**Interactive HPC Deep Learning with Jupyter**

**Motivation**
- Deep learning (DL) can enable scientific discovery
- Training of complex networks can take days
- Architecture design and parameter choice is an iterative process aided by human intuition, brute-force scans and automated optimization
- Batch HPC submission means many slow iteration cycles
- Most DL frameworks are python-based: iterating within notebook infrastructure

**Example Notebooks**
- Distributed training
  - Use-case: CNN for particle physics (LHC) classification
  - IPyParallel and Keras + Horovod-MPI
  - MPI code in a notebook
  - Scales well - no overhead from notebook infrastructure

**Distributed Hyperparameter Optimization (HPO)**
- Run HPO tasks with load-balanced IPyParallel scheduler

**HPO Widgets**
- Live plots of model output
- Interactive table with status
- Buttons and form for stopping/restarting models
- Built-in Jupyter widget library
- IPyParallel engines publish data that is monitored via background threads

**More advanced HPO**
- Genetic Algorithms and Population-based Training (PBT)
- Create training schedule with customized hyperparameters across epochs

**Conclusions**
- Popular interactive Jupyter development approach
- Complex Deep learning models and HPO experiments + Distributed HPC resources = clear win for science
- Developing software, infrastructure and policy for this on Cori at NERSC
- Demonstrated notebook-driven examples for
  - Distributed deep learning with negligible impact on scaling performance
  - Distributed HPO with widgets for real-time feedback and interaction

**NERSC Jupyter Architecture**
- Extended Jupyter infrastructure to run kernels on Cori Supercomputer at NERSC
- Allocate nodes on Cori interactive queue and start iPyParallel or Dask cluster
- Developed %ipcluster magic - setup in notebook
- Compute nodes do not have external address
- Implications for network config / policy decisions
- Distributed training communication is via MPI (e.g. Horovod)

**Kale Jupyter Extension for HPC**
- Enables human-in-the-loop computing for HPC with Jupyter
- Control (start/stop) remote tasks from Notebooks
- Resource monitoring (Task/Node)
- RESTful API

**References**
1. https://github.com/uber/horovod
4. https://github.com/bloomberg/bqplot
5. https://github.com/quantopian/qgrid