JAX – "Python-Native Machine Learning Research"

Gordian Edenhofer[1,2]

[1]Max Planck Institute for Astrophysics, Garching, DE
[2]Faculty of Physics, LMU, Munich, DE

ERUM-IFT Invited Talk, Monday 20th November, 2023

JAX as Machine Learning Backbone

Core Concepts

Hands-On

Concluding Remarks

JAX as Machine Learning Backbone

Core Concepts

Hands-On

Concluding Remarks

JAX

- Versatile library for expressing and composing numerical computations by Google (DeepMind)
- Think: NumPy + SciPy + Auto-Differentiation
- Compile code for CPU, GPU and TPU (via XLA) but with constrained computations

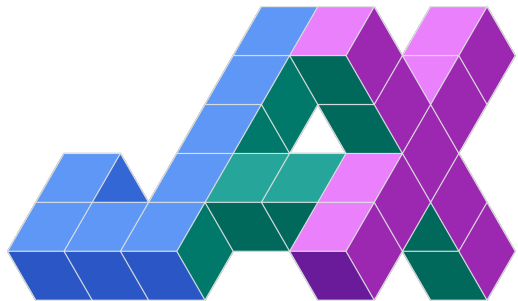

Figure: Logo.[1]

[1] Google, *deepmind/dm-haiku:/README.md*.

As Accelerated NumPy

- Automatic vectorization
- Automatic differentiation
- Just In Time (JIT) compilation
- Parallel evaluation

As Accelerated NumPy

```
import jax
import jax.numpy as jnp

t = jnp.array([1., 2., 3.])

def melu(x):
  return jnp.sum(jnp.exp(x))

g_melu = jax.grad(melu)
%timeit g_melu(t).block_until_ready()

g_melu = jax.jit(g_melu)
g_melu(t)  # Warm up
%timeit g_melu(t).block_until_ready()
```

As Machine Learning Library

- Building blocks for all fields of machine learning
- More versatile than an ordinary machine learning library

[2] https://deepmind.com/blog/article/using-jax-to-accelerate-our-research.

As Machine Learning Library

- Building blocks for all fields of machine learning
- More versatile than an ordinary machine learning library

- Haiku (Deepmind): OOP neural network modules
- Flax (Google): Haiku but with batteries included
- Obtax (Google): optimizer framework
- Jaxopt (Google): optimizer framework
- . . . [2]

[2] https://deepmind.com/blog/article/using-jax-to-accelerate-our-research.

Versus TensorFlow

JAX

- Seamlessly integrates with other python functions[3]
- General purpose numerics interface (i.e. NumPy + SciPy)

TensorFlow

- Closed framework

[3] Google, *Primitives for calling from JAX accelerator code to Python functions on the host.*

JAX as Machine Learning Backbone

Core Concepts

Hands-On

Concluding Remarks

Pure functions

- Function depends exclusively and deterministically on its arguments[4]
  - No mutable `global` variables
  - No mutable references as argument
- Function has not side-effects
  - No mutations
  - No input- or output-streams

[4] Google, *Common Gotchas in JAX*.

In-Place Updates

Updates are impure and thus not permitted

```
import numpy as np
import jax.numpy as jnp

a_np = np.array([1., 2., 3.])
a_np[2:] = 10.  # impure

a_jnp = jnp.array([1., 2., 3.])
a_jnp = a_jnp.at[2:].set(10.)  # pure
```

Control Flow with JIT

- Imposes much tighter constraints than required by e.g. JAX's `grad`
- Computation must be transparent to JAX
  - Use `jax.lax.cond` in favor of `if`
  - Use `jax.lax.while_loop`, `lax.fori_loop`, ... in favor of `while`, `range`, ...
- Output shape and data-type may **only** depend on input shape and data-type

Miscellaneous

- Out-of-bounds indices are clipped by default
- Strict differentiation between lists and arrays
- No dynamic shapes

JAX as Machine Learning Backbone

Core Concepts

Hands-On

Concluding Remarks

Hands-On

JAX as Accelerated NumPy

Enough talking; let's code

in bash

```
pip install "jax[cpu]"
```

in python

```
import jax
import jax.numpy as jnp
import numpy as np
from jax import random

jax.config.update("jax_enable_x64", True)
```

JAX as Accelerated NumPy

iPython

Stax as Neural Networks Framework in JAX

- Split neural network state (i.e. parameters) from network application
- Akin to Haiku[5]

in python

```
import jax
import jax.numpy as jnp
import numpy as np
from jax import random
from jax.example_libraries import optimizers, stax

jax.config.update("jax_enable_x64", True)
```

[5] Hennigan et al., *Haiku: Sonnet for JAX*; Deepmind, *deepmind/dm-haiku:/README.md*.

Stax as Neural Networks Framework in JAX

iPython

JAX as Machine Learning Backbone

Core Concepts

Hands-On

Concluding Remarks

JAX

- Jax is differentiable, vectoriatizing, JIT-compilable framework with first-class GPU and TPU support
- As easy as NumPy/SciPy and as powerful as TensorFlow

Appendix

Appendix

Pytrees

- Nested structure of (by default) lists, tuples, and dicts
- Every tree can be represented by a tuple of the form `(children, metadata)`
- Leaf are usually scalars or JAX tensors

Appendix

Differentiation

- Jacobian-Vector-Product (`jax.jvp`, `jax.linearize`)
- Vector-Jacobian-Product (`jax.vjp`)