CS131, Spring 2018
Homework 6 Report

1 Abstract

Tensorflow is an open-source machine learning library developed by Google[3]. It currently leads in popularity, especially since Theano, its primary alternative which it borrowed much of its declarative syntax from, is not being actively developed anymore[4]. It excels in comparison to NumPy in terms of performance and development speed due to its CUDA GPU support, and automatic differentiation capabilities for efficient back propagation[5].

Although Python and C++ bindings are officially supported, a number of other bindings have been developed by the open-source community. These include bindings for Java, OCaml, Crystal, F#, and Vala. In this review, we will briefly examine the suitability of some of these languages with respect to ease of use, flexibility, generality, performance, and reliability.

2 Background

The predominant language in use with Tensorflow is Python, which gained popularity due to its dynamic typing, regular expressions, and power of list comprehensions inspired by functional languages. This gives it a concise style that's easy to follow. This attracted the scientific community to build a numerical analytic and linear algebra library, NumPy[2]; and Matplotlib, inspired by the MATLAB 2D plotting library. Together, they gave Python functionality comparable to MATLAB. Later, Pandas was added to provide econometric data analysis[1] with data structures similar to statistical language R. The Scikit-Learn package added easy-to-use machine learning models.

Python, then, was an obvious choice for machine learning prototyping. Its excellent libraries, including web frameworks Django and Flask, and interactive visualization library Bokeh, gives it the generality to excel in multiple domains.

The open-source nature of these library give them disparate qualities of documentation that's often overcome by strength of the online community. For example, the Matplotlib documentation is poorly maintained. This slightly affects reliability. More pertinent, however, is that while dynamic typing is concise, it has the tendency to hide run-time errors in larger programs. Interpretation also means that Python slower in performance, although superset Cython compiles to C with similar syntax[6].

Note, however, that while classes exist in Python, it is not a truly object-oriented language. For example, it was not designed to allow complete encapsulation. Private variable names are mangled by the compiler, but not inaccessible by a savvy coder.
Therefore, while flexible and easy to use, there is some loss of security.

From version 3.4, Python introduced `asyncio`, for asynchronous operation on an event loop[7]. This allows single-threaded concurrency using non-blocking co-routines by prefixing a function name with `async` and return value with `await`. The event loop works by adding markers to a queue that is popped when the call stack is empty.

3 Java

In contrast, Java is a purely object-oriented language. This opinionated style does not allow for mixed paradigms without boilerplate code, since every method must be wrapped in a class. Thus, flexibility suffers. Recent versions of Java introduced some of the functional programming features to make up for this, such as lambdas from version 8[8].

Positively, it has a syntax similar to that of C; therefore, it is familiar to most programmers. Its specification doubles as a fully comprehensive documentation[9]. It's statically typed, and it disallows pointers, typically, which increases reliability at the cost of flexibility, since it makes memory bugs much less likely. It also uses garbage collection and a just-in-time compiler to compile critical sections to C. Therefore, compared to Python, Java is less flexible and slightly harder to use, since it encourages a specific paradigm, but it has better performance and reliability.

Since Java has been one of the most popular languages by far, it has been actively supported and developed, such that it generalizes to many domains. For example, Google's Android is built on Java, and it is also popular for desktop applications. One disadvantage is that Java is proprietary, and owned by the litigious Oracle, ever since it acquired Sun in 2009[10], and consequently, Java and Solaris. Although this is unlikely to affect smaller companies, Oracle had sued Google over Android using Java in 2010[11].

Unlike Python's `asyncio`, Java hides details of the event loop by exposing only events and event listeners. Implementation details are hidden in its many interfaces, but at some level event handlers insert a marker on an event loop.

4 Ocaml

Ocaml is an impure functional language with some procedural features. It even supports objects and polymorphism, although these are not idiomatic uses of the language. Like

Python, it can run from a read-eval-print-loop (REPL), an interactive interpreter, that makes it suitable for quick prototyping with immediate feedback. Unlike Python, Ocaml compiles down to bytecode or native code with `ocamlc` and `ocamlopt`, respectively. This makes it extremely fast, and memory efficient when the program is properly implemented with tail recursion.

While the impurity lends some flexibility, the functional style is often terse and unintuitive. It is statically typed so it is reliable; however, types are usually left out and inferred by the type inference engine. When included, it looks rather busy. Regardless, larger programs are harder to follow, making it harder to use. All errors are caught at compile time, since the compiler complains when case matches are not exhaustive.

Ocaml has a decent assortment of libraries, like Ocsigen web framework, Owl numerical library, and PLPlot and Archimedes for plotting. But its rigidity is unlikely to attract the mainstream. It's relative unpopularity, 82 on the May 2018 TIOBE index[12] means open source development is not as active. Generality is present, but not as robust as Python or Java.

Type inferencing is integral to Ocaml's `async` package, which introduces a `Deferred` type as a placeholder for when non-blocking functions are still running. `bind` takes the return and passes off to another non-blocking function[13]. The process is similar to Javascript's Promises. This is harder to use compared to Python's `asyncio`.

5 Crystal

Crystal is a newer language that tries to improve on interpreted languages with a few key features. Specifically, the slogan claims it has the speed of C, but the syntax advantages of Ruby. Unfortunately, it doesn't have the other advantages of Ruby, like object literals and meta-programming. Nevertheless, it achieves a succinct style by using type inferencing as in functional languages, as well as blocks and lambdas. It dodges null reference errors by doing a union with special type `Nil`. It uses fibers for concurrency, inspired from Go. It supports pointers, have native bindings for C, and compiles down to it[14].

Thus, it attempts to marry popular functional programming features with the object oriented paradigm, with support for classes along with functions as first-class citizens, lambdas and currying. Like Ruby's gems, Crystal introduces shards to easily distribute libraries, like bare bones web framework Amber[15]. However, being a relatively new language, support isn't as active, and there's no guarantee that the language would still be alive in the future. So while it has ease of use, flexibility, performance, and reliability at least so far,

future reliability and generality is a huge unknown.

6 Conclusion

By purely language features, Crystal seems like the most suitable alternative to Python for this task. It has type safety, easy multi-threading, and a beautiful style, on top of being fast. However, Crystal is relatively new and unpopular, so its continued development is up in the air. It would be a bad idea for any company to commit to such a language unless it is prepared to potentially become the main contributor to the open source repository.

For that reason only, if speed is the main concern, I would select Java. Java has the longest history of these languages (although Ocaml, being a relative of Standard ML, is close!). It is guaranteed to continued support for years to come. It has the largest collection of libraries and clear documentation. A smaller company is unlikely to attract the attention of Oracle, but definitely cannot afford to tie down its entire architecture to a language whose future is uncertain.

7 References

[1] Volthihong, Phuong, et. al. *Python: End-to-end Data Analysis*. Birmingham, Packt, 2017.

[2] Millman, K. Jarrod, and Michael Aivazis. "Python for Scientists and Engineers". *Computing in Science and Engineering*, vol. 13 no. 2, 2011. pp. 9-12. https://www.computer.org/csdl/mags/cs/2011/02/mcs2011020009.html. Accessed 28 May 2018.

[3] "Tensorflow Architecture". *Google Tensorflow*. https://www.tensorflow.org/extend/architecture. Accessed 28 May 2018.

[4] Bengio, Yoshua. "MILA And the Future of Theano". *Google Groups*, 28 Sept, 2017. https://groups.google.com/forum/#!msg/theano-users/7Poq8BZutbY/rNCIfvAEAwAJ. Accessed 28 May 2018.

[5] Aloni, Dan. "Back propagation with Tensorflow". *Dan Aloni's Blog*, 6 Mar 2017. http://blog.aloni.org/posts/backprop-with-tensorflow/. Accessed 28 May 2018.

[6] "About Cython". *C-Extensions for Python*, 13 Apr 2018. http://cython.org. Accessed 28 May 2018.

[7] "18.5. asyncio – Asynchronous I/O, event loop, coroutines and tasks". *Python Software Foundation*, 28 May 2018. https://docs.python.org/3/library/asyncio.html. Accessed 28 May 2018.

[8] "Java SE 8: Lambda Quick Start". *Oracle Tech Network*. http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Lambda-QuickStart/index.html. Accessed 28 May 2018.

[9] Gosling, James. "The Java Language Specification: Java SE 10 Edition". *Oracle*, 20 Feb 2018. https://docs.oracle.com/javase/specs/jls/se10/jls10.pdf.

[10] Tillman, Karen. "Oracle Buys Sun". *Oracle*, 20 Apr 2009. http://www.oracle.com/us/corporate/press/018363. Accessed 28 May 2009.

[11] Woolf, Nicky. "Google wins six-year legal battle with Oracle over Android code copyright". *The Guardian*, 28 May 2016. https://www.theguardian.com/technology/2016/may/26/google-wins-copyright-lawsuit-oracle-java-code. Accessed 28 May 2018.

[12] "TIOBE Index for May 2018". *TIOBE*, May 2018. https://www.tiobe.com/tiobe-index/. Accessed 28 May 2018.

[13] Minsky, Yaron, et. al. *Real World Ocaml*. Sebastobol, O'Reilly, Nov 2013.

[14] "The Crystal Programming Language". *Crystal Lang*. https://crystal-lang.org/. Accessed 28 May 2018.

[15] Perez, Elias. "Why you should try Amber Framework". *Codeburst*, 7 Jan 2018. https://codeburst.io/amber-crystal-web-framework-1d76f5abfe2b. Accessed 28 May 2018.