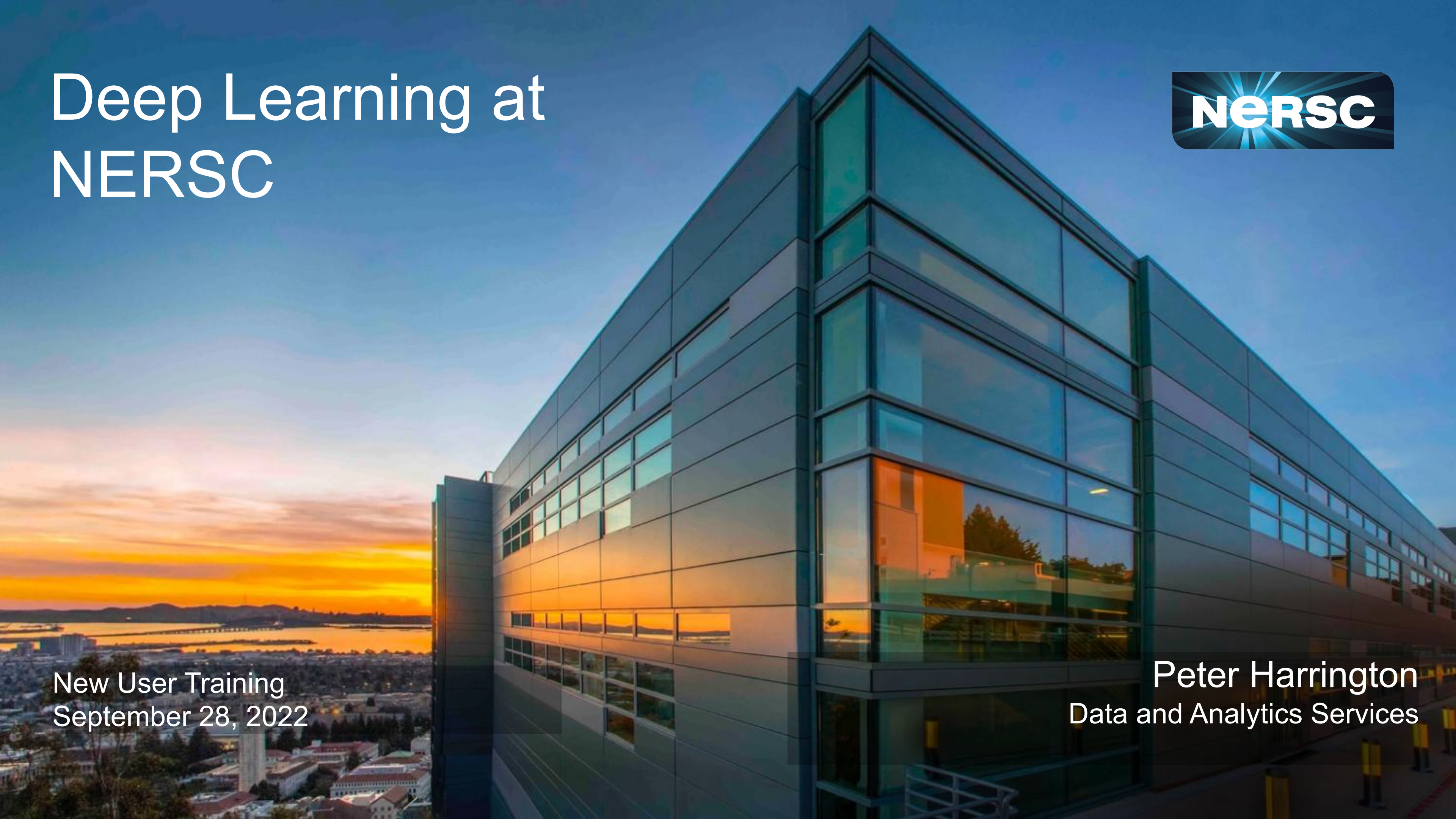
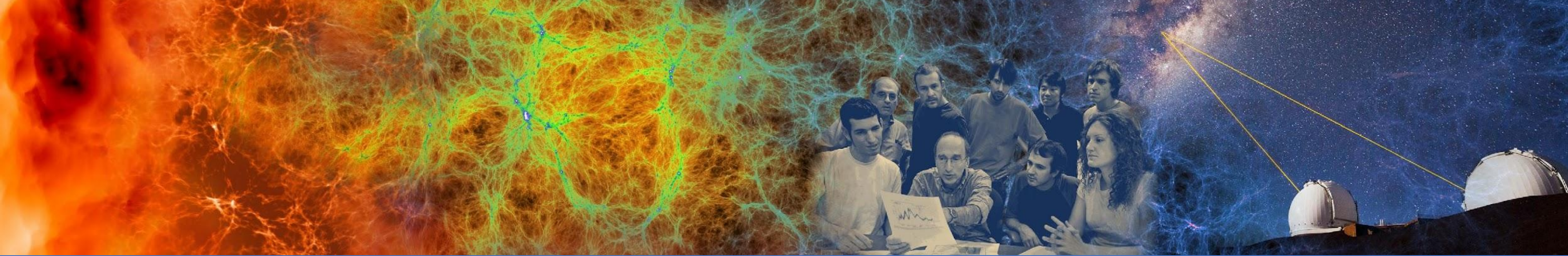


Deep Learning at NERSC

NERSC

Peter Harrington
Data and Analytics Services

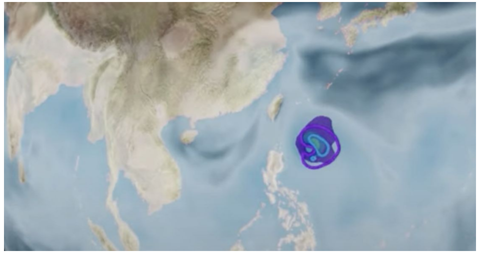
New User Training
September 28, 2022



Outline

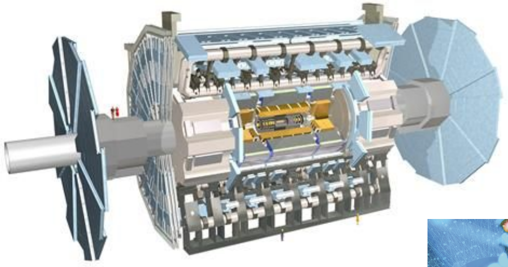
- Deep learning for science @ NERSC
- Deep learning stack on Perlmutter
- How to use DL tools and frameworks on Perlmutter

Deep Learning is transforming science



It can enhance various scientific workflows

- Analysis of large, complex datasets
- Accelerating expensive simulations



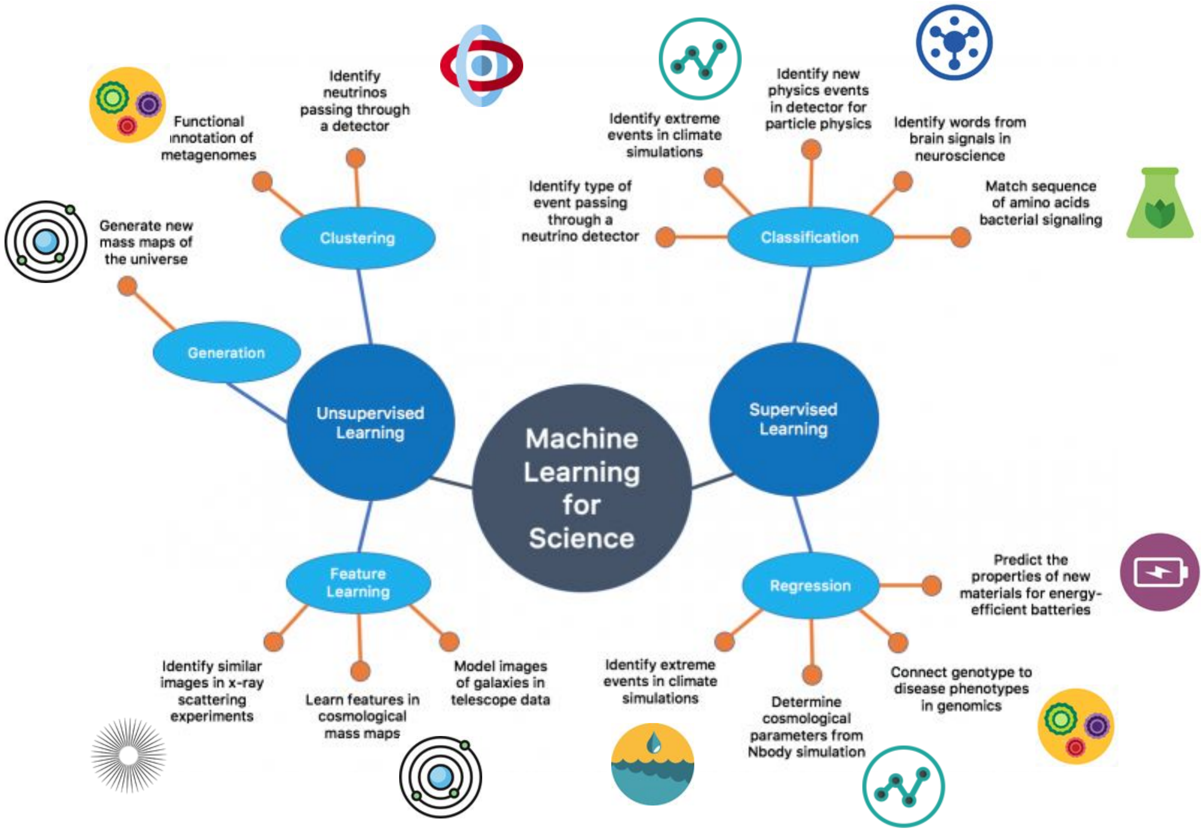
Adoption is on the rise in the science communities

- Rapid growth in ML+science conferences
- Recognition of AI achievements:
  2018 Turing Award; 2018, 2020 Gordon Bell prizes
- HPC centers awarding allocations for AI,
  optimizing next-gen systems for AI

The DOE is investing heavily in AI for science

- Funding calls from ASCR (and other funding agencies)
- Popular, enthusiastic AI4Science town hall series, [300 page report]

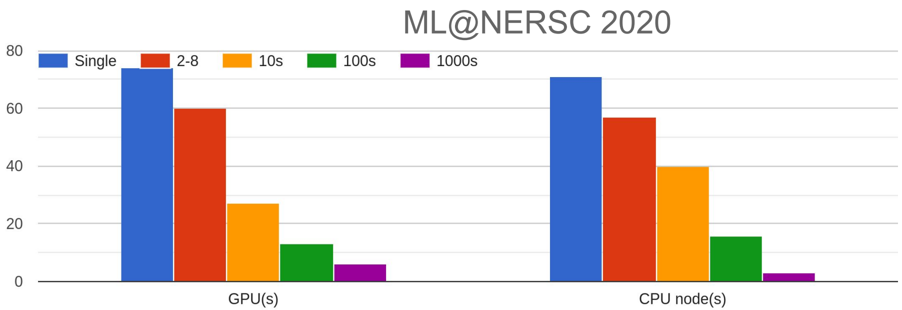
Scientific ML: endless possibilities!



More complex tasks, bigger models, more compute

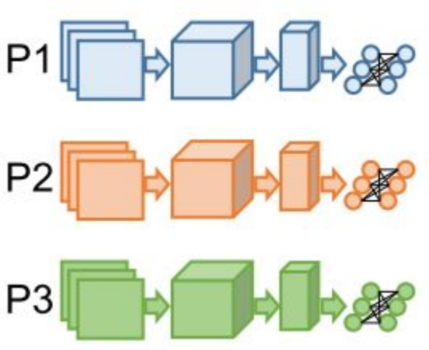
Credit: NVIDIA

At what scale do you train your models? (include current and future plans).



Models get bigger and more compute intensive as they tackle more complex tasks

Deep Learning parallelization strategies

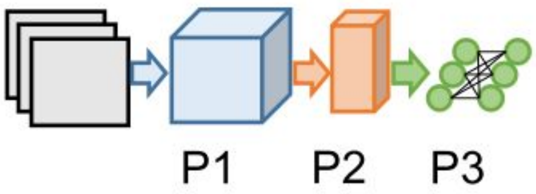


Data Parallelism

Distribute input samples.

Model Parallelism

Distribute network structure (layers).

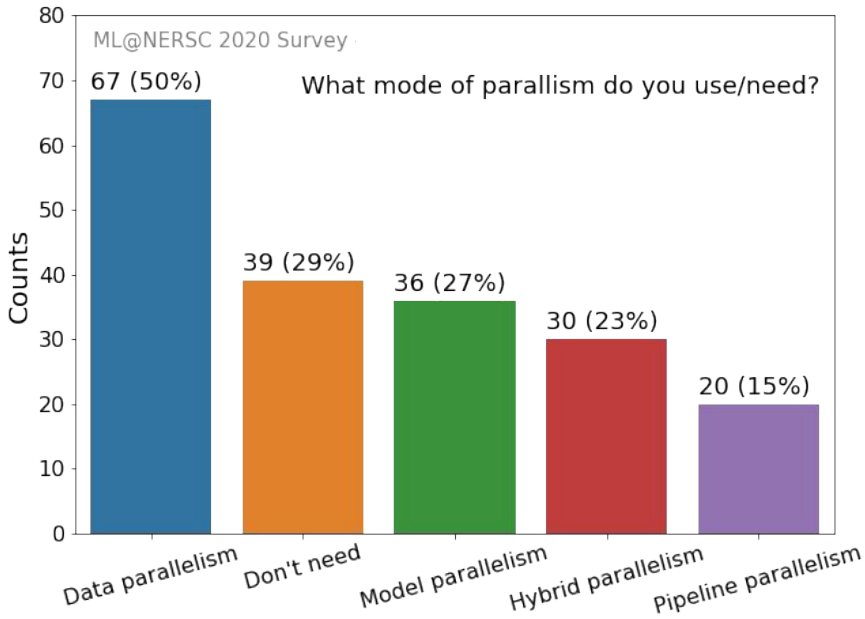


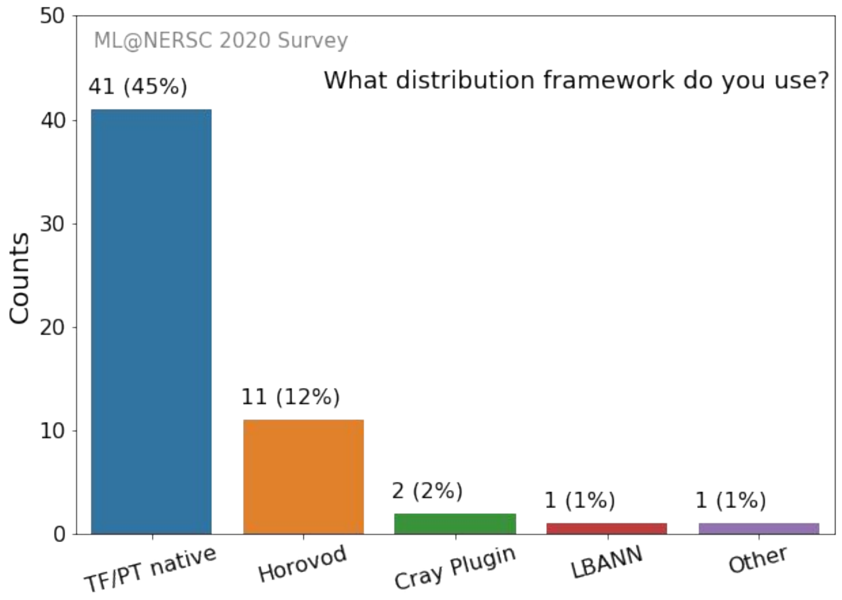
Layer Pipelining

Partition by layer.

Fig. credit: Ben-Nun and Hoefler arXiv:1802.09941

Deep Learning parallelization strategies





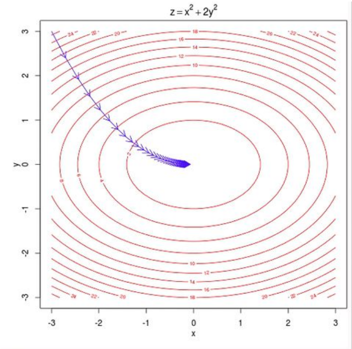
Data parallelism is the most common strategy in practice, especially for inter-node scaling.

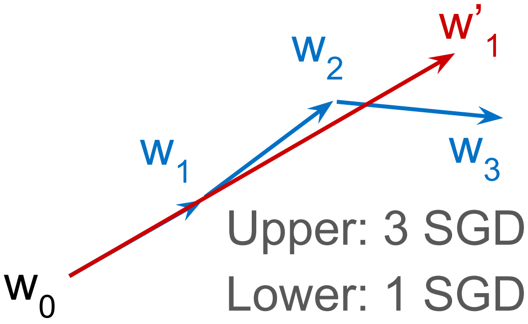
TensorFlow and PyTorch support data and intra-node pipeline parallelism natively. Horovod is the leading non-native distribution framework. All support MPI and/or NCCL backends.

Data-parallel training considerations

Weak scaling: converge faster by taking fewer, bigger, faster steps

- i.e., more GPUs, larger batch sizes, larger learning rates





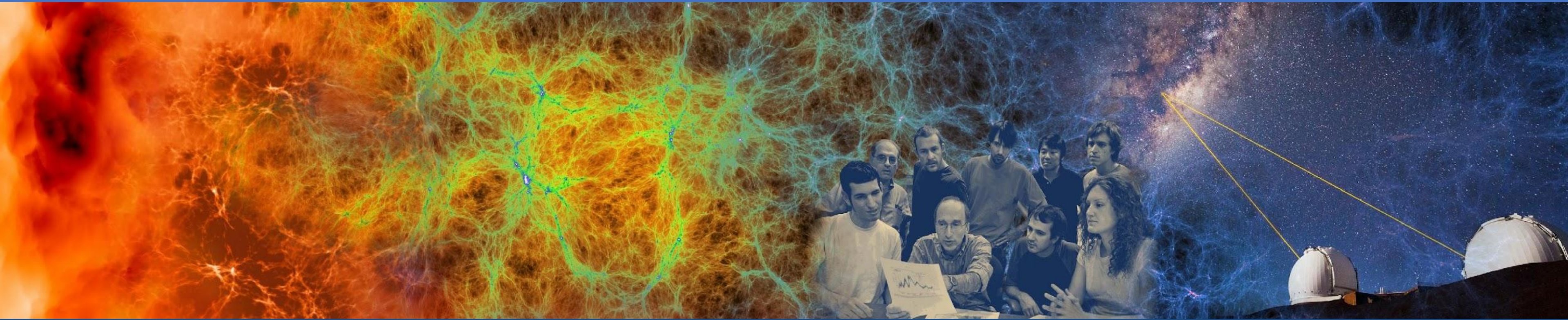
Upper: 3 SGD steps w. learning-rate = η

Lower: 1 SGD step w. learning-rate = 3 * η

Caveat: for stability & convergence, requires tuning

- Warm-up+scale learning rate, adaptive optimizers, etc
- See our SC21 “Deep Learning at Scale” tutorial for more tips

Deep Learning on Perlmutter: Software stack and best practices



NERSC

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

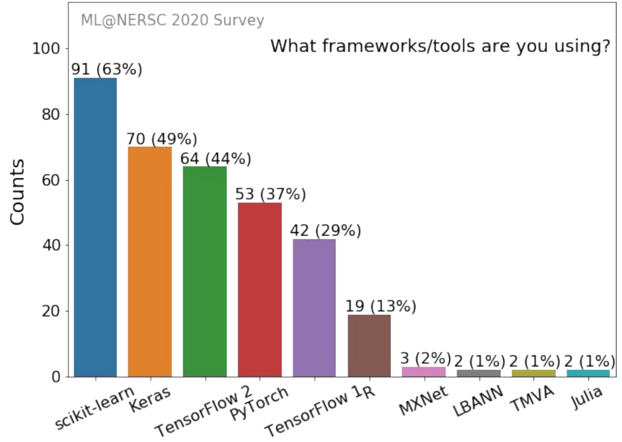
Perlmutter deep learning software stack overview

General strategy:

- Provide functional, performant installations of the most popular frameworks and libraries
- Enable flexibility for users to customize and deploy their own solutions

Frameworks:

TensorFlow Keras PyTorch



Distributed training libraries:

- Horovod
- PyTorch distributed

Productive tools and services:

- Jupyter, Shifter

https://docs.nersc.gov/machinelearning/

How to use the Perlmutter DL software stack

We have modules you can load which contain python and DL libraries:

```
module load tensorflow/2.9.0
module load pytorch/1.11.0
```

Check which software versions are available with:

```
module spider pytorch
```

You can install your own packages on top to customize:

```
pip install --user MY-PACKAGE
```

Or, clone a conda environment from our modules:

```
conda create -n my-env --clone /path/to/module/installation
```

Or, create custom conda environments from scratch:

```
conda create -n my-env MY-PACKAGES
```

More on how to customize your setup can be found in the docs (TensorFlow, PyTorch).

Containerized DL: using Shifter on Perlmutter

NERSC currently supports [containers with Perlmutter via Shifter]()

- Easy, performant: our top500 entry used a container!

To see images currently available:

```
shifterimg images | grep pytorch
```

To pull desired docker images onto Perlmutter:

```
shifterimg pull <dockerhub_image_tag>
```

To use interactively:

```
shifter --module gpu --image=nvcr.io/nvidia/pytorch:22.05-py3
```

Use Slurm image shifter options for best performance in batch jobs:

```
#SBATCH --image=nersc/pytorch:ngc-22.05_v1
srun shifter python my_python_script.py
```

SHIFTER

Best Practices for DL + Shifter on Perlmutter

NVIDIA provides containers optimized for deep learning on GPUs with

- Pytorch or TensorFlow+Horovod
- Optimized drivers, CUDA, NCCL, cuDNN, etc
- Many different versions available

We also provide images based on NVIDIA's, which have a few useful extras

You can also build your own custom containers (easy to build on top of NVIDIA's)

Notes

- Customization: from inside the container, do `pip install --user MY-PACKAGE` (make sure to set `$PYTHONUSERBASE` to a custom path for the desired container)

- NVIDIA NGC containers use OpenMPI, which requires specific options if you require MPI. Instructions: https://docs.nersc.gov/development/shifter/how-to-use/#shifter-mpich-module

Guidelines - TensorFlow distributed training

TensorFlow at NERSC docs:
https://docs.nersc.gov/analytics/machinelearning/tensorflow/

For distributed training, we recommend using Horovod

- Easy to use and launch with SLURM
- Can use MPI and NCCL as appropriate
- Horovod examples: https://github.com/horovod/horovod/tree/master/examples

TensorFlow has some nice built-in profiling capabilities

- TF profiler in TF 2: https://www.tensorflow.org/guide/profiler

Guidelines - PyTorch distributed training

PyTorch at NERSC docs:
https://docs.nersc.gov/analytics/machinelearning/pytorch/

PyTorch

For distributed training, use PyTorch's DistributedDataParallel

- Simple model wrapper, native to Pytorch
- Works on CPU and GPU
- Highly optimized for distributed GPU training
- Docs:
  https://pytorch.org/tutorials/intermediate/ddp_tutorial.html

Distributed backends

- On Perlmutter, use the NCCL backend for optimized GPU communication

General guidelines for deep learning at NERSC

NERSC documentation: https://docs.nersc.gov/analytics/machinelearning/overview/

Use our provided modules/containers if appropriate

- They have the recommended builds and libraries tested for functionality and performance
- We can track usage which informs our software support strategy

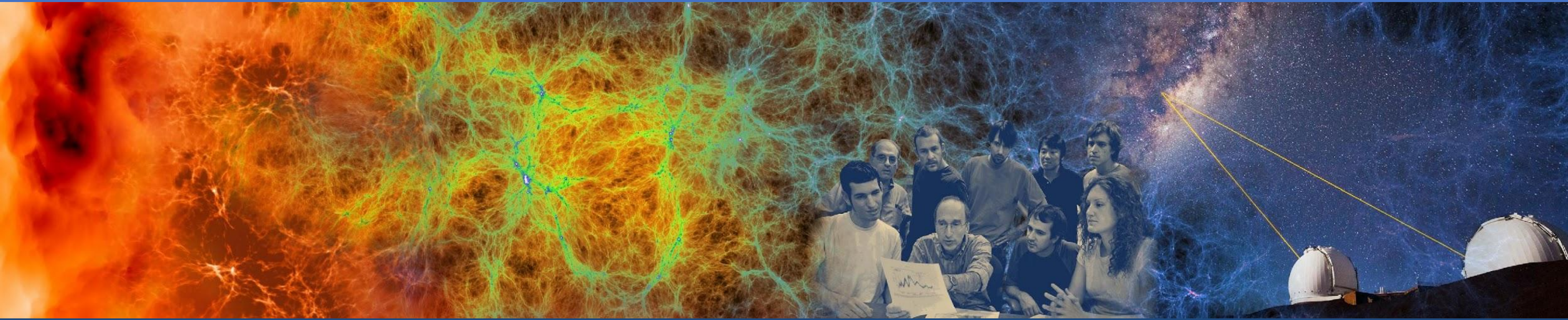
For developing and testing your ML workflows

- Use interactive QOS or Jupyter for on-demand compute resources
- Visualize your models and results with TensorBoard or Weights & Biases

For performance tuning

- Check cpu/gpu utilization to indicate bottlenecks (e.g. with top, nvidia-smi)
- Data pipeline is the most common source of bottlenecks
  - Use framework-recommended APIs/formats for data loading
  - Use multi-threaded data loaders and stage data if possible
- Profile your code, e.g. with Nvidia Nsight Systems or TensorBoard Profiler

Deep Learning on Perlmutter: Workflow tools



NERSC

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

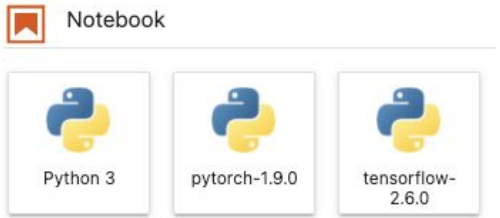
Jupyter for deep learning

JupyterHub service provides a rich, interactive notebook ecosystem on Cori

- Very popular service with hundreds of users
- A favorite way for users to develop ML code

Users can run their deep learning workloads

- on dedicated Perlmutter GPU nodes
- using our pre-installed DL software kernels
- using their own custom kernels

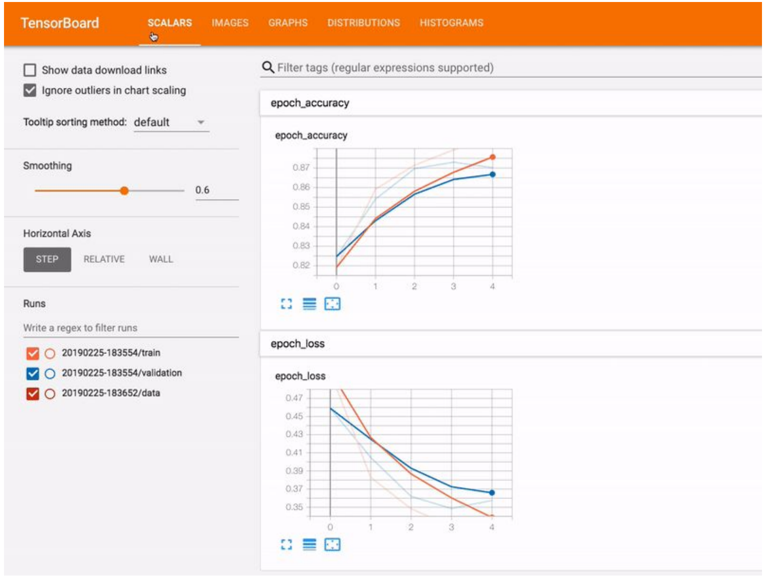


	Shared CPU Node	Shared GPU Node	Exclusive GPU Node	Exclusive Large Memory Node	Configurable GPU	Configurable DGX
Perlmutter	start		start		start	
Cori	start	start		start	start	
Resources	Use a node shared with other users' notebooks but outside the batch queues.		Use your own node within a job allocation using defaults.		Use multiple compute nodes with s	
Use Cases	Visualization and analytics that are not memory intensive and can run on just a few cores.		Visualization, analytics, machine learning that is compute or memory intensive but can be done on a single node.		Multi-node analytics jobs, jobs in reservations, custom project charging, and more.	

TensorBoard at NERSC

TensorBoard is the most popular tool for visualizing and monitoring DL experiments, widely adopted by TensorFlow and PyTorch communities.
We recommend running TensorBoard in Jupyter using nersc-tensorboard helper module.

```
import nersc_tensorboard_helper
%load_ext tensorboard
%tensorboard --logdir YOURLOGDIR --port 0
```



then get an address to your TensorBoard GUI:

```
nersc_tensorboard_helper.tb_address()
```

Hyper-parameter optimization (HPO) solutions

Model selection/tuning are critical for getting the most out of deep learning

- Many methods and libraries exist for tuning your model hyper-parameters
- Usually very computationally expensive because you need to train many models
  => Good for large HPC resources

Users can use whatever tools work best for them

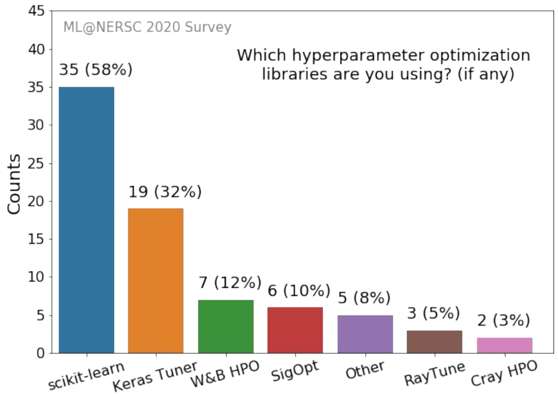
- Ask us for help if needed!

RAY tune

Weights & Biases

SIGOPT

OPTUNA

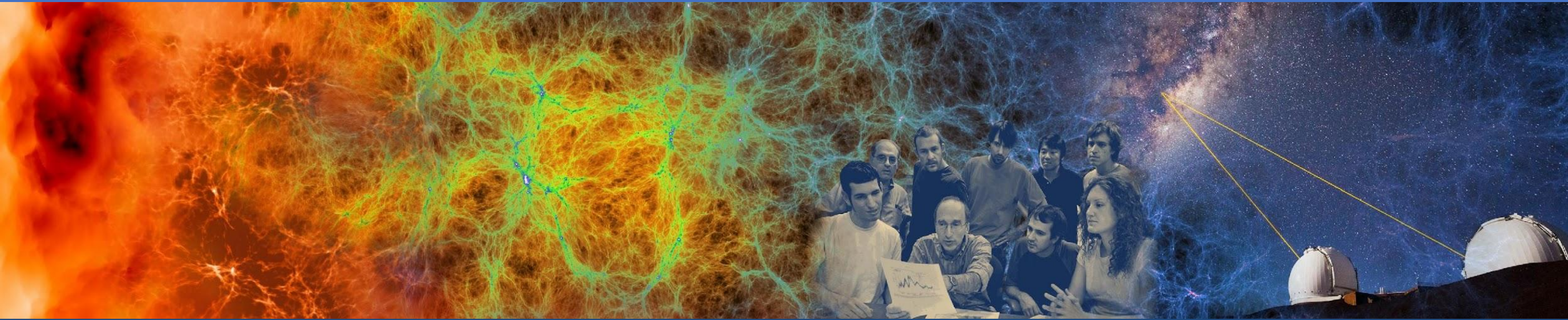


NERSC

BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY | Office of Science

Outreach & additional resources



Training events

The Deep Learning for Science School at Berkeley Lab (https://dl4sci-school.lbl.gov/)

- Comprehensive program with lectures, demos, hands-on sessions, posters
- You can view the full 2019 material (videos, slides, code) online: https://sites.google.com/lbl.gov/dl4sci2019
- 2020 webinar series – recorded talks: https://dl4sci-school.lbl.gov/agenda

The Deep Learning at Scale Tutorial

- Jointly organized with NVIDIA (& Cray in previous years)
- Presented at SC18-21, ECP Annual 2019, ISC19
- Detailed lectures + hands-on material:
  - Distributed training, profiling & optimization on Perlmutter
  - Basis for today's hands-on exercises
- See the full SC21 material here

NERSC Data Seminar Series:
https://github.com/NERSC/data-seminars

Conclusions

Deep learning for science is here and growing

- Powerful capabilities
- Enthusiastic community
- Increasing HPC workloads

Perlmutter has a productive, performant software stack for deep learning

- Optimized frameworks and solutions for small to large scale DL workloads
- Support for productive workflows (Jupyter, HPO)

Join the NERSC Users Slack

Take the ML@NERSC Survey

Thank You and Welcome to NERSC!