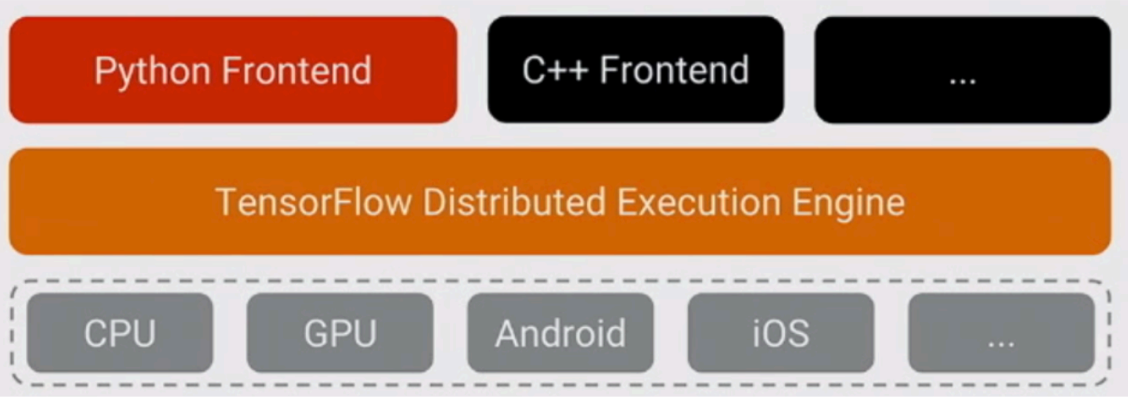
Language bindings for TensorFlow

CS 131 UCLA, Winter 2018

TensorFlow was developed by the engineers of Google working in research organization of Machine intelligence and neural networks. This is an open source library that dealt with numerical computations using data flow graphs. Everything seems to represent numerically in the world of big data. Machine learning is heavy with the mathematical operations and all the languages currently available out there may not be suitable because of the issues like bottleneck, effective, flexibility, generality, performance, and reliability etc. that can be experienced when choose wrong language. Currently although Python is duck typing, it is widely accepted language in machine learning because of its varieties of highly sophisticated libraries and functionality of built-in functions that it offers. However, in TensorFlow, using python slows down the process and there might be a good programming languages that can be overcome this issue.



1. **Introduction**
   When it comes to realization about TensorFlow, the language under the hood that are vastly being used to write the core part is not in python. Rather it is written by using the combination of highly-optimized C++, and CUDA (Nvidia's language for programming GPUs). Furthermore, the library like Eigen, a C++, and CUDA numerical library, and NVidia's cuDNN, an DNN optimized library for NVidia GPUs for functional programming such as convolutions, has been used. But programmer happens to use the models of python. However, when Python is run, model itself not actually executed. Instead dataflow graph is created which can take inputs and assigning particular operation takes place. Then results are treated as an input for other operation and process repeats. The model is executed in C++ code, and not in Python.

   However, interesting fact about why then python is being used in TensorFlow. The answer to this question is that Python is a natural fit as it is very simple and comfortable language that has been using for large range of data scientist's studies. It can easily be integrated with C++ in the backend, and is being widely used as an open source. With high performance, NumPy library makes lots easier to do pre-processing in Python. However, faster performance cannot be achieved using python and behaves as bottleneck in some application. Below are the three different languages going compare with Python.

2. **Java Vs Python**
   Java is a static typing, age old language, highly used in many different industries because of its core functionality like object oriented, and to solve the problem like memory leak, efficiency, fast, flexibility,

error handling, many libraries and more. However, Java doesn't seems often used in machine learning platform, but when it comes to using with TensorFlow it can do better work compare to Python.

Compare to Python, since java is static type checking, it can be more effective while debugging the program. In early developing process, while writing code, error can be found earlier in the compilation time, rather than run time. Furthermore, Java can be portable as well because it can be compiled to byte code and can run on JVM regardless of machine architecture. Doing so would help not to worry too much handling the lower level machine specific problems. The other very important aspect to be considered is that since java is very old program, there is massive community and thus thousands of documents can be find when need help in the writing process and large libraries for external modules will be found to be integrated into the application. For the memory allocation and deallocation, both program uses the garbage collection method. But garbage collection methods are different. Python uses reference count, while Java destroy the object immediately when it doesn't need. But for Python, garbage collection can lead to memory leaks in circular reference. For an example, suppose X reference Y, Y reference Z, and Z reference Y. The counts for reference Y and z still be 1 even after X drops. This potentially leads a memory leakage.

However, Python is duck typing i.e. strongly typed, high level programming language. It has magic methods, that makes it logic/meta programming language. It offers programs to write using less code. It compiler in native bycode.

3. OCaml Vs Python

Ocaml is a functional language and has capacity of object oriented programming style. Hence, being a functional language, it is mathematical in nature that might help to solve operational problem during data analysis, and has higher performance profile, thus probably will be a good fit to use with TensorFlow

Ocaml has a clear, condensed syntax but is not a dynamic programming language like Python. Like Java, it also has static typing and compiles to native code. It can be considered very efficient in solving the problem in quantitative prospective as scripting languages are seems to be more convenient as has access to variety of libraries and is easy to read and write the code for the programmers. To creating an application with TensorFlow that gives better speed than python, Ocaml will be handy because of functional nature, provides reliability, and is fast like C/C++ which is not something than can be done with Python. Using Ocaml, it's very easy to do complex heavy duty mathematical task/operations in few lines and can work well and fast in trivial parallelization and multithreading environment. So, math is closely tied with functional programming language and hence it can provide a high-level paradigm.

Having known that Ocaml benefits over Python, its performance in parallel or numerical code is not strong. Ocaml has high performance in multiprocessing but not in multithreading because of GIL. But it's performance can gets close to C/C++ when it server as the middle-ground in many aspects. In this situation, we can use Owl library, which is a numbering library for Ocaml. Owl is like a NumPy/Julia in Python, and has very high performance and flexibility using it.

Unlike Ocaml, in addition to functional and imperative language, Python also support object oriented programming. So, it is a multi-paradigm language and is supportive structure in nature.

4. Haskell Vs Python

Like OCaml, Haskell is also a pure functional imperative language like lisp, mostly used for the real project rather than experimental. Contrast to Python, It is a static type like Java or Ocaml, hence it helps to find bugs at the earlier stage (compile time) while writing programs. Python also somehow (depends in area), support functional programming style, but it is lacking built-in tail recursion, so Haskell can be use instead for the better performance

with the tensor flow. In the compile time, Haskell give more information than python. The similarities between two languages is that both use indentation as syntax. In terms of learning, for the programmer, Haskell has stepper learning curve. Comparing with python, in ML Haskell will be more reliable. In Tensorflow and machine learning, generally low level of computation involves in simple math operations. Because of the data scale is very large, these operations have to be fast as possible. We need both software and hardware optimization the can preforms task fast. So, a low- level code representation is needed to make application work faster and Haskell can do better job on this. We can achieve those tasks using functional interface with low overhead and make performance faster. The way we can achieve this goal is, first we are going to build a graph of all the computations, and once we described our graph, we can "run" our graph using "session" using bunch of functions. Doing so will help to lower the overhead. In Haskell, this process called action. Action is a description of operations. It is not a series of command, rather as the moment show up in the code, we execute them.
Our computation graph is nothing but node of each tensor. we can take tensor as an input, and make nodes for "operation". For example, we can make a node by adding two tensors together. Tensor Flow has a really great feature, one of those is that Tensor Board application that allows to visualize the graph computations. Moreover, reference counting means an extra instruction, hence slow down the execution process. Each time objects creates or destroys, the VM/interpreter needs to check if the counts gone to 0. Therefore, reference counting in python has penalize the performance.

5. Summary

Having its own pros and cons of the language mentioned above, Python can be language for the TensorFlow with slight improvement on it, because of the enrich ML libraries and it allows to extent the module by creating it in C++ or C. It is possible to interface with native code and still get the same advantages that Python offers. However, switching from python to entirely other languages would cost more and alternatives would be to program most important part in other languages like C++, Java to get around the bottleneck of the current application. This technique will be reliable, and as well effective to solve the TensorFlow application's problem currently experiencing that are mentioned above.

6. Reference:

[1]. https://stackoverflow.com/questions/1924367/why-do-hedge-funds-and-financial-services-often-use-ocaml
[2]. https://www.tensorflow.org/extend/architecture
[3]. https://wiki.python.org/moin/PythonVsHaskell
[4]. https://mmhaskell.com/blog/2017/8/14/starting-out-with-haskell-tensor-flow
[5]. https://stackoverflow.com/questions/35677724/tensorflow-why-was-python-the-chosen-language