

嵌入式智慧影像分析與實境界面

Fall 2021

Instructor : Yen-Lin Chen(陳彥霖), Ph.D.

Professor

Dept. Computer Science and Information Engineering

National Taipei University of Technology

Project 7

YOLOv4-tiny 道路標示辨識

安裝YOLOv4-tiny



YoloV4-tiny訓練模型

- YoloV4-tiny訓練範例，範例是以人臉為例
- 或自行下載darknet完成YoloV4 tiny模型權重訓練。
- 本次專案是以偵測交通號誌為目的，偵測包含四種道路上常見的交通號誌，讓Jetobt能依照道路上的指示來行駛。
- 建議使用運算能力較高或是Google Colab來訓練YoloV4 模型權重。



安裝執行環境

- 安裝教學參考:[NVIDIA Jetson Nano — 使用 yolov4-tiny 進行人臉偵測 | by 張銘 | Medium](#)
- 訓練教學參考:[在 Colab 上使用 yolov4-tiny 訓練一個人臉偵測模型 - 張銘 - Medium](#)
- 雲端連結提供下載:[trt_yolv4-tiny-master.zip - Google 雲端硬碟](#)
- 連結中的檔案解壓縮後，放至根目錄中。



安裝環境指令

- \$ sudo pip install protobuf==3.8.0
- \$ sudo apt-get install protobuf-compiler libprotoc-dev
- \$ sudo pip install onnx==1.4.1
- \$ sudo pip install --upgrade pip
- \$ sudo pip uninstall enum34
- 安裝pycuda部分，先修改bashrc
 - export PATH=/usr/local/cuda-10.0/bin:\${PATH}
 - export LD_LIBRARY_PATH=/usr/local/cuda-10.0/lib64:\${LD_LIBRARY_PATH}
 - 關閉terminal重新開啟或是使用source ~/.bashrc
- 在terminal輸入以下指令
 - \$ export CUDA_ROOT=/usr/local/cuda
 - \$ pip3 install pycuda --verbose

```
Text Editor
.bashrc (~) - gedit

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:lo

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo termina
(history|tail -n1|sed -e '\''s/^s*[0-9]\{+\}s*//;s/[/;|]\s*alert$/'\''')

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

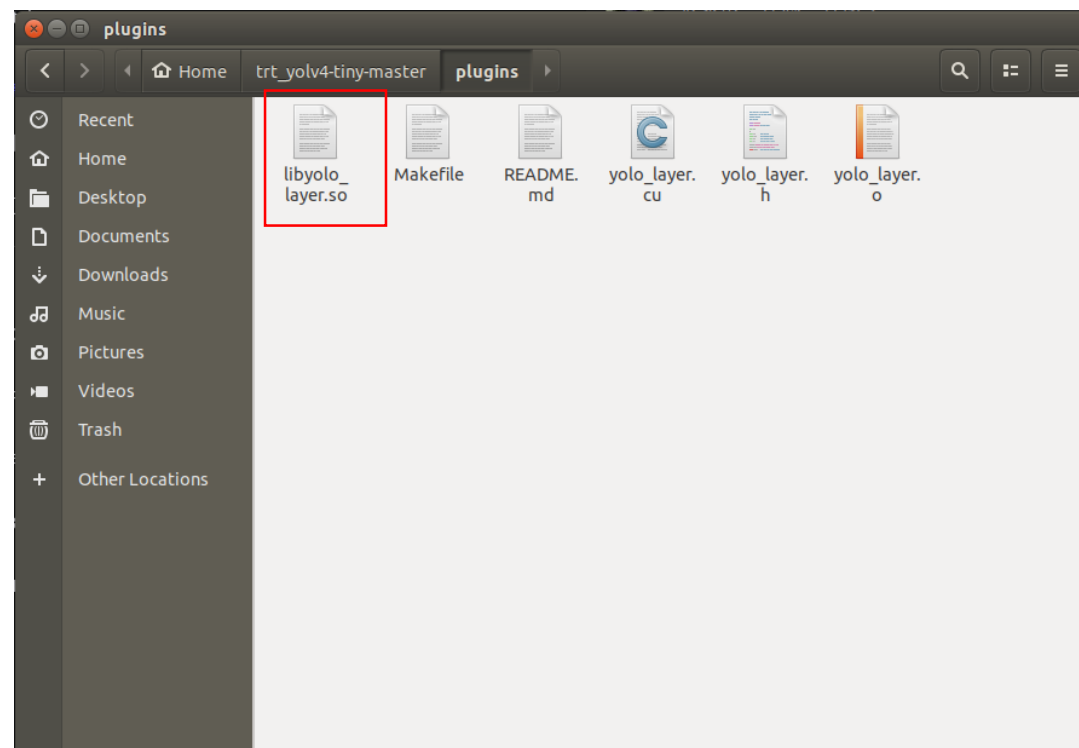
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

export PATH=/usr/local/cuda-10.0/bin:${PATH}
export LD_LIBRARY_PATH=/usr/local/cuda-10.0/lib64:${LD_LIBRARY_PATH}
```



安裝環境指令

- 進入plugins資料夾建立yolo的plugin
 - `$ cd ~/trt_yolov4-tiny/plugins`
- make 完後會得到 libyolo_layer.so 檔
 - `$ make -j4`





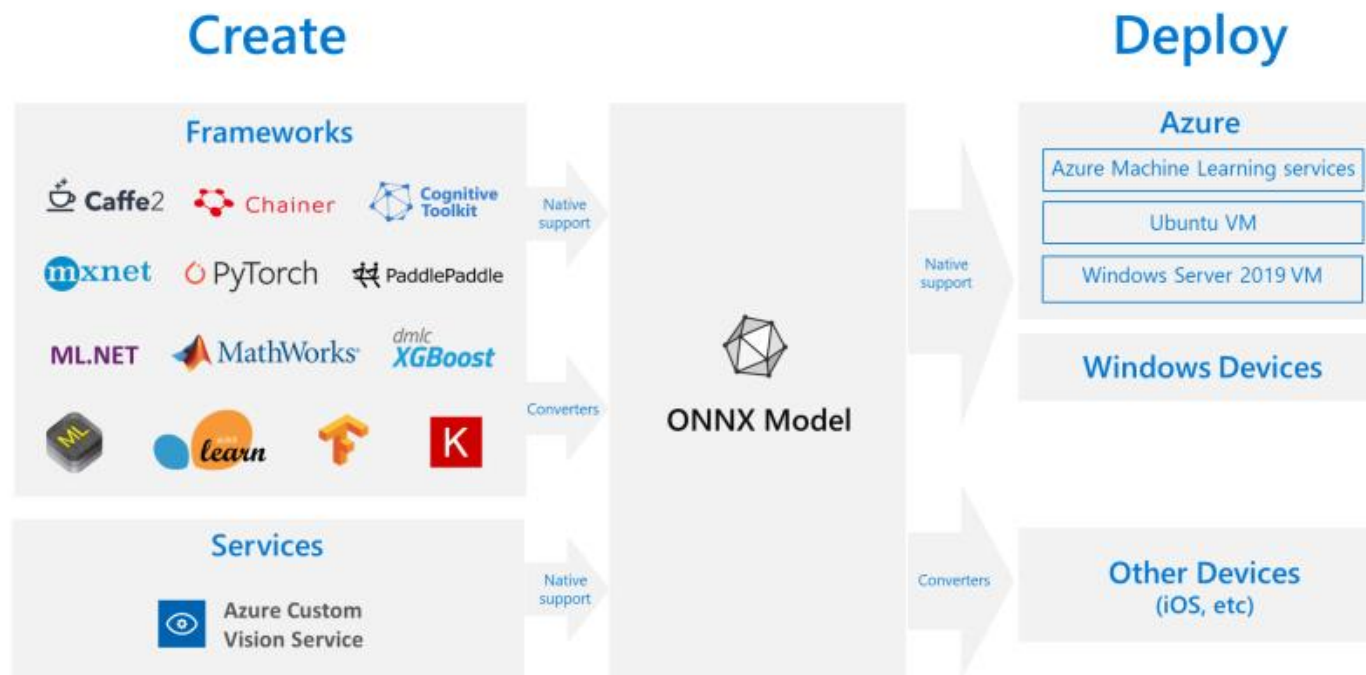
Open Neural Network Exchange(ONNX)(1/2)

- ONNX是一套開放神經網路交換格式，有助於將機器學習模型的判斷最佳化。Microsoft 和Intel、AMD、ARM等合作夥伴建立了機器學習的模型標準，增加神經網路架構的互通性，讓不同標準的深度學習模型可以互相轉換，加快AI人工智慧的發展。
- ONNX是一個高效能的運算推理引擎，可將 ONNX 模型安排到訓練環境中。它已針對雲端和邊緣最佳化，並可在Linux、Windows 和Mac上運作。支援DNN和傳統ML模型，並在不同硬體上提升運算速度。



Open Neural Network Exchange(ONNX)(2/2)

- 許多架構（包括 TensorFlow、PyTorch、Scikit-learn、Keras、Chainer、MXNET、MATLAB 和 SparkML）的模型都可以匯出或轉換成標準 ONNX 格式。一旦模型採用 ONNX 格式，就可以在各種不同的平台和裝置上執行。





Yolo模型轉換成ONNX檔

- 運用權重(.weight)與設定檔案(.cfg)轉換成.onnx檔

```
$ cd ~/trt_yolov4-tiny/yolo
```

- 影像大小取決於訓練的input_size

```
$ python3 yolo_to_onnx.py -c 1 -m yolov4-tiny-288
```

- .onnx轉成.trt檔

```
$ python3 onnx_to_tensorrt.py -c 1 -m yolov4-tiny-288
```

- 最後，將取得.trt來測試集影像。

- -c : 訓練時類別數量

- -m : 權重檔名稱



測試集影像預測

```
$python3 trt_yolo.py --image test/test.jpg -c 1 -m yolov4-tiny-288
```

- 以人臉資料集為範例:





測試集影片預測

- `$ python3 trt_yolo.py --video test/test.mp4 -c 1 -m yolov4-tiny-288`
- 以人臉資料集為範例:





即時交通號誌影像預測

- `$ python3 trt_yolo.py --onboard 0 -c class_num -m yolov4-tiny-288`
- `--onboard 0` : Jetson提供即時影像。

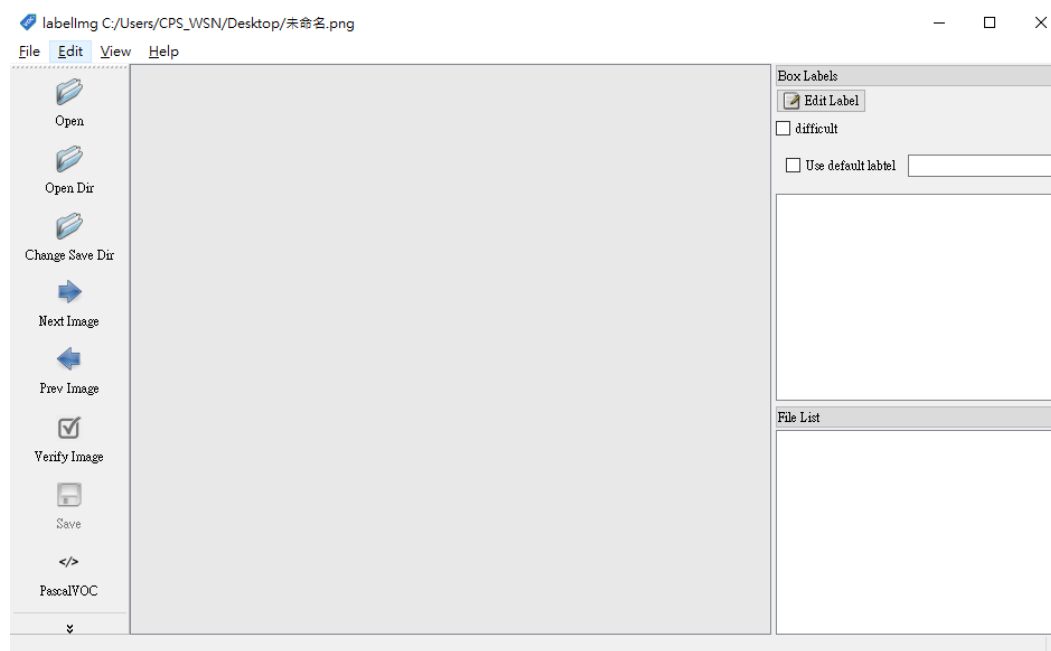


使用LabelImage工具



LabelImg

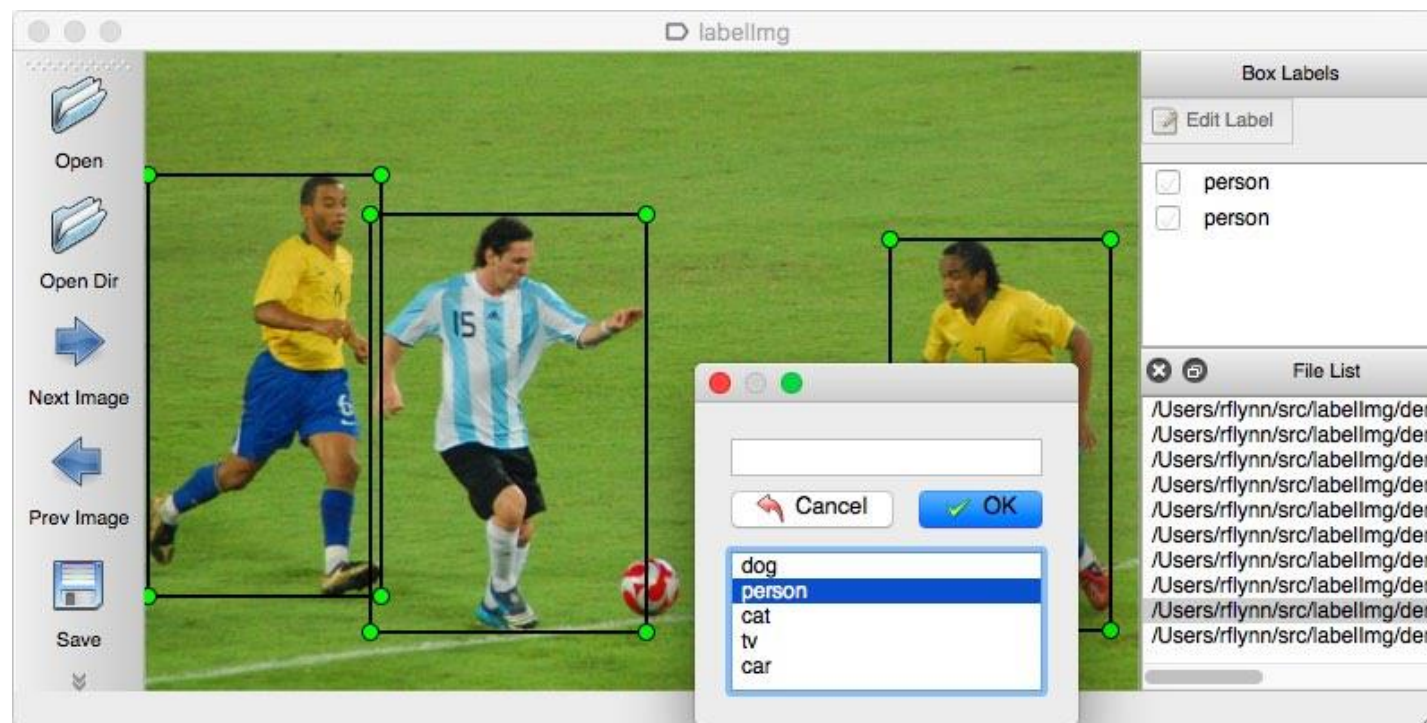
- LabelImg是一款Open Source軟體，專用於影像標記：
<https://github.com/tzutalin/labelImg>
- 到[Release](#)下載Windows版，若要編譯需下載Source Code。





LabelImg

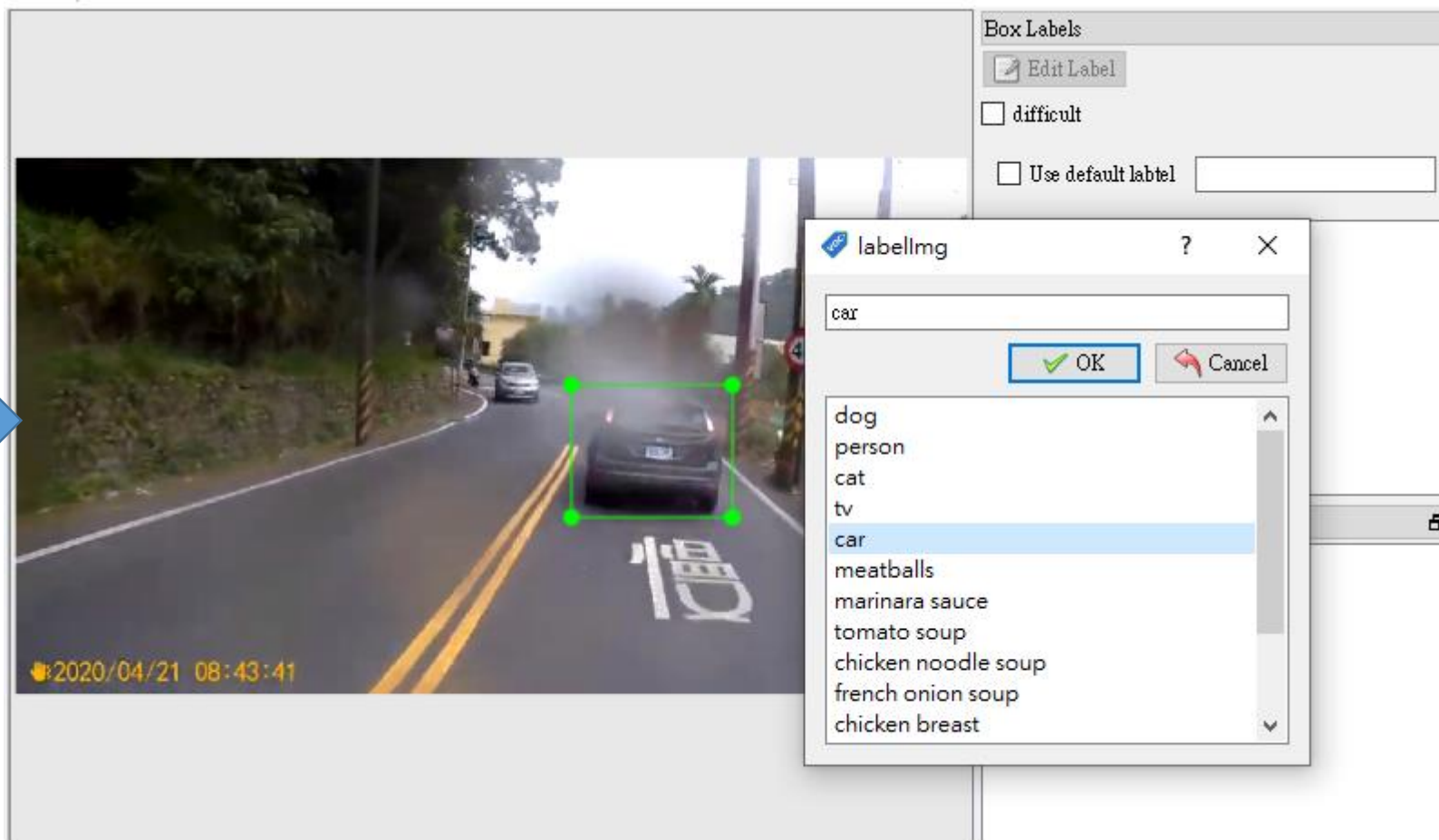
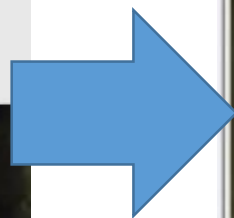
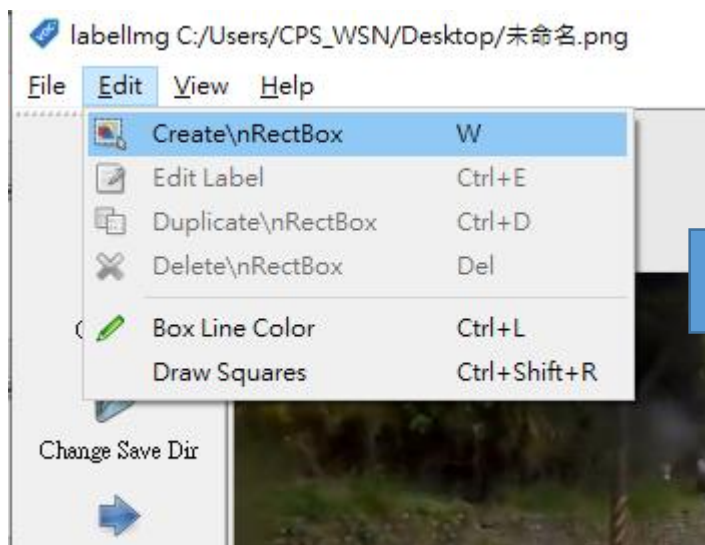
- 產生給機器學習和深度學習使用的樣本
- 支援普遍使用的PASCAL VOC或是YOLO的資料型態



- LabelImg程式示意圖

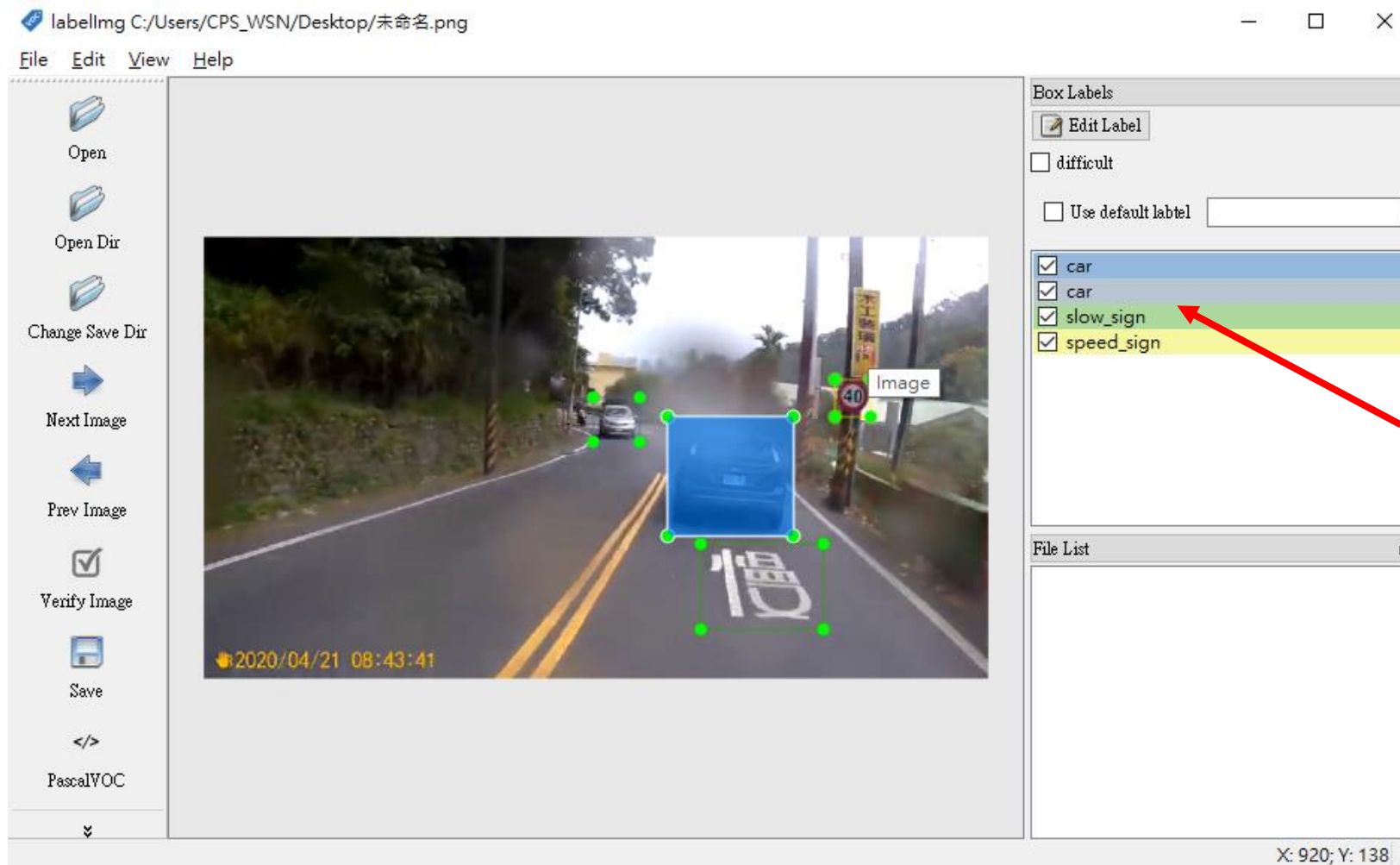


Labellmg操作





LabelImg操作



已經標過的物件會放在這



LabelImg操作



資料格式可以換成YOLO，pascal_voc儲存的副檔名為xml，YOLO則為txt



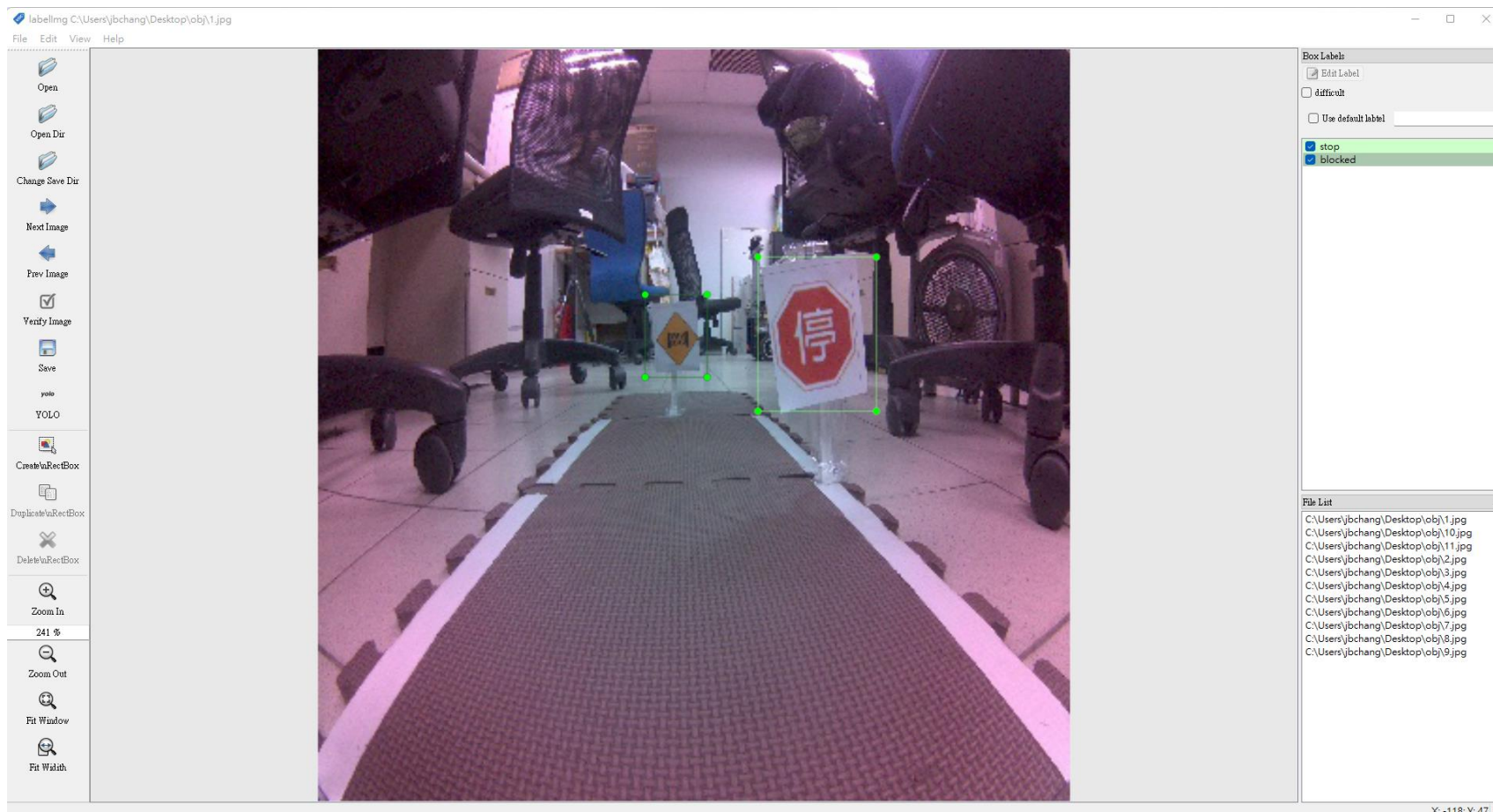
LabelImg操作

- 若存成YOLO用的.txt檔，格式為：
- 存成PascalVOC的.xml檔，格式為：

1. 物件的類別編號
2. bndBox的中心座標與圖片寬高的比值，是bndBox正規化後x的中心座標(除以圖片寬度)
3. bndBox的中心座標與圖片高度的比值，是bndBox正規化後y的中心座標(除以圖片高度)物件寬度(除以圖片寬度)
4. bndBox的寬度與輸入圖像寬度的比值，是bndBox歸一化後的寬度座標(除以圖片寬度)
5. bndBox的高度與輸入圖像高度的比值，是bndBox歸一化後的高度座標(除以圖片高度)

```
4 0.671089 0.541463 0.160110 0.269919
4 0.525618 0.413008 0.059469 0.100813
15 0.711345 0.791057 0.156450 0.193496
16 0.827081 0.364228 0.045746 0.084553
```

```
<?xml version="1.0"?>
- <annotation>
  <folder>Desktop</folder>
  <filename>未命名.png</filename>
  <path>C:/Users/CPS_WSN/Desktop/未命名.png</path>
+ <source>
- <size>
  <width>1093</width>
  <height>615</height>
  <depth>3</depth>
</size>
<segmented>0</segmented>
+ <object>
+ <object>
+ <object>
- <object>
  <name>speed_sign</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  - <bndbox>
    <xmin>879</xmin>
    <ymin>198</ymin>
    <xmax>929</xmax>
    <ymax>250</ymax>
  </bndbox>
</object>
</annotation>
```



使用GoogleColab雲端運算平台



使用Google Colab雲端運算平台

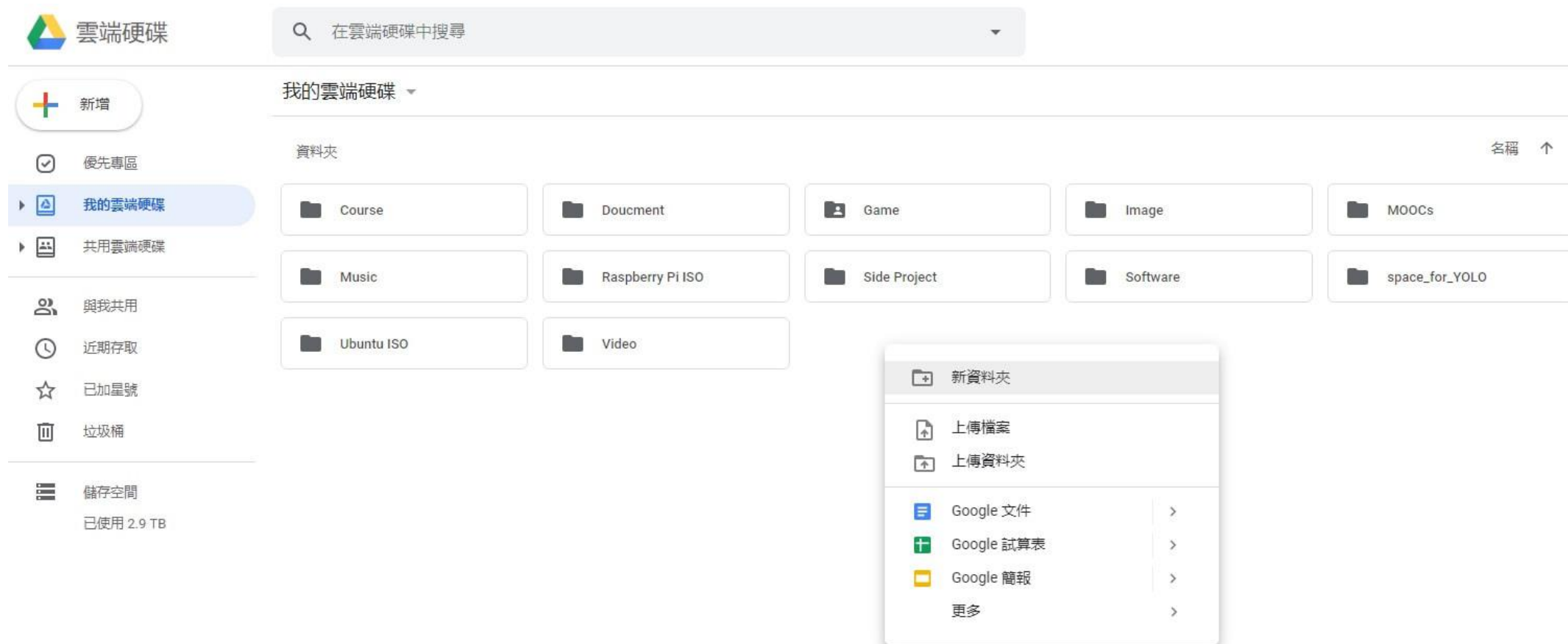
- Google Colaboratory (簡稱為Google Colab) 可讓你在瀏覽器上撰寫及執行 Python，且具備下列優點：
 - 不必進行任何設定
 - 免費使用 GPU
 - 輕鬆共用
- Colab 筆記本可讓你在單一文件中結合可執行的程式碼和 RTF 格式，並附帶圖片、HTML、LaTeX 等其他格式的內容。你建立的 Colab 筆記本會儲存到你的 Google 雲端硬碟帳戶中。你可以輕鬆將 Colab 筆記本與同事或朋友共用，讓他們在筆記本上加上註解，或甚至進行編輯。





使用Google Colab雲端運算平台

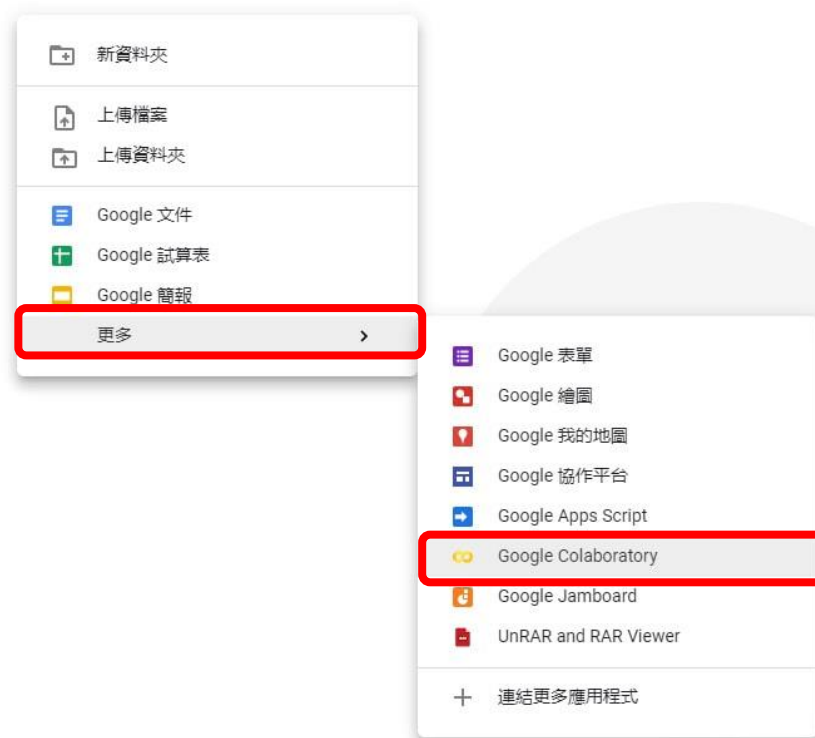
- 先在個人的 Google Drive 上新增一個資料夾。



使用Google Colab雲端運算平台

- 進入該資料夾後新增一個 Colab 文件

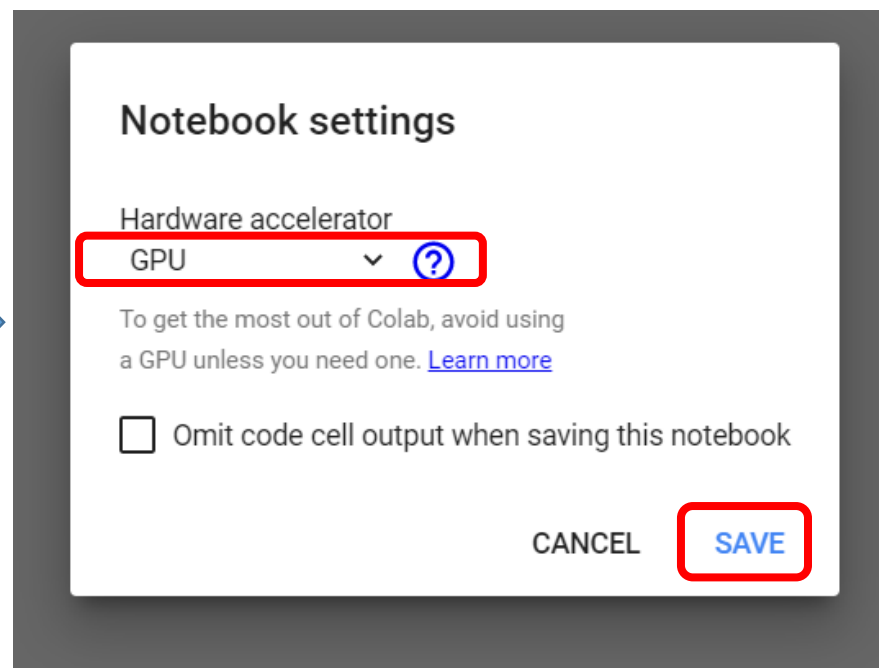
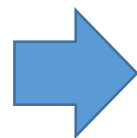
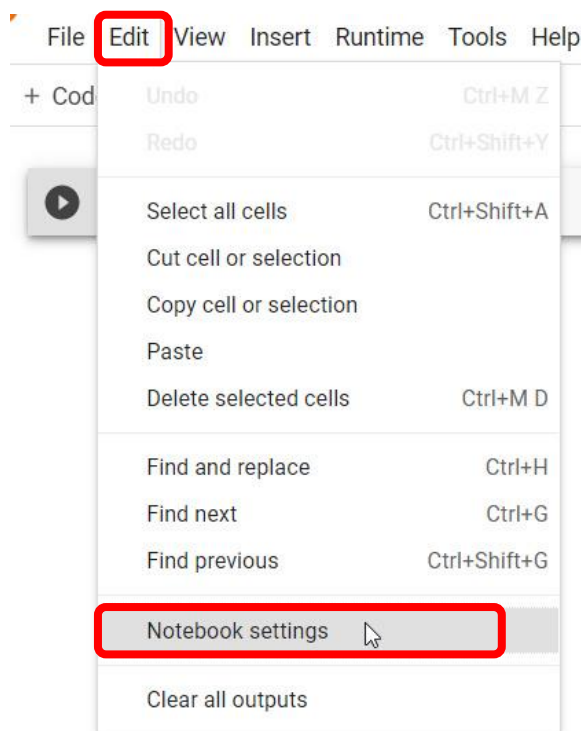
我的雲端硬碟 > space_for_YOLO ▾





使用Google Colab雲端運算平台

- 開啟 Google Colab 然後啟用免費的GPU
- 網址：<https://colab.research.google.com/>





使用Google Colab雲端運算平台

- 首先我們先 Copy 一份 darknet 到你的Colab空間

▼ Step 1 : Download the darknet repository

```
[ ] !git clone https://github.com/AlexeyAB/darknet
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 13221 (delta 0), reused 1 (delta 0), pack-reused 13218
Receiving objects: 100% (13221/13221), 11.92 MiB | 11.19 MiB/s, done.
Resolving deltas: 100% (9042/9042), done.
```



使用Google Colab雲端運算平台

- 下載完成後，我們需要修改 Makefile 裡面的四個參數，分別是：
 - GPU=0 要改成 GPU=1 (開啟GPU加速)
 - OPENCV=0 要改成 OPENCV=1 (用來讀取影像、影片、畫框等功能)
 - CUDNN=0 要改成 CUDNN=1 (用於加速tensorflow、pytorch等深度學習框架)
 - CUDNN_HALF=0 要改成 CUDNN_HALF=1 (建構tensor核心，加速偵測物件)

▼ Step 2 : Modify the Makefile to have GPU and OpenCV enabled

```
[ ] %cd darknet
    !sed -i 's/GPU=0/GPU=1/' Makefile
    !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
    !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
    !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
```

📁 /content/darknet

備註：若以上指令無法更改可自colab左邊目錄直接打開
/content/darknet/Makefile檔案，進行修改。



使用Google Colab雲端運算平台

- Makefile 修改完成後，我們就 make 指令來生成 darknet 這個深度學習引擎了。

▼ Step 3 : Make darknet

```
[ ] !make
```

- 在 make 完成後，其實 darknet 這套深度學習引擎就已經安裝完畢了，接下來我們會想要測試他是否能正常work，所以這時候我們就先去下載一些已經預先 train 好的 weights 檔做為測試用。



使用Google Colab雲端運算平台

▼ Step 4: Download pretrained YOLOv3 and YOLOv4 weights

YOLOv3 and YOLOv4 has been trained already on the coco dataset which has 80 classes that it can predict. We will grab these pretrained weights so that we can run YOLOv3 and YOLOv4 on these pretrained classes and get detections.

```
[ ] download_from_official = False

if download_from_official:
    # download weights form official
    !wget https://pjreddie.com/media/files/yolov3.weights
    !wget https://pjreddie.com/media/files/darknet53.conv.74
    !wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
    !wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
else:
    # download weights form Jason Chen's google drive (should be faster)
    !gdown https://drive.google.com/uc?id=1hSaT4Yc19atZZulW3Q3BUDFIohfGteXN
    !gdown https://drive.google.com/uc?id=1XpMMC_eUfHKAipfmxZa71IyocMW6ssCb
    !gdown https://drive.google.com/uc?id=1v0lvou7Pgv36l-ahej5IIIfYdb_9nmy0A
    !gdown https://drive.google.com/uc?id=1Zl3rh0ZOPVj4DPZdae8WqqZU0NLX7Ncp
```

```
📁 Downloading...
From: https://drive.google.com/uc?id=1hSaT4Yc19atZZulW3Q3BUDFIohfGteXN
To: /content/darknet/yolov3.weights
248MB [00:03, 62.3MB/s]
Downloading...
From: https://drive.google.com/uc?id=1XpMMC\_eUfHKAipfmxZa71IyocMW6ssCb
To: /content/darknet/darknet53.conv.74
162MB [00:01, 100MB/s]
Downloading...
From: https://drive.google.com/uc?id=1v0lvou7Pgv36l-ahej5IIIfYdb\_9nmy0A
To: /content/darknet/yolov4.weights
258MB [00:02, 101MB/s]
Downloading...
From: https://drive.google.com/uc?id=1Zl3rh0ZOPVj4DPZdae8WqqZU0NLX7Ncp
To: /content/darknet/yolov4.conv.137
170MB [00:00, 205MB/s]
```



使用Google Colab雲端運算平台

- 在 weights 下載好之後，我們就可以使用：
`!./darknet detect <path of .cfg file> <path of .weights file> <path of picture>`
這個指令進行辨識，辨識的結果會以 "predictions.jpg" 存在跟 darknet 同一個目錄底下，為了可以直接 show 在 notebook 上面，我們可以多寫一個 imshow 的 function 來實現。



使用Google Colab雲端運算平台

▼ Step 5 : Run Object Detection with Darknet and YOLOv3/v4

Define the show image function

```
[ ] def imShow(path):  
    import cv2  
    import matplotlib.pyplot as plt  
    %matplotlib inline  
  
    image = cv2.imread(path)  
    height, width = image.shape[:2]  
    resized_image = cv2.resize(image, (3*width, 3*height), interpolation = cv2.INTER_CUBIC)  
  
    fig = plt.gcf()  
    fig.set_size_inches(18, 10)  
    plt.axis("off")  
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))  
    plt.show()
```




使用Google Colab雲端運算平台

▼ >>> 5-1. Run detection with YOLOv3

```
[ ] # run darknet detection
    !./darknet detect cfg/yolov3.cfg yolov3.weights data/person.jpg

    # show result
    imshow('predictions.jpg')
```

▼ >>> 5-2. Run detection with YOLOv4

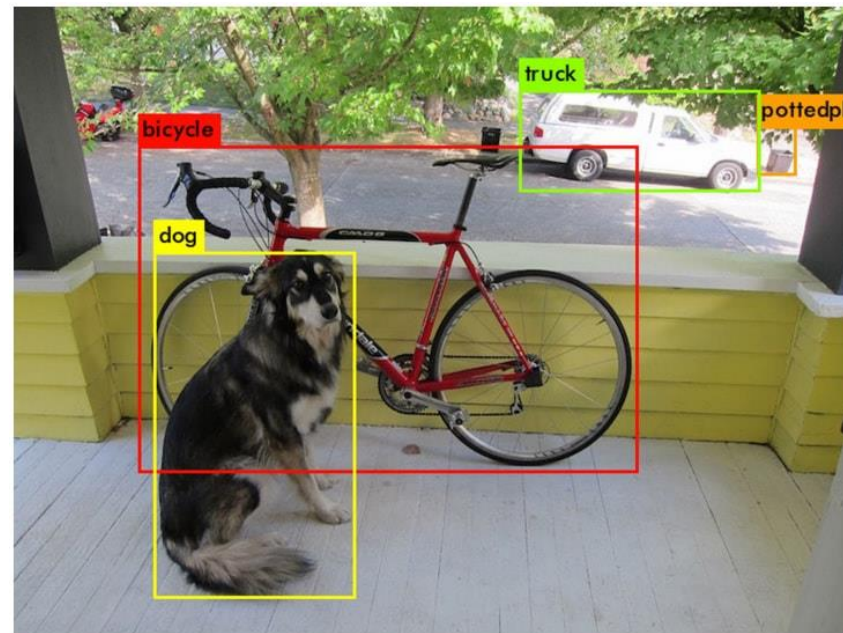
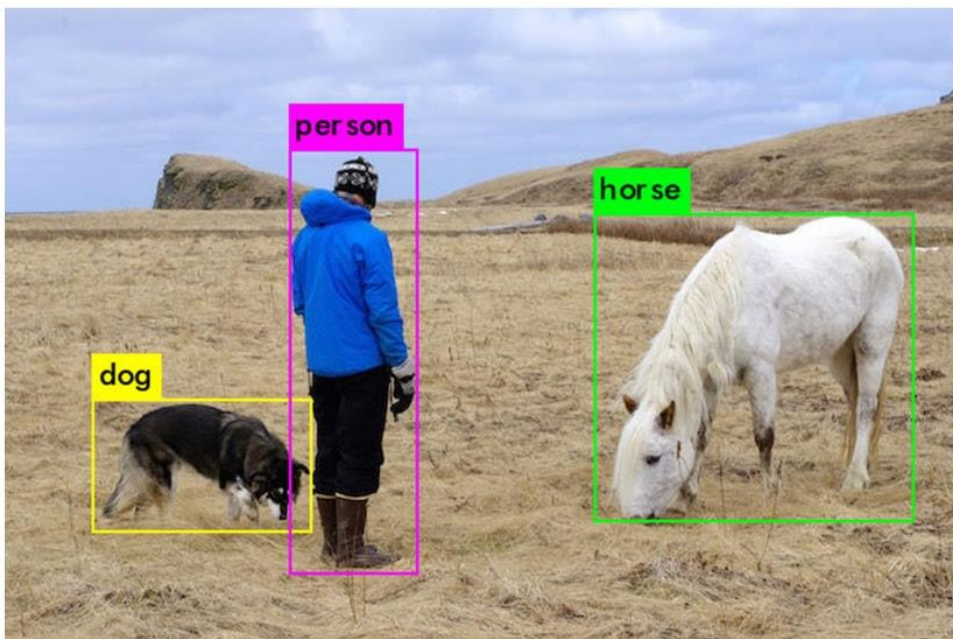
```
[ ] # run darknet detection
    !./darknet detect cfg/yolov4.cfg yolov4.weights data/dog.jpg

    # show result
    imshow('predictions.jpg')
```



使用Google Colab雲端運算平台

- 如果你的 darknet 能夠正常 work 的話，你應該能夠看到下方這兩張圖的辨識結果。





使用Google Colab雲端運算平台

- Google Colab連結自己的Google雲端硬碟空間
 - 由於Colab的空間是一個臨時的工作空間，在建立當初是沒有任何檔案的，此時可將Colab連結自己的Google雲端硬碟空間，以利將事先準備好的obj.names、obj.data等檔案複製到Colab的工作空間，或是將訓練好的.weight權重檔案複製回雲端。



#掛載自己的google雲端硬碟

```
from google.colab import drive  
drive.mount('/content/drive')
```

#將google雲端硬碟路徑簡寫為mydrive

```
!ln -s /content/drive/My\ Drive/ /mydrive  
!ls /mydrive
```



使用Google Colab雲端運算平台

- 準備好訓練YOLOv4-tiny模型所需檔案，並上傳到雲端：
 1. obj.zip (包含資料集影像與同名的.txt座標文件(使用LabelImg工具取得))
 2. obj.names (自己資料集的類別名稱)
 3. obj.data (相關路徑及各種參數設定)
 4. Yolov4-tiny-custom.cfg (自己根據資料及類別數量修改)
 5. generate_train.py (用來產生train.txt)



使用Google Colab雲端運算平台

- 上傳obj.zip到雲端：
 - 通過LabelImg或Yolo_mark等標記工具生成專案所需的與圖片同名的.txt檔，將原圖片與同名.txt文件放在同一文件obj下，將資料夾壓縮為obj.zip並上傳到Google雲端的yolov4資料夾下。obj.zip包含的檔案如下所示。

名稱	修改日期	類型	大小
1.jpg	2021/5/5 下午 07:53	JPG 檔案	232 KB
1.txt	2021/5/10 下午 03:03	文字文件	1 KB
2.jpg	2021/5/5 下午 07:53	JPG 檔案	233 KB
2.txt	2021/5/5 下午 08:53	文字文件	1 KB
10.jpg	2021/5/5 下午 07:53	JPG 檔案	240 KB
10.txt	2021/5/10 下午 01:54	文字文件	1 KB
11.jpg	2021/5/5 下午 07:53	JPG 檔案	241 KB
11.txt	2021/5/5 下午 09:00	文字文件	1 KB
12.jpg	2021/5/5 下午 07:53	JPG 檔案	241 KB
12.txt	2021/5/5 下午 09:04	文字文件	1 KB
13.jpg	2021/5/5 下午 07:53	JPG 檔案	240 KB
13.txt	2021/5/10 下午 01:56	文字文件	1 KB
14.jpg	2021/5/5 下午 07:53	JPG 檔案	241 KB
14.txt	2021/5/10 下午 02:05	文字文件	1 KB
15.jpg	2021/5/5 下午 07:53	JPG 檔案	240 KB
15.txt	2021/5/10 下午 02:20	文字文件	1 KB
16.jpg	2021/5/5 下午 07:53	JPG 檔案	240 KB
16.txt	2021/5/10 下午 03:03	文字文件	1 KB
17.jpg	2021/5/5 下午 07:53	JPG 檔案	240 KB
17.txt	2021/5/10 下午 03:03	文字文件	1 KB



使用Google Colab雲端運算平台

- 上傳obj.data和obj.names到雲端：
 - obj.data 根據自己資料集的類別個數進行修改而得到。
 - obj.names中每行寫入自己資料集的類別名稱。

```
1 classes=4
2 train = data/train.txt
3 valid = data/train.txt
4 names = data/obj.names
5 backup = backup
```

obj.data

```
1 stop
2 pedestrian
3 rail
4 blocked
```

obj.names



使用Google Colab雲端運算平台

- 上傳yolov4-tiny-custom.cfg到雲端：
 - Yolov4-tiny-custom.cfg在darknet/cfg/yolov4-tiny-custom.cfg的基礎上，根據自己資料集的類別個數進行修改得到。
 - 修改一：
 - batch = 64, subdivisions = 16不變當訓練運行顯示“out of memory”時，可以嘗試將subdivisions升至32或64。
 - max_batches = ? 為自己資料集種類個數*2000，如果有一類即為2000
 - Steps = ?, ? 分別為max_batches的80%和90%

(可依據自己專案情況決定增加or減少訓練迭代次數)

```
batch=64
subdivisions=32
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000
max_batches = 2000
policy=steps
steps=1600,1800
scales=.1,.1
```




使用Google Colab雲端運算平台

- 修改二：
 - $\text{filters} = (5 + \text{classes}) \times 3$ 如資料集種類個數classes為4 則filters=27
 - 共要修改兩個部分，如下圖所示：

與yolov4不同，此專案只有兩個部份要修改

```
[convolutional]
size=1
stride=1
pad=1
filters=27
activation=linear

[yolo]
mask = 3,4,5
anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
classes=4
num=6
jitter=.3
```

```
[convolutional]
size=1
stride=1
pad=1
filters=27
activation=linear

[yolo]
mask = 0,1,2
anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
classes=4
num=6
jitter=.3
```




使用Google Colab雲端運算平台

- 上傳generate_train.py文件到雲端：
 - 此yolov4專案訓練自己資料集還需要train.txt，train.txt檔可通過運行generate_train.py生成，所以在此之前需要創建generate_train.py，並上傳到雲端中，以便後續複製到Colab中並運行。generate_train.py文件如下：

```
1 import os
2
3 image_files = []
4 os.chdir(os.path.join("data", "obj"))
5 for filename in os.listdir(os.getcwd()):
6     if filename.endswith(".jpg"):
7         image_files.append("data/obj/" + filename)
8 os.chdir("..")
9 with open("train.txt", "w") as outfile:
10     for image in image_files:
11         outfile.write(image)
12         outfile.write("\n")
13     outfile.close()
14 os.chdir("..")
```

執行此python檔會根據obj.zip內檔案內容自動產生所有路徑的train.txt檔案



使用Google Colab雲端運算平台

在train自己的model時，須有一個預訓練權重檔案當基底

```
!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg data/yolov4-tiny.conv.29 -dont_show
```

進行訓練

obj.data位置

.cfg位置

預訓練權重檔案

訓練過程不要顯示任何視窗

此次專案使用yolo-tiny.conv.29作為預訓練權重檔



訓練完的模型轉換成ONNX檔

- 將yolov4-tiny-custom.cfg和yolov4-tiny-custom_final.weights複製到yolo資料夾並改名為yolov4-tiny-416.cfg和yolov4-tiny-416.weights

```
$ cd ~/trt_yolov4-tiny/yolo
```

- 將weights轉成.onnx檔

```
$ python3 yolo_to_onnx.py -c 4 -m yolov4-tiny-416
```

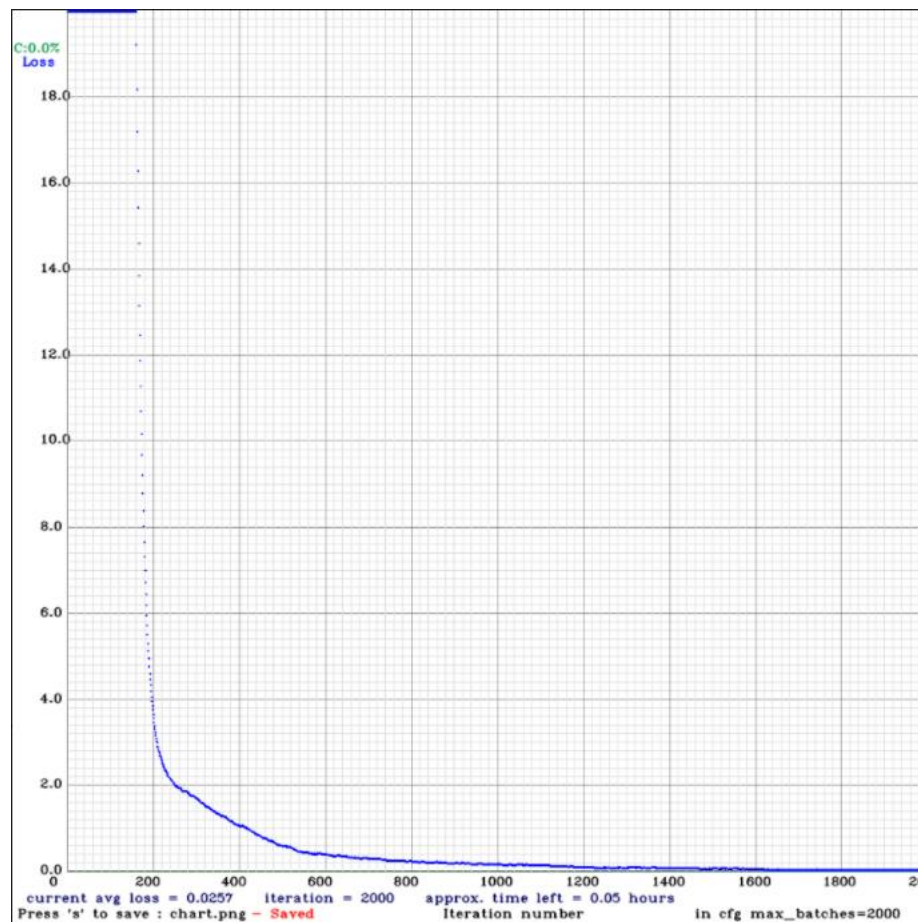
- .onnx轉成.trt檔

```
$ python3 onnx_to_tensorrt.py -c 4 -m yolov4-tiny-416
```

- 最後，將取得.trt來測試集影像。

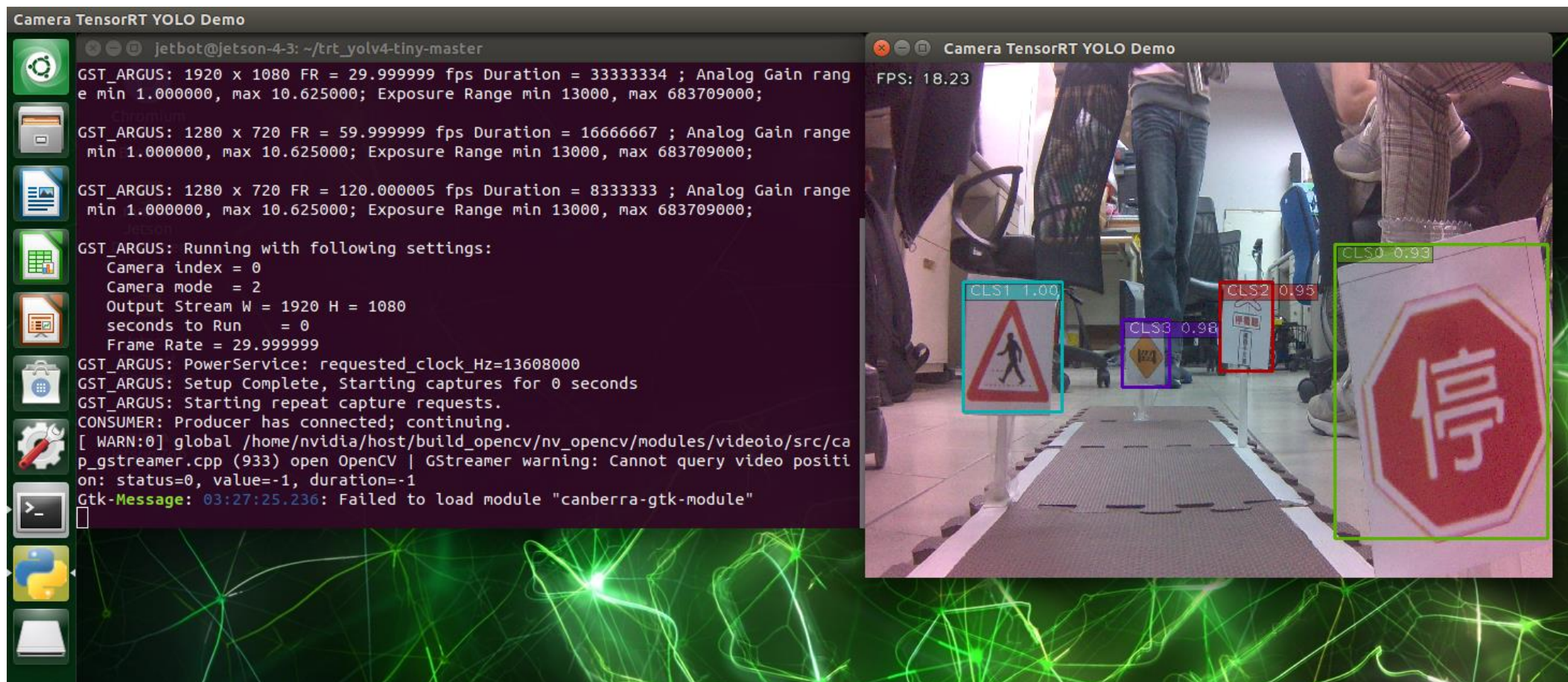


YOLOv4-tiny訓練





YOLOv4-tiny訓練成果









Project Demo





專案要求

- 收集資料集的部分，將交通號誌放置於道路左或右，讓Jetbot收集不同角度的影像資料集，最後彙整資料集帶入Yolov4-tiny中。
- 偵測交通號誌需執行的動作：
 -  • 停車在開:偵測到此號誌時，於原地停止2秒後再繼續行駛，停止位置不得超過交通號誌及距離遠於15公分。
 -  • 鐵路平交道:偵測到此號誌時，於原地停止5秒後再繼續行駛，停止位置不得超過交通號誌及距離遠於15公分。
 -  • 當心行人:偵測到此號誌時，減速後再繼續行駛，減速位置不得超過交通號誌及距離遠於15公分。
 -  • 道路封閉:偵測到此號誌時停止Jetbot，停止位置不得超過交通號誌及距離遠於15公分。



小組報告格式規定

- 專案情境
 - 小組所討論出來的議題，並簡明扼要描述議題的情境。
- 定義問題
 - 將議題中的問題定義出來，並收斂問題方向。
- 方案構思
 - 簡單描述如何解決定義好的問題，並預計使用的技術。
- 解決方法
 - 說明實際上如何完成此議題的方案構思。
- 分工
 - 說明小組成員分工內容與比例。



個人報告內容

- 個人報告內容須要有以下內容：
 - 你一開始所提出的議題是?你的議題是否有被選為小組議題候選?
 - 你在小組議題中，提出了那些問題與解決方案?是否有被小組接受?
 - 如個人所提出的方案沒被接受，是因為那些原因?
 - 為了這個議題，你去找了那些資料?你是如何分析找到的資料?
 - 其他小組成員所提出的提議有哪些?而你對於其他人的提議意見如何?
 - 在小組決定小組議題過程中，你對於小組最後提出的議題討論是否能接受?接受理由為何?不接受理由為何?
 - 你是否能接受最後的議題與方案?如接受請說明接受與否的理由?
 - 本次專案個人的心得
 - 本次你認為小組成員的貢獻比例及理由



專案繳交規則

- 專案成果實體驗收請於110/12/24課程結束前找助教檢查
- 小組報告繳交期限:110/12/24 23:59(以I學園上傳時間為基準)
- 個人報告繳交期限:110/12/24 23:59(以I學園上傳時間為基準)
- 補交規則
 - 超過正常繳交期限兩周內成績**打8折**
 - 超過正常期限兩周後不接受補交