

Pytorch vs. Tensorflow

What is Pytorch?

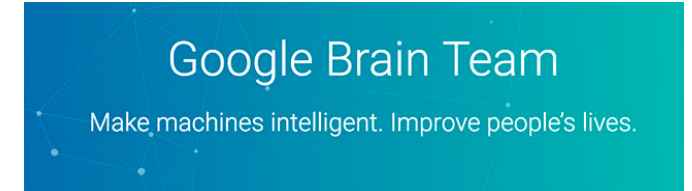


- **Pytorch**

- python based deep learning library for researching and developing Deep learning models
- origin from lua-based deep learning library, Torch.
- Pytorch is written by native python language. (Not a simple set of wrapper to support Python)
- Actively used at Facebook
- essentially a GPU enabled version for NumPy with higher-level functionality for building and training deep neural networks.

What is Tensorflow?

- **Tensorflow**



- a Deep learning library for researching, developing, and distributing deep learning models.
- developed by Google Brain and actively used at Google.
- a programming language embedded within Python
(Tensorflow codes are complied into a graph by Python and then run by the Tensorflow execution engine)

Difference - Adoption

- **Tensorflow**

- well documented
- many users
- many tutorials are available
- hundreds of implemented and trained models on github

- **Pytorch**

- quickly getting its momentum
- still in beta version (currently v. 0.2.0)
- nice documentation and official tutorials
- several computer vision architectures available

Difference – Graph definition

- Both frameworks operate on tensors and view models as a directed acyclic graph(DAG).
 - They are different in a way to define DAGs.
- | | |
|---|--|
| <ul style="list-style-type: none">• Tensorflow<ul style="list-style-type: none">• Use Static graph
(Graph is defined before a model can run.)• All communication is performed via tf.session and tf.Placeholder• support limited dynamic inputs | <ul style="list-style-type: none">• Pytorch<ul style="list-style-type: none">• Use Dynamic graph
(Graphs can be defined, changed, and executed as model runs)• Being tightly integrated with Python language, give more native and free way to work with models.• Dynamic neural net like RNNs can benefit from this dynamic approach |
|---|--|

Difference – Data loading

- **Tensorflow**

- relatively not intuitive for data loading.
- adding preprocessing code in parallel into Tensorflow graph is not straight-forward.

- **Pytorch**

- APIs for data loading are well designed.
- a data loader takes a dataset and produces an iterator over the dataset.
- Parallelizing data loading is simple.

Difference – Debugging

- **Tensorflow**

- Need to use a special debugging tool, **tfdbg**
- **tfdbg** allows to evaluate tensorflow expressions at runtime and browse all tensors and operations in session scope.

- **Pytorch**

- Graph is defined at runtime
- You can use your favorite Python debugging tools such as **pdb, ipdb, Pycharm debuggers.**

Difference – Visualization

- **Tensorflow**

- has its own visualization tool **Tensorboard**.

- Tensorboard can
 - display model graph
 - plot scalar variables
 - visualize distributions and histograms
 - visualize images
 - visualize embeddings
 - play audio

- **Pytorch**

- Currently, no equivalent for Tensorboard.
 - Integrations to tensorboard exist
 - standard plotting tools (matplotlib, seaborn) can be used

Difference – Serialization

- Both frameworks provides simple way to saving and loading models

- **Tensorflow**

- **Tf.Saver** object provides easy way to save models and check-points.
- Entire graph can be saved as a protocol buffer.
- Graph can be loaded in other supported languages (C++, JAVA)

- **Pytorch**

- provides simple API that can save all the weights of a model.

Difference – Custom extensions

- Building or binding custom extensions written in C, C++ or CUDA is doable in both frameworks

- **Tensorflow**

- requires more boilerplate code for custom extensions.

- **Pytorch**

- You can make extensions by simply write an interface and correponding implementation for each of the CPU and GPU versions.

Difference – Deployment

- **Tensorflow**

- **Tensorflow Serving** provides a easy way to deploy models
- models can be deployed into embeded system or mobile platforms.
- Provide high performance server-side deployments
- support **Distributed training**

- **Pytorch**

- For a small-scale server-side deployments are easy to wrap using Flask web server.
- support **Distributed training**

Summary

- Both frameworks provides useful abstractions to reduce repeated codes and speed up the model development.
- **PyTorch**
 - provides flexible dynamic graph definition
 - more 'pythonic' way for developing and debugging
 - an object-oriented approach.
 - PyTorch is better for rapid prototyping in research, and small scale projects.
- **Tensorflow**
 - provide great visualization and deployment tools.
 - A good choice if you develop a model for production and deploying on mobile platforms
 - Tensorflow is better for large-scale deployments especially when cross-platform and embedded deployment is a consideration