Introducing the Model Optimization Toolkit for TensorFlow

TensorFlow · Follow
Published in TensorFlow · 4 min read · Sep 19, 2018

We are excited to introduce a new optimization toolkit in TensorFlow: a suite of techniques that developers, both novice and advanced, can use to optimize machine learning models for deployment and execution.

While we expect that these techniques will be useful for optimizing any TensorFlow model for deployment, they are particularly important for TensorFlow Lite developers who are serving models on devices with tight memory, power constraints, and storage limitations. If you haven't tried out TensorFlow Lite yet, you can find out more about it here.

Optimize models to reduce size, latency and power for negligible loss in accuracy

The first technique that we are adding support for is post-training quantization to the TensorFlow Lite conversion tool. This can result in up to 4x compression and up to 3x faster execution for relevant machine learning models.

By quantizing their models, developers will also gain the additional benefit of reduced power consumption. This can be useful for deployment in edge devices, beyond mobile phones.

Enabling post-training quantization

The post-training quantization technique is integrated into the TensorFlow Lite conversion tool. Getting started is easy: after building their TensorFlow model, developers can simply enable the 'post_training_quantize' flag in the TensorFlow Lite conversion tool. Assuming that the saved model is stored in saved_model_dir, the quantized tflite flatbuffer can be generated:

```
converter=tf.contrib.lite.TocoConverter.from_saved_model(saved_model_dir)
converter.post_training_quantize=True
tflite_quantized_model=converter.convert()
open("quantized_model.tflite", "wb").write(tflite_quantized_model)
```

tf_lite_command_line hosted with ❤ by GitHub view raw

Our tutorial walks you through how to do this in depth. In the future, we aim to incorporate this technique into general TensorFlow tooling as well, so that it can be used for deployment on platforms not currently supported by TensorFlow Lite.

Benefits of post-training quantization

- 4x reduction in model sizes
- Models, which consist primarily of convolutional layers, get 10–50% faster execution
- RNN-based models get up to 3x speed-up
- Due to reduced memory and computation requirements, we expect that most models will also have lower power consumption

See graphs below for model size reduction and execution time speed-ups for a few models (measurements done on Android Pixel 2 phone using a single core).

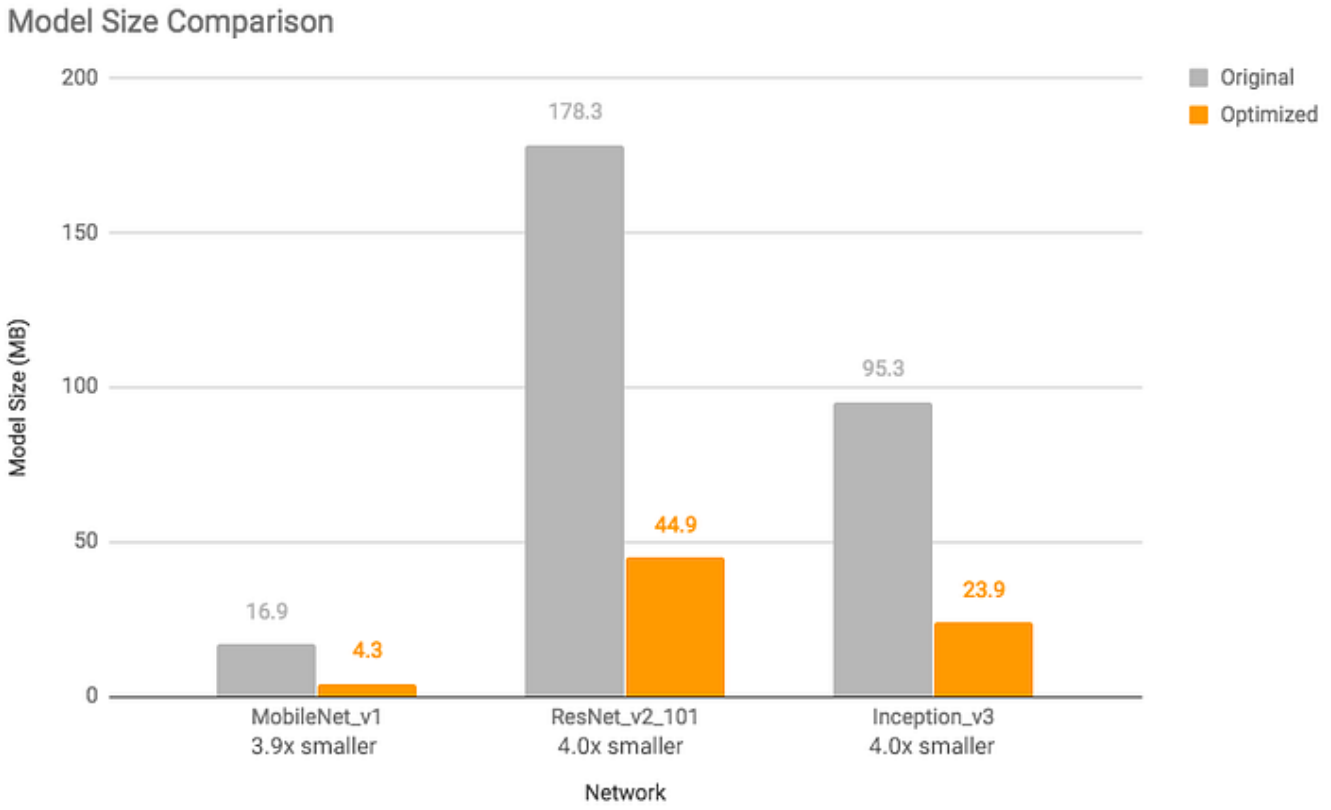


Figure 1: Model Size Comparison: Optimized models are almost 4x smaller

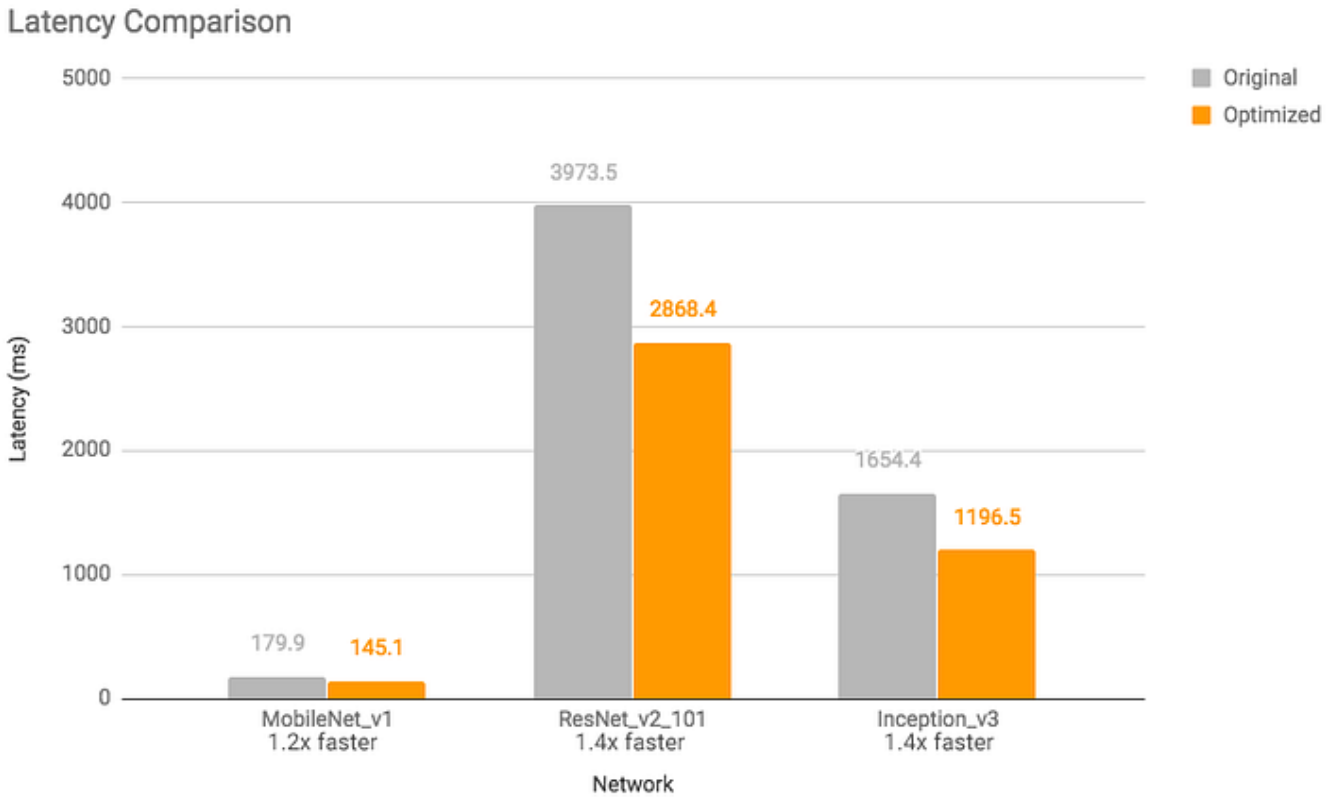


Figure 2: Latency Comparison: Optimized models are 1.2 to 1.4x faster

These speed-ups and model size reductions occur with little impact to accuracy. In general, models that are already small for the task at hand (for example, mobilenet v1 for image classification) may experience more accuracy loss. For many of these models we provide pre-trained fully-quantized models.

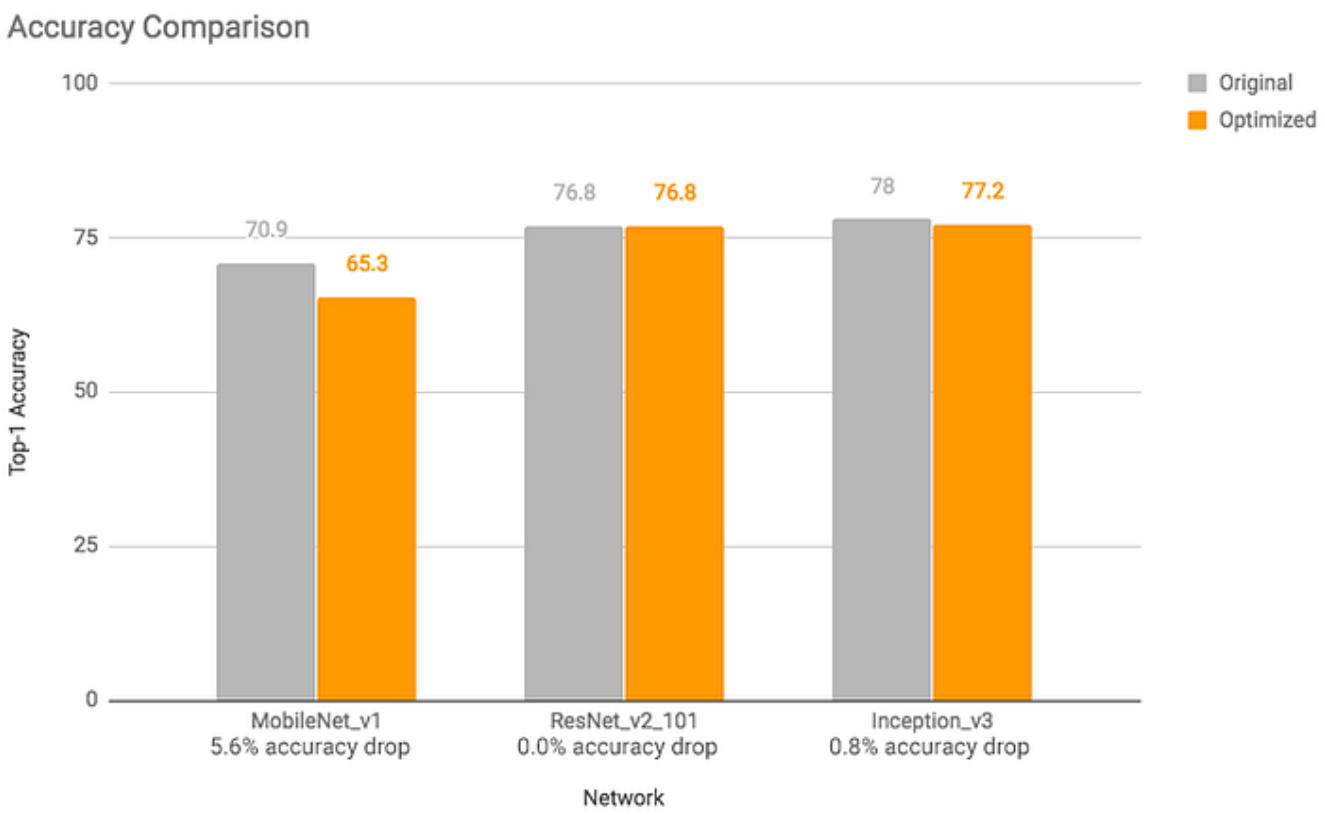


Figure 3: Accuracy Comparison: Optimized models show negligible accuracy drop, except for mobilenets

We expect to continue improving our results in the future, so please see the model optimization guide for the latest measurements.

How post-training quantization works

Under the hood, we are running optimizations (otherwise referred to as quantization) by lowering the precision of the parameters (i.e. neural network weights) from their training-time 32-bit floating-point representations into much smaller and efficient 8-bit integer ones. See the post-training quantization guide for more details.

These optimizations will make sure to pair the reduced-precision operation definitions in the resulting model with kernel implementations that use a mix of fixed- and floating-point math. This will execute the heaviest computations fast in lower precision, but the most sensitive ones with higher precision, thus typically resulting in little to no final accuracy losses for the task, yet a significant speed-up over pure floating-point execution. For operations where there isn't a matching "hybrid" kernel, or where the Toolkit deems it necessary, it will reconvert the parameters to the higher floating point precision for execution. Please see the post-training quantization page for a list of supported hybrid operations.

Future work

We will continue to improve post-training quantization as well as work on other techniques which make it easier to optimize models. These will be integrated into relevant TensorFlow workflows to make them easy to use.

Post-training quantization is the first offering under the umbrella of the optimization toolkit that we are developing. We look forward to getting developer feedback on it.

Please file issues at GitHub and ask questions at Stack Overflow.

Acknowledgements

We would like to acknowledge core contributions from Raghu Krishnamoorthi, Raziel Alvrarez, Suharsh Sivakumar, Yunlu Li, Alan Chiao, Pete Warden, Shashi Shekhar, Sarah Sirajuddin and Tim Davis. Mark Daoust helped create the colab tutorial. Billy Lamberta and Lawrence Chan helped with the creation of the website.

Machine Learning | Model Optimization | TensorFlow | Mobile

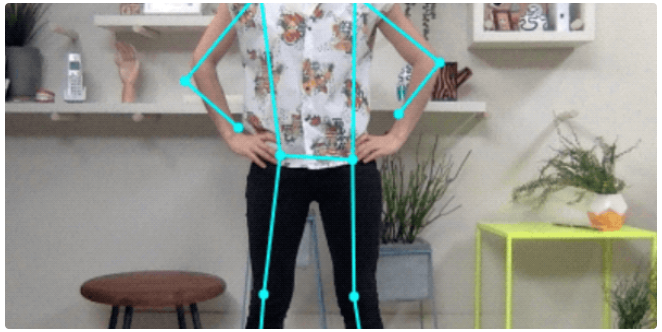
Written by TensorFlow

30K Followers · Editor for TensorFlow

Follow

TensorFlow is a fast, flexible, and scalable open-source machine learning library for research and production.

More from TensorFlow and TensorFlow



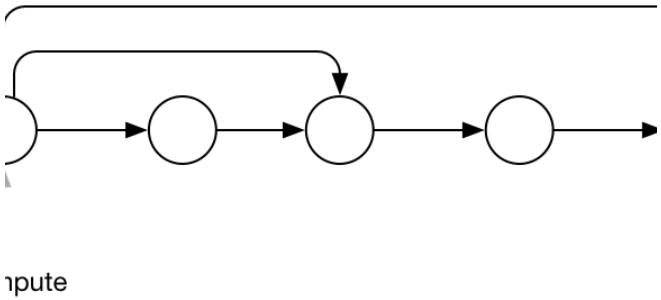
TensorFlow in TensorFlow

Real-time Human Pose Estimation in the Browser with TensorFlow.js

Posted by: Dan Oved, freelance creative technologist at Google Creative Lab, gradua...

13 min read · May 7, 2018

7.9K 35



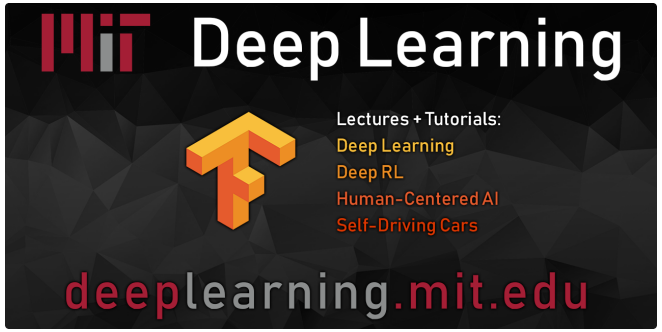
Yaroslav Bulatov in TensorFlow

Fitting larger networks into memory.

TLDR; we (OpenAI) release the python/Tensorflow package openai/gradient...

12 min read · Jan 14, 2018

3.4K 10



Lex Fridman in TensorFlow

TensorFlow in TensorFlow

MIT Deep Learning Basics: Introduction and Overview with...

As part of the MIT Deep Learning series of lectures and GitHub tutorials, we are coverin...

8 min read · Feb 4, 2019

5.9K 18

Colab: An easy way to learn and use TensorFlow

Colaboratory is a hosted Jupyter notebook environment that is free to use and requires...

2 min read · May 3, 2018

2.4K 13

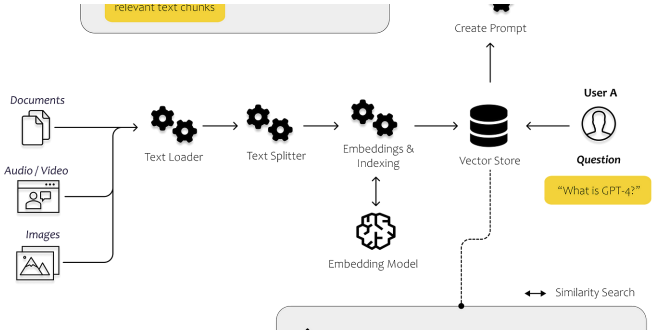
See all from TensorFlow

See all from TensorFlow

Recommended from Medium

Jari Roomer in Better Humans

How I Eliminated Procrastination From My Life (Using Neuroscience)



Dominik Polzer in Towards Data Science

All You Need to Know to Build Your First LLM App

Keep this part of the brain in optimal condition if you want to stop procrastinating.

6 min read · Jun 22

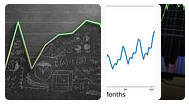
10.7K 120

A step-by-step tutorial to document loaders, embeddings, vector stores and prompt...

26 min read · Jun 22

2.5K 22

Lists

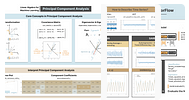


Predictive Modeling w/ Python

18 stories · 135 saves

Natural Language Processing

411 stories · 61 saves

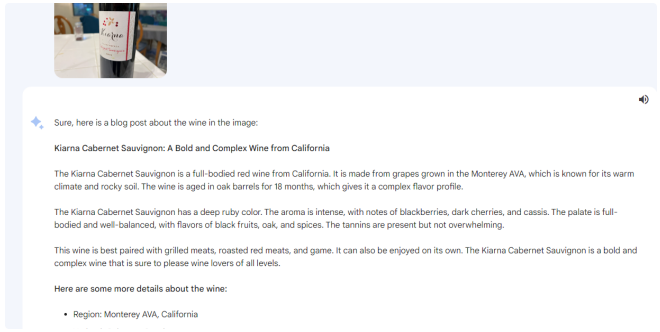


Practical Guides to Machine Learning

10 stories · 148 saves

The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 54 saves



Thomas Smith in The Generator

Google Bard's New Visual Feature is a Game Changer

Afaque Um... in Artificial Intelligence in Plain Engli...

🎬 TMDB 🤝 Streamlit 🔥: Build Your Own Movie Recommendatio...

Chatbots can officially see the world

✦ · 6 min read · Jul 14

5.7K 118

Understanding Key Concepts and the Science Behind Recommendation Engines while...

20 min read · Jul 7

290 1

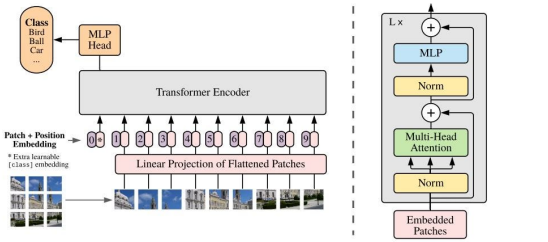
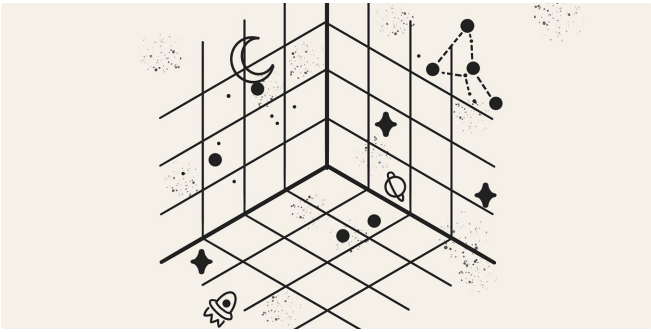


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by



Fahim Rustamy, PhD

Vision Transformers vs. Convolutional Neural Networks

This blog post is inspired by the paper titled AN IMAGE IS WORTH 16×16 WORDS:...

7 min read · Jun 5

154 1

Leonie Monigatti in Towards Data Science

Explaining Vector Databases in 3 Levels of Difficulty

From noob to expert: Demystifying vector databases across different backgrounds

✦ · 8 min read · Jul 4

1.4K 17

See more recommendations