Scaling ML follow up

Paolo Calafiura & Walter Hopkins
With input from many

Proposed Milestones

1. Identify target ML models in collaboration with experiments
2. Port, train, and run at least two target models on two different HPC systems
3. Compare two data parallel training solutions for at least one target model
4. Compare two hyperparameter optimization tools on at least one target model
5. Setting up a prototype Inference as-a-service platform on at least one DOE HPC system

Task list sign up sheet

Possible target ML models

Categories:

- **Simulation**
  - FastCaloGAN -> a lot of human intervention to make the GANs converge. LBANN has multi-generator, multi-discriminator framework that is only possible with scaling.
  - Cosmological simulations, DES adversarial domain adaptation
- **Reconstruction**
  - Flavor tagging, tracking, DUNE reconstruction
- **Analysis**
  - Simulation-based inference, LSST image processing
- **Resource constrained (FPGA/ASIC) model**
  - **HPO** is more important vs offline models
  - Size of model vs performance
  - Quantization slows down training
  - Smart pixels (6-layer CNN) takes 3 days (tracking related)

Next step: request repos and input data

Future scaling of foundation models

- Generic particle flow
- Calorimeter simulation
- Optimal experimental design

Need to canvas the community to find foundation models

Port, train, and run at least two target models on two different HPC systems

- Find existing open source solutions
  - Documentation and/or simplification of interface
    - Examples: weights and biases (or open source alternative)
- Start with simple model (ParticleNet, a classification GNN*) and scale to one of the larger models
- HPC choice: two different architectures (e.g., ALCF vs NERSC vs OLCF)

*Special use case: GNNs (data and model parallel are intertwined)

Compare two data parallel training solutions for at least one target model

Possible solutions: KubeFlow, LBANN, DeepSpeed, DDP, FSDP (latter two are PyTorch based)

Figure of merits:

- User experience
- Time to convergence
- Resource utilization across CPU, GPU, network, RAM
- Stability of the solution

Compare two hyperparameter optimization tools on at least one target model

Possible HPO tools: DeepHyper, OmniOpt, HypPO, and others TBD

Figure of merits:

- User experience
- Time to convergence
- Resource utilization across CPU, GPU, network, RAM
- Support for frameworks

Setting up a prototype Inference as-a-service platform on at least one DOE HPC system

Candidate framework: SONIC

- TorchServe, TF.Serve only work with PyTorch and TF
- How to make use of HPC resources? **Infrastructure practicalities**
  - Current framework of asking for servers as latency increases doesn't work with HPC allocations
  - Security system
  - Integrated Research Infrastructure (IRI) could be a solution

Action: get in connect SONIC and NERSC experts