Language Bindings for TensorFlow

CS 131

Abstract

This is an executive summary that compares Java, OCaml, and Julia to each other and to Python in application of machine learning algorithms and TensorFlow.

1. Introduction

TensorFlow is an open source library for data flow graphs computation, where each node represents mathematical operations and the graph edges represent multidimensional data arrays (the tensors) [1]. It can be used for machine learning applications, such as neural networks.

The rest of this report will go over the benefits and disadvantages of using Java, OCaml, Julia, or Python to implement the application involving TensorFlow and some machine learning algorithms.

2. Java

Java is strongly typed and statically typed checked, meaning a variable name is bounded to both a type and an object, in which the object has the option of being null. Once a variable has been bounded, it can only then be bound to objects of that certain type. [2] Having a strongly typed language makes the code easier to understand, because there is less unseen behavior, but it does make the code lengthier than some others.

Some benefits of Java being a statically typed language are that it has better code completion and performance in terms of the type constraints offering more opportunities for compiler optimizations.

Java applications run on the Java virtual machine, which tracks what is going on in a Java application and dynamically optimizes it as it goes. A downfall to this is that dynamic runtime can lead to problems; you cannot rely on static compilation and predictable allocation with the JVM. However, Java is easier to use, has rapid developmental use of APIs and standard libraries, and is portable. [3] Java's JVM also handles dynamic resource management, allocating and deallocating memory, maintaining threads, and organizing executable instructions.

Java's binding for TensorFlow is relatively new, so the API is currently experimental and in the process of being updated. It supports features such as, graph reprentation with the class, *Graph*, computations, with the class, *Operation*, and a handful more of other classes that helps execute the graphs and the tensor objects. [5]

For running a predefined graph, Java is capable of creating sessions, using *getSession()*, run queries, and get Tensor results, which results in creating a graph for TensforFlow applications. You can also invoke TensorFlow C++ from Java by using Java Native Interface to directly invoke and use C++ to create the graph and tensors; there is also an open-source library, JavaCpp, that helps with this process. [6]

Java TensorFlow API does not currently allow to train models, only pre-trained models. (Python does).

Event-driven programming can be applicable in implementing TensorFlow's graph construction and functions. Graph construction requires one function per defined TensorFlow ops and functions define subgraphs that may be called in multiple places, which reminds me of Python's asyncio coroutines.

According to the TensorFlow document, "at a minimum, a language binding should support running a predefined graph, but most should also support graph construction," and Java seems to be capable of both. [7]

3. OCaml

OCaml is statically typed, in which you do not need to write explicit type declaration since the compiler infers your types and OCaml can detect inconsistent types. There are also no runtime errors, because of this and makes it easier to write and understand the code, while ensuring reliability. [9]

OCaml has the option of concurrent programming with async, which could be relevant to TensorFlow's running a predefined graph. This mean that in OCaml you can have an operating system thread for each task so that it can be blocked without stopping the entire program or you can have a single-threaded program that runs an event loop. The system thread requires a lot of memory and the programmer to use locks and condition variables to protect shared resources, while on the other hand, an event loop requires various callbacks, which can be confusing. This is similar to Python's aysncio TCP echo client and server use.

TensorFlow bindings for OCaml are still in its early stages, therefore there are features and operators not yet supported for TensorFlow implementation. There is also the possibility of segfaults in the use of the bindings for

OCaml, but overall, OCaml is capable of training a network with various optimizers. [8] OCaml bindings also include API for Graph and building neural networks with FNN API, as seen in the GitHub page for OCaml bindings. [12]

As like Java, OCaml should also be capable of doing the minimum for TensorFlow; however, there is still not a lot of information on using OCaml for TensorFlow, so it will be more difficult to do so than compared to using Java or Python.

4. Julia

Julia is a dynamic programming language used for numerical computing. Dynamic programming is used to solve problems that overlap so that one problems solution can be used in another. Julia is JIT compiled, similar to Java, meaning it can, at its best, match the speed of C. In addition, Julia is dynamically typed, like Python, but also has some static type features that makes it easier indicate that certain values are of certain types.

Julia has coroutines features, lightweight "green" threading used for tasks (like the tasks in Python's asyncio) and can be used for parallelism and cloud computing. Green threads are threads schedule by a runtime library of VM that can share data memory. Julia also has a *Future*, *fetch()*, and *@sync*, features that are similar to Python's asyncio *Base Event Loop*. In addition Julia has *tasks*, which are essentially coroutines to switch between different computations, which can be useful in TensorFlow graph nodes computations.

According to the TensorFlow.jl documentation, Julia with the TensorFlow binding, can do basic math functions, use neural network operations and trainers, and do basic image-loading and resizing operations. There are still features that are not yet wrapped, which include graph execution, control flow operations, and PyBoard graph visualization. [11] Also, from the documentation, Julia uses MNIST to implement the basic requirements of TensorFlow (i.e. the session, graphs, training the model, creating the neural network, etc.).

Julia can interface with Python code using PyCall library, so data between Python and Julia can be shared as well. This is similar to Java and C++ as explained previously of converting C++ to Java for the TensorFlow graphs.10]

Overall, even though Julia is not a widely known or used language, it is quite similar to Python in some aspects, which can then be applied to the implementation of TensorFlow.

5. Python

Compared to the other three languages, Python is the only one that supports the neural network library feature, because there are no plans to support this in any other language but Python, according to TensorFlow's documents. Also, support for gradients, functions, and control flow operations are only currently available in Python.

Python was chosen as the current main supportive language for TensorFlow, because it is the easiest and "most comfortable" to work with and has NumPy that makes it easier to do pre-processing with Python at a high performance before sending it off to TensorFlow to use.

5.1. Java vs. Python

Java and Python are differently typed checked, Java is static while Python is dynamic. Because of this and Python's use indentation, there is a difference for the user to get use to when programming in both languages; with dynamically typed checking, the types are checked at runtime, so you have to make sure you are doing the right thing/using the right type when using Python while in Java the types have to match before it can run. This also makes Python less verbose than Java and overall, easier to write, though it is harder to understand what is going on than if you are using Java.

Java's JVM and JIT compilation allows for a lot of efficiency compared to Python, making its speed faster than Python.

Java and Python are quite different in terms of what features they have, but when asked to do the same thing, they are about on the same level. The preference of which level depends on the programmer and what is wanted to be done.

5.2. OCaml vs. Python

OCaml and Python both have relatively the same aspects for asynchronous implementations and Python can be translated to OCaml to do TensorFlow applications like Python does, with the exception of some API features that OCaml's bindings may currently lack. Despite this, it is difficult to do dynamic programming in a purely functional style, meaning solving problems in a way that treats all computations as mathematical function evaluations is fairly difficult. The main difference between OCaml and Python is that using OCaml's functional language compared to Python is very different and be confusing to understand and get used to at first, compared to how easy Python can be learned to be used.

5.3. Julia vs. Python

Julia beats Python in which it is faster, because of the JIT compilation and type declaration, though Python

could also be fast if you are using extra imported features. Python and Julia both have garbage collectors, so it is convenient if you are switching form Python to Julia or vice versa, in memory management aspects.

Some disadvantages of Julia compared to Python are that Julia is inconveniently array indexed starting at 1, Julia is still new compared to Python, so there is not as much information on Julia as there are on Python, and in general, Python is more widely known and used by other programmers than Julia is.

In terms of using Python or Julia to implement TensforFlow application, there is definitely more features and documentation on Python's TensorFlow binding, but compared to Java and OCaml, Julia seems like the best choice to replace Python.

6. Conclusion

Between Java, OCaml, and Julia, I believe that Julia is the best choice of the three to be used in replace of Python for the TensorFlow application implementation. Though all three languages have similar features to Python, Julia's PyCall should make it a lot easier when switching from Python to Julia. Also, because Julia is more math based than Java or OCaml, which is relevanat to the TensorFlow graphs and nodes computations. As for with machine learning algorithms, it was mentioned that Julia is also a good choice to use in machine learning.

However, I think that any of the languages used in place of Python will still be fine, because performance wise they all have advantages and disadvantages that overlap each other or balance out, depending on what you want to sacrifice over the other and the bindings for each of the three languages are all still in the process of being updated.

References

[1] TensorFlow. (n.d.). Retrieved March 15, 2018, from https://www.tensorflow.org/

[2] Static vs. dynamic typing of programming languages. (2012, April 08). Retrieved March 15, 2018, from https://pythonconquerstheuniverse.wordpress.com/2009/10/03/static-vs-dynamic-typing-of-programming-languages/

[3] Andreasson, E. (2012, August 21). JVM performance optimization, Part 1: A JVM technology primer. Retrieved March 15, 2018, from https://www.javaworld.com/article/2078623/core-java/jvm-performance-optimization-part-1-a-jvm-technology-primer.html

[4] How to Use Sessions. (n.d.). Retrieved March 16, 2018, from https://docs.oracle.com/cd/E19857-01/819-6518/gcxvp/index.html

[5] org.tensorflow | TensorFlow. (n.d.). Retrieved from https://www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary

[6] Lakshmanan, L. (2016, July 19). How to invoke a trained TensorFlow model from Java programs. Retrieved March 16, 2018, from https://medium.com/google-cloud/how-to-invoke-a-trained-tensorflow-model-from-java-programs-27ed5f4f502d

[7] TensorFlow in other languages | TensorFlow. (n.d.). Retrieved March 16, 2018, from https://www.tensorflow.org/extend/language_bindings

[8] OPAM - tensorflow. (n.d.). Retrieved March 16, 2018, from https://opam.ocaml.org/packages/tensorflow/

[9] Strong Static Typing with Type Inference. (n.d.). Retrieved March 16, 2018, from http://www2.lib.uchicago.edu/keith/ocaml-class/static.html

[10] Yegulalp, S. (2017, December 20). Julia vs. Python: Julia language rises for data science. Retrieved March 16, 2018, from https://www.infoworld.com/article/3241107/python/julia-vs-python-julia-language-rises-for-data-science.html

[11] Malmaud, J. (n.d.). TensorFlow.jl. Retrieved March 16, 2018, from https://malmaud.github.io/tfdocs/

[12] L. (2018, February 18). LaurentMazare/tensorflow-ocaml. Retrieved March 16, 2018, from https://github.com/LaurentMazare/tensorflow-ocaml/