楊佳誠
最後由 楊佳誠 於 2023年12月30日 編輯
貢獻者：

Cross Compile Te...
Train your own y...
全部展開
回到頂部
移至底部
讚賞
編輯
3

Cross Compile Tensorflow Lite and Opencv

1 1
1

Build tensorflow lite

Cross compilation TensorFlow Lite with CMake

- Build with Cmake
  - Overview
  - Cross Compilation for ARM

```
git clone https://github.com/tensorflow/tensorflow.git tensorflow_src
```

- downgrade tensorflow lite

```
cd tensorflow_src
git checkout tags/v2.5.0
```

```
cd ..
mkdir tensorflow_build
cd tensorflow_build
```

```
ARMCC_PREFIX=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/bin/a
ARMCC_FLAGS="-DWITH_PARALLEL_PF=OFF -funsafe-math-optimizations"
cmake -DCMAKE_C_COMPILER=${ARMCC_PREFIX}gcc \
  -DCMAKE_CXX_COMPILER=${ARMCC_PREFIX}g++ \
  -DCMAKE_C_FLAGS="${ARMCC_FLAGS}" \
  -DCMAKE_CXX_FLAGS="${ARMCC_FLAGS}" \
  -DCMAKE_VERBOSE_MAKEFILE:BOOL=ON \
  -DCMAKE_SYSTEM_NAME=Linux \
  -DCMAKE_SYSTEM_PROCESSOR=arm \
  -DTFLITE_ENABLE_XNNPACK=OFF \
  /home/roman/tensorflow_src/tensorflow/lite
```

> 💡 You should upgrade your cmake
> Upgrade your cmake

```
cmake --build . -j2
```

- 生成static library .a檔，並利用自己的方法將 .a 檔撈出

Cross Compile Opencv

- download and downgrade

```
git clone https://github.com/opencv/opencv.git
cd opencv
git checkout 3.4.7
```

```
cd ~/opencv/platforms/linux
mkdir -p build_hardfp
cd build_hardfp
```

- add code into opencv/CMakeLists.txt (search "ocv_include_directories")

```
ocv_include_directories(./3rdparty/zlib)
```

- cross compile

```
ARMCC_PREFIX=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/bin/a
ARMCC_FLAGS="-DWITH_PARALLEL_PF=OFF -funsafe-math-optimizations"
cmake -DCMAKE_C_COMPILER=${ARMCC_PREFIX}gcc \
  -DCMAKE_CXX_COMPILER=${ARMCC_PREFIX}g++ \
  -DCMAKE_C_FLAGS="${ARMCC_FLAGS}" \
  -DCMAKE_CXX_FLAGS="${ARMCC_FLAGS}" \
  -DCMAKE_VERBOSE_MAKEFILE:BOOL=ON \
  -DCMAKE_SYSTEM_NAME=Linux \
  -DCMAKE_SYSTEM_PROCESSOR=arm \
  -DBUILD_SHARED_LIBS=OFF \
  /home/roman/opencv
```

```
make -j4
sudo make install (including lib, include, you need)
```

- show linking options, including the paths to the required library files and the library names.

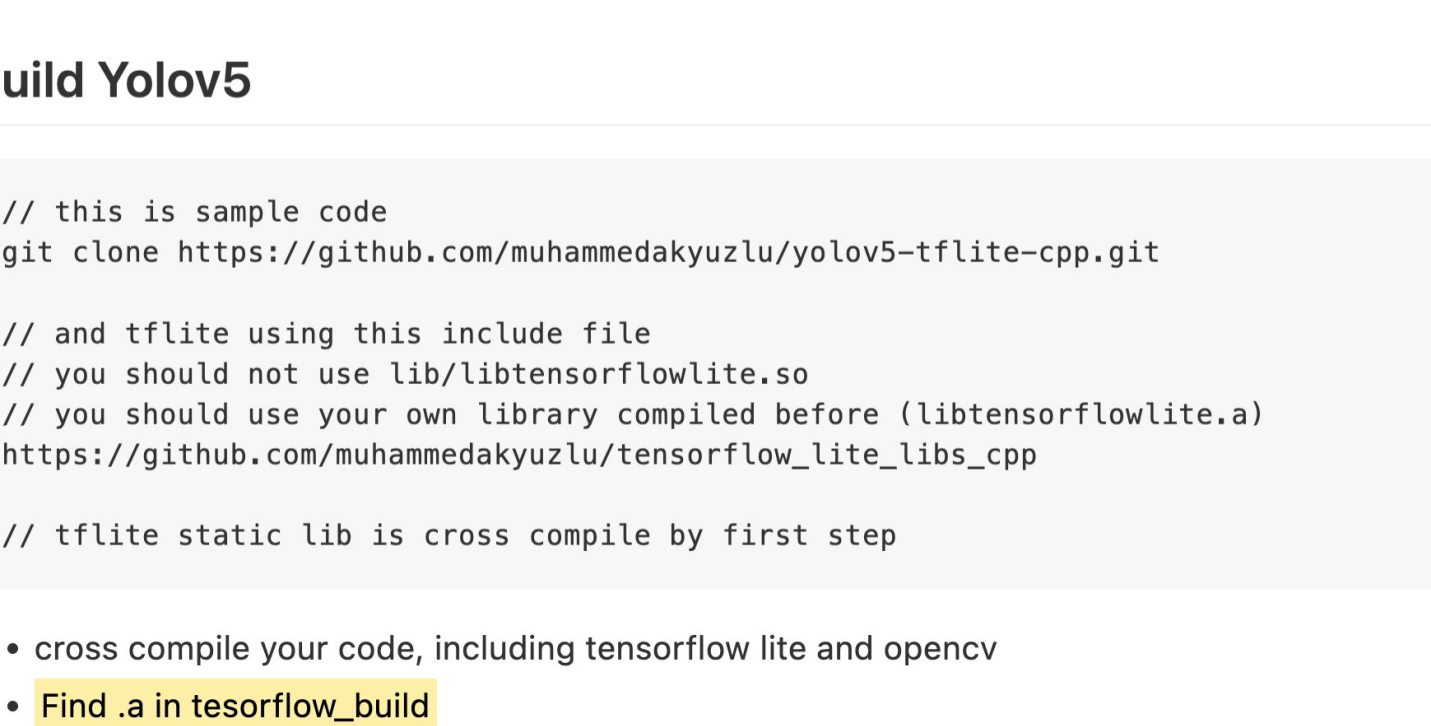
```
pkg-config --cflags --libs opencv.pc
```

```
-lopencv_dnn -lopencv_highgui -lopencv_ml -lopencv_objdetect -lopencv_shape -lopen
-lopencv_superres -lopencv_videostab -lopencv_calib3d -lopencv_videoio -lopencv_im
-lopencv_features2d -lopencv_video -lopencv_photo -lopencv_imgproc -lopencv_flann
```

Build Yolov5

```
// this is sample code
git clone https://github.com/muhammedakyuzulu/yolov5-tflite-cpp.git

// and tflite using this include file
// you should not use lib/libtensorflowlite.so
// you should use your own library compiled before (libtensorflowlite.a)
https://github.com/muhammedakyuzulu/tensorflow_lite_libs_cpp

// tflite static lib is cross compile by first step
```

- cross compile your code, including tensorflow lite and opencv
- Find .a in tesorflow_build



- Likewise find .a in /home/roman/opencv/platforms/linux/build_hardfp, there are 3rdparty .a
- Others opencv.a may be found in /usr/local/lib
- The location of .a can be found when executing make install, it will list the path
- copy all these .a to /home/roman/lib
- $-L$: library 搜尋路徑
- $-l$: 要搜索的library名稱(去掉lib)
- ex. $libray.a - > -lray$
- $-I$: include標頭檔搜尋路徑

```
arm-linux-gnueabihf-g++ src/yolov5_tflite.cpp src/main2.cpp -o test2 \
-I /opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/include/ \
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/ar
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/qt
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/qt
-Wl,-rpath-link=/opt/EmbedSky/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/qt
-I/usr/local/include \
-I/usr/local/include/opencv \
-I/usr/local/include/opencv2 \
-I/home/roman/yolov5-tflite-cpp/tensorflow_lite_libs_cpp-main/include \
-L/home/roman/lib \
-lpthread \
-std=c++11 \
-ltensorflow-lite \
-ldl \
-lopencv_dnn -lopencv_highgui -lopencv_ml -lopencv_objdetect -lopencv_shape -lopen
-lopencv_superres -lopencv_videostab -lopencv_calib3d -lopencv_videoio -lopencv_im
-lopencv_features2d -lopencv_video -lopencv_photo -lopencv_imgproc -lopencv_flann
-lzlib -ltegra_hal -lquirc -llibwebp -llibtiff -llibprotobuf -llibpng -llibjpeg-tu
-llibjasper -littnotify \
-lflatbuffers \
-lfarmhash \
-lfft2d_fftsg2d \
-lfft2d_fftsg \
-lruy \
-labsl_synchronization \
-labsl_graphcycles_internal \
-labsl_strings_internal \
-labsl_cord \
-labsl_strings \
-labsl_str_format_internal \
-labsl_base \
-labsl_throw_delegate \
-labsl_log_severity \
-labsl_dynamic_annotations \
-labsl_malloc_internal \
-labsl_raw_logging_internal \
-labsl_spinlock_wait \
-labsl_debugging_internal \
-labsl_symbolize \
-labsl_stacktrace \
-labsl_demangle_internal \
-labsl_status \
-labsl_time \
-labsl_time_zone \
-labsl_civil_time \
-labsl_hash \
-labsl_city \
-labsl_flags_config \
-labsl_flags_program_name \
-labsl_flags \
-labsl_flags_registry \
-labsl_flags_internal \
-labsl_flags_marshalling \
-labsl_bad_variant_access \
-labsl_bad_optional_access \
-labsl_int128
```

Modify the code

- modify the code in yolov5-tflite-cpp-main/src/main.cpp for your own need
- compile again and get the execution file
- example

```
#include "yolov5_tflite.h"

int main(int argc, char *argv[])
{
    if (argc != 5)
    {
        std::cout << "\nError! Usage: <path to tflite model> <path to classes name
                  << std::endl;
        return 1;
    }

    Prediction out_pred;
    const std::string model_path = argv[1];
    const std::string names_path = argv[2];
    const std::string video_path = argv[3];
    const std::string save_path = argv[4];

    std::vector<std::string> labelNames;

    YOLOV5 model;

    // conf
    // Modify confThreshold to 0.1 to improve performance
    model.confThreshold = 0.30;
    model.nmsThreshold = 0.40;
    model.nthreads = 4;

    // Load the saved_model
    model.loadModel(model_path);

    // Load names
    model.getLabelsName(names_path, labelNames);

    std::cout << "\nLabel Count: " << labelNames.size() << "\n"
              << std::endl;

    /*cv::VideoCapture capture;
    if (all_of(video_path.begin(), video_path.end(), ::isdigit) == false)
        capture.open(video_path);
    else
        capture.open(stoi(video_path));

    cv::Mat frame;
    if (!capture.isOpened())
        throw "\nError when reading video steam\n";*/
    // cv::namedWindow("w", 1);

    // save video config
    bool save = true;
    /*auto fourcc = capture.get(cv::CAP_PROP_FOURCC);
    int frame_width = capture.get(cv::CAP_PROP_FRAME_WIDTH);
    int frame_height = capture.get(cv::CAP_PROP_FRAME_WIDTH);
    //cv::VideoWriter video(save_path, fourcc, 30, cv::Size(frame_width, frame_hei
        cv::Mat frame;
    for (int ii =0;ii<10;ii++)
    {
    /*    capture >> frame;*/
        frame = cv::imread("bus.jpg");
        if (frame.empty())
            break;
        // start
        auto start = std::chrono::high_resolution_clock::now();
        // Predict on the input image
        model.run(frame, out_pred);
        auto stop = std::chrono::high_resolution_clock::now();
        auto duration = std::chrono::duration_cast<std::chrono::milliseconds>(stop
        std::cout << "\nModel run time 'milliseconds': " << duration.count() << "\
                  << std::endl;

        // add the bbox to the image and save it
        auto boxes = out_pred.boxes;
        auto scores = out_pred.scores;
        auto labels = out_pred.labels;

        for (int i = 0; i < boxes.size(); i++)
        {
            auto box = boxes[i];
            auto score = scores[i];
            auto label = labels[i];
            cv::rectangle(frame, box, cv::Scalar(255, 0, 0), 2);
            cv::putText(frame, labelNames[label], cv::Point(box.x, box.y), cv::FONT
        }
        cv::cvtColor(frame, frame, cv::COLOR_BGR2RGB);
        //cv::imshow("output", frame);
        out_pred = {};
        //if (save == false)
        {
            //cv::resize(frame, frame, cv::Size(frame_width, frame_height), 0, 0,
            //video.write(frame);
            cv::imwrite("./out.jpg",frame);
        }
        //cv::waitKey(10);
    }/*
    capture.release();

    if (save == true)
    {
        video.release();
    }
    cv::destroyAllWindows();*/
}
```

Execute

```
./test2 ./models/yolov5n-int8.tflite ./models/coco.names . .
```

```
./test2 ./models/weight/yolov5n-int8.tflite ./models/coco.names . .
```

Train your own yolov5

Download coco2017

```
import fiftyone as fo
import fiftyone.zoo as foz

# List available zoo datasets
print(foz.list_zoo_datasets())

#
# Load the COCO-2017 validation split into a FiftyOne dataset
#
# This will download the dataset from the web, if necessary
#

dataset = foz.load_zoo_dataset(
    "coco-2017",
    label_types=["detections"],
    classes=["umbrella",
    "cup",
    "fork",
    "spoon",
    "dining table",
    "tv",
    "laptop",
    "mouse",
    "remote",
    "keyboard",
    "cell phone",
    "book",
    "scissors"],
)



# Give the dataset a new name, and make it persistent so that you can
# work with it in future sessions
dataset.name = "custom"
dataset.persistent = True

# Visualize the in the App
session = fo.launch_app(dataset)

```

Make custom dataset

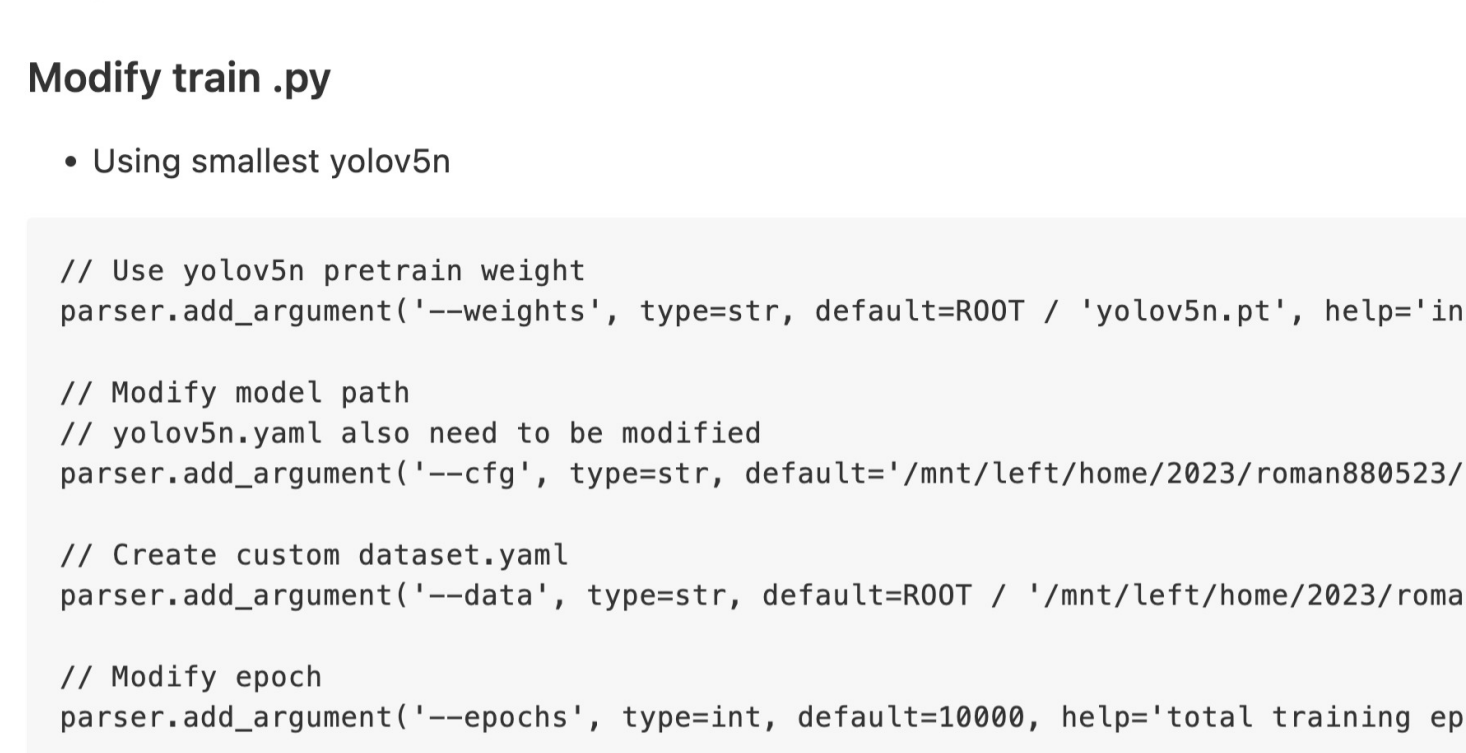
Converting a custom dataset from COCO format to YOLO format

```
import json
import cv2
import os
import matplotlib.pyplot as plt
import shutil


input_path = "/mnt/left/home/2023/roman880523/fiftyone/coco-2017/train/data"
output_path = "/mnt/left/home/2023/roman880523/fiftyone/yolo_dataset/train/"

f = open('/mnt/left/home/2023/roman880523/fiftyone/coco-2017/train/labels.js
data = json.load(f)
f.close()
file_names = []
label_dict = {}
label_list = [28,
47,
48,
50,
55,
67,
72,
74,
76,
77,
84,
87]
for i in range(12):
    label_dict[label_list[i]] = i
print(label_dict)


def get_img_ann(image_id):
    img_ann = []
    isFound = False
    for ann in data['annotations']:
        if ann['image_id'] == image_id:
            img_ann.append(ann)
            isFound = True
    if isFound:
        return img_ann
    else:
        return None


def get_img(filename):
  for img in data['images']:
    if img['file_name'] == filename:
      return img

def load_images_from_folder(folder):
  count = 0
  for filename in os.listdir(folder):
        source = os.path.join(folder,filename)
        destination = f"{output_path}images/{filename}.jpg"

        try:
            # Extracting image
            img = get_img(filename)
            img_id = img['id']
            # Get Annotations for this image
            img_ann = get_img_ann(img_id)
            if img_ann:
                for ann in img_ann:
                    if ann['category_id'] in label_list:
                        shutil.copy(source, destination)
                        break
            #print("File copied successfully.")
        # If source and destination are same
        except shutil.SameFileError:
            print("Source and destination represents the same file.")

        file_names.append(filename)
        count += 1





load_images_from_folder(input_path)
count = 0

for filename in file_names:
  # Extracting image
  img = get_img(filename)
  img_id = img['id']
  img_w = img['width']
  img_h = img['height']

  # Get Annotations for this image
  img_ann = get_img_ann(img_id)
  if img_ann:
    for ann in img_ann:
      if ann['category_id'] in label_list:

        # Opening file for current image
        file_object = open(f"{output_path}labels/{filename}.txt", "a")
        current_category = ann['category_id'] - 1 # As yolo format labels st
        current_category = label_dict[ann['category_id'] ]
        # print(current_category)
        current_bbox = ann['bbox']
        x = current_bbox[0]
        y = current_bbox[1]
        w = current_bbox[2]
        h = current_bbox[3]

        # Finding midpoints
        x_centre = (x + (x+w))/2
        y_centre = (y + (y+h))/2

        # Normalization
        x_centre = x_centre / img_w
        y_centre = y_centre / img_h
        w = w / img_w
        h = h / img_h

        # Limiting upto fix number of decimal places
        x_centre = format(x_centre, '.6f')
        y_centre = format(y_centre, '.6f')
        w = format(w, '.6f')
        h = format(h, '.6f')

        # Writing current object
        file_object.write(f"{current_category} {x_centre} {y_centre} {w} {h}
        file_object.close()
      else:
        pass
        #print("None")


    #count += 1  # This should be outside the if img_ann block.

# need to -1
# 0 28 "umbrella"
# 1 47 "cup"
# 2 48 "fork"
# 3 50 "spoon"
# 4 55 "orange"
# 5 67 "dining table"
# 6 72 "tv"
# 7 74 "mouse"
# 8 76 "keyboard"
# 9 77 "cell phone"
# 10 84 "book"
# 11 87 "scissors"

# label.json
# "categories": [{"supercategory": "person","id": 1,"name": "person"}

```

Check correctness

Roboflow

- Use this tool to check the correctness of label



Train yolov5

- yolov5

Modify train .py

- Using smallest yolov5n

```
// Use yolov5n pretrain weight
parser.add_argument('--weights', type=str, default=ROOT / 'yolov5n.pt', help='init

// Modify model path
// yolov5n.yaml also need to be modified
parser.add_argument('--cfg', type=str, default='/mnt/left/home/2023/roman880523/fi

// Create custom dataset.yaml
parser.add_argument('--data', type=str, default=ROOT / '/mnt/left/home/2023/roman8

// Modify epoch
parser.add_argument('--epochs', type=int, default=10000, help='total training epoc

// Modify input size to accelerate the inference time
// This is really important !!!
parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=160, help=

// Use gpu
parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or 0,1,2,3 c

// Save pt period to backup
parser.add_argument('--save-period', type=int, default=20, help='Save checkpoint e
```

Modify yolov5n.yaml

```
# Parameters
nc: 12  # number of classes
```

coco_custom.yaml

```
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
# COCO128-seg dataset https://www.kaggle.com/ultralytics/coco128 (first 128 i
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov5
# └── datasets
#     └── coco128-seg  ← downloads here (7 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or
path: /mnt/left/home/2023/roman880523/fiftyone/yolo_dataset/  # dataset root
train: train  # train images (relative to 'path') 128 images
val: validation  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
names:
  0: umbrella
  1: cup
  2: fork
  3: spoon
  4: orange
  5: dining table
  6: tv
  7: mouse
  8: keyboard
  9: cell phone
  10: book
  11: scissors
```

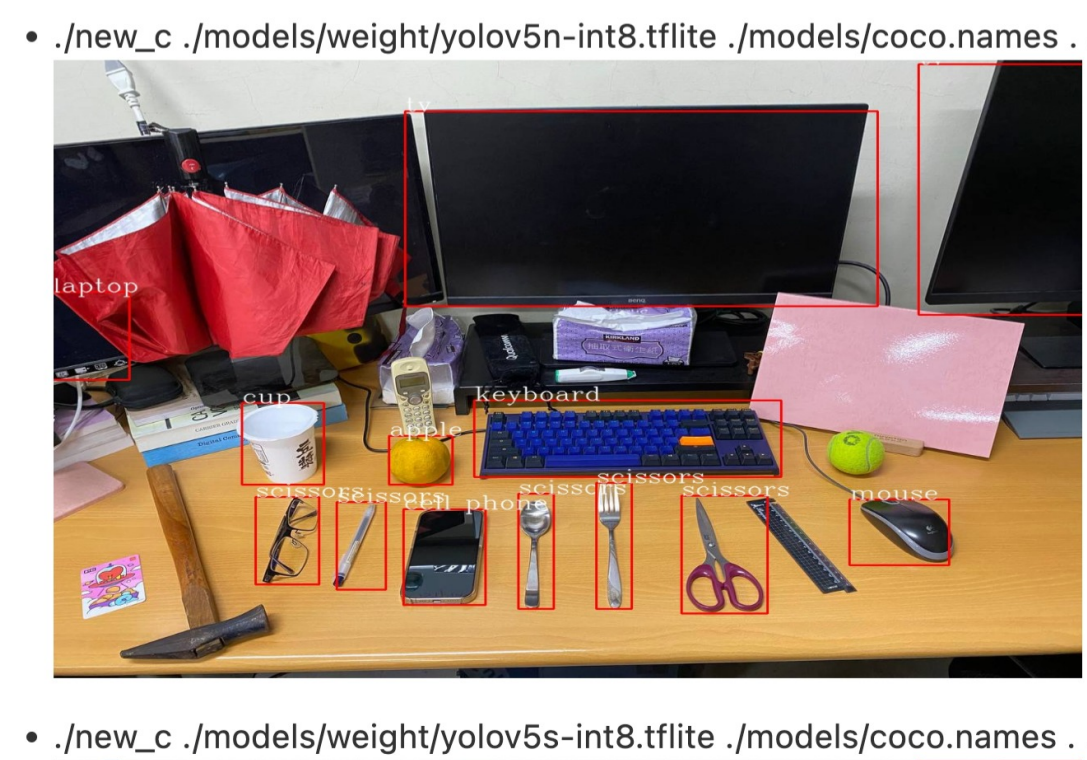
Export .tflite

```
python export.py \
--weights best.pt \
--include tflite  --int8 \
--img 160
```

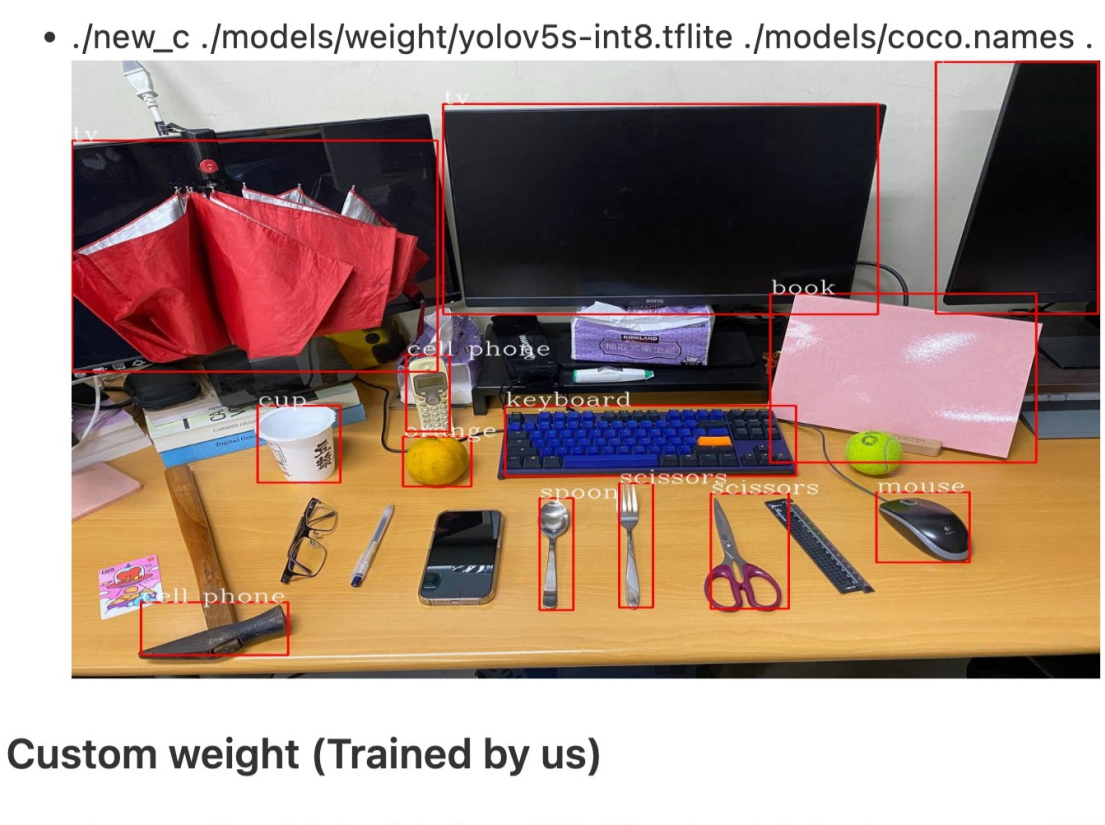
Result

Weight in link

- ./new_c ./models/weight/yolov5n-int8.tflite ./models/coco.names . . 15000ms

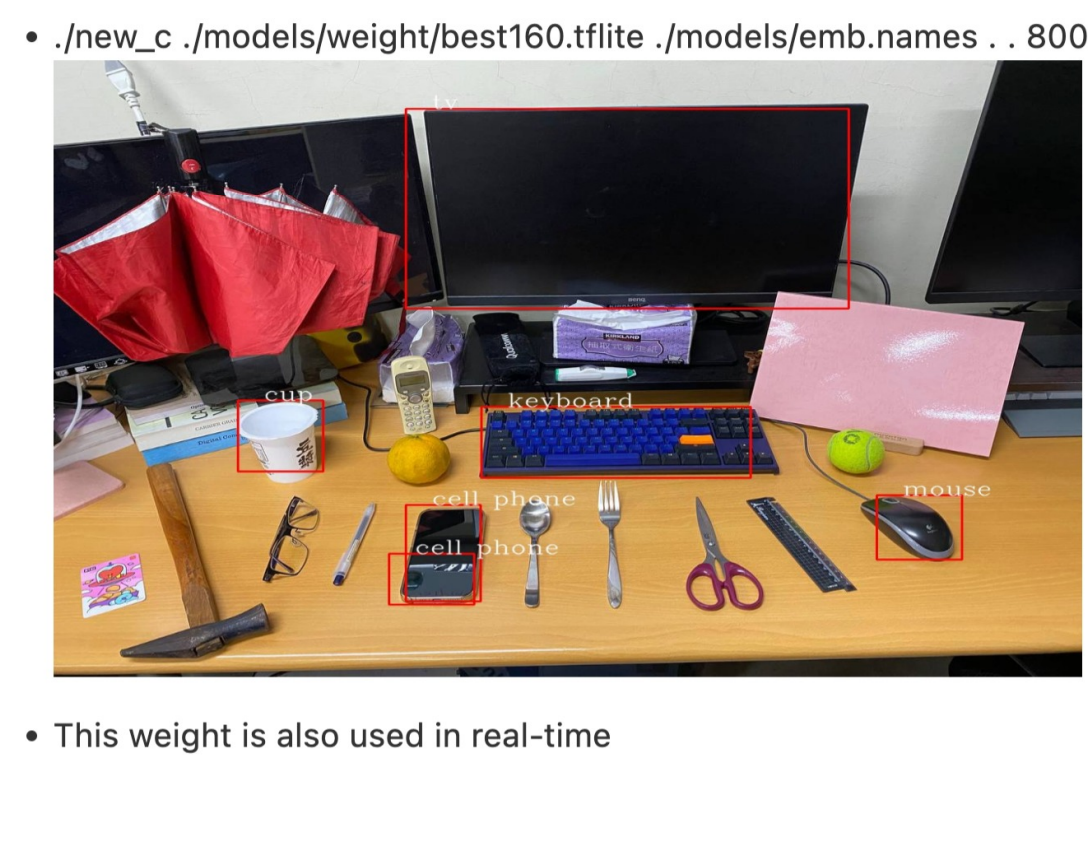


- ./new_c ./models/weight/yolov5s-int8.tflite ./models/coco.names . . 50000ms



Custom weight (Trained by us)

- ./new_c ./models/weight/best160.tflite ./models/emb.names . . 800ms



- This weight is also used in real-time

楊佳誠
最近編輯
3
188
加入留言