***Training Model***

[*Train yolo v3 model*](#Train_YoloV3)

*[Train SSD MobileNet v1 model](#Train_SSD_MobileNet_Model)*

***Android APP***

[*Tensorflow demo code based on Android Studio*](#Android_Tensorflow)

[*Yolo v1/v2 demo code based on Android Studio*](#Android_yolov1v2)

[*Object detection of pepper based on Android Studio*](#pepper_object_detection)

***Tensorflow Packages***

[*Tensorflow優化模型套件*](#Tensorflow_OptModel)

***Model Conversion***

[*Convert model from \*.h5 to \*.pb*](#Convert_Model_h5_to_pb)

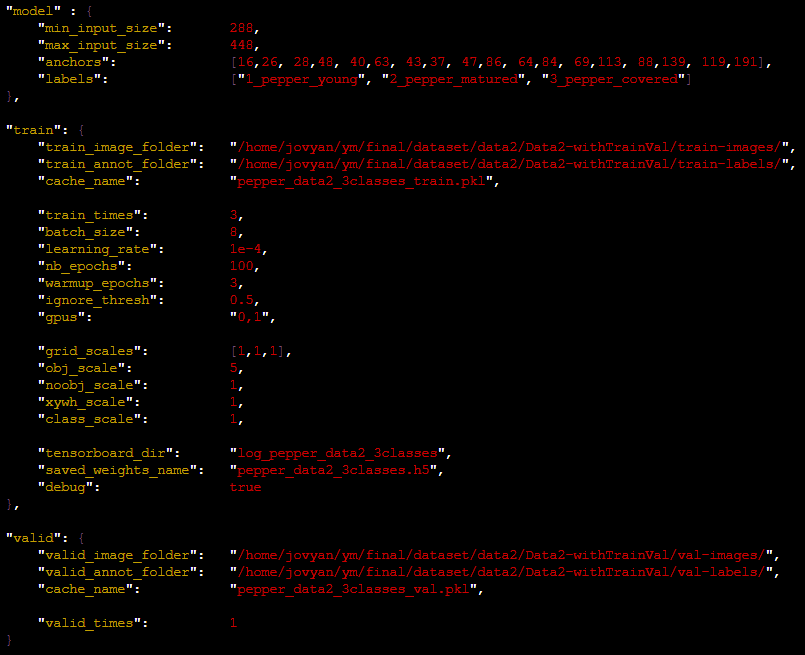
***Train yolo v3 model***

**環境需求: Windows or Ubuntu** (此範例在Ubuntu)

1. 解壓縮程式碼keras-yolo3.tar.gz

tar zxvf keras-yolo3.tar.gz

2. 配置keras-yolo/zoo/config\_pepper-data2-3classes-TrainVal.json文件



3. 訓練指令

cd keras-yolo3

python train.py -c zoo/config\_pepper-data2-3classes-TrainVal.json

4. 評估指令

python evaluate.py -c zoo/config\_pepper-data2-3classes-TrainVal.json

5. 辨識MP4的方式

python predict.py -c zoo/config\_pepper-data2-3classes-TrainVal.json -i <TO\_YOUR\_PATH>/video.mp4

***References***

1. 訓練keras-yolo3的方式

<https://github.com/experiencor/keras-yolo3>

***Train SSD MobileNet v1 model***

**環境需求: Windows or Ubuntu** (此範例兩個都有用到)

1. 生成Tensorflow模型可支援的TFRecords輸入格式 (Windows環境)

1-1. 先轉成CSV; 資料結構如下, 分為images及labels資料夾, 訓練和評估資料先放一起

1-2. 修改xml\_to\_csv.py的參數

#label的資料夾路徑

image\_path = r'C:\labels'

#輸出CSV名稱

outputCSV = r'outputCSV\pepper\_labels.csv'

1-3. 執行轉csv程式

開啟cmd

cd <TO\_YOUR\_PATH>\xml2tfrecord

python xml\_to\_csv.py

成功會看到以下訊息



1-4. 分割訓練/評估資料

使用jupyter notebook開啟split labels.ipynb直接執行

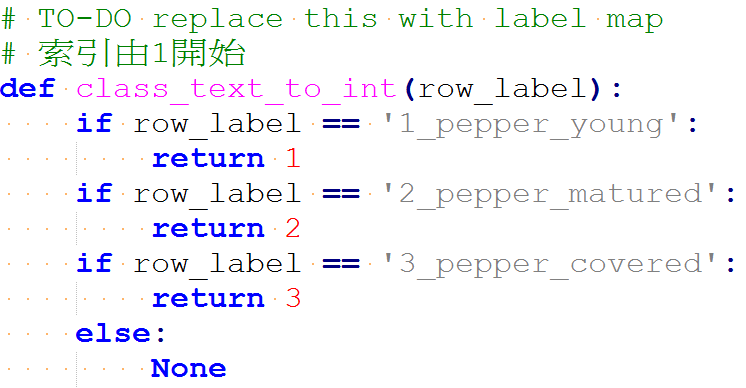
Note: 預設評估的圖片張數為圖片總數的20%

1-5. 再由分割後的CSV轉為TFRecords; 修改generate\_tfrecord.py兩個部份

# 圖片資料夾路徑

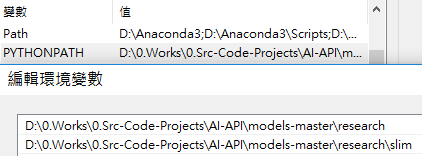
path = r'C:\images'

#定義辨識種類索引



1-6. 下載Tensorflow models程式碼, 並設定環境變數

<https://github.com/tensorflow/models.git>



1-7. 執行轉TFRecords程式

python generate\_tfrecord.py --csv\_input=outputCSV\train\_labels.csv --output\_path=outputTF\train.record

python generate\_tfrecord.py --csv\_input=outputCSV\test\_labels.csv --output\_path=outputTF\test.record

成功後會顯示以下訊息



2. 下載Tensorflow source code, 並配置環境變數(以下都在Ubuntu, 也可以在Windows, 自行修改路徑即可)

git clone <https://github.com/tensorflow/models.git>

cd <TO\_YOUR\_PATH>/models/research

export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim

Note: Windows參照1-6步驟

3. 下載預訓練模型, 並配置config文件

3-1. 進入網頁

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

3-2. 下載ssd\_mobilenet\_v1\_coco預訓練模型並解壓縮

下載後的檔名: ssd\_mobilenet\_v1\_coco\_2018\_01\_28.tar.gz

3-3. 配置config文件(必改部分)

num\_classes: 4

辨識種類的數量

fine\_tune\_checkpoint: "<TO\_YOUR\_PATH>/ssd\_mobilenet\_v1\_coco\_2018\_01\_28/model.ckpt"

下載的預訓練模型路徑

train\_input\_reader {

label\_map\_path: "<TO\_YOUR\_PATH>/pepper\_label\_map.pbtxt"

定義彩椒標記種類

tf\_record\_input\_reader {

input\_path: "<TO\_YOUR\_PATH>/train.record"

Tensorflow格式的訓練資料

}

}

eval\_config {

num\_examples: 125

評估指令會用到的部分, 評估圖片的張數(非標記的總數)

max\_evals: 10

use\_moving\_averages: false

}

eval\_input\_reader {

label\_map\_path: "<TO\_YOUR\_PATH>/pepper\_label\_map.pbtxt"

定義彩椒標記種類

shuffle: false

num\_readers: 1

tf\_record\_input\_reader {

input\_path: "<TO\_YOUR\_PATH>/test.record"

Tensorflow格式的驗證資料

}

}

4. 利用Tensorflow object detection API進行訓練

cd <TO\_YOUR\_PATH>/models/research

python object\_detection/legacy/train.py --logtostderr --pipeline\_config\_path=<TO\_YOUR\_PATH>/ssd\_mobilenet\_v1\_coco\_2018\_01\_28/pipeline.config --train\_dir=<TO\_YOUR\_PATH>/train-mobilenet-v1\_log

自動生成資料夾

5. 評估指令

#安裝相依套件

pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI

python object\_detection/legacy/eval.py --logtostderr --pipeline\_config\_path=<TO\_YOUR\_PATH>/ssd\_mobilenet\_v1\_coco\_2018\_01\_28/pipeline.config --checkpoint\_dir=<TO\_YOUR\_PATH>/train-mobilenet-v1\_log/ --eval\_dir=<TO\_YOUR\_PATH>/eval-mobilenet-v1\_log

自動生成資料夾

6. 將訓練完成的.ckpt模型格式轉為Android支援的.pb格式, 移植到APP請[參考](#pepper_object_detection)

python object\_detection/export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path=<TO\_YOUR\_PATH>/ssd\_mobilenet\_v1\_pepper.config --trained\_checkpoint\_prefix=<TO\_YOUR\_PATH>/model.ckpt-<STEP\_NUMBER> --output\_directory=<TO\_YOUR\_PATH>/output\_inference\_graph.pb

匯出檔名

選擇第幾個STEP模型匯出

***Tensorflow demo code based on Android Studio***

**環境及軟體需求:**

**Windows或Ubuntu** (此範例在Windows)

**Android Studio**

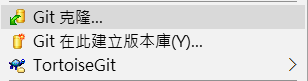
**Git & TortoiseGit**

1. 安裝 Android Studio

參考官網安裝即可

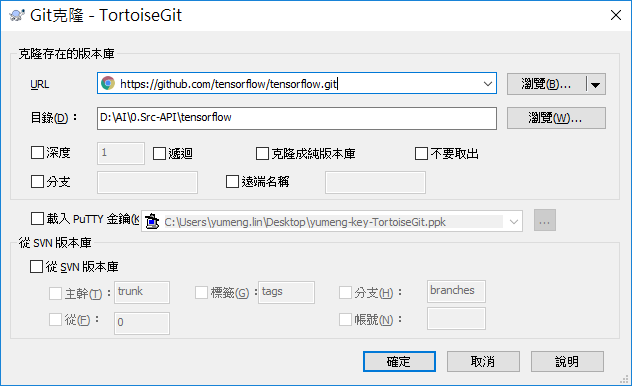
2. 下載Tensorflow source code

在個人工作目錄按右鍵, 點擊Git克隆



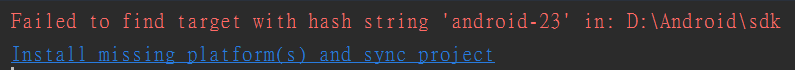
輸入URL下載位置, 點擊確定

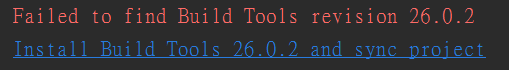
<https://github.com/tensorflow/tensorflow.git>

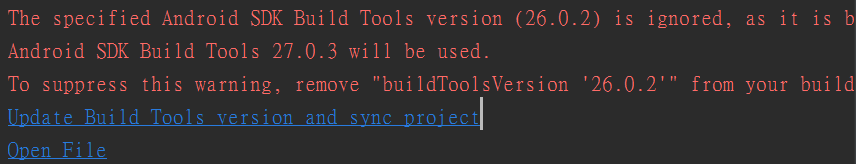


3. 使用Android Studio開啟<TO\_YOUR\_PATH>\tensorflow\examples\android專案

按照 Android Studio 提示下載安裝缺乏的套件包, 例如







開啟 build.gradle 文件, 修改如下

// set to 'bazel', 'cmake', 'makefile', 'none'

def nativeBuildSystem = 'none'

4. 找一隻Android 中高階手機, 執行範例程式

Object Detector有三種模式選擇(TF\_OD\_API, MULTIBOX, YOLO), 預設使用TF\_OD\_API模式; 如要使用yolo模式, 參考[連結](#Android_yolov1v2)

***References***

1. Building TensorFlow on Android

<https://www.tensorflow.org/lite/tfmobile/android_build>

***Yolo v1/v2 demo code based on Android Studio***

***(Yolo v3 目前不支援weight轉pb)***

1. 將yolo v1/v2 .weight模型轉換為tensorflow .pb模型 (需使用Ubuntu)

1-1. 下載轉換工具並編譯安裝

git clone https://github.com/pjreddie/darknet

git clone https://github.com/thtrieu/darkflow

cd darkflow

sudo pip3 install Cython

python3 setup.py build\_ext --inplace

sudo pip install -e .

1-2. 下載yolo cfg & weights (建議用點擊下載, 不要使用wget, 否則有可能是html格式)或者壓縮包裡面有

yolo.weight: <https://pjreddie.com/media/files/yolo.weights>

tiny-yolo-voc-graph.pb: <https://github.com/szaza/android-yolo-v2/blob/master/assets/tiny-yolo-voc-graph.pb>

1-3. 轉換指令

yolo v1 example:

flow --model <TO\_YOUR\_PATH>/tiny-yolo-voc.cfg --load <TO\_YOUR\_PATH>/tiny-yolo-voc.weights --savepb

yolo v2 example:

flow --model <TO\_YOUR\_PATH>/yolov2-tiny.cfg --load <TO\_YOUR\_PATH>/yolov2-tiny.weights --labels <TO\_YOUR\_PATH>/coco-labels-2014\_2017.txt --savepb

2. 將模型.pb移植到Android APP

2-1. 將tiny-yolo-voc.pb放到<TO\_YOUR\_PATH>\tensorflow\examples\android\assets\底下(Windows or Ubuntu環境都可以)

2-2. 修改DetectorActivity.java後, 執行APP

private static final DetectorMode MODE = DetectorMode.YOLO;

private static final String YOLO\_MODEL\_FILE = "file:///android\_asset/tiny-yolo-voc.pb";

***Object detection of pepper based on Android Studio***

1. 將期末專題資料夾裡面的Android source code解壓縮

2. 使用Android Studio開啟專案, 路徑避免中文, 執行連結手機直行即可

預設使用ssd\_mobilenet\_v2模型分3類彩椒結果, 如果要更改模型或標記, 請參照以下部分修改

a) 將模型及標記文件放到assets資料夾下

pepper-ssdv2-3classes.pb

pepper-label\_3classes.txt

b) 修改模型及標記文件路徑

private static final String TF\_OD\_API\_MODEL\_FILE = "file:///android\_asset/pepper-ssdv2-3classes.pb";

private static final String TF\_OD\_API\_LABELS\_FILE = "file:///android\_asset/pepper-label\_3classes.txt";

***Tensorflow優化模型套件***

**環境需求: Ubuntu**

1. 下載Tensorflow source code

git clone --recurse-submodules <https://github.com/tensorflow/tensorflow>

--recurse-submodules 引數必須要加, 用於獲取 Tensorflow 依賴的 protobuf 庫

2. 安裝Bazel編譯工具

#安裝Bazel依賴庫

sudo apt-get install openjdk-8-jdk openjdk-8-source

sudo apt-get install pkg-config zip zlib1g-dev unzip

#下載安裝Bazel

wget <https://github.com/bazelbuild/bazel/releases/download/0.19.1/bazel-0.19.1-installer-linux-x86_64.sh>

chmod +x bazel-0.19.1-installer-linux-x86\_64.sh

./bazel-0.19.1-installer-linux-x86\_64.sh --user

#配置環境變數

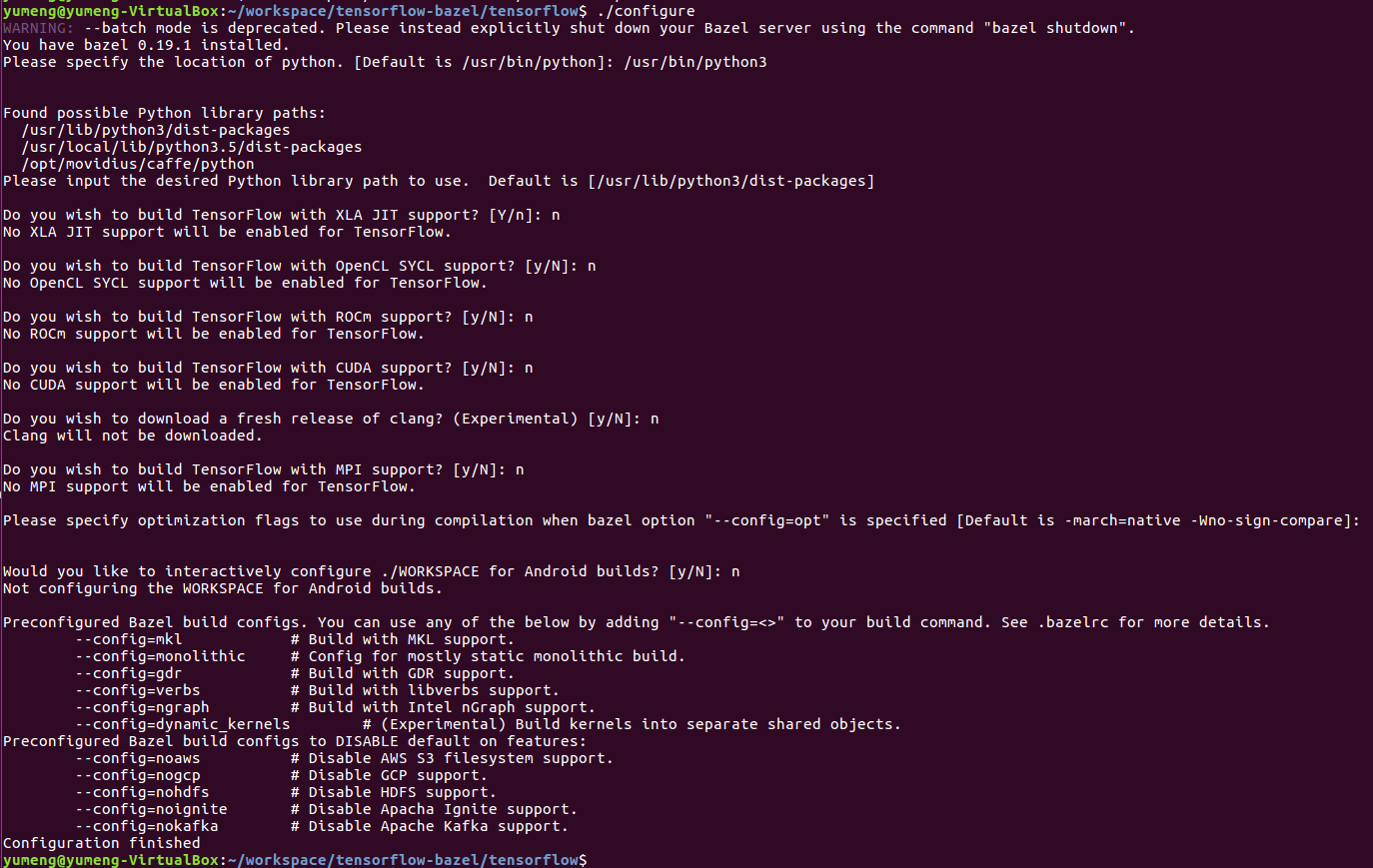
echo "export PATH=\"$PATH:$HOME/bin\"" >> ~/.bashrc

3. 編譯Tensorflow優化套件

#配置編譯參數

cd tensorflow

./configure



輸入python3的執行路徑

以下的選項都輸入n即可

#擴充swap內存, 才能成功編譯Tensorflow套件

sudo dd if=/dev/zero of=/mnt/4096Mb.swap bs=1M count=4096 #4G容量

sudo mkswap /mnt/4096Mb.swap

sudo swapon /mnt/4096Mb.swap

free -m #查看swap狀態

sudo swapoff swapfile #不用時可以關閉

#編譯優化模型相關套件

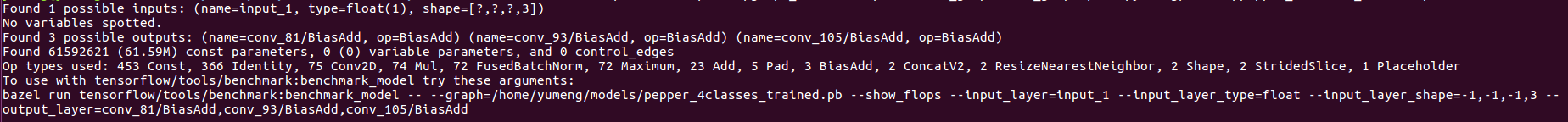
bazel build tensorflow/tools/graph\_transforms:summarize\_graph #2m

bazel build tensorflow/tools/graph\_transforms:transform\_graph #40m

bazel build tensorflow/tools/quantization:quantize\_graph #網友實驗優化差別不大, 建議使用transform\_graph方法

4. 利用summarize\_graph確定input/output name (下一個優化步驟需要使用)

bazel-bin/tensorflow/tools/graph\_transforms/summarize\_graph --in\_graph=<TO\_YOUR\_PATH>/pepper.pb



5. 利用transform\_graph優化模型

Test command 1: 測試OK

bazel-bin/tensorflow/tools/graph\_transforms/transform\_graph --in\_graph=<TO\_YOUR\_PATH>/pepper.pb --out\_graph=<TO\_YOUR\_PATH>/optimized\_pepper.pb --inputs='input\_1' --outputs='conv\_105/BiasAdd' --transforms='quantize\_weights'

Test command 2: (沒測過)

bazel-bin/tensorflow/tools/graph\_transforms/transform\_graph --in\_graph=<TO\_YOUR\_PATH>/pepper.pb --out\_graph=<TO\_YOUR\_PATH>/optimized\_pepper.pb --inputs='input\_1' --outputs='conv\_105/BiasAdd' --transforms='strip\_unused\_nodes(type=float, shape="256\*64")

remove\_nodes(op=Identity, op=CheckNumerics)

fold\_constants(ignore\_errors=true)

fold\_batch\_norms

fold\_old\_batch\_norms'

***References***

1. Bazel & tensorflow版本對照

<https://www.tensorflow.org/install/source>

2. How to install Bazel

<https://docs.bazel.build/versions/master/install-ubuntu.html>

3. Tensorflow模型量化壓縮

<https://blog.csdn.net/xygl2009/article/details/80596392>

***Convert model from \*.h5 to \*.pb***

**環境需求: Windows or Ubuntu** (此範例在Windows)

1. 下載轉換程式碼

<https://github.com/amir-abdi/keras_to_tensorflow>

2. 轉換指令

開啟cmd

cd <TO\_YOUR\_PATH>\keras\_to\_tensorflow

Test command 1: 包含weight參數及模型架構的方式

python keras\_to\_tensorflow.py --input\_model=<TO\_YOUR\_PATH>\model.h5" --output\_model=<TO\_YOUR\_PATH>\model.pb

Test command 2: 針對weight參數及模型架構分開的方式

python keras\_to\_tensorflow.py --input\_model=<TO\_YOUR\_PATH>\model.h5 --input\_model\_json=<TO\_YOUR\_PATH>\model.json --output\_model=<TO\_YOUR\_PATH>\model.pb