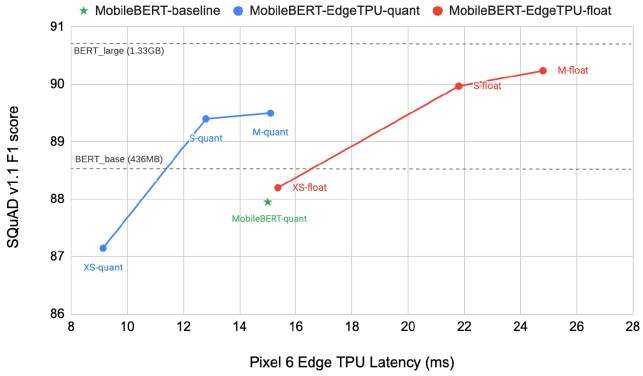
MobileBERT-EdgeTPU



Performance of MobileBERT-EdgeTPU models on the SQuAD v1.1 dataset.

Note: For MobileBERT baseline float model, NNAPI delegates parts of the computing ops to CPU, making the latency much higher.

Note: The accuracy numbers for BERT_base and BERT_large are from the training results. These models are too large and not feasible to run on device.

Deploying low-latency, high-quality transformer based language models on device is highly desirable, and can potentially benefit multiple applications such as automatic speech recognition (ASR), translation, sentence autocompletion, and even some vision tasks. By co-designing the neural networks with the Edge TPU hardware accelerator in Google Tensor SoC, we have built EdgeTPU-customized MobileBERT models that demonstrate datacenter model quality meanwhile outperforms baseline MobileBERT's latency.

We set up our model architecture search space based on MobileBERT and leverage AutoML algorithms to find models with up to 2x better hardware utilization. With higher utilization, we are able to bring larger and more accurate models on chip, and meanwhile the models can still outperform the baseline MobileBERT latency. We built a customized distillation training pipeline and performed exhaustive hyperparameters (e.g. learning rate, dropout ratio, etc) search to achieve the best accuracy. As shown in the above figure, the quantized MobileBERT-EdgeTPU models establish a new pareto-frontier for the question answering tasks and also exceed the accuracy of the float BERT_base model which is 400+MB and too large to run on edge devices.

We also observed that, unlike most vision models, the accuracy drops significantly for MobileBERT/MobileBERT-EdgeTPU with plain post training quantization (PTQ) or quantization aware training (QAT). Proper model modifications, such as clipping the mask value, are necessary to retain the accuracy for a quantized model. Therefore, as an alternative to the quant models, we also provide a set of Edge TPU friendly float models which also produce a better (though marginally) roofline than the baseline MobileBERT quant model. Notably, the float MobileBERT-EdgeTPU-M model yields accuracy that is even close to the BERT_large, which has 1.3GB model size in float precision. Quantization now becomes an optional optimization rather than a prerequisite, which can greatly benefit/unblock some use cases where quantization is infeasible or introduce large accuracy deterioration, and potentially reduce the time-to-market.

Pre-trained Models

Note: the SQUAD score is measured with SQUAD V1.1 dataset by adding the BertSpanLabeler task head.

Model name (checkpoint)	# Parameters	MLM	SQUAD (float)	SQUAD (int8)	TFhub link
MobileBERT (baseline)	24.6M	71.4%	89.02%	87.95%	n/a
MobileBERT-EdgeTPU-XS	27.1M	71.2%	88.20%	87.15%	link
MobileBERT-EdgeTPU-S	38.3M	72.8%	89.97%	89.40%	link
MobileBERT-EdgeTPU-M	50.9M	73.8%	90.24%	89.50%	link

Restoring from Checkpoints

To load the pre-trained MobileBERT checkpoint in your code, please follow the example below or check the `serving/export_tflite_squad` module:

```
import tensorflow as tf
from official.nlp.projects.mobilebert_edgetpu import params

bert_config_file = ...
model_checkpoint_path = ...

# Set up experiment params and load the configs from file/files.
experiment_params = params.EdgeTPUBERTCustomParams()

# change the input mask type to tf.float32 to avoid additional casting op.
experiment_params.student_model.encoder.mobilebert.input_mask_dtype = 'float32'
pretrainer_model = model_builder.build_bert_pretrainer(
    experiment_params.student_model,
    name='pretrainer',
    quantization_friendly=True)

checkpoint_dict = {'model': pretrainer_model}
checkpoint = tf.train.Checkpoint(**checkpoint_dict)
checkpoint.restore(FLAGS.model_checkpoint).assert_existing_objects_matched()
```

Use TF-Hub models

In addition to the checkpoint, MobileBERT-EdgeTPU models are also available in Tensorflow Hub. To use the models for finetuning on a downstream task (e.g. Question Answering):

```
import tensorflow as tf
import tensorflow_hub as hub
from official.nlp.modeling import models

encoder_network = hub.KerasLayer(
    'https://tfhub.dev/google/edgetpu/nlp/mobilebert-edgetpu/s/1',
    trainable=True)
squad_model = models.BertSpanLabeler(
    network=encoder_network,
    initializer=tf.keras.initializers.TruncatedNormal(stddev=0.01))
```