Training and Evaluation with TensorFlow 2

This page walks through the steps required to train an object detection model. It assumes the reader has completed the following prerequisites:

1. The TensorFlow Object Detection API has been installed as documented in the installation instructions.
2. A valid data set has been created. See this page for instructions on how to generate a dataset for the PASCAL VOC challenge or the Oxford-IIIT Pet dataset.

Recommended Directory Structure for Training and Evaluation

```
.
├── data/
│   ├── eval-00000-of-00001.tfrecord
│   ├── label_map.txt
│   ├── train-00000-of-00002.tfrecord
│   └── train-00001-of-00002.tfrecord
└── models/
    └── my_model_dir/
        ├── eval/                 # Created by evaluation job.
        ├── my_model.config
        └── model_ckpt-100-data@1 #
        └── model_ckpt-100-index  # Created by training job.
        └── checkpoint            #
```

Writing a model configuration

Please refer to sample TF2 configs and configuring jobs to create a model config.

Model Parameter Initialization

While optional, it is highly recommended that users utilize classification or object detection checkpoints. Training an object detector from scratch can take days. To speed up the training process, it is recommended that users re-use the feature extractor parameters from a pre-existing image classification or object detection checkpoint. The `train_config` section in the config provides two fields to specify pre-existing checkpoints:

- `fine_tune_checkpoint` : a path prefix to the pre-existing checkpoint (ie:"/usr/home/username/checkpoint/model.ckpt-#####").
- `fine_tune_checkpoint_type` : with value `classification` or `detection` depending on the type.

A list of classification checkpoints can be found here

A list of detection checkpoints can be found here.

Local

Training

A local training job can be run with the following command:

```
# From the tensorflow/models/research/ directory
PIPELINE_CONFIG_PATH={path to pipeline config file}
MODEL_DIR={path to model directory}
python object_detection/model_main_tf2.py \
    --pipeline_config_path=${PIPELINE_CONFIG_PATH} \
    --model_dir=${MODEL_DIR} \
    --alsologtostderr
```

where `${PIPELINE_CONFIG_PATH}` points to the pipeline config and `${MODEL_DIR}` points to the directory in which training checkpoints and events will be written.

Evaluation

A local evaluation job can be run with the following command:

```
# From the tensorflow/models/research/ directory
PIPELINE_CONFIG_PATH={path to pipeline config file}
MODEL_DIR={path to model directory}
CHECKPOINT_DIR=${MODEL_DIR}
MODEL_DIR={path to model directory}
python object_detection/model_main_tf2.py \
    --pipeline_config_path=${PIPELINE_CONFIG_PATH} \
    --model_dir=${MODEL_DIR} \
    --checkpoint_dir=${CHECKPOINT_DIR} \
    --alsologtostderr
```

where `${CHECKPOINT_DIR}` points to the directory with checkpoints produced by the training job. Evaluation events are written to `${MODEL_DIR/eval}`

Google Cloud VM

The TensorFlow Object Detection API supports training on Google Cloud with Deep Learning GPU VMs and TPU VMs. This section documents instructions on how to train and evaluate your model on them. The reader should complete the following prerequistes:

1. The reader has create and configured a GPU VM or TPU VM on Google Cloud with TensorFlow >= 2.2.0. See TPU quickstart and GPU quickstart
2. The reader has installed the TensorFlow Object Detection API as documented in the installation instructions on the VM.
3. The reader has a valid data set and stored it in a Google Cloud Storage bucket or locally on the VM. See this page for instructions on how to generate a dataset for the PASCAL VOC challenge or the Oxford-IIIT Pet dataset.

Additionally, it is recommended users test their job by running training and evaluation jobs for a few iterations locally on their own machines.

Training

Training on GPU or TPU VMs is similar to local training. It can be launched using the following command.

```
# From the tensorflow/models/research/ directory
USE_TPU=true
TPU_NAME="MY_TPU_NAME"
PIPELINE_CONFIG_PATH={path to pipeline config file}
MODEL_DIR={path to model directory}
python object_detection/model_main_tf2.py \
    --pipeline_config_path=${PIPELINE_CONFIG_PATH} \
    --model_dir=${MODEL_DIR} \
    --use_tpu=${USE_TPU} \  # (optional) only required for TPU training.
    --tpu_name=${TPU_NAME} \  # (optional) only required for TPU training.
    --alsologtostderr
```

where `${PIPELINE_CONFIG_PATH}` points to the pipeline config and `${MODEL_DIR}` points to the root directory for the files produces. Training checkpoints and events are written to `${MODEL_DIR}`. Note that the paths can be either local or a path to GCS bucket.

Evaluation

Evaluation is only supported on GPU. Similar to local evaluation it can be launched using the following command:

```
# From the tensorflow/models/research/ directory
PIPELINE_CONFIG_PATH={path to pipeline config file}
MODEL_DIR={path to model directory}
CHECKPOINT_DIR=${MODEL_DIR}
MODEL_DIR={path to model directory}
python object_detection/model_main_tf2.py \
    --pipeline_config_path=${PIPELINE_CONFIG_PATH} \
    --model_dir=${MODEL_DIR} \
    --checkpoint_dir=${CHECKPOINT_DIR} \
    --alsologtostderr
```

where `${CHECKPOINT_DIR}` points to the directory with checkpoints produced by the training job. Evaluation events are written to `${MODEL_DIR/eval}`. Note that the paths can be either local or a path to GCS bucket.

Google Cloud AI Platform

The TensorFlow Object Detection API supports also supports training on Google Cloud AI Platform. This section documents instructions on how to train and evaluate your model using Cloud ML. The reader should complete the following prerequistes:

1. The reader has created and configured a project on Google Cloud AI Platform. See Using GPUs and Using TPUs guides.
2. The reader has a valid data set and stored it in a Google Cloud Storage bucket. See this page for instructions on how to generate a dataset for the PASCAL VOC challenge or the Oxford-IIIT Pet dataset.

Additionally, it is recommended users test their job by running training and evaluation jobs for a few iterations locally on their own machines.

Training with multiple GPUs

A user can start a training job on Cloud AI Platform following the instruction https://cloud.google.com/ai-platform/training/docs/custom-containers-training.

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

# From the tensorflow/models/research/ directory
cp object_detection/dockerfiles/tf2_ai_platform/Dockerfile .

docker build -t gcr.io/${DOCKER_IMAGE_URI} .

docker push gcr.io/${DOCKER_IMAGE_URI}
```

```
gcloud ai-platform jobs submit training object_detection_`date +%m_%d_%Y_%H_%M_%S` \
    --job-dir=gs://${MODEL_DIR} \
    --region us-central1 \
    --master-machine-type n1-highcpu-16 \
    --master-accelerator count=8,type=nvidia-tesla-v100 \
    --master-image-uri gcr.io/${DOCKER_IMAGE_URI} \
    --scale-tier CUSTOM \
    -- \
    --model_dir=gs://${MODEL_DIR} \
    --pipeline_config_path=gs://${PIPELINE_CONFIG_PATH}
```

Where `gs://${MODEL_DIR}` specifies the directory on Google Cloud Storage where the training checkpoints and events will be written to and `gs://${PIPELINE_CONFIG_PATH}` points to the pipeline configuration stored on Google Cloud Storage, and `gcr.io/${DOCKER_IMAGE_URI}` points to the docker image stored in Google Container Registry.

Users can monitor the progress of their training job on the ML Engine Dashboard.

Training with TPU

Launching a training job with a TPU compatible pipeline config requires using the following command:

```
# From the tensorflow/models/research/ directory
cp object_detection/packages/tf2/setup.py .
gcloud ai-platform jobs submit training `whoami`_object_detection_`date
+%m_%d_%Y_%H_%M_%S` \
    --job-dir=gs://${MODEL_DIR} \
    --package-path ./object_detection \
    --module-name object_detection.model_main_tf2 \
    --runtime-version 2.1 \
    --python-version 3.6 \
    --scale-tier BASIC_TPU \
    --region us-central1 \
    -- \
    --use_tpu true \
```

```
    --model_dir=gs://${MODEL_DIR} \
    --pipeline_config_path=gs://${PIPELINE_CONFIG_PATH}
```

As before `pipeline_config_path` points to the pipeline configuration stored on Google Cloud Storage (but is now must be a TPU compatible model).

Evaluating with GPU

Evaluation jobs run on a single machine. Run the following command to start the evaluation job:

```
gcloud ai-platform jobs submit training object_detection_eval_`date
+%m_%d_%Y_%H_%M_%S` \
    --job-dir=gs://${MODEL_DIR} \
    --region us-central1 \
    --scale-tier BASIC_GPU \
    --master-image-uri gcr.io/${DOCKER_IMAGE_URI} \
    -- \
    --model_dir=gs://${MODEL_DIR} \
    --pipeline_config_path=gs://${PIPELINE_CONFIG_PATH} \
    --checkpoint_dir=gs://${MODEL_DIR}
```

where `gs://${MODEL_DIR}` points to the directory on Google Cloud Storage where training checkpoints are saved and `gs://{PIPELINE_CONFIG_PATH}` points to where the model configuration file stored on Google Cloud Storage, and `gcr.io/${DOCKER_IMAGE_URI}` points to the docker image stored in Google Container Registry. Evaluation events are written to `gs://${MODEL_DIR}/eval`

Typically one starts an evaluation job concurrently with the training job. Note that we do not support running evaluation on TPU.

Running Tensorboard

Progress for training and eval jobs can be inspected using Tensorboard. If using the recommended directory structure, Tensorboard can be run using the following command:

```
tensorboard --logdir=${MODEL_DIR}
```

where `${MODEL_DIR}` points to the directory that contains the train and eval directories. Please note it may take Tensorboard a couple minutes to populate with data.