TENSORFLOW VS PYTORCH: CHOOSING YOUR ML FRAMEWORK



Among the most used machine learning (ML) frameworks, two are quite popular:

- PyTorch is Facebook's ML package
- TensorFlow is from Google

Both allow you to build Machine Learning models, both have easy out-of-the-box models, and both are highly customizable. Whether you are new to the field of an expert, these libraries can satisfy all your needs—from testing to deployment. Let's take a look at the differences between them.

Note: For those looking to choose a programming language, both libraries are written in Python. I machine learning is the direction you intend to go, learning Python is a common denominator. TensorFlow now has TensorFlow JS, so it can be used with JavaScript as well.

(This tutorial is part of our Guide to Machine Learning with TensorFlow & Keras. Use the right-hand menu to navigate.)

Getting started

TensorFlow is older and more widespread. Initially, it had a lot more community support and tutorials out on the web, but PyTorch has caught up. There is now extensive documentation for both, so learning one will not be any easier than the other. The key is to just get started.

With modelling, too, you rarely need to start from scratch. You can both find pre-built models, and models with pre-trained weights:

- TensorFlow has a model garden or Keras

- PyTorch has its PyTorch Hub
- Huggingface.co has a good repo of both

Huggingface, too, is a great ML community that uses both PyTorch and TensorFlow models interchangeably (they literally have a tool to convert models and weights between the two), and their new Hugging Face course can walk you through how to get started with Machine Learning.

Additional resources include:

- PyTorch tutorials
- TensorFlow tutorials
- Repository of TensorFlow Demos on Colab (Bonus: Use Google's Colab Notebooks for a free GPU to train your models)

Code style

The code for each looks different. PyTorch is more pythonic and uses the Object-Oriented Programming styles. TensorFlow has made things easy for coders, but, in doing so, it has removed some of the normal kind of coding developers are used to.

Tensors

Both frameworks use:

- Tensors to pass data through to its models
- Graphs to define their models

TensorFlow has a statically defined graph that gets exposed to the user through the commands tf.session and tf.Placeholder. PyTorch has dynamically defined graphs that become useful for some more complex models. TensorFlow Fold enables dynamism.

Comparing TensorFlow vs PyTorch

Now, let's turn to a comparison of both frameworks.

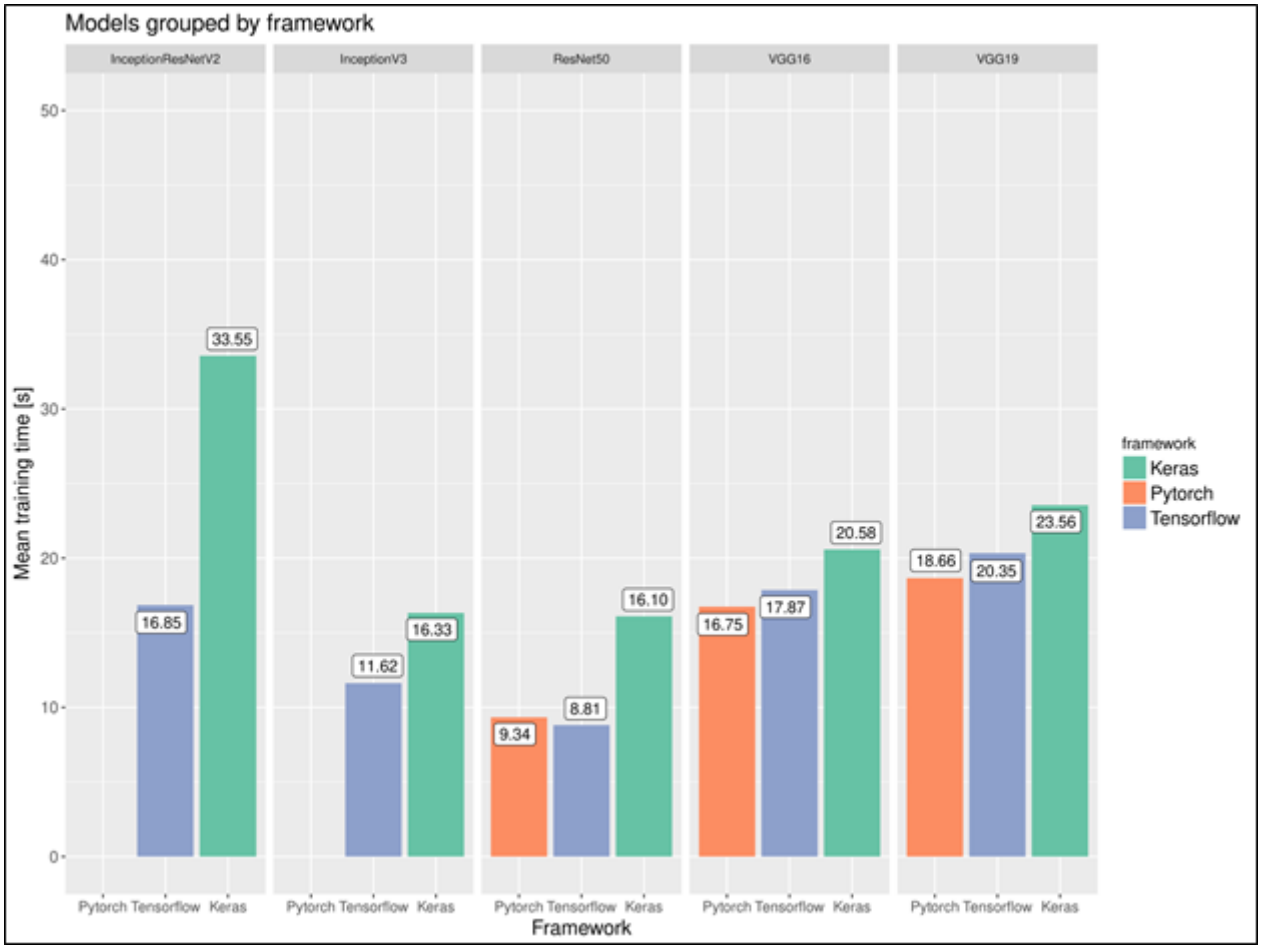
Performance tests

At the beginning of 2020, OpenAI standardized on PyTorch. They found it easier and quicker to test new ideas.

PyTorch reports slower training times, which, given some models these companies use, training can take thousands of hours of compute so the training time is directly associated with costs. PyTorch tends to take more memory during training.

Accuracy

Both model frameworks can get the same accuracy. (A Comparison of Two Popular Machine Learning Frameworks)



This comparison is a little dated (2017), but, from a report, its results appear to still hold.

Distributed training

Distributed training is getting closer to a must for large models. The big idea is to train a model on a Kubernetes cluster. On PyTorch, implementing a distributed training model is easy:

```
import torch.distributed as dist
from torch.nn.parallel import DistributedDataParallel
dist.init_process_group("nccl", world_size=2)
model = DistributedDataParallel(
        model)
```

On TensorFlow, you'll have to do a lot of the coding yourself.

Deploying models

Models are great for research and backend purposes, but they need to work out in the world, too. This means the model and its weights need to get either:

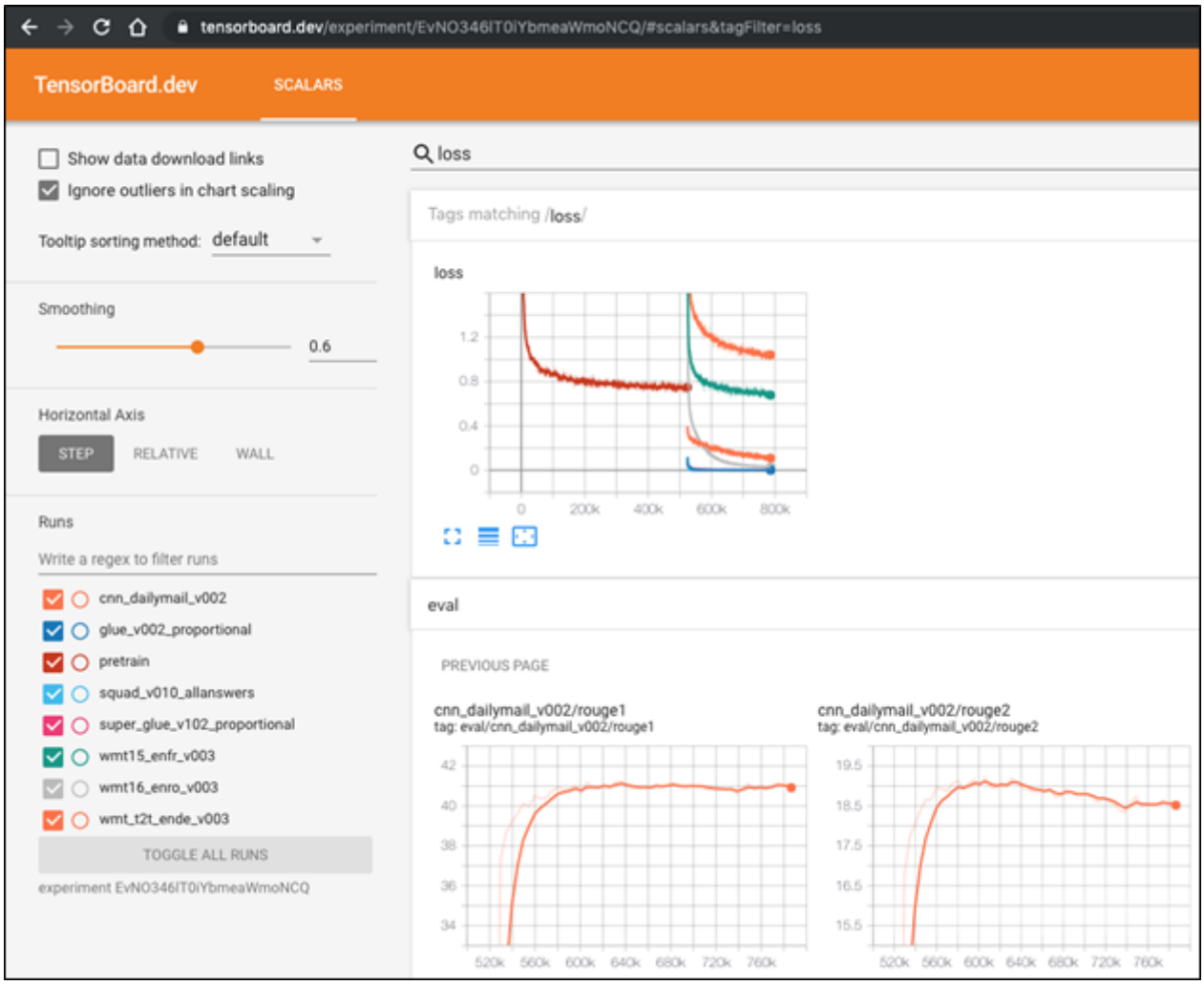
- Loaded onto a mobile or edge device
- Turned into an API so it can be used for inference

At first, TensorFlow was the best framework for this, but as time has passed, both frameworks can be used to deploy models into production and there is plenty of documentation on how to do so.

Now, given Google has its own cloud framework, if you are using Google Cloud, TensorFlow can be integrated with Google's services pretty easily, such as saving a TF Lite model onto its Firestore account and delivering the model to a mobile application. Though some of the TF stuff integrates well with Google's services, there are plenty of workarounds to get PyTorch models working, too.

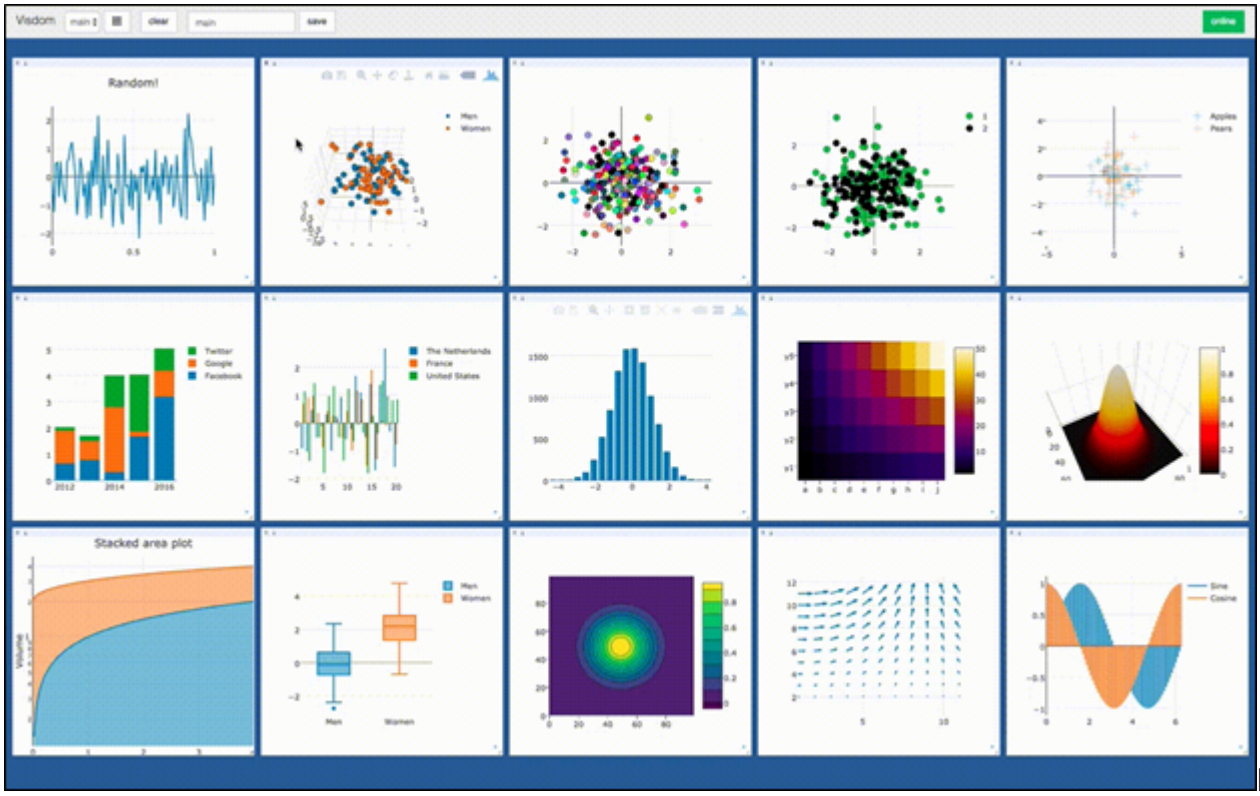
Data visualization

TensorFlow has TensorBoard which lets you see multiple aspects around the machine learning model from the model graph to the loss curve. It has lots of options to explore your model, as you can see:



(Source)

PyTorch has no native visualization tool. Instead, there exists a graph visualization package called Visdom.



If needed, there is a way to display your PyTorch models on Tensorboard.

That concludes this comparison of TensorFlow and PyTorch. Browse the Guide to learn more.

Related reading

- BMC Machine Learning & Big Data Blog
- Anomaly Detection with Machine Learning: An Introduction
- Bias & Variance in Machine Learning: Concepts & Tutorials
- Interpretability vs Explainability: The Black Box of Machine Learning
- React JavaScript Library: Concepts & Tutorials for Getting Started
- The Data Visualization Beginner's Guide