Evaluation of Different Languages for Application Involving Machine Learning

Jessica Cheng

Abstract

We compare Flutter, which is written in Dart, with Python, OCaml, and Java on the criteria of ease of use, flexibility, generality, performance, and reliability in order to find the best fit for developing an application that uses image recognition to identify objects and prices in a photo of a garage sale before performing price comparison on the objects.

Introduction

We are trying to produce GarageGarner, an application for people who want to buy things from garage sales. Ideally, a user would be able to take a panorama of items and their prices at a garage sale, and the application would tell them what the best deals present at the sale are. To do so, there has to be digit/price recognition. Because sending images to a central server for computation is too slow and uses too much bandwidth, the application shall use AI accelerator technology on cell phones to process images.

The technology we have chosen to do run models on cellphones with is TensorFlow Lite. The question that remains is what technology to use for writing and running the application's interface: the Flutter interface toolkit written in Dart, which has a plugin called tflite, alongside other languages OCaml, Java, and Python.

Dart is a language released by Google in 2011, and has C-like syntax. It is multi-paradigm, meaning that it can support more than one programming paradigm—in Dart's case, it supports imperative programming (like C, where code describes how a program operates), object-oriented programming (like Java and Python, which supports data structured as objects, and objects can access and even modify its own data), functional programming (like OCaml and Scheme, where programs are more akin to a series of mathematical computations that produce the same results for the same inputs each time). These features make for interesting comparisons against other programming languages.

Flutter is a mobile user-interface toolkit that is written in Dart, and on most mobile machines Dart can be used to compile it for ARM.

tflite is a Flutter plugin using TensorFlow Lite, a framework for inference on mobile devices. It will be able to support image classification and object detection, which are required for our purposes.

1. Application Design

1.1 Local vs Server

In the design of our application, we must decide how much computation should be computer on the client side and on the server side. Some tasks should be done by the mobile devices only, others are done by pre-processing on the client side to prepare for heavier computation on the server end, while other tasks are done entirely on the server side.

One possible design is to have a pre-trained model. When a user inputs a new photo, the application automatically detects the items and their associated price tags. Here, the user can verify the results of the recognition, and this user input would generate new data to train.

1.2 Need for Accelerators

AI accelerators are needed for our application because a GPU typically runs 10 times faster than the CPU on neural network models, since there are a lot of matrix multiplications involved and GPUs are able to do these computations in parallel. New, higher-end phones have the capability to do these computations quickly.

1.3 Important Considerations

- Since we are developing an application, the client-side user interface must exist.
- Integration with machine learning technologies via AI accelerators is also important. In other words, the languages must be able to support this reasonably.

2. Ease of Use

2.1 Dart

The Flutter user interface has many built-in functionalities including UI rendering components, device API access, testing, and many libraries. This is very beneficial for app development as this means there is good support for our application development goals, allowing for smooth development without the need to look for third-party libraries.

Further, tflite has pre-trained models that are capable of image recognition, so the labor of training image recognition models is mitigated.

However, Flutter lacks easy installation with native package managers for specific operating systems. This may mean delay in getting the project off the ground as all developers will need it to run properly on their machines manually. Also, because Dart is a nascent language compared to the others in this paper, there are limited resources for learning Dart in comparison, upping the difficulty of using it.

2.2 Python

Python offers high ease of use for our purposes. It is among one of the easier languages to learn, is easy to integrate with other programming languages (if there is a need for other

software components to be written in another language), and have many built-in libraries that support scientific computations, which will be critical for TensorFlow.

Also, tflite has a built-in Python API, so we can take advantage of the pre-trained models, as is the case with Flutter.

2.3 OCaml

OCaml is not a very commonly used language in day-to-day software engineering, given the fact that it is a functional programming language, and thus there has not been a lot of effort in integrating machine learning into the language. Furthermore, there are also very few resources in learning how to do machine learning in OCaml, meaning that there is a very steep learning curve that does not facilitate good ease of use.

There is a framework for OCaml bindings for TensorFlow, but it is experimental in nature. Jane Street, which develops its software in OCaml, notes that 'the machine learning ecosystem in OCaml is nowhere as well-developed as it is in Python'.

2.4 Java

Java, like Python, has the advantages of having many built-in libraries and ease of learning. Also, tflite has a built-in Java API, so we can take advantage of the pre-trained models, as is the case with Flutter and Python.

3. Flexibility

3.1 Dart

Dart is extremely flexible. It can be compiled into JavaScript to be executed by browsers, and allows code reuse between mobile and web apps, with as high as a code reuse rate of 70%. Dart can also be compiled as a statically-typed language ahead of time, or as a dynamically-typed language that is compiled just in time. Furthermore, Dart also has its own virtual machine, which means that Dart can execute code natively.

3.2 Python

Python is quite flexible in some ways, such as its dynamically-typed nature removing the need for compilation, allowing for a rapid developing environment. Also, the ducktyping design of the language allows for high design flexibility (such as class designs) with a lower chance of runtime errors.

However, GUI applications and native mobile apps in Python require additional libraries and frameworks to develop, meaning that third-party frameworks and libraries are needed to enhance its functionality and performance, making it less flexible than Dart.

3.3 OCaml

OCaml is the least flexible entry here. There is no good option for native mobile apps, and requires mobile cross-compilers from third party sources with limited maintenance efforts.

Cross-platform development is possible, but the tool available does not use native widgets, thus significantly limiting the ability for widget integration—specifically, machine learning integration, which our application needs.

3.4 Java

Java offers a lot of flexibilities. There is also good native mobile development support, allowing for great flexibility in development. One can feasibly develop mobile apps, server apps, and web apps along with handling tflite computations with just Java.

Programs can easily be run across servers, making distributed computations simpler for our purposes. Also, the Java Virtual Machine allows for runtime analysis and optimizations that can be customized for our needs to allow for better performance.

4. Generality

The general machine learning process involves ETL (extract-transform-load), meaning that data can be represented in a different context than the sources. This fact makes generality important as we would want to reuse code for different data types.

4.1 Dart

Dart allows for generality as it supports dynamic typing, meaning that in a large machine learning program, there is no need to worry about keeping track of class types, even for data being sent across the network for additional computation.

4.2 Python

Python has good generality because of its ducktyping feature. Again, with Python there is less chance for encountering runtime errors based on type—as long as object classes have similar functionalities, machine learning computations can be performed on them without many problems.

4.3 OCaml

OCaml has good generality because it is a functional programming language. Because of this, much code reuse can be achieved.

4.4 Java

Java is not very general because it is statically typed. In order to avoid errors, it is the developer's responsibility to be aware of the data members being used by tflite are such that all type statements agree, which increases the complexity of the program produced.

5. Performance

tflite allows for optimized on-device performance, so all languages that support it—Dart, Java, and Python—can take

advantage of AI accelerators on clients' phones to boost performance.

5.1 Dart

Dart can be compiled in both AOT and JIT, which is useful because during development, JIT can be used to facilitate a rapid development, while AOT can be used during release for better speed optimization. Other good performance features include concurrent garbage collection, which means garbage collection without the need to perform locking.

Also, Dart can achieve concurrency using Isolates, which are independent workers like threads but don't share memory, and can take advantage of multicore machines. On the flipside, communication between Isolates must be serializable, limiting concurrency.

With that being said, Dart performs much slower than Java, for example, in many benchmarking tests.

5.2 Python

Python is the weakest here in terms of performance. Because Python is dynamically and strongly typed, it must infer types at runtime, which increases the performance overhead. It also uses a global interpreter lock that does not allow multiple accesses to a single object, meaning that it cannot be run on multiple cores on a single processor.

5.3 OCaml

OCaml is known for its strong performance as it performs well in benchmarks.

For our purposes specifically, OCaml is good for prototyping in machine learning. Machine learning consists mostly of compositing functions over a matrix, so in essence OCaml's focus on functional implementations makes these computations perform well.

However, there are significant performance drawbacks. OCaml uses a global interpreter lock, meaning that it is confined to a single core. Also, there is no good support for tflite-plugins that can take advantage of AI accelerators on phones.

5.4 Java

One of Java's greatest strengths in the context of performance is its concurrency features. Multicore and multithreaded programming are supported in Java, using the processors to their full capabilities.

However, there is overhead in interpreting Java's compiled bytecode into machine instructions to be executed by the machine, making it slower than native executables. This performance cost is slightly offset the JVM's runtime optimizations attempt to boost application performance.

6. Reliability

6.1 Dart

Flutter has good reliability features such as hot reload, which allows it to reload and resume code execution in under a second, with the application state mostly preserved. This is a good feature for our application because a lack of reliability or responsibleness would decrease a client's likelihood in using our application again.

6.2 Python

Python is a reliable programming language, given its age and popularity, though hot reload is not as readily supported as it is by Dart. It is also less reliable than statically-typed languages because type errors can occur during runtime, leading to a higher chance of crashes on the client side (though this can be mitigated by having good programmers!)

6.3 OCaml

OCaml is very reliable because it has static type checking, which eliminates the danger of runtime type errors in a vast machine learning program. It also has mostly deterministic behavior (aside from garbage collection) due to its functional nature.

6.4 Java

Java is quite reliable due to its robust features. It has static type checking, and also does not allow multiple inheritance, which eliminates inheritance errors prevalent in C++.

However, Java cannot be reliable in its *performance*. The JIT compilation, alongside the runtime decisions made by the JVM, means that there can be much variation between runtimes.

7. Conclusions and Recommendations

Dart, Python, OCaml and Java all excel in some areas at the expense of others. I would not recommend seriously considering OCaml due to its lack of support for both application development and machine learning when compared to its peers, which all can take advantage of tflite without hassle. Amongst the three other languages, Dart with Flutter is most able to leverage robust web support with some of the best features of others. It can take advantage of the fast development of just-in-time compilation and the performance gains of ahead-of-time compilation; is flexible between flatforms, web and mobile; it is general and good for machine learning. In my opinion, these advantages make up for the one-time need for complicated installation and performance costs (when compared to Java) and thus Flutter should be used to build our application.

References

[1] *FAQ,* Flutter. https://flutter.dev/docs/resources/faq

[2] *Matrix-Matrix Multiplication on the GPU with Nvidia CUDA,* QuantStart.

https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA/

[3] *An Exclusive Look Inside Apple's A13 Bionic Chip,* WIRED. https://www.wired.com/story/apple-a13-bionic-chip-iphone/

[4] *Flutter vs React Native: A Developer's Perspective*, Nevercode. https://nevercode.io/blog/flutter-vs-react-native-a-developers-perspective/

[5] *OCaml bindings for TensorFlow*, Laurent Mazare. https://github.com/LaurentMazare/tensorflow-ocaml

[6] *Deep Learning Experiments in OCaml*, Jane Street Tech Blog. https://blog.janestreet.com/deep-learning-experiments-in-ocaml/

[7] *Why Flutter Uses Dart,* Wm Leler. https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf

[8] *How Flexible is Python?,* Harri Srivastav. http://www.allaboutweb.biz/how-flexible-is-python/

[9] *Revery*, Bryan Phelps. https://github.com/revery-ui/revery

[10] *14 Programming Languages for Mobile App Development,* Ian Blair. https://buildfire.com/programming-languages-for-mobile-app-development/

[11] *dart:isolate library*, Dart. https://api.dart.dev/stable/2.3.1/dart-isolate/dart-isolate-library.html

[12] *Dart versus Java Fastest Programs*, The Computer Language Benchmarks Game. https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/dart-java.html

[13] *OCaml versus Java Fastest Programs*, The Computer Language Benchmarks Game. https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/ocaml-java.html