFlow installation not found - running with reduced feature set.

NOTE: Using experimental fast data loading logic. To disable, pass
    "--load_fast=false" and report issues on GitHub. More details:
    https://github.com/tensorflow/tensorboard/issues/4784

Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.8.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

Tensor Board Visualization Command

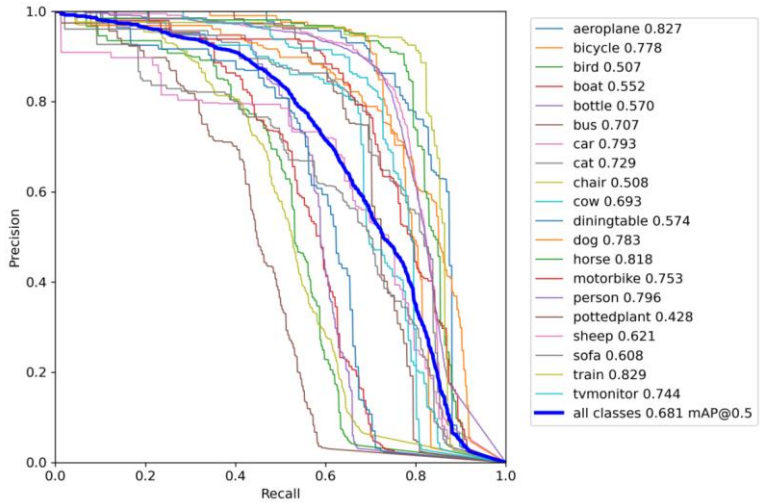


Tensor Board Training Result with 100 epochs

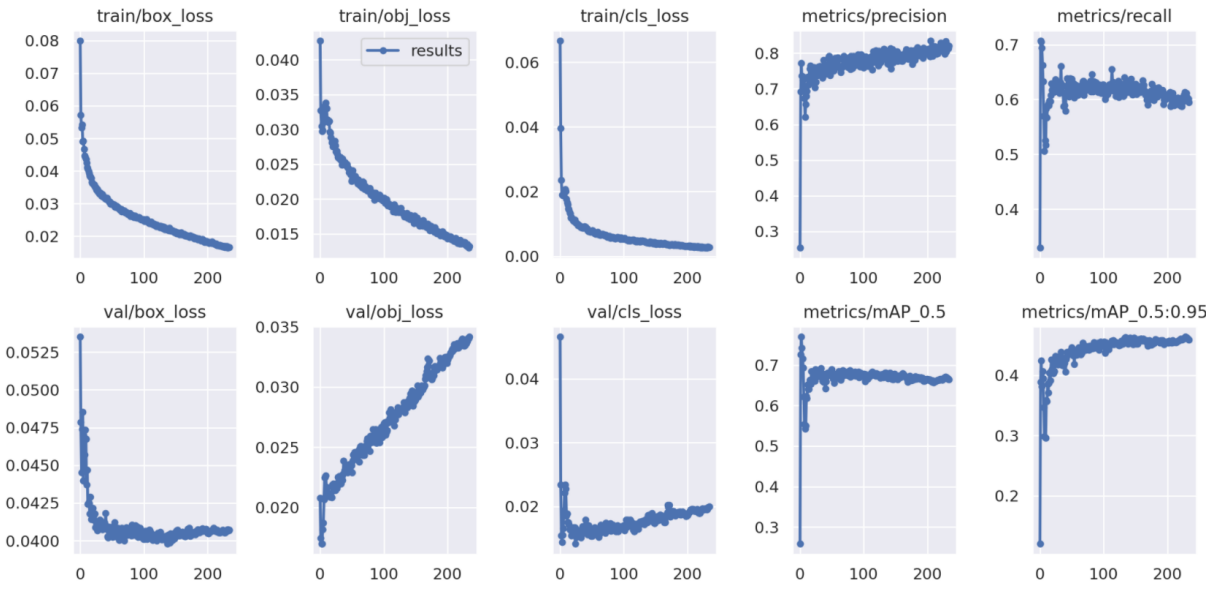


Tensor Board Validation Result with 100 epochs

< Yolov3 (new) >



YOLOv3 Training PR Curve



YOLOv3 Training Result

B. Show your source codes and trained model parameters

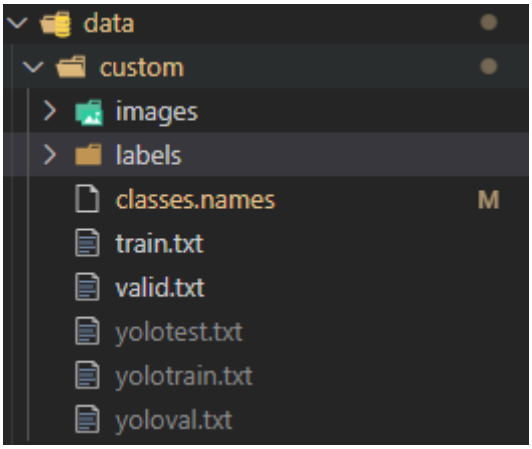
I downloaded YOLOv3 from github(GitHub - eriklindernoren/PyTorch-YOLOv3: Minimal PyTorch implementation of YOLOv3) and proceeded. This code is a code that can simply execute yolov3 using COCO dataset. Therefore, Pascal VOC 2007 data was converted into the same format as COCO data and proceeded. The Convert2Yolo(https://github.com/ssaru/convert2Yolo) code was used to convert the Dataset. Using this, Pascal VOC 2007 dataset was converted like COCO data.

```
f = open('train.txt', 'r')
# f = open('test.txt', 'r')
# f = open('val.txt', 'r')

line_data = f.readline()

while line_data:
    print("data/custom/images/" + line_data[:-1] + ".jpg")
    line_data = f.readline()

f.close()
```

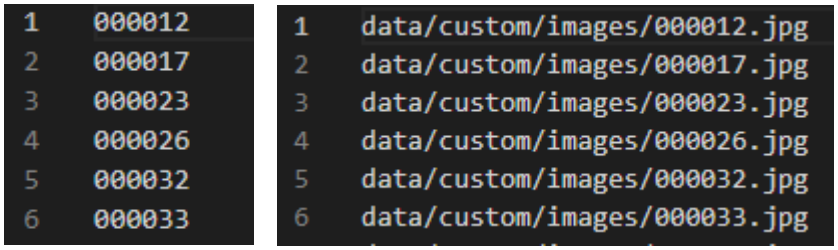
Make_datset.py

I generated txt data in YOLOv3/VOCconvert2yolo/labels/ from the attached file. It then copied and used all of the above generated txt files in PyTorch-YOLOv3/data/custom/labels/. And all the images from the previous assignment VOCdevkit/VOC2007/JPEGImages/ were copied and used inside PyTorch-YOLOv3/data/custom/images/. Additionally, train.txt, test.txt, and val.txt in VOCdevkit/VOC2007/ImageSets/Main/ were modified using YOLOv3/makedataset.py to modify the path. The modified new files were copied into PyTorch-YOLOv3/data/custom/ from yolotrain.txt,

yolotest.txt, and yoloval.txt files.

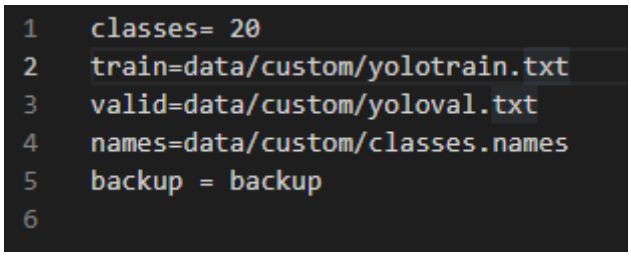


Configuration of PyTorch-YOLOv3/data/



train.txt (Left) / yolotrain.txt (Right)

It then ran PyTorch-YOLOv3/config/create_custom_model.sh to generate the same config file as yolov3-custom.cfg and copied it to PyTorch-YOLOv3. Additionally, the custom.data file was also used in the same location. In the custom.data file, the path of train and validation was set.



custom.data

The following is the information from learning the YOLOv3 model, and pretrained-weights were used together.

[training] python ./pytorchyolo/train.py --model yolov3-custom.cfg --data custom.data --pretrained_weights darknet53.conv.74

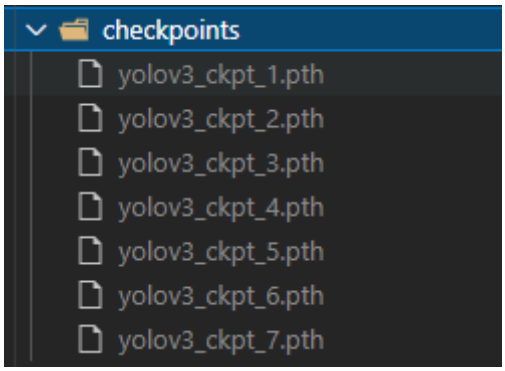
```
(hhh) user@7ffe62bf4ffe:/home/DL/assn3/PyTorch-YOLOv3$ sh Train_YOLOv3.sh
Environment information:
System: Linux 5.13.0-37-generic
Current Version: pytorchyolo 1.6.2
Current Commit Hash: 3f7f04b
Command line arguments: Namespace(checkpoint_interval=1, conf_thres=0.1, data='custom.data', epochs=100, evaluation_interval=1, iou_thres=0.5, logdir='lo
gs', model='yolov3-custom.cfg', multiscale_training=False, n_cpu=8, nms_thres=0.5, pretrained_weights='darknet53.conv.74', seed=-1, verbose=False)
```

YOLOv3 Train Summary with hyperparameters

Currently, I have searched various ISSUE boards related to YOLOv3 model. As a result, there were many opinions that this model and VOC dataset are not well learned. There was an opinion that this algorithm does not fit well with the current VOC dataset. And there was also an opinion that even if the epoch was increased, it would not improve significantly. I could get a lot of opinions through the following link.

https://github.com/eriklindernoren/PyTorch-YOLOv3/issues/75

And the weights were stored as checkpoints for each epoch in which learning was conducted.



Checkpoints(.pth) with YOLOv3 model

assn3 > PyTorch-YOLOv3 > Test_YOLOv3.sh

```
1  # python ./pytorchyolo/test.py --model yolov3-custom.cfg --data custom.data --weights ./checkpoints/yolov3_ckpt_100.pth --iou_thres 0.5
2
3  # python ./pytorchyolo/test.py --model yolov3-custom.cfg --data custom.data --weights ./checkpoints/yolov3_ckpt_100.pth --iou_thres 0.6
4
5  # python ./pytorchyolo/test.py --model yolov3-custom.cfg --data custom.data --weights ./checkpoints/yolov3_ckpt_100.pth --iou_thres 0.7
6
7  # python ./pytorchyolo/test.py --model yolov3-custom.cfg --data custom.data --weights ./checkpoints/yolov3_ckpt_100.pth --iou_thres 0.8
8
9  python ./pytorchyolo/test.py --model yolov3-custom.cfg --data custom.data --weights ./checkpoints/yolov3_ckpt_100.pth --iou_thres 0.9
```

Evaluation .sh file

Based on the existing reference, the performance did not come out well, so I downloaded a new model and proceeded with it. We proceeded based on the following link.

https://github.com/ultralytics/yolov3

Data from the existing reference were copied and used in yolov3/data. In other words, the entire data in PyTorch-YOLOv3/data/custom was taken and used. And I modified the data path in voc.yaml.

```yaml
# YOLOv3 🚀 by Ultralytics, GPL-3.0 license
# PASCAL VOC dataset http://host.robots.ox.ac.uk/pascal/VOC
# Example usage: python train.py --data VOC.yaml
# parent
# ├── yolov3
# └── datasets
#     └── VOC  ← downloads here


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
train: # train images (relative to 'path')  16551 images
  - data/custom/yolotrain.txt
val: # val images (relative to 'path')  4952 images
  - data/custom/yoloval.txt
test: # test images (optional)
  - data/custom/yolotest.txt

# Classes
nc: 20  # number of classes
names: ['aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog',
        'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor']  # class names
```

Modified voc.yaml

Then, in order to use the cfg file as it is, the yolov3-custom.cfg file was copied and used. But it didn't matter to use models/yolov3.yaml file.

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=16
subdivisions=1
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1
```

YOLOv3 network parameter

In order to learn model, .sh file was created and learned through the following command. Basically, I learned 300 epochs.

YOLOv3 Train Summary

[training] python train.py --data ./data/voc.yaml --cfg ./models/yolov3.yaml --weight yolov3.pt --

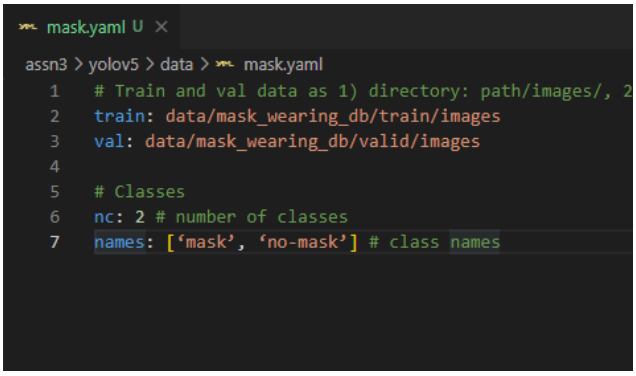
device 0

Each mAP was checked using the model in which the learning was completed. The test was conducted using the weight of the model, runs/train/exp9/weights/best.pt. The results were confirmed by changing the mAP values to 0.5, 0.6, 0.7, 0.8 and 0.9 in val.py.

[test] python val.py --data ./data/voc.yaml --weight ./runs/train/exp9/weights/best.pt

4. [Problem #4] Training and Testing YOLOv5 on Mask Wearing Dataset

In order to use the YOLOv5 model, a yaml file must be made as follows. To use the mask wearing dataset, the number of paths, classes, and names were set.

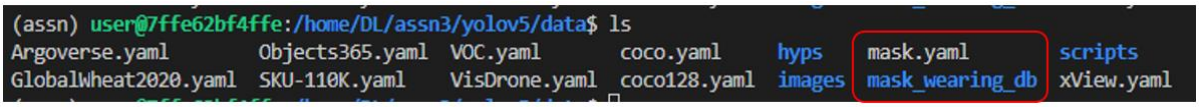


mask.yaml

And actually, in the case of data set, I downloaded it through following link.

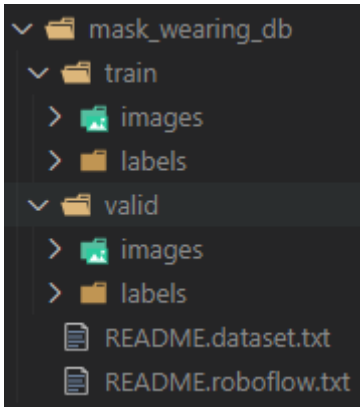
http://imlab.postech.ac.kr:5000/sharing/NGQtxtK5B.

Then, mask.yaml file and mask_wearing_db folder are ready.



yolov5/data/mask.yaml and mask_wearing_db

In the case of mask_wearing_db file, it is configured as follows.



configuration of mask_wearing_db

And I can train the YOLOv5s model. Here, the hyper-parameter is as follows.

```
hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box
=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, t
ranslate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
```

Hyperparameters of yolov5s(Train)

After the training, the learning results are as follows.

```
YOLOv5s summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|          | 1/1 [00:00<00:00,  4.83it/s]
                 all         29        162      0.866       0.68      0.811      0.501
              'mask'         29        142      0.932      0.761      0.871      0.526
           'no-mask'         29         20        0.8        0.6      0.751      0.476
```

Train Summary with mask.yaml(data), yolov5s.yaml(cfg), yolov5s.pt(weight)

```
YOLOv5x summary: 444 layers, 86180143 parameters, 0 gradients, 204.0 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|          | 1/1 [00:00<00:00,  3.90it/s]
                 all         29        162      0.812      0.837      0.861      0.548
              'mask'         29        142      0.851      0.824      0.886        0.6
           'no-mask'         29         20      0.773       0.85      0.836      0.496
```

Train Summary with mask.yaml(data), yolov5x.yaml(cfg), yolov5x.pt(weight)

In the process of actually evaluating using the learned model, the performance was confirmed by changing the IOU threshold value corresponding to the mAP.

```python
# Configure
model.eval()
cuda = device.type != 'cpu'
is_coco = isinstance(data.get('val'), str) and data['val'].endswith(f'coco{os.sep}val2017.txt')  # COCO dataset
nc = 1 if single_cls else int(data['nc'])  # number of classes
iouv = torch.linspace(0.5, 0.95, 10, device=device)  # iou vector for mAP@0.5:0.95
niou = iouv.numel()

# Dataloader
if not training:
    if pt and not single_cls:  # check --weights are trained on --data
        ncm = model.model.nc
        assert ncm == nc, f'{weights[0]} ({ncm} classes) trained on different --data than what you passed ({nc} ' \
                          f'classes). Pass correct combination of --weights and --data that are trained together.'
    model.warmup(imgsz=(1 if pt else batch_size, 3, imgsz, imgsz))  # warmup
    pad = 0.0 if task in ('speed', 'benchmark') else 0.5
    rect = False if task == 'benchmark' else pt  # square inference for benchmarks
    task = task if task in ('train', 'val', 'test') else 'val'  # path to train/val/test images
    dataloader = create_dataloader(data[task],
                                   imgsz,
                                   batch_size,
                                   stride,
                                   single_cls,
                                   pad=pad,
                                   rect=rect,
                                   workers=workers,
                                   prefix=colorstr(f'{task}: '))[0]

seen = 0
confusion_matrix = ConfusionMatrix(nc=nc)
names = {k: v for k, v in enumerate(model.names if hasattr(model, 'names') else model.module.names)}
class_map = coco80_to_coco91_class() if is_coco else list(range(1000))
s = ('%20s' + '%11s' * 6) % ('Class', 'Images', 'Labels', 'P', 'R', 'mAP@.5', 'mAP@.5:.95')
dt, p, r, f1, mp, mr, map50, map = [0.0, 0.0, 0.0], 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

4.1. Fill out the following blanks in terms of mean average precision (mAP) and inference times

(FPS) where mAP@# means that a prediction is positive if IoU ≥ # and discuss your experimental results

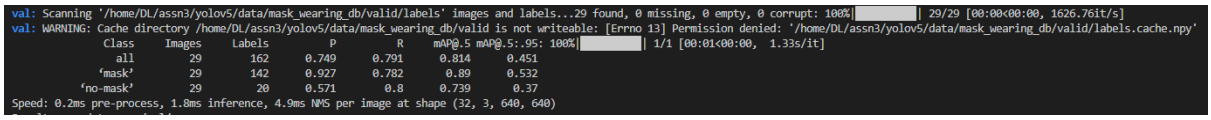
In general, training and test were conducted through the following command.

[training] python train.py --data mask.yaml --img-size 416 --cfg ./models/yolov5s.yaml --weight yolov5s.pt --device 0

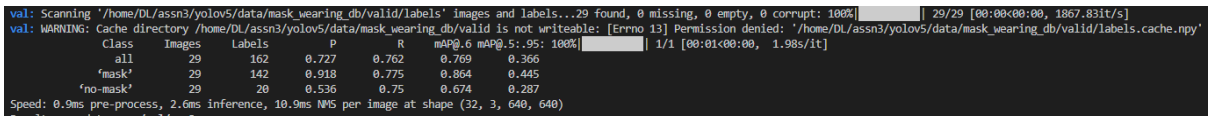
[test] python val.py --data mask.yaml --weight ./runs/train/exp2/weights/best.pt

The following mAP results can then be obtained.

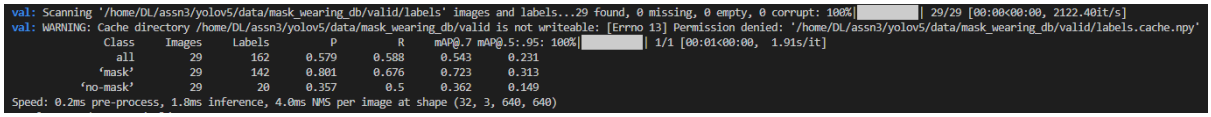
	mAP@0.5	mAP@0.6	mAP@0.7	mAP@0.8	mAP@0.9	FPS
YOLOv5s	0.814	0.769	0.543	0.298	0.0169	1721.062



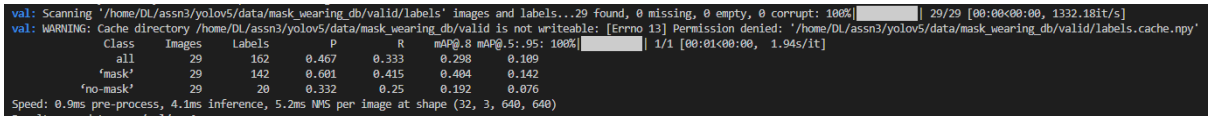
YOLOv5s mAP@0.5



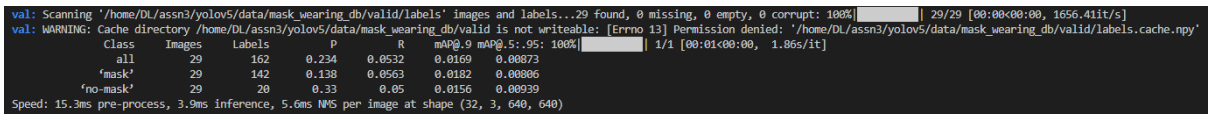
YOLOv5s mAP@0.6



YOLOv5s mAP@0.7



YOLOv5s mAP@0.8



YOLOv5s mAP@0.9

	mAP@0.5	mAP@0.6	mAP@0.7	mAP@0.8	mAP@0.9	FPS
YOLOv5x	0.811	0.779	0.691	0.413	0.0487	1445.598

```
val: Scanning '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels' images and labels...29 found, 0 missing, 0 empty, 0 corrupt: 100%|██████████| 29/29 [00:00<00:00, 1277.46it/s]
val: WARNING: Cache directory /home/DL/assn3/yolov5/data/mask_wearing_db/valid is not writeable: [Errno 13] Permission denied: '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels.cache.npy'
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|██████████| 1/1 [00:02<00:00,  2.04s/it]
                 all         29        162      0.858      0.705      0.811      0.506
              'mask'         29        142      0.915      0.761      0.866      0.588
           'no-mask'         29         20        0.8       0.65      0.756      0.423
Speed: 1.0ms pre-process, 11.6ms inference, 5.9ms NMS per image at shape (32, 3, 640, 640)
```

YOLOv5x mAP@0.5

```
val: Scanning '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels' images and labels...29 found, 0 missing, 0 empty, 0 corrupt: 100%|██████████| 29/29 [00:00<00:00, 1374.78it/s]
val: WARNING: Cache directory /home/DL/assn3/yolov5/data/mask_wearing_db/valid is not writeable: [Errno 13] Permission denied: '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels.cache.npy'
               Class     Images     Labels          P          R     mAP@.6 mAP@.5:.95: 100%|██████████| 1/1 [00:01<00:00,  1.89s/it]
                 all         29        162      0.845      0.695      0.779       0.43
              'mask'         29        142       0.89      0.739      0.825      0.528
           'no-mask'         29         20        0.8       0.65      0.733      0.331
Speed: 0.9ms pre-process, 11.1ms inference, 6.0ms NMS per image at shape (32, 3, 640, 640)
```

YOLOv5x mAP@0.6

```
val: Scanning '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels' images and labels...29 found, 0 missing, 0 empty, 0 corrupt: 100%|██████████| 29/29 [00:00<00:00, 1526.62it/s]
val: WARNING: Cache directory /home/DL/assn3/yolov5/data/mask_wearing_db/valid is not writeable: [Errno 13] Permission denied: '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels.cache.npy'
               Class     Images     Labels          P          R     mAP@.7 mAP@.5:.95: 100%|██████████| 1/1 [00:01<00:00,  1.41s/it]
                 all         29        162      0.782      0.638      0.691      0.309
              'mask'         29        142       0.88      0.725      0.803      0.425
           'no-mask'         29         20      0.684       0.55      0.579      0.193
Speed: 1.0ms pre-process, 11.1ms inference, 4.2ms NMS per image at shape (32, 3, 640, 640)
```

YOLOv5x mAP@0.7

```
val: Scanning '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels' images and labels...29 found, 0 missing, 0 empty, 0 corrupt: 100%|██████████| 29/29 [00:00<00:00, 1323.64it/s]
val: WARNING: Cache directory /home/DL/assn3/yolov5/data/mask_wearing_db/valid is not writeable: [Errno 13] Permission denied: '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels.cache.npy'
               Class     Images     Labels          P          R     mAP@.8 mAP@.5:.95: 100%|██████████| 1/1 [00:02<00:00,  2.16s/it]
                 all         29        162      0.452      0.542      0.413      0.153
              'mask'         29        142      0.582      0.585      0.539       0.24
           'no-mask'         29         20      0.323        0.5      0.286     0.0659
Speed: 0.2ms pre-process, 11.0ms inference, 14.7ms NMS per image at shape (32, 3, 640, 640)
```

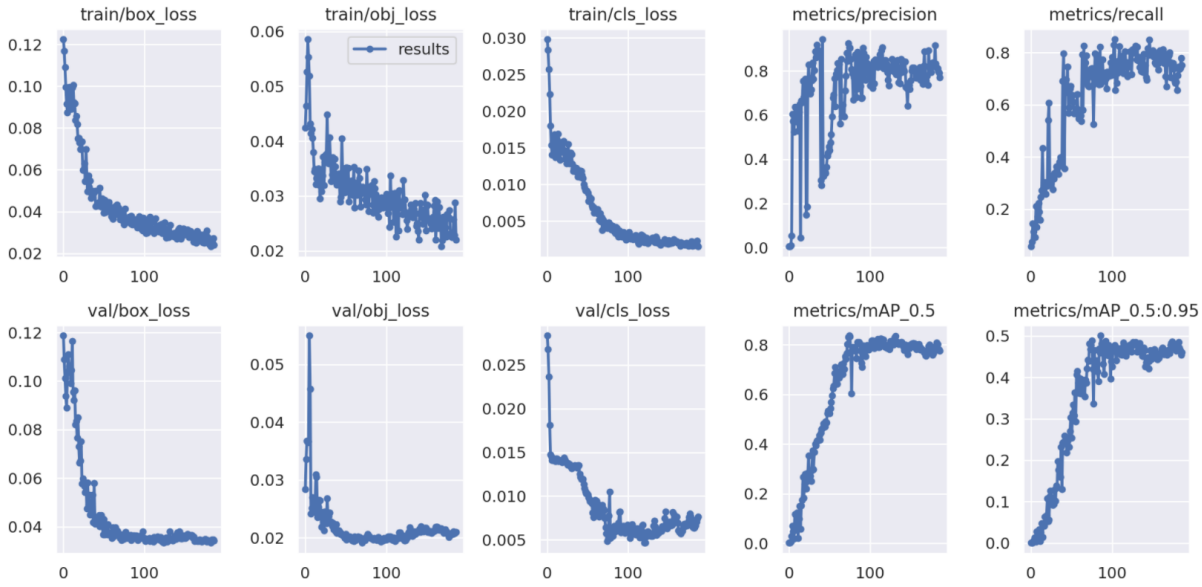
YOLOv5x mAP@0.8

```
val: Scanning '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels' images and labels...29 found, 0 missing, 0 empty, 0 corrupt: 100%|██████████| 29/29 [00:00<00:00, 1725.49it/s]
val: WARNING: Cache directory /home/DL/assn3/yolov5/data/mask_wearing_db/valid is not writeable: [Errno 13] Permission denied: '/home/DL/assn3/yolov5/data/mask_wearing_db/valid/labels.cache.npy'
               Class     Images     Labels          P          R     mAP@.9 mAP@.5:.95: 100%|██████████| 1/1 [00:02<00:00,  2.03s/it]
                 all         29        162      0.157     0.0986     0.0487     0.0199
              'mask'         29        142      0.314      0.197     0.0975     0.0399
           'no-mask'         29         20          0          0          0          0
Speed: 12.7ms pre-process, 18.0ms inference, 5.8ms NMS per image at shape (32, 3, 640, 640)
```

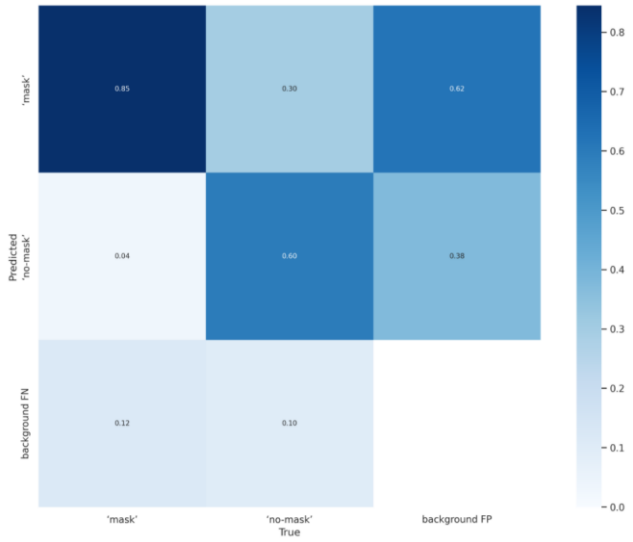
YOLOv5x mAP@0.9

4.2. Show training losses (regression, objectness and classification) and performance metrics (precision and recall)

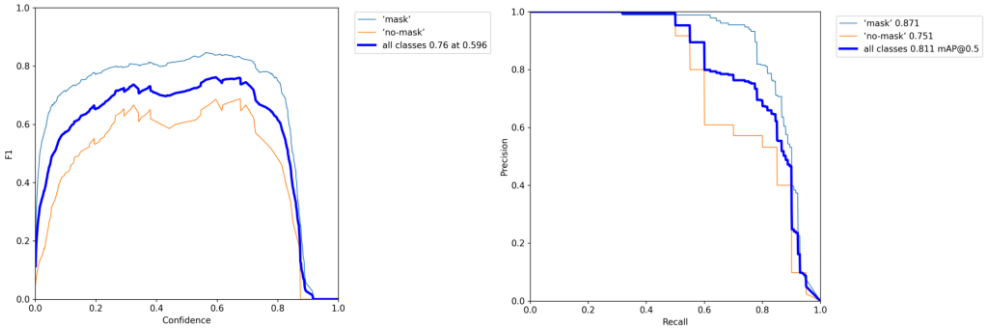
The following is the result of training loss and precision-recall curve after learning YOLOv5s model.



YOLOv5s Training Loss



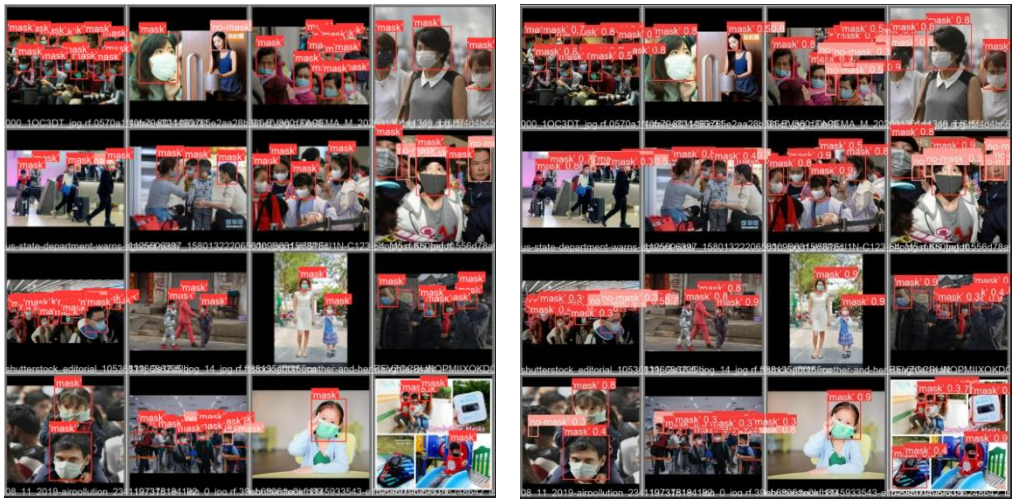
YOLOv5s Confusion Matrix



YOLOv5s F1 Curve (Left) / YOLOv5s PR Curve (Right)

4.3. Show the resulting images

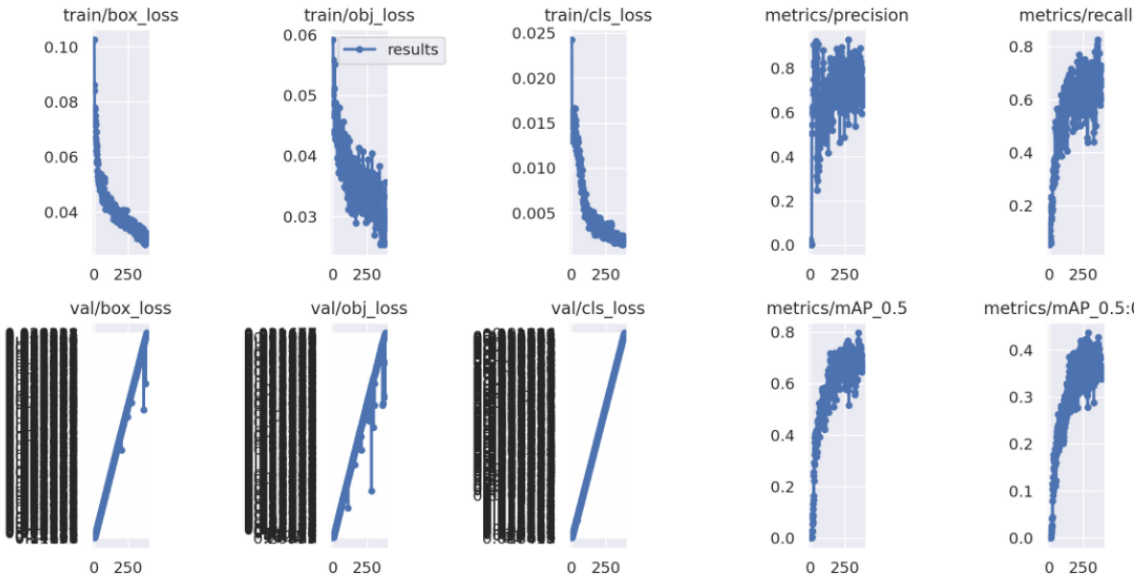
The following is the result of the actual label and model prediction after learning the YOLOv5s model.



YOLOv5s Result (Labels (Left) / Prediction (Right))

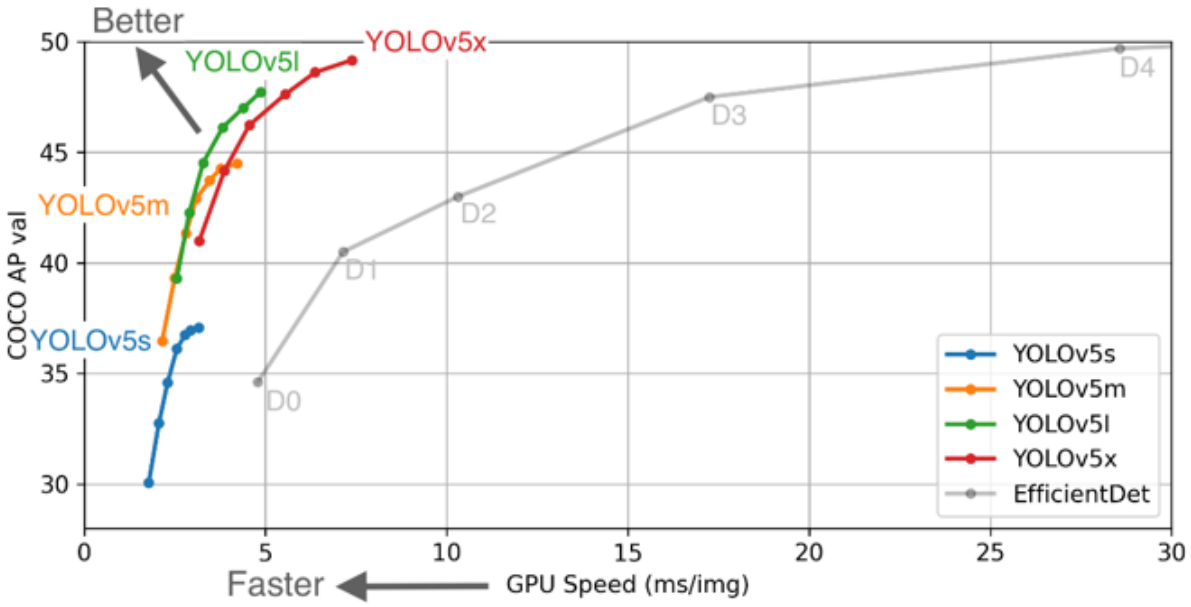
4.4. Try the efforts to improve the performances on your network models, such as your learning techniques or your network improvements that are not provided by basic codes

In the case of the initial model of YOLOv5s, 300 epochs were learned. Subsequently, the hyperparameter was modified or further learned to improve performance. However, it can be seen that the performance is not good because it has already been overfitting. Various attempts have been made, such as increasing the epoch or changing the optimizer to Adam.



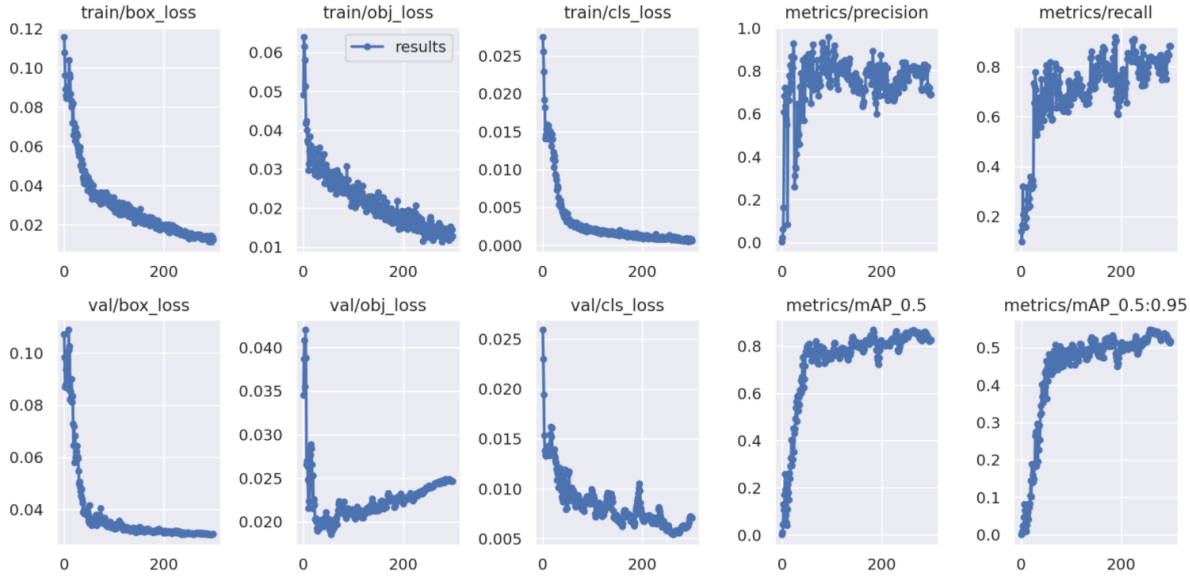
YOLOv5s Additional Training Loss

So I changed the model from YOLOv5s to YOLOv5x and trained it. Looking at the performance comparison on the reference page, it can be seen that YOLOv5x has the best performance. Mask dataset was also worth training with a better AP performance yolov5x because of its high FPS.



YOLOv5 Model Comparison

Still, it was confirmed that the mAP results were better than that of yolov5s. The comparison results are summarized in the table above.



YOLOv5x Training Loss

p.s. The learning information of the model I want to submit is so large that I saved it last. So, I submitted only the last .pth file, etc.

```
simjy98@csemlmi4:~$ du -sh submission/*
14G     submission/FasterRCNN_and_MaskRCNN
403M    submission/YOLOv3_new
23G     submission/YOLOv3_old
599M    submission/YOLOv5
simjy98@csemlmi4:~$ ls
GNN_Study_2022  submission
simjy98@csemlmi4:~$ cd submission
simjy98@csemlmi4:~/submission$ ls
FasterRCNN_and_MaskRCNN  YOLOv3_new  YOLOv3_old  YOLOv5
simjy98@csemlmi4:~/submission$ du -sh YOLOv3_old
23G     YOLOv3_old
simjy98@csemlmi4:~/submission$ cd YOLOv3_old
simjy98@csemlmi4:~/submission/YOLOv3_old$ du -sh *
23G     checkpoints
4.0K    custom.data
4.0K    make_dataset.py
4.0K    Test_YOLOv3.sh
4.0K    Train_YOLOv3.sh
148K    yolotest.txt
76K     yolotrain.txt
12K     yolov3-custom.cfg
76K     yoloval.txt
simjy98@csemlmi4:~/submission/YOLOv3_old$
```

My Submission(before)