一、 Experiment Setup

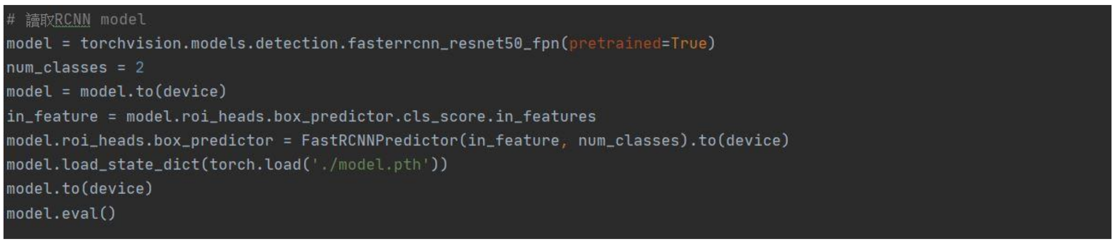
1. Details of model:
   - ➢ Faster RCNN.
2. Pretrain:
   - ➢ Imagenet
3. Training:
   - ➢ Wilder face

二、 Screenshot and explain code(train.py, trainSE.py)

1. Train model: train.py

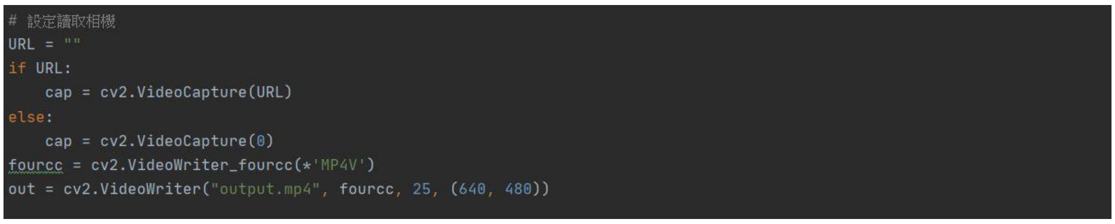
2. Tracking on laptop camera or webcam: track_on_laptop.py

   (1) Load RCNN model:



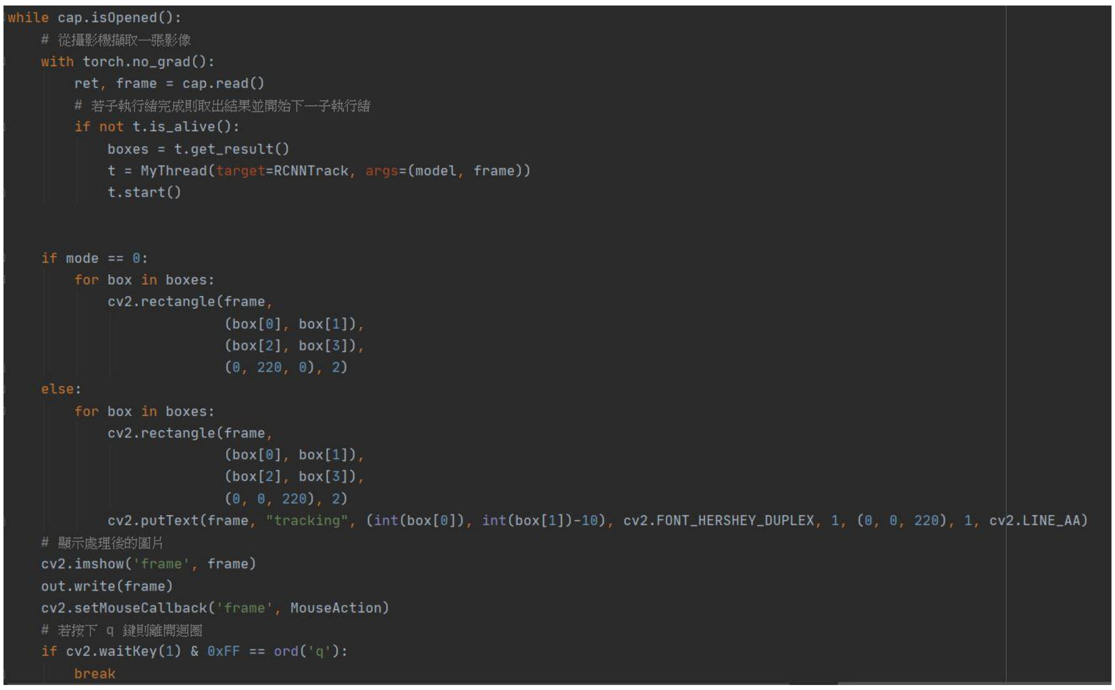
```python
# 讀取RCNN model
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
num_classes = 2
model = model.to(device)
in_feature = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_feature, num_classes).to(device)
model.load_state_dict(torch.load('./model.pth'))
model.to(device)
model.eval()
```

   (2) Load camera:

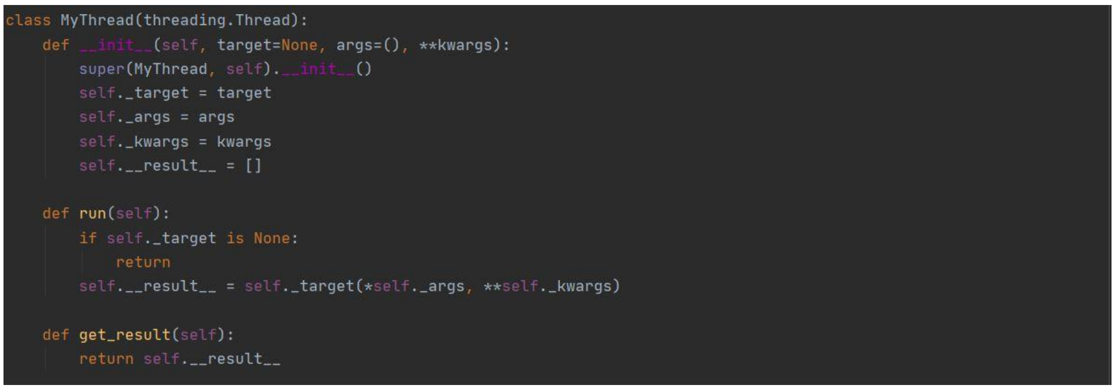


```python
# 設定讀取相機
URL = ""
if URL:
    cap = cv2.VideoCapture(URL)
else:
    cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
out = cv2.VideoWriter("output.mp4", fourcc, 25, (640, 480))
```

   (3) Tracking main function:

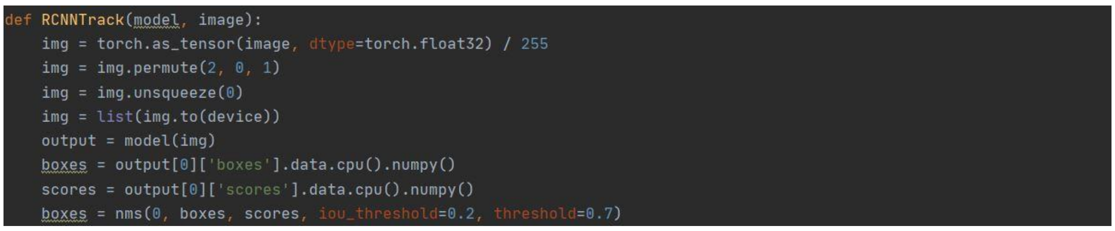

```python
while cap.isOpened():
    # 從攝影機擷取一張影像
    with torch.no_grad():
        ret, frame = cap.read()
        # 若子執行緒完成則取出結果並開始下一子執行緒
        if not t.is_alive():
            boxes = t.get_result()
            t = MyThread(target=RCNNTrack, args=(model, frame))
            t.start()


    if mode == 0:
        for box in boxes:
            cv2.rectangle(frame,
                          (box[0], box[1]),
                          (box[2], box[3]),
                          (0, 220, 0), 2)
    else:
        for box in boxes:
            cv2.rectangle(frame,
                          (box[0], box[1]),
                          (box[2], box[3]),
                          (0, 0, 220), 2)
            cv2.putText(frame, "tracking", (int(box[0]), int(box[1])-10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 220), 1, cv2.LINE_AA)
    # 顯示處理後的圖片
    cv2.imshow('frame', frame)
    out.write(frame)
    cv2.setMouseCallback('frame', MouseAction)
    # 若按下 q 鍵則離開迴圈
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

(4) Construct multithreading:

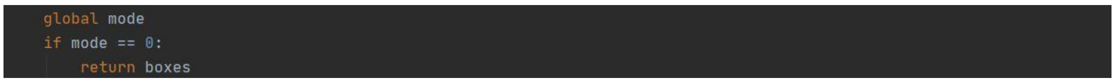

```python
class MyThread(threading.Thread):
    def __init__(self, target=None, args=(), **kwargs):
        super(MyThread, self).__init__()
        self._target = target
        self._args = args
        self._kwargs = kwargs
        self.__result__ = []

    def run(self):
        if self._target is None:
            return
        self.__result__ = self._target(*self._args, **self._kwargs)

    def get_result(self):
        return self.__result__
```

(5) RCNN tracking function: RCNNTrack(model, image)

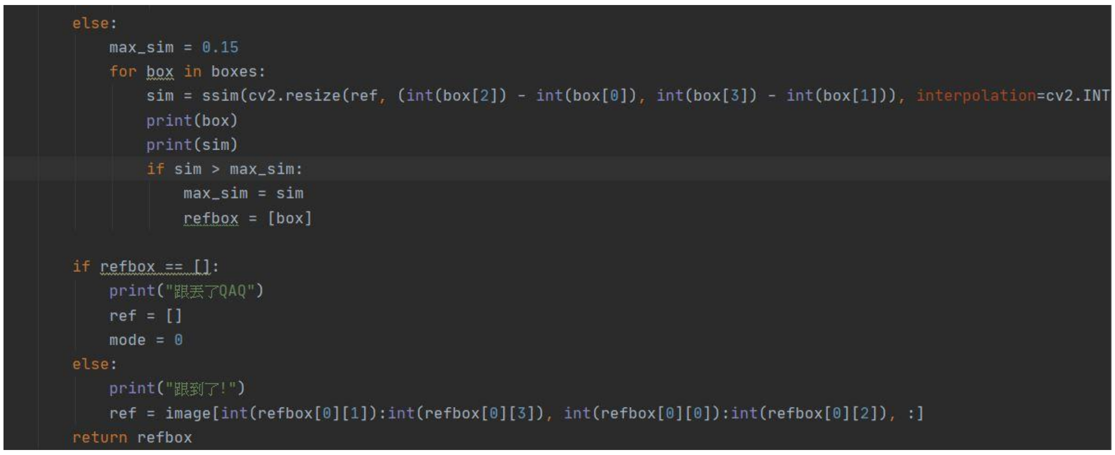
i. Get the image and compute predict bounding boxex:


```python
def RCNNTrack(model, image):
    img = torch.as_tensor(image, dtype=torch.float32) / 255
    img = img.permute(2, 0, 1)
    img = img.unsqueeze(0)
    img = list(img.to(device))
    output = model(img)
    boxes = output[0]['boxes'].data.cpu().numpy()
    scores = output[0]['scores'].data.cpu().numpy()
    boxes = nms(0, boxes, scores, iou_threshold=0.2, threshold=0.7)
```

ii. If not tracking mode (mode = 0) > output all bounding boxes:

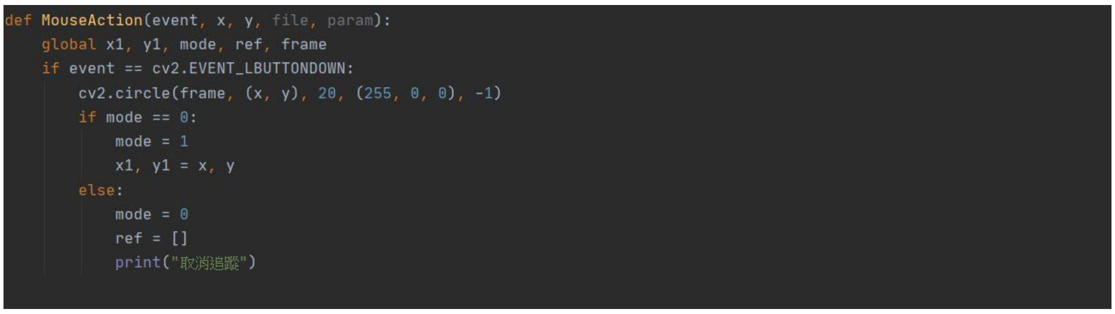

```python
    global mode
    if mode == 0:
        return boxes
```

iii. If tracking mode (mode = 1) > output the bounding box which has maximum similarity with ref (ssim):



```
        else:
            max_sim = 0.15
            for box in boxes:
                sim = ssim(cv2.resize(ref, (int(box[2]) - int(box[0]), int(box[3]) - int(box[1])), interpolation=cv2.INT
                print(box)
                print(sim)
                if sim > max_sim:
                    max_sim = sim
                    refbox = [box]

        if refbox == []:
            print("跟丟了QAQ")
            ref = []
            mode = 0
        else:
            print("跟到了!")
            ref = image[int(refbox[0][1]):int(refbox[0][3]), int(refbox[0][0]):int(refbox[0][2]), :]
        return refbox
```

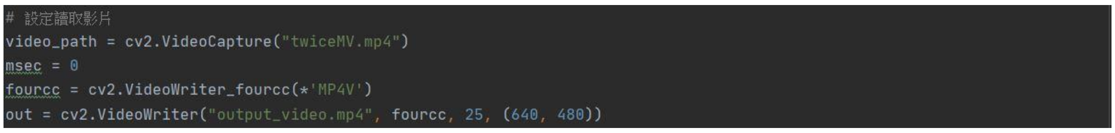
(6) Mouse callback function: MouseAction(event, x, y, file, param)



```
def MouseAction(event, x, y, file, param):
    global x1, y1, mode, ref, frame
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(frame, (x, y), 20, (255, 0, 0), -1)
        if mode == 0:
            mode = 1
            x1, y1 = x, y
        else:
            mode = 0
            ref = []
            print("取消追蹤")
```

3. Tracking on video: track_on_video.py

(1) Set video path:



```
# 設定讀取影片
video_path = cv2.VideoCapture("twiceMV.mp4")
msec = 0
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
out = cv2.VideoWriter("output_video.mp4", fourcc, 25, (640, 480))
```

三、 Result:

1. Track on camera: output.mp4
2. Track on video: output_video.mp4

四、 Discussion:

1. Why multithreading:
   因為 RCNN 跑太慢了(我的筆電問題)，為了讓 camera 畫面不卡頓，讓 RCNN 在子執行緒上跑，跑完再更新 bounding boxes，而影片則持續更新，結果是 bounding boxes 有點 lag。