

# Homework 2 - Object Detection

系級:智能系統 學號:312581006 姓名:張宸瑋

## ● Experiment Setup (Data pre-process, Hyperparameters,...)

### Without SE module

```
def preproc(img, input_size, swap=(2, 0, 1)):
    # 如果圖片是三通道的
    if len(img.shape) == 3:
        # 創建一個全為灰色 (114) 的三通道圖片 · 大小為指定的input_size
        padded_img = np.ones((input_size[0], input_size[1], 3), dtype=np.uint8) * 114
    else:
        # 如果是單通道的 · 則創建一個單通道的灰色圖片 · 大小為指定的input_size
        padded_img = np.ones(input_size, dtype=np.uint8) * 114

    # 計算將原始圖片縮放到模型指定輸入大小的縮放比例
    r = min(input_size[0] / img.shape[0], input_size[1] / img.shape[1])

    # 使用OpenCV的resize函數 · 將原始圖片按照計算得到的縮放比例進行縮放
    resized_img = cv2.resize(
        img,
        (int(img.shape[1] * r), int(img.shape[0] * r)),
        interpolation=cv2.INTER_LINEAR,
    ).astype(np.uint8)

    # 在預處理後的圖片中將縮放後的圖片放置到左上角 · 未被覆蓋的地方保持灰色
    padded_img[: int(img.shape[0] * r), : int(img.shape[1] * r)] = resized_img

    # 將預處理後的圖片進行通道交換 · 預設為(2, 0, 1) · 即將原始圖片的通道順序由HWC轉換為CHW
    padded_img = padded_img.transpose(swap)

    # 將預處理後的圖片轉換為連續的內存佈局 · 並轉換數據類型為float32
    padded_img = np.ascontiguousarray(padded_img, dtype=np.float32)

    # 返回預處理後的圖片和縮放比例
    return padded_img, r
```

### Data pre-process

		seed	None
		output_dir	'./YOLOX_outputs'
		print_interval	10
		eval_interval	1
		dataset	None
		num_classes	1
		depth	0.33
		width	0.5
		act	'silu'
		data_num_workers	4
		input_size	(704, 704)
		multiscale_range	5
		data_dir	'datasets/HW2_ObjectDetection_2023'
		train_ann	'train_labels.json'
		val_ann	'val_labels.json'
		test_ann	'instances_test2017.json'
		mosaic_prob	1.0
		mixup_prob	1.0
		hsv_prob	1.0
		flip_prob	0.5
		degrees	10.0
		translate	0.1
		mosaic_scale	(0.8, 2)

enable_mixup	True
mixup_scale	(0.5, 1.5)
shear	2.0
warmup_epochs	5
max_epoch	300
warmup_lr	0
min_lr_ratio	0.05
basic_lr_per_img	0.00015625
scheduler	'yoloxwarmcos'
no_aug_epochs	15
ema	True
weight_decay	0.0005
momentum	0.9
save_history_ckpt	False
exp_name	'yolox_s'
test_size	(704, 704)
test_conf	0.01
nmsthre	0.65

## Hyperparameters

## With SE module

```
def preproc(img, input_size, swap=(2, 0, 1)):
    # 如果圖片是三通道的
    if len(img.shape) == 3:
        # 創建一個全為灰色 (114) 的三通道圖片 · 大小為指定的input_size
        padded_img = np.ones((input_size[0], input_size[1], 3), dtype=np.uint8) * 114
    else:
        # 如果是單通道的 · 則創建一個單通道的灰色圖片 · 大小為指定的input_size
        padded_img = np.ones(input_size, dtype=np.uint8) * 114

    # 計算將原始圖片縮放到模型指定輸入大小的縮放比例
    r = min(input_size[0] / img.shape[0], input_size[1] / img.shape[1])

    # 使用OpenCV的resize函數 · 將原始圖片按照計算得到的縮放比例進行縮放
    resized_img = cv2.resize(
        img,
        (int(img.shape[1] * r), int(img.shape[0] * r)),
        interpolation=cv2.INTER_LINEAR,
    ).astype(np.uint8)

    # 在預處理後的圖片中將縮放後的圖片放置到左上角 · 未被覆蓋的地方保持灰色
    padded_img[: int(img.shape[0] * r), : int(img.shape[1] * r)] = resized_img

    # 將預處理後的圖片進行通道交換 · 預設為(2, 0, 1) · 即將原始圖片的通道順序由HWC轉換為CHW
    padded_img = padded_img.transpose(swap)

    # 將預處理後的圖片轉換為連續的內存佈局 · 並轉換數據類型為float32
    padded_img = np.ascontiguousarray(padded_img, dtype=np.float32)

    # 返回預處理後的圖片和縮放比例
    return padded_img, r
```

## Data pre-process

2023-11-10 10:40:07 | INFO | yolox.core.trainer:131 Exp

keys	values
seed	None
output_dir	'./YOLOX_outputs'
print_interval	10
eval_interval	1
dataset	None
num_classes	1
depth	0.33
width	0.5
act	'silu'
data_num_workers	4
input_size	(704, 704)
multiscale_range	5
data_dir	'datasets/HW2_ObjectDetection_2023'
train_ann	'train_labels.json'
val_ann	'val_labels.json'
test_ann	'instances_test2017.json'
mosaic_prob	1.0
mixup_prob	1.0
hsv_prob	1.0
flip_prob	0.5
degrees	10.0
translate	0.1
mosaic_scale	(0.5, 2)

enable_mixup	True
mixup_scale	(0.5, 1.5)
shear	2.0
warmup_epochs	5
max_epoch	350
warmup_lr	0
min_lr_ratio	0.05
basic_lr_per_img	0.00015625
scheduler	'yoloxwarmcos'
no_aug_epochs	15
ema	True
weight_decay	0.0005
momentum	0.9
save_history_ckpt	False
exp_name	'yolox_s'
test_size	(704, 704)
test_conf	0.01
nmsthre	0.65

Hyperparameters

- Explain which layer you add SE modules to and compare the corresponding results

```
class CSPlayer(nn.Module):
    """C3 in yolov5, CSP Bottleneck with 3 convolutions"""

    def __init__(
        self,
        in_channels,
        out_channels,
        n=1,
        shortcut=True,
        expansion=0.5,
        depthwise=False,
        act="silu",
    ):
        """
        Args:
            in_channels (int): input channels.
            out_channels (int): output channels.
            n (int): number of Bottlenecks. Default value: 1.
        """
        # ch_in, ch_out, number, shortcut, groups, expansion
        super().__init__()
        hidden_channels = int(out_channels * expansion)  # hidden channels
        self.conv1 = BaseConv(in_channels, hidden_channels, 1, stride=1, act=act)
        self.conv2 = BaseConv(in_channels, hidden_channels, 1, stride=1, act=act)
        self.conv3 = BaseConv(2 * hidden_channels, out_channels, 1, stride=1, act=act)
        self.se = SELayer(2 * hidden_channels)
        module_list = [
            Bottleneck(
                hidden_channels, hidden_channels, shortcut, 1.0, depthwise, act=act
            )
            for _ in range(n)
        ]
        self.m = nn.Sequential(*module_list)

    def forward(self, x):
        x_1 = self.conv1(x)
        x_2 = self.conv2(x)
        x_1 = self.m(x_1)
        x = torch.cat((x_1, x_2), dim=1)
        x = self.se(x)
        return self.conv3(x)
```

修改 YOLOX backbone 中 CSPDarknet 的 CspLayer，將 SE module 加入到其中

```
(dark2): Sequential(
  (0): BaseConv(
    (conv): Conv2d(32, 64, kernel_size=(1, 1), stride=(2, 2), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (1): CSPlayer(
    (conv1): BaseConv(
      (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
      (act): SiLU(inplace=True)
    )
    (conv2): BaseConv(
      (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
      (act): SiLU(inplace=True)
    )
    (conv3): BaseConv(
      (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
      (act): SiLU(inplace=True)
    )
    (se): Sequential(
      (0): Bottleneck(
        (conv1): BaseConv(
          (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
          (act): SiLU(inplace=True)
        )
        (conv2): BaseConv(
          (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
          (act): SiLU(inplace=True)
        )
      )
    )
  )
)
```

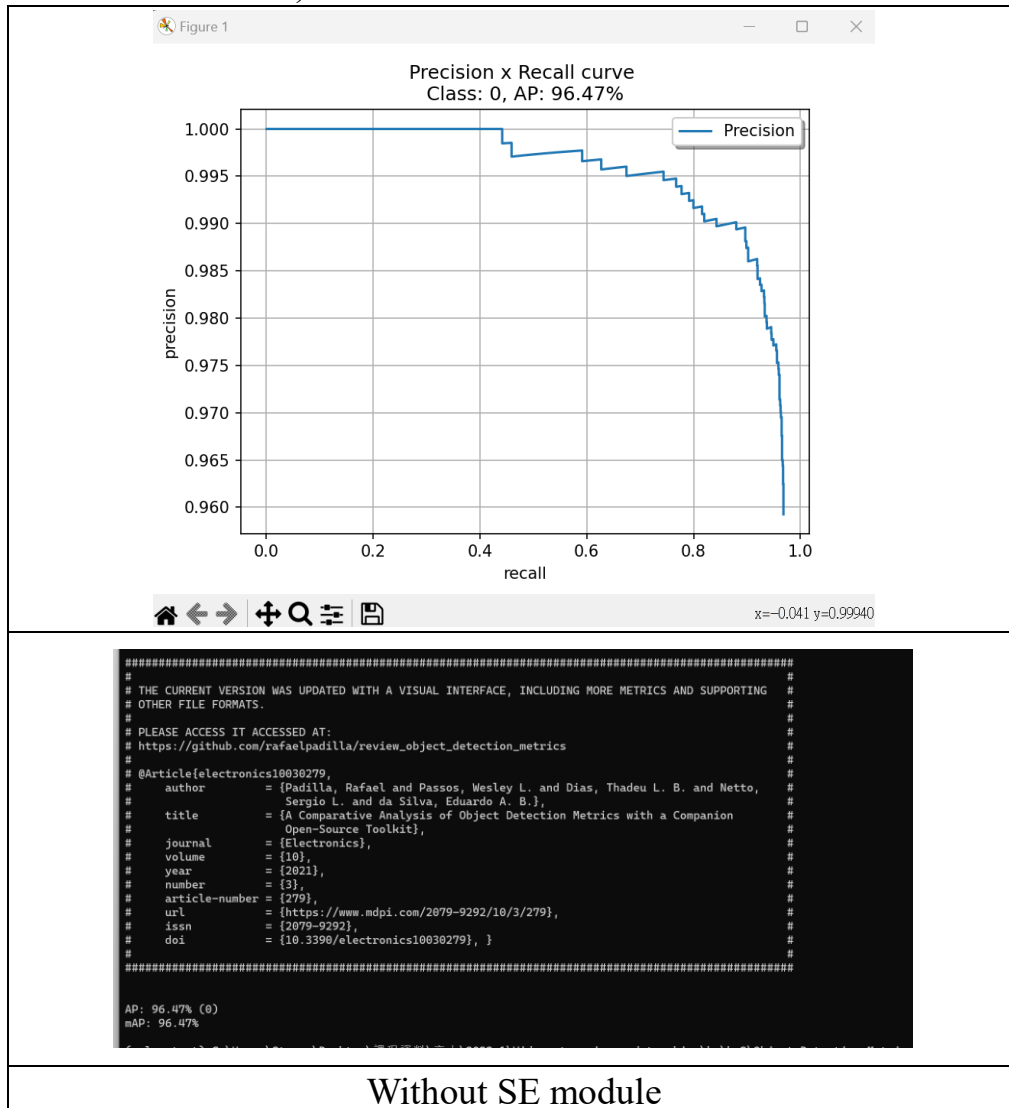
```
(dark2): Sequential(
  (0): BaseConv(
    (conv): Conv2d(32, 64, kernel_size=(1, 1), stride=(2, 2), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (1): CSPlayer(
    (conv1): BaseConv(
      (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
      (act): SiLU(inplace=True)
    )
    (conv2): BaseConv(
      (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
      (act): SiLU(inplace=True)
    )
    (conv3): BaseConv(
      (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
      (act): SiLU(inplace=True)
    )
    (se): SELayer(
      (avg_pool1): AdaptiveAvgPool2d(output_size=1)
      (fc): Sequential(
        (0): Linear(in_features=64, out_features=4, bias=True)
        (1): ReLU(inplace=True)
        (2): Linear(in_features=4, out_features=64, bias=True)
        (3): Sigmoid()
      )
    )
  )
  (2): BaseConv(
    (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
)
```

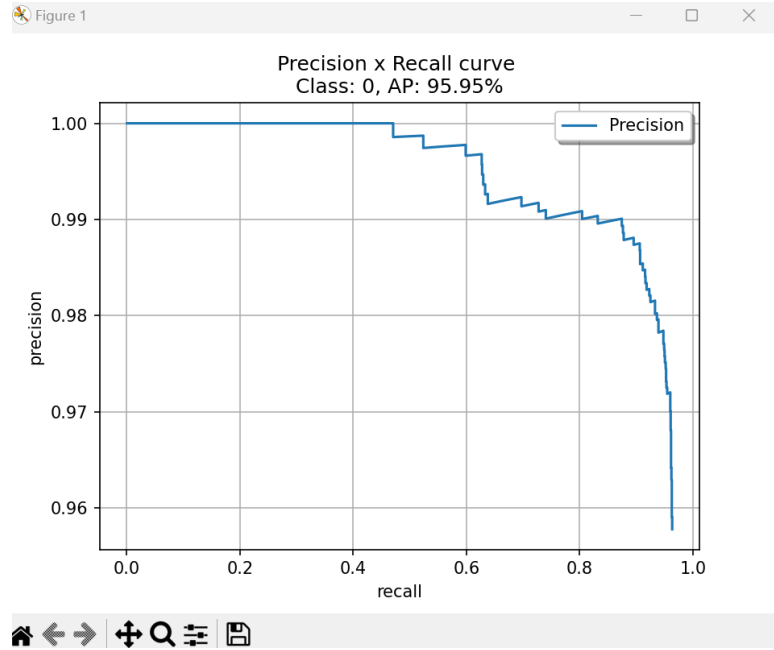
左圖為未加入與右圖加入 SE module 結構比較

結果討論：

這次在添加 SE Module 後的結果看起來不但沒有提升，跟原本的表現相比，還差了一點，我認為可能的原因可能是因為模型已經足夠複雜，已經能充份的學習到這些訊息，且因為這次提供的資料及較小，因此對於添加 SE Module 並不會有太大的改善。

- Screenshot your validation results on your two models (with / without SE module)





```
#####
#
# THE CURRENT VERSION WAS UPDATED WITH A VISUAL INTERFACE, INCLUDING MORE METRICS AND SUPPORTING
# OTHER FILE FORMATS.
#
# PLEASE ACCESS IT ACCESSED AT:
# https://github.com/rafaelpadilla/review_object_detection_metrics
#
# @Article{electronics10030279,
#   author      = {Padilla, Rafael and Passos, Wesley L. and Dias, Thadeu L. B. and Netto,
#                 Sergio L. and da Silva, Eduardo A. B.},
#   title       = {A Comparative Analysis of Object Detection Metrics with a Companion
#                 Open-Source Toolkit},
#   journal     = {Electronics},
#   volume      = {10},
#   year        = {2021},
#   number      = {3},
#   article-number = {279},
#   url         = {https://www.mdpi.com/2079-9292/10/3/279},
#   issn        = {2079-9292},
#   doi         = {10.3390/electronics10030279}, }
#
#####

AP: 95.95% (0)
mAP: 95.95%
```

With SE module

## ● Installation

```
(yolox_test) C:\Users\Steven>cd "C:\Users\Steven\Desktop\課程資料\交大\2023-1\Video streaming and tracking\hw\hw2\Original"
```

移動到專案資料夾(Original 或 SE 都可以)



```
(yolox_test) C:\Users\Steven\Desktop\課程資料\交大\2023-1\Video streaming and tracking\hw\hw2\Original>pip3 install -v -e .
Using pip 23.3 from C:\Users\Steven\anaconda3\envs\yolox_test\lib\site-packages\pip (python 3.10)
Obtaining file:///C:/Users/Steven/Desktop/%E8%AA%B2%E7%A8%8B%E8%B3%87%E6%96%99/%E4%B%A4%E5%A4%A7/2023-1/Video%20streaming%20and%20tracking/hw/hw2/Original
Running command python setup.py egg_info
C:\Users\Steven\anaconda3\envs\yolox_test\lib\site-packages\setuptools\__init__.py:84: _DeprecatedInstaller: setuptools.installer and fetch_build_eggs are deprecated.
!!

*****
Requirements should be satisfied by a PEP 517 installer.
If you are using pip, you can try 'pip install --use-pep517'.
*****

!!
dist.fetch_build_eggs(dist.setup_requires)
[WARNING] Unable to import torch, pre-compiling ops will be disabled.
running egg_info
creating C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info
writing C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info\PKG-INFO
writing dependency_links to C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info\dependency_links.txt
writing requirements to C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info\requires.txt
writing top-level names to C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info\top_level.txt
writing manifest file 'C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info\SOURCES.txt'
reading manifest file 'C:\Users\Steven\AppData\Local\Temp\pip-pip-egg-info-tkum53k3\yolox.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no files found matching '*.cu' under directory 'yolox'
```

## 輸入 pip3 install -v -e . 安裝環境

```
Anaconda Prompt - conda dx
(yolox_test) C:\Users\Steven\Desktop\課程資料\交大\2023-1\Video streaming and tracking\hw\hw2\Original>conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 -c pytorch -c conda-forge
Collecting package metadata (current_repodata.json): / DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /conda-forge/win-64/current_repodata.json HTTP/1.1" 200
None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTTP/1.1" 304 0
None
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /conda-forge/noarch/current_repodata.json HTTP/1.1" 200
None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTTP/1.1" 304 0
done
Solving environment: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): / DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
```

這裡要注意的是 PyTorch 要另外安裝，不然會下載到 cpu 版本的，根據自己的環境安裝對應的 PyTorch，以我的例子來說透過 `conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 -c pytorch -c conda-forge` 指令下載。

**Quick Start**

▼ Installation

Step1. Install YOLOX from source.

```
git clone git@github.com:Megvii-BaseDetection/YOLOX.git
cd YOLOX
pip3 install -v -e . # or python3 setup.py develop
```

安裝流程也可以參考 [YOLOX](#) 官方提供步驟

注意: 也可以透過 `env.yml` 建立環境，但需要自行下載 PyTorch

- 
- Instruction

```
pillow
done

(yolox_test) C:\Users\Steven\Desktop\課程資料\英文\2023-1\Video streaming and tracking\hw2\Original>python -m yolox.tools.demo image -f exps/example/custom/yolox_s.py -c weight
s/best_ckpt.pth --path assets/HW2_ObjectDetection_2023/val2017 --save_result --device gpu
2023-11-15 17:18:28.288 INFO | _main_:main:278 - Model Summary: Params: 8.94M, Gflops: 32.38
2023-11-15 17:18:28.422 INFO | _main_:main:291 - Loading checkpoint
2023-11-15 17:18:28.578 INFO | _main_:main:295 - loaded checkpoint done.
2023-11-15 17:18:30.280 INFO | _main_:inference:165 - Infer time: 0.6678s
2023-11-15 17:18:35.286 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\10.txt
2023-11-15 17:18:35.443 INFO | _main_:inference:165 - Infer time: 0.1014s
2023-11-15 17:18:35.486 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\106.txt
2023-11-15 17:18:35.587 INFO | _main_:inference:165 - Infer time: 0.0992s
2023-11-15 17:18:35.589 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1010.txt
2023-11-15 17:18:35.651 INFO | _main_:inference:165 - Infer time: 0.0189s
2023-11-15 17:18:35.653 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1028.txt
2023-11-15 17:18:35.718 INFO | _main_:inference:165 - Infer time: 0.0189s
2023-11-15 17:18:35.722 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1057.txt
2023-11-15 17:18:35.778 INFO | _main_:inference:165 - Infer time: 0.0182s
2023-11-15 17:18:35.772 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1061.txt
2023-11-15 17:18:35.832 INFO | _main_:inference:165 - Infer time: 0.0224s
2023-11-15 17:18:35.835 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1077.txt
2023-11-15 17:18:35.891 INFO | _main_:inference:165 - Infer time: 0.0189s
2023-11-15 17:18:35.896 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1090.txt
2023-11-15 17:18:35.953 INFO | _main_:inference:165 - Infer time: 0.0219s
2023-11-15 17:18:35.954 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1154.txt
2023-11-15 17:18:36.014 INFO | _main_:inference:165 - Infer time: 0.0189s
2023-11-15 17:18:36.016 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1162.txt
2023-11-15 17:18:36.069 INFO | _main_:inference:165 - Infer time: 0.0179s
2023-11-15 17:18:36.077 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1174.txt
2023-11-15 17:18:36.138 INFO | _main_:inference:165 - Infer time: 0.0175s
2023-11-15 17:18:36.142 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1183.txt
2023-11-15 17:18:36.192 INFO | _main_:inference:165 - Infer time: 0.0179s
2023-11-15 17:18:36.196 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1185.txt
2023-11-15 17:18:36.251 INFO | _main_:inference:165 - Infer time: 0.0166s
2023-11-15 17:18:36.255 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1201.txt
2023-11-15 17:18:36.305 INFO | _main_:inference:165 - Infer time: 0.0169s
2023-11-15 17:18:36.307 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1202.txt
2023-11-15 17:18:36.367 INFO | _main_:inference:165 - Infer time: 0.0199s
2023-11-15 17:18:36.368 INFO | _main_:image_demo:211 - Saving detection result in ./YOLOX_outputs/yolox_s\vis_res\2023_11_15_17_18_28\1221.txt
2023-11-15 17:18:36.432 INFO | _main_:inference:165 - Infer time: 0.0189s
```

透過 `python -m yolox.tools.demo image -f exps/example/custom/yolox_s.py -c weights/best_ckpt.pth --path assets/HW2_ObjectDetection_2023/val2017 --save_result --device gpu` 指令，將結果輸出到指定資料夾(請將資料集放到 `assets` 下)

```
(yolox_test) C:\Users\Steven\Desktop\課程資料\英文\2023-1\Video streaming and tracking\hw2\Original>python -m yolox.tools.train -f exps/example/custom/yolox_s.py -d 1 -b 64 --
fp16 -o
2023-11-15 17:20:12 INFO | yolox_core.trainer:130 - args: Namespace(experiment_name='yolox_s', name=None, dist_backend='nccl', dist_url=None, batch_size=64, devices=1, exp_f
ile='exps/example/custom/yolox_s.py', resume=False, ckpt=None, start_epoch=None, num_machines=1, machine_rank=0, fp16=True, caches=None, occupy=True, logger='tensorboard', opts=[])
2023-11-15 17:20:12 INFO | yolox_core.trainer:131 - exp value:


| keys             | values                                              |
|------------------|-----------------------------------------------------|
| seed             | None                                                |
| output_dir       | './YOLOX_outputs'                                   |
| print_interval   | 10                                                  |
| eval_interval    | 1                                                   |
| dataset          | None                                                |
| num_classes      | 1                                                   |
| depth            | 0.33                                                |
| width            | 0.5                                                 |
| act              | 'silu'                                              |
| data_num_workers | 4                                                   |
| input_size       | (768, 768)                                          |
| multiscale_range | 5                                                   |
| data_dir         | '/kaggle/input/gtadataset/HW2_ObjectDetection_2023' |
| train_ann        | 'train_labels.json'                                 |
| val_ann          | 'val_labels.json'                                   |
| test_ann         | 'instances_test2017.json'                           |
| mosaic_prob      | 1.0                                                 |


```

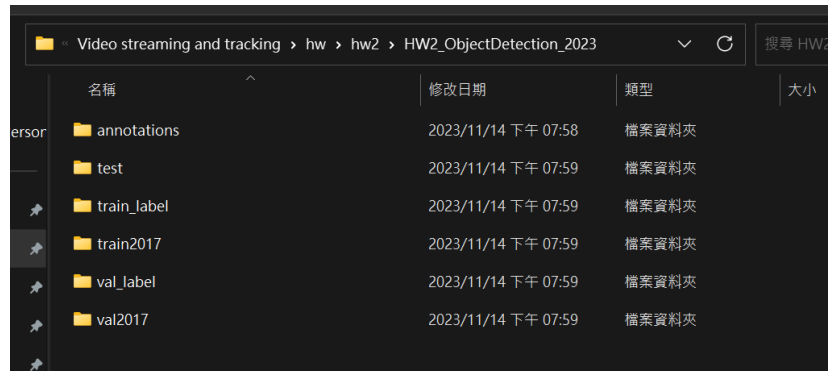
透過 `python tools/train.py -f exps/example/custom/yolox_s.py -d 1 -b 64 -fp16 -o` 指令執行模型訓練(請將資料集放到 `datasets` 下)

```
(yolox_test) C:\Users\Steven\Desktop\課程資料\英文\2023-1\Video streaming and tracking\hw2\Object-Detection-Metrics>python pascalvoc.py -gt ../val_label -det SE -t 0.85 -gtformat xywh -detfo
rmat xyrb -gtcoords rel -detcoords abs -imgsize "(1920,1080)" -sp results

#####
# THE CURRENT VERSION WAS UPDATED WITH A VISUAL INTERFACE, INCLUDING MORE METRICS AND SUPPORTING #
# OTHER FILE FORMATS. #
# PLEASE ACCESS IT ACCESSED AT: #
# https://github.com/rafaelpadilla/review_object_detection_metrics #
# @Article{electronics10030279, #
#   author = {Padilla, Rafael and Passos, Wesley L. and Dias, Thadeu L. B. and Netto, #
#   title = {A Comparative Analysis of Object Detection Metrics with a Companion #
#   journal = {Electronics}, #
#   volume = {10}, #
#   year = {2021}, #
#   number = {23}, #
#   article-number = {279}, #
#   url = {https://www.mdpi.com/2079-9292/10/2/279}, #
#   issn = {2079-9292}, #
#   doi = {10.3390/electronics10030279}, } #
#####

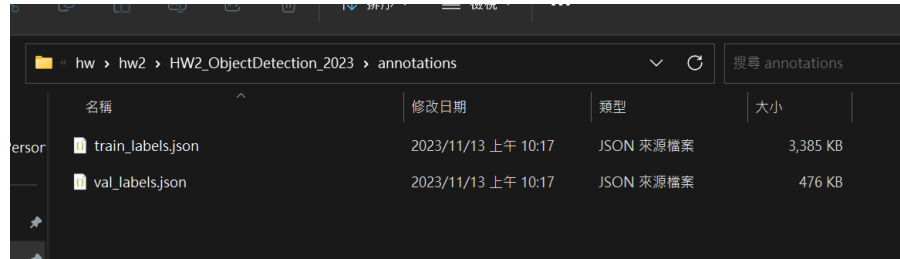
AP: 95.95% (6)
mAP: 95.95%
```

透過 `python pascalvoc.py -gt ../val_label -det SE -t 0.85 -gtformat xywh -detformat xyrb -gtcoords rel -detcoords abs -imgsize "(1920,1080)" -sp results` 指令計算 mAP



Video streaming and tracking > hw > hw2 > HW2\_ObjectDetection\_2023

名稱	修改日期	類型	大小
annotations	2023/11/14 下午 07:58	檔案資料夾	
test	2023/11/14 下午 07:59	檔案資料夾	
train_label	2023/11/14 下午 07:59	檔案資料夾	
train2017	2023/11/14 下午 07:59	檔案資料夾	
val_label	2023/11/14 下午 07:59	檔案資料夾	
val2017	2023/11/14 下午 07:59	檔案資料夾	



hw > hw2 > HW2\_ObjectDetection\_2023 > annotations

名稱	修改日期	類型	大小
train_labels.json	2023/11/13 上午 10:17	JSON 來源檔案	3,385 KB
val_labels.json	2023/11/13 上午 10:17	JSON 來源檔案	476 KB

資料集目錄結構

## ● Reference

[Megvii-BaseDetection/YOLOX: YOLOX is a high-performance anchor-free YOLO, exceeding yolov3~v5 with MegEngine, ONNX, TensorRT, ncnn, and OpenVINO supported. Documentation: https://yolox.readthedocs.io/ \(github.com\)](https://yolox.readthedocs.io/)