TensorFlow Bindings: Choosing a Language for Ease and Efficiency

Robert Geil, *UCLA Computer Science Department*

Abstract

TensorFlow is a series of high and low level APIs intended to enable machine learning through deep neural networks. Language support for TensorFlow is available across many popular languages, with Python being the primarily supported language. Many other languages have bindings supporting TensorFlow development. In order to improve performance we recommend the usage of the Java bindings, due to the speed of Java, as well as the deeper support for Java within the TensorFlow community as compared with OCaml and Kotlin.

1. A Brief Description of TensorFlow

TensorFlow is a system developed by Google[1] to allow for accessible development of neural networks for deep learning. The central data unit is the *tensor*, which is a multi-dimensional matrix. These matrices can be used to perform linear regressions and can be trained as machine learning models, providing valuable information for a variety of programs. TensorFlow was initially designed for work in Python, and that remains the primary language of usage. However, like many heavy-duty programs the main source code for the resource-intensive portion is written in C and then interacted with in Python. The fact that the core program is written in C allows for other languages to act as wrappers around TensorFlow, creating bindings in languages like Java, OCaml and Kotlin. However, each of these languages has the downside that support is more limited than the Python version, and certain bindings don't exist in certain languages.

2. Improving TensorFlow Performance

TensorFlow performance is typically bounded by the low-level operations done in C to train and perform calculations with the neural networks. However, in the cases of certain requests to an already trained model, it is possible that other points of execution become bottlenecks for performance. In addition, other language offer benefits that Python lacks. Static type-checking, which exists in Java, OCaml and Kotlin, helps to reduce runtime errors, which is a feature lacking in Python, making Python programs less reliable compared to these other languages. In addition, these languages offer different paradigms of programming, which may prove useful for machine learning development.

2.1. The Python Bottleneck

One primary location that can be bottlenecked in small queries is through Python code. Python, as an interpreted language, runs atop an interpreter. This can cause sharp performance hits when compared to execution that is optimized by other strategies, whether these be compiled bytecode, Just In Time compilation, or true compiling to machine code. In cases where Python acts as a bottleneck, other alternatives to this language exist and have supported bindings in TensorFlow

2.2. Java

One major language that TensorFlow has bindings for is Java. Java has the benefits of being a large and mature language, with many developers already familiar with the language. In terms of using Java as a replacement for Python with TensorFlow, the Java bindings have the benefit of being supported as part of the main TensorFlow repository, rather than a less officially supported language. In addition, Java has many speed benefits as compared to Python. Java source code is compiled into bytecode, which is then run atop an interpreter. However, Java also supports Just In Time compiling, meaning that performance is greater than a purely interpreted language. Unfortunately Java is not as fully supported as Python for use with TensorFlow. For example, Java is not covered under the "TensorFlow API stability guarantees."[1], joining other languages that don't have full support, like Go, C++ and JavaScript.

2.2. OCaml

Another language that has bindings for TensorFlow is the primarily functional language OCaml. The OCaml bindings are defined within the public repository

https://github.com/LaurentMazare/tensorflow-ocaml/. OCaml benefits from its multi-paradigm programming, focusing on functional style, but including imperative and object oriented programming[2]. In addition, OCaml code can be compiled to produce speed improvements over Python, and has the flexibility to be run as an interpreted language as well. One major downside of using OCaml is the reliance on Laurent Mazare for support with the bindings. As the project has only 11 contributors, as per the GitHub page, there is significantly less support than Python or even Java.

2.3. Kotlin

Kotlin is a newer language, created by NetBeans[3]. It is interesting in that source code is compiled into Java bytecode and then interpreted by the Java Virtual Machine as if it were Java. This benefits the language in that there is already wide support for Java across many platforms. In addition, Kotlin also benefits from developments made in Java performance including Just In Time compilation, as well as performance features like multithreading that are well supported in Java. One issue with using Kotlin for development with TensorFlow is that the primary bindings have been developed as a one-off project and don't have the greatest support[4]. Other benefits of Kotlin include the fact that as a very modern language, it supports new paradigms of programming, including optional typing, and is very safe as compared to languages like Java, as whole classes of errors are eliminated by non-nullable objects[3]. Kotlin offers both the strength of static type checking, along with the programming speed of implicitly declared types, and the option of explicitly declaring types, which helps to make a more readable and faster to program language.

2.4. Local Machine Learning

Another option to improve the performance of small queries is to respond on the client, rather than on a server. For small queries, it is possible for the fully trained model to be given to the client, from which a query, assuming all the required information was available, could be made. This would help to speed up results available to clients, as network delay can add milliseconds or worse to getting a query response back. Since many platforms have support for executing Python code, or one of the other languages like Java or Kotlin, it is definitely possible to do some predictions in-house, utilizing a fully trained model sent from the server.

2. Recommendations for Improvement

In order to meet the requirements to speed small queries through TensorFlow, I would recommend utilization of Java as the language of choice. Java has many benefits, both specifically for TensorFlow and as a general purpose language. As a language originally targeted towards server development, there is great support for multithreaded approaches to development. This aspect becomes particularly powerful when considering the context of the problem, writing code for server herds to respond to queries. Java has networking to the core, making it an ideal choice for backend development and for this project in particular. In addition, Java is much more performant than Python. By using precompiled bytecode, as compared to on-the-fly interpretation of Python, along with Just In Time compilation, Java provides much more performance and helps to reduce bottlenecks when compared to Python. In addition, TensorFlow support for Java is much greater than the bindings of other languages. Having the Java bindings as a part of the core TensorFlow repository helps focus developers of TensorFlow on Java, building up support for the language. This is true compared to OCaml, where the bindings are maintained by just 11 individuals [5], and even more so compared to Kotlin, where the bindings appear to be a project just worked on by a single person. As TensorFlow continues to evolve, the benefits of having deeper Java support will continue, as new features are more quickly added to the Java bindings than those maintained outside the core TensorFlow community. Despite this, keeping Python as an option for TensorFlow development is still a valuable idea, as Python remains the central language supported by TensorFlow and has the most up-to-date options available to it.

3. References

1. TensorFlow Public Source Code, 2019. https://github.com/tensorflow/tensorflow
2. OCaml Website, 2019 http://www.ocaml.org/
3. Kotlin Website, 2019 https://kotlinlang.org/

4. TensorFlow in Kotlin Native, 2019

   https://juliuskunze.com/tensorflow-in-kotlin-native.html

5. Source Code Repository for OCaml TensorFlow bindings, 2019

   https://github.com/LaurentMazare/tensorflow-ocaml/