TensorFlow Model Optimization Toolkit — Post-Training Integer Quantization

TensorFlow
Jun 12, 2019 · 5 min read

Posted by the TensorFlow Model Optimization Team

Since we introduced the Model Optimization Toolkit — a suite of techniques that both novice and advanced developers can use to optimize machine learning models for deployment and execution — we have been working hard to reduce the complexity of quantizing machine learning models.

Initially, we supported post-training quantization via “hybrid operations”, which is quantizing the parameters of the model (i.e. weights), but allowing certain parts of the computation to take place in floating point. Today, we are happy to announce the next addition to our tooling: post-training integer quantization. Integer quantization is a general technique that reduces the numerical precision of the weights and activations of models to reduce memory and improve latency.

Quantize models to reduce size, latency, and power consumption with negligible accuracy loss

Why you should use post-training integer quantization

Our previously released “hybrid” post training quantization approach reduced the model size and latency in many cases, but it has the limitation of requiring floating point computation, which may not be available in all hardware accelerators (i.e. Edge TPUs), but makes it suitable for CPU.

Our new post-training integer quantization enables users to take an already-trained floating-point model and fully quantize it to only use 8-bit signed integers (i.e. `int8`). By leveraging this quantization scheme, we can get reasonable quantized model accuracy across many models without resorting to retraining a model with quantization-aware training. With this new tool, models will continue to be 4x smaller, but will see even greater CPU speed-ups. Fixed point hardware accelerators, such as Edge TPUs, will also be able to run these models.

Compared to quantization-aware training, this tool is much simpler to use, and offers comparable accuracy on most models. There may still be use cases where quantization-aware training is required, but we expect this to be rare as we continue to improve post-training tooling.

In summary, a user should use “hybrid” post training quantization when targeting simple CPU size and latency improvements. When targeting greater CPU improvements or fixed-point accelerators, they should use this integer post training quantization tool, potentially using quantization-aware training if accuracy of a model suffers.

How to enable post-training integer quantization

Our integer quantization tool requires a small calibration set of representative data. By simply providing the representative_dataset generator to the converter, the optimization parameter will perform integer quantization on the input model.

Is the model entirely quantized?

Just like the existing post-training quantization functionality, by default, the operations ("ops") that do not have quantized implementations will automatically be left in floating point. This allows conversion to occur smoothly, and will produce a model that will always execute on a typical mobile CPU — consider that TensorFlow Lite will execute the integer operations in the integer-only accelerator, falling back to CPU for the operations involving floating point. To execute entirely on specialized hardware that does not support floating point operations at all (for example, some machine learning accelerators, including the Edge TPU), you can specify a flag in order to output only integer operations:

When this flag is used and an operation has no integer quantizable counterpart, the TensorFlow Lite Converter will throw an error.

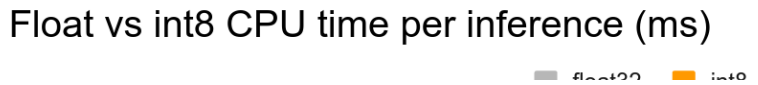
Very little data is needed

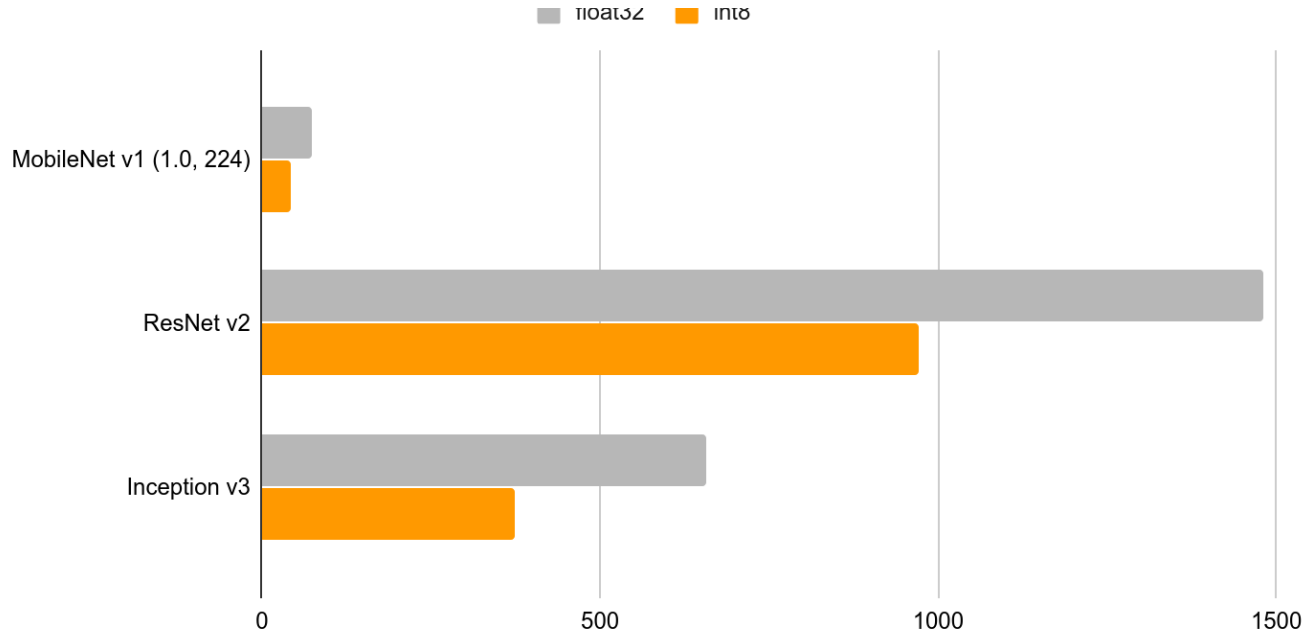
In our experiments, we found that a few dozen examples that are representative of what the model will see during execution are sufficient to get the best accuracy. For instance the accuracy numbers below are from models calibrated on only 100 images from the ImageNet dataset.

Results

Latency

Compared to their float counterparts, quantized models are up to 2–4x faster on CPU and 4x smaller. We expect further speed-ups with hardware accelerators, such as Edge TPUs.





Accuracy

With just 100 calibration images from ImageNet dataset, fully quantized integer models have comparable accuracy with their float versions (MobileNet v1 loses 1%).

Model	Float baseline	Quantization during training	Quantization after training
MobileNet v1 (1.0, 224)	70.95%	69.97%	69.568%
ResNet v2	76.8%	76.7%	76.652%
Inception v3	77.9%	77.5%	77.782%

How these integer models work

Recording dynamic ranges

Our new tool works by recording dynamic ranges, running multiple inferences on a floating point TensorFlow Lite model, using the user-provided representative dataset as input. We use the values logged from inferences to determine the scaling parameters needed to execute all tensors of the model in integer arithmetic.

Int8 quantization scheme

It is important to note that our new quantization specification enabled this post-training use case that uses per-axis quantization for certain operations. Prior to our addition of per-axis quantization, post-training integer quantization was impractical due to accuracy

drops; but the accuracy benefits of per-axis bring the accuracy much closer to float for many models.

8-bit quantization approximates floating point values using the following formula:

```
real_value = (sint8_value — zero_point) * scale.
```

Per-axis (also known as “per-channel”) or per-layer weights represented by int8 two’s complement values in the range [-127, 127] with zero-point equal to 0.

Per-layer activations/inputs represented by int8 two’s complement values in the range [-128, 127], with a zero-point in range [-128, 127].

For more details, see the full quantization specification.

What about quantization aware training?

We believe in making quantization as simple as possible. Hence, enabling a way to quantize models after training is something that we are very excited about! However, we also know that some models preserve the best quality when they are trained with quantization. That’s why we are also working on a quantization aware training API. In the meantime, we encourage you to try post-training quantization, since it may be all your model needs!

Documentation and tutorial

On the TensorFlow website you can find out more about post-training integer quantization, our new quantization spec, and a post-training integer quantization tutorial. We’d love to hear how you use this — share your story!

Acknowledgements

Suharsh Sivakumar, Jian Li, Shashi Shekhar, Yunlu Li, Alan Chiao, Raziel Alvarez, Lawrence Chan, Daniel Situnayake, Tim Davis, Sarah Sirajuddin