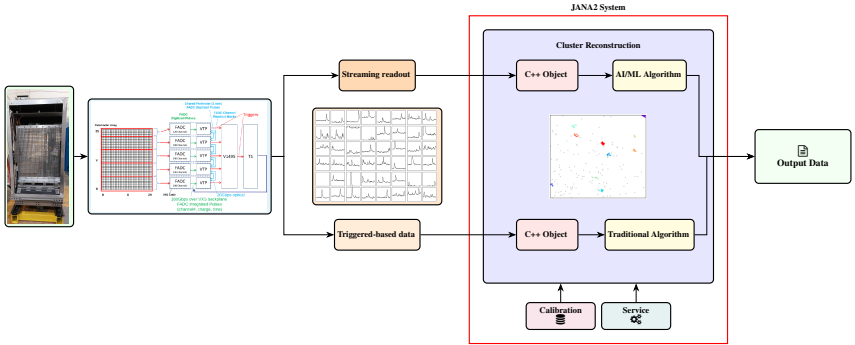
NPS Prototype beam test with streaming readout and AI/ML-based clustering algorithms in HallC, JLab

Joint ePIC/EICUG Meeting (AI Town Hall), July 17, 2026

Chi-Kin Tam

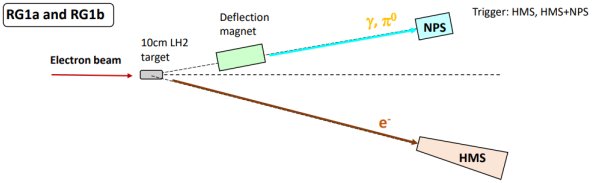
tamc@cua.edu / ckin@jlab.org

Motivation and Goal of the NPS Beam Test



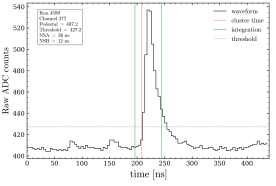
- Future JLab and EIC experiments require very high luminosity to study hadronic structure with high precision (e.g. Sullivan-process meson structure measurements) ⇒ Streaming Readout (SRO)
- Develop AI/ML-based clustering algorithms for real-time reconstruction and event filtering
- Validate the DAQ / reconstruction framework for streaming readout
- Provide an estimation of achievable throughput and latency
- Explore feasibility of fully streaming next NPS run group experiments (expected to run in 4-5 years).

- high precision measurements of cross sections of γ, π^0
- HMS – NPS coincidence, $\theta \in [5.5°, 60°]$
- 30 columns and 36 rows of $PbWO_4$ crystals, each is $2 \times 2 \times 20$ cm^3

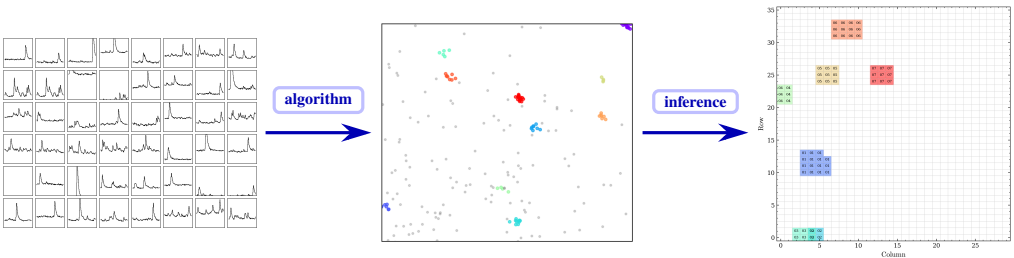


Loose trigger mode for pseudo-streaming :

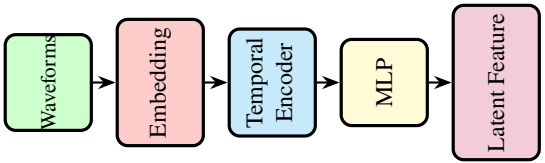
- each readout with PMT, connected to Jlab **FADC250** module
- generate pulse integral and rise time
- VXS Trigger Processor (VTP) module triggers waveform readout



- **Input** : Point Cloud of signals with the Waveform as features.
  - pedestal provided to the model, either from VME config or calibration.
- **Truth** : Cluster IDs, either from `Geant4` simulation, `VTP` FPGA logic or `HCana` reconstruction.
- **Goal** : Assign cluster memberships to each block.



Attention-based Encoder

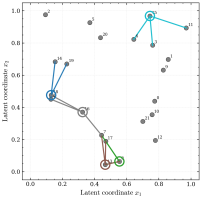


- Input **pedestal-subtracted waveform** [1080][110]
- Embedding waveform sample into higher dimension.
- Attention-based or Long-Short Term Memory (LSTM) to compress the temporal information of the waveform.

see Thursday's talk for detail on ML.

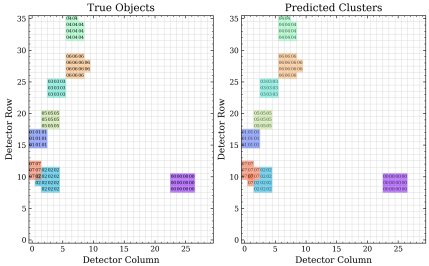
Link to the pytorch model for cluster reconstruction

GNN GravNet



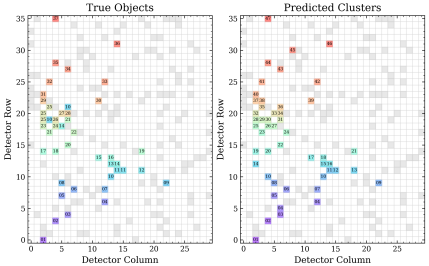
- Input : NPS geometry [1080][2]
- projection to latent space
- build kNN graph
- aggregate features from neighbors scaled by **Grav**itational potential.

Simulation Data



- SIMC + Geant4 simulation
- High-level features as input : E_{dep}, t_0, x, y
- Clean clustering of objects.
- No background generated simulation.

NPS Real Data



- Traditional reconstruction (Hcana)
- Waveforms : Real signals + noise + bkg.
- Clean separation of real signals from noise
- Medium-level clustering due to data bias.

Open Neural Network Exchange (ONNX)

- portability of AI/ML models is always an issue
- provides a standardized model format ⇒ Works across multiple frameworks
- compatible with JANA2, consistent with EIC implementation

ONNX

Train model in any framework → Export as ONNX model → Run inference with ONNX Runtime

using GPU in JANA2

- Set up ONNX Runtime with GPU provider in JANA2
- Test on the frame with CPU and GPU (CUDA) providers

`link to example use of ONNX Runtime`

`link to repo with GPU implementation in JANA2`

Preliminary Latency Test

Scenario

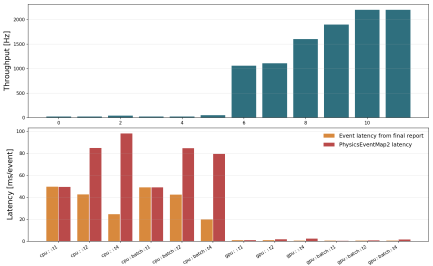
- Traditional CPU reconstruction is expected to dominate latency.
- In an optimistic streaming setup with only AI/ML reconstruction, the key question is whether GPU is required to meet throughput targets.

Configuration

- Threading is managed only by `JANA2`; `ONNX` is fixed to single-thread mode.
- **Random** event source with only the ML reconstruction factory enabled.
- NVIDIA A800 node.

Test Cases

- Compare CPU and GPU runtime.
- `nthreads = 1, 2, 4` → only a small latency change.
- Batch events at the factory level.
- **GPU + batching gives a large throughput gain(**~ 2 kHz**).**



Reported latency excludes traditional reconstruction time, which remains the main bottleneck.

JANA2 and System Integration

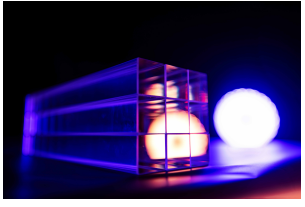
- Integrated ONNX Runtime inference into JANA2 via ONNX JService.
- Verified real-time cluster reconstruction on NPS Run G1a waveform data.
- Preliminary latency study: GPU + batching reaches $\sim$ 2 kHz throughput.

ML Pipeline

- Developed models for cluster reconstruction using high-level features.
- Developed models for cluster reconstruction using waveform with Attention + GNN architecture.
- Built PyTorch $\rightarrow$ ONNX Runtime inference workflow.

Upcoming Beam Test in Hall D

- ML, DAQ, FPGA setup tested independently.
- PMT tested, assembling 3x3 prototype.
- 1 FADC250, 1 VTP, 1 TI module required.
- preparation for DAQ on going.
- Beam test in Hall D in coming August (tentative).

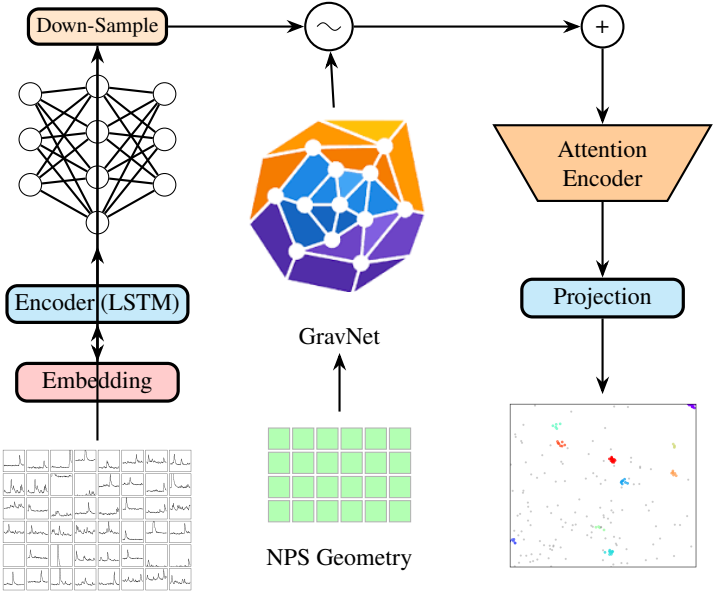


References

Collaborations : Tanja Horn, Dmitry Romanov, Sergey Furletov, Benjamin Raydo, Brad Sawatzky, Joshua Crafts, Avnish Singh, Chi-Kin Tam, and other NPS collaborators.

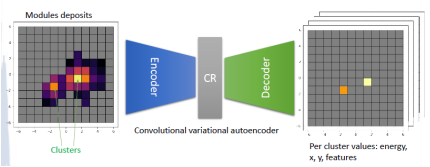
- A declarative implementation of NPS reconstruction in JANA2 : `https://github.com/tck199732/test_jana2_nps.git`
- **Pytorch (geometric) for cluster reconstruction**: `https://github.com/JeffersonLab/nps-sro-ml`
- straight-forward **ONNX Runtime** implementation: `https://github.com/tck199732/ONNX-Runtime-Inference/tree/graph-data`
- Geant4 simulation of NPS detector response with primary vertex from SIMC (by Avnish) : `https://github.com/avnishphy/HallC_SIMC_Geant`

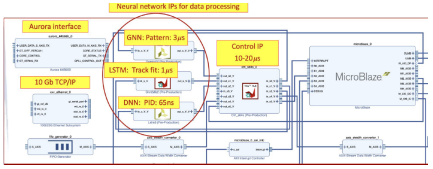
Full Architecture

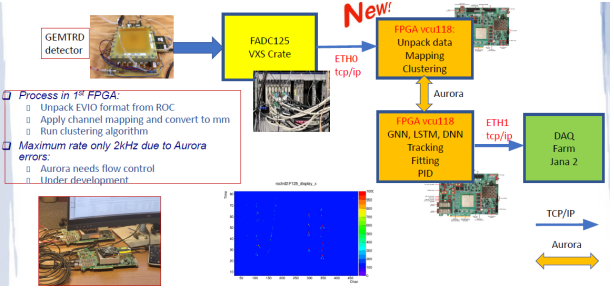


FPGA ML efforts from Sergey, F. et al.

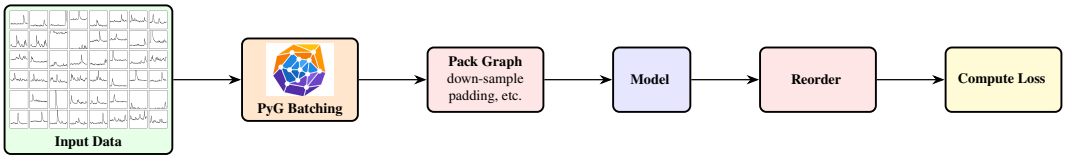
- developed from GEMTRD detector
- GNN tracking
- VAE CNN for calorimeter PIDs (Dmitry Romanov)
- testing for fpga background filtering
- for detail see : ai4eic-2025, ieee 25th talks.







Remark on building model and export to ONNX



- All dynamical control flow must be moved out of the model (conditionals, etc ...).
- For instance, move graph packing/unpacking operations before and after model execution.
- Prefer custom implementation of non-ONNX-supported operations in existing libraries (e.g. `torch_scatter`, `pyg.GravNet`)
- After that, simply `torch.onnx.export()` the model.