CP8321-A6: A Comparison of Deep Learning Libraries for Convolutional Neural Networks

Christopher Indris
Department of Computer Science
Toronto Metropolitan University
Toronto, Canada
cindris@cs.ryerson.ca

***Abstract*—This paper compares three of the leading Python deep learning libraries—PyTorch, TensorFlow, Keras—in the context of convolutional neural networks (CNNs). Areas of assessment include each library's requirements, their strengths and weaknesses, and their user experience during the creation of a CNN to classify MNIST digits.**

***Index Terms*—PyTorch, TensorFlow, Keras, deep learning, CNN, API, ML**

I. Introduction

Machine learning has been a leading source of excitement among the various subfields of technology during the past decade. Researchers and hobbyists regard it as tremendously promising for many disciplines. Prior to the rise of frameworks and APIs that abstract away most of the technical details, the difficulty of building, training and deploying a machine learning model prohibited those without considerable computer science or engineering experience from involvement in the machine learning pipeline. There are now several libraries that can make working with neural networks far more concise than they otherwise would be, though it may not be initially clear which one is the best choice for a particular application or skill set.

II. Overview

The key neural network architecture data structures that users of these libraries encounter are layers (for creating the model) and the models themselves. They may provide finer control over the architecture; there is the Keras Functional API for creating models that are more complex than a sequence of layers, and researchers can even subclass existing classes to build entirely new layers and models.

PyTorch is a framework that was developed by Meta AI and pushed to open-source in 2017. The principal functionalities provided by PyTorch are GPU-accelerated tensor computation (`torch`) and deep neural networks (`torch.nn`) with autograd (`torch.autograd`).

TensorFlow is a machine learning platform and community hub to simplify research and the creation of applications. Google Brain launched TensorFlow 1.0 in 2015 with powerful and flexible low-level functionality. The low-level nature of TensorFlow 1.0 meant that it was difficult and tedious to work with, so in 2019 Google released TensorFlow 2.0. It now came paired with the high-level API of Keras so that writing a script using it became more straightforward, and did away with the need for a session environment. TensorFlow now supports ragged tensors (arrays with non-uniform shapes) and maintains its mission to make ML accessible by providing Trax, an end-to-end library focused on speed and clear code.

Keras is a high-level neural network API front-end with the aim of being accessible and modular. Created in 2015, it was adopted and integrated into TensorFlow in 2017 and would later become part of TensorFlow 2.0. Currently, TensorFlow is the only backend that Keras supports, and Keras will generally be used as part of TensorFlow, although Keras can still be used without TensorFlow (if the user is satisfied with the limited functionality that would result from not having the TensorFlow backend).

III. Hardware and Software Requirements

PyTorch can be run on the macOS Catalina and Windows 7 operating systems and their successors, as well as Linux-based systems with GNU C Library (glibc) version 2.17 or later (for Ubuntu, this means 13.04 at the earliest). At an early phase of release is the end-to-end workflow for mobile deployment with PyTorch 1.3, so that PyTorch could be used on iOS or Android devices. PyTorch can be used for the C++, Java and Python languages; for Windows, Python 3.7-3.9 are supported and Anaconda environments are highly recommended due to their sandboxed nature (by default, Windows does not come with Python), though pip is a common package manager alternative. PyTorch can also be installed from source code, with a BLAS library (MKL) for optimized mathematic routines in an Anaconda environment. The NVIDIA CUDA toolkit and cuDNN library will be critical for CUDA GPU support for the deep learning code. Binary files on macOS do not support CUDA, so to have CUDA the user should install from source. PyTorch can run on ARM-based Macs.

TensorFlow can be used by machines operating with macOS Sierra, Windows 7 and Ubuntu 16.04 and their successors. It is available for Python 3.7 through to 3.10. For GPU support (NVIDIA CUDA architecture), the NVIDIA CUDA toolkit, cuDNN SDK and appropriate drivers are needed, with TensorRT as a useful optional extra for better latency and throughput for ML inference. On macOS, GPUs are not supported by default, but a Miniconda environment is a reccomended workaround. On Windows, the last version supporting a GPU is TensorFlow 2.10; for later versions, the user is advised to use WSL2 (Windows Subsystem for Linux). For macOS devices with an M1 chip, the `tensorflow-metal` plug-in is used, though this version cannot handle working with multiple GPUs. Most users will use Keras with TensorFlow (`tensorflow.keras`), and so the hardware and software requirements follow from TensorFlow.

IV. Installation Process and Related Requirements

To install PyTorch, simplest method is typically to run the `pip install torch` command. PyTorch can also be built from source code, or run with a pre-built Docker image. The image provides virtualization in the form of a container. Arguably the easiest method of all is to use Google Colaboratory, which runs in a browser and comes pre-installed with PyTorch. The user can then run `import torch` in their Python script or console. PyTorch lists NumPy and Cmake as dependencies; the former since tensors and NumPy ndarrays are closely related, the latter since PyTorch was written mainly in C++.

TensorFlow is also pre-installed on Google Colaboratory, can be installed from source code on Linux and macOS, and can be installed on the command line using `pip install tensorflow` for the current stable CPU/GPU version or `tf-nightly` for the potentially unstable preview build. In my Anaconda environment, some key dependencies are recognizable: NumPy, for arrays; TensorBoard for visualization; Keras, the API; Bazel, for building (needed when TensorFlow is being installed from source code).

Keras depends on Python's suite of scientific tools (NumPy and SciPy), BLAS for the CPU, and the CUDA drivers and cuDNN so that the GPU can manage DNN code.

V. Strengths and Shortcomings

The fast tape system for autograd is a selling point for PyTorch. When a neural network is built in PyTorch, a dynamic graph is built. When the graph (the network behaviour) is changed, PyTorch can quickly adapt by using the tape recording and reverse-mode autodiff so that there is no need to start the graph from scratch. It is open-source with a large community. For shortcomings, the included visualization features are not as sophisticated as those in TensorFlow and an API server is needed when putting a program into production.

TensorFlow visualizes training well with TensorBoard and is production-ready. Again, there is a large open-source community. Unfortunately, the graphs are static and the debugging methods are not as good; together, these make it harder to implement quick fixes to the users' programs.

Keras supports mutliple GPUs. The combination of modularity, available pre-trained models and fast deployment make it user-friendly. However, it is slow compared to the backend and working with the low-level TensorFlow functionality may lead to errors that Keras does not catch. Without the TensorFlow backend, freedom is quite limited.

VI. Software of Choice

In general, TensorFlow (with Keras) is easier to use, while PyTorch is more complex but provides the programmer with a considerable amount of freedom and good debugging capabilities. For Assignment 5, my group and I put together a visualizer for the backpropagation of a CNN being trained on the MNIST dataset. We chose PyTorch for our implementation because the functionality we wanted for our program—the GUI updates in (just about) real-time as the network is training in a worker thread—seemed to be easier to do in PyTorch. In PyTorch, the backward() function can be called explicitly, giving us a good indication of where the backpropagation was happening in the code. In TensorFlow, the updating weights could be recorded using callback functionality (as the model is being fit, the process can be written to a "history" variable). This may have required custom callback code to save the weights, and it did not seem like the ideal interface for writing to the GUI in real time.

References

[1] D. F. Team, "Python keras advantages and limitations," DataFlair, 25-Aug-2021. [Online]. Available: https://data-flair.training/blogs/python-keras-advantages-and-limitations/. [Accessed: 15-Nov-2022].

[2] J. Terra, "Pytorch vs tensorflow vs keras: Here are the difference you should know," Simplilearn.com, 22-Sep-2022. [Online]. Available: https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article. [Accessed: 15-Nov-2022].

[3] Keras-Team, "Keras-Team/Keras: Deep Learning for Humans," GitHub, 2015. [Online]. Available: https://github.com/keras-team/keras. [Accessed: 15-Nov-2022].

[4] PyTorch, "Pytorch," PyTorch. [Online]. Available: https://pytorch.org/get-started/locally/. [Accessed: 15-Nov-2022].

[5] Pytorch, "Pytorch/Pytorch: Tensors and dynamic neural networks in python with strong GPU acceleration," GitHub, 2017. [Online]. Available: https://github.com/pytorch/pytorch. [Accessed: 15-Nov-2022].

[6] S. Gupta, "Tensorflow 1.0 vs. tensorflow 2.0: What's the difference?," Springboard Blog, 20-Jul-2022. [Online]. Available: https://www.springboard.com/blog/data-science/tensorflow-1-0-vs-tensorflow-2-0/. [Accessed: 15-Nov-2022].

[7] TensorFlow, "Install tensorflow 2," TensorFlow. [Online]. Available: https://www.tensorflow.org/install. [Accessed: 15-Nov-2022].

[8] TensorFlow, "Ragged tensors" TensorFlow. [Online]. Available: https://www.tensorflow.org/guide/ragged_tensor. [Accessed: 15-Nov-2022].

[9] Tensorflow, "Tensorflow/tensorflow: An open source machine learning framework for everyone," GitHub, 2015. [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed: 15-Nov-2022].

[10] TensorFlow, "Trax Quick Intro," Google Colab. [Online]. Available: https://colab.research.google.com/ github/google/trax/blob/master/-trax/intro.ipynb. [Accessed: 15-Nov-2022].

[11] V. Kurama, "Pytorch vs. tensorflow: Which framework is best for your deep learning project?," Built In, 2022. [Online]. Available: https://builtin.com/data-science/pytorch-vs-tensorflow. [Accessed: 15-Nov-2022].

TABLE I
SIMILARITIES AND DIFFERENCES

Library	Open Source	Platform	CUDA Support	Autodiff	Speed	API level	Debugging
PyTorch	Yes	Linux/macOS/Windows	Yes	Yes	Fast	Low	Easy
TensorFlow	Yes	Linux/macOS/Windows	Yes	Yes	Fast	Both	Hard
Keras	Yes	Linux/macOS/Windows	Yes	Yes	Slow	High	Easy