This wiki describes how to work with object detection models trained using TensorFlow Object Detection API. OpenCV 3.4.1 or higher is required.

Run network in TensorFlow

Deep learning networks in TensorFlow are represented as graphs where every node is a transformation of its inputs. They could be common layers like `Convolution` or `MaxPooling` and implemented in C++. Custom layers could be built from existing TensorFlow operations in python.

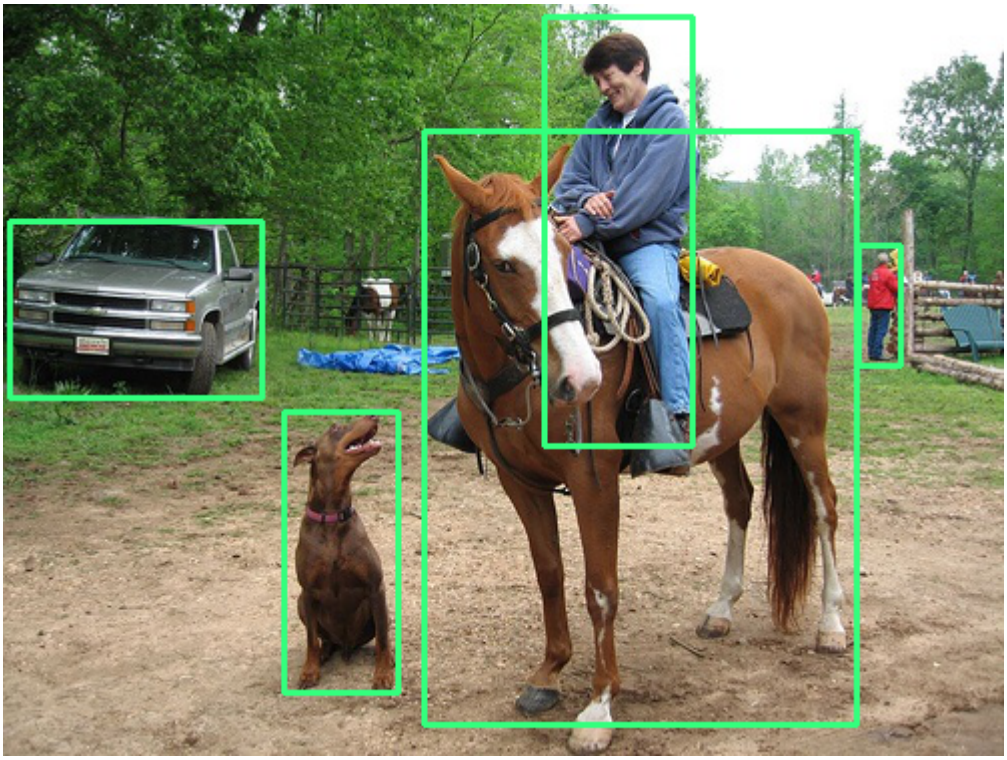
TensorFlow object detection API is a framework for creating deep learning networks that solve object detection problem. There are already trained models in Model Zoo. You can build your own model as well.

The result of training is a binary file with extension `.pb` contains both topology and weights of the trained network. You may download one of them from Model Zoo, in example `ssd_mobilenet_v1_coco` (MobileNet-SSD trained on COCO dataset).

Create and run a python script to test a model on specific picture:

```
import numpy as np
import tensorflow as tf
import cv2 as cv

# Read the graph.
with tf.gfile.FastGFile('frozen_inference_graph.pb', 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())

with tf.Session() as sess:
    # Restore session
    sess.graph.as_default()
    tf.import_graph_def(graph_def, name='')

    # Read and preprocess an image.
    img = cv.imread('example.jpg')
    rows = img.shape[0]
    cols = img.shape[1]
    inp = cv.resize(img, (300, 300))
    inp = inp[:, :, [2, 1, 0]]  # BGR2RGB

    # Run the model
    out = sess.run([sess.graph.get_tensor_by_name('num_detections:0'),
                    sess.graph.get_tensor_by_name('detection_scores:0'),
                    sess.graph.get_tensor_by_name('detection_boxes:0'),
                    sess.graph.get_tensor_by_name('detection_classes:0')],
                   feed_dict={'image_tensor:0': inp.reshape(1, inp.shape[0],
inp.shape[1], 3)})

    # Visualize detected bounding boxes.
    num_detections = int(out[0][0])
    for i in range(num_detections):
        classId = int(out[3][0][i])
        score = float(out[1][0][i])
```

```
        bbox = [float(v) for v in out[2][0][i]]
        if score > 0.3:
            x = bbox[1] * cols
            y = bbox[0] * rows
            right = bbox[3] * cols
            bottom = bbox[2] * rows
            cv.rectangle(img, (int(x), int(y)), (int(right), int(bottom)), (125,
255, 51), thickness=2)

cv.imshow('TensorFlow MobileNet-SSD', img)
cv.waitKey()
```



Run network in OpenCV

OpenCV needs an extra configuration file to import object detection models from TensorFlow. It's based on a text version of the same serialized graph in protocol buffers format (protobuf).

Use existing config file for your model

You can use one of the configs that has been tested in OpenCV. This choice depends on your model and TensorFlow version:

Model	Version		
MobileNet-SSD v1	2017_11_17	weights	config
MobileNet-SSD v1 PPN	2018_07_03	weights	config

MobileNet-SSD v2	2018_03_29	weights	config
Inception-SSD v2	2017_11_17	weights	config
MobileNet-SSD v3 (see #16760)	2020_01_14	weights	config
Faster-RCNN Inception v2	2018_01_28	weights	config
Faster-RCNN ResNet-50	2018_01_28	weights	config
Mask-RCNN Inception v2	2018_01_28	weights	config
EfficientDet-D0 (see #17384)		weights	config

Generate a config file

Use one of the scripts which generate a text graph representation for a frozen `.pb` model depends on its architecture:

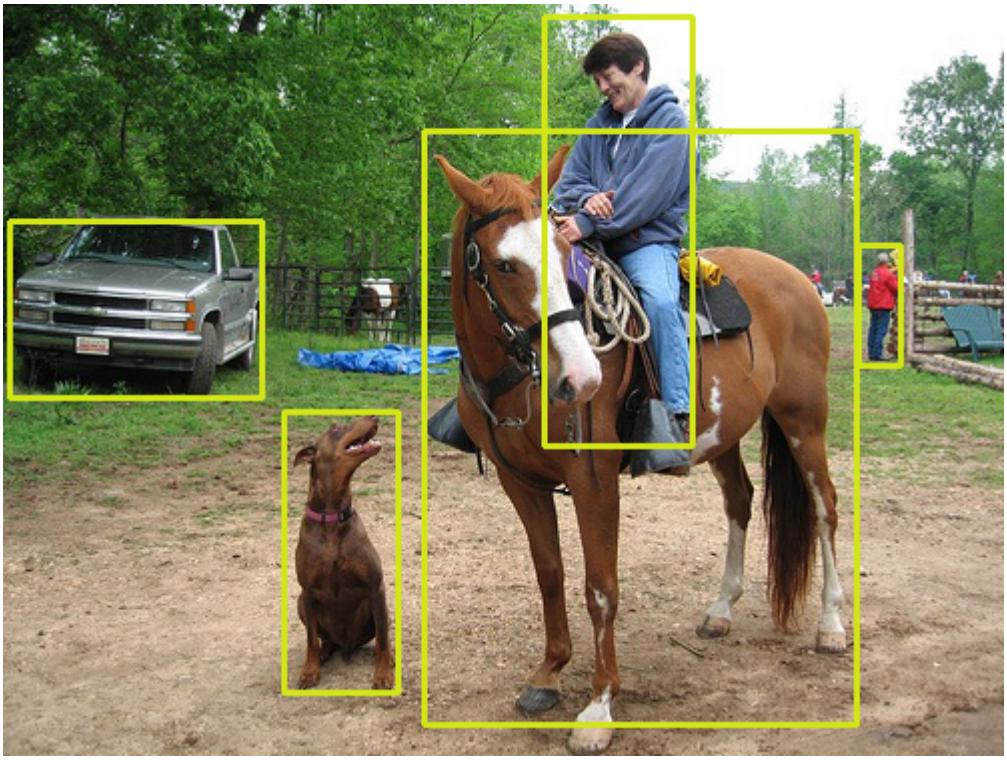
- tf_text_graph_ssd.py
- tf_text_graph_faster_rcnn.py
- tf_text_graph_mask_rcnn.py
- tf_text_graph_efficientdet.py

Pass a configuration file which was used for training to help script determine hyper-parameters.

```
python tf_text_graph_faster_rcnn.py --input /path/to/model.pb --config
/path/to/example.config --output /path/to/graph.pbtxt
```

Try to run the model using OpenCV:

```
import cv2 as cv

cvNet = cv.dnn.readNetFromTensorflow('frozen_inference_graph.pb', 'graph.pbtxt')

img = cv.imread('example.jpg')
rows = img.shape[0]
cols = img.shape[1]
cvNet.setInput(cv.dnn.blobFromImage(img, size=(300, 300), swapRB=True, crop=False))
cvOut = cvNet.forward()

for detection in cvOut[0,0,:,:]:
    score = float(detection[2])
    if score > 0.3:
        left = detection[3] * cols
        top = detection[4] * rows
        right = detection[5] * cols
        bottom = detection[6] * rows
        cv.rectangle(img, (int(left), int(top)), (int(right), int(bottom)), (23,
230, 210), thickness=2)

cv.imshow('img', img)
cv.waitKey()
```



References

- TensorFlow library
- COCO dataset
- Google Protobuf
- OpenCV object detection sample: C++, Python
- OpenCV Mask R-CNN sample