CME 216, ME 343 - Spring 2020

Eric Darve, ICME

Stanford University

What is TensorFlow?

TensorFlow

TF can do many things.

It's free and open source.

Obviously it can be used for deep learning.

But at its base, it allows to express computations as a graph.

Then this graph can be differentiated, e.g., we can calculate gradients and solve optimization problems using gradient descent methods.

For now, we will skip these general features of TF and focus on DNN but we will come back to that when discussing physics-informed machine learning or PhysML.

TensorFlow was developed by the Google Brain team for internal use at Google.

Then it was made open source in November 2015.

What is Keras then?

K Keras

Keras is an API (application programming interface) for DNN calculations.

API means that Keras defines a specific interface to write computer programs.

Keras is written in Python.

Keras has multiple backends (that is libraries that are responsible for doing the actual calculations).

At this time they include:

- TensorFlow
- CNTK (from Microsoft)
- Theano (University of Montréal, Canada)

Compared to TF, Keras is focused on easy and fast prototyping, through

- user friendliness,
- modularity, and
- extensibility

Although TF can be used as a backend for Keras, it is recommended to use `tf.keras`, which is the implementation of Keras in TF.

This is what we will do in this class.

What about TF 1 and TF 2?

TensorFlow is constantly being modified and improved. But there was a major change when TF transitioned from TF1 to TF2.

TF1 was originally developed and made available around 2015.

But TF1 was somewhat hard to use and not intuitive for many users.

Shortly after, libraries like Keras were developed to make the task of implementing DNNs easier.

TF2 was released in 2019 and is closely integrated with Keras.

The goal of TF2 was to:

- be higher-level than TF1 (e.g., less confusing code and simpler ways of writing common tasks and structures)
- simplified API
- greater versatility

One of the big changes from TF1 to TF2 was that TF2 allows the so-called "eager" execution.

This is a technical point so we will not elaborate much.

Briefly, in TF1, the user would have to first declare the structure of the DNN.

Then the DNN would be "compiled" and executed.

In eager mode, we simply declare the sequence of operations to perform and the operations are evaluated immediately.

For us, this means that it's easier to get started and it eliminates a lot of boiler plate material.

What about PyTorch?

We will not cover PyTorch in this class but this is also an excellent library.

It's very easy to use.

Contrary to TF1, PyTorch had "eager" execution from the beginning.

For now, it's unclear what the best tool is...

It's still a heated discussion.