Application of Machine Learning for Trigger System

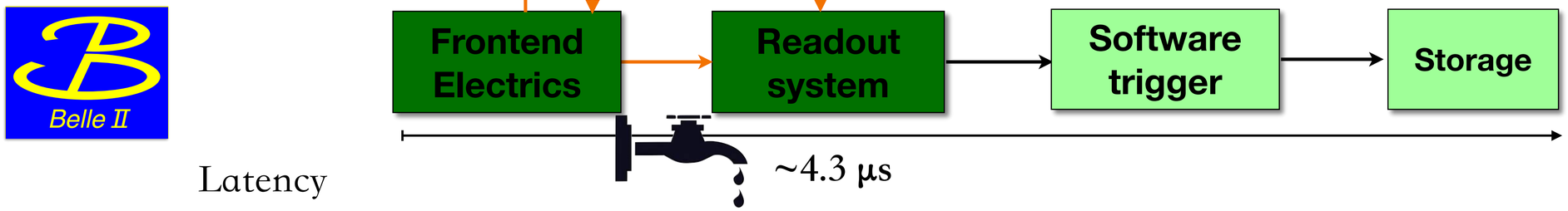
Qi-Dong ZHOU
Institute of Frontier and Interdisciplinary Science,
Shandong Univ. (Qingdao)

16-20 Aug. 2026, Shenzhen
Quantum Computing and Machine Learning Workshop 2026

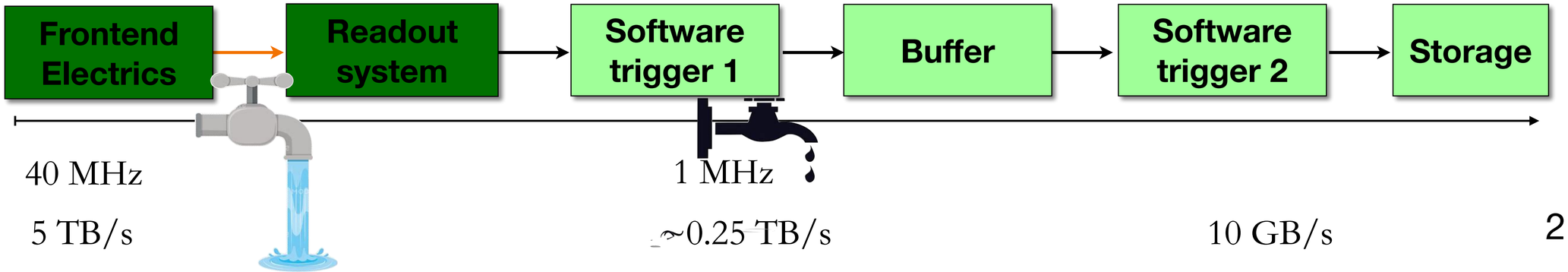
Data processing system (Belle II vs. LHCb)

- Belle II:
  Level 1 (L1) trigger + High Level Trigger (HLT)
  - Trigger efficiency:
    - Had. B physics∼100% τ physics 70~95%

- LHCb: "triggerless" readout & DAQ
  - CPU+GPU based software trigger
  - ~350 GPU RTX A5000
  - Part of online data processing with FPGA for upgrade

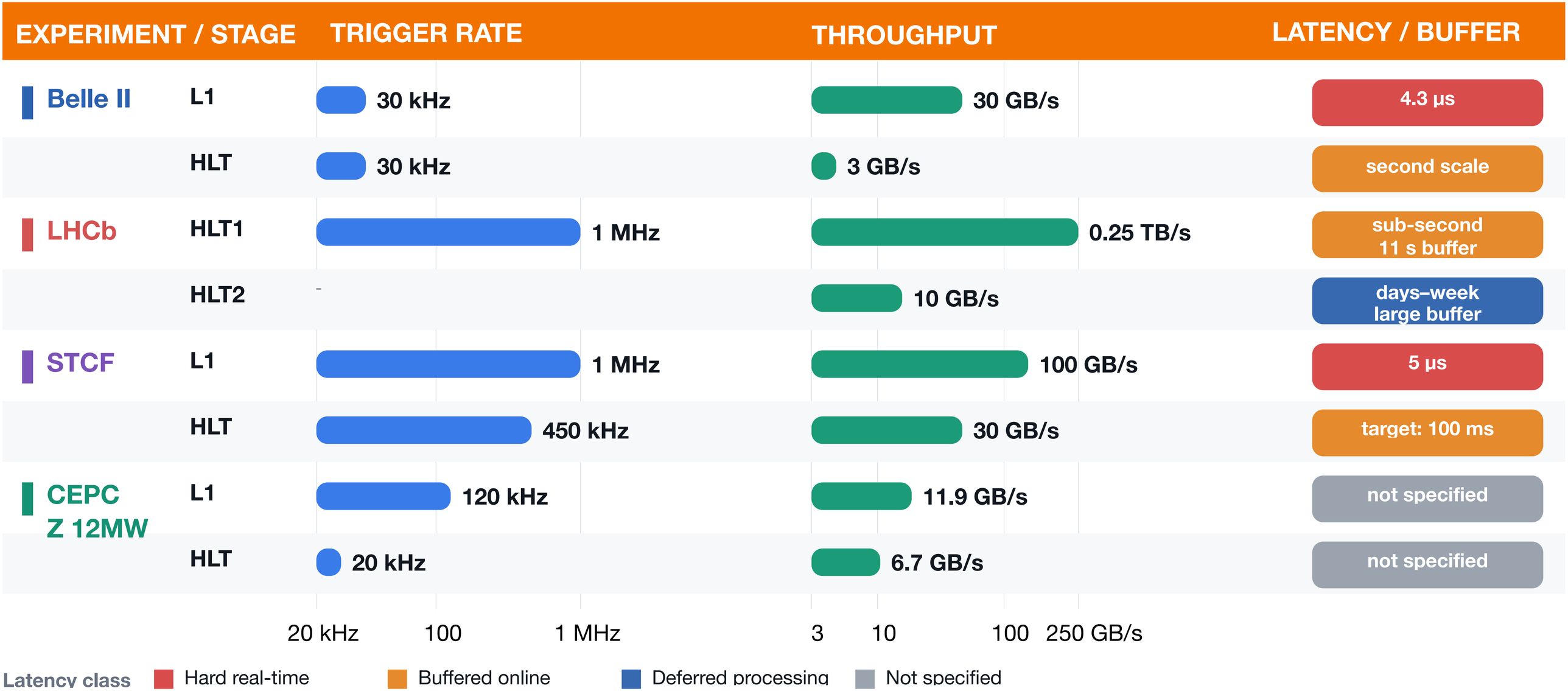


Latency		~4.3 μs	
Trigger rate	127 MHz	30 kHz	30 kHz
Throughput	3(33) GB/s	2 (32) GB/s	3 GB/s



Trigger rate	40 MHz		1 MHz	
Throughput	5 TB/s		~0.25 TB/s	10 GB/s

Different occupy for trigger system



Roadmap of techniques for data processing

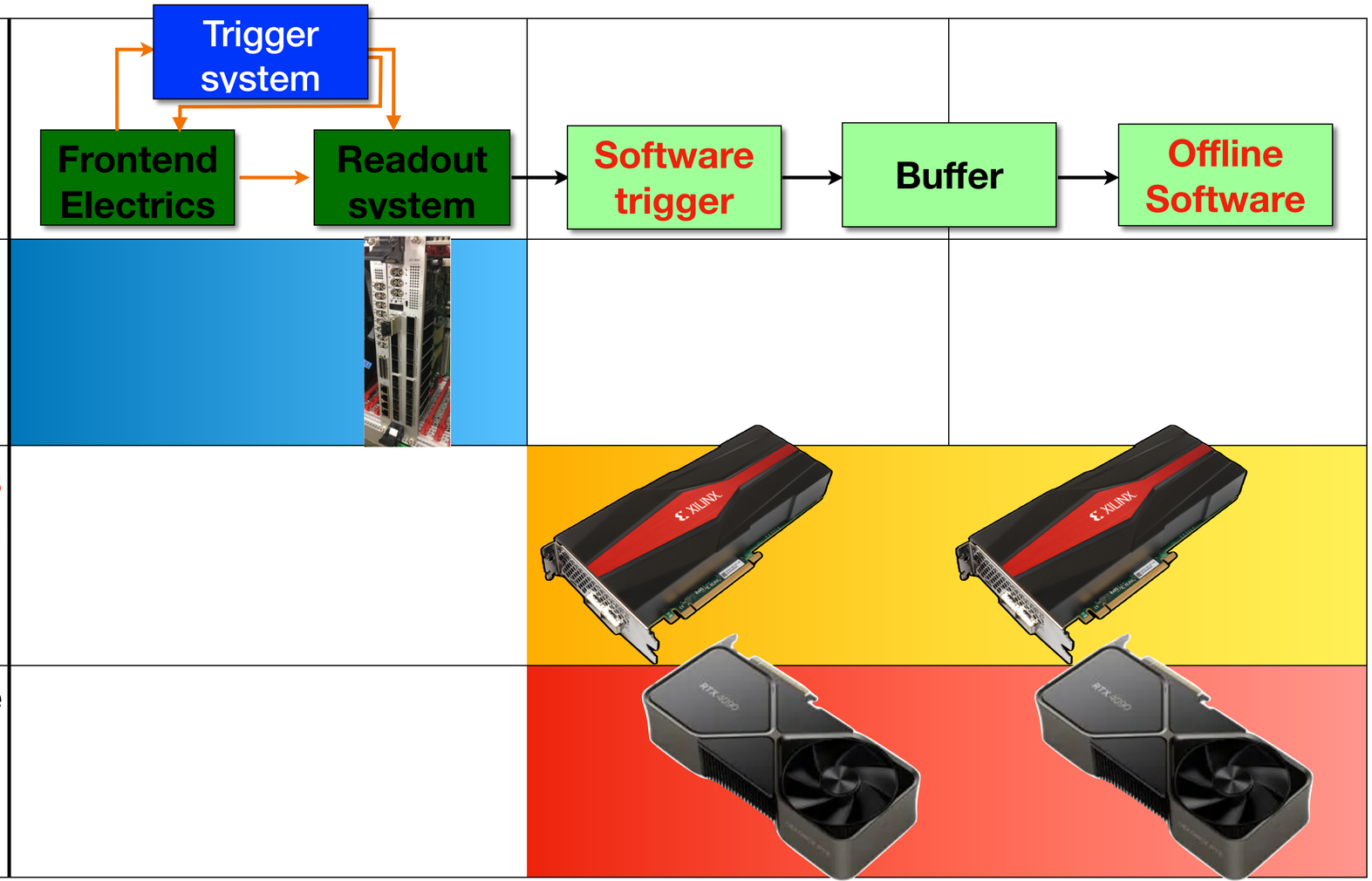
Exp.	Run time	Data (PB)	Total
BESIII	2008-2028	0.5	10
STCF	-	300-500	-
CEPC	-	1.5-3(H) 500-50000 (Z)	-

ECFA detector R&D

Heterogenous computing system

- System integrated with different devices
  - FPGA widely used in frontend and trigger electrics
  - FPGA, AI engine, DPU, System On Chip, Network On Chip for computing acceleration
  - GPU, CPU used for HLT system

Devices	Specifics
FPGA	• Level 1 trigger: low latency w/ **Ous** • Simple ML algorithm available
FPGA/ AI engine/ DPU	• Software trigger w/ **Oms** • CPU/FPGA/GPU system on chip • CNN can run on DPU
GPU	• Software trigger & offline software • User friendly and widely supported



FPGA and GPU development toolchains

- A ML model running on different device
  - CUDA and NVCC provide a user-friendly interface for NVIDIA GPU
  - FPGA-specific toolchains improving, still need specific developments

	FPGA-SPECIFIC	SHARED	GPU-SPECIFIC
MODEL & ALGORITHM	RTL · HLS C++	PyTorch · TensorFlow / Keras C / C++ · Python	CUDA C++ · HIP Triton Language
IR & QUANTIZATION	Brevitas · QKeras QONNX	ONNX · PTQ / QAT DLPack	torch.export StableHLO
COMPILATION & OPTIMIZATION	FINN · hls4ml Conifer · Vitis HLS	LLVM / MLIR TVM / IREE	TorchInductor · TensorRT XLA · NVCC / hipcc
VERIFICATION & PROFILING	C/RTL Co-sim · xsim ILA · Timing Reports	Unit Tests · Regression Numerical Validation	Nsight Systems / Compute rocprof · PyTorch Profiler
IMPLEMENTATION & RUNTIME	Vivado · Vitis XRT / PYNQ	CMake · Linux Python / C++ APIs	CUDA / ROCm Runtime ONNX Runtime · cuDNN / cuBLAS
DEPLOYMENT & SCALING	Bitstream / PDI PCIe / DMA · Board Drivers	Docker / Apptainer CI/CD · Monitoring	Triton Server · Kubernetes NCCL / RCCL

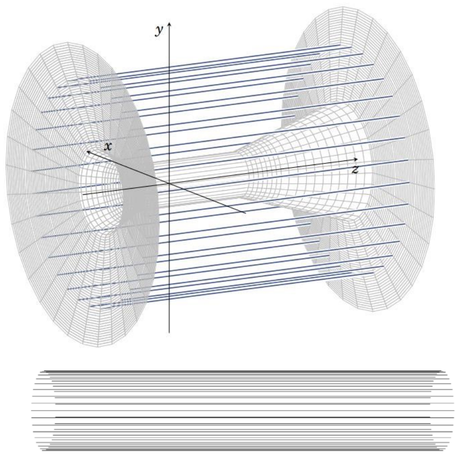
ML application for trigger system

- Some examples of ML application for different level of trigger system

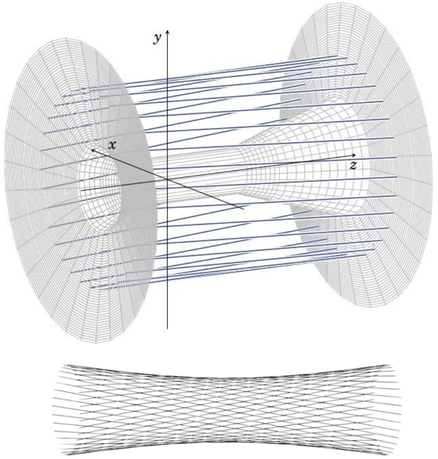
Direction	Representative work	Stage / platform	Maturity	Main limitation
Vertex / track regression	Belle II Neuro-Z	L1 · FPGA	**live data-taking**	fixed latency; calibration
GNN noise filtering	BESIII / STCF MDC	pretracking / HLT	**simulation + prototype**	graph construction; backgrounds
GNN track finding	Exa.TrkX; ETX4VELO	HLT / GPU	**integration R&D**	memory; graph build; tail latency
Quantized tagging	hls4ml jet / tau models	L1 · FPGA	**Phase-2 prototypes**	accuracy-resource trade-off
Anomaly detection	CMS AXOL1TL	L1 · FPGA	**live data-taking**	rate stability; interpretability

Disclaimer: only some example biased by my personal knowledge

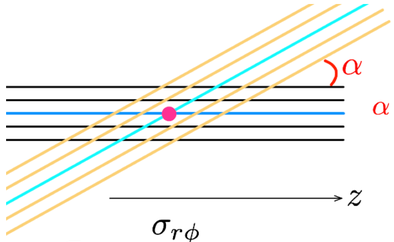
Basics of L1 trigger for draft chamber



Axial wire



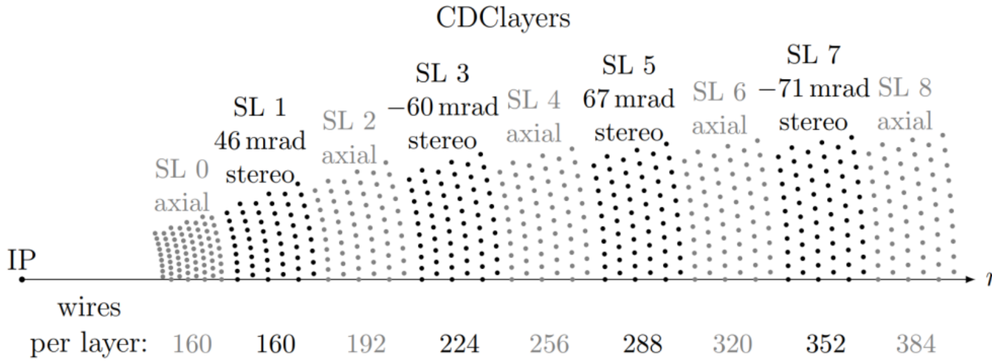
Stereo wire

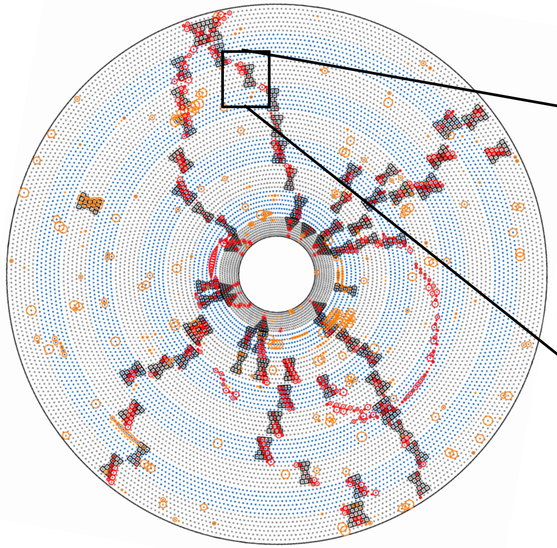


$\alpha = 60 \sim 80\text{mrad}$ (BelleII)

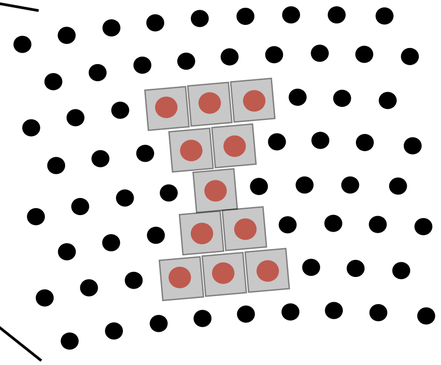
$$\sigma_z = \frac{\sigma_{r\phi}}{\sin\alpha}$$

$$\sigma_z = 1.9 \sim 2.5\text{mm}(\sigma_{r\phi} = 0.15\text{mm})$$

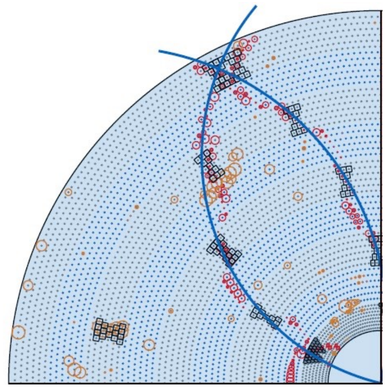




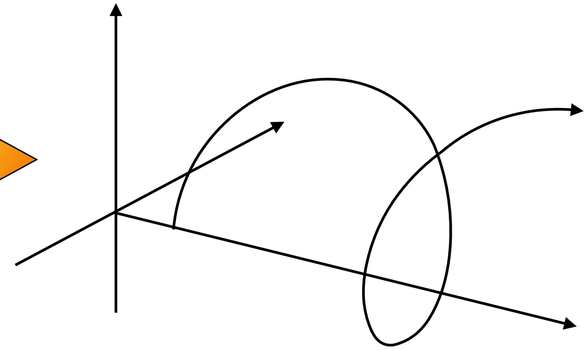
CDC 2D hit map



Track Segment (TS) Finder in every super layer

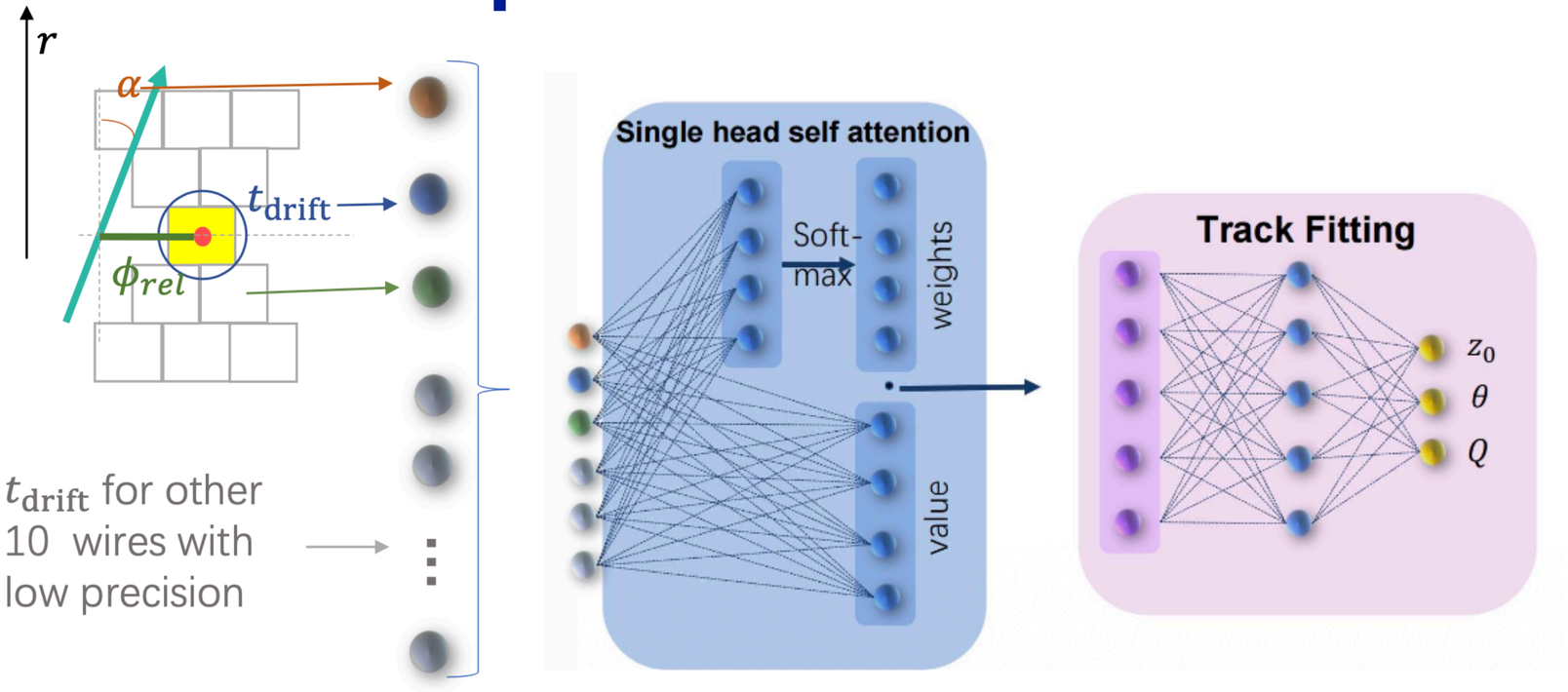


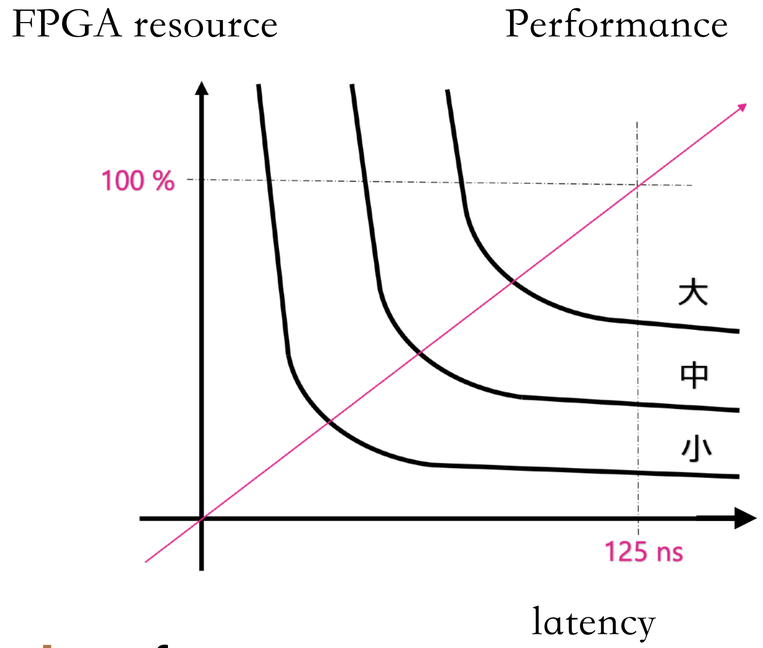
Build 2D track with axial hits using Hough transformation



3D Track trigger base on 2D track and Stereo TS

Deep Neural Network for Z trigger



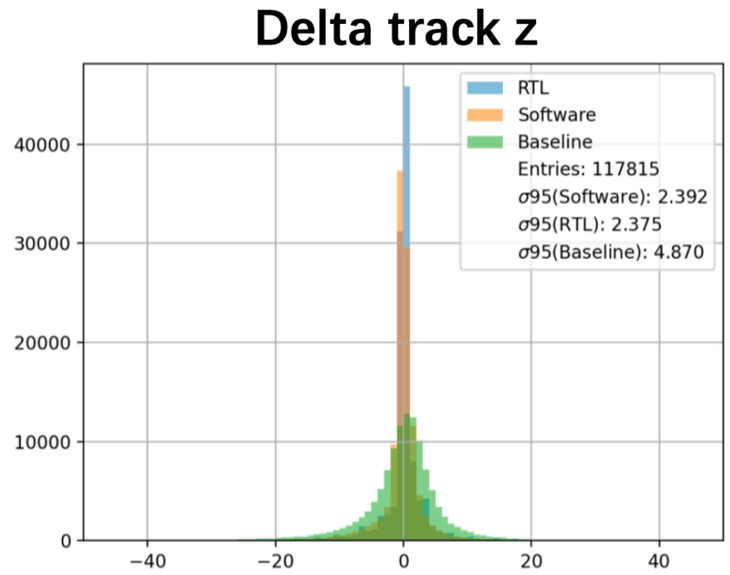


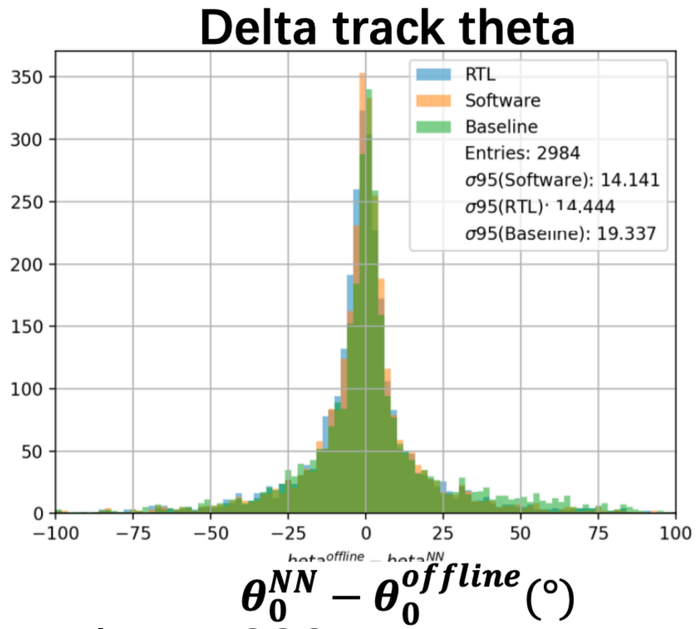
- Inputs: **Drift time t_{drift}**, **wires relative location ϕ_{rel}**, **Crossing angle α** for priority wires + **Drift time for all other wires**
- Introduce the **self-attention architecture** to "focus" on certain inputs
- Output track vertex z_0, track θ and **signal/ background classifier output (Q)**

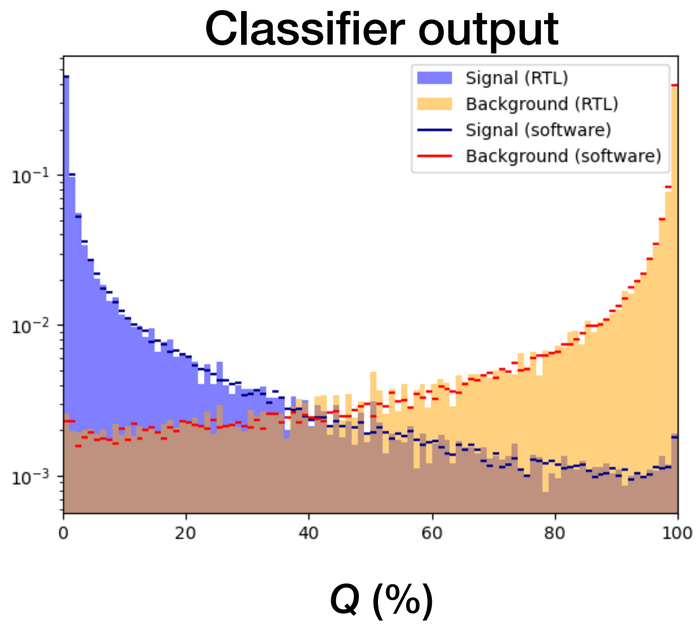
Parameter	#Attention value	#hidden nodes	#hidden layer	activate	precision	Total multiplier
Values	**27**	**27**	**2**	**Leaky Relu**	**Float 16**	**4,185**

- **DNN trigger with HARDWARE is operating for data-taking**

Performance of DNN algorithm

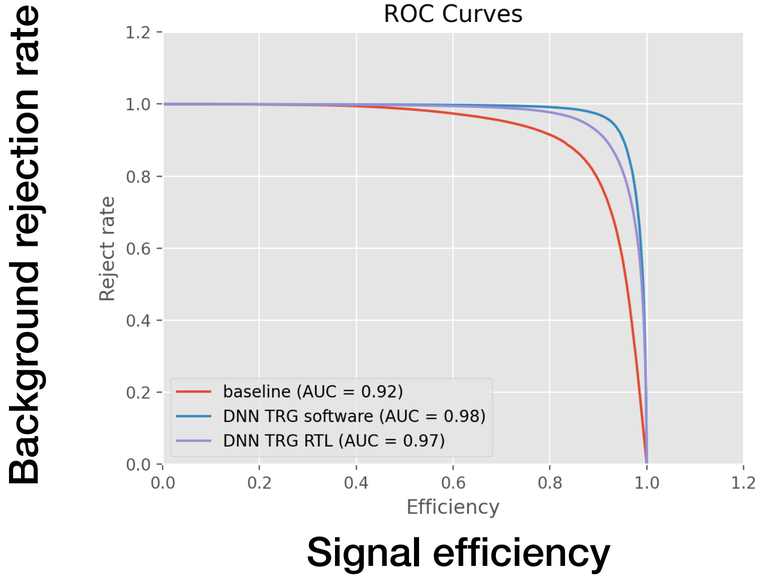




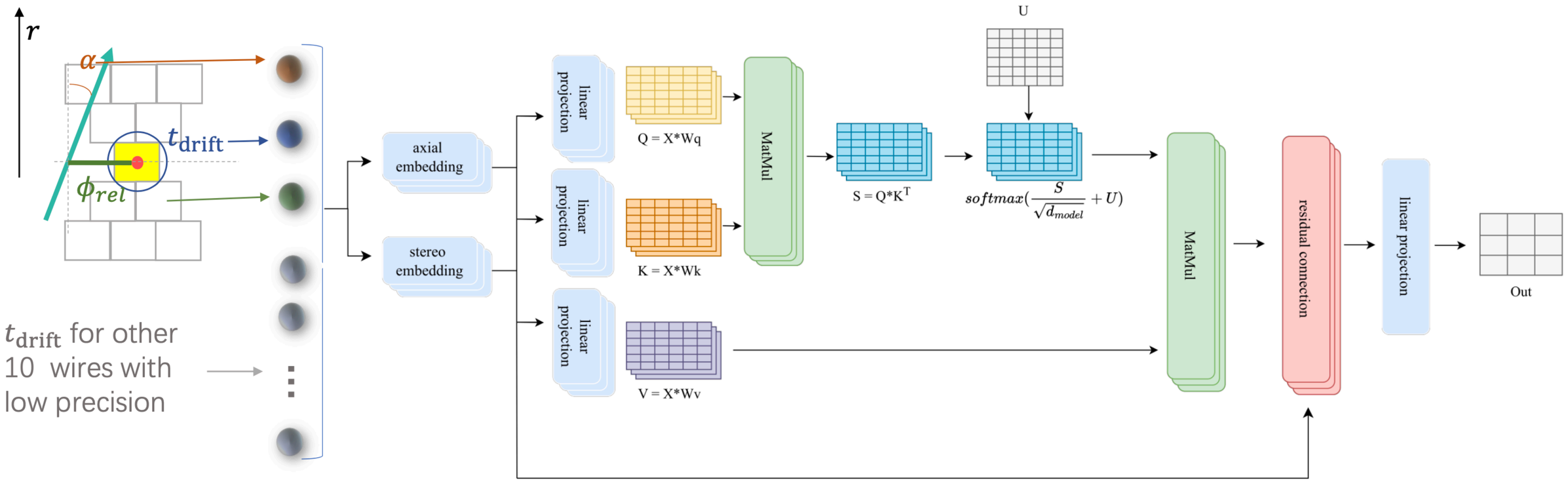


- Latency : 76 clock = **592.2 ns** ;require: < 600ns
- FPGA resource (UT4: Virtex UltraScale XCVU160) usage:
  - DSP: ~75%, LUT: ~45%, others <30%
- AUC do not get large drop comparing RTL and software simulation
- At signal efficiency ~95%
  - Background rejection rate ~85%
- NN 3D tracking reconstruction algorithm for STCF
  - [Y. Yao et. al. IEEE TNS 72, 3 429-437]

[Y. Liu et. al. NIMA 1048 (2025) 171248]



Transform-based Z trigger



- Inputs: **Drift time t_{drift}**, **wires relative location ϕ_{rel}**, **Crossing angle α** for priority wires + **Drift time for all other wires**
- Detector-aware tokenization, Physics-constrained attention, FPGA-oriented lightweight design
- Output track vertex z_0, track θ and **signal/ background classifier output (Q)**

Model	Train. Para.	Multipli.	σ_z	ROC
DNN	4296	4212	4.8cm	0.97
Transform-based	5827	15338	4.9cm	0.97

Development flow of NN-based algorithm on FPGA

Software → Hardware → FPGA

Keras + TensorFlow

PyTorch

- Machine Learning model
- Parameter

hls4ml

- C/C++ transition

Vivado™ HLS

- Translate into Verilog/ VHDL FPGA language

VIVADO

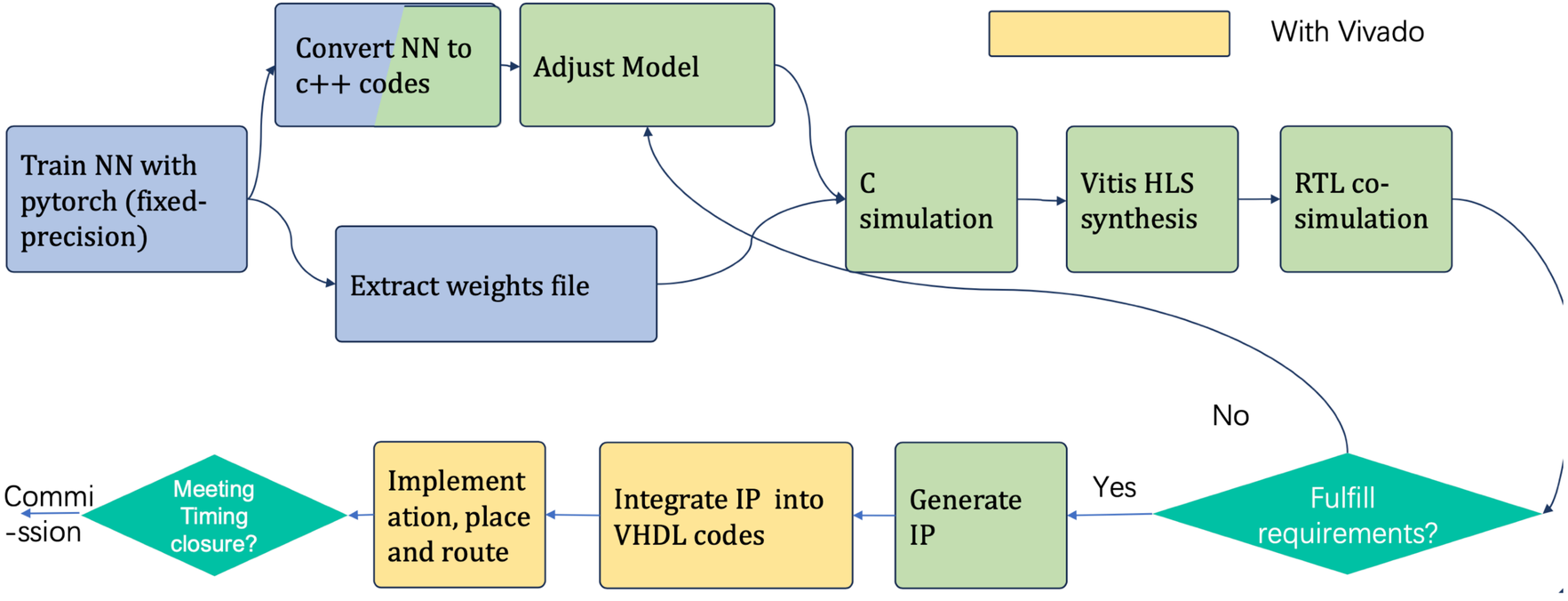
- Start fitter

*include some function from hls4ml lib

With Python

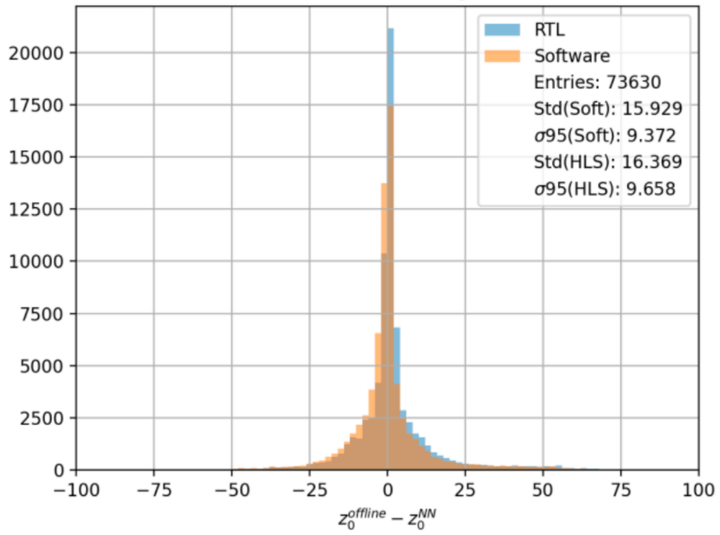
With Vitis HLS

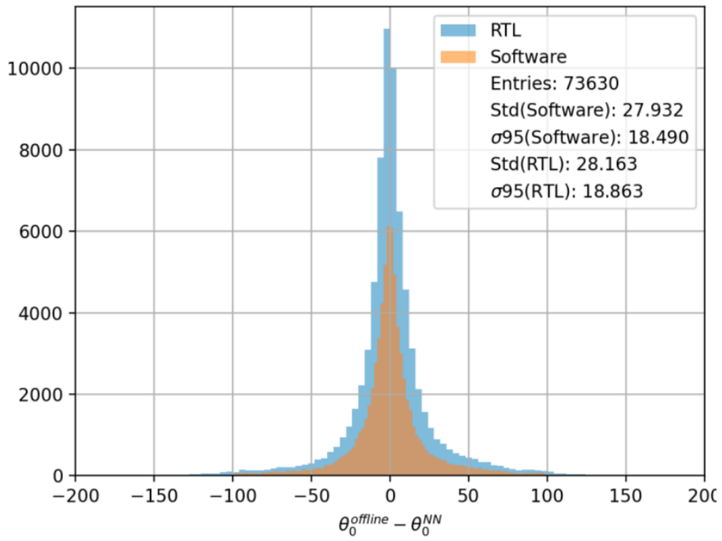
With Vivado

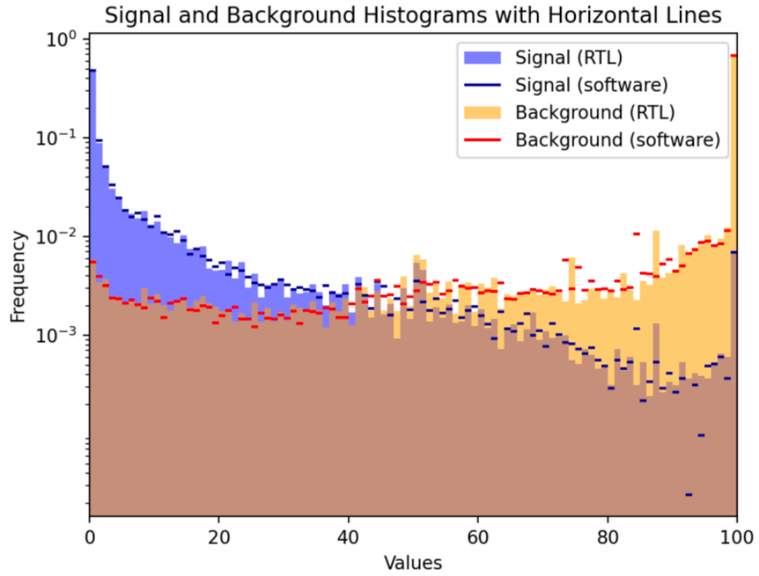


Performance of Transform-based algorithm

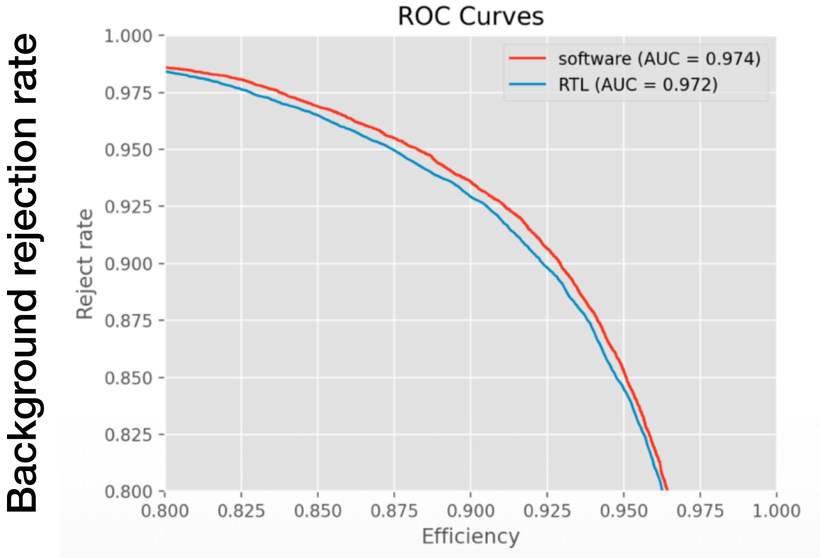
Preliminary



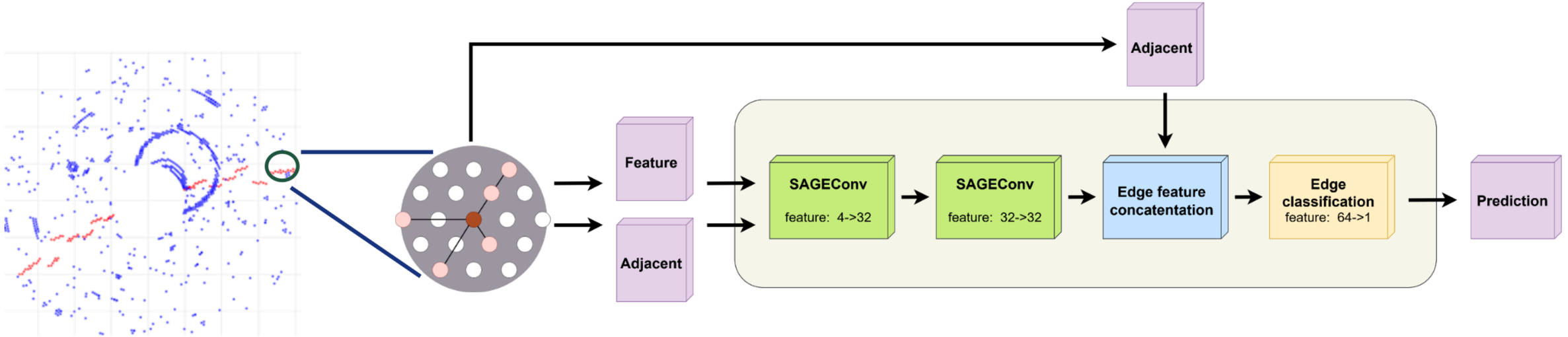




- Latency : 71 clock = **554 ns** ;require: < 600ns
- FPGA (VPK120: Versal premium) resource usage:
  - DSP: ~60%, LUT: ~65%, others <15%
- AUC do not get large drop comparing RTL and software simulation
- At signal efficiency ~95%, background rejection rate ~88%
- Z resolution: 4.5 cm, theta resolution: 18°

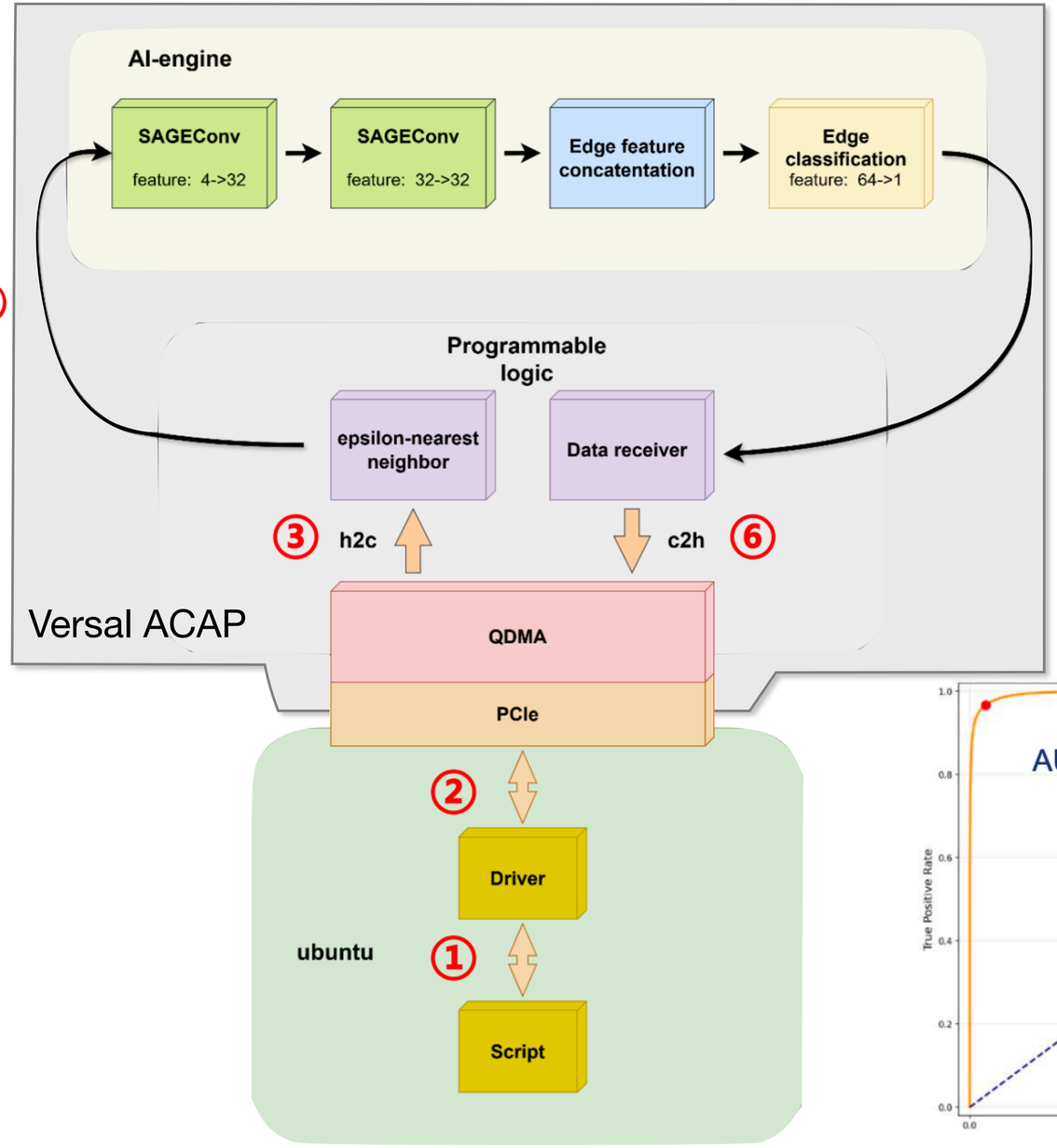


GNN algorithm for noise filtering

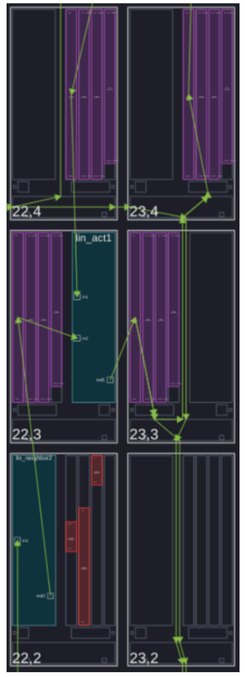


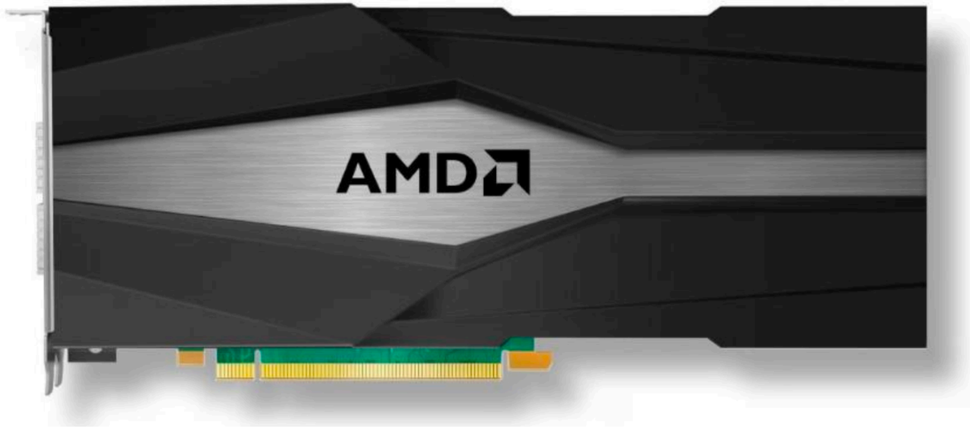
- A GNN model used for background noise filtering
- GNN:
  - Graph construction: epsilon nearest-neighbor
  - Sageconv: message passing
  - Edge feature concatenation: output feature of edges
- Node feature: TDC, ADC, r, ϕ
- Adjacent representation: source and destination node indices

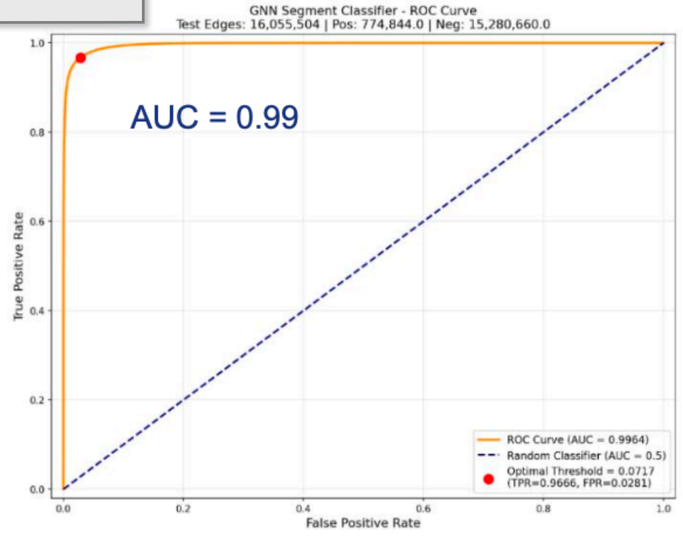
GNN algorithm implementation on Versal ACAP



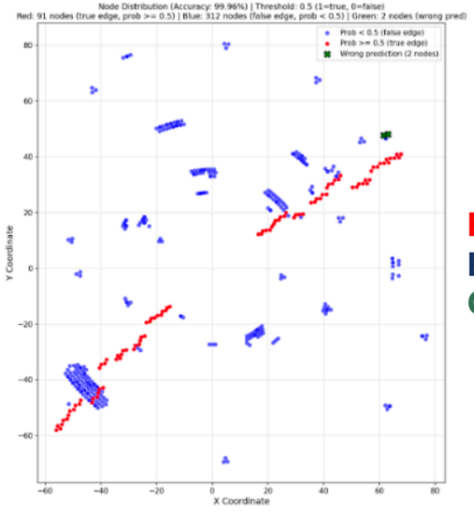
⑤







ROC map



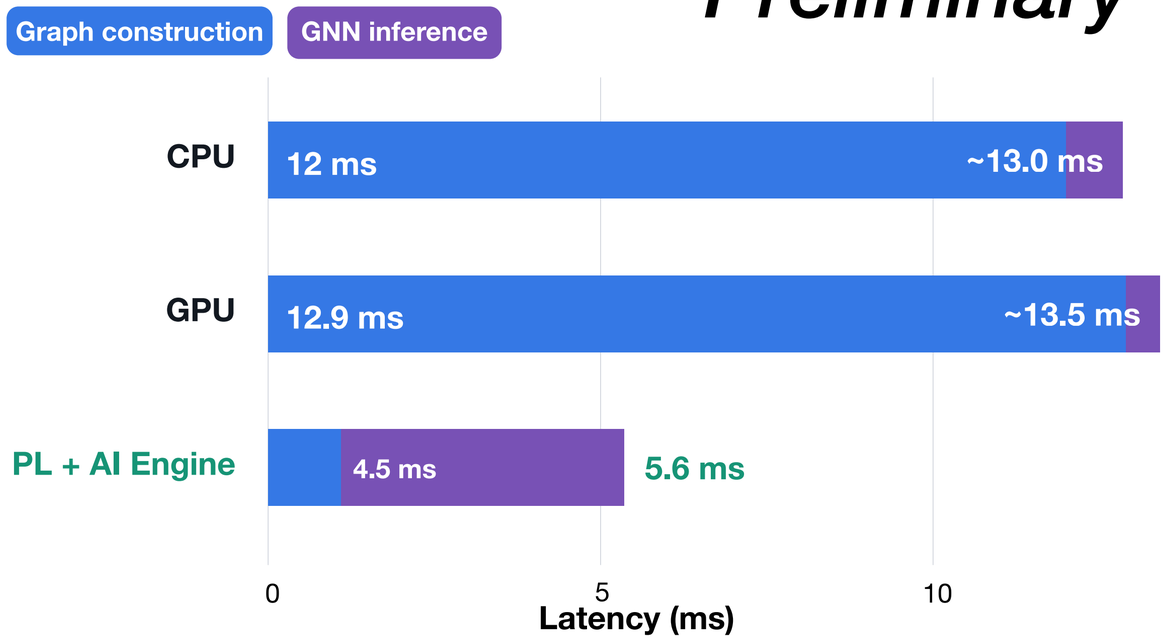
Hit map after training

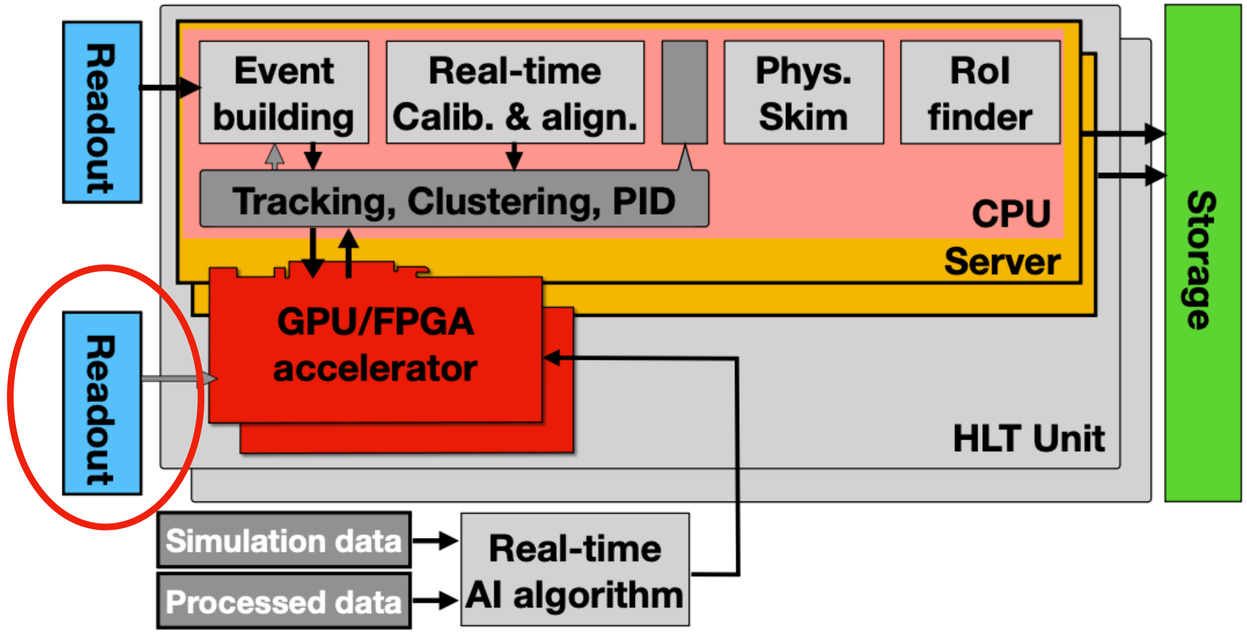
Red true signal
Blue true background
Green false

GNN algorithm for noise filtering

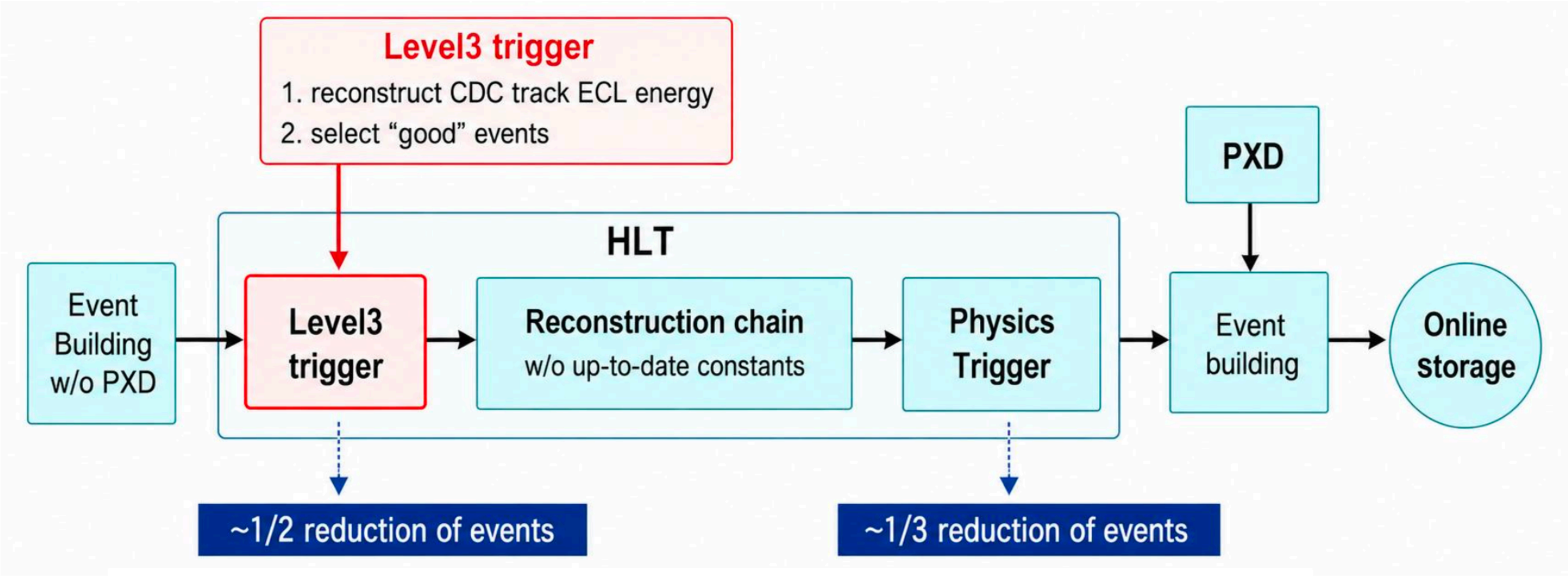
- One event; 200 warmups + 800 runs for CPU/GPU
  - CPU: one thread
  - GPU: RTX 3070
  - NPU: VCK190; no warmup
- A low-clock, dataflow-oriented PL implementation is ~11× faster for graph construction
  - the remaining opportunity is AI Engine inference.

Preliminary





HLT algorithm for Belle II

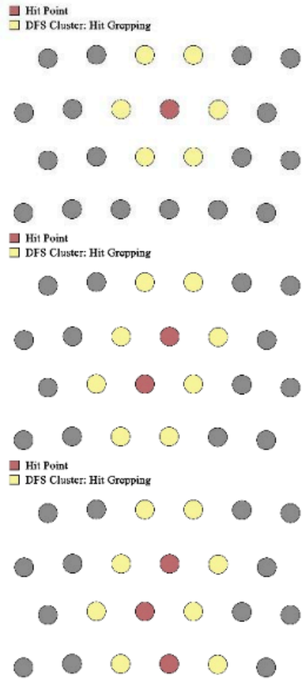


- Fast/simplify tracking and clustering algorithm developed for HLT
  - The module is need for higher luminosity and backgroud
- This Level 3 project is basically for CPU-only HLT, Belle II is trying to run it on GPUs

Tracking algorithm of level 3 trigger

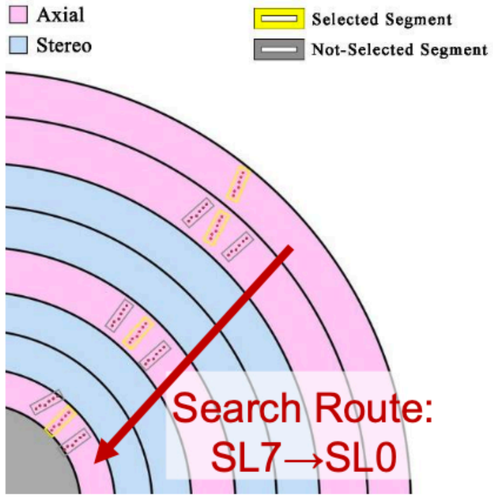
1) Reconstruct Segment

DFS Clustering



2) Link Axial Segment

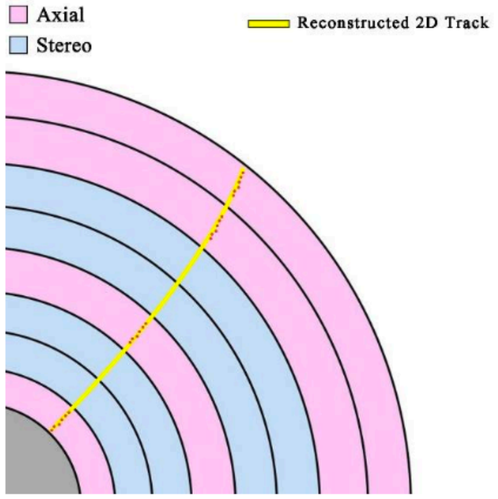
Min χ^2 of Segments' κ



3) Fit 2D Track

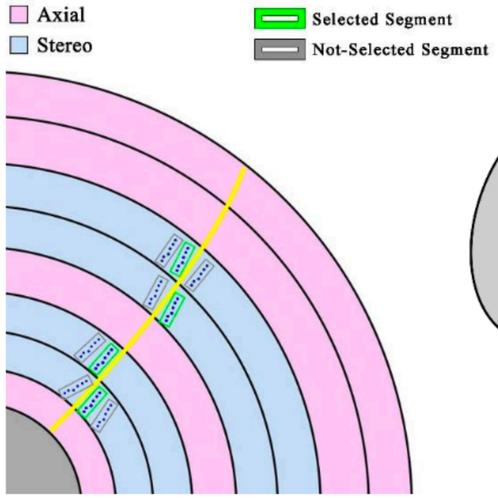
Algebraic Circle Fit

($r-\phi$ plane)



4) Link Stereo Segment

Winner of $s-z$ Consistency

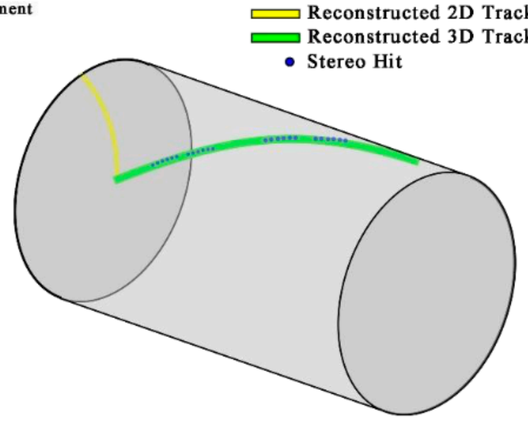


*Check Details about $s-z$ Consistency in Backup

5) Fit 3D Track

WLS Line Fit

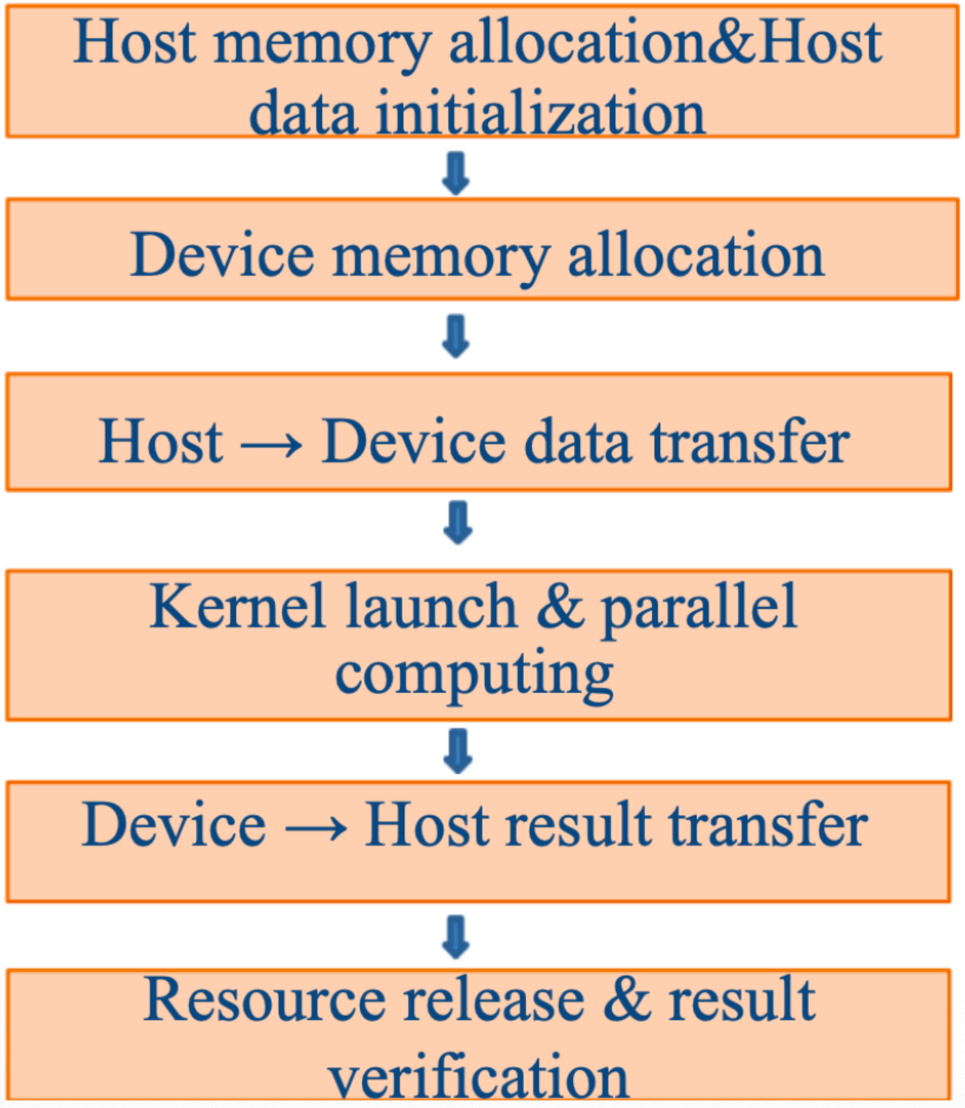
($s-z$ plane)



Hits to **Segments** → **Link** Segments in **Axial SLs** → **Hits** in Linked Segments to Reconstructed **2D Track** & **Collision Point** → **Link** Segments in **Stereo SLs** with Reconstructed **2D Track** → **Hits** in Linked Pair to Reconstructed **3D Track**

Convert the algorithm from CPU to GPU

CUDA program workflow



- Data transfer modification
  - FTSegment POD test
  - Wire coordinates + superlayer parameters copied to GPU
  - CPU/GPU values matched in validation
  - GPU-readable Belle II data layout is feasible

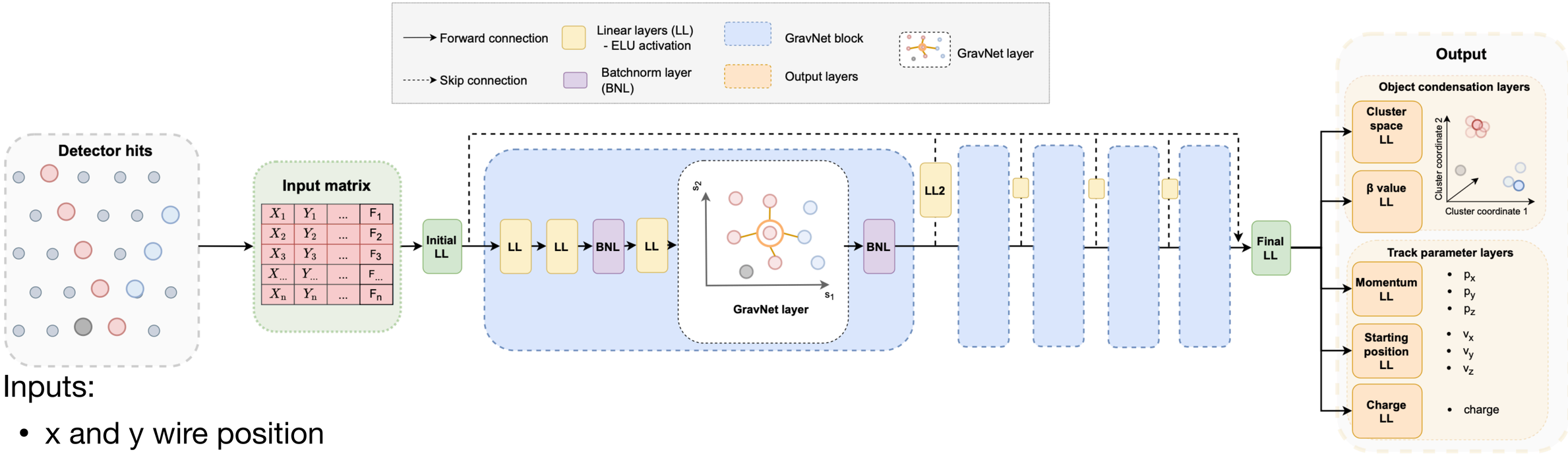
- Kernel-level acceleration
  - Level 3 clustering test
  - CPU serial: ~46 ms
  - GPU kernel: ~0.16 ms
  - Parallel computation is effective; end-to-end time cost is now transfer/setup dominated

GNN for track finding

- Find track parameters: momentum, starting position and charge
- Find unknown number of tracks → Object Condensation (arXiv:2002.03605)
- Computing resource and time constraint may reducible

Noise filtering → Clustering → Fitting

L. Reuter et. al. BELLE2-NOTE-TE-2024-008

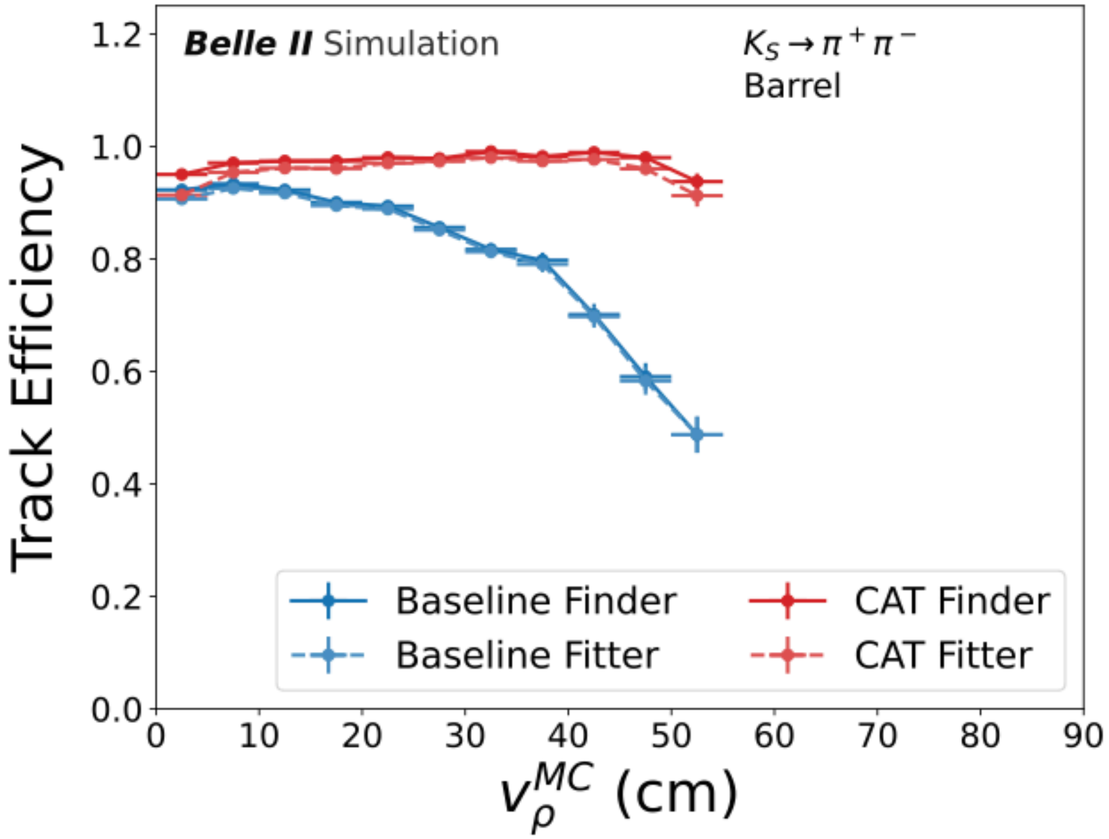


- Inputs:
  - x and y wire position
  - TDC and ADC of signal information
  - layer, superlayer, and layer info. with suprlayer
- Adjustable Parameters
  - 797,812 trainable parameters (3MB weight files)

Performance of GNN

- Efficiency of displaced vertex tracks improved from 85.4% with a fake rate of 2.5%, compared to 52.2% and 4.1%
  - The other performance similar as original algorithm
- Momentum p_x , p_y , p_z starting position v_x , v_y , v_z ,charge
  - Provide initial inputs for GENFIT
- GNN prediction is drawn according to the track parameters predicted by the GNN

L. Reuter et. al. Comp. Soft. for Big. Sci. 9, 6 (2025)



Summary

- AI/ML based trigger system is essential for future collider experiment
- NN, DNN, light-transform based algorithm with hardware based L1 track trigger improve the background rejection
- GNN implemented on Versal ACAP show a solution for HLT
- CPU based tracking algorithm port to GPU for parallel computing showing acceleration power
- GNN with software based track finder to improve the efficiency of displaced vertex tracks

Backup