Language for TensorFlow Binding

Terry Ye - UCLA

1. Overview of TensorFlow

TensorFlow is the library that is used for dataflow programming and especially machine learning application. Its architecture consists of client, distributive master and workers. Client is created by the user to build a computation graph that can define the computation and be passed to master for evaluation later. Distributive master can partition the whole graph into small pieces, distribute them to workers and initiate the executions of those in workers. Worker can use the kernel implementation to do the calculation and share the result among other workers as well. The key features of implementing TensorFlow is to run predefined graphs and construct the graph-like model by itself based on the input dataset.

2. Java as replacement of Python

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented.

2.1 Ease of use

From a lazy programmer's perspective, Python is a lot easier to use than Java, which might be one of the biggest features and advantages of Python. Python uses dynamic (duck) typing while Java uses the strict static typing. Python determines the type of variable based on its behavior and the user doesn't have to define it and follow it all the way, so the user has to endure the overhead of declaring variable type and following it strictly in Java. Another thing that makes Java harder to use is that type casting in Java is stricter and harder and doesn't allow data container of different types so the user has to implement it in a class by himself. This is important because in TensorFlow there will be input data and feature columns that may require a lot of type casting and it would be easier to store the information if there exists container that can take different types.

2.2 Flexibility

Java is also less flexible compared with Python. The variable type is Java is defined when declaring variables and cannot be modified easily. The user has to carefully go through all related code of the variable if it wants to change while Python can change the type dynamically. The biggest inflexibility of Java I think is that the operators in Java cannot be overridden and modified to have custom behavior. This might be troublesome when implementing TensorFlow because there might be chance that the user of TensorFlow wants to try its own math model with different operation definition

2.3 Generality

Java is not a language that can be used very generally. The base class in Java must be declared 'abstract' if a class wants to inherit it. In a large program like TensorFlow, it is kind of hard to keep track of which classes are declared as 'abstract', so the code may not be reused easily because the new user has to find the original 'abstract' ones by himself. Plus, Java has to use a lot of external IDE to implement the TensorFlow, which makes it harder for others to know the code.

2.4 Performance

Although none of them would be an optimal choice considering performance, the performance of Java code would be better than Python because first Python's dynamic typing requires the interpreter to check the types of variables during runtime while Java's types are already defined and checked during compile time. Then the most important part of Java's better performance is that JIT compiler(Just In Time) can have better optimization of the code by translating bytecode into native machine code while Python relies on an interpreter and can do just limited optimization. JIT requires processor time and memory usage, so if there are a large number of methods getting compiled, the startup time for these will be great. However, in this TensorFlow implementation, the code is dealing with relatively small data so it will not affect Java's performance that much which makes it still better than Python's.

2.5 Reliability

Java has less possibility of problem occuring because of its strict static type checking system, so the behavior of code is more predictable. Java also doesn't allow multiple inheritance of classes which prevents a "diamond problem" which is an architecture of 4 classes in the diagram at last page. When both B and C override a method in A and D doesn't have a definition of the method itself, which method D would inherit for this method remains uncertain. This is a huge but really common problem for multiple inheritance that might make the code unreliable and unstable and is where Java shines.

3. Ocaml as replacement of Python

Ocaml is a functional programming language with object-oriented construct.

3.1 Ease of use

Both Python and Ocaml are easy to write I think, but with different styles of coding. Although Ocaml uses strong static type checking, users don't need to write variable type definition in Ocaml thanks to its type inference. Ocaml code looks readable and simple and is not hard to write itself, but the biggest problem is that most programmers are not very used to functional programming idea, so they cannot fully exploit Ocaml's ability.

3.2 Flexibility

Ocaml has generic type which can be matched to any specific type based on the matched pattern which makes it flexible to modify. And user-defined type is really easy to implement and use in Ocaml as well, so the user who use it only has to modify the type definition if they want to. Ocaml also has labels that can document the code and gives more flexibility to function application. Furthermore, normal variant in Ocaml is assigned to a unique type when defined, so a constructor can only belong to one type and thus is not very flexible. However, Ocaml provides polymorphic variants that doesn't belong to any specific type and is flexible to change. These features are really helpful when implementing TensorFlow. For example, the fnn.ml for handling failure cases below takes uses of user-defined type and polynomial variants to make the code easy to change. So Ocaml is approximately equally flexible compared with Python.

3.3 Generality

Ocaml is general because it is a relatively pure functional programming language that focus on function, so it is clearer and easier to reuse the code from others.

3.4 Performance

Ocaml has much better performance compared with Python. Generally, it uses a compiler to generate assembly code for the program which is a lot easier and faster for the computer to execute than Python interpreters' commands. When generating code, Ocaml uses method like inlining which takes a function and expands it from its definition. This will improve the performance as well because it avoids the overhead of an extra function call and give the optimizor more chances to optimize.

3.5 Reliability

Ocaml should be more reliable than Python. First, it uses static type checking system which prevents the type error during runtime. Second, Python cannot distinguish between values and null at compile time but Ocaml can do so and prevent a program crash because of NullPointerException.

4. Haskell as replacement of Python

Haskell is a standardized, general-purpose purely functional programming language, with non-strict semantics and strong static typing.

4.1 Ease of Use

Haskell looks kind of like Ocaml as it is also static type checking but with type inference, so it doesn't have to define variable type. The rule of types is actually stricter in Haskell. There is no type coercion at all so a converting function has to be called every time when there is type conversion which makes it really troublesome to write especially TensorFlow code where there are a lot of calculation between int and float. Also like Ocaml, another big problem of Haskell is that it requires the programmer to have decent math knowledge which makes it suitable for only part of people.

4.2 Flexibility

Haskell is really not flexible as all objects in it are immutable, so instead of altering existing value, altered copies are created. And because it is like a pure functional language, there is no efficient mutable array to easily store data which is a key problem when trying to implement TensorFlow in Haskell.

4.3 Generality

Haskell is great when talking about generality. Its codes are referentially transparent which makes others easy to reuse because they don't need to read the whole source code to determine if there's some mutable state or subtle reference. There are only few constructs, so the amount of special-case syntax is small and people can easily know the language once know the rules for function application.

4.4 Performance

Performance of Haskell should be close to the one of Ocaml and also a lot faster than Python's. Like Ocaml, it can use compiler like GHC to compile the code into bytecode and outperforms Python's interpreter approach. It also use similar techniques like profiling and inlining to improve its performance.

4.5 Reliability

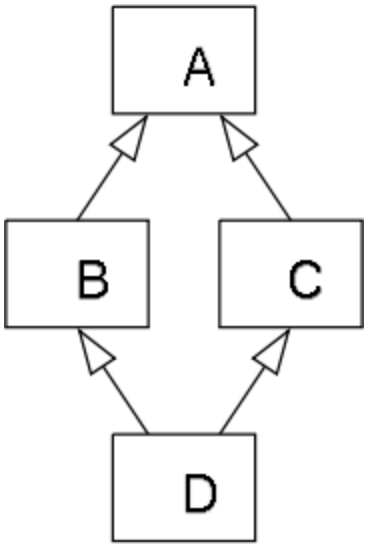
Haskell is really reliable as it completely eliminates the dangerous dynamic checking and type coercion, so it is impossible to have problem like type incompatibility. Haskell also forces functions and programs to not affect the state of any other program or function, making it more reliable.

5. Conclusion

The three languages above all fulfill the basic requirement of implementation --- better performance than original Python code. Among those, I think Ocaml seems like the most competitive language to implement TensorFlow considering its speed and flexibility. Java is also ok but the performance doesn't improve much compared with Ocaml and it's sometimes too verbose to develop. Haskell seems like the least suitable for the task as it is so strictly ruled that makes it hard to write.

6.References

[1] Mani Goswami, *An Introduction to TensorFlow Architecture*,
https://www.slideshare.net/ManiGoswami/into-to-tensorflow-architecture-v2
[2] Thomas Leonard, *Replacing Python: Candidates*,
http://roscidus.com/blog/blog/2013/06/09/choosing-a-python-replacement-for-0install/
[3] Kevlin Henney, *Java vs Python: Which One Is Best For You*,
https://blog.appdynamics.com/engineering/java-vs-python-which-one-is-best-for-you/
[4] *Labels and Variants*,
https://caml.inria.fr/pub/docs/manual-ocaml/lablexamples.html
[5] *Performance and Profiling*,
https://ocaml.org/learn/tutorials/performance_and_profiling.html
[6] *Python vs Haskell*,
https://www.slant.co/versus/110/1537/~python_vs_haskell




```
type init = [ `const of float | `normal of float | `truncated_normal of float ]

type pool =
  { filter : int * int
  ; strides : int * int
  ; padding : [ `same | `valid ]
  ; avg_or_max : [ `avg | `max ]
  }
```