# DUNE and FNAL AI Infrastructure

Tingjun Yang, Ken Herner
With inputs from Andrew Mogan, Shekhar Mishra, Mike Wang

FNAL AI Infrastructure Workshop
6 April 2023

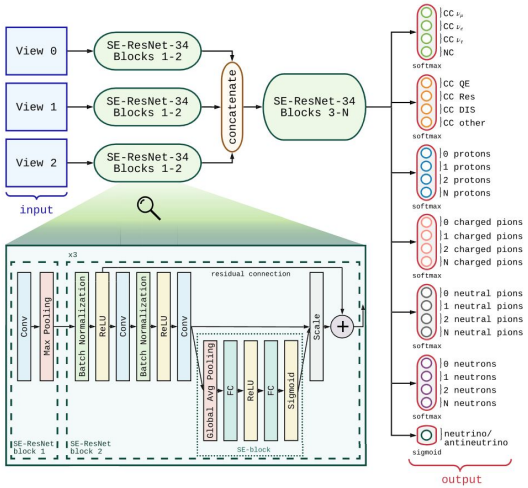
# AI in DUNE

- AI is being used at several fronts in DUNE
- Simulation and reconstruction algorithms
  - CVN neutrino ID for DUNE far detector
  - CNN track/shower/Michel ID for ProtoDUNE
  - 3D reconstruction for near detector
  - Fast simulation of detector responses
- Real time applications
  - Physics Inspired Neural Nets (PINNs).
  - AI for event triggering

# AI Algorithms for FD/ProtoDUNE

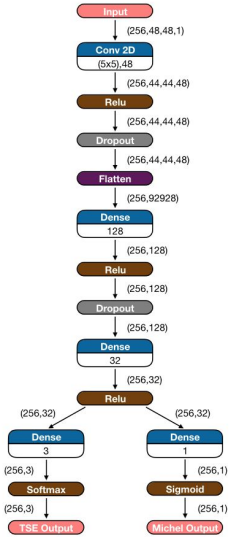
**Neutrino interaction classification with a convolutional neural network in the DUNE far detector**

B. Abi et al. (DUNE Collaboration)
Phys. Rev. D 102, 092003



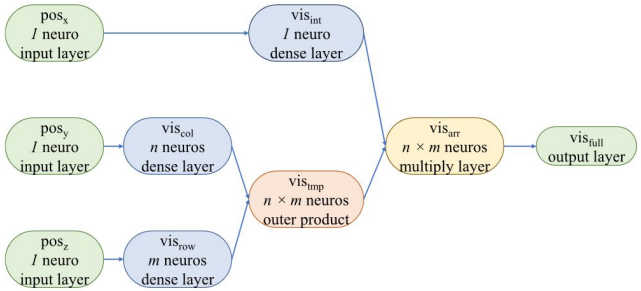

**Separation of track- and shower-like energy deposits in ProtoDUNE-SP using a convolutional neural network**

A. Abed Abud et al. (DUNE Collaboration)
*Eur. Phys. J. C* **82**, 903 (2022).



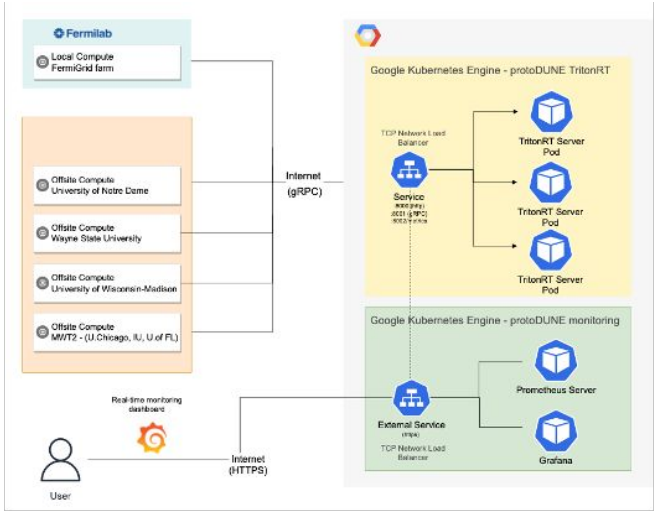

**Photon detection probability prediction using one-dimensional generative neural network**

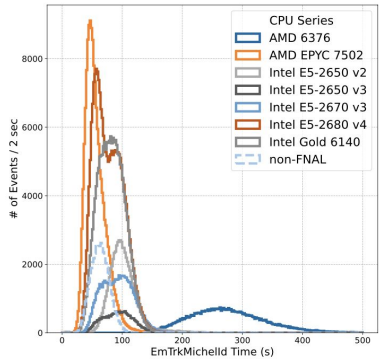
Wei Mu et al
2022 Mach. Learn.: Sci. Technol. 3 015033

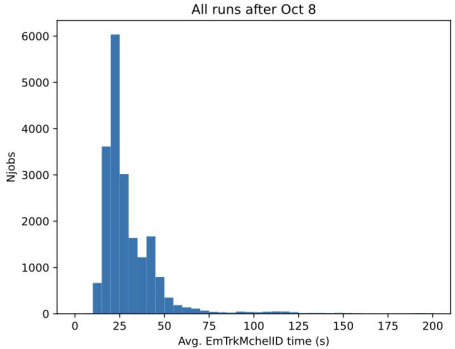


# AI Algorithms for FD/ProtoDUNE

- All the algorithms on the last page are trained using tensorflow.
- They are all integrated into larsoft using the tensorflow C++ API.
- Normally the ML inferences are done using CPUs
  - This is typically slow especially for the ProtoDUNE reconstruction.
- We have experimented acceleration of ML inferences using GPU as a Service (GPUaaS)
  - Proof of concept done in 2021 - Front. Big Data 3 (2021) 604083.
  - Large scale ProtoDUNE production done in 2022 using GPUs on google cloud - arXiv:2301.04633.



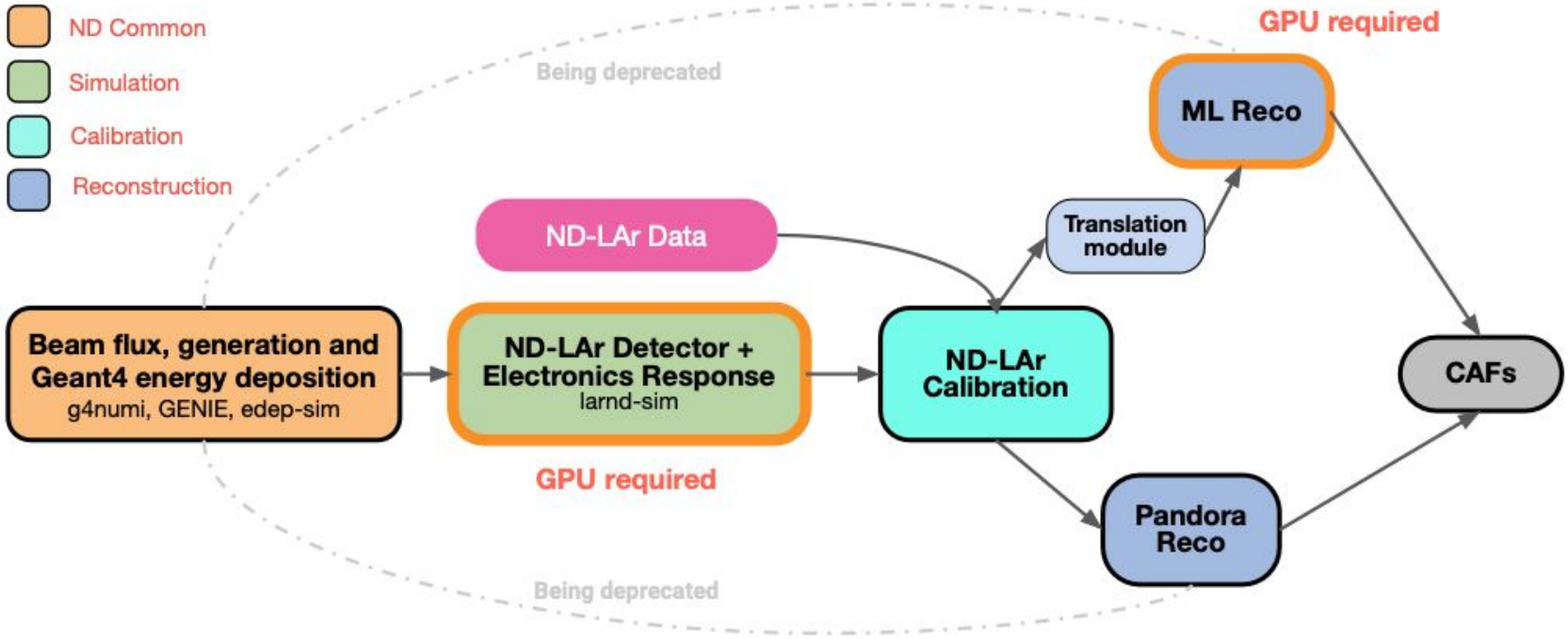
# ProtoDUNE: GPUaaS

- Large-scale tests showed overall event speedup of a factor of ~2 (including network latency to cloud servers) against CPUs bought in 2019
- GPUaaS setup allows one to run local server as well; abstracts away TF interface
- Recent tests on Perlmutter (GPU queue worker node) suggest a speedup of roughly 9 wrt CPU-only algo on same machine
  - Needs to be confirmed at scale with multiple jobs per worker node
- Longer term, some kind of edge service at larger compute centers may be optimal
- Many tradeoffs to consider on what the optimum throughput would look like, but an interesting problem!

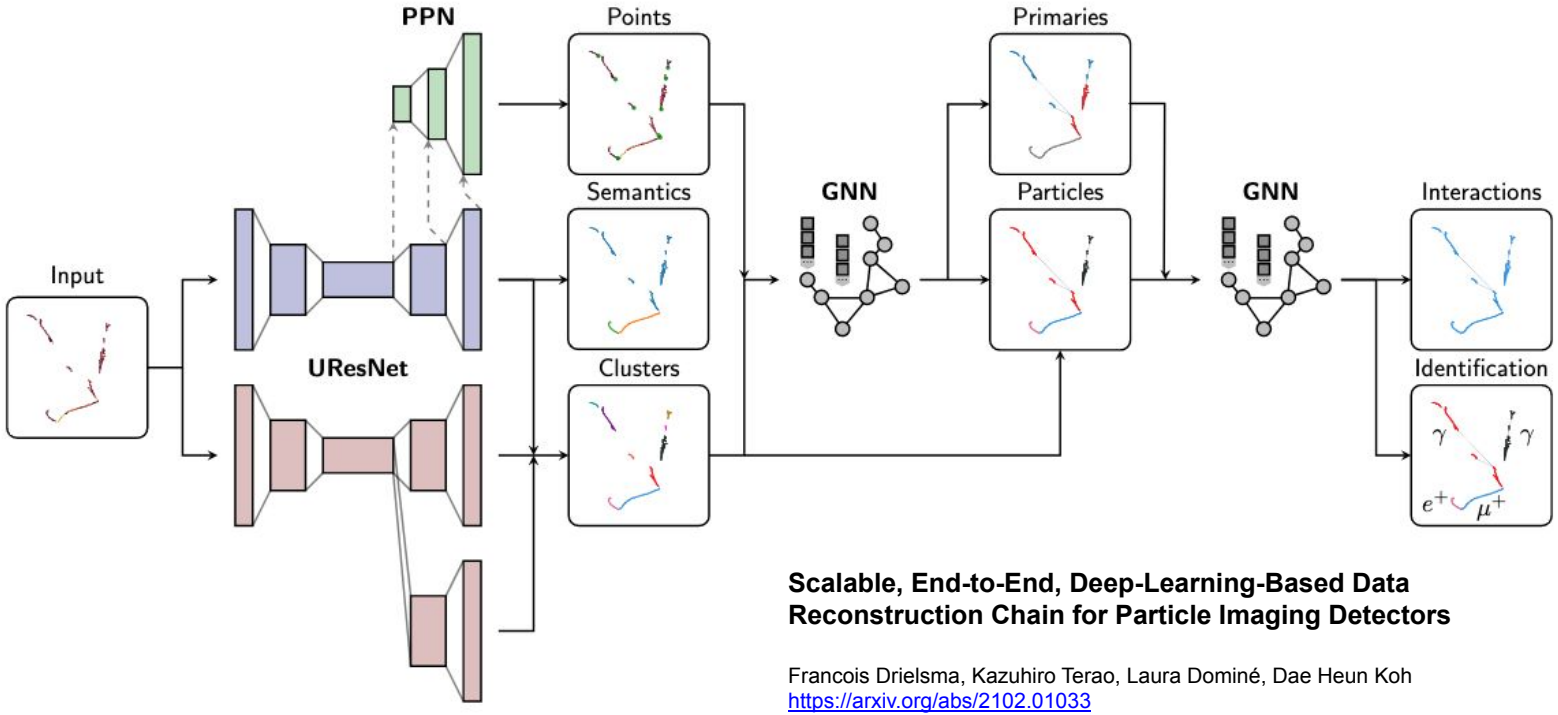
| TimeTracker printout (sec) | Min | Avg | Max | Median | RMS | nEvts |
|---|---|---|---|---|---|---|
| Full event | 20.0613 | 33.6036 | 56.4237 | 32.497 | 7.2699 | 25 |
| source:RootInput(read) | 0.000368222 | 0.0011237 | 0.0163442 | 0.000488585 | 0.00310762 | 25 |
| decode:nhitsfilter:NumberOfHitsFilter | 0.0117825 | 0.0329805 | 0.170938 | 0.0180349 | 0.0415881 | 25 |
| decode:emtrkmichelid:EmTrackMichelId | 18.3042 | 30.4672 | 40.0209 | 30.1073 | 5.39004 | 25 |
| decode:pandoracali:CalibrationdEdXPDSP | 0.0409597 | 0.672156 | 15.1086 | 0.06069 | 2.94706 | 25 |
| decode:pandoracalipid:Chi2ParticleID | 0.0021641 | 0.00302217 | 0.00393549 | 0.00295689 | 0.000442241 | 25 |
| decode:pandora2cali:CalibrationdEdXPDSP | 0.043217 | 0.0904453 | 0.265679 | 0.0774664 | 0.0410401 | 25 |
| decode:pandora2calipid:Chi2ParticleID | 0.00479743 | 0.00799624 | 0.0111908 | 0.0079871 | 0.00161512 | 25 |
| [art]:TriggerResults:TriggerResultInserter | 1.4017e-05 | 1.72345e-05 | 7.243e-05 | 1.4898e-05 | 1.12845e-05 | 25 |
| end_path:out1:RootOutput | 3.426e-06 | 3.90876e-06 | 6.302e-06 | 3.767e-06 | 5.58755e-07 | 25 |
| end_path:out1:RootOutput(write) | 1.4788 | 2.32779 | 3.53506 | 2.19592 | 0.449198 | 25 |

| TimeTracker printout (sec) | Min | Avg | Max | Median | RMS | nEvts |
|---|---|---|---|---|---|---|
| Full event | 3.43546 | 5.95233 | 30.888 | 4.89546 | 5.14123 | 25 |
| source:RootInput(read) | 0.000378281 | 0.00136893 | 0.022163 | 0.000514524 | 0.004245 | 25 |
| decode:nhitsfilter:NumberOfHitsFilter | 0.0104422 | 0.0206682 | 0.0995407 | 0.0171788 | 0.0164156 | 25 |
| decode:emtrkmichelid:EmTrackMichelIdTl | 1.87257 | 3.40395 | 23.7092 | 2.55544 | 4.15671 | 25 |
| decode:pandoracali:CalibrationdEdXPDSP | 0.0416773 | 0.235691 | 4.42326 | 0.0600129 | 0.854845 | 25 |
| decode:pandoracalipid:Chi2ParticleID | 0.00209497 | 0.00304089 | 0.00403937 | 0.00298882 | 0.000492726 | 25 |
| decode:pandora2cali:CalibrationdEdXPDSP | 0.0422489 | 0.0865292 | 0.250596 | 0.0762102 | 0.0392132 | 25 |
| decode:pandora2calipid:Chi2ParticleID | 0.00472792 | 0.00804703 | 0.0112821 | 0.0080594 | 0.00170492 | 25 |
| [art]:TriggerResults:TriggerResultInserter | 1.4028e-05 | 1.70954e-05 | 6.6248e-05 | 1.4929e-05 | 1.00478e-05 | 25 |
| end_path:out1:RootOutput | 3.347e-06 | 4.07268e-06 | 6.181e-06 | 4.037e-06 | 4.93593e-07 | 25 |
| end_path:out1:RootOutput(write) | 1.44199 | 2.19209 | 2.91108 | 2.19387 | 0.397284 | 25 |

# Near Detector LAr Workflows



# ND-LAr ML Reco Architecture




**Scalable, End-to-End, Deep-Learning-Based Data Reconstruction Chain for Particle Imaging Detectors**

Francois Drielsma, Kazuhiro Terao, Laura Dominé, Dae Heun Koh
https://arxiv.org/abs/2102.01033

# 2x2 Prototype

- ND prototype plans to use ML as part of reconstruction step of the sim chain
- Eventually want to generate 10E22 POT through full chain; expect that to take O(100k) GPU node hours at NERSC
- Work underway to understand how to integrate workflow into other setups in the future
- Expect initial dataset this spring to be generated at NERSC

# Comments from Andrew Mogan (CSU)

Andrew pointed out there are some difficulties in running the ND-LAr ML reconstruction chain at Fermilab

- Lack of readily available GPU resources
  - He was told by many people the wait times on the Wilson Cluster are untenable since there are so few nodes.
- Stubborn dependencies that seem to not like SL7
  - There was an attempt to build a Docker container that can house the necessary ML reco dependencies and run at Fermilab
  - He could not get Minkowski Engine to work on a non-Ubuntu system
- Currently ND-LAr ML people use SLAC's SDF and NERSC for their production need. They do not seem to be eager to get their chain to work at Fermilab because of the difficulties mentioned.

# AI on the edge: real time applications

Aside from production and offline analyses, there is an ongoing effort at the lab to explore the use of AI for online and trigger applications in the LArTPC-based neutrino experiments:

- ***"Designing efficient edge AI with physics phenomena"***:
  - ~$3M project to explore physics-inspired neural nets (PINNs) to design more efficient and robust models for AI applications on the "Edge" in CMS, DUNE, and accelerator physics
  - DUNE component focusing on efficient AI for identifying low-energy LArTPC interactions in real-time with applications for calibration, supernova neutrino detection, solar neutrinos, and other low-energy neutrino physics
  - For inference, models will be implemented on embedded devices like FPGAs for low power and low latencies
  - Proof-of-concept will be demonstrated in the real-world environment of the ICEBERG test facility at Fermilab

# AI on the edge: real time applications

Other ongoing AI on the edge applications for neutrinos:

- ***In-storage computing for trigger applications LDRD project***:
  - Exploring the use of AI and computational storage devices to address big-data issues facing experiments like DUNE
  - Implementing AI on hardware embedded in storage devices like SmartSSDs to process buffered supernova neutrino burst data *in situ*

- ***In-network computing for trigger/DAQ applications:***
  - Exploring the use of AI and smart programmable network switches to offload trigger and DAQ tasks normally done on DAQ nodes into the network
  - AI inference will be performed on programmable ASICs and FPGAs found on these smart network switches

# Summary

- Several DUNE workflows already rely on AI methods, especially in reconstruction steps
- Ease of access and data transfer appear to be primary drivers of where such workflows are currently running

# BACKUP

# A word on future Columnar Analysis

- Our analyses naturally lends themselves to this sort of thing
- EAF seems like a natural place to do them
- To date DUNE hasn't yet seen a strong push in this direction; nearly all ProtoDUNE analysis has been HEP-traditional C++/ROOT macros and the like
  - Lately, hasn't been such a large need since datasets aren't yet as large as they will be by the end of the decade– traditional processing is "fast enough"
  - With 150 APAs per module, that thinking may change