# Lecture note 4: Eager execution and interface

CS 20: TensorFlow for Deep Learning Research

cs20.stanford.edu

Prepared by Chip Huyen and Akshay Agrawal

Contact: chiphuyen@cs.stanford.edu, akshayka@{cs.stanford.edu, google.com}

Up until this point, we’ve implemented two simple models in TensorFlow: linear regression to predict life expectancy from birth rate , and logistic regression to do an Optical Character Recognition task on the MNIST dataset. We’ve learned that a TensorFlow program often has two phases: assembling the computation graph and executing that graph. But what if you could execute TensorFlow operations imperatively, directly from Python? This can make debugging our TensorFlow models a lot less intimidating.

In this lecture, we introduce eager execution, rewriting our linear regression model with eager.

## Eager execution

Eager execution is (1) a NumPy-like library for numerical computation with support for GPU acceleration and automatic differentiation, and (2) a flexible platform for machine learning research and experimentation. It's available as tf.contrib.eager, starting with version 1.50 of TensorFlow.

* Motivation:
  + TensorFlow today: Construct a graph and execute it.
    - This is *declarative* programming. Its benefits include performance and easy translation to other platforms; drawbacks include that declarative programming is non-Pythonic and difficult to debug.
  + What if you could execute operations directly?
    - Eager execution offers just that: it is an *imperative* front-end to TensorFlow.
* Key advantages: Eager execution …
  + is compatible with Python debugging tools
    - pdb.set\_trace() to your heart's content!
  + provides immediate error reporting
  + permits use of Python data structures
    - e.g., for structured input
  + enables you to use and differentiate through Python control flow
* Enabling eager execution requires two lines of code

import tensorflow as tf

import tensorflow.contrib.eager as tfe

tfe.enable\_eager\_execution() # Call this at program start-up

and lets you write code that you can easily execute in a REPL, like this

x = [[2.]] # No need for placeholders!

m = tf.matmul(x, x)

print(m) # No sessions!

# tf.Tensor([[4.]], shape=(1, 1), dtype=float32)

For more details, check out [lecture slides 04](https://docs.google.com/presentation/d/1e1gE2JJXipWm1UJgor_y8pHcM8L8oMaCVtvQvZUBlQY/edit?usp=sharing).