

Training Model

[*Train yolo v3 model*](#)

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

Android APP

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

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

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

Tensorflow Packages

[*Tensorflow 優化模型套件*](#)

Model Conversion

[*Convert model from *.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

```
"model": {
  "min_input_size": 288,
  "max_input_size": 448,
  "anchors": [16,26, 28,48, 40,63, 43,37, 47,86, 64,84, 69,113, 88,139, 119,191],
  "labels": ["1_pepper_young", "2_pepper_matured", "3_pepper_covered"]
},
"train": {
  "train_image_folder": "/home/jovyan/ym/final/dataset/data2/Data2-withTrainVal/train-images/",
  "train_annot_folder": "/home/jovyan/ym/final/dataset/data2/Data2-withTrainVal/train-labels/",
  "cache_name": "pepper_data2_3classes_train.pkl",

  "train_times": 3,
  "batch_size": 8,
  "learning_rate": 1e-4,
  "nb_epochs": 100,
  "warmup_epochs": 3,
  "ignore_thresh": 0.5,
  "gpus": "0,1",

  "grid_scales": [1,1,1],
  "obj_scale": 5,
  "noobj_scale": 1,
  "xywh_scale": 1,
  "class_scale": 1,

  "tensorboard_dir": "log_pepper_data2_3classes",
  "saved_weights_name": "pepper_data2_3classes.h5",
  "debug": true
},
"valid": {
  "valid_image_folder": "/home/jovyan/ym/final/dataset/data2/Data2-withTrainVal/val-images/",
  "valid_annot_folder": "/home/jovyan/ym/final/dataset/data2/Data2-withTrainVal/val-labels/",
  "cache_name": "pepper_data2_3classes_val.pkl",

  "valid_times": 1
}
```

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)

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

images						labels			
						名稱	修改日期	類型	大小
						20181111_141821.xml	2018/11/21 ...	XML Docum...	1 KB
						20181111_141845.xml	2018/11/18 ...	XML Docum...	1 KB
						20181111_141858.xml	2018/11/18 ...	XML Docum...	1 KB
						20181111_141909.xml	2018/11/21 ...	XML Docum...	2 KB
						20181111_142035.xml	2018/11/21 ...	XML Docum...	2 KB
						20181111_142046.xml	2018/11/18 ...	XML Docum...	2 KB
						20181111_142058.xml	2018/11/18 ...	XML Docum...	2 KB
						20181111_142112.xml	2018/11/18 ...	XML Docum...	1 KB
						20181111_142125.xml	2018/11/18 ...	XML Docum...	1 KB
						20181111_142139.xml	2018/11/18 ...	XML Docum...	1 KB
						20181111_142157.xml	2018/11/18 ...	XML Docum...	1 KB

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

```
Successfully converted xml to csv.
```

1-4. 分割訓練/評估資料

使用 jupyter notebook 開啟 `split_labels.ipynb` 直接執行

Note1: 尚未整合, 所以需使用 jupyter notebook 執行

Note2: 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

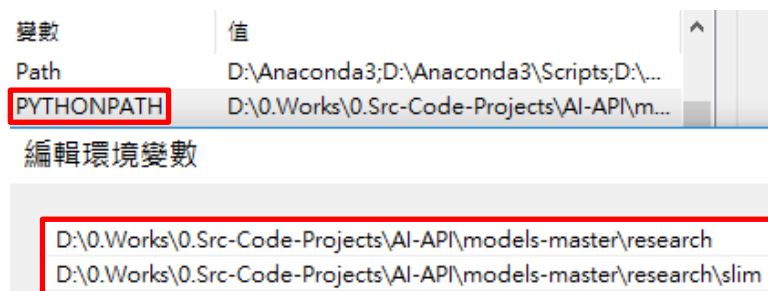
```
path = r'C:\images'
```

#定義辨識種類索引

```
# TO-DO: replace this with label map
# 索引由1開始
def class_text_to_int(row_label):
    if row_label == '1_pepper_young':
        return 1
    if row_label == '2_pepper_matured':
        return 2
    if row_label == '3_pepper_covered':
        return 3
    else:
        return None
```

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

```
Successfully created the TFRecords
```

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](#)

```
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

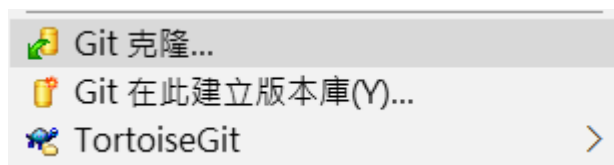
Git & TortoiseGit

1. 安裝 Android Studio

參考官網安裝即可

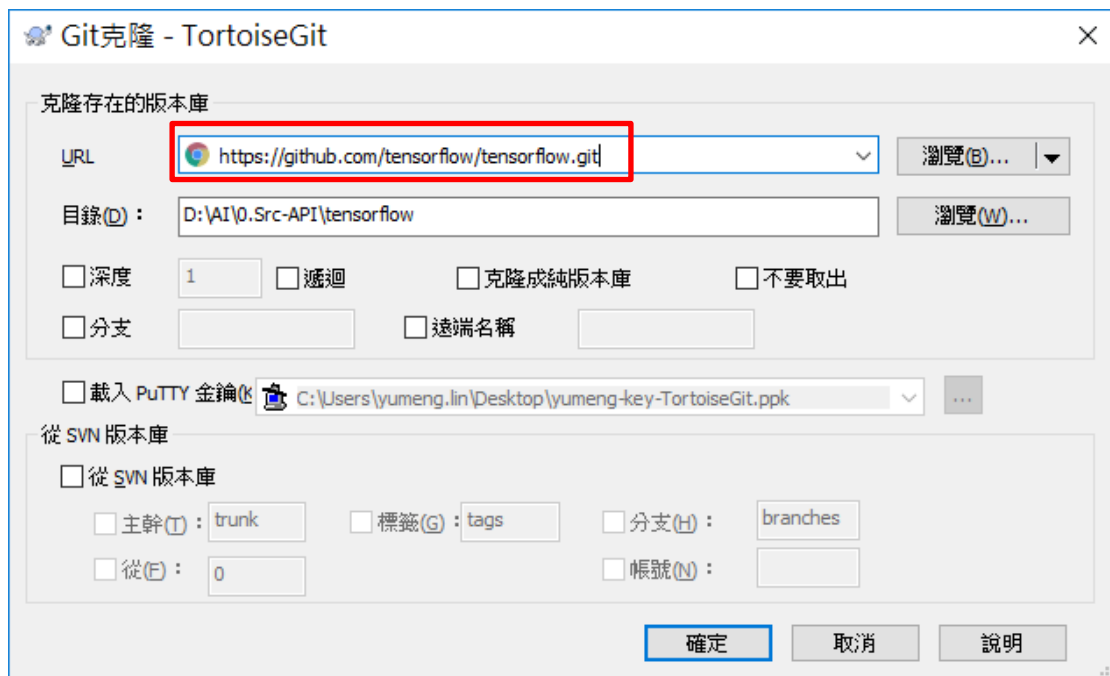
2. 下載 Tensorflow source code

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


```
Failed to find target with hash string 'android-23' in: D:\Android\sdk
Install missing platform(s) and sync project
```

```
Failed to find Build Tools revision 26.0.2
Install Build Tools 26.0.2 and sync project
```

```
The specified Android SDK Build Tools version (26.0.2) is ignored, as it is b
Android SDK Build Tools 27.0.3 will be used.
To suppress this warning, remove "buildToolsVersion '26.0.2'" from your build
Update Build Tools version and sync project
Open File
```

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](#)

References

1. Building TensorFlow on Android

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

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";
```

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";
```

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
```

```

yumeng@yumeng-VirtualBox:~/workspace/tensorflow-bazel/tensorflow$ ./configure
WARNING: --batch mode is deprecated. Please instead explicitly shut down your Bazel server using the command "bazel shutdown".
You have bazel 0.19.1 installed.
Please specify the location of python. [Default is /usr/bin/python] /usr/bin/python3
Found possible Python library paths:
/usr/lib/python3/dist-packages
/usr/local/lib/python3.5/dist-packages
/opt/novidius/caffe/python
Please input the desired Python library path to use. Default is [/usr/lib/python3/dist-packages]
Do you wish to build TensorFlow with XLA JIT support? [Y/n]: n
No XLA JIT support will be enabled for TensorFlow.
Do you wish to build TensorFlow with OpenCL SYCL support? [y/N]: n
No OpenCL SYCL support will be enabled for TensorFlow.
Do you wish to build TensorFlow with ROCm support? [y/N]: n
No ROCm support will be enabled for TensorFlow.
Do you wish to build TensorFlow with CUDA support? [y/N]: n
No CUDA support will be enabled for TensorFlow.
Do you wish to download a fresh release of clang? (Experimental) [y/N]: n
Clang will not be downloaded.
Do you wish to build TensorFlow with MPI support? [y/N]: n
No MPI support will be enabled for TensorFlow.
Please specify optimization flags to use during compilation when bazel option "--config=opt" is specified [Default is -march=native -Wno-sign-compare]:
Would you like to interactively configure ./.WORKSPACE for Android builds? [y/N]: n
Not configuring the WORKSPACE for Android builds.
Preconfigured Bazel build configs. You can use any of the below by adding "--config=<>" to your build command. See .bazelrc for more details.
--config=mkl          # Build with MKL support.
--config=monolithic   # Config for mostly static monolithic build.
--config=gdr          # Build with GDR support.
--config=verbs        # Build with libverbs support.
--config=ngraph        # Build with Intel nGraph support.
--config=dynamic_kernels # (Experimental) Build kernels into separate shared objects.
Preconfigured Bazel build configs to DISABLE default on features:
--config=noaws        # Disable AWS S3 filesystem support.
--config=nogcp        # Disable GCP support.
--config=nohdfs        # Disable HDFS support.
--config=noignite     # Disable Apache Ignite support.
--config=nokafka      # Disable Apache Kafka support.
Configuration finished
yumeng@yumeng-VirtualBox:~/workspace/tensorflow-bazel/tensorflow$

```

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
```

```

Found 1 possible inputs: (name=input_1, type=float(1), shape=[?, ?, 3])
No variables spotted.
Found 3 possible outputs: (name=conv_81/BiasAdd, op=BiasAdd) (name=conv_93/BiasAdd, op=BiasAdd) (name=conv_105/BiasAdd, op=BiasAdd)
Found 61592621 (61.59M) const parameters, 0 (0) variable parameters, and 0 control edges
Op types used: 453 Const, 366 Identity, 75 Conv2D, 74 Mul, 72 FusedBatchNorm, 72 Maximum, 23 Add, 5 Pad, 3 BiasAdd, 2 ConcatV2, 2 Re
To use with tensorflow/tools/benchmark:benchmark_model try these arguments:
bazel run tensorflow/tools/benchmark:benchmark_model -- --graph=/home/yumeng/models/pepper_4classes_trained.pb --show_flops --input_
output_layer=conv_81/BiasAdd,conv_93/BiasAdd,conv_105/BiasAdd

```

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
```