Xây dựng ứng dụng phát hiện đối tượng với Tensorflow và OpenCV

Github: https://github.com/doanthuan/travel_assistant

Youtube: https://youtu.be/7wYfqkuYA5I

Mục tiêu

Ứng dụng hỗ trợ người tham gia giao thông lái xe an toàn và tránh vi phạm bằng cách phát hiện biển báo hoặc cảnh sát giao thông thông qua camera

Nội dung

Bài viết này mình sẽ hướng dẫn chi tiết từng bước cần làm để tự xây dựng cho mình 1 ứng dụng có khả năng dò tìm 1 đối tượng trong bức ảnh hoặc video. Gồm 2 phần chính, một là cách tự tạo dataset và train cho mình 1 model, hai là ứng dụng model được train vào ứng dụng dò tìm đối tượng. Ứng dụng này được viết bằng python, và sử dụng 1 số thư viện như OpenCV và module Object Detection của Tensorflow.

Cài đặt

Giả sử các bạn đã có sẵn Ubuntu 16.04 và Python

Các thư viện cần cài đặt: OpenCV, Tensorflow, Object Detection API

Tạo bộ dữ liệu

Bước đầu tiền của bài toán Machine Learning là tạo bộ dữ liệu. Hiện tại có rất nhiều bộ dữ liệu sẵn có để dùng cho việc nhận dạng đối tượng như: PASCAL VOC dataset hay Oxford Pet dataset. Nhưng đối tượng mình cần quan tâm là "cảnh sát giao thông", nên mình cần tự tạo bộ dữ liệu riêng.

Mình đã lấy dữ liệu từ Google Image và dùng labelImg để đánh nhãn cho chúng, nó sẽ tự lưu với định dạng của PASCAL VOC. Do không có nhiều thời gian nên mình chỉ lấy và gắn nhãn khoảng 150 hình, 120 hình cho trainning và 30 hình cho testing.

Do định dạng data yêu cầu của Tensorflow Object Detection API là TFRecord nên mình cần chuyển từ PASCAL VOC về TFRecord. Google đã cung cấp sẵn script để chuyển: create_pascal_tf_record.py, họ cũng hướng dẫn mình cách tự tạo script để chuyển về dạng TFRecord (mình tự tạo script create_tf_record.py với cách này)

Code

Sau khi cài đặt xong, các bạn thiết lập biến môi trường (như trong phần cài đặt)

```
# From tensorflow/models/research/
export PYTHONPATH=`pwd`:`pwd`/slim
```

Sau đó các bạn tạo thư mục data như sau:

```
# From tensorflow/models/research/data
+ images ( chứa hình)
+ annotations ( file xml tạo từ labelImg)
+ create_tf_record.py ( file script để tạo TFRecord file )
+ label_map.pbtxt: file label map (với nội dung đơn giản như sau:
item {
  id: 1
  name: 'police'
})
```

Rồi chạy:

```
python create_tf_record.py --label_map_path=label_map.pbtxt --data_dir=`pwd` --
output_dir=`pwd`
```

2 files: train.record và test.record sẽ được tạo ra. Đây là dataset của mình cần để thực hiện bước kế tiếp là train model .

Train Model với Google Cloud

Có 2 cách để train model: đó là train local (nếu máy bạn có GPU) hoặc dùng Google Cloud. Do máy mình không có GPU nên mình sẽ hướng dẫn các bạn cách dùng Google Cloud để train model.

Thiết lập Google Cloud

Chúng ta sẽ sử dụng Google Cloud Machine Learning để train model, các bạn cần làm 1 số bước sau:

1. Tạo Google Cloud Project
2. Cài đặt Google Cloud SDK trên máy. Công cụ này sẽ giúp bạn tải file lên Google Cloud Storage và thực hiện ML training jobs.
3. Bật ML Engine APIs
4. Tạo Google Cloud Storage (GCS) bucket. Chúng ta cần upload data lên đây bởi vì ML Engine jobs chỉ có thể truy cập files từ GCS

Sau khi tạo GCS bucket, các bạn export ra biến môi trường để tiện làm việc

```
export YOUR_GCS_BUCKET=${YOUR_GCS_BUCKET}
```

Sau đó upload bộ dataset của bạn lên GCS bucket

```
# From tensorflow/models/research/data
gsutil cp train.record gs://${YOUR_GCS_BUCKET}/data/ train.record
gsutil cp test.record gs://${YOUR_GCS_BUCKET}/data/test.record
gsutil cp label_map.pbtxt gs://${YOUR_GCS_BUCKET}/data/pet_label_map.pbtxt
```

Sử dụng pretrained Model cho việc Transfer Learning

Việc training 1 model cho việc nhận dạng đối tượng từ đầu có thể mất rất nhiều thời gian (vài ngày). Do đó để tiết kiệm thời gian, chúng ta sẽ sử dụng 1 số model được train sẵn với nhiều loại dataset có tốc độ và độ chính xác khác nhau. Các bạn có thể tham khảo tại đây.

Do ứng dụng mình sẽ được deploy lên những thiết bị di động nên mình sẽ dùng model: ssd_mobilenet_v1_coco.

Tải model, giải nén và copy những file model.ckpt* lên GCS bucket

```
wget http://download.tensorflow.org/models/object_detection/
ssd_mobilenet_v1_coco_2017_11_17.tar.gz
tar -xvf ssd_mobilenet_v1_coco_2017_11_17.tar.gz
```

```
gsutil cp ssd_mobilenet_v1_coco_2017_11_17/model.ckpt.* gs://${YOUR_GCS_BUCKET}/data/
```

Cấu hình Object Detection Pipeline

Các bạn cần có 1 file cấu hình để chứa tham số cho model, cho việc training và đánh giá. Chi tiết tham khảo tại đây.

Chúng ta sẽ dùng 1 file config mẫu có sẵn ở trong

object_detection/samples/configs/ssd_mobilenet_v1_coco.config

Các bạn cần chỉnh sửa file đó bằng cách:

+ Thay PATH_TO_BE_CONFIGURED bằng gs://${YOUR_GCS_BUCKET}/data/

+ Cập nhật input_path và label_map_path bằng tên file TFRecord và labelmap của bộ dataset của bạn.

+ Chỉnh num_classes: 1 (Nếu chỉ cần detect 1 đối tượng là police)

Sau đó các bạn tải file config này lên GCS bucket.

```
# Copy edited template to cloud.
gsutil cp object_detection/samples/configs/ssd_mobilenet_v1_coco.config \
    gs://${YOUR_GCS_BUCKET}/data/ssd_mobilenet_v1_coco.config
```

Kiểm tra lại Google Cloud Storage Bucket

Lúc này các bạn đã có trong GCS bucket bộ training/validation dataset (bao gồm label map), bộ ssd_mobilenet model và file cấu hình cho việc train. Bạn dùng Google Cloud Storage Browser sẽ thấy kết quả như sau:

```
+ ${YOUR_GCS_BUCKET}/
  + data/
    - ssd_mobilenet_v1_coco.config
    - model.ckpt.index
    - model.ckpt.meta
    - model.ckpt.data-00000-of-00001
    - label_map.pbtxt
    - train.record
    - test.record
```

Training và Evalution trên Google Cloud ML Engine

Đầu tiên các bạn cần đóng gói thư viện Tensorflow Object Detection.

```
# From tensorflow/models/research/
python setup.py sdist
```

```
(cd slim && python setup.py sdist)
```

Chúng ta sẽ có được 2 files: `dist/object_detection-0.1.tar.gz` và `slim/dist/slim-0.1.tar.gz`

Thực hiện command sau để bắt đầu 1 training job trên Cloud ML

```
gcloud ml-engine jobs submit training `whoami`_object_detection_`date +%s` \
    --job-dir=gs://${YOUR_GCS_BUCKET}/train \
    --packages dist/object_detection-0.1.tar.gz,slim/dist/slim-0.1.tar.gz \
    --module-name object_detection.train \
    --region us-central1 \
    --config object_detection/samples/cloud/cloud.yml \
    -- \
    --train_dir=gs://${YOUR_GCS_BUCKET}/train \
    --pipeline_config_path=gs://${YOUR_GCS_BUCKET}/data/ssd_mobilenet_v1_coco.config
```

Khi training job được start, các bạn có thể start 1 evalution job để đánh giá model:

```
gcloud ml-engine jobs submit training `whoami`_object_detection_eval_`date +%s` \
    --job-dir=gs://${YOUR_GCS_BUCKET}/train \
    --packages dist/object_detection-0.1.tar.gz,slim/dist/slim-0.1.tar.gz \
    --module-name object_detection.eval \
    --region us-central1 \
    --scale-tier BASIC_GPU \
    -- \
    --checkpoint_dir=gs://${YOUR_GCS_BUCKET}/train \
    --eval_dir=gs://${YOUR_GCS_BUCKET}/eval \
    --pipeline_config_path=gs://${YOUR_GCS_BUCKET}/data/ ssd_mobilenet_v1_coco.config
```

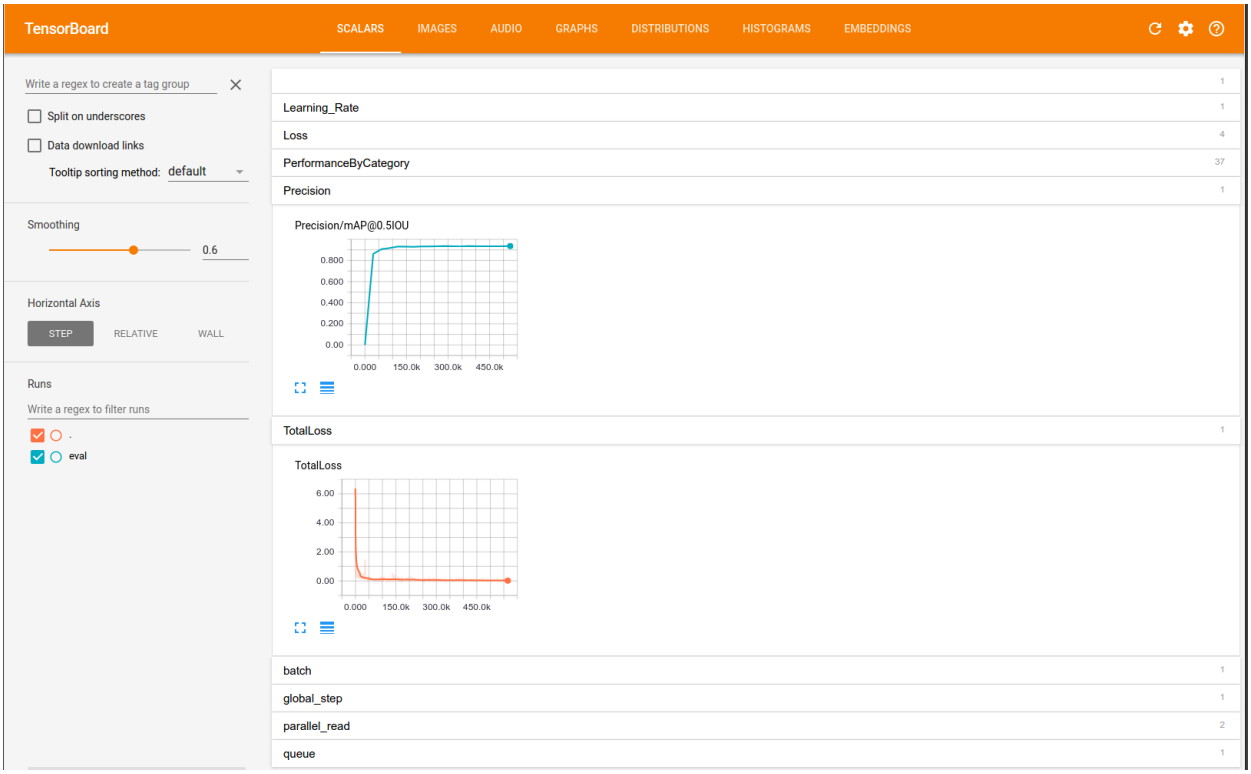
Theo dõi tiến trình training với TensorBoard

Các bạn có thể theo dõi tiến trình train và eval bằng cách chạy TensorBoard ở local

```
gcloud auth application-default login

tensorboard --logdir=gs://${YOUR_GCS_BUCKET}
```

Sau khi chạy câu lệnh trên, các bạn mở trình duyệt vào địa chỉ localhost:6006 sẽ thấy như sau:



Lưu và xuất model

Sau khi train xong, model của bạn nằm trong `${YOUR_GCS_BUCKET}/train`, gồm 3 file ứng với mỗi check point:

- `model.ckpt-${CHECKPOINT_NUMBER}.data-00000-of-00001`
- `model.ckpt-${CHECKPOINT_NUMBER}.index`
- `model.ckpt-${CHECKPOINT_NUMBER}.meta`

Sau đó bạn nên chọn check point number lớn nhất và chép về local.

```
gsutil cp gs://${YOUR_GCS_BUCKET}/train/model.ckpt-${CHECKPOINT_NUMBER}.* .
```

Cuối cùng xuất model ra thành Tensorflow graph proto.

```
python object_detection/export_inference_graph.py \
    --input_type image_tensor \
    --pipeline_config_path
object_detection/samples/configs/ssd_mobilenet_v1_coco.config \
    --trained_checkpoint_prefix model.ckpt-${CHECKPOINT_NUMBER} \
    --output_directory my_model
```

Thư mục my_model chứa model được train và frozen graph dùng cho việc object detection

Viết ứng dụng cho model vừa được train

Tới bước này, bạn đã hoàn thành việc train model cho việc nhận dạng đối tượng cảnh sát giao thông.

Bạn có thể kiểm chứng model vừa được train bằng cách sử dụng Jupyter notebook được cung cấp cùng với thư viện Tensorflow. Bạn có thể xem demo cách dò tìm 1 đối tượng từ model trong TensorFlow: object_detection_tutorial.ipynb.

Hoặc bạn cũng có thể dùng OpenCV để mở 1 file video (hoặc từ camera) để thực hiện việc dò tìm đối tượng trên video đó. Đây là ví dụ minh họa cách đọc khung hình từ video trong OpenCV

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Sau đó ứng với mỗi khung hình, các bạn sẽ gọi hàm để detect object trong khung hình đó. Mình đã tham khảo demo trong object_detection_tutorial.ipynb và viết riêng 1 hàm để detect object từ image.

```
def detect_objects(image_np, sess, detection_graph):

    image_np_expanded = np.expand_dims(image_np, axis=0)
    image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

    boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

    scores = detection_graph.get_tensor_by_name('detection_scores:0')
    classes = detection_graph.get_tensor_by_name('detection_classes:0')
    num_detections = detection_graph.get_tensor_by_name('num_detections:0')

    (boxes, scores, classes, num_detections) = sess.run(
        [boxes, scores, classes, num_detections],
        feed_dict={image_tensor: image_np_expanded})

    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8)
    return image_np
```

Mã nguồn đầy đủ trên github của mình: police_detection.py

Và cuối cùng các bạn thực hiện lệnh sau để chạy chương trình:

```
python police_detection.py –src test_videos/test1.mp4
```

Kết quả: https://youtu.be/8BrRf2Em4VI

Phần kết

Hiện tại do bộ dữ liệu train của mình không nhiều nên độ chính xac không được cao.

Hướng đi kế tiếp của dự án này là mình sẽ cố gắng deploy ứng dụng lên mobile, và hỗ trợ thêm nhiều đối tượng như: các biển báo giao thông nguy hiểm, kẹt xe, tai nạn, sự cố giao thông.. và gửi cảnh báo tới nhiều thiết bị mobile khác có cài đặt ứng dụng.