# Quantize DeepLab model for faster on-device inference

This page describes the steps required to quantize DeepLab model and convert it to TFLite for on-device inference. The main steps include:

1. Quantization-aware training
2. Exporting model
3. Converting to TFLite FlatBuffer

We provide details for each step below.

## Quantization-aware training

DeepLab supports two approaches to quantize your model.

1. **[Recommended]** Training a non-quantized model until convergence. Then fine-tune the trained float model with quantization using a small learning rate (on PASCAL we use the value of 3e-5) . This fine-tuning step usually takes 2k to 5k steps to converge.

2. Training a deeplab float model with delayed quantization. Usually we delay quantization until the last a few thousand steps in training.

In the current implementation, quantization is only supported with 1) `num_clones=1` for training and 2) single scale inference for evaluation, visualization and model export. To get the best performance for the quantized model, we strongly recommend to train the float model with larger `num_clones` and then fine-tune the model with a single clone.

Here shows the commandline to quantize deeplab model trained on PASCAL VOC dataset using fine-tuning:

```
# From tensorflow/models/research/
python deeplab/train.py \
    --logtostderr \
    --training_number_of_steps=3000 \
    --train_split="train" \
    --model_variant="mobilenet_v2" \
    --output_stride=16 \
    --train_crop_size="513,513" \
    --train_batch_size=8 \
    --base_learning_rate=3e-5 \
    --dataset="pascal_voc_seg" \
    --quantize_delay_step=0 \
    --tf_initial_checkpoint=${PATH_TO_TRAINED_FLOAT_MODEL} \
    --train_logdir=${PATH_TO_TRAIN_DIR} \
    --dataset_dir=${PATH_TO_DATASET}
```

## Converting to TFLite FlatBuffer

First use the following commandline to export your trained model.

```
# From tensorflow/models/research/
python deeplab/export_model.py \
    --checkpoint_path=${CHECKPOINT_PATH} \
    --quantize_delay_step=0 \
    --export_path=${OUTPUT_DIR}/frozen_inference_graph.pb
```

Commandline below shows how to convert exported graphdef to TFlite model.

```
# From tensorflow/models/research/
python deeplab/convert_to_tflite.py \
  --quantized_graph_def_path=${OUTPUT_DIR}/frozen_inference_graph.pb \
  --input_tensor_name=MobilenetV2/MobilenetV2/input:0 \
  --output_tflite_path=${OUTPUT_DIR}/frozen_inference_graph.tflite \
  --test_image_path=${PATH_TO_TEST_IMAGE}
```

**[Important]** Note that converted model expects 513x513 RGB input and doesn't include preprocessing (resize and pad input image) and post processing (crop padded region and resize to original input size). These steps can be implemented outside of TFlite model.

## Quantized model on PASCAL VOC

We provide float and quantized checkpoints that have been pretrained on VOC 2012 train_aug set, using MobileNet-v2 backbone with different depth multipliers. Quantized model usually have 1% decay in mIoU.

For quantized (8bit) model, un-tar'ed directory includes:

- a frozen inference graph (frozen_inference_graph.pb)
- a checkpoint (model.ckpt.data*, model.ckpt.index)
- a converted TFlite FlatBuffer file (frozen_inference_graph.tflite)

| Checkpoint name | Eval OS | Eval scales | Left-right Flip | Multiply-Adds | Quantize | PASCAL mIOU |
|---|---|---|---|---|---|---|
| mobilenetv2_dm05_coco_voc_trainaug | 16 | [1.0] | No | 0.88B | No | 70.19% (val) |
| mobilenetv2_dm05_coco_voc_trainaug_8bit | 16 | [1.0] | No | 0.88B | Yes | 69.65% (val) |
| mobilenetv2_coco_voc_trainaug | 16 | [1.0] | No | 2.75B | No | 75.32% (val) |
| mobilenetv2_coco_voc_trainaug_8bit | 16 | [1.0] | No | 2.75B | Yes | 74.26% (val) |

Note that you might need the nightly build of TensorFlow (see here for install instructions) to convert above quantized model to TFLite.