Deep Learning Stack at NERSC

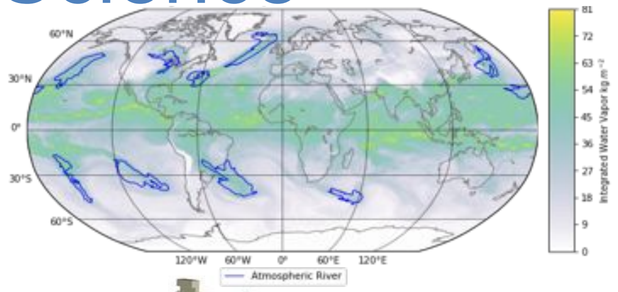
New User Training
June 16, 2020

Mustafa Mustafa
Data And Analytics Group

Outline

- Deep Learning for science
- Deep learning stack at NERSC
- How to use DL tools and frameworks at NERSC
- Resources to communities and research activities

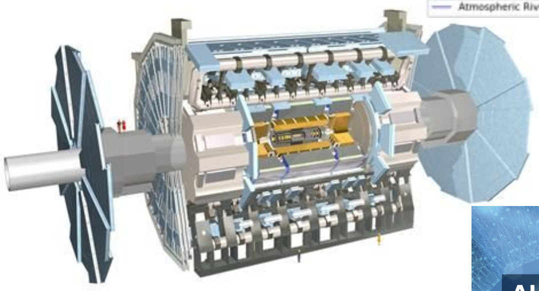
Deep Learning is Transforming Science



It can enhance various scientific workflows

- Analysis of large datasets
- Accelerating expensive simulations
- Real time control and design of experiments

Adoption is on the rise in the science communities



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

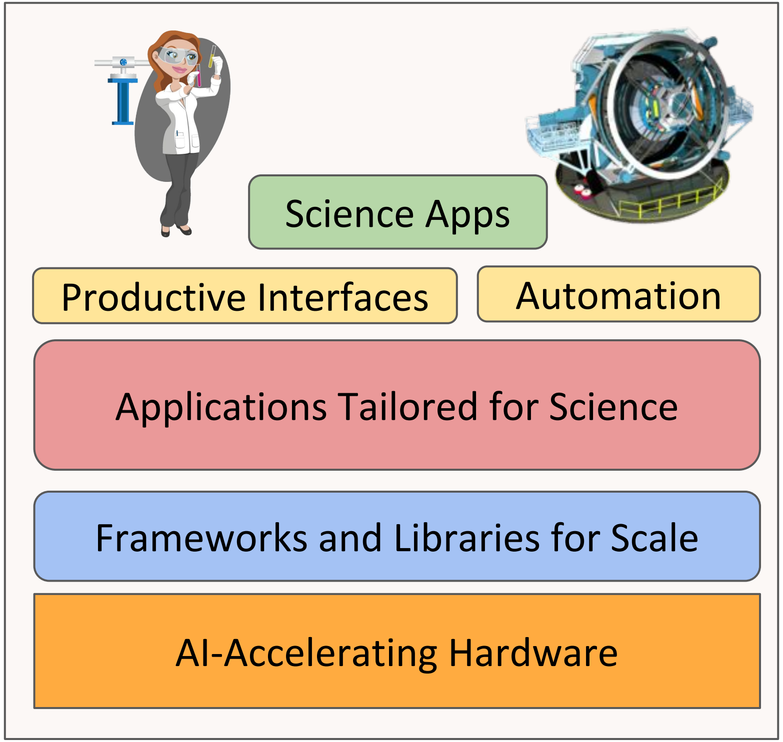
The DOE is investing heavily in AI for science

- Funding calls from ASCR (and other funding agencies), ECP ExaLearn
- Popular, enthusiastic AI4Science town hall series, *300 page report*
- Anticipated ECP-like program on AI4Science

NERSC Provides a Platform for Scientific Deep Learning at Scale

For Cori, Perlmutter and Beyond

- ***Optimized DL software for hardware and scale, working closely with***
    - hardware vendors (Cray, Intel, NVidia)
    - and industry partners (Google, FaceBook)
- ***Productive platforms***
    - Interactivity and Jupyter notebooks
    - Allow model exploration and reuse
- ***Training, Consulting, and Collaborations***
    - Ensure state-of-the-art deep-learning applications for science



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

NERSC 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:

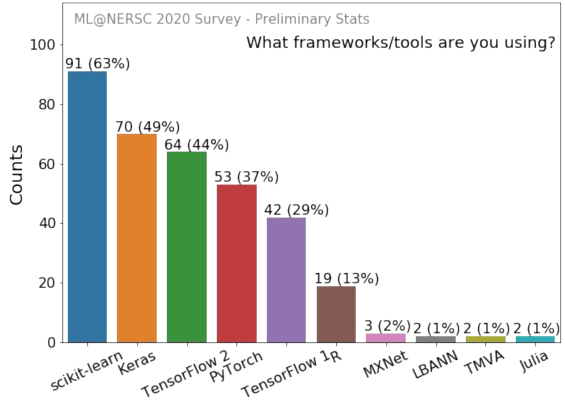
TensorFlow K Keras PyTorch

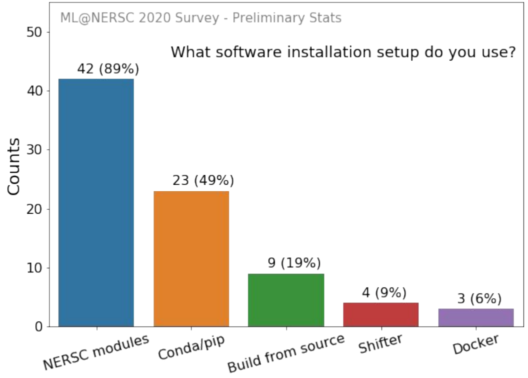
Distributed training libraries:

- Horovod
- PyTorch distributed
- Cray Plugin

Productive tools and services:

- Jupyter, Shifter





How to Use NERSC DL Software Stack

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

```
module load tensorflow/intel-2.1.0-py37
module load pytorch/v1.5.0
```

Check which software versions are available with:

```
module avail tensorflow
```

You can install your own packages on top to customize:

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

Or you can create your 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).

We also have pre-installed Jupyter kernels.

Software Stack in Shifter (Cori GPU and Perlmutter)

We are working on providing TensorFlow/PyTorch Shifter images based on NVidia's GPU Cloud Containers (NGC) which are optimized for best performance on GPUs.

To use interactively:

```
shifter  --volume="/dev/infiniband:/sys/class/infiniband_verbs" \
    --module none --image=nersc/pytorch:1.5.0_v0
```

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

```
#SBATCH --image=nersc/pytorch:1.5.0_v0
#SBATCH --volume="/dev/infiniband:/sys/class/infiniband_verbs"
srun shifter python my_python_script.py
```

Images currently available: `pytorch:1.5.0_v0`, `tensorflow:ngc-20.03-tf1-v0`, `tensorflow:ngc-20.03-tf1-v0`

We also provide Jupyter kernels based on these images.

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

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
  - Try the Burst Buffer for data-intensive applications
- Profile your code with cProfile, NVIDIA DLProf (containers only), framework-specific tools

Guidelines - TensorFlow Distributed Training

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

For distributed training, we recommend to use Uber's 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
- Keras TensorBoardCallback in TF 1

Guidelines - PyTorch Distributed Training

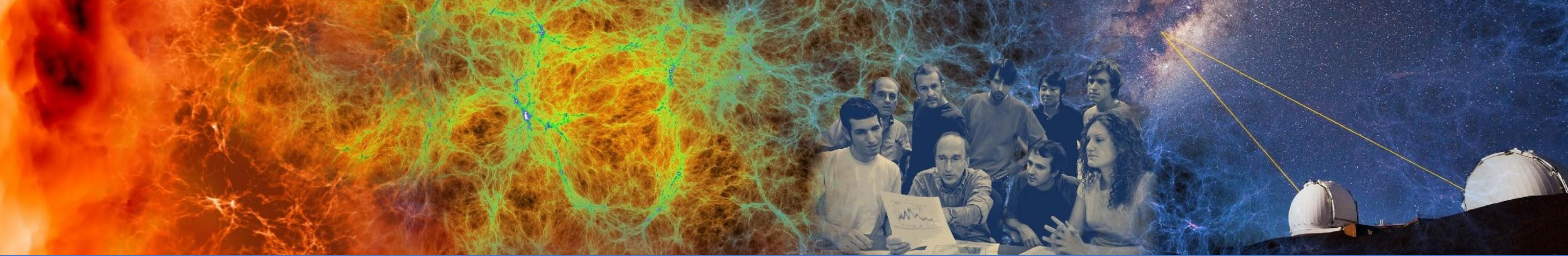
PyTorch at NERSC docs:
https://docs.nersc.gov/analytics/machinelearning/pytorch/

For distributed training, use PyTorch's DistributedDataParallel model wrapper

- Very easy to use
- Works on CPU and GPU
- Highly optimized for distributed GPU training
- Docs: https://pytorch.org/tutorials/intermediate/ddp_tutorial.html

Distributed backends

- On Cori CPU, use the MPI backend
- On Cori GPU, use the NCCL backend (example setup)



Workflow Tools

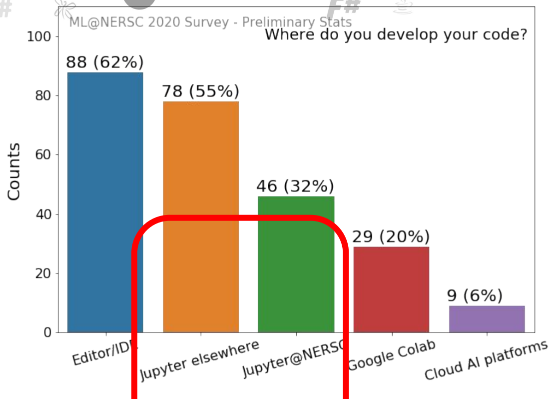
Jupyter for Deep Learning

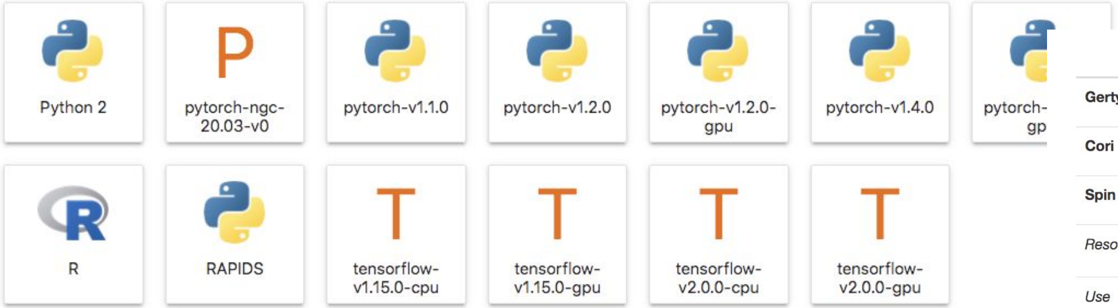
JupyterHub service provides a rich, interactive notebook ecosystem on Cori

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

Users can run their deep learning workloads

- on Cori CPU and Cori GPU
- using our pre-installed DL software kernels
- [using their own custom kernels]()





	Shared CPU Node	Shared GPU Node	Exclusive CPU Node
Gerty	start		start
Cori	stop server	start	start
Spin	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 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.

TensorBoard at NERSC

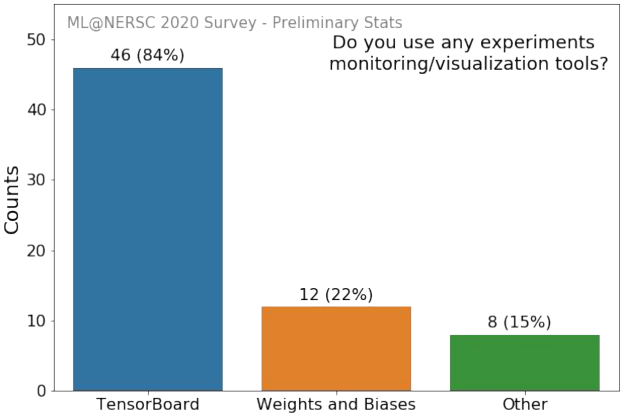
TensorBoard is the most popular tool for visualizing and monitoring DL experiments, widely adopted by TensorFlow and PyTorch communities.

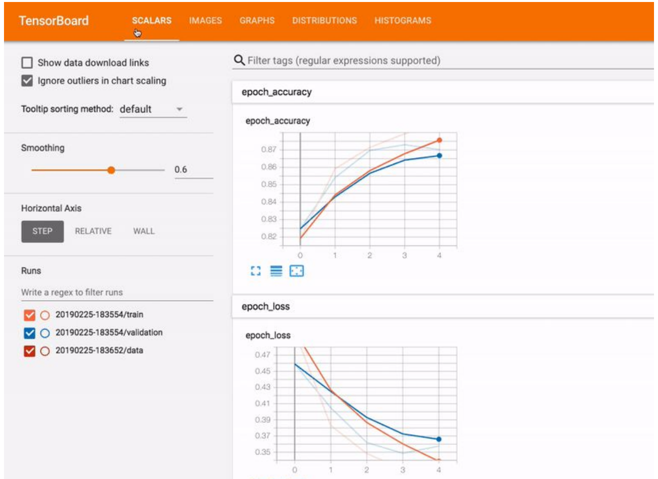
We recommend running TensorBoard in Jupyter using nersc-tensorboard helper module.

```
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 Solutions

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

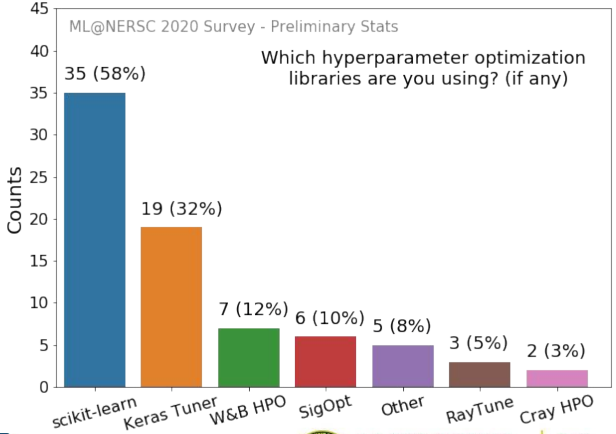
- Many methods and libraries exist for tuning your model hyper-parameters
- Usually very computationally expensive because you need to train many models
  => Good for large HPC resources

We have prioritized support for tools that map well onto our systems

- Cray HPO
- Ray Tune

Users can use whatever tools work best for them

- Supports
- And ask us for help if needed!



Cray HPO

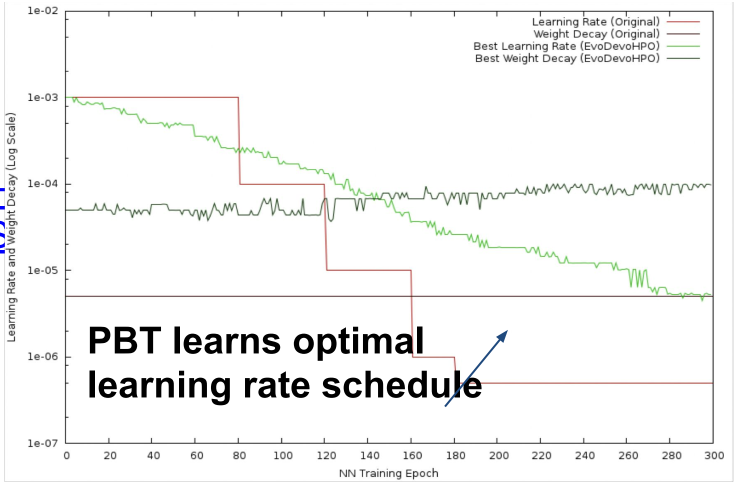
a Hewlett Packard Enterprise company

Cray's Hyper-parameter Optimization tool is built for HPC systems

- Seamlessly integrates with SLURM to manage allocations, launch training tasks
- Features popular HPO algorithms: random/grid search, genetic search, population based training (PBT)
- User defined training script steered with command line arguments; allows for any framework, distributed training, etc.

See Ben Albrecht's lecture and hands-on material from the 2019 DL4Sci school at Berkeley Lab

- https://drive.google.com/open?id=1KlF9P2meZgc7BJqMz7nlH
- https://www.youtube.com/watch?v=u_vKXRiDXe8&list=PL20S ex=18&t=0s



Cray HPO

CRAY
a Hewlett Packard Enterprise company

NERSC documentation: https://docs.nersc.gov/analytics/machinelearning/hpo/#cray-hpo

Official Cray documentation: https://cray.github.io/crayai/hpo/hpo.html

Example Jupyter notebook for running at NERSC: https://github.com/sparticlesteve/cori-intml-examples/blob/master/CrayHPO_rpv.ipynb

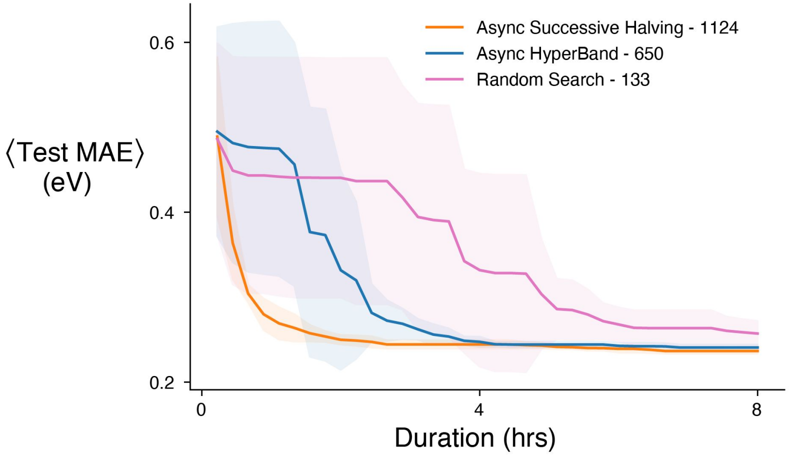
Genetic search example

```python
#!/usr/bin/env python3

from crayai import hpo

evaluator = hpo.Evaluator('python3 source/train.py')

params = hpo.Params([['--lr', 0.001, (1e-5, 0.1)],
                     ['--optimizer', 'Adam', ['Adam', 'Adadelta', 'Nadam']]])

optimizer = hpo.GeneticOptimizer(evaluator,
                                 generations=10,
                                 num_demes=4,
                                 pop_size=4,
                                 mutation_rate=0.05,
                                 crossover_rate=0.33)

optimizer.optimize(params)

print(optimizer.best_fom)
print(optimizer.best_params)
```

Ray Tune

Tune is an open-source Python library for experiment execution and hyperparameter tuning at any scale.

- Supports any ML framework
- Implements state of the art HPO strategies
- Natively integrates with optimization libraries (HyperOpt, BayesianOpt, and Facebook Ax)
- Integrates well with Slurm
- **Handles trials micro scheduling on multi-gpu-node resources** (no GPU binding boilerplate needed)

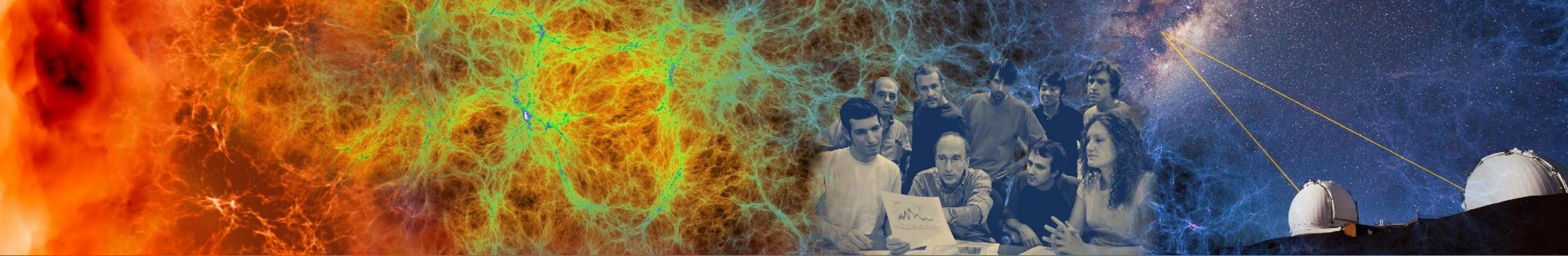


Example of Multi-node HPO using RayTune used by NESAP team to optimize Graph Neural Network models for catalysis applications (Brandon Wood et al.)

See NERSC slurm-ray-cluster for slurm scripts and how to run at NERSC.

```
/project/.../ray_tune/slurm-ray-cluster$ vim submit-ray-cluster.sbatch
/project/.../ray_tune/slurm-ray-cluster$
```

NERSC



Applications, 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 in-person event canceled (COVID-19);
  planning summer webinar series instead

The Deep Learning at Scale Tutorial

- Jointly organized with Cray (and now NVIDIA)
- Presented at SC18, ECP Annual 2019, ISC19, SC19
- Lectures + hands-on material for distributed training on Cori
- See the full SC19 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

NERSC 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

Please fill our ML@NERSC User Survey

Thank You and Welcome to NERSC!