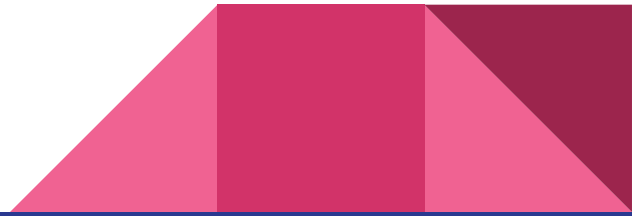
JAX and Flax

Background

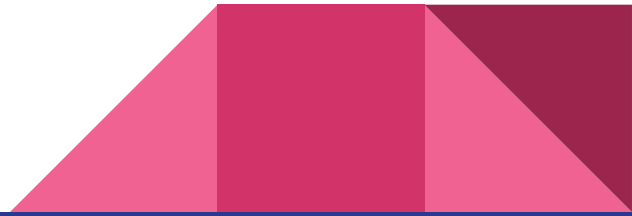
What is JAX?

- JAX is Autograd and XLA, brought together for high-performance numerical computing and machine learning research. It provides composable transformations of Python+NumPy programs: differentiate, vectorize, parallelize, Just-In-Time compile to GPU/TPU, and more.
- In simpler words: JAX is NumPy on the CPU, GPU, and TPU, with great automatic differentiation for high-performance machine learning research.



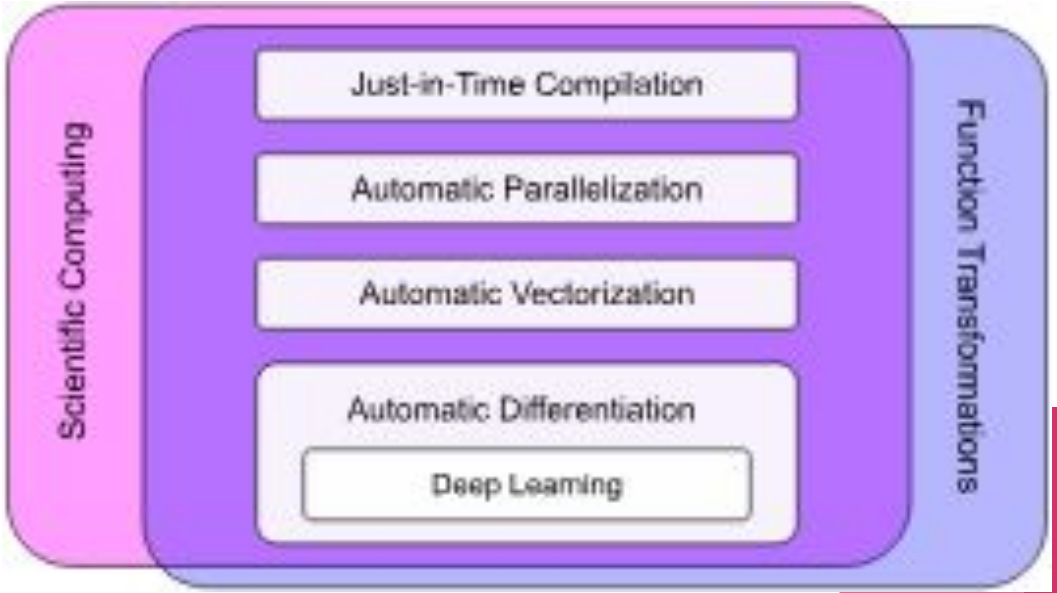
What is XLA?

- XLA: Accelerated Linear Algebra, lies at the foundation of what makes JAX so powerful. Developed by Google, XLA is a domain-specific, graph-based, just-in-time compiler for linear algebra.
- It significantly improves execution speed and lowers memory usage by fusing low-level operations.



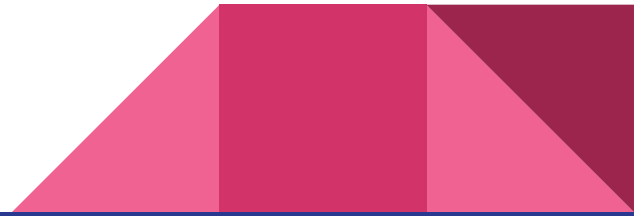
What can JAX be used for?

JAX is a high performance, numerical computing library which incorporates composable function transformations[1]. As we can see, Deep Learning is just a small subset of what JAX can do:

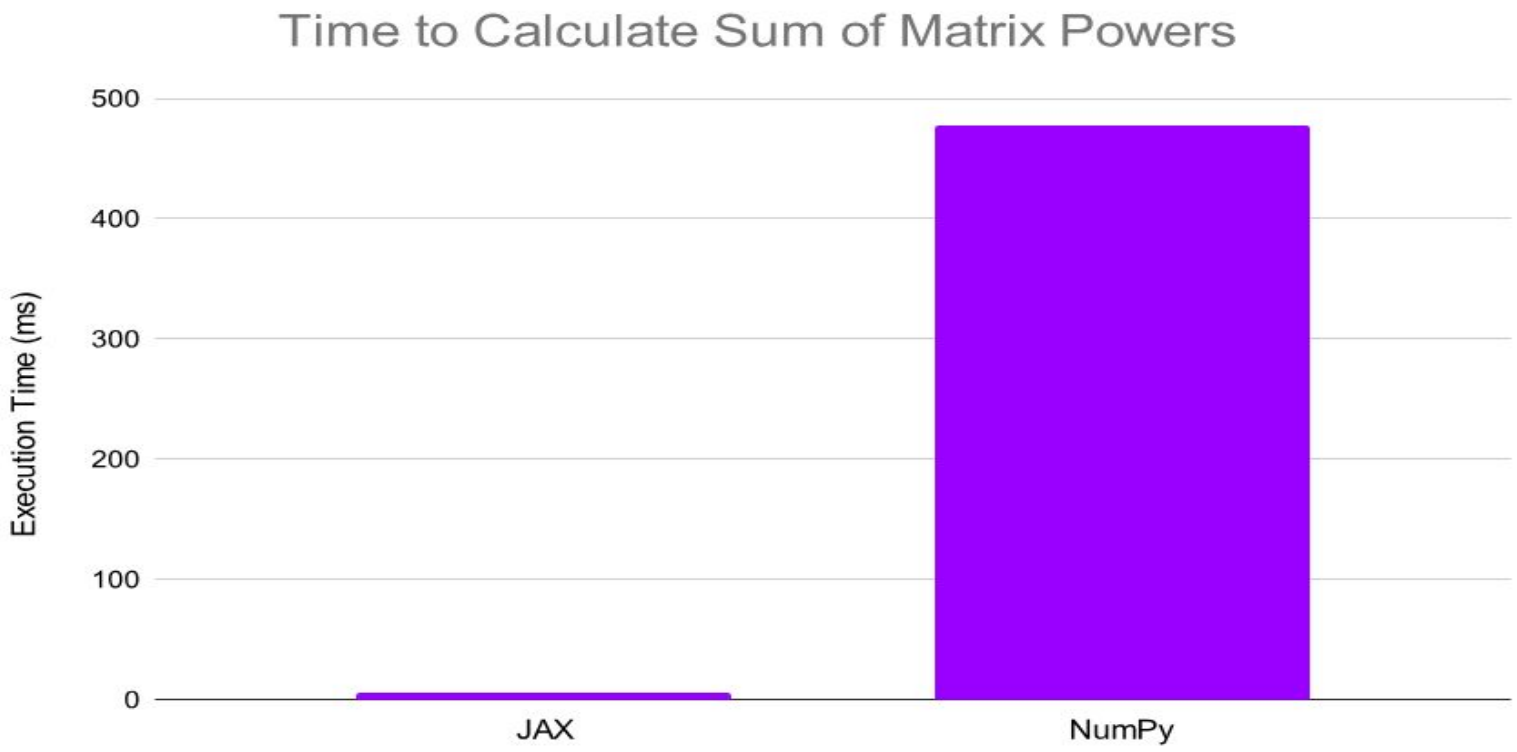


Why Should We Care About JAX?

In short - *speed*. This is the universal aspect of JAX that is relevant for any use case.

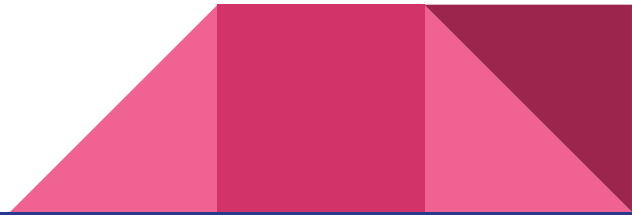


Why Should We Care About JAX?



Reasons why we might want to use JAX?

1. **NumPy on Accelerators** - NumPy is one of the fundamental packages for scientific computing with Python, but it is compatible only with CPU. JAX provides an implementation of NumPy (with a near-identical API) that works on *both* GPU *and* TPU extremely easily. For many users, this *alone* is sufficient to justify the use of JAX.
2. **XLA** - XLA, or Accelerated Linear Algebra, is a whole-program optimizing compiler, designed specifically for linear algebra. JAX is built on XLA, raising the computational-speed ceiling significantly[1].
3. **JIT** - JAX allows you to transform your *own* functions into just-in-time (JIT) compiled versions using XLA[7]. This means that you can increase computation speed by potentially *orders of magnitude* by adding a simple function decorator to your computational functions.



Reasons why we might want to use JAX?

4. **Auto-differentiation** - The JAX documentation refers to JAX as "Autograd and XLA, brought together"[1]. The ability to automatically differentiate is crucial in many areas of scientific computing, and JAX provides several powerful auto-differentiation tools.

5. **Deep Learning** - While not a Deep Learning framework itself, JAX certainly provides a more-than-sufficient foundation for Deep Learning purposes. There are many libraries built on top of JAX that seek to build out Deep Learning capabilities, including Flax, Haiku, and Elegy. JAX's highly efficient computations of Hessians are also relevant for Deep Learning, given that they make higher-order optimization techniques much more feasible.

6. **General Differentiable Programming Paradigm** - While it is certainly possible to use JAX in order to build and train Deep Learning models, it also provides a framework for *general* Differentiable Programming. This means that JAX can exploit *prior knowledge* in a given field, built up through decades of research, by using a model-based Machine Learning approach to solving a problem.