

使用 Google Colab 訓練 YOLOv5 模組:

此份文件是參閱 Roboflow 訓練 YOLOv5 模組，詳細內容請參閱。

<https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/>

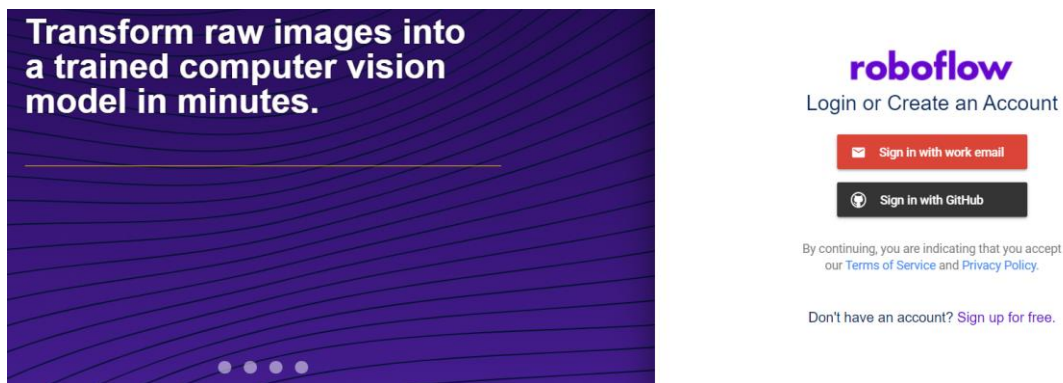
此份文件以仍未建立自己的資料數據庫進行步驟解說，不包含程式碼解說。

步驟一、建立資料庫

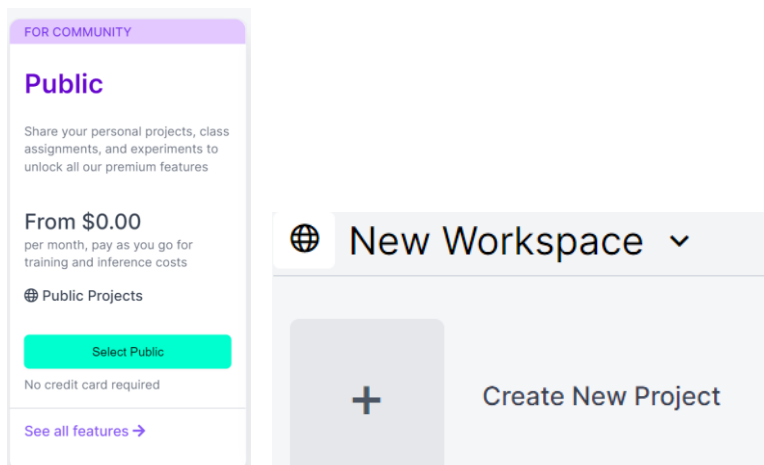
資料庫建立首先需要拍攝一系列自己所需辨識的照片，檔案需統一圖片檔(.jpg, .png, .bmp) 或 影片(.mov, .mp4, .avi)。#影片的訓練我沒用過，詳細我不是很了解。

步驟二、特徵標註

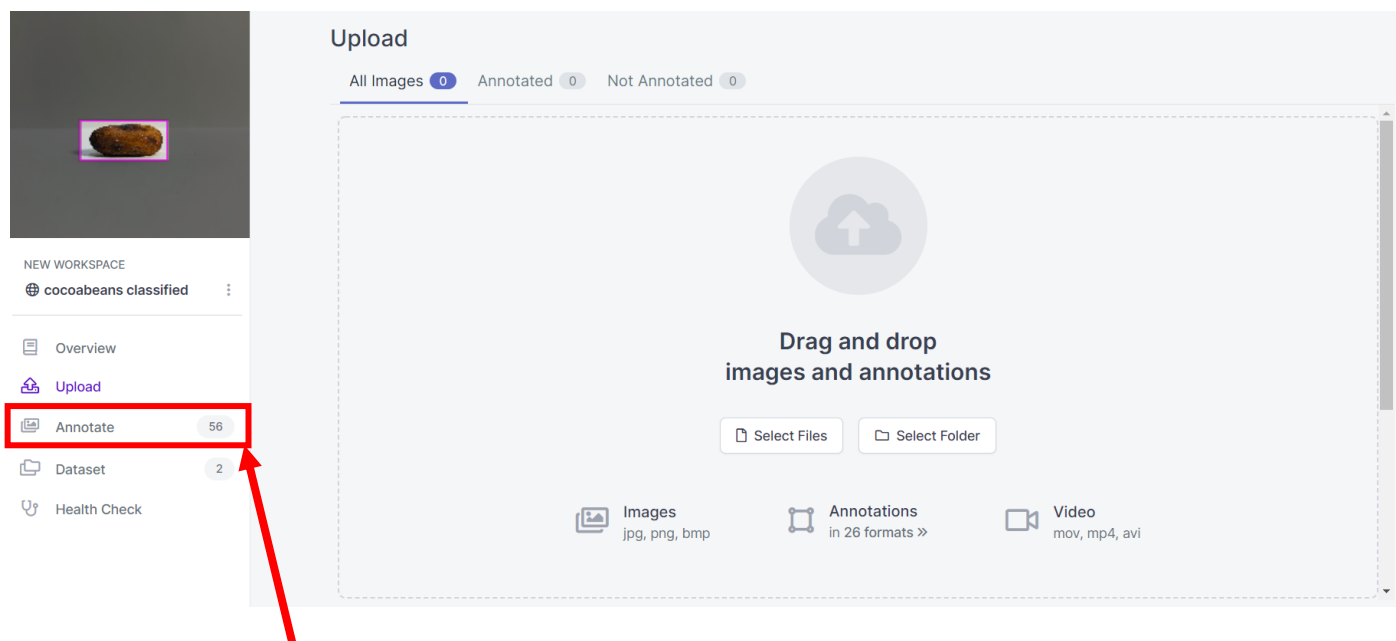
至 roboflow 申請帳號，上傳自建資料庫資料，使用免費線上標註資源。



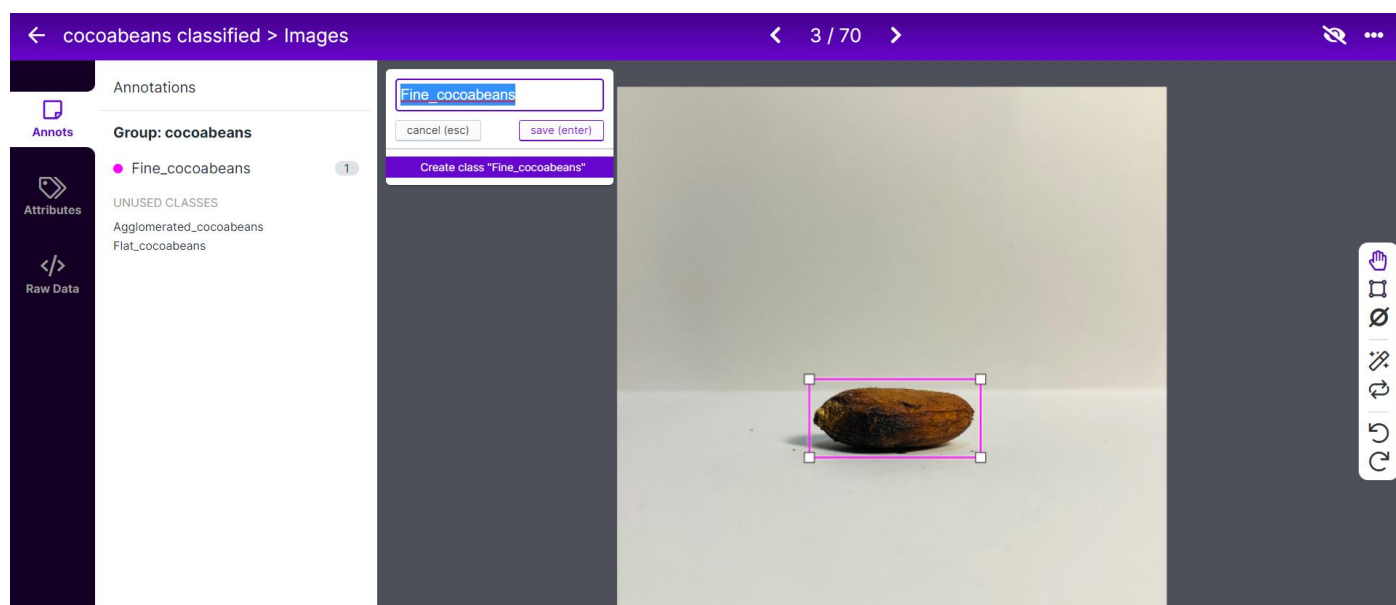
登入後選 Public，然後建立自己的 Project。



然後上傳自己的數據資料，上傳後 Roboflow 會自動規畫你的資料並分成訓練、測試及驗證圖片，當然訓練、測試及驗證圖片數量比例也可以自行調整。



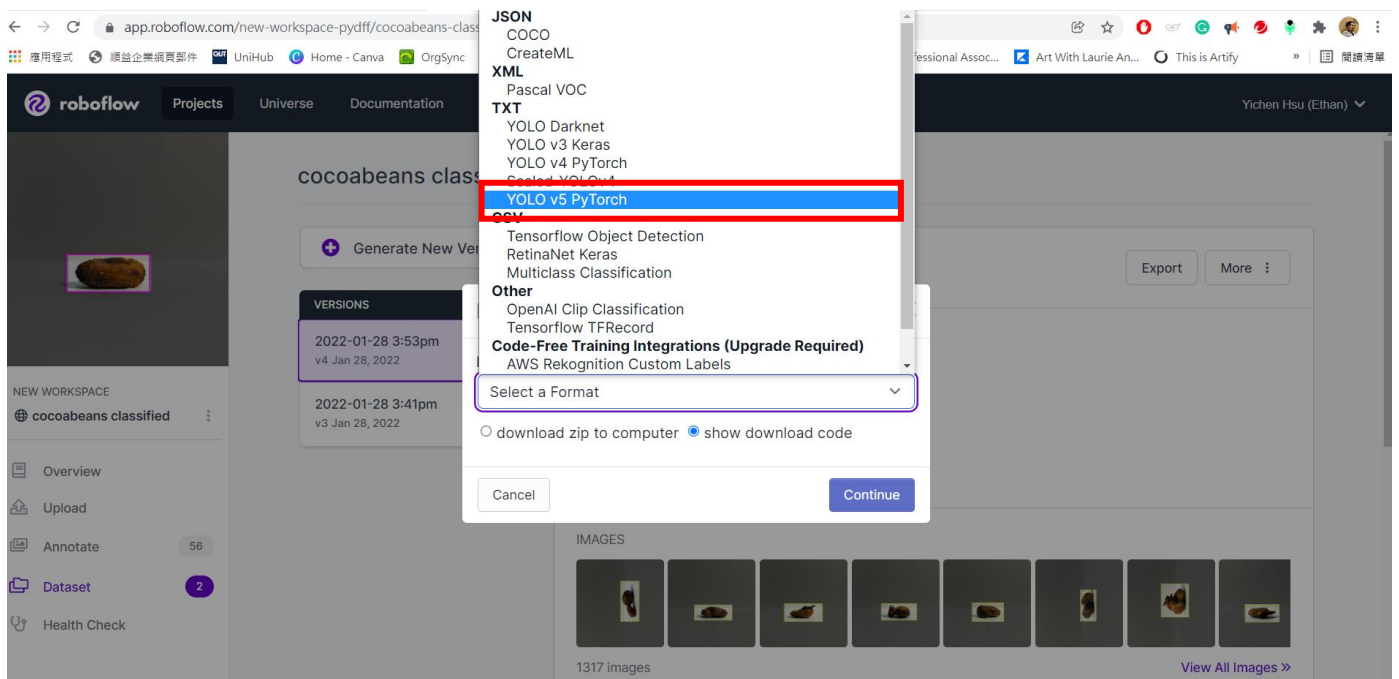
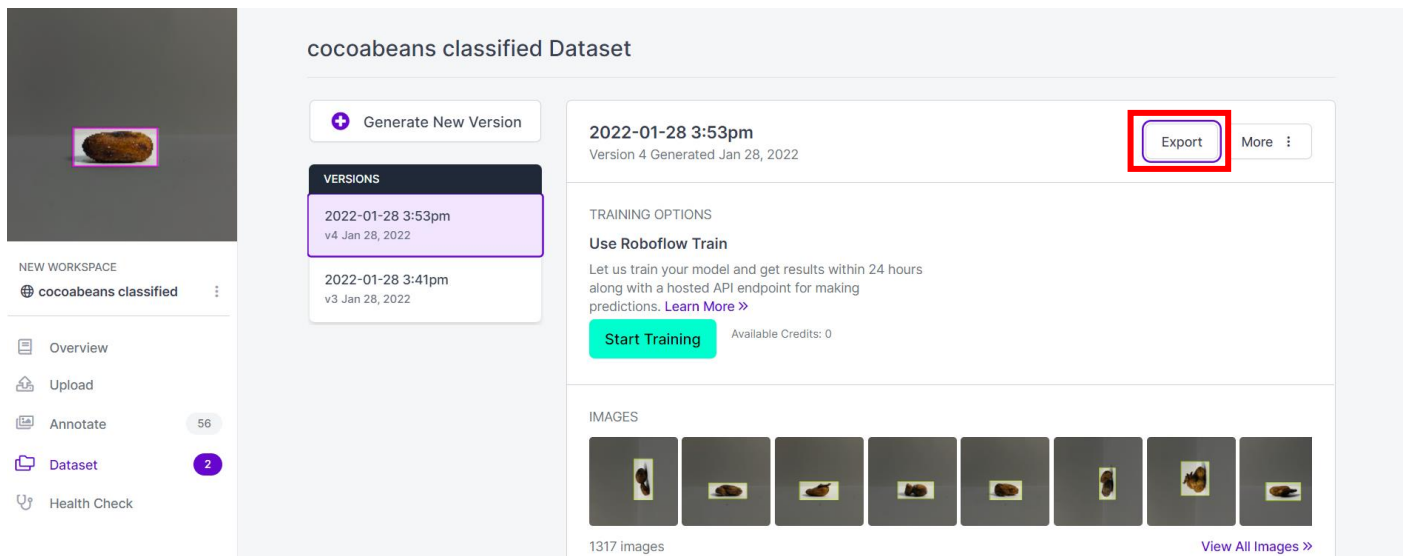
然後就可以在 Annotate 選項，開始進行特徵標註。(只要直接在圖片中拉一框框並在上面打上自己定義的標籤即可，更換圖片可以用鍵盤左右建控制。)



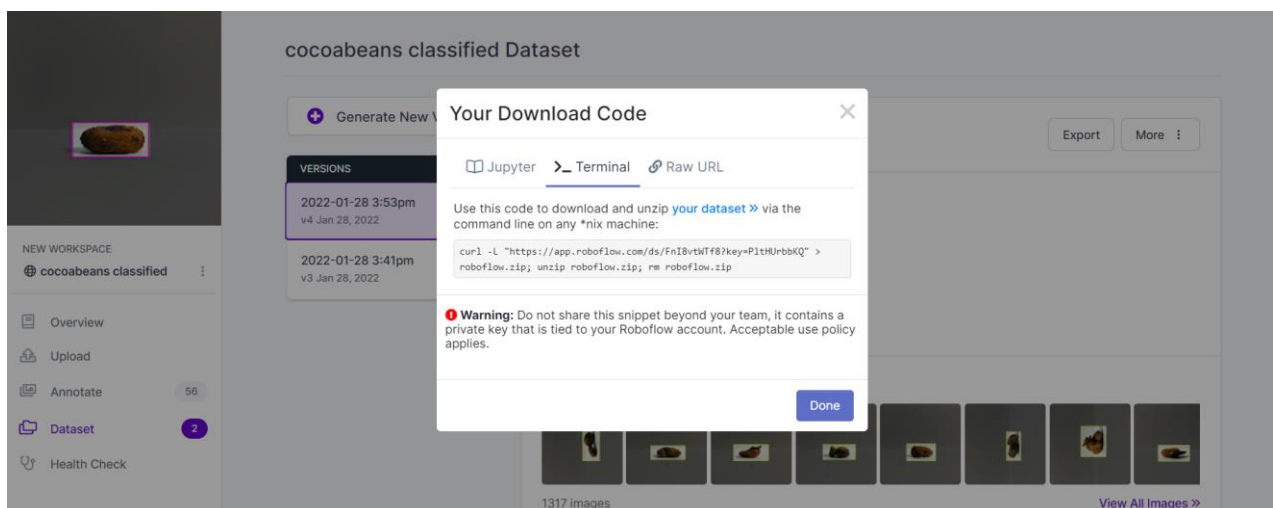
標註後框框無法去除，但可以重新定義。

步驟三、匯出標註好的檔案。

至 Dataset，按下 Export。輸出 YOLOv5 PyTorch，點選 show download code。

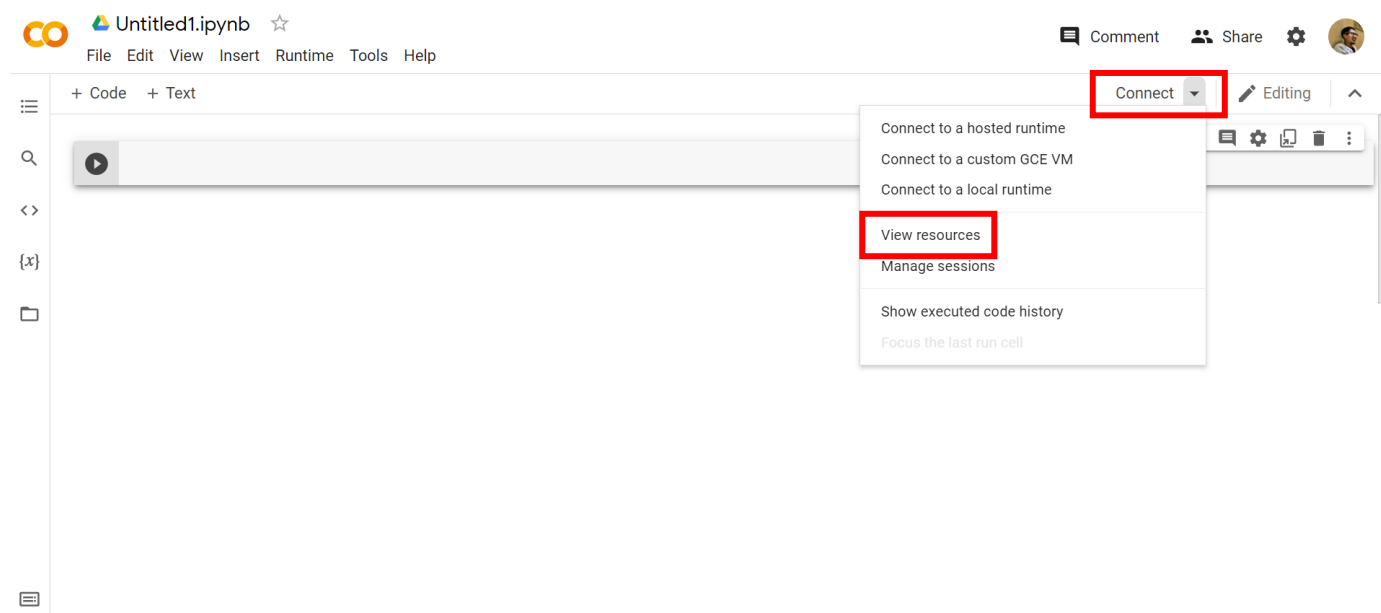


按下 continue 後，點選 Terminal，複製下面的 code，之後將貼上在 Google Colab 上以下載自己的資料檔，完成標註檔案建立！

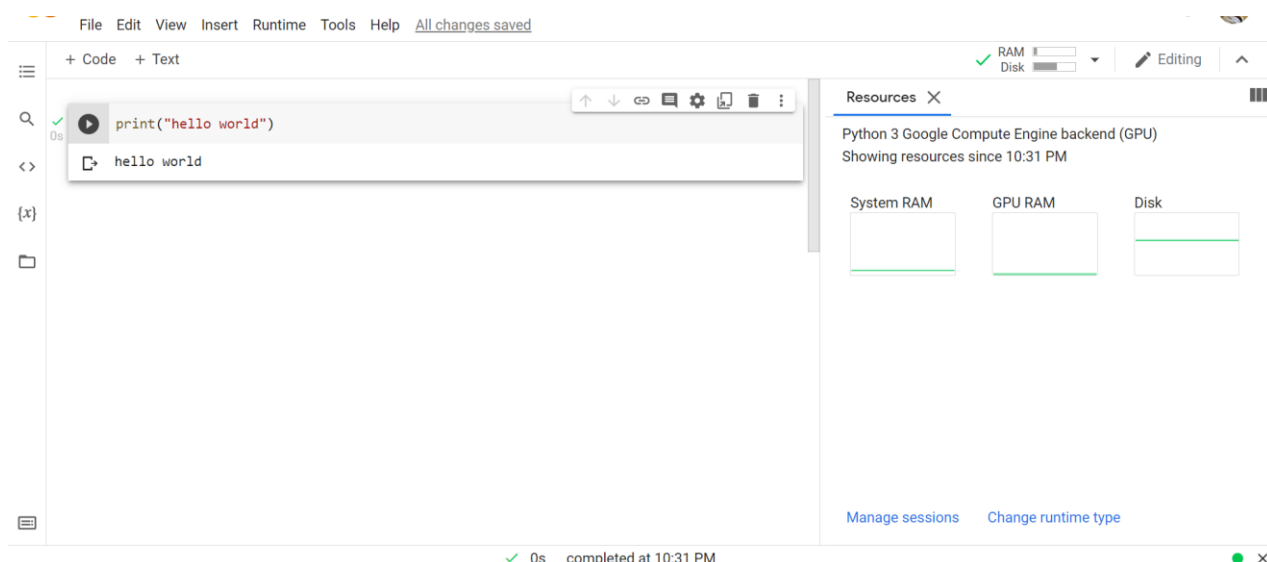
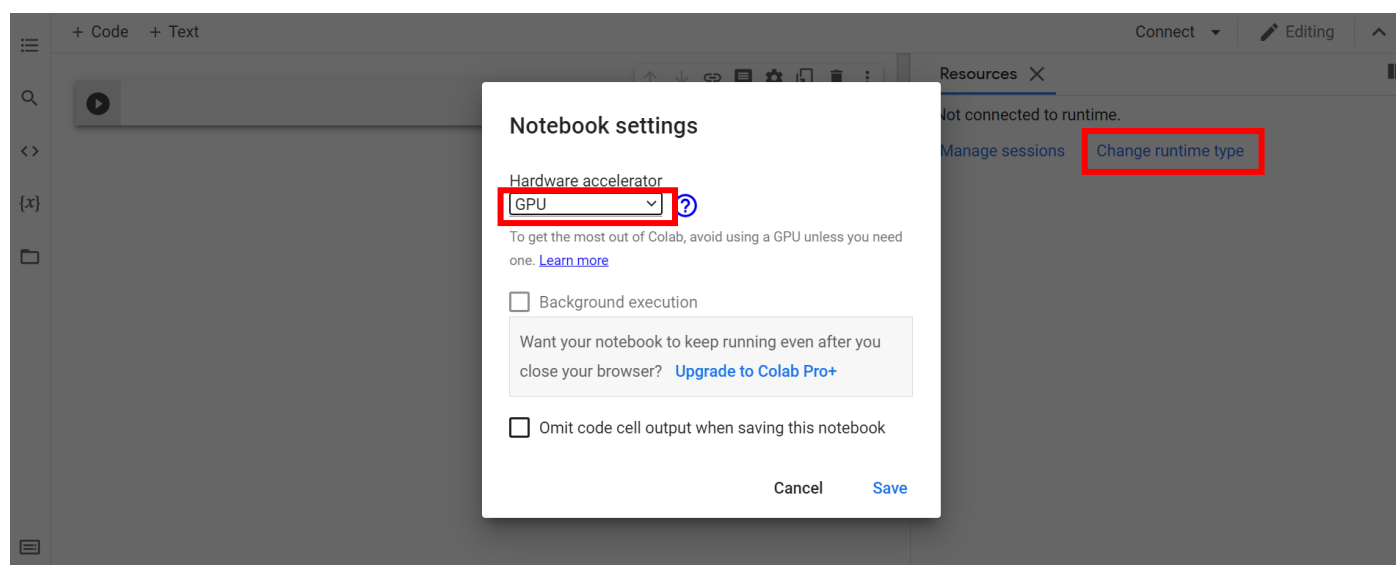


步驟四、打開 Google Colab 設定 GPU 加速。

點選連結，查看資源。



變更運行種類，使用 GPU。



步驟五、在 google colab 上安裝 yolov5 需求環境。

```
!git clone https://github.com/ultralytics/yolov5 # clone repo
!pip install -U -r yolov5/requirements.txt # install dependencies

%cd /content/yolov5
```

步驟六、輸入並下載步驟三建置好的資料檔案。

```
!curl -L "https://app.roboflow.com/ds/FnI8vtWtF8?key=PltHUrbbKQ" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

步驟七、訓練模組

(注意：訓練、測試及驗證資料須在 content 資料夾底下)

```
%%time
%cd /content/yolov5/
!python train.py --img 928 --batch 16 --epochs 100 --
data '/content/yolov5/data.yaml' --cfg ./models/yolov5s.yaml --weights '' --
name yolov5s_results --nosave --cache
```

The screenshot shows the Google Colab interface. On the left, the 'Files' pane displays a directory structure. The 'content' folder is highlighted with a red rectangle, and its subfolders are listed: 'sample_data', 'test', 'train', 'valid', and 'yolov5'. The 'yolov5' folder is further expanded, showing 'data', 'models', 'runs', and 'utils'. The main terminal area shows the execution of the training command. The output includes the command being run, the current directory, and the results of the training process, including the number of classes, the model path, and the weights path. The output also shows the hyperparameters used for training, such as learning rate, momentum, and weight decay. The terminal output is as follows:

```
with open("data.yaml", "r") as stream:
[ ] num_classes = str(yaml.safe_load(stream)['nc'])

!%%time
!python train.py --img 928 --batch 16 --epochs 1 --data '/content/yolov5/data.yaml' --cfg ./models/yolov5s.yaml --weights '' --name yolov5s_results --nosave --cache

/content/yolov5
train: weights=, cfg=./models/yolov5m.yaml, data=/content/yolov5/data.yaml, hyp=data/hyps/hyp.scratch.yaml
github: up to date with https://github.com/ultralytics/yolov5 ✓
YOLOv5 v6.0-218-g7539cd7 torch 1.10.2+cu102 CUDA:0 (Tesla K80, 11441MiB)

hyperparameters: lr0=0.01, lrf=0.1, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_moment
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 runs (RECOMMEND
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=3

from n params module arguments
0 -1 1 5280 models.common.Conv [3, 48, 6, 2, 2]
1 -1 1 41664 models.common.Conv [48, 96, 3, 2]
2 -1 2 65280 models.common.C3 [96, 96, 2]
3 -1 1 166272 models.common.Conv [96, 192, 3, 2]
4 -1 4 444672 models.common.C3 [192, 192, 4]
5 -1 1 664320 models.common.Conv [192, 384, 3, 2]
```

步驟八、儲存訓練權重(weights)

訓練完後的模組權重會儲存在檔案夾 runs 裡面。

```
▼ yolov5
  ▸ data
  ▸ models
  ▼ runs
    ▼ train
      ▼ yolov5s_results
        ▸ weights
        ▸ events.out.tfevent...
        ▸ hyp.yaml
        ▸ opt.yaml
```

```
train: Caching images (2.9GB ram): 100% 1133/1133 [00:13<00:00, 86.39it/s]
val: Scanning '../valid/labels' images and labels...123 found, 0 missing, 3 empty, 0 corrupt: 100% 123/
val: New cache created: ../valid/labels.cache
val: Caching images (0.3GB ram): 100% 123/123 [00:02<00:00, 46.30it/s]
Plotting labels to runs/train/yolov5s_results2/labels.jpg...

AutoAnchor: 3.42 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to da
Image sizes 928 train, 928 val
Using 2 dataloader workers
Logging results to runs/train/yolov5s_results2
Starting training for 1 epochs...

Epoch  gpu_mem    box    obj    cls  labels  img_size
0/0      11.2G    0.1214 0.0416 0.04362    29      928:  13% 9/71 [00:51<05:39,  5.47s/
```

步驟九、使用 Tensorboard 查看訓練歷史過程

```
%load_ext tensorboard
%tensorboard --logdir runs
```

以上完成在 Google Colab 訓練 YOLOv5 自定義資料庫的過程。