Video streaming and tracking HW4

組別:14

組員:

李孟霏 311553003

陳泰元 310551076

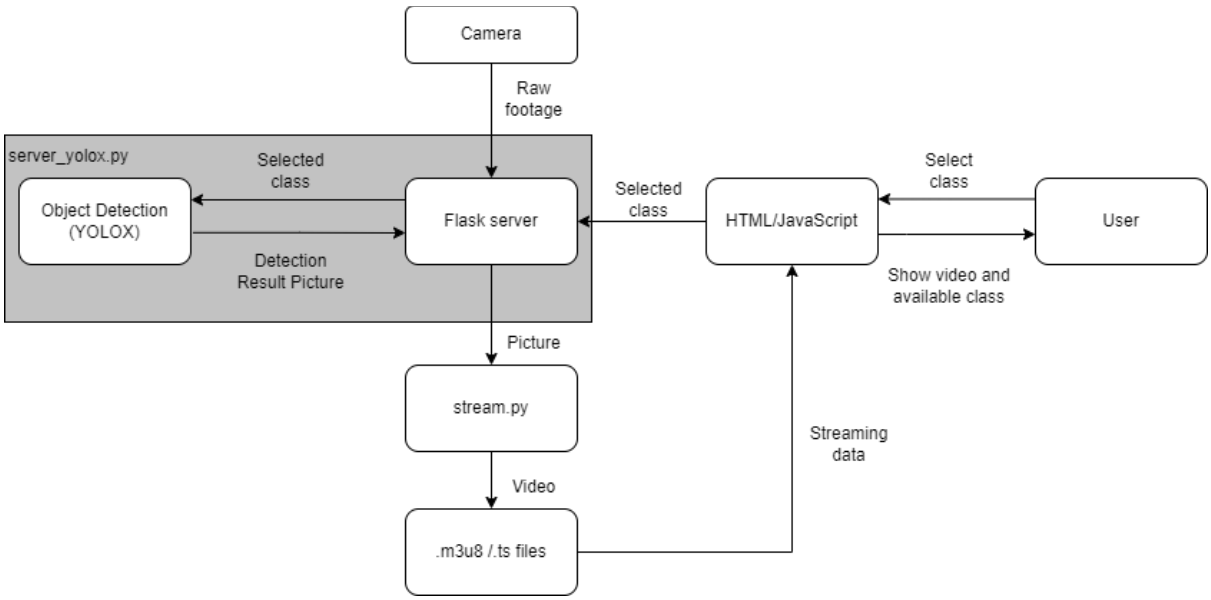
朱柏綸 3105511179

(這份report僅僅只是為了加快QA流程，無需做任何美工)

環境:
Windows
Python 3.8.15
PyTorch 1.7+

Streaming Server 架設:

(簡單寫、也可以附流程圖或code)



http_server.py:架設靜態http server，顯示介面

sever_yolox.py:架設flask server，處理request(video_feed、get_cls)。

將從camera讀進的圖片經由yolox-s model產生detection後的圖片以及目前圖片上所有的class。

```
while True:
    _,img0=cam.read()

    if frame_id %2 > -1:
        img, _, _, _ = letterbox(img0, width=w, height=h)
        outputs, img_info = predictor.inference(img)
        output = outputs[0].cpu()
        bboxes = output[:, 0:4] / img_info["ratio"]
        cls = output[:, 6]
        scores = output[:, 4] * output[:, 5]
        # cls:是此frame所含的object class
        online_im, detect_cls_list = vis(img, bboxes, scores, cls, 0.75, COCO_CLASSES, select_cls_list)
        public_cls = detect_cls_list.copy()
        ret, jpeg = cv2 .imencode('.jpg', online_im)
        frame =  jpeg.tobytes()

    frame_id += 1
```

browser會針對不同 content-type header做反應，而在此使用header content type為`multipart/x-mixed-replace`，`x-mixed-replace`為server利用http推送串流的技術其一，將圖片作為資料區塊傳送，而更新的要點是利用`multipart`回應，在header指名`boundary=frame`的分割名稱。

```
@app.route('/video_feed')
def video_feed():
    # 回傳影片資訊
    return Response(eval_seq(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

接著每一個資料區塊用`--frame`作為區塊分割的標記，content type `image/jpeg`表示為jpeg資料，接著是data length和data本身。(ref)

```
yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n'
b'Content-Length: ' + f'{len(frame)}'.encode() + b'\r\n'
b'\r\n' + frame + b'\r\n')
```

stream.py:向flask server request 得來detection result，使用ffmpeg segementer 依照HLS(HTTP-based adaptive bitrate streaming communications protocol)，指定bit rate，產生一個play list file(.m3u8 index file)、多個segment files(.ts)

```
# Opening a Resource
video = ffmpeg_streaming.input(f'http://localhost:{config["flask_port"]}/video_feed')
#Specify the value of kilo bite rate and size for each stream explicitly.
_360p  = Representation(Size(640, 360), Bitrate(276*1024, 128*1024))
_720p  = Representation(Size(1280, 720), Bitrate(2048*1024, 320*1024))
```

```
#It creates a playlist file, and one or more segment files automatically.
#The output filename specifies the playlist filename.
####
hls = video.hls(Formats.h264(video='libx264', audio='aac', **codec_options), hls_time=1)

hls.representations(*res) #create according to specified resolution

hls.output(os.path.join(hls_dir, 'hls.m3u8')) #output to specified local folder
###
```

index.html:呈現頁面，包括影片、play/pause button、選擇畫質的下拉式選單、供client選擇追蹤class的button。使用hls.js(library幫助瀏覽器播放 HLS 協議的影片)綁定html video tag element播放影片。

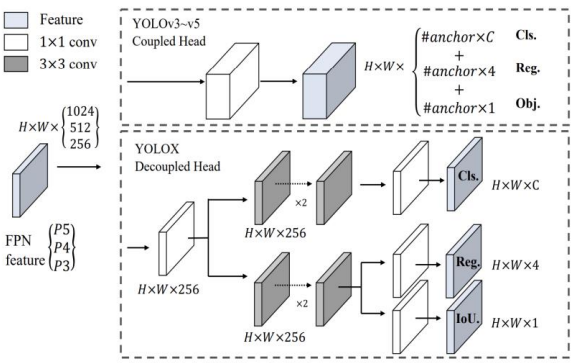
```
var videoTag = document.getElementById('video');
var videoSrc = 'hls/hls.m3u8';
var hls = new Hls();//create Hls object
```

```
//check browser if HLS.js is supported
if (Hls.isSupported()) {
  hls.loadSource(videoSrc);//load input video source(.m3u8)
  hls.attachMedia(videoTag);//bind with html video element
  //listen on event hls.on(event name,call back function)
  hls.on(Hls.Events.MANIFEST_LOADED, createDropdownMenu);
}//check for native browser HLS support
else if (videoTag.canPlayType('application/vnd.apple.mpegurl')) {
  videoTag.src = videoSrc;
}
```

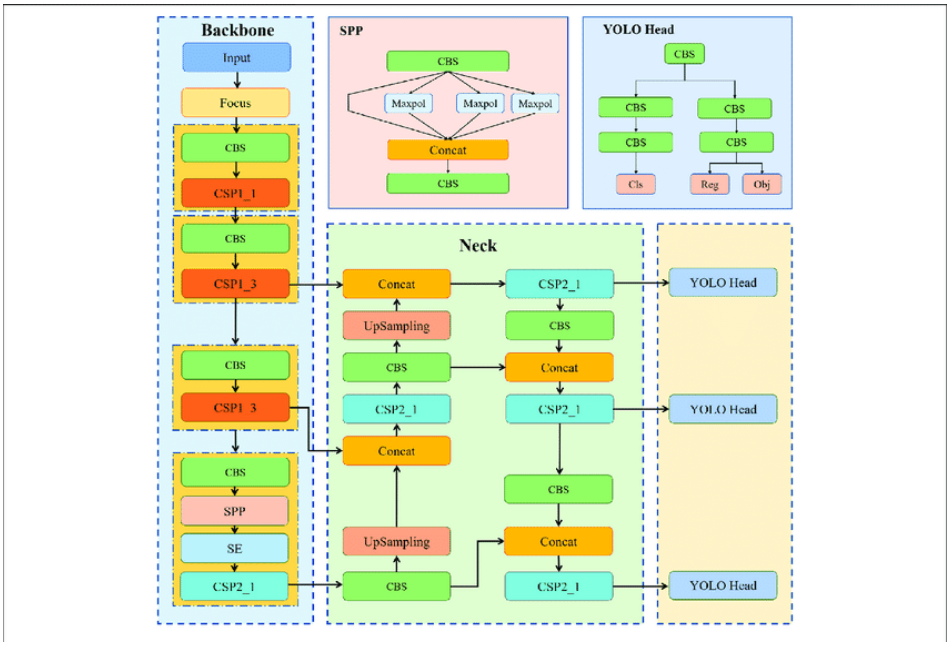
index.js:監聽html上發生事件進行處理，或是一定時間發送請求給server更新顯示的value。

DNN架構:YOLOX-s

github:https://github.com/Megvii-BaseDetection/YOLOX



Decouple head of YOLOX



YOLOX-s model architecture

額外功能&How:

選定class顯示:

- 在index.js每一段時間向server發送request，server回傳當前detector所有偵測到的class，在右側avaliable處以button形式顯示所有avaliable class

```
// Listen
function getNowObjectCls(){
    console.log("click class-select-menu");
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open('GET', `http://${server_ip}:${flask_port}/get_cls`, true);
    xmlHttp.send(null);
    xmlHttp.onload = function () {
        var buttonGroup = document.getElementById('class-select-button-group');
        var buttonGroupSelected = document.getElementById('class-select-button-group-selected');
        buttonGroup.replaceChildren()
        JSON.parse(xmlHttp.responseText).target.forEach( function (cls){
            if (!selected_element.includes(cls)){
                var node = document.createElement('button');
                node.className = `${cls}`
                node.innerHTML = `${cls}`;
                node.id=`${cls}`;
                node.onclick = selectTrackObject;
                node.style.backgroundColor = "#4CAF50";
                buttonGroup.appendChild(node);
            }
        });
    }
}
setInterval(getNowObjectCls, 800)
```

```
public_cls = list() #裝目前偵測到的cls類別
select_cls_list = list() #裝目前選定的cls類別
```

```
@app.route('/get_cls')
def get_cls():
    global public_cls
    # public_cls= ["1","2"]
    return json.dumps({'success':True, 'target':public_cls}), 200, {'ContentType': 'appliccation/json'}
```

- user可以來點選在avaliable的按鍵選擇顯示某個class或點選在selected的按鍵取消顯示某個class
- 在接收到Click event 的時候index.js向server發送request傳送選擇的class以及執行的動作，GET到回傳值後，

  select:從avalialbe刪除對應class的button，並在selected新生成對應class button

  deselect:從selected刪除對應class的button

```
let selected_element = [];
```

```
function selectTrackObject(event){
    console.log("click object from buttonGroup list");
    const select_button_value = event.target.className;
    console.log("sev:"+select_button_value);
    var xmlHttp = new XMLHttpRequest();
   xmlHttp.open(
        'GET',
        `http://${server_ip}:${flask_port}/control_cls?select_cls=` + select_button_value + '&' +  "act=select",
        true
    );

    xmlHttp.onload = function () {
        document.getElementById(select_button_value).remove();

        var buttonGroupSelected = document.getElementById('class-select-button-group-selected');
        var node = document.createElement('button');
        node.className = `${select_button_value}`
        node.innerHTML = `${select_button_value}`;
        node.id=`${select_button_value}`;
        node.onclick = deselectTrackObject;
        node.style.backgroundColor = "#b350af";
        buttonGroupSelected.appendChild(node);
        selected_element.push(select_button_value)
    }
    xmlHttp.send(null);

}
```

```
function deselectTrackObject(event){
    console.log("click object from selected buttonGroup list");
    const select_button_value = event.target.className;
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open(
        'GET',
        `http://${server_ip}:${flask_port}/control_cls?select_cls=` + select_button_value + '&' +  "act=deselect",
        true
    );
    xmlHttp.send(null);
    xmlHttp.onload = function () {
        //selected_element.pop(select_button_value);

        const index = selected_element.indexOf(select_button_value);
        if (index > -1) { // only splice array when item is found
            selected_element.splice(index, 1); // 2nd parameter means remove one item only

            console.log(selected_element);
            document.getElementById(select_button_value).remove();
        }

    }
}
```

```
@app.route('/control_cls')
@cross_origin()
def control_cls():
    select_cls = request.args.get('select_cls')
    # act: select -> 選取, deselect -> 取消
    print("select_cls: ",select_cls)
    act = request.args.get('act')
    print("act: ",act)
    if act == "select":

        cls_id = COCO_CLASSES.index(select_cls)
        print("select_cls_id ",cls_id)
        select_cls_list.append(cls_id)
        print("select_cls_list: ",select_cls_list)
    else:

        cls_id = COCO_CLASSES.index(select_cls)
        print("diselect_cls_id ",cls_id)
        if(select_cls_list and cls_id in select_cls_list):
            select_cls_list.remove(cls_id)
        print("select_cls_list: ",select_cls_list)
    return json.dumps({'success':True, 'target':select_cls}), 200, {'ContentType': 'appliccation/json'}
```

- server在做detected bounding box visiualization的時候僅顯示選擇的class

```
# cls:是此frame所含的object class
online_im, detect_cls_list = vis(img, bboxes, scores, cls, 0.75, COCO_CLASSES, select_cls_list)
# print(f"type: {type(cls)}")
public_cls = detect_cls_list.copy()
```

```python
def vis(img, boxes, scores, cls_ids, conf=0.5, class_names=None, select_cls_list=None):
    clstext=set()
    for i in range(len(boxes)):
        box = boxes[i]
        cls_id = int(cls_ids[i])
        score = scores[i]
        if score < conf :
            continue
        else:#get all avaliable class
            clstext.add(class_names[cls_id])

        if cls_id not in select_cls_list:#only draw select class
            continue
        #draw bbox,and retrun  avaliable class
```

調整畫質

- 載入HLS manifest後，使用JavaScript從HLS物件中取得可以使用的畫質等級
- 用一個dropdown menu來選擇不同的畫質

```javascript
function setLevel(elem) {
    try {
        hls.currentLevel = parseInt(elem.getAttribute('level-idx'));
        var dropdownButton = document.getElementById('dropdown-button');
        dropdownButton.innerHTML = elem.innerHTML; // 將元素elem的內部html語法敘述直接轉移
    } catch (e) {
        console.log(e);
        console.log(`cannot load level #${elem.getAttribute('level-idx')}`);
    }
}

function createDropdownMenu() {
    var dropdown = document.getElementById('dropdown-content');
    // hls.levels 回傳有效的品質選項的陣列
    hls.levels.forEach(function (item, index) {
        var node = document.createElement('div'); //建立一個Element當作node
        node.innerHTML = `${item.height}p`; //顯示360p, 720p等
        node.setAttribute('level-idx', index); //設定每個選項的leve idx
        node.setAttribute('onclick', 'setLevel(this)'); //設定每個選項按下去就呼叫setLevel,並以自己這個物件當作參數
        dropdown.appendChild(node); //增加子節點
    });
    console.log('Dropdown menu created');
}
```

play/pause

- 利用html5 video tag有的function控制play/pause

```
// Play/Pause BEGIN
const playButton = document.getElementById('play');
function togglePlay() {
    console.log("toggle Play")
    if (video.paused || video.ended) {
        video.play();
    }
    else {
        video.pause();
    }
}
playButton.addEventListener('click', togglePlay);
```

Reference:

Project Framework: https://github.com/eugene87222/NCTU-Video-Streaming-and-Tracking

UI design:

https://freshman.tech/custom-html5-video/

https://ithelp.ithome.com.tw/articles/10220978

https://www.w3schools.com/css/css_dropdowns.asp

HLS streaming:

https://video.aminyazdanpanah.com/python/start

https://ithelp.ithome.com.tw/articles/10203792

HLS.js:

https://ithelp.ithome.com.tw/articles/10206181

https://ithelp.ithome.com.tw/articles/10310697

Flask server: https://towardsdatascience.com/video-streaming-in-web-browsers-with-opencv-flask-93a38846fe00

https://shengyu7697.github.io/python-flask-camera-streaming/