Deep Learning Frameworks

Tensorflow, Pytorch, JAX

Argonne
NATIONAL LABORATORY

Deep Learning Frameworks

- Tensorflow, Pytorch and JAX are the core deep learning frameworks supported on ALCF production resources.
- All three frameworks are accessible in python (and a few other languages, for tf/torch) and offer the core elements of:
  - Automatic differentiation;
  - GPU offload and acceleration from python;
  - Library of essential building blocks of machine learning operations;
  - Performant ways to scale codes out to multiple devices
  - An ecosystem of extensions and custom tools to make your life easier;
  - Export your trained models to open source inference engines (ONNX, etc)
  - All are open source; All will be supported on Aurora. Pick the one that makes sense for your problem!

Tensorflow

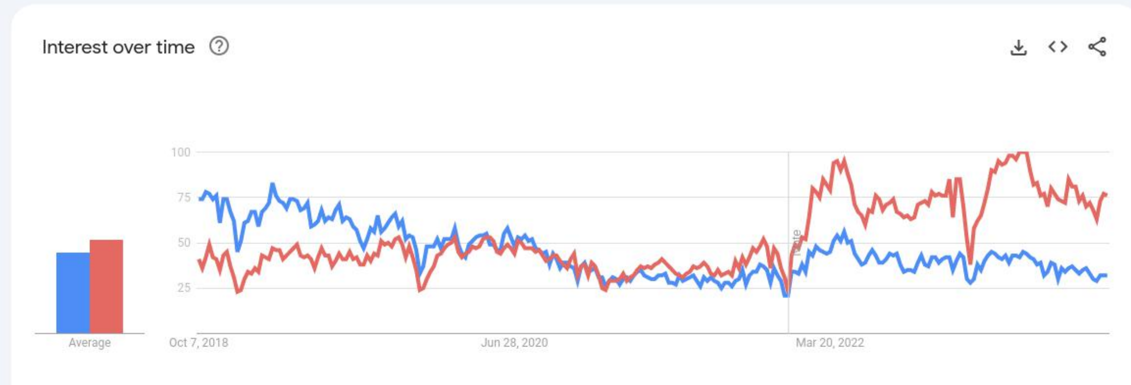
- The oldest of the frameworks shown today, tensorflow is developed by Google.
- Excellent performance on both CPU and GPU
  - Why care about CPU? Large scale inference on CPU-only systems.
- Major version change between v1.X and 2.X - only 2.X is "officially" supported in our installs, focus on 2.X
- Important links:
  - Tensorflow basics - start here for basic syntax, etc
  - Keras - A high level API to make tensorflow even easier. May or may not fit your needs but a great entry point.
  - Mixed Precision - Essential to acheive peak performance on Polaris
  - XLA - Worth a try for models that have fusable operations, especially in reduced precision

Tensorflow at ALCF

- Installed in our conda modules:
  - conda/2023-10-02 has TF v2.13.0 in python 3.10.12
- Performance considerations:
  1. Use mixed precision if possible!
     1. A100 gpus have TensorCore accelerators for reduced precision (TF32, FP16)
     2. Easiest way to enable is via keras mixed_precision.Policy("mixed_float16")
  2. Use `tf.function` syntax on your high level functions to enable graph tracing and operation merging.
     1. It's as simple as putting `@tf.function` as decorators on your functions
     2. It has a number of "gotchas" if you need dynamic models - often its useful to graph-compile subsets of your code instead of the entire thing!
  3. Use XLA (or at least test w/ XLA) to check for performance boosts from XLA compilation.
     1. This is in addition to tf.function! XLA will only compile code that is traced in a graph.
     2. Can enable with either:
        1. just one environment variable change: TF_XLA_FLAGS=--tf_xla_auto_jit=2
        2. Arguments to tf.function(jit_compile=True)
     3. Downsides: profiling XLA compiled code is more challenging due to operator fusion and renaming.

Pytorch

- The other main DL framework, developed by Facebook and more “numpy-like” than tensorflow.
- Until v2.0, didn’t support compilation like Tensorflow - this is now changed but your mileage may vary while this becomes more widespread.
- Pytorch is more “pythonic” than tensorflow, and features dynamic operations instead of graph computation as the main mode.
- Pytorch has a larger ecosystem of extensions and has been growing faster than tensorflow in recent years
- Out of the box performance is on par with Tensorflow.
- For models with many small operations, compilation is worth exploring.



	A100 (full GPU)		
	FP32	TF32	FP16
CosmicTagger (TF)	9.5	11.8	12.4
CosmicTagger (TF + XLA)	22.0	29.1	38.9
CosmicTagger (Torch)	14.7	15.5	15.5

Pytorch at ALCF

- Installed in our conda modules:
  - conda/2023-10-02 has torch v2.0.1 in python 3.10.12
- Performance and other considerations:
  - Pytorch 2.0 is backwards compatible with v1.X
  - Reduced precision is easiest with automatic reduced precision
  - Graph compilation is technically available but your experience on your model may be unique.
    - We welcome reports of success/failure and performance changes with graph compilation - it's expected to be a useful feature for performance moving forward!
  - If you require low latency or integration into another C++ code, pytorch has a native C++ frontend "libtorch"
  - Torch 2.0 claims to implement high-efficiency functional transforms inspired by JAX, to enable hessians, jacobians, JVPs.
  - Pytorch ecosystem has gaussian processes, graph networks, geospatial data enablements, and many more.

JAX

- JAX is the new framework, arising from a combination of autograd (automatic differentiation for numpy) and XLA enablement of numpy (aka, numpy operations on the GPU).
- JAX is **purely functional** - no sideeffects allowed in your traced functions.
- JAX utility driven in large part by functional transformations:
  - jit is functional tracing which can enable massive performance gains.
  - vmap/pmap enable automatic vectorization (and not just over batch size - any axis!) Write a function over tensor sizes that make sense and let JAX / XLA help you scale it up.
  - grad computation and other derivatives are likewise functional transforms - grad(f) returns a function that computes the gradient of f. It does not explicitly return the gradient!
    - This means that you can apply vmap or other transformations to gradient tranformations too, enabling efficient differentiation in ways that are challenging to do in TF / Torch

- JAX documentation has the best autodifferentiation documentation of all frameworks - even if you want to use tf/torch, JAX might have the best explanation of how the operations work and what modes get the best performance.

JAX at ALCF

- Installed in our conda modules:
  - conda/2023-10-04 has JAX v0.4.17 in python 3.10.12

- Jax can also leverage reduced precision but it is not as clear-cut and simple: changing the datatype of a function's input can cause retracting, and much of it is manual.
- JAX also has some "sharp bits": non-mutable arrays, deterministic random number generators, etc.
  - Check out the guide to the sharp bits for more info.
- There are two straightforward ways to scale out:
  - officially supported pmap
  - mpi-enabled mpi4jax
  - They can also be used in concert with each other: pmap on a node, mpi4jax for multi-node communication

Porting Existing Projects

- You're encouraged to bring your existing code to LCF systems!
  - Often, the frameworks we install are the lowest barrier to entry: we've tested to ensure everything works, built many things from source to ensure compatibility with the latest drivers and mpi.
  - You can **extend** our modules with virtual environments.
    - There is no possibility to install every package that everyone wants in our conda environments.
    - So, load the module and create a python virtual environment to install your additional packages
    - python -m venv --system-site-packages /path/to/desired/virtualenv/folder
  - You can also use the `--user` option when doing `pip install` or building a package from source.
    - Please note that most ALCF production systems share a home directory, and its very easy to install something on Polaris that breaks your workloads on ThetaGPU (for example).
    - This really isn't the recommended path for software installation!
- If you want to install everything yourself - go for it!
  - conda is a good way to get a python install and package manager
  - Any issues or challenges should be reported to support@alcf.anl.gov for assistance.
  - Be careful with "all in one" installs - the pypi packages that deliver cuda components can easily conflict between two packages in a way that will break your application. Reach out if you need guidance.

Create a new Frameworks App

- Which framework should you choose?
  - Do you need CPU-based inference after training on another system?
    - Tensorflow, or exportable to ONNX, is useful here
  - Do you want ability for fast prototyping of models?
    - Pytorch is generally considered the “easiest” framework.
  - Do you need unusual operations, differentiation steps (hessians, or higher order grads)?
    - JAX is likely your best bet
  - Do you have large-scale requirements? High efficiency scale out, or the need for deepspeed or other model-parallelization?
    - Pytorch DDP and Deepspeed are well optimized on polaris.
  - Integrate into a C++ framework?
    - Libtorch is a good fit
  - Other unique constraints? Feel free to brainstorm with ALCF staff at this workshop!

Where to get help?

- support@alcf.anl.gov is your best stop to definitely get help. Frameworks-based questions will make it to the relevant experts in the datascience team.
- Ask us at this workshop:
  - Multiple members of the datascience team here and happy to help.

Questions?