

Running DeepLab on Cityscapes Semantic Segmentation Dataset

This page walks through the steps required to run DeepLab on Cityscapes on a local machine.

Download dataset and convert to TFRecord

We have prepared the script (under the folder `datasets`) to convert Cityscapes dataset to TFRecord. The users are required to download the dataset beforehand by registering the [website](#).

```
# From the tensorflow/models/research/deeplab/datasets directory.
sh convert_cityscapes.sh
```

The converted dataset will be saved at `./deeplab/datasets/cityscapes/tfrecord`.

Recommended Directory Structure for Training and Evaluation

```
+ datasets
+ cityscapes
+ leftImg8bit
+ gtFine
+ tfrecord
+ exp
+ train_on_train_set
+ train
+ eval
+ vis
```

where the folder `train_on_train_set` stores the train/eval/vis events and results (when training DeepLab on the Cityscapes train set).

Running the train/eval/vis jobs

A local training job using `xception_65` can be run with the following command:

```
# From tensorflow/models/research/
python deeplab/train.py \
  --logtostderr \
  --training_number_of_steps=90000 \
  --train_split="train_fine" \
  --model_variant="xception_65" \
  --atrous_rates=6 \
  --atrous_rates=12 \
  --atrous_rates=18 \
  --output_stride=16 \
  --decoder_output_stride=4 \
  --train_crop_size="769,769" \
  --train_batch_size=1 \
```

```
--dataset="cityscapes" \
--tf_initial_checkpoint=${PATH_TO_INITIAL_CHECKPOINT} \
--train_logdir=${PATH_TO_TRAIN_DIR} \
--dataset_dir=${PATH_TO_DATASET}
```

where `${PATH_TO_INITIAL_CHECKPOINT}` is the path to the initial checkpoint (usually an ImageNet pretrained checkpoint), `${PATH_TO_TRAIN_DIR}` is the directory in which training checkpoints and events will be written to, and `${PATH_TO_DATASET}` is the directory in which the Cityscapes dataset resides.

Note that for `{train,eval,vis}.py`:

1. In order to reproduce our results, one needs to use large batch size (> 8), and set `fine_tune_batch_norm = True`. Here, we simply use small batch size during training for the purpose of demonstration. If the users have limited GPU memory at hand, please fine-tune from our provided checkpoints whose batch norm parameters have been trained, and use smaller learning rate with `fine_tune_batch_norm = False`.
2. The users should change `atrous_rates` from `[6, 12, 18]` to `[12, 24, 36]` if setting `output_stride=8`.
3. The users could skip the flag, `decoder_output_stride`, if you do not want to use the decoder structure.
4. Change and add the following flags in order to use the provided dense prediction cell. Note we need to set `decoder_output_stride` if you want to use the provided checkpoints which include the decoder module.

```
--model_variant="xception_71"
--
dense_prediction_cell_json="deeplab/core/dense_prediction_cell_branch5_top1_cityscapes
--decoder_output_stride=4
```

A local evaluation job using `xception_65` can be run with the following command:

```
# From tensorflow/models/research/
python deeplab/eval.py \
  --logtostderr \
  --eval_split="val_fine" \
  --model_variant="xception_65" \
  --atrous_rates=6 \
  --atrous_rates=12 \
  --atrous_rates=18 \
  --output_stride=16 \
  --decoder_output_stride=4 \
  --eval_crop_size="1025,2049" \
  --dataset="cityscapes" \
  --checkpoint_dir=${PATH_TO_CHECKPOINT} \
  --eval_logdir=${PATH_TO_EVAL_DIR} \
  --dataset_dir=${PATH_TO_DATASET}
```

where `${PATH_TO_CHECKPOINT}` is the path to the trained checkpoint (i.e., the path to `train_logdir`), `${PATH_TO_EVAL_DIR}` is the directory in which evaluation events will be written to, and `${PATH_TO_DATASET}` is the directory in which the Cityscapes dataset resides.

A local visualization job using `xception_65` can be run with the following command:

```
# From tensorflow/models/research/
python deeplab/vis.py \
  --logtostderr \
  --vis_split="val_fine" \
  --model_variant="xception_65" \
  --atrous_rates=6 \
  --atrous_rates=12 \
  --atrous_rates=18 \
  --output_stride=16 \
  --decoder_output_stride=4 \
  --vis_crop_size="1025,2049" \
  --dataset="cityscapes" \
  --colormap_type="cityscapes" \
  --checkpoint_dir=${PATH_TO_CHECKPOINT} \
  --vis_logdir=${PATH_TO_VIS_DIR} \
  --dataset_dir=${PATH_TO_DATASET}
```

where `${PATH_TO_CHECKPOINT}` is the path to the trained checkpoint (i.e., the path to `train_logdir`), `${PATH_TO_VIS_DIR}` is the directory in which evaluation events will be written to, and `${PATH_TO_DATASET}` is the directory in which the Cityscapes dataset resides. Note that if the users would like to save the segmentation results for evaluation server, set `also_save_raw_predictions = True`.

Running Tensorboard

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

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

where `${PATH_TO_LOG_DIRECTORY}` points to the directory that contains the train, eval, and vis directories (e.g., the folder `train_on_train_set` in the above example). Please note it may take Tensorboard a couple minutes to populate with data.