Yolov7介紹與使用

MPD實驗室

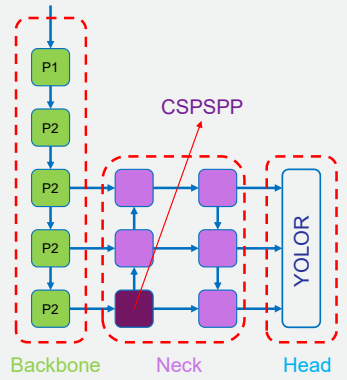
Yolov7 架構簡介

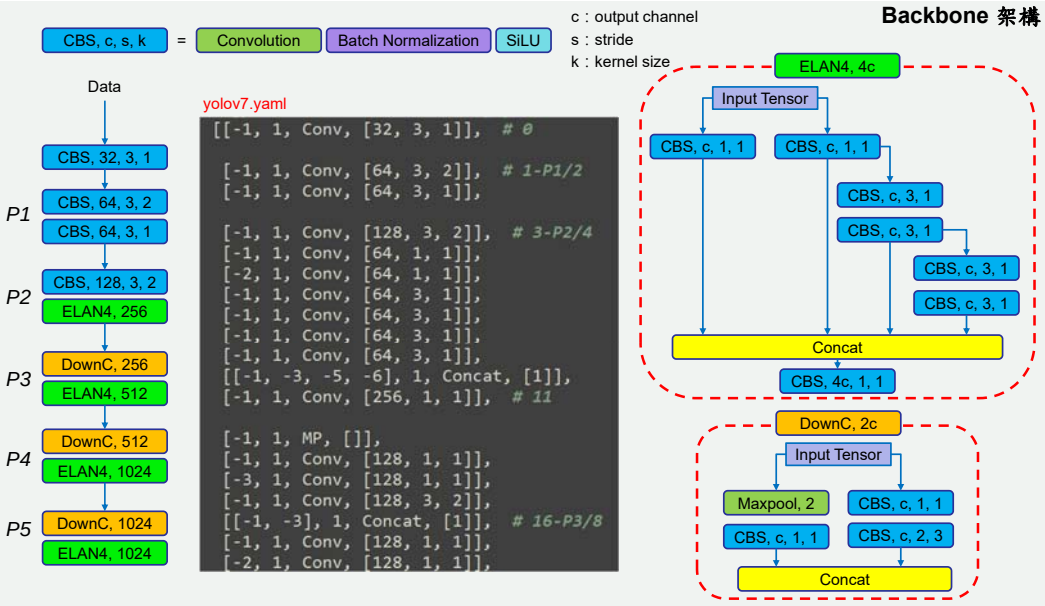
1. 在一個深度學習網路架構中，會由三個部分所組成：Backbone、Neck、Head

(1)Backbone：為整個網路架構中的骨幹，也就是進行提取特徵的部分，獲取輸入圖片的訊息，在Yolov7中使用的是ELAN與E-ELAN

(2)Neck：neck介於backbone與head之間，主要在融合不同尺度的特徵，以便更好完成在head的任務，在Yolov7中使用的是CSPSPP+(ELAN, E-ELAN)PAN

(3)Head：head為整個網路中輸出的部分，head會根據前面提取的特徵進行預測，在Yolov7中使用的是YOLOR





執行環境 – Jetson Nano

1. Jetson Nano為NVIDEA公司所開發的小型電腦，內部的裝置專門為了人工智慧系統而設計，70 X 45 mm的大小能在嵌入式物聯網中能有很好的表現，模組中搭載了四核心的ARM Cortex處理器，GPU採用了128核心NVIDIA Maxwell架構的GPU，可同步處理數個感應器

Jetson Nano 規格	
GPU	128 核心 NVIDIA Maxwell™ 架構 GPU
CPU	四核心 ARM® Cortex®-A57 MPCore 處理器
記憶體	4GB 64 位元 LPDDR4
儲存空間	microSD
影片編碼器	1x 4K30 | 2x 1080p60 | 4x 1080p30 | 9x 720p30 (H.264/H.265)
影片解碼器	1x 4K60 | 2x 4K30 | 4x 1080p60 | 8x 1080p30 | 18x 720p30 (H.264/H.265)
網路	Gigabit 乙太網路、M.2 Key E
其他 I/O	40 pin 針座 (UART、SPI、I2S、I2C、PWM、GPIO) 12 pin 自動化針座 4 pin 風扇針座 4 pin POE 針座 直流電源連接器 電源、強制復原和重設按鈕
大小	100mm x 79mm x 30.21mm (高度包括載板、模組和散熱解決方案)

Yolov7 安裝流程 - Linux

開啟 terminal 並依序執行以下指令

1. $ sudo apt update (更新基本套件)
2. $ sudo apt install -y python3-pip (安裝pip)
3. $ pip3 install --upgrade pip (更新pip)
4. 下載Yolov7：$ git clone https://github.com/WongKinYiu/yolov7.git
5. 安裝YoloV7需要套件：$ cd yolov7, $ vim requirements.txt
6. 安裝必要套件：$ sudo apt install -y libfreetype6-dev
7. 安裝執行requirements.txt：$ pip3 install -r requirements.txt

模型訓練 – 訓練資料準備

1. 訓練資料集分成三個部分，分別為訓練集(training set)、驗證集(validation set)和測試集(test set)
2. 常使用的分配比率為：

   70% train, 15% val, 15% test

   80% train, 10% val, 10% test

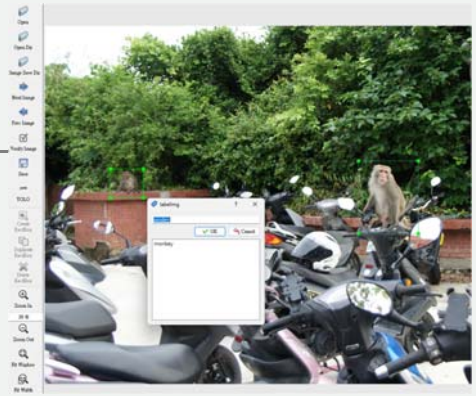
   60% train, 20% val, 20% test

   沒有一個絕對的比率，需要透過測試得到精度較高的比率

3. 訓練的資料須準備影像檔(.jpg, .png)與標記檔(.txt)，標記檔透過Labelimg來產生。

Labelimg

1. 將所有圖片都框選並標註標籤
2. 這時每一張圖片會產生1個txt，裡面會儲存圖片標籤的訊息
3. 以下圖為例，0為第0個標籤，後面為框選方框的四個角位置
4. 一張照片可以有多個標籤



模型訓練 – 設定檔

1. 在data資料夾內創建一個.yaml檔案(可從已有的檔案中複製)，開啟後修改以下內容：
   (1)訓練資料路徑(train, valid, test)
   (2)類別數量
   (3)類別名稱

```
train: ./data/monkey_train.txt
val: ./ data /monkey_valid.txt
test: ./coco/ monkey_test.txt

# number of classes
nc: 1

# class names
names: [ 'monkey' ]
```

2. 在cfg/training資料夾內同樣創建一個.yaml檔案(可從已有的檔案中複製)，開啟後修改Class數量
3. 最後在Yolov7官方Github上下載預訓練權重

```
# parameters
nc: 1  # number of classes
```

開始訓練

1. 可以透過指令進行訓練：

python3 train.py --device 0 --batch-size 8 --data .\data\mydata.yaml --img 640 --cfg .\cfg\training\yolov7_custom.yaml --weights .\yolov7.pt

指令內主要會增加幾個項目：

(1) device：使用GPU來訓練

(2) batch：每次batch學習採用多少的樣本資料

(3) img：將圖片壓縮成指定的大小後進行訓練

也可以輸入指令python3 train.py help來新增其他設定來調整訓練

Batch size

1. 在訓練模型時我們很常看到Batch、Epoch、Iteration等參數，這三項的參數彼此之間都會互相影響，而且也會影響到整體的Learning Rate，batch size的設定並沒有一定的答案，而且與訓練的電腦記憶體大小有關，需要多次的嘗試才可以知道此次訓練最好的設定

(1)Epoch：指的是訓練時一筆數據集的過程與資料狀態，假設使用了300筆的圖片做訓練，就會需要經過300次的epoch

(2)Batch：通常每一筆數據及資料量都很大，記憶體無法一次全部儲存，因此就需要將資料分批的訓練，因此Batch size就會決定每次會將幾小筆資料送進神經網路中

(3)Iteration：Iteration意思為迭代，代表總過需要進行幾次的神經網路計算，假設epoch為20，且每筆epoch內有10筆數據，batch size為2，因此總會會需要進行10/2*20=100次迭代

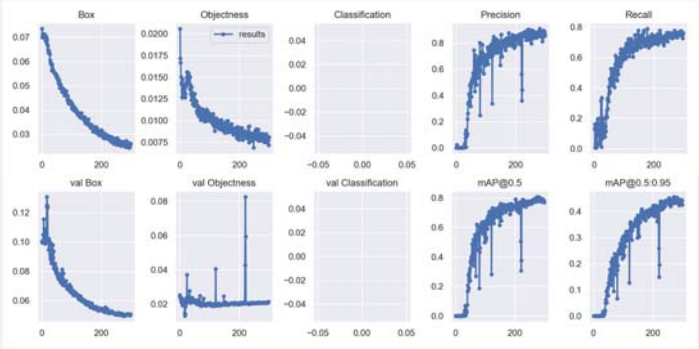
Iteration = (Data set size / Batch size) x Epoch

訓練完成

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   299/299     11.2G   0.02605  0.008077         0   0.03412         7       640: 100%|| 52/52 [06:16<00:00,  7.23s/it
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100%|| 7/7 [00:16<00:00,
                  all         100         194       0.855       0.758       0.769       0.425
300 epochs completed in 33.061 hours.

Optimizer stripped from runs\train\exp2\weights\last.pt, 142.1MB
Optimizer stripped from runs\train\exp2\weights\best.pt, 142.1MB
```

1. 訓練完成後我們就可以在runs/train資料夾內找到我們訓練好的權重檔與訓練時的一些數據



影像辨識

1. 完成訓練後就可以透過以下指令來測試訓練結果：

python3 detect.py --weights best.pt --conf 0.25 --img-size 640 --source monkey_test.jpg --view-img

(1) --weights：指定使用的權重檔

(2) --conf：設定信心程度(confidence)，若辨識出的信心程度低於此設定的數值，則不會記錄該辨識出的結果

(3) --img_size：將辨識的圖片壓縮成指定大小

(4) --source：辨識圖片路徑

(5) --view-img：顯示辨識結果

產生辨識結果

1. 在我們使用辨識完圖片後，Yolov7會自動將辨識好的圖片會預設存放在“runs/detect”資料夾內，並再自動產生一個“exp”的資料夾存放每次辨識好的圖片。

E.g.指令：**python3 detect.py --weights best.pt --conf 0.25 --source inference/images/image3.jpg**

辨識結果儲存

要將辨識出的結果儲存，我們只需要在指令中加上“--save-txt”和“--save-conf”即可

E.g.指令： **python3 detect.py --weights best.pt --conf 0.25 --save-txt --save-conf --source inference/images/image3.jpg**

detect.py – for loop

```
# Write results
for *xyxy, conf, cls in reversed(det):
```

1. det變數存放的內容為辨識出來的結果，裡面包括辨識出的xy位置、信心程度和Classes。格式為 [x1, y1, x2, y2, Confidence, Class]，其中x1 y1表示辨識框左上角的位置，x2 y2為右下角的位置
2. 因此在每一次for迴圈開始時，都會將det內的值賦予到變數xyxy, conf, cls中，因為xyxy的資料為4筆，因此使用*xyxy來儲存(x1, y1, x2, y2)。而det的排序方式是根據信心程度的大小由大到小排序，因此在for迴圈開始時使用reversed()將det的資料倒過來，改成由小到大排序

detect.py – picture’s gain

```
gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
```

1. 我們會將辨識框原本的xyxy格式(x1, y1, x2, y2) ，轉換成xywh格式(X.center, Y.center, width, high)，因此我們就必須透過圖片的大小去計算出一個Gain(增益)，並將xy軸的值去除以這個gain，才能去計算我們width和high的值
2. 我們可以再detect.py 的109行找到上面的程式，im0是我們辨識完成的圖片，首先使用函式.shape得到我們圖片的shape，內容就是這張圖片的高度和寬度以及通道(RGB)
3. 再來我們使用Pytorch的函式torch.tensor將im0.shape(list)的值轉換成tensor(張量)，並依照後面[[1, 0, 1, 0]]的格式產生式我們要的gn(gain)，也就是元素1，元素0，元素1，元素0的格式

detect.py - xywh

```
if save_txt:  # Write to file
    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normalized xywh
```

1. 了解gn是如何得到之後，我們就可以將xyxy轉換成xywh的格式了。首先同樣使用torch.tensor()將xyxy轉換成tensor，並使用Pytorch的函式.view()將xyxy變數reshape成(1, 4)的形式，也就是跟gn同樣的格式，因此xyxy就可以除以gn了
2. 出來的結果會使用xyxy2xywh函式計算出X.center, Y.center, width和high，並同樣使用.view()將結果reshape成1D的張量，.view(-1)是若我們不確定要reshape成多少，我們就可以寫成(-1)
3. 最後的結果因為要寫入txt檔中，因此還是要將數值轉換成python的list，這邊就使用了.tolist()這個函式

detect.py - write file

```
line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)  # label format
```

1. 最後我們就可以將所有的數值組合成一個字串line了，其中若我們有輸入--save-conf指令，才會將confidence寫入字串中，否則字串的數值就只會有class和位置的數值
2. 使用open開啟一個名稱為txt_path的txt檔，寫入的格式會將數值轉換成floating point(%g)，寫入的字數透過len()來取得，並使用.rstrip()將字串結尾的空白刪除，定義好格式後就可以使用 “%”將字串line套用到這個格式上了，字串的結尾會再加上換行符號

伺服端建立 – server.py

1. 我們要撰寫一個server端的程式來收取Yolov7(client端)辨識的結果，程式的內容也根據前一張投影片的架構來完成，因此主要會有以下幾個內容：

(1) 設定Port編號與Server端IP位址

(2) 建立Socket

(3) 將Ip與Port進行Bind

(4) Listen等待Client進行連線

(5) Accept Client端的連線

(6) 傳送與接收資料

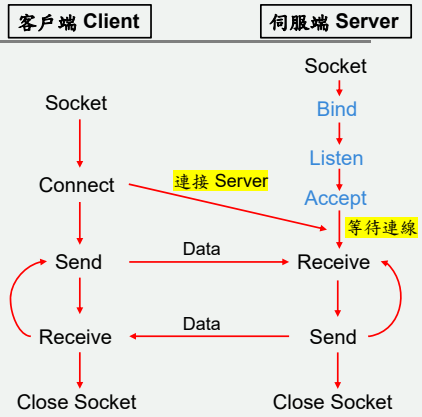
(7) 關閉連線

```
import socket
import threading
import time

SERVERIP = '127.0.0.1'
PORT = 4100
ACCEPTSIZE = 5
BUFFERSIZE = 1024
```

偵測資料傳送 – TCP/IP

1. TCP/IP為網路通訊模型，又稱IPS(Internet Protocol Suite，網際網路協定套組)，主要由應用層、傳輸層、網路層、連結層所構成的一種網路協定堆疊，與我們常聽到的OSI七層模型類似，為OSI模型的簡化版。
2. TCP協定通常有一個Server端來監聽(Listen)多個Client端，兩端皆需要建立一個Socket，並透過Bind將IP位址與通訊埠(Port)結合，組成Socket Address，因此兩端就能依據Socket Address來傳送資料，且不影響同一Port上其他的使用者。



伺服端建立 – server.py

1. 我們要撰寫一個server端的程式來收取Yolov7(client端)辨識的結果，程式的內容也根據前一張投影片的架構來完成，因此主要會有以下幾個內容：

(1) 設定Port編號與Server端IP位址

(2) 建立Socket

(3) 將Ip與Port進行Bind

(4) Listen等待Client進行連線

(5) Accept Client端的連線

(6) 傳送與接收資料

(7) 關閉連線

```
import socket
import threading
import time

SERVERIP = '127.0.0.1'
PORT = 4100
ACCEPTSIZE = 5
BUFFERSIZE = 1024
```

客戶端建立 – detect.py

1. Client端的程式我們可以直接在Yolov7中的程式進行撰寫，使程式辨識時也直接的將辨識結果傳出，Client端的程式同樣也以前面的架構完成：

(1) 設定Port編號(需要與Server端相同)與Server端IP位址

(2) 建立Socket

(3) 與Server端進行連線

(4) 傳送與接收資料

(5) 關閉連線

```
recv_data = bytes(123)
# Write results
for *xyxy, conf, cls in reversed(det):
    if save_txt:  # Write to file
        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normalized xywh
        line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)  # label format
        result_out = 'Class : ' + names[int(cls)] + ' ' + ('%g ' * len(line)).rstrip() % line + '\n'
        if recv_data.decode() != None :
            sck.send(result_out.encode())
            recv_data = sck.recv(BUFFERSIZE)
            print('recv: ' + recv_data.decode())
```

影像抓取 - FFmpeg

1. FFmpeg 為一個Open Source的軟體，可以進行音訊與視訊的錄影、轉檔、串流，壓縮，並提供了音訊與視訊的格式轉換函示庫，因此我們透過使用此軟體來開啟相機並進行錄影
2. 安裝：$ sudo apt-get install ffmpeg
3. 完成安裝後確認裝置(預設為video0)：$ ffmpeg -f v4l2 -list_formats all -i /dev/video0
4. 開始錄影與截圖：$ ffmpeg -f v4l2 -i /dev/video0 –vf fps=10 –update 1 image.png

此指令會開起攝影機後約每0.1秒(10fps)截取一次圖片，並儲存為 “image.png”， -update 會將每次截取的圖片覆蓋到 “image.png”檔案中，因此我們就可以使用Yolov7不斷讀取 “image.png”並進行辨識

建立執行環境

1. 在Yolov7資料夾內新建一個資料夾作為ffmpeg截圖後存放的位置
2. 撰寫一個.tcl檔，內容如下：

```
del image.png
ffmpeg -f v4l2 -i video=“Your Device Name” -vf fps=10 -update 1 image.png
timeout /t 5
python3 detect.py --weights best.pt --conf 0.25 --save-txt --save-conf --source ffmpeg_save/image/image.png
```

內容可以根據需求進行修改

都建置完成後就可以執行 .tcl開始進行錄影與辨識了

總結

1. 開始執行後就相機就會每0.1進行一次截圖，並且重複覆蓋“image.png”圖片，而yolov7就會重複辨識該圖片進行辨識，並且我們可以搭配前面所提到到tcp/ip資料傳送，將我們辨識出來的結果傳送到其他裝置上

感謝聆聽