Tensorflow Object Detection API Walkthrough

Topics

Intro

Install packages

Prepare dataset

Training

Evaluation

Intro

open source framework presented by Google

built on top of TensorFlow

covers constructing, training and evaluating object detection models

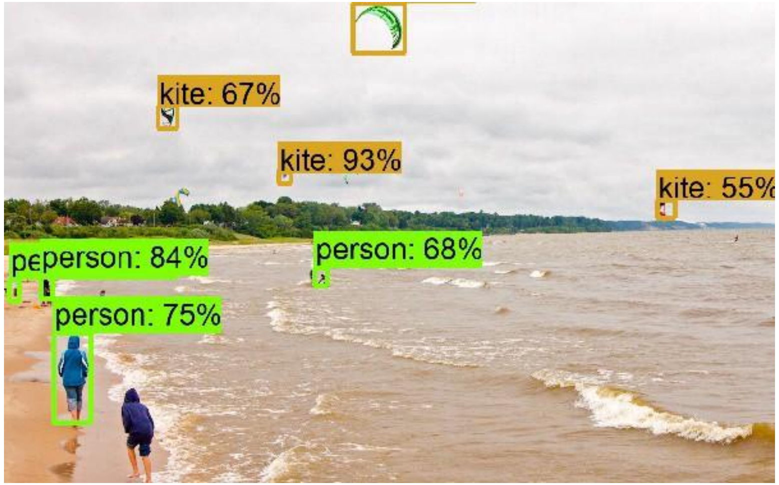
Task: localizing and identifying multiple objects in a single image

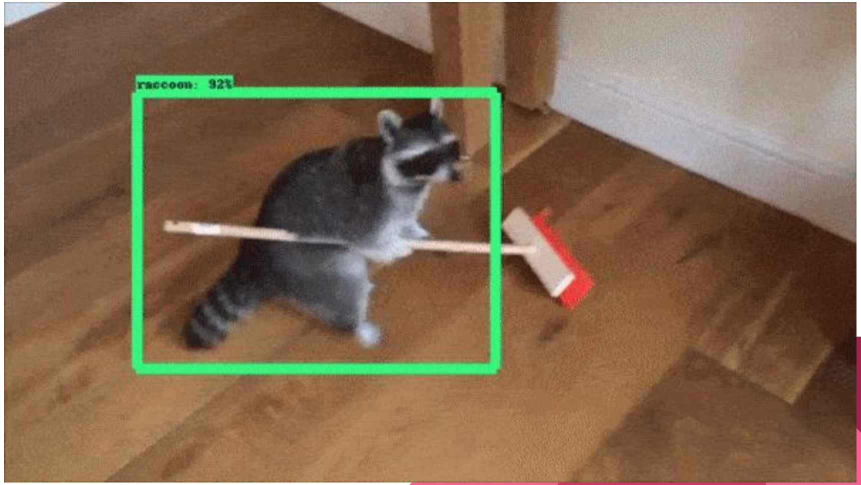
https://github.com/tensorflow/models/tree/master/research/object_detection

Intro

off-the-shelf pre-trained models from computer vision challenges

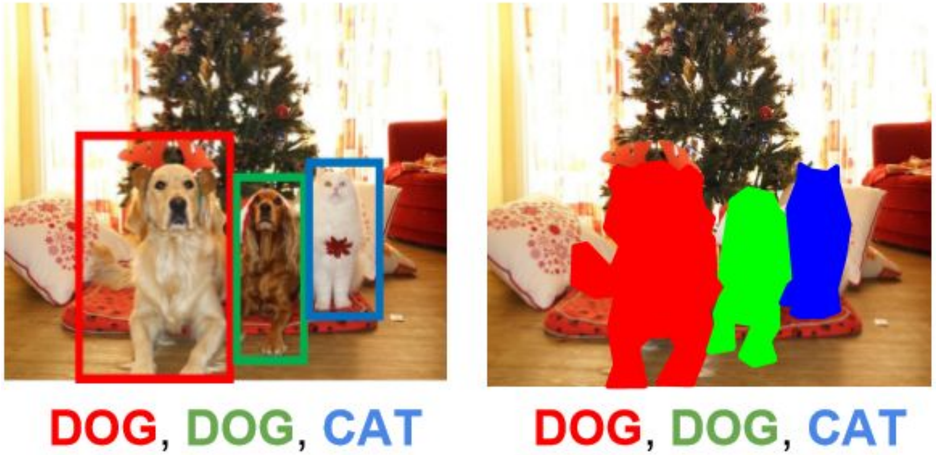
Train (fine tune) your own model with customized dataset





Intro

Object detection(bounding boxes) & segmentation (object outline)

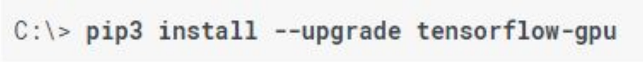


Only convolutional, no recurrent neural nets

Installation - TensorFlow GPU

Python & pip

tensorflow-gpu,



```
C:\> pip3 install --upgrade tensorflow-gpu
```

https://www.tensorflow.org/install/install_windows

From Nvidia: CUDA (GPU driver included) , cuDNN

Note: use versions required by tensorflow, not the latest version!

https://developer.nvidia.com/cuda-toolkit-archive

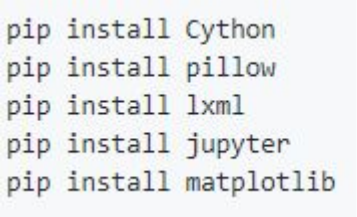
https://developer.nvidia.com/cudnn

Installation - Object Detection API

Download repository from GitHub and unzip to local directory

https://github.com/tensorflow/models/tree/master/research

Install python packages

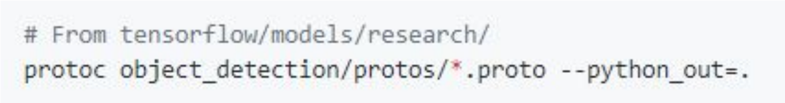


```
pip install Cython
pip install pillow
pip install lxml
pip install jupyter
pip install matplotlib
```

Installation - Object Detection API

Protobuf Compilation (convert protocols to python files)

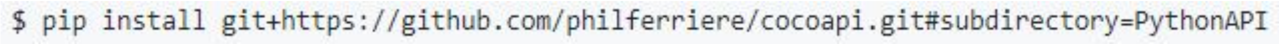
Download protoc.exe v3.4.0 from https://github.com/google/protobuf/releases



```
# From tensorflow/models/research/
protoc object_detection/protos/*.proto --python_out=.
```

COCO API installation (some evaluation metrics come from this API)

Install GitHub for desktop, Visual C++ 2015 build tools, then run

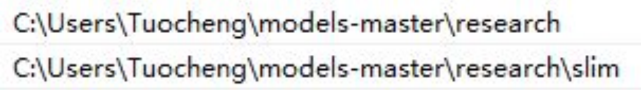


```
$ pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI
```

*Original COCO API has no support on Windows machines

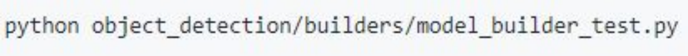
Installation - Object Detection API

Add Libraries to PYTHONPATH as system variables

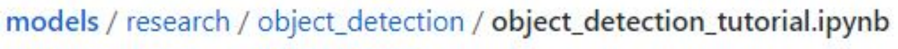


```
C:\Users\Tuocheng\models-master\research
C:\Users\Tuocheng\models-master\research\slim
```

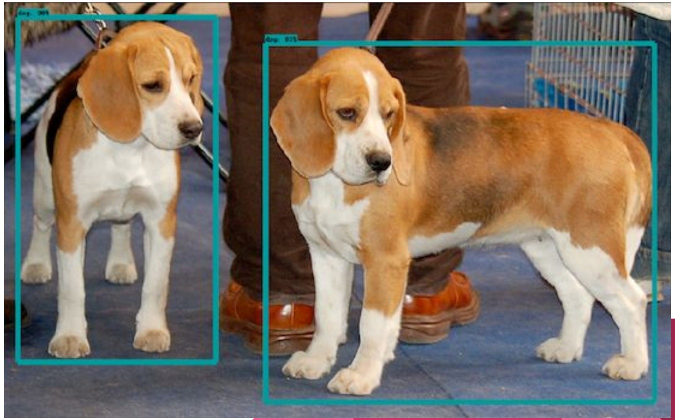
Testing the Installation



```
python object_detection/builders/model_builder_test.py
```



models / research / object_detection / object_detection_tutorial.ipynb

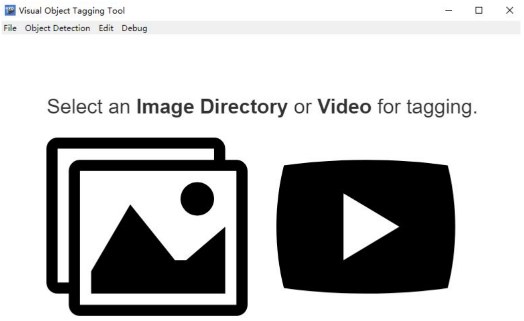


Prepare Dataset

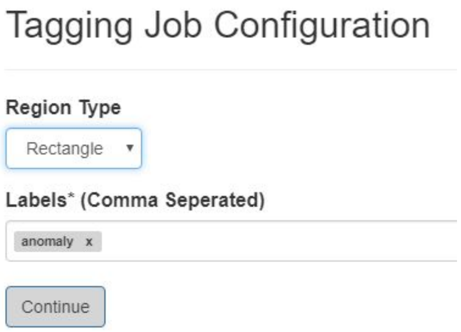
Convert pictures and annotations (labels) into TFrecord (.record files)

Label objects using Microsoft VoTT (visual object tagging tool)

https://github.com/Microsoft/VoTT/releases



Select an **Image Directory** or **Video** for tagging.



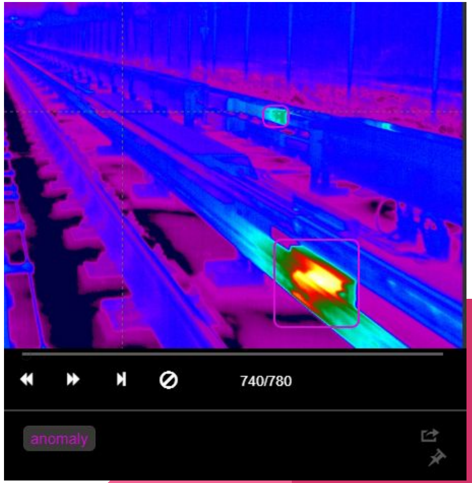
Tagging Job Configuration

Region Type

Rectangle

Labels* (Comma Seperated)

anomaly x

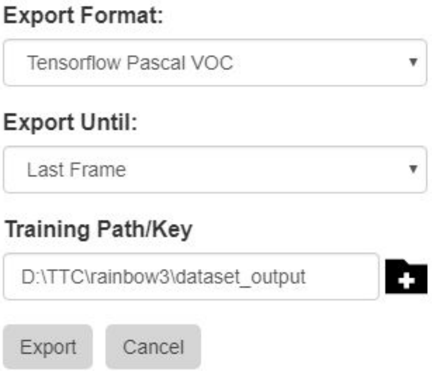


Prepare Dataset

When all pictures labeled, export tags in Pascal VOC format

Images are automatically splitted into train/validation sets, modify if necessary

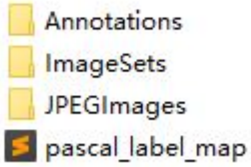
Convert dataset folder into TFrecords using modified scripts in API



Export Format: Tensorflow Pascal VOC

Export Until: Last Frame

Training Path/Key: D:\TTC\rainbow3\dataset_output

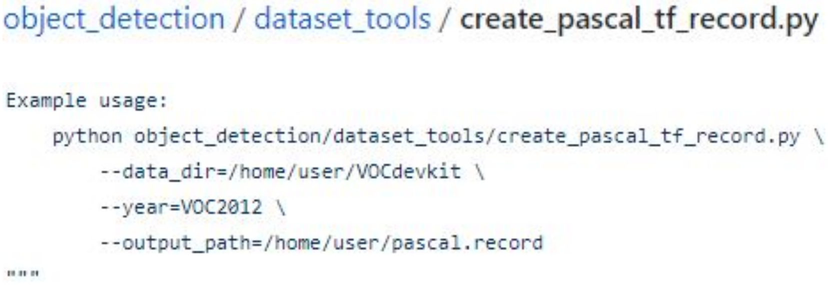


Annotations

ImageSets

JPEGImages

pascal_label_map



object_detection / dataset_tools / create_pascal_tf_record.py

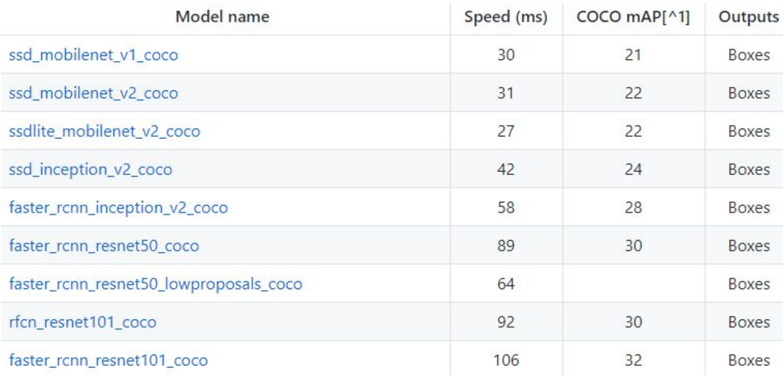
```
Example usage:
    python object_detection/dataset_tools/create_pascal_tf_record.py \
        --data_dir=/home/user/VOCdevkit \
        --year=VOC2012 \
        --output_path=/home/user/pascal.record
"""
```

Training

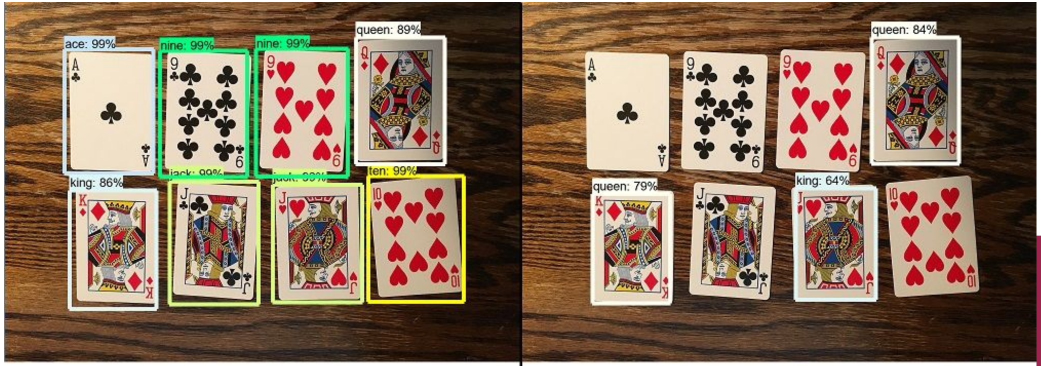
Choose a pre-trained model from the Model Zoo

Use ssd mobilenet model for faster detection speed

Use faster_rcnn model for better performance



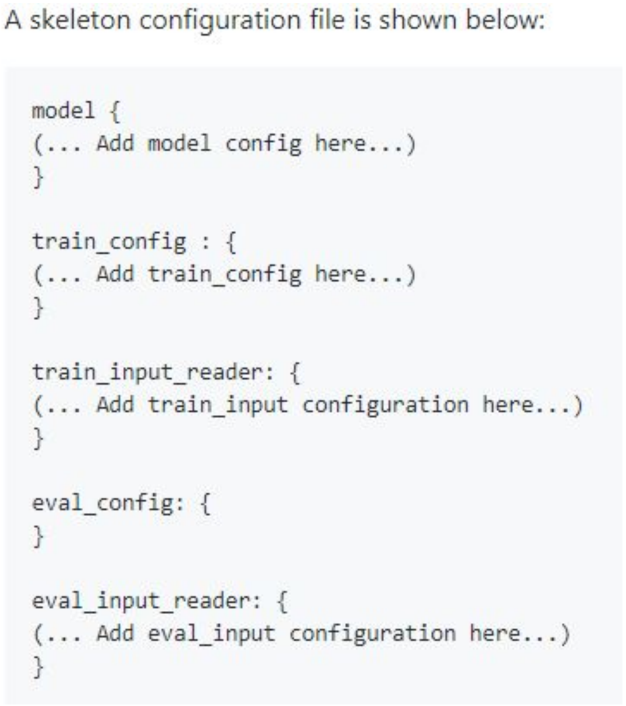
Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes



Model: Faster-RCNN-Inception-V2
Training Time: 3 hours

Model: MobileNet-SSD-V1
Training Time: 8 hours

Training - Tune the parameters in configuration file



A skeleton configuration file is shown below:

```
model {
(... Add model config here...)
}

train_config : {
(... Add train_config here...)
}

train_input_reader: {
(... Add train_input configuration here...)
}

eval_config: {
}

eval_input_reader: {
(... Add eval_input configuration here...)
}
```

Model config

Image resizer, box predictor, feature extractor, loss…

*** Must change num_classes suited to your dataset

Train config

Batch size, optimizer, learning rate, input preprocessing…

*** Provide path of fine tune checkpoint (pre-trained model)

Eval config

Set of metrics used for evaluation, mAP as default

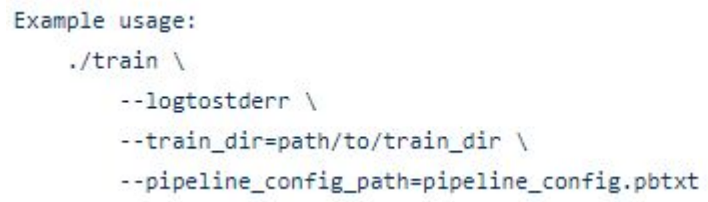
Input readers

*** Path of train/eval dataset TFrecords and label mapping

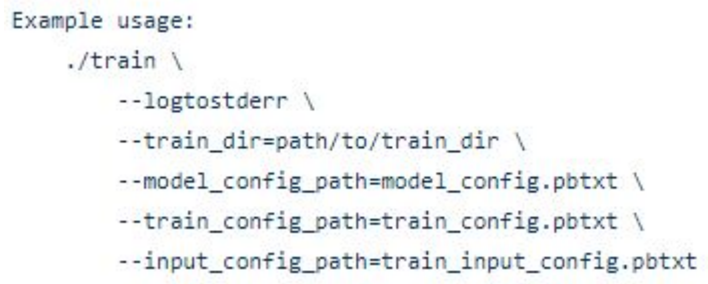
Training

Run python script from command line

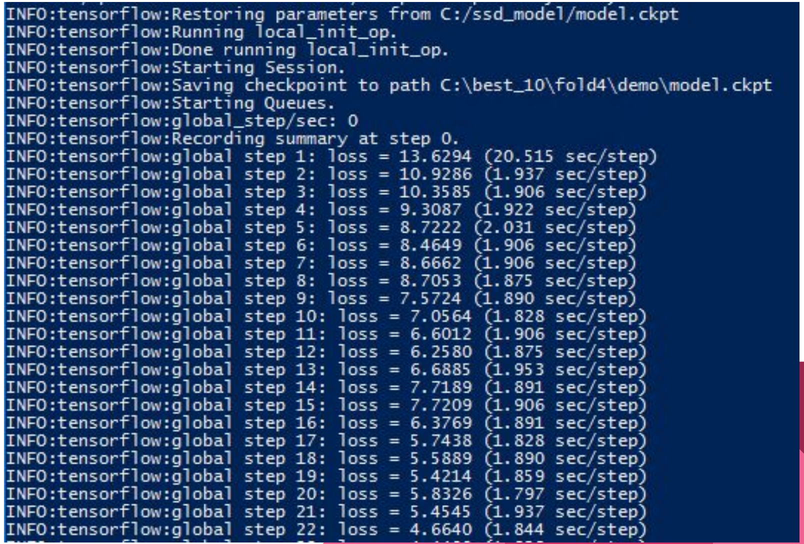
With path of training checkpoints and config file



```
Example usage:
    ./train \
        --logtostderr \
        --train_dir=path/to/train_dir \
        --pipeline_config_path=pipeline_config.pbtxt
```



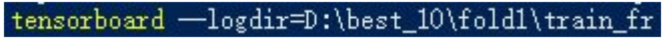
```
Example usage:
    ./train \
        --logtostderr \
        --train_dir=path/to/train_dir \
        --model_config_path=model_config.pbtxt \
        --train_config_path=train_config.pbtxt \
        --input_config_path=train_input_config.pbtxt
```



```
INFO:tensorflow:Restoring parameters from C:/ssd_model/model.ckpt
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Starting Session.
INFO:tensorflow:Saving checkpoint to path C:\best_10\fold4\demo\model.ckpt
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 1: loss = 13.6294 (20.515 sec/step)
INFO:tensorflow:global step 2: loss = 10.9286 (1.937 sec/step)
INFO:tensorflow:global step 3: loss = 10.3585 (1.906 sec/step)
INFO:tensorflow:global step 4: loss = 9.3087 (1.922 sec/step)
INFO:tensorflow:global step 5: loss = 8.7222 (2.031 sec/step)
INFO:tensorflow:global step 6: loss = 8.4649 (1.906 sec/step)
INFO:tensorflow:global step 7: loss = 8.6662 (1.906 sec/step)
INFO:tensorflow:global step 8: loss = 8.7053 (1.875 sec/step)
INFO:tensorflow:global step 9: loss = 7.5724 (1.890 sec/step)
INFO:tensorflow:global step 10: loss = 7.0564 (1.828 sec/step)
INFO:tensorflow:global step 11: loss = 6.6012 (1.906 sec/step)
INFO:tensorflow:global step 12: loss = 6.2580 (1.875 sec/step)
INFO:tensorflow:global step 13: loss = 6.6885 (1.953 sec/step)
INFO:tensorflow:global step 14: loss = 7.7189 (1.891 sec/step)
INFO:tensorflow:global step 15: loss = 7.7209 (1.906 sec/step)
INFO:tensorflow:global step 16: loss = 6.3769 (1.891 sec/step)
INFO:tensorflow:global step 17: loss = 5.7438 (1.828 sec/step)
INFO:tensorflow:global step 18: loss = 5.5889 (1.890 sec/step)
INFO:tensorflow:global step 19: loss = 5.4214 (1.859 sec/step)
INFO:tensorflow:global step 20: loss = 5.8326 (1.797 sec/step)
INFO:tensorflow:global step 21: loss = 5.4545 (1.937 sec/step)
INFO:tensorflow:global step 22: loss = 4.6640 (1.844 sec/step)
```

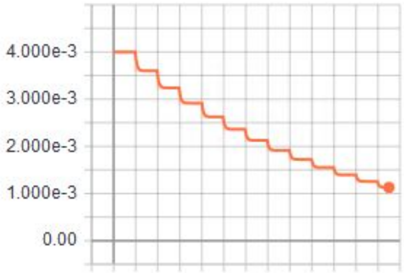
Training

Use TensorBoard to monitor the training progress

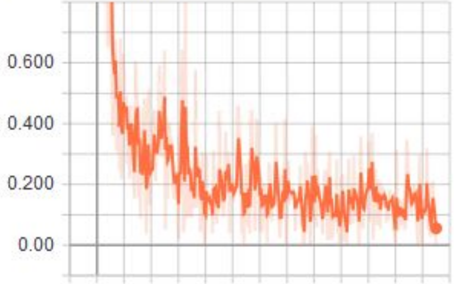


```
tensorboard —logdir=D:\best_10\fold1\train_fr
```

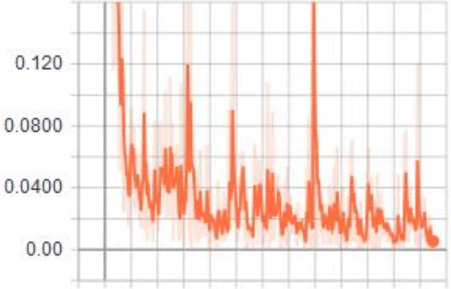
LearningRate/LearningRate/learning_rate



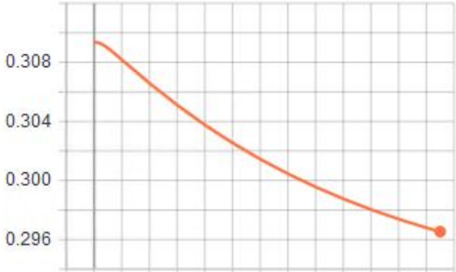
Losses/Loss/classification_loss



Losses/Loss/localization_loss



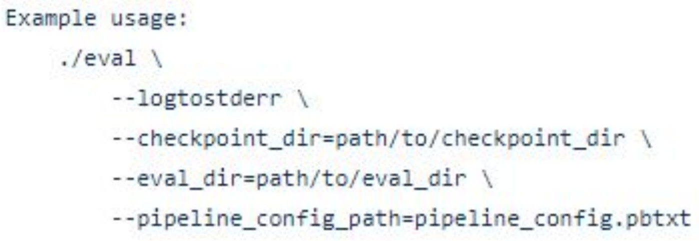
Losses/Losses/regularization_loss



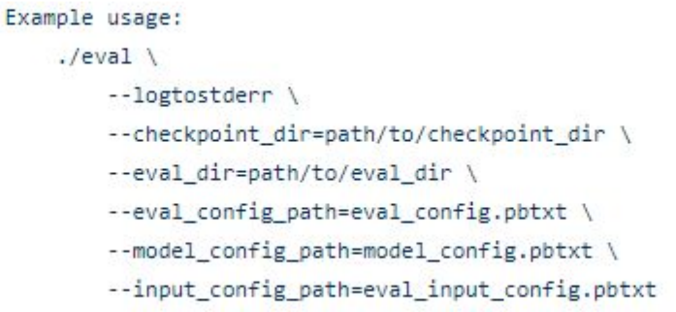
Evaluate

Run python script from command line

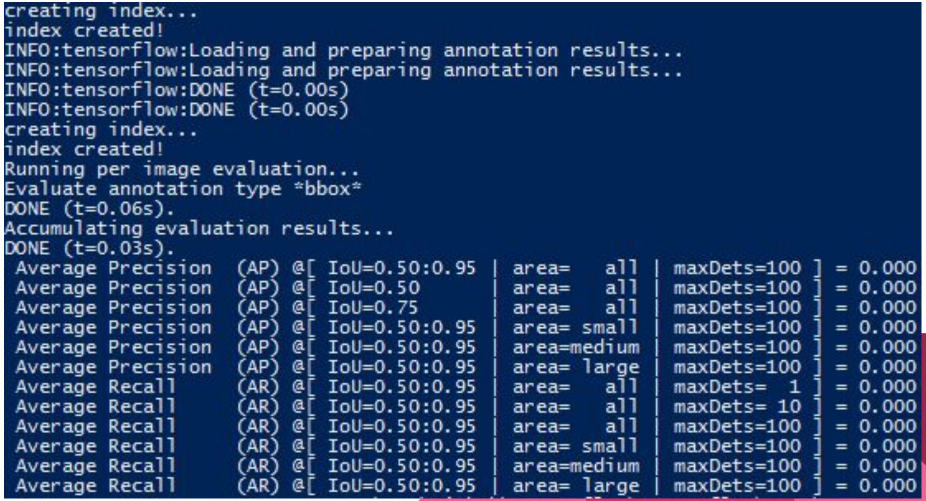
With path of train & eval checkpoints and config file



```
Example usage:
    ./eval \
        --logtostderr \
        --checkpoint_dir=path/to/checkpoint_dir \
        --eval_dir=path/to/eval_dir \
        --pipeline_config_path=pipeline_config.pbtxt
```



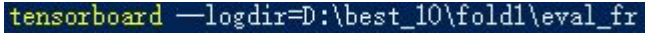
```
Example usage:
    ./eval \
        --logtostderr \
        --checkpoint_dir=path/to/checkpoint_dir \
        --eval_dir=path/to/eval_dir \
        --eval_config_path=eval_config.pbtxt \
        --model_config_path=model_config.pbtxt \
        --input_config_path=eval_input_config.pbtxt
```



```
creating index...
index created!
INFO:tensorflow:Loading and preparing annotation results...
INFO:tensorflow:Loading and preparing annotation results...
INFO:tensorflow:DONE (t=0.00s)
INFO:tensorflow:DONE (t=0.00s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.06s).
Accumulating evaluation results...
DONE (t=0.03s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
```

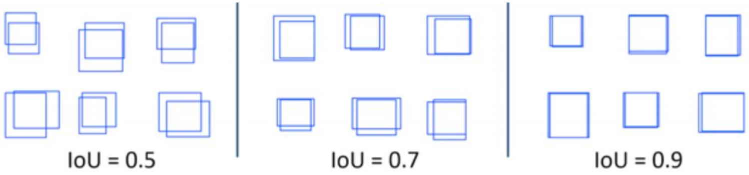
Evaluate

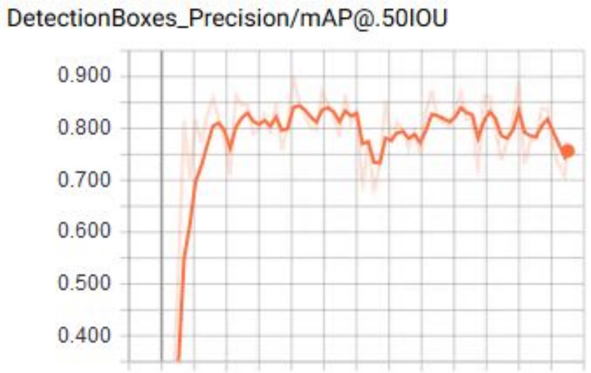
Use TensorBoard to monitor the evaluations

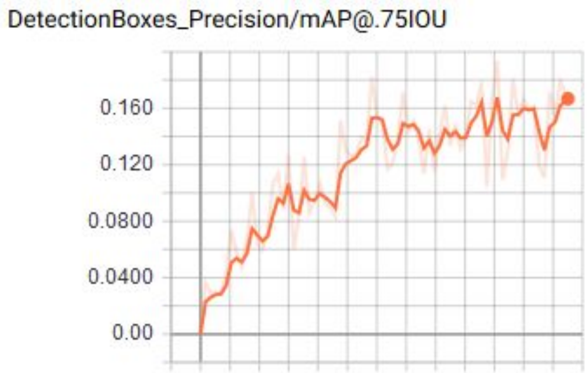


```
tensorboard —logdir=D:\best_10\fold1\eval_fr
```

http://image-net.org/challenges/talks/2016/ECCV2016_workshop_presentation_detection_segmentation.







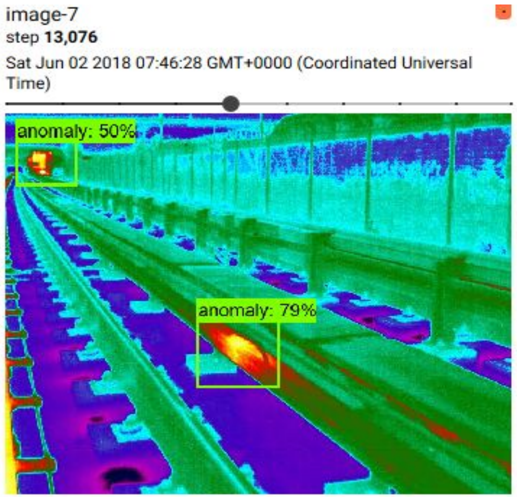
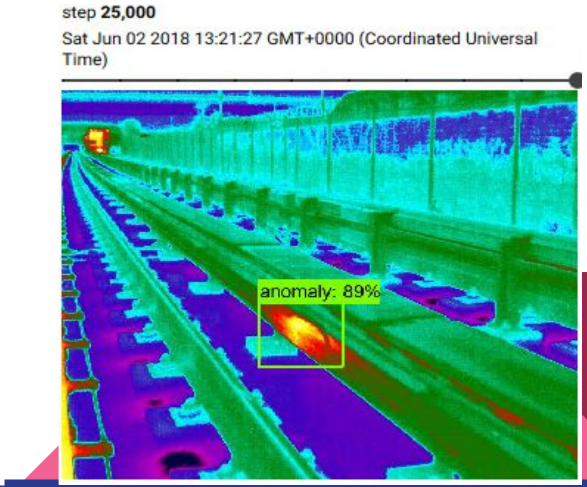


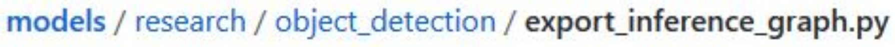
image-7
step **13,076**
Sat Jun 02 2018 07:46:28 GMT+0000 (Coordinated Universal Time)



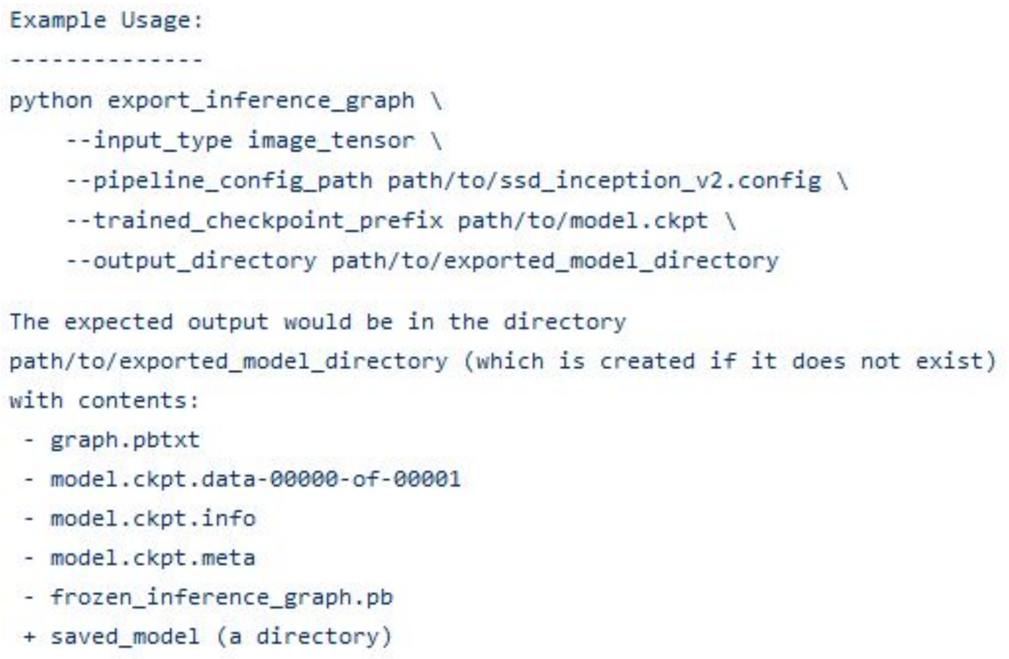
step **25,000**
Sat Jun 02 2018 13:21:27 GMT+0000 (Coordinated Universal Time)

Freeze trained graph

Run python script from command line



models / research / object_detection / export_inference_graph.py



```
Example Usage:
--------------
python export_inference_graph \
    --input_type image_tensor \
    --pipeline_config_path path/to/ssd_inception_v2.config \
    --trained_checkpoint_prefix path/to/model.ckpt \
    --output_directory path/to/exported_model_directory

The expected output would be in the directory
path/to/exported_model_directory (which is created if it does not exist)
with contents:
 - graph.pbtxt
 - model.ckpt.data-00000-of-00001
 - model.ckpt.info
 - model.ckpt.meta
 - frozen_inference_graph.pb
 + saved_model (a directory)
```

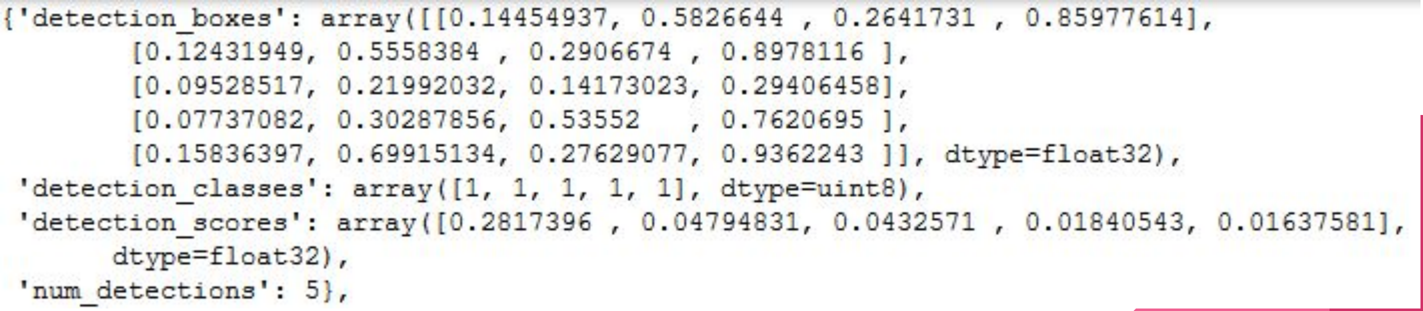
Output

For each picture, detection result output as a python dict

Coordinates of Detection Boxs [X_min, Y_min, X_max, Y_max]

Index of detection class

List of detection scores



```
{'detection_boxes': array([[0.14454937, 0.5826644 , 0.2641731 , 0.85977614],
        [0.12431949, 0.5558384 , 0.2906674 , 0.8978116 ],
        [0.09528517, 0.21992032, 0.14173023, 0.29406458],
        [0.07737082, 0.30287856, 0.53552   , 0.7620695 ],
        [0.15836397, 0.69915134, 0.27629077, 0.9362243 ]], dtype=float32),
 'detection_classes': array([1, 1, 1, 1, 1], dtype=uint8),
 'detection_scores': array([0.2817396 , 0.04794831, 0.0432571 , 0.01840543, 0.01637581],
       dtype=float32),
 'num_detections': 5},
```

Q&A

Many incompatibility issues between software packages and versions,

Stackoverflow and github issue reporting page usually have solutions

Model config samples usually work fine as they come from challenges

No need to mess up with parameters unless you clearly understand the outcome

Set learning rate suitable to your dataset to gain optimal training efficiency

Need multiple trails to gain insights, but grid search is time consuming.