

# 多媒體技術與應用 Spring 2021

Instructor : Yen-Lin Chen(陳彥霖), Ph.D.

Professor

Dept. Computer Science and Information Engineering

National Taipei University of Technology

# Lecture 8

影像深度學習網路模型使用與Google Colab雲端運算平台



# 簡介

- 本投影片將介紹：
  1. 使用LabelImg標記工具
  2. Google Colab使用說明
  3. 使用YOLOv4 (訓練模型、執行模型)

# 使用 Yolo v4





# 執行YOLOv4

- 請注意：本節執行YOLOv4為「在Ubuntu作業系統且具備顯示卡並安裝好CUDA、cuDNN等套件」環境下所執行的過程，由於在Windows系統下執行YOLO需進行較複雜的設定，而「Google Colab雲端運算平台」已事先將環境設定完成，僅需對darknet進行make即可直接執行YOLO。因此若非前述紅字提及的系統環境，建議可以直接參考本投影片後半段「使用Google Colab 雲端運算平台」部分即可。



# 執行YOLOv4

- 將專案clone出來並切換至darknet目錄
- `git clone https://github.com/AlexeyAB/darknet.git`

```
evil1323@evil1323:~$ git clone https://github.com/AlexeyAB/darknet.git
fatal: destination path 'darknet' already exists and is not an empty directory.
evil1323@evil1323:~$ git clone https://github.com/AlexeyAB/darknet.git
Cloning into 'darknet'...
remote: Enumerating objects: 14291, done.
remote: Total 14291 (delta 0), reused 0 (delta 0), pack-reused 14291
Receiving objects: 100% (14291/14291), 12.84 MiB | 283.00 KiB/s, done.
Resolving deltas: 100% (9751/9751), done.
Checking connectivity... done.
evil1323@evil1323:~$ ls
3.4.7.zip          Desktop            examples.desktop  Public
anaconda3          d_labelImg        f_labeling        pytorch-YOLOv4
Anaconda3-2019.10-Linux-x86_64.sh  dlib              Music             Templates
darknet            Documents          opencv-3.4.7     Videos
DataSet           Downloads         Pictures          yolo_face
evil1323@evil1323:~$ cd darknet/
evil1323@evil1323:~/darknet$
```



# 執行YOLOv4 (cont.)

- 修改Makefile的內容，如下圖

```
Makefile (-/YoloV3_Demo/darknet) - gedit
Open Save
GPU=1
CUDNN=1
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBS0=1
ZED_CAMERA=0 # ZED SDK 3.0 and above
ZED_CAMERA_v2_8=0 # ZED SDK 2.X

# set GPU=1 and CUDNN=1 to speedup on GPU
# set CUDNN_HALF=1 to further speedup 3 x times (Mixed-precision on Tensor Cores) GPU: Volta,
# set AVX=1 and OPENMP=1 to speedup on CPU (if error occurs then set AVX=0)

USE_CPP=0
DEBUG=0

ARCH= -gencode arch=compute_53,code=[sm_53,compute_53] \
      -gencode arch=compute_62,code=[sm_62,compute_62]

OS := $(shell uname)
```





# 執行YOLOv4 (cont.)

- 修改Makefile的內容，如下圖

```
Makefile
~/YoloV3_Demo/darknet

# ARCH= -gencode arch=compute_35,code=[sm_35,compute_35]

# For Jetson Tx2 or Drive-PX2 uncomment:
# ARCH= -gencode arch=compute_62,code=[sm_62,compute_62]

VPATH=./src/
EXEC=darknet
OBJDIR=./obj/

ifeq ($(LIBS), 1)
LIBNAMES0=libdarknet.so
APPNAMES0=uselib
endif

ifeq ($(USE_CPP), 1)
CC=g++
else
CC=gcc
endif

CPP=g++ -std=c++11
NVCC=/usr/local/cuda-9.0/bin/nvcc
OPTS=-Ofast
LDFLAGS= -lm -pthread
COMMON= -Iinclude/ -I3rdparty/stb/include
CFLAGS=-Wall -Wfatal-errors -Wno-unused-result -Wno-unknown-pragmas -fPIC
```





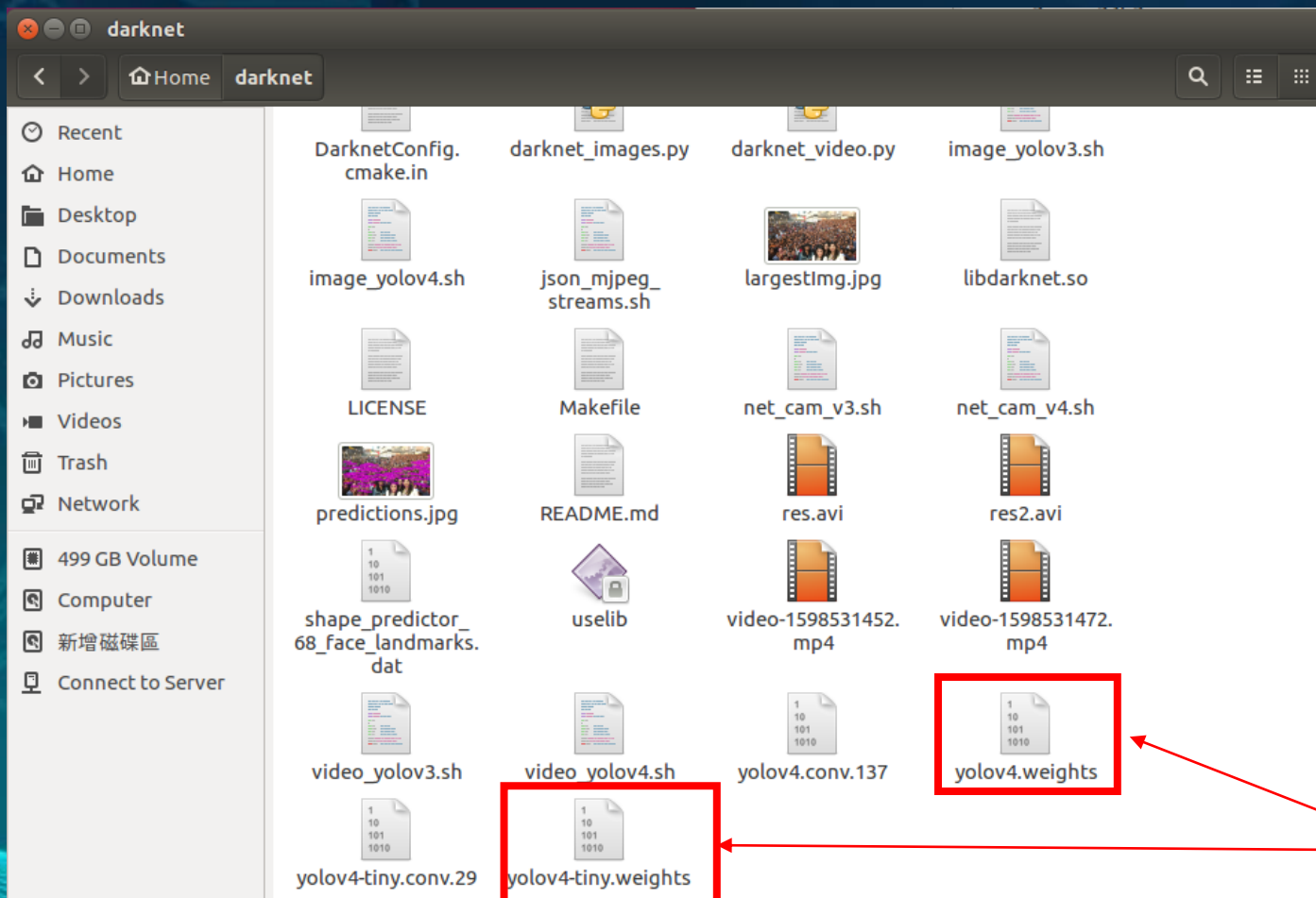
# 執行YOLOv4 (cont.)

- 修改Makefile的內容後
- 輸入指令: `sudo make`

```
evil1323@evil1323:~/darknet$ sudo make
[sudo] password for evil1323:
Sorry, try again.
[sudo] password for evil1323:
mkdir -p ./obj/
mkdir -p backup
chmod +x *.sh
g++ -std=c++11 -std=c++11 -Iinclude/ -I3rdparty/stb/include -Wall -Wfatal-errors
-Wno-unused-result -Wno-unknown-pragmas -fPIC -Ofast -c ./src/image_opencv.cpp
-o obj/image_opencv.o
```



# 執行YOLOv4 (cont.)



下載權重檔案，並放置於資料夾內。



# Darknet\_video.py程式碼說明

```
darknet_video.py (~/YoloV3_Demo/darknet) - gedit
Open Save
command x darknet_video.py x
from ctypes import *
import math
import random
import os
import cv2
import numpy as np
import time
import darknet

def convertBack(x, y, w, h):
    xmin = int(round(x - (w / 2)))
    xmax = int(round(x + (w / 2)))
    ymin = int(round(y - (h / 2)))
    ymax = int(round(y + (h / 2)))
    return xmin, ymin, xmax, ymax

def cvDrawBoxes(detections, img):
    for detection in detections:
        x, y, w, h = detection[2][0],\
            detection[2][1],\
            detection[2][2],\
            detection[2][3]
        xmin, ymin, xmax, ymax = convertBack(
            float(x), float(y), float(w), float(h))
        pt1 = (xmin, ymin)
        pt2 = (xmax, ymax)
        cv2.rectangle(img, pt1, pt2, (0, 255, 0), 1)
        cv2.putText(img,
            detection[0].decode() +
            " [" + str(round(detection[1] * 100, 2)) + "]",
            (pt1[0], pt1[1] - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
            [0, 255, 0], 2)
    return img
```

畫出偵測到的物件的Bounding box

putText的參數為:

(影像,添加的文字,左上角坐標,字體,字體大小,顏色,字體粗細)

detection[0].decode()為偵測到的物件類別名稱,可透過這部分來實作出所偵測到的目標物數量。



# Darknet\_video.py程式碼修改

```
darknet_video.py (~/.YoloV3_Demo/darknet) - gedit
Open [icon] Save
    pass
except Exception:
    pass
cap = cv2.VideoCapture(1)
#cap = cv2.VideoCapture("test.mp4")
cap.set(3, 1280)
cap.set(4, 720)
out = cv2.VideoWriter(
    "output.avi", cv2.CV_FOURCC('M', 'J', 'P', 'G'), 10.0,
    (darknet.network_width(netMain), darknet.network_height(netMain)))
print("Starting the YOLO loop...")
```

```
cap = cv2.VideoCapture("test.mp4")
cap.set(3, 1280)
cap.set(4, 720)
out = cv2.VideoWriter(
    "output.avi", cv2.VideoWriter_fourcc(*"MJPG"), 10.0,
    (darknet.network_width(netMain), darknet.network_height(netMain)))
print("Starting the YOLO loop...")
```

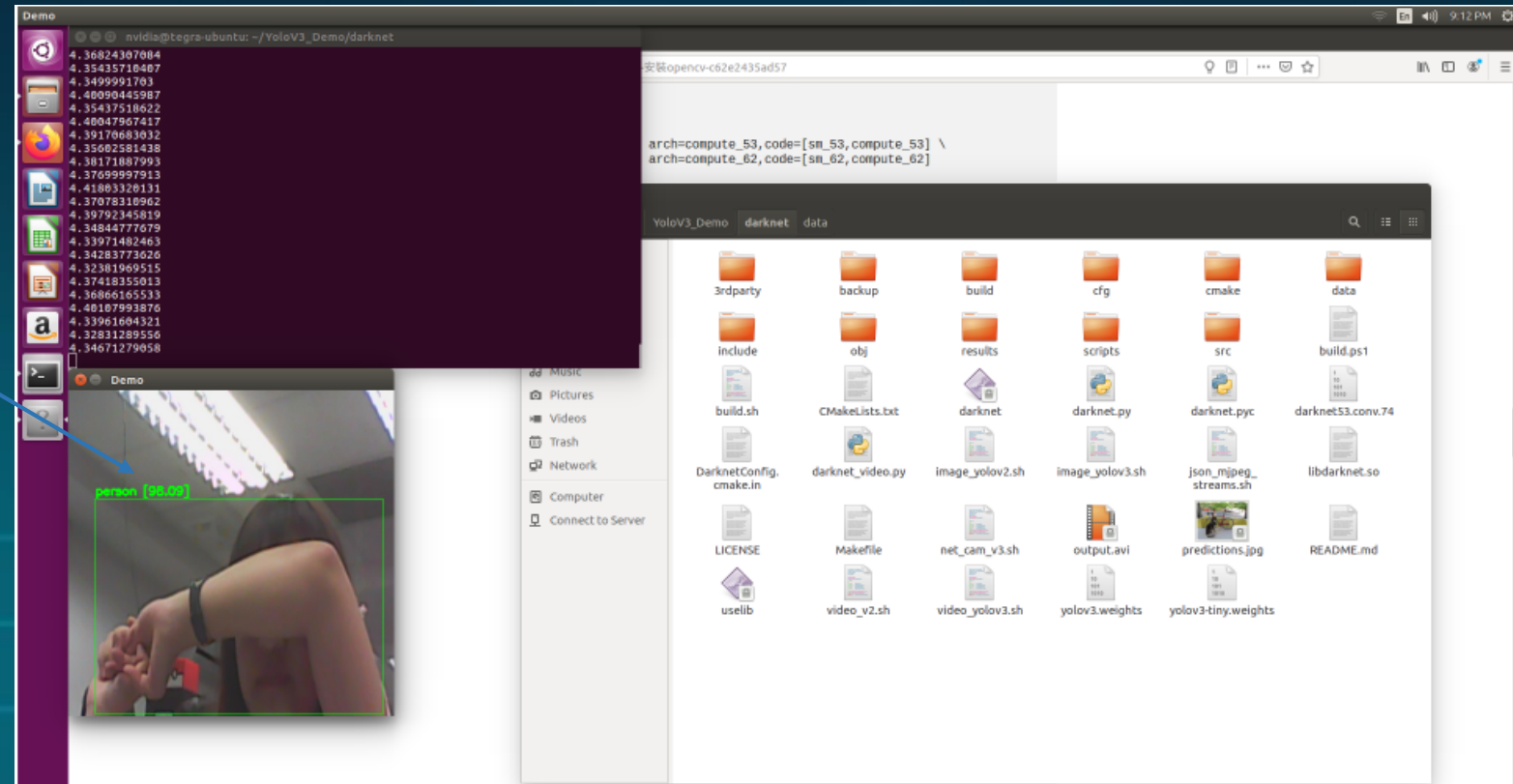
- 此為opencv2.4.3版本的寫法，若是要使用opencv3寫法請參考黃色框處



# 使用YOLOv4偵測結果

- 在terminal上執行python darknet\_video.py的結果

執行出來的結果為此圖

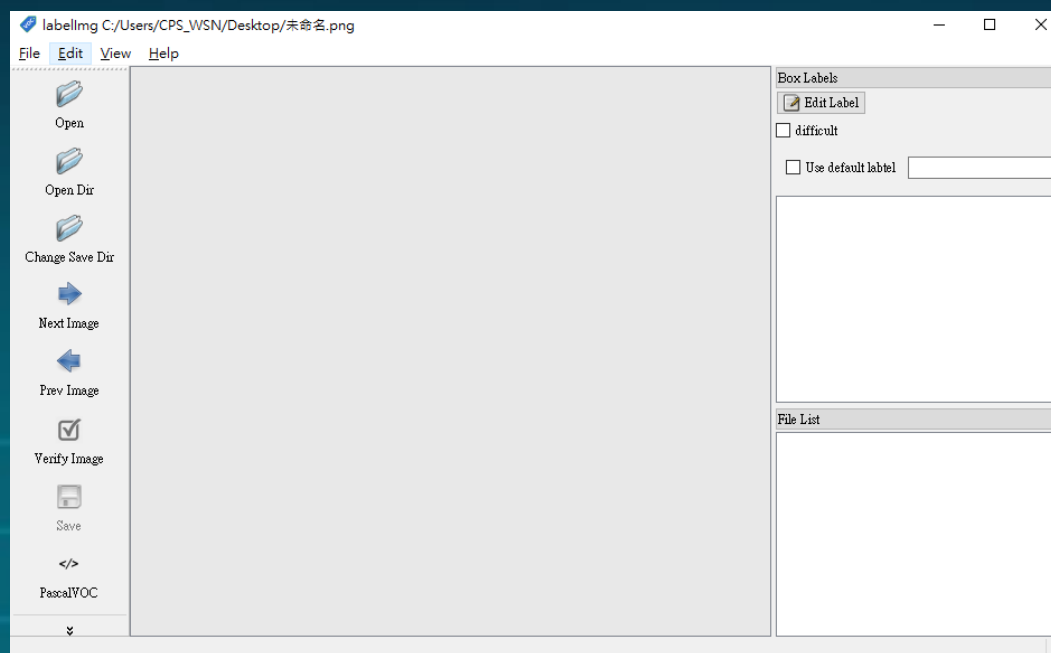


# 使用LabelImge工具



# 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。

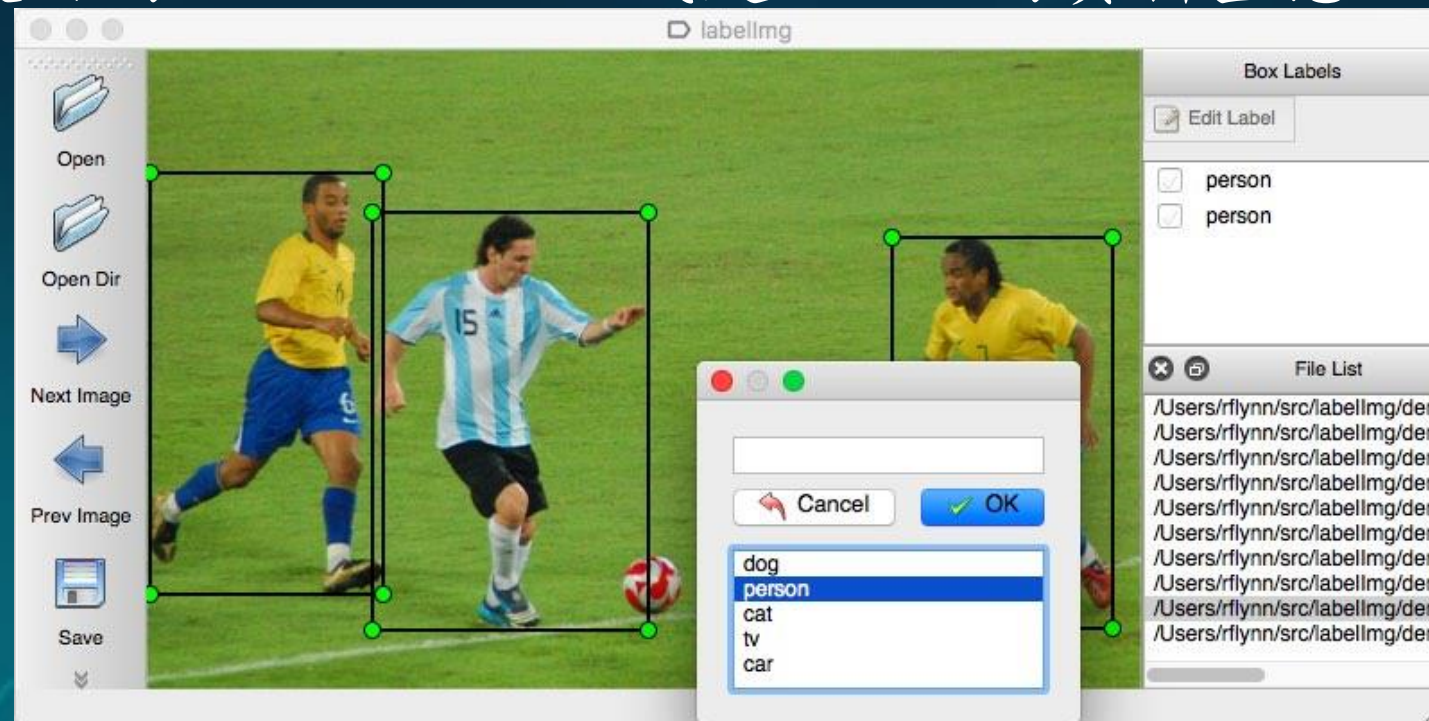






# LabelImg

- 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

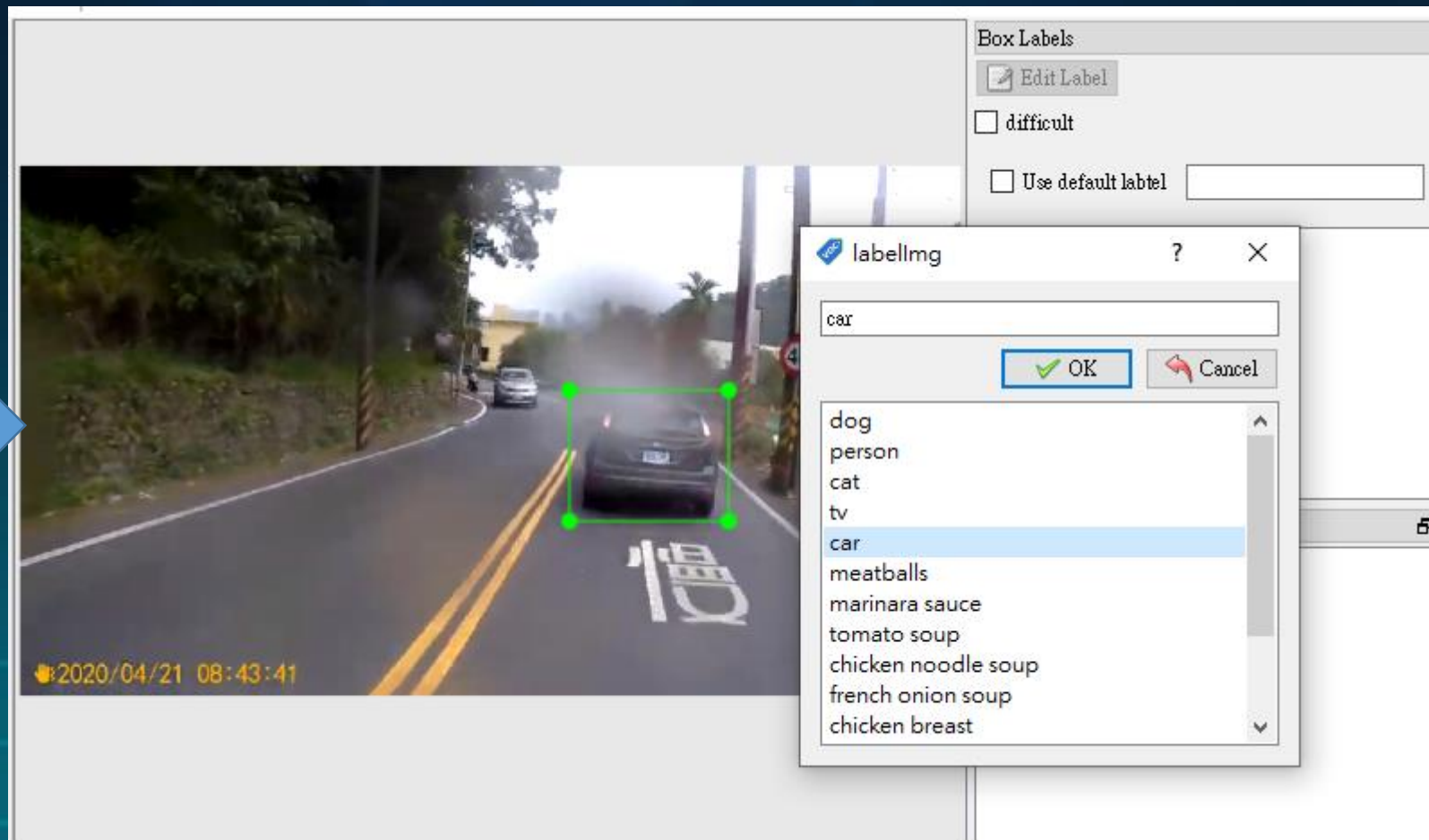
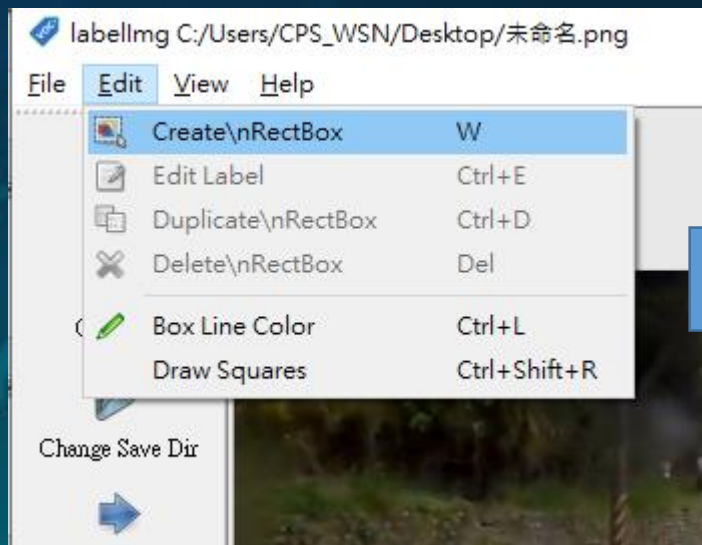


- LabelImg程式示意圖



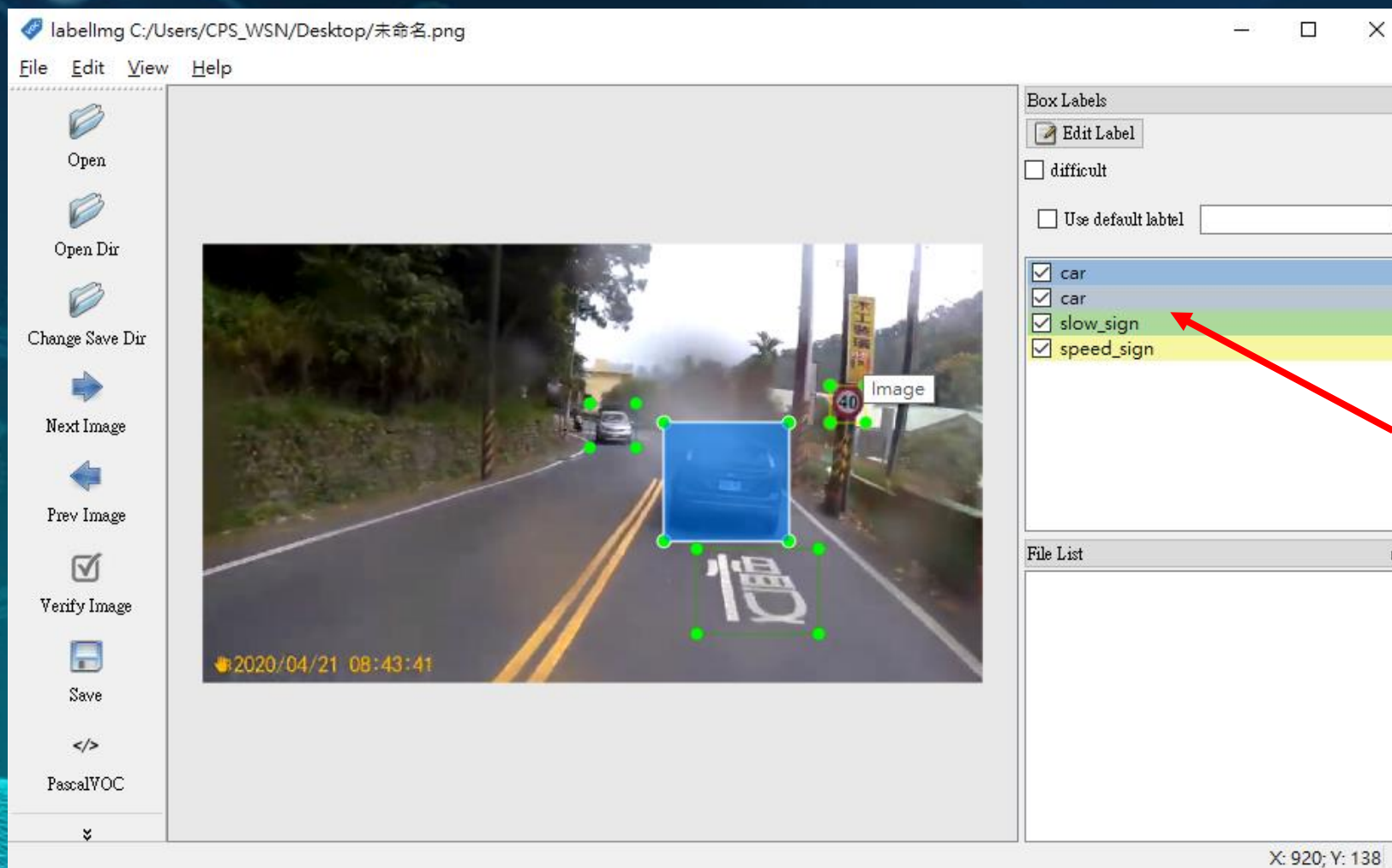


# LabelImg操作





# LabelImg操作



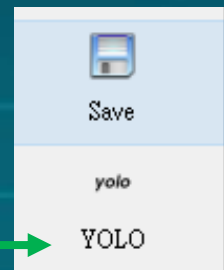
已經標過的物件會放在這



# 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. 左上點的x座標(除以圖片寬度)
3. 左上點的y座標(除以圖片長度)
4. 物件寬度(除以圖片寬度)
5. 物件長度(除以圖片長度)

```
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

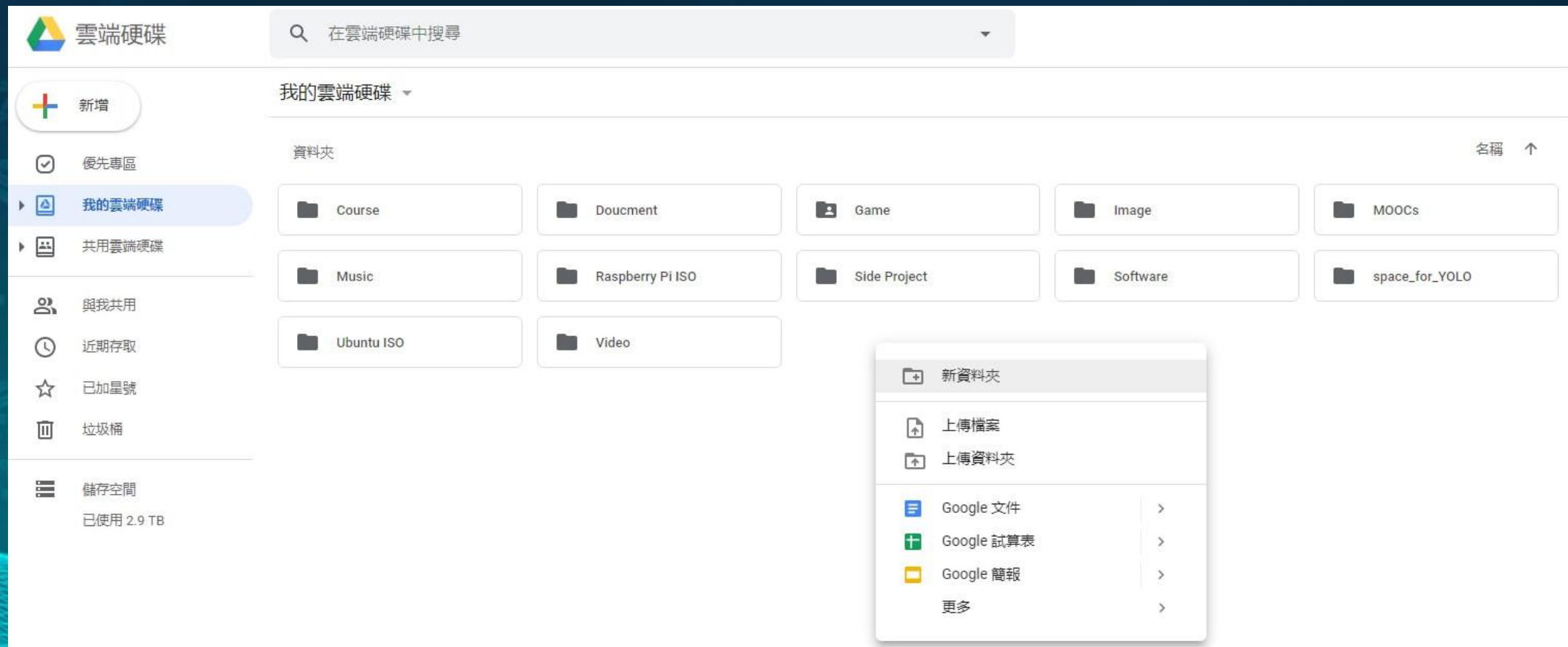
- Google Colaboratory (簡稱為 Google Colab) 可讓你在瀏覽器上撰寫及執行 Python，且具備下列優點：
  - 不必進行任何設定
  - 免費使用 GPU
  - 輕鬆共用
- Colab 筆記本可讓你在單一文件中結合可執行的程式碼和 RTF 格式，並附帶圖片、HTML、LaTeX 等其他格式的內容。你建立的 Colab 筆記本會儲存到你的 Google 雲端硬碟帳戶中。你可以輕鬆將 Colab 筆記本與同事或朋友共用，讓他們在筆記本上加上註解，或甚至進行編輯。





# 使用 Google Colab 雲端運算平台

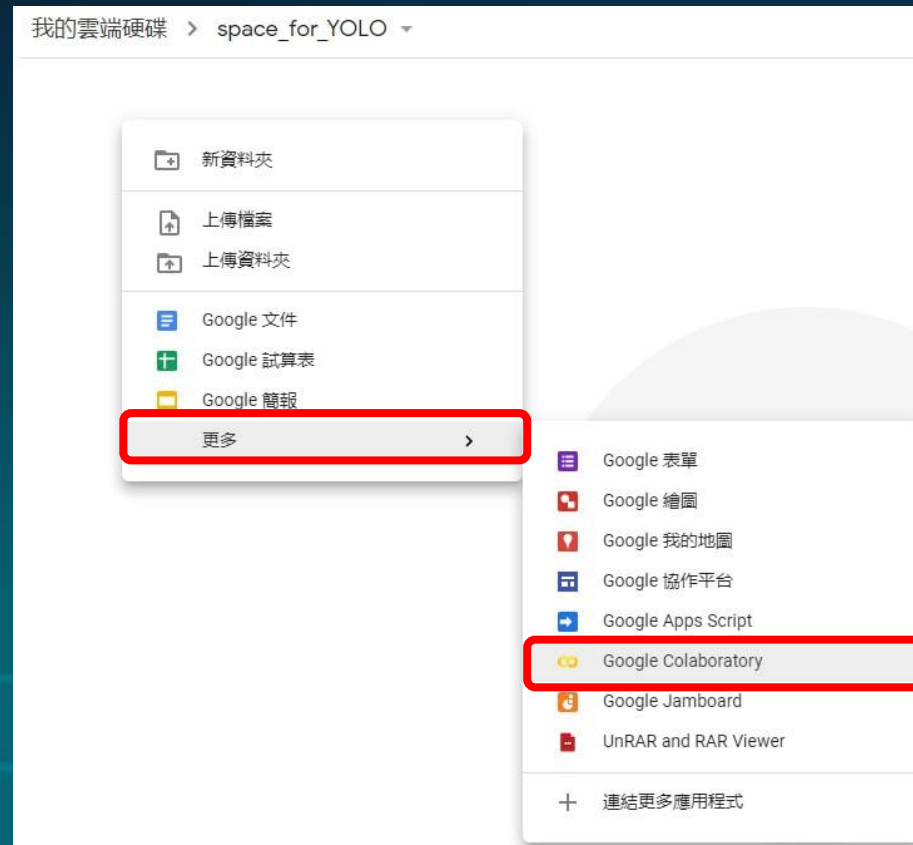
- 先在個人的 Google Drive 上新增一個資料夾。





# 使用Google Colab雲端運算平台

- 進入該資料夾後新增一個 Colab 文件

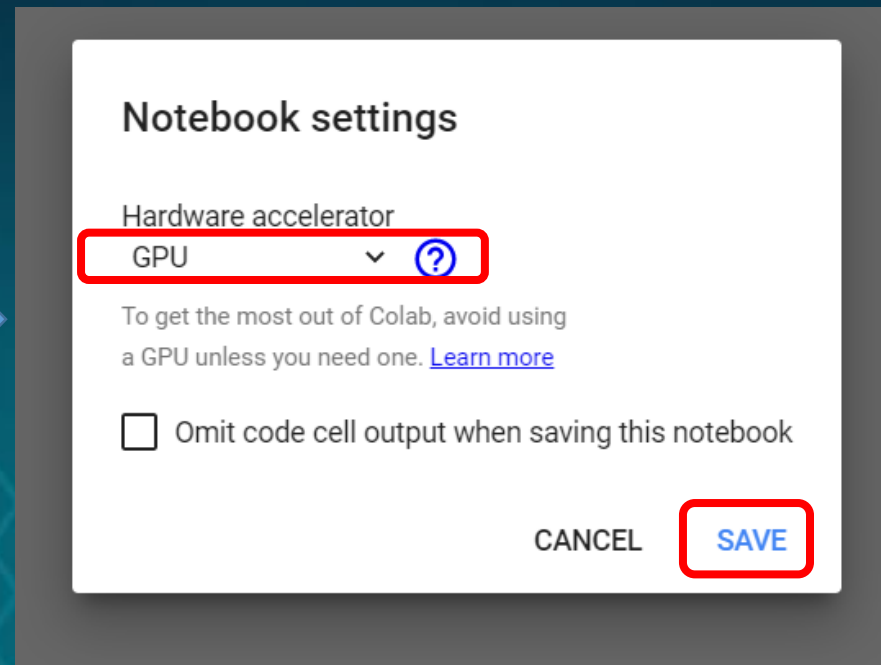
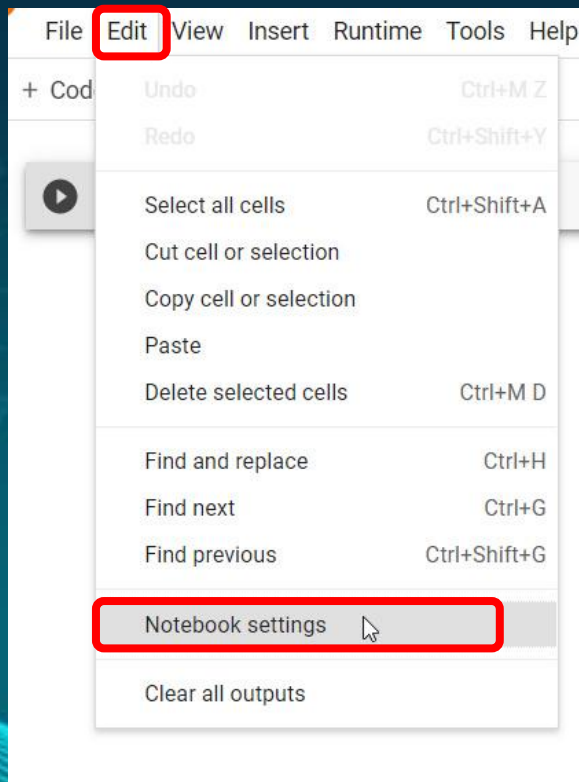






# 使用 Google Colab 雲端運算平台

- 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

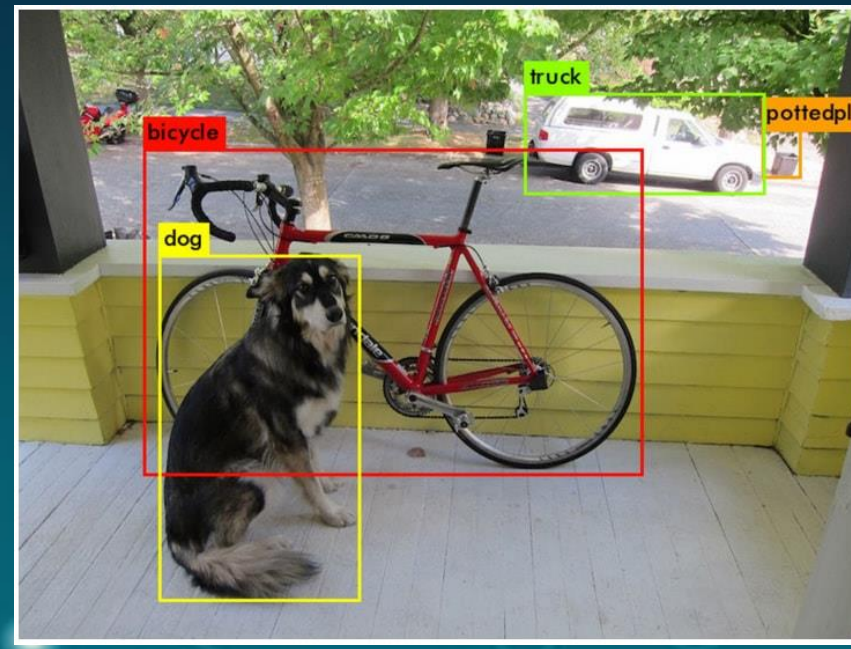
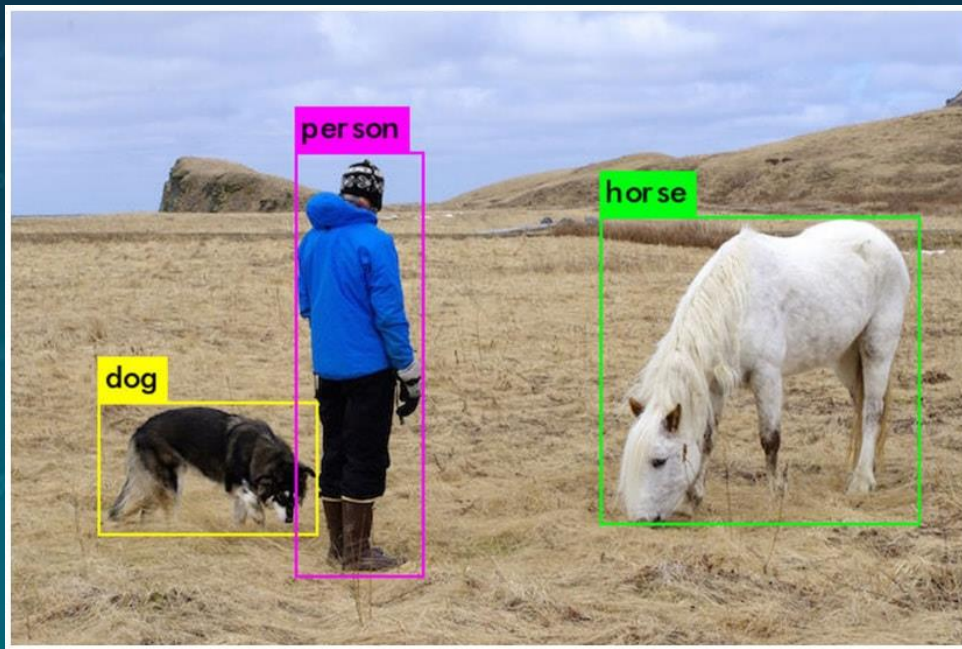
# show result
imshow('predictions.jpg')
```





# 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\_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雲盤的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= 1
2 train = data/train.txt
3 valid = data/train.txt
4 names = data/obj.names
5 backup = backup/
6
7
```

obj.data

```
1 fish
2
```

obj.names



# 使用Google Colab雲端運算平台

- 上傳yolov4\_custom.cfg到雲端：
  - yolov4\_custom.cfg在darknet/cfg/yolov4-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)

```
1 [net]
2 # Testing
3 #batch=1
4 #subdivisions=1
5 # Training
6 batch=64
7 subdivisions=32
8 width=608
9 height=608
10 channels=3
11 momentum=0.949
12 decay=0.0005
13 angle=0
14 saturation = 1.5
15 exposure = 1.5
16 hue=.1
17
18 learning_rate=0.001
19 burn in=1000
20 max_batches = 2000
21 policy=steps
22 steps=1600,1800
23 scales=.1,.1
24
```



# 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1 則filters=18
  - classes = 6 (分類個數)
  - 共要修改三個段，如下圖所示：

```
959 [convolutional]
960 size=1
961 stride=1
962 pad=1
963 filters=18
964 activation=linear
965
966 [yolo]
967 mask = 0,1,2
968 anchors = 12, 16, 19, 36, 40
969 classes=1
970 num=9
971 jitter=.3
972 ignore_thresh = .7
973 truth_thresh = 1
```

```
1047 [convolutional]
1048 size=1
1049 stride=1
1050 pad=1
1051 filters=18
1052 activation=linear
1053
1054 [yolo]
1055 mask = 3,4,5
1056 anchors = 12, 16, 19, 36,
1057 classes=1
1058 num=9
1059 jitter=.3
1060 ignore_thresh = .7
1061 truth_thresh = 1
```

```
1135 [convolutional]
1136 size=1
1137 stride=1
1138 pad=1
1139 filters=18
1140 activation=linear
1141
1142 [yolo]
1143 mask = 6,7,8
1144 anchors = 12, 16, 19, 36, 40
1145 classes=1
1146 num=9
1147 jitter=.3
1148 ignore_thresh = .7
1149 truth_thresh = 1
```





# 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

- 將訓練所需檔案上傳至雲端後，依照右側指令依序copy至Colab工作空間，全部準備完成後就可以開始訓練自己的YOLOv4模型了。

```
[ ] #確保當前路徑在darknet下
!ls
#將資料及圖片及標記的txt文件的壓縮檔上傳
!cp /mydrive/yolo_fish/obj.zip ../
#解壓縮到data文件下
!unzip ../obj.zip -d data/obj

[ ] #從google drive上傳根據自己資料集修改的.cfg文件
!cp /mydrive/yolo_fish/yolov4_custom.cfg ./cfg

[ ] #從google drive上傳自己資料集的obj.data obj.names
0
!cp /mydrive/yolo_fish/obj.data ./data
!cp /mydrive/yolo_fish/obj.name ./data

[ ] #上傳generate_train.py
!cp /mydrive/yolo_fish/generate_train.py ./

[ ] #運行在data文件下生成train.txt
!python generate_train.py
```



# 使用Google Colab雲端運算平台

- 開始訓練

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_custom.cfg yolov4_conv.137 -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

- 如果訓練時因不明原因中斷，此後想要在上次訓練權重的基礎上繼續訓練可採用以下指令

```
!./darknet detector train data/obj.data cfg/yolov4_custom.cfg backup/yolov4_custom_last.weights -dont_show
```



# 使用Google Colab雲端運算平台

```
#備份.weight檔案  
!cp ./backup/yolov4_custom_last.weights /mydrive/yolo_fish
```

```
[ ] #copy影片至darknet資料夾  
!cp /mydrive/yolo_fish/fish4.mp4 ./
```

```
[ ] #進行影片辨識  
!./darknet detector demo data/obj.data cfg/yolov4_custom.cfg backup/yolov4_custom_last.weights -dont_show fish4.mp4 -out_filename fish_out.mp4
```

進行辨識

obj.data位置

.cfg位置

訓練好的權重檔案

過程不要顯示任何視窗

src filename

output filename

```
[ ] #copy影片至雲端  
!cp ./fish_out.mp4 /mydrive/yolo_fish
```





# 使用Google Colab雲端運算平台

- YOLOv4深度學習模型辨識影片範例(以魚群辨識為例)：





# 使用 Google Colab 雲端運算平台

- 使用 Google Colab 需要注意的事項：
  - 由於 Google Colab 提供的是免費使用的運算平台，資源總是有限，因此若是訓練時間置過久會可能會主動中斷連結，這時候已執行的結果都會被清空。
  - Colab 最長的執行時間為 12 小時，但訓練 YOLO 可能長達數天以上，因此可參考下方參考資料建立一個專用的 Colab disk 空間，讓每次重新執行 Colab 時，不會遺失訓練結果，更可以很快設定好訓練環境並從上次中斷的地方繼續訓練。請參考：<https://makerpro.cc/2020/02/use-google-colab-to-train-yolo/>





# 參考資料

- LabelImg影像標記程式：<https://github.com/tzutalin/labelImg>
- YOLOv4 darknet：<https://github.com/AlexeyAB/darknet>
- Google Colab上執行Yolo教學1：  
<https://jason-chen-1992.weebly.com/home/-google-colab-yolov4>
- Google Colab上執行Yolo教學2：  
<https://blog.csdn.net/longlong068/article/details/105791941>
- Google Colab上執行Yolo教學3：  
<https://makerpro.cc/2020/02/use-google-colab-to-train-yolo/>
- YOLOv4 darknet作者提供的colab教學：  
[https://colab.research.google.com/drive/12QusaaRj\\_IUwCGDvQNfICpa7kA7\\_a2dE](https://colab.research.google.com/drive/12QusaaRj_IUwCGDvQNfICpa7kA7_a2dE)