Language Bindings for Tensorflow Packaging

University of California - Los Angeles

Abstract

In this paper, we discuss the limitations of implementing Tensorflow packages in different languages and observer the different performances in order to determine the best choice for our application. We focus specifically on the syntax, ease of implementation, and performance in terms of speed on each of the application. Speed is focused on more than memory due to the fact that making memory larger happens and follows closely to Moore's Law, whereas speed has bottlenecked in recent years.

1 Introduction

At *GarageGarner*, we specialize in creating an application for a large variety of clients who take interest in garage sales. One of our features is the capability to take a panoramic view of a sale and then proceed to send this data back to us. Once we receive this data, we want to implement a search function that gives users a good idea of how much the object they are buying would cost elsewhere, and what the best deal is. However, bottlenecking can be an issue should we not optimally use the AI accelerator built into today's phones.

2 OCaml

OCaml is a functional programming language that has the ability to do object-oriented style programming and is in continuous development by Google. It is a multi-paradigm programming language which extends the Caml dialect of ML.

2.1 Advantages

OCaml is an extremely powerful language and focuses on making the programmer simply create a functional application that can be run on any machine as it compiles down to byte-code. In addition to this, OCaml supports type inference, which allows the programmer to quickly string together code without having to first define types. In addition to its type inference, OCaml runs type-checking before compilation, which allows it to notify programmers when it catches issues in code due to inconsistent types. This language has powerful APIs like that in Python which allow it to have lots of predefined functionality. [3][4]

2.2 Disadvantages

Due to OCaml being a non-traditional language in that it's syntax and style of thinking are different, it tends to be a very difficult language for many programmers to learn, so much of the beginning time would be spent catching programmers up to speed with the language. This is an issue if there are mission-critical fixes that need to be done and a new developer just transitioned into the position. This is a massive issue as we want to prioritize the speed of the application without compromising speed of the debugging process. An additional issue here is OCaml's Just-In-Time (JIT) compiler, which requires extra dependencies. This would cause a larger memory requirement in our application. [3][4]

3 Java

Java is a statically typed language with extremely good memory management and garbage

collection that allows it to run optimally in many situations compared to many other languages. Additionally, Java supports the use of multithreading, which allows it to run multiple processes in parallel without worry of bottlenecking as in Python.

3.1 Advantages

As a fairly old language, Java has a lot of things going for it, in terms of support. Java forces its developers to define its variables, which seems like an issue at first, but can be a very good thing as it forces programmers to be extra careful in checking their own types. Similar to OCaml, Java supports compilation down to byte-code, making it a very versatile language that can be run on many machines. Java's age also gives it more libraries to use with extremely well documented and tested functionality on the libraries, making our program less prone to having unusual bugs. However, this can also be an issue as it means a more restricted approach to using APIs. [1][2]

3.2 Disadvantages

Java's JVM compiler faces similar issues to OCaml's JIT compiler in that it will require extra dependencies to compile. Additionally, the JVM compiler is machine dependent, meaning that it will not be available on every machine. Due to the dependency on Linux machines, installation will take time out of the programmer's days and will cause the development to be rather restrictive. [1][2]

4 Python

Python is a widely used language for TensorFlow and has great capabilities. In recent updates, Python has moved its garbage collection methods to be similar to that of Java's. This is capable of being run on multiple systems including the cloud, where we can use Jupyter Notebooks to simulate and see what would happen with Python were it to use a

4.1 Advantages

Python is a very powerful language with asyncio that allows it to take advantage of machines with single cores and create a scheduler for running tasks. The language is very straightforward to learn, and many programmers tend to enter the industry at least knowing Python. Python can be used to prototype and test quickly and has a lot of libraries and APIs at your disposal to quickly create applications that are functional. Additionally, Python implements type-inference and checking similar to that of OCaml. The most noticeable trait Python also supports is development without using curly-brackets, square-brackets, and parentheses, allowing programmers to focus on if their logic is correct rather than if they have matched the brackets properly. [6][8]

4.2 Disadvantages

Python is an extremely high-level language that does not allow much control or optimization using the AI accelerator system in many modern machines. Although we can prototype in Python very well, this causes us to also have a large amount of latency and causes it to be slow. Additionally, Python does not support multithreading, which means it will run suboptimally for machines that have multithreaded support or multiple cores. [6][8]

5 Dart

Similarly to Java, Dart is an Object oriented language with capabilities of multithreading and multi-paradigm similar to OCaml. It's syntax itself, mirrors that of C, however.

5.1 Advantages

Dart is a promising candidate for the application development. Firstly, it takes all of OCaml and Java's advantages and integrates them within its own language. However, the language is much more straightforward, following closely to that of C. Furthermore, it takes advantage of OCaml and Python's asyncio library and allows us to optimize in systems that have no multithreaded support. This low-level gives programmers a lot of room to work with and optimize the system as per device. From this, we can get the best performance for each different system. As it also compiles into byte-code, this is extremely portable. [5][7]

5.2 Disadvantages

Due to Dart's young age, the main issue we will run into is the lack of support on Dart's part. It still has many functions that lack testing, which means we are much more likely to run into an issue that no one before has. Additionally, Dart is a language not normally binded to systems, which means a higher number of dependencies for this program will exist. [5][7]

6 Conclusion

From all of this, I believe that our best choice is to go with the Dart language first, Java second, then OCaml, and lastly Python. As we are focusing on making sure our implementation does not bottleneck, I believe it is best that we focus on learning the low-level languages and implementing their optimization principles in order to create a thoroughly supported application. We want to highlight the strengths of a system in our code and make sure that it is being used to its full potential. Although Dart is a younger language than Java, I chose it due to it supporting more features to allow better functionality in the code and optimizing performance in a way that incorporates all of the strengths from the other three languages.

7 Works Cited

[1] Chandrakant, K. (2019, October 22). Introduction to Tensorflow for Java. Retrieved from https://www.baeldung.com/tensorflow-java

[2] Landup, D. (2019, June 24). How to Use TensorFlow with Java. Retrieved from https://stackabuse.com/how-to-use-tensorflow-with-java/

[3] LaurentMazare. (n.d.). LaurentMazare/tensorflow-ocaml. Retrieved from https://github.com/LaurentMazare/tensorflow-ocaml

[4] Mazare, L. (n.d.). Deep learning experiments in OCaml. Retrieved from https://blog.janestreet.com/deep-learning-experiments-in-ocaml/

[5] Muthu, N. (2019, May 19). Real-Time Object Detection with Flutter, TensorFlow Lite and Yolo - Part 1. Retrieved from https://blog.francium.tech/real-time-object-detection-on-mobile-with-flutter-tensorflow-lite-and-yolo-android-part-a0042c9b62c6

[6] Python TensorFlow Tutorial - Build a Neural Network. (2020, January 18). Retrieved from https://adventuresinmachinelearning.com/python-tensorflow-tutorial/

[7] TensorFlow Lite. (n.d.). Retrieved from https://www.tensorflow.org/lite

[8] Tutorials : TensorFlow Core. (n.d.). Retrieved from https://www.tensorflow.org/tutorials