INDEX

Important Links

Tensorflow object detection:

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

hello world object detection

https://github.com/tensorflow/models/blob/master/research/object_detection/object_detection_tutorial.ipynb

distributed training on cloud

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_pets.md

installation

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

Installation

1. Follow the tutorial step by step beginning from the installation.
    a. Protobuf mentioned is 3+ but Ubuntu default is 2.6
    b. https://launchpad.net/~maarten-fonville/+archive/ubuntu/protobuf
    c. sudo add-apt-repository ppa:maarten-fonville/protobuf
    d. sudo apt-get update
    e. then install protobuf-compiler
    f. if it does not work, check this out https://gist.github.com/sofyanhadia/37787e5ed098c97919b8c593f0ec44d8

2. follow the pycocotools installation if running on google cloud, nevermind otherwise

3. always remember to run the following commands on every login at the terminal
    a. protoc object_detection/protos/*.proto --python_out=.
    b. export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
    c. Make sure that there is no issue with the installation by running the following
        i. python object_detection/builders/model_builder_test.py
        ii. (it will show some packages as deprecated and will finish with 15 tests)

Distributed Training

4. Follow the distributed training tutorial step by step (make sure that you have gpu enabled in your google cloud instance)
    a. The command *export YOUR_GCS_BUCKET=${YOUR_GCS_BUCKET}* did not have any result for me, so don't depend on it.
    b. After running the command to create tfrecord files, note that the created files would be
        i. pet_train_with_masks.record instead of the pet_train.record
        ii. pet_val_with_masks.record instead of the pet_val.record
        iii. make sure they are not empty or you might have to do some changes in the code like https://github.com/tensorflow/models/issues/2958
    c. while configuring the object detection pipeline,
        i. do not merely execute the command

```
"sed -i "s|PATH_TO_BE_CONFIGURED|"gs://${YOUR_GCS_BUCKET}"/data|g" \
object_detection/samples/configs/faster_rcnn_resnet101_pets.config
```

        ii. go to the config file to make the following changes
            1. change the name of .record files to match your record files
    d. while submitting jobs to google cloud, try the default, if it does not work
        i. change runtime to 1.3, if it still does not work, 1.4,1.5,1.6 also
        ii. if that does not work, (pycocotools not found error) check this
            1. https://github.com/tensorflow/models/issues/3367
            2. https://stackoverflow.com/questions/48747208/tensorflow-object-detection-evaluation-pycocotools-missing
        iii. if `AttributeError: 'module' object has no attribute 'data`
            1. https://github.com/tensorflow/models/issues/2879
            2. https://github.com/pytorch/pytorch/issues/2656
        iv. If this does not work,
            1. https://github.com/tensorflow/models/issues/4002
            2. https://github.com/tensorflow/models/issues/4058
            3. https://github.com/tensorflow/models/issues/3937
        v. If it still does not work,
            1. Check the open issue https://github.com/tensorflow/models/issues/3071
            2. Placeholder for 1.2 runtime supporting github commit

Running Locally

5. To run the jobs locally https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_locally.md
    a. Complete the installation, create tfrecord files and create pipeline
        i. To create a pipeline, merely edit the sample config file and copy it to the tensorflow/models/research/object_detection_models/model

`object_detection/samples/configs/faster_rcnn_resnet101_pets.config`

        ii. Make sure the directory structure is as shown in the tutorial and edit the paths of the files in the config file
            1. Prepend a './ ' to the path and make sure there are no spaces in the path
            2. Edit the names of the files, append *_with_masks* to the names
            3. Check my sample config file below

6. Run Jobs
    a. a Training Job
        i. python train.py --logtostderr --train_dir=./models/model/train --pipeline_config_path=./models/model/faster_rcnn_resnet101_pets.config
            1. make sure that you include the train-dir flag before the pipeline_config flag else you'll have errors
            2. if all your GPU's are enabled, they'll get filled now
        ii. if you get the following error,
            1. hape[1,600,901,3] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU_0_bfc
                a. That means your gpu is already processing something else.

        iii.
        iv. aas
            1. Resource exhausted: OOM when allocating tensor with s

        v.
    b. Eval Job
        1. ```
           # From the tensorflow/models/research/object_detection directory
           python eval.py --logtostderr --pipeline_config_path=-checkpoint_dir=${PATH_TO_TRAIN_DIR} --eval_dir=${PATH_TO_EVAL_DIR}
           ```
           ```

2. `Example:`

   `python eval.py --logtostderr`

   `--pipeline_config_path=./models/model/faster_rcnn_resnet10`

   `1_pets.config --checkpoint_dir=./models/model/train`

   `--eval_dir=./models/model/eval`

3.

ii. Note: Eval job will most likely fail so one of two things should be done

1. Either add Gpu directives and run training job and eval job separately
   a. https://github.com/tensorflow/models/issues/1854
   b. which did not work for me, ultimately switched to cpu
2. or run in a virtual environment
3.

iii. Running Tensorboard

c. Tensorboard Job

i. pip install --update tensorboard

ii. `tensorboard --logdir=${PATH_TO_MODEL_DIRECTORY}`

iii. which in my case was --logdir=./models/model

d. visit 127.0.0.1:6006

Docker

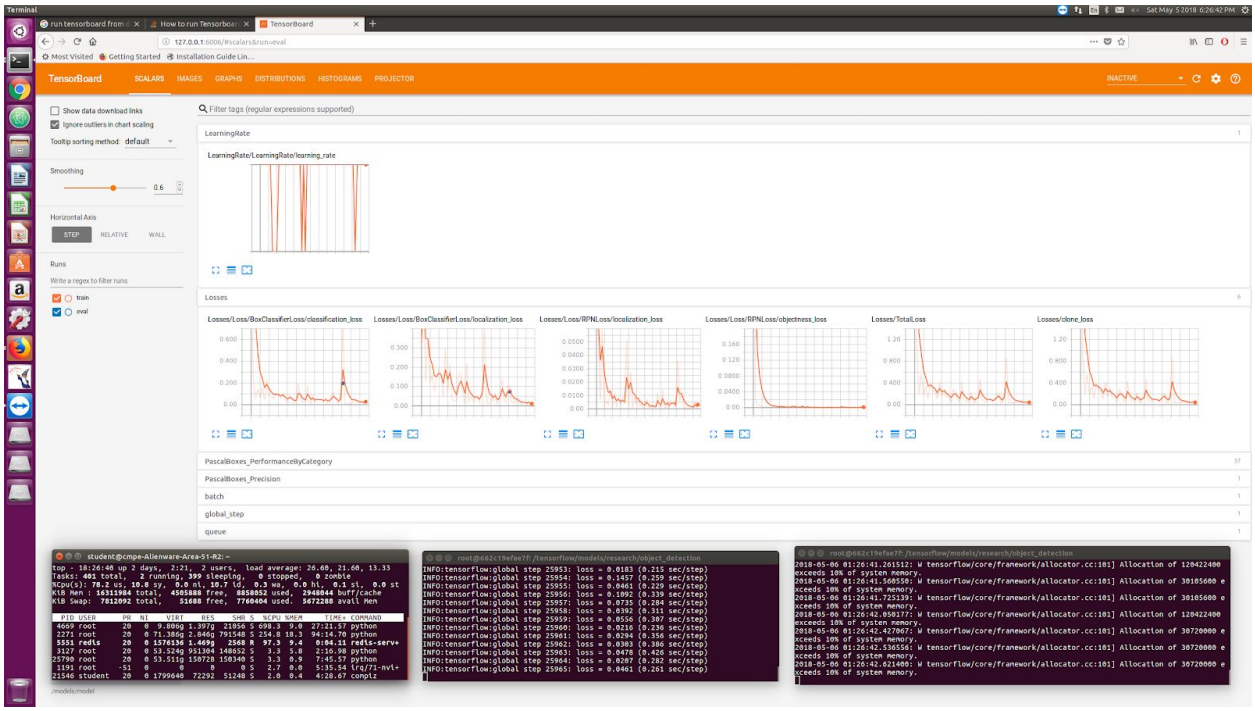
7. Make your own image
8. Use mine
    a. Make sure that You have nvidia-docker https://github.com/NVIDIA/nvidia-docker installed as per the latest version on github
    b. Make sure you are running Ubuntu natively on hardware and not in a VM, else enable GPU pass through
        i. Either install in a USB and make persistent and use Nvidia-docker to create a docker container
        ii. or install Ubuntu in a HDD and add an entry to the windows boot manager after you disable the secure boot in bios
    C. `make sure the the command shows your GPU's`

        ```
        docker run --runtime=nvidia --rm nvidia/cuda nvidia-smi
        ```

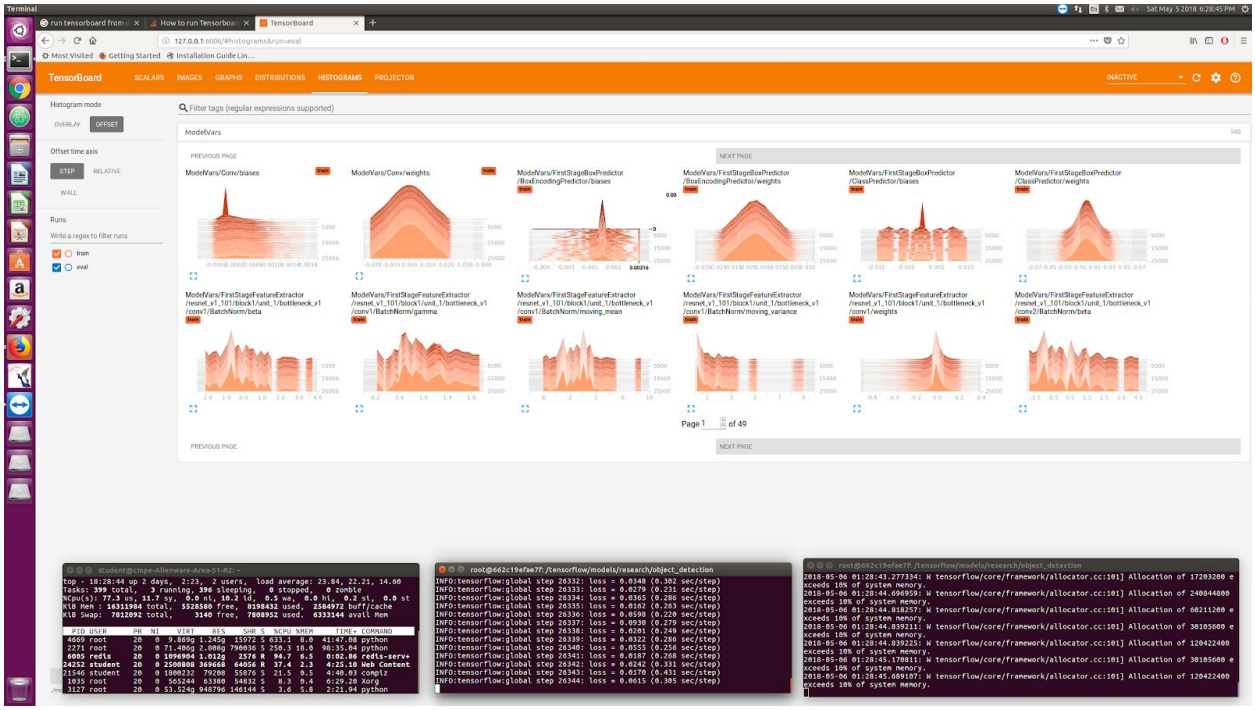
    d. Run the following commands
        i. docker login
        ii. docker pull tfgpu/od:1
        iii. nvidia-docker run -it -p 6006:6006 -p 8888:8888 -d tfgpu/od:1
        iv. docker
        v. docker ps (and copy the docke id from the output)
        vi. docker exec -it _*image_id*_ bash
        vii. python train.py --logtostderr --train_dir=./models/model/train --pipeline_config_file=./models/model/faster_rcnn_resnet101_pets.config
        viii. in another terminal, docker exec -it _image_id_ bash
        ix. python eval.py --logtostderr --pipeline_config_path=./models/model/faster_rcnn_resnet101_pets.config --checkpoint_dir=./models/model/train --eval_dir=./models/model/eval
        x. in yet another terminal, docker exec -it _image_id_ bash
        xi. tensorboard --logdir ./models/model/
        xii. when you want to save the changes, exit
        xiii. docker tag _*image_id*_ username/repo:tag
        xiv. docker commit _*image_id*_ -p username/repo:tag
    e.
9. a

Output

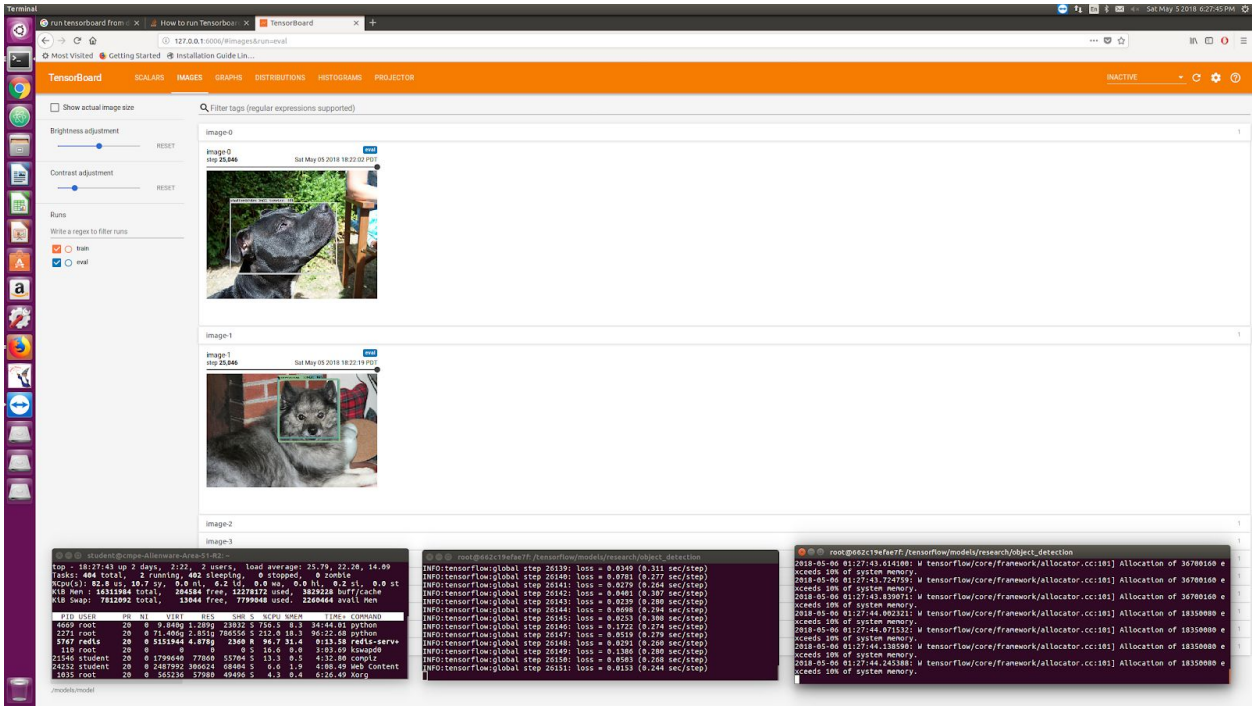
1. Scalars



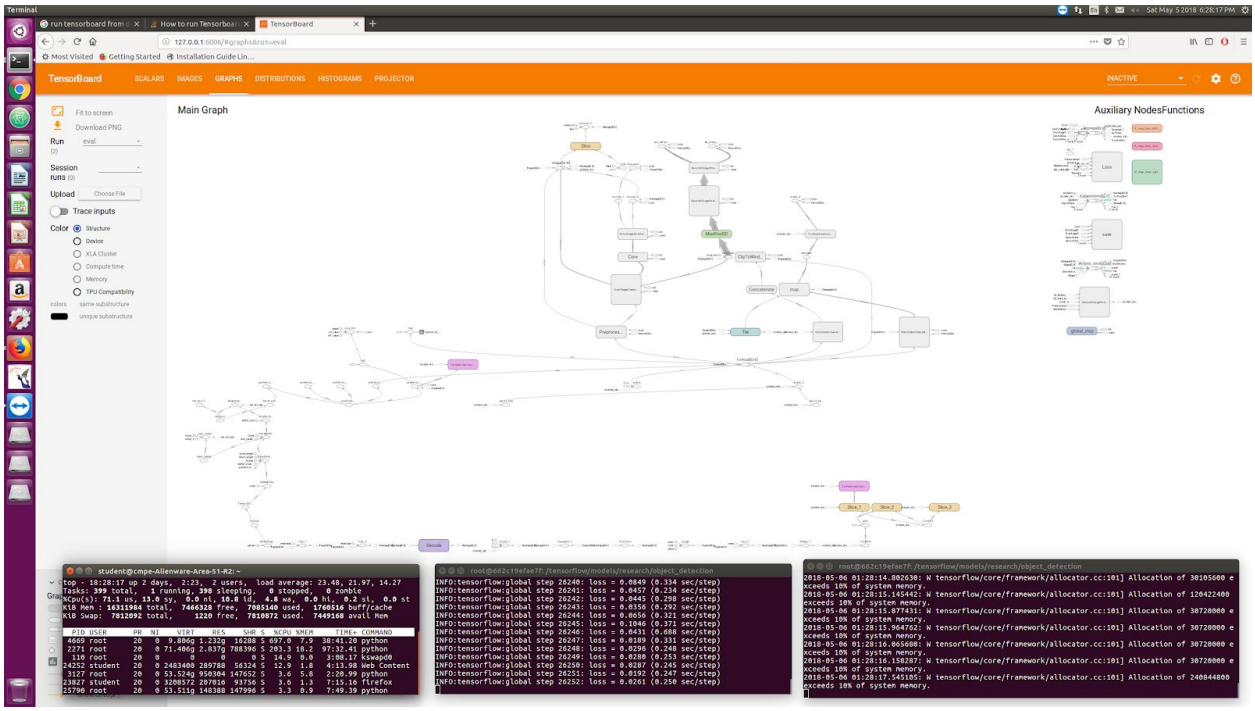
2. Histograms



3. Images



4. Graphs



Sample config file

```
model {
 faster_rcnn {
  num_classes: 37
  image_resizer {
   keep_aspect_ratio_resizer {
    min_dimension: 600
    max_dimension: 1024
   }
  }
  feature_extractor {
   type: 'faster_rcnn_resnet101'
   first_stage_features_stride: 16
  }
  first_stage_anchor_generator {
   grid_anchor_generator {
    scales: [0.25, 0.5, 1.0, 2.0]
    aspect_ratios: [0.5, 1.0, 2.0]
    height_stride: 16
    width_stride: 16
   }
  }
  first_stage_box_predictor_conv_hyperparams {
   op: CONV
   regularizer {
    l2_regularizer {
     weight: 0.0
    }
   }
```

```
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
}
    }
first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight: 2.0
    first_stage_objectness_loss_weight: 1.0
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        use_dropout: false
        dropout_keep_probability: 1.0
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0
            }
          }
          initializer {
            variance_scaling_initializer {
              factor: 1.0
              uniform: true
              mode: FAN_AVG
```

```
          }
        }
      }
    }
  }
  second_stage_post_processing {
    batch_non_max_suppression {
      score_threshold: 0.0
      iou_threshold: 0.6
      max_detections_per_class: 100
      max_total_detections: 300
    }
    score_converter: SOFTMAX
  }
  second_stage_localization_loss_weight: 2.0
  second_stage_classification_loss_weight: 1.0

}
}

train_config: {
 batch_size: 1
 optimizer {
  momentum_optimizer: {
   learning_rate: {
    manual_step_learning_rate {
     initial_learning_rate: 0.0003
     schedule {
      step: 900000
      learning_rate: .00003
```

```
        }
        schedule {
          step: 1200000
          learning_rate: .000003
        }
      }
    }
    momentum_optimizer_value: 0.9
  }
  use_moving_average: false
 }
 gradient_clipping_by_norm: 10.0
 fine_tune_checkpoint: "./data/model.ckpt"
 from_detection_checkpoint: true
 # Note: The below line limits the training process to 200K steps, which we
 # empirically found to be sufficient enough to train the pets dataset. This
 # effectively bypasses the learning rate schedule (the learning rate will
 # never decay). Remove the below line to train indefinitely.
 num_steps: 200000
 data_augmentation_options {
  random_horizontal_flip {
  }
 }
}

train_input_reader: {
 tf_record_input_reader {
input_path: "./data/pet_train_with_masks.record"
 }
 label_map_path: "./data/pet_label_map.pbtxt"
```

```
}

eval_config: {
  num_examples: 2000
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "./data/pet_val_with_masks.record"
  }
  label_map_path: "./data/pet_label_map.pbtxt"
  shuffle: false
  num_readers: 1
}
```