CS131 Homework 6 Report

Devyan Biswas
804-988-161

Abstract

TensorFlow is a machine learning library that uses high level languages like Python for the setup of the overall model, the dataflow/graphs, and the variable bindings before using a language like C/C++ to perform the actual calculations and operations on the training data. Within the context of the project, we are now looking towards languages that are better suited to the set up of the models than Python, since it appears that this is where most of the time is being spent. This executive summary will compare Java, OCaml, and Kotlin in the search for a suitable replacement for Python.

1. Java

Java's developers intent for the language was to be highly portable. It is an object oriented language with automatic garbage collection and static typing. However, there are also other things to consider about the language we choose to replace Python.

1.1 Advantages

It is a statically typed language, therefore, all variables must be of clearly defined types at compile time. As is the case with statically vs dynamically typed languages, this can cause the program to be slower, as type checking must all be done before runtime, but also allows for more exhaustive checking. This means that the process of debugging will be much easier. And once compilation is done, it will run faster than a dynamically typed language, like Python, that has to continuously check.

Java is a language that is half interpreted, half compiled. To address the idea of portability, it is important to understand how Java actually compiles/runs code. Simply put, the Java compiler compiles code into byte code that is interpretable by the JVM, for example, and is then able to execute on any computer architecture. There are also optimizations that can be made, such as the Just In Time (or JIT) compiler, which caches machine code translated from this byte code that correlate to frequently used blocks (maybe functions or globals, for example).

The automatic garbage collection, which Java shares with Python, is especially beneficial to have, and so it is good that this feature still remains. This makes it easier for development, but Python's link count garbage collection algorithm can be pricey with frequent accesses/additions/updates to variables, but Java's mark and sweep algorithm for garbage collection could resolve this particular issue by reducing the time spent per access to data structures, instead taking out time overall to do garbage collection, which is faster (in this situation) and can be more optimizable.

Another great feature is the parallelization you can do with Java over Python. A Java based server will be able to handle more throughput than a Python server, and this will be able to handle more tasks in parallel at once.

1.2 Disadvantages

Of course, as mentioned earlier, static type checking can be a hindrance in the early stages of development. The focus can be lost from TensorFlow to making sure functions return correct types, correct operations are performed despite abstraction, etc….

Additionally, with regard to actually setting up an asynchronous server system, it will be more challenging in Java than in Python. Python has libraries built for these applications, such as aiohttp and asyncio, that are tried and tested. In Java, while there are such libraries, is is more niche and far more tedious to implement.

2. OCaml

OCaml is a modern variant of the ML family of languages. It offers support for the functional, object oriented, and imperative paradigms of programming languages. OCaml also boasts a robust typing system, being able to statically check types but also allow for interpretable types and variable function return types. This summary will focus more on its functional benefits, but it is important to note that that is not the only aspect of OCaml. Also, it is important to note that the TensorFlow C API can be used as a language binding, since OCaml is not explicitly supported.

2.1 Advantages

OCaml's memory management is, like Python's, very autonomous, making creation and deletion very easy. OCaml utilizes a mark and sweep garbage collection algorithm like Java, which as discussed earlier, is very effective for this particular

application (save for the benefit of C/C++ direct allocation).

Additionally, OCaml's type checking, as mentioned above, is very robust and helpful with arbitrary and ambiguous functions. OCaml can also be run on an interactive top level interpreter, which can be very helpful for debugging. OCaml also compiles to native binary, which is highly optimized for speed. This also means that it is optimized for every machine it is compiled and run on, which saves on time.

2.2 Disadvantages

OCaml's compilation to native binaries, however, is also a problem. Since it is directly compiled to an executable, OCaml code will have to be recompiled for every new computer architecture it is to be run on. This creates a big problem for a system that is supposed to be highly robust and distributed.

Additionally, like Python, OCaml has very limited core utilization, only utilizing single core and at most allowing for some basic concurrency. There is not really any support for multithreading, and cannot achieve true parallelism, at most being able to have traditional interleaving of tasks.

OCaml's coding style, while its type checking is rather helpful, is not as supportive. It is restrictive in the fact that it is a functional programming language, which can be a difficult jump for those used to Python and C/C++, and the requirements for structure are more asking than Python.

3. Kotlin

Kotlin is a language that can compile to Java code and can compile to the JVM bytecode, Javascript, or native. Structurally, it is very similar to Java. Kotlin is also statically typed, which can be beneficial for runtime as explained earlier. Kotlin was very much intended to be used with Android applications, and offers compatibility with different IDE's, including android studio. Additionally, Kotlin can be worked with in either an object oriented mindset of a functional programming mindset.

3.1 Advantages

The language is built for single core Android applications/single core processes, and so it will be optimized in that regard (though this is a problem Java deals with via multithreading). Additionally, though the language lacks the support that Java has, it does have support for coroutines, which are paramount in the server herd architecture, as they form the basis for the server's asynchronous processing of tasks.

Kotlin allows for an easier transition from Java due to its similarity, and definitely would help with those already familiar in Java. Kotlin's type checking is very similar to Java's, and with regard to TensorFlow specifically, Kotlin has the flex <Any>/<T> type checking, which means it can accept different types of variables.

Kotlin also boasts high portability, has protection from NullPointExceptions, and is slightly more readable.

3.2 Disadvantages

The main issue is that Kotlin is not supported by TensorFlow directly, and the open-source development repositories are not reliable and do not seem to be well maintained. We would need to use the C API, like with OCaml. This may cause problems due to Kotlin's static typechecking, but, as mentioned above, it is robust enough to not be completely impossible/too difficult.

4. Conclusion

Python's main problems arise from dynamic typing, inefficient memory management, and lack of parallelization. Both Kotlin and Java are reasonable solutions to this problem once those are considered. OCaml is far too similar to Python in terms of threading, and while it does have a relatively robust typing system, that may serve as a downside rather than a benefit in terms of function complexity. In short, it does not fit the model for the asynchronous application our product requires.

Based on this, we can choose Kotlin as our best option. Kotlin is built for mobile devices, which is part of our project requirement, and can utilize the single core of the device better than Java, allowing for more request processing (granted, though, it may not be as fast) and the requested option of processing the query on the device instead of the server if the request is small enough. And overall, it shares a lot of similarities to Java, but adds the benefit of being mobile friendly as well as more new and, therefore, more scalable and more robust framework. The only downside is the relatively low diversity of TensorFlow support, even with the C API, but that is a price to pay for the benefits.

5. Sources

[1] https://belitsoft.com/apps-development-services/java-vs-kotlin

[2] https://www.tensorflow.org/guide/extend/architecture

[3] https://www.tensorflow.org/guide/extend/bindings

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

[5] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/java

[6] https://github.com/LaurentMazare/tensorflow-ocaml/

[7] https://kotlinlang.org/docs/reference/