NEUTRINO PHYSICS AND MACHINE LEARNING 2026

University of California, Irvine · June 15–19, 2026

Dual-Temporal Attention for Direction Reconstruction in Liquid Scintillator Detectors under High-Scintillation Conditions

Xiangyu Qin[1], Shaokai Yang[1], Ziping Ye[2]

[1]Department of Physics, University of Alberta · [2]Department of Physics and Astronomy, University of Pennsylvania

OVERVIEW

Outline

1

The SNO+ Detector

Multi-purpose liquid scintillator · Direction reconstruction motivation

2

The Pipeline: ATHANOR

Data preprocessing · Model architecture · Engineering implementation

3

Direction Reconstruction Validation

Simulation (MC) · Real data

Detector challenge → ML pipeline → validation

SECTION 1

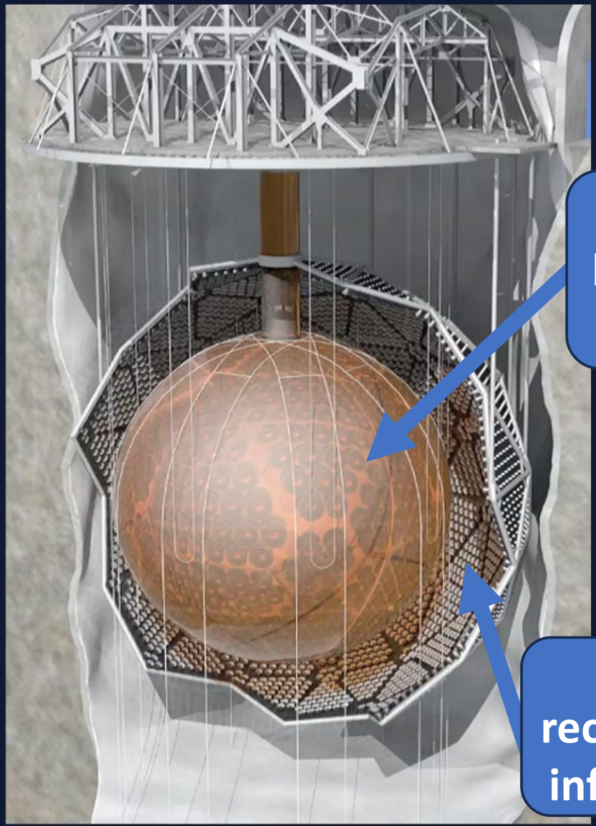
The SNO+ Detector

1

Overview of the SNO+ Detector and current challenges for direction reconstruction for high light yield scintillator

A Multi-Purpose Liquid Scintillator Detector

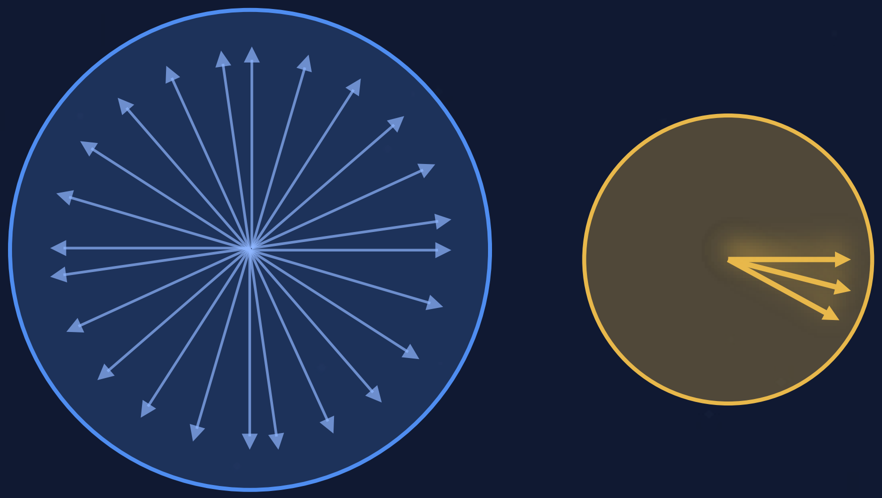
- **~2 km underground**
  Located at SNOLAB, Sudbury, for ultra-low background
- **780 t scintillator**
  Held in the central acrylic vessel (AV)
- **LAB + 2.2 g/L PPO**
  High light-yield liquid scintillator (LS) cocktail
- **PMT hit patterns**
  Events recorded as photomultiplier tube (PMT) hits



SNO+ Detector

Why Direction Reconstruction Is Hard

Two light components



Dominant isotropic scintillation

Weak early Cherenkov-like directional light

The challenge

Under high-scintillation conditions, directional information is **sparse** and embedded in a **large isotropic light background.**

The goal

Learn whether early spatiotemporal hit patterns contain recoverable directional information.

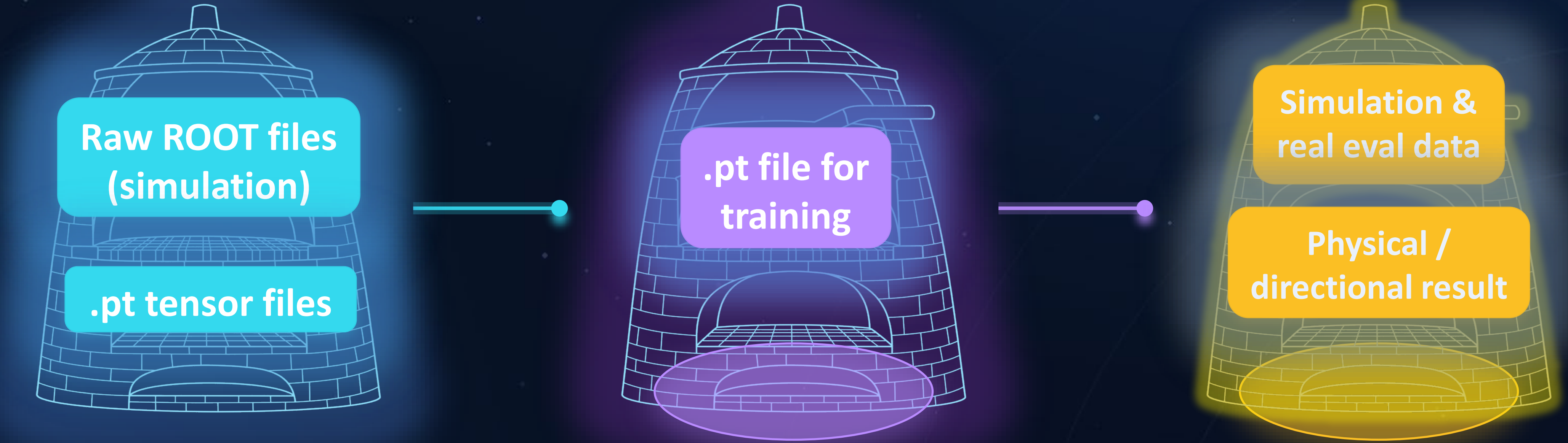
SECTION 2

The Pipeline: ATHANOR

A reproducible PyTorch pipeline: ROOT simulation → 17D tensor cache → training → direction prediction.

The Pipeline: ATHANOR

An alchemical furnace for data — a PyTorch deep-learning pipeline that transmutes raw simulation into directional insight.



Data Preprocessing (CPU)

ROOT → 17D tensor cache

Training (GPU)

.pt shards → model

Evaluation (CPU and GPU)

model → physics

The ATHANOR Pipeline at a Glance

ATHANOR converts variable-length PMT hit patterns from RAT ROOT files into **reproducible fixed-size tensor caches** for direction reconstruction.

- ROOT → tensor once
- Training decoupled from ROOT I/O
- Manifest views = train/val/test
- Model learns hit point clouds

Why Preprocess?

UNIVERSITY OF ALBERTA

Without cache

- ✗ ROOT reading during training
- ✗ CPU bottleneck — GPU waits, not computes
- ✗ Repeated filtering changes sample identity
- ✗ Slow, hard to reproduce

With ATHANOR cache

- ✓ Fixed .pt tensor shards on disk
- ✓ Stable, frozen event identity
- ✓ Fast GPU training, no ROOT parsing
- ✓ Reproducible manifest views

Cache once, train many times.

Step 1 — ROOT Input Data

ROOT Data Analysis Framework

RAT ROOT files

```
electron_…_RiseXpYY/
  run_0001.root
  run_0002.root
  run_0003.root …
```

Multiple Rise conditions enrich the sample set

→

event_ntuple

one row per event

Event-level

`energy, radius, sf_Valid, …`

Hit-level PMT arrays

`hit_time_v, hit_PMT_id_v, …`

Truth labels (MC)

`true_dir, true_pos, sf_pos`

Metadata

`deposit energy, nhits, …`

Step 2 — Event Selection

UNIVERSITY OF ALBERTA

Event-level cuts define a stable fiducial and reconstruction-quality sample before hit-level tensor construction.

All simulated events
↓
Successful reconstructed event cut
↓
Energy cut
↓
Radius cut
↓
Selected events

Reconstruction quality

A valid reconstructed vertex must exist

Energy window

Focus on the certain region of interest

Fiducial volume

Inner radial cut removes edge / external backgrounds

Step 3 — Time Smearing

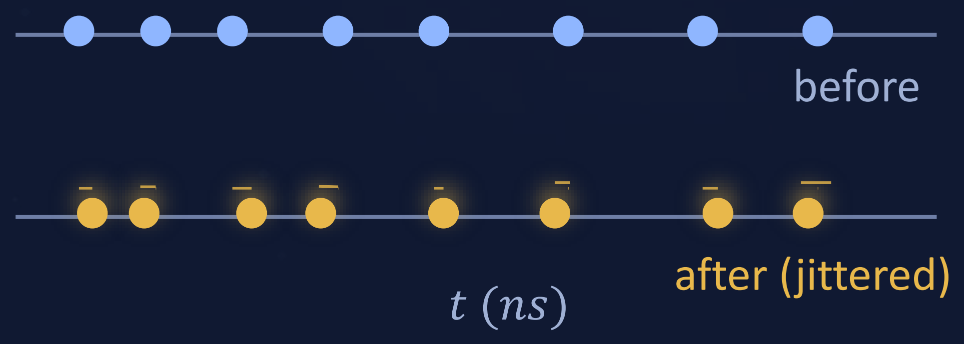
Preprocessing-level smearing (before hit filtering)

$$\varepsilon_i \sim N(0, \sigma^2)$$

$$\begin{aligned} hit_{time}{'}_i &= hit_{time_i} + \varepsilon_i \\ res'_{mci} &= res_{mc_i} + \varepsilon_i \\ res'_{sfi} &= res_{sf_i} + \varepsilon_i \end{aligned}$$

σ scan: `0 ns` `0.05 ns` `0.10 ns` …

Hit times: before → after



Smearing is a preprocessing-level data axis — not a training-time transform.

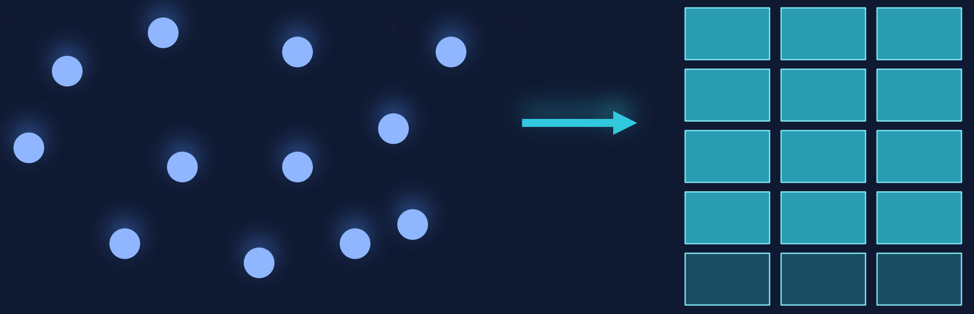
It changes which hits survive and how they are ordered: **time-window survival, hit ordering, residual-window survival, n_keep, and the final tensor contents.**

Step 4 — Hit Processing

Variable-length PMT hit sequences become fixed-length masked tensors while preserving timing and geometry.

Time-window filter → PMT geometry lookup → Remove invalid PMTs → Sort by hit time → Truncate / pad to max_len → Valid-hit mask

Variable PMT hits



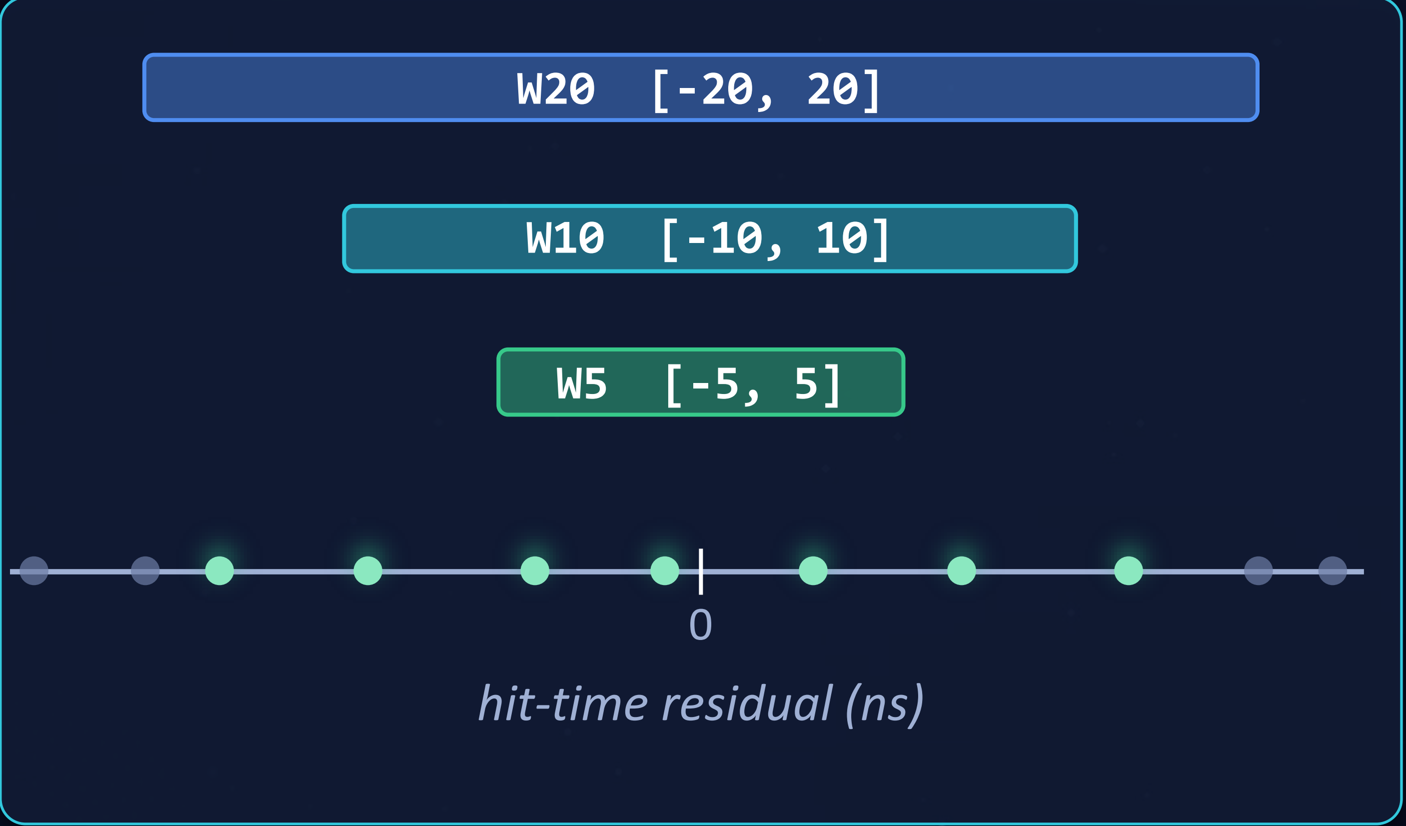
masked tensor

Default hit-level settings

time_window	[50, 500] ns
max_hits_step1	1024
min_hits_step1	500
residual_window	[-20, 20] ns
min_hits_step6	500
max_len	1024

Step 5 — Residual-Window Selection

The residual-window cut uses **res_sf** (residual vs. reconstructed vertex) — available for both MC and real data.



Width trade-off

Narrower windows

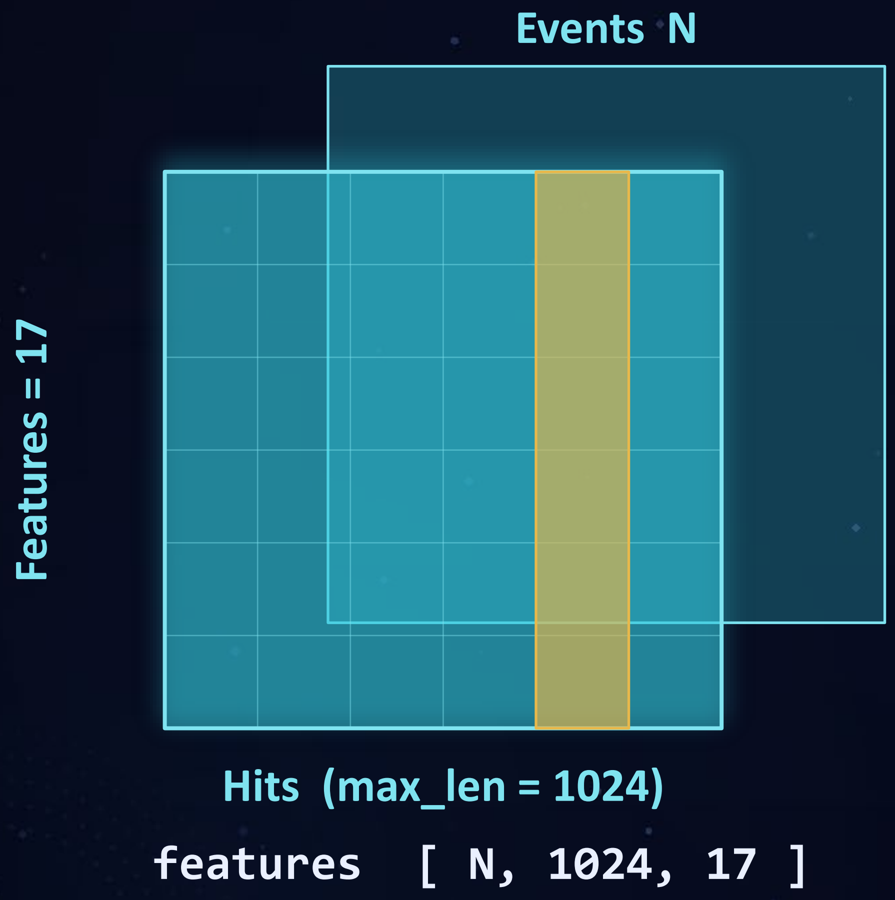
suppress late / inconsistent hits...

...but may also remove **useful directional information.**

W20 is the preprocessing baseline. W10 and W5 apply tighter cuts at the Dataset layer — no ROOT reprocessing required. Same τ applied at training and MC validation.

Step 6 — Build the 17D Tensor

Each selected event becomes a fixed-format hit-feature tensor, packaged with masks, labels, and metadata.



`features`	[N, 1024, 17]
`hit_mask`	[N, 1024] (True = real hit)
`true_dir`	[N, 3] supervised label
`true_pos/sf_pos`	MC & Reconstructed vertices
`metadata`	deposit_energy, nhits, runID, gtid, …

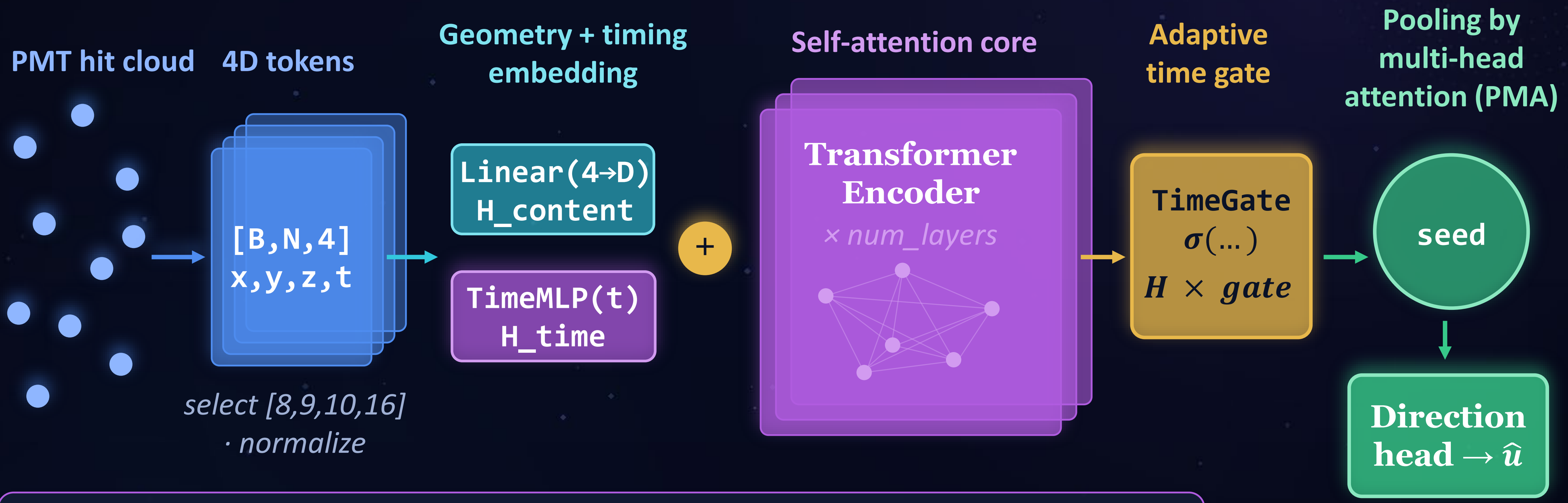
The 17D Feature Schema

F0–F2	PMT unit vector from detector center
F3	hit_time − mean(hit_time)
F4	F3 normalized by std(hit_time)
F5	hit_time − reference quantile
F6	F5 normalized by std(hit_time)
F7	hit-time rank in [0, 1]
F8–F10	PMT absolute x / y / z
F11	residual vs. MC true vertex
F12	residual vs. reconstructed vertex
F13–F15	PMT position relative to hit center
F16	raw hit_time

Baseline input `feature_columns = [8,9,10,16]`

= PMT x, y, z + raw hit_time (a compact 4D geometry + timing start)

Dual-Temporal Attention — Architecture



Direction head & loss

LayerNorm → Linear → GELU → Linear(3) → L2-normalize ⇒ $\hat{u}_{pred} \in \mathbb{R}^3$

loss = mean(1 − $\hat{u}_{pred} \cdot \hat{u}_{true}$)

Dual-temporal design: explicit time embedding + adaptive time-gated token weighting.

The Dual-Temporal Mechanism

SNO+
UNIVERSITY OF ALBERTA

The model uses timing twice — first as a learned embedding before attention, then as an adaptive gate before pooling.

`raw hit time t`

1 · Temporal embedding

- **Multi-Layer Perceptron (MLP)**
  raw time → learned temporal representation
- **Added to token**
  timing injected into the token embedding
- **Model correlations**
  transformer can use timing-dependent structure

2 · Temporal gate

- **TimeGate**
  produces a per-hit timing weight $\sigma(\cdot)$
- **Reweights tokens**
  gated features H × gate before pooling
- **Early-time emphasis**
  learned to emphasize informative early structure

Note: early-hit emphasis is learned from data — it is a reconstruction mechanism, not an explicit optical-component (Cherenkov / scintillation) separation.

Reproducible Data Boundary

CPU · Preprocessing

`Scan_rise_dirs.py/build_root_lists.py`

`Preprocess_mc.py`

→ **17D tensor cache shards (.pt)**

`validate_cache_schema.py`

`build_cache_manifests.py`

GPU · Training

`direction_dataset.py (reads .pt + manifest)`

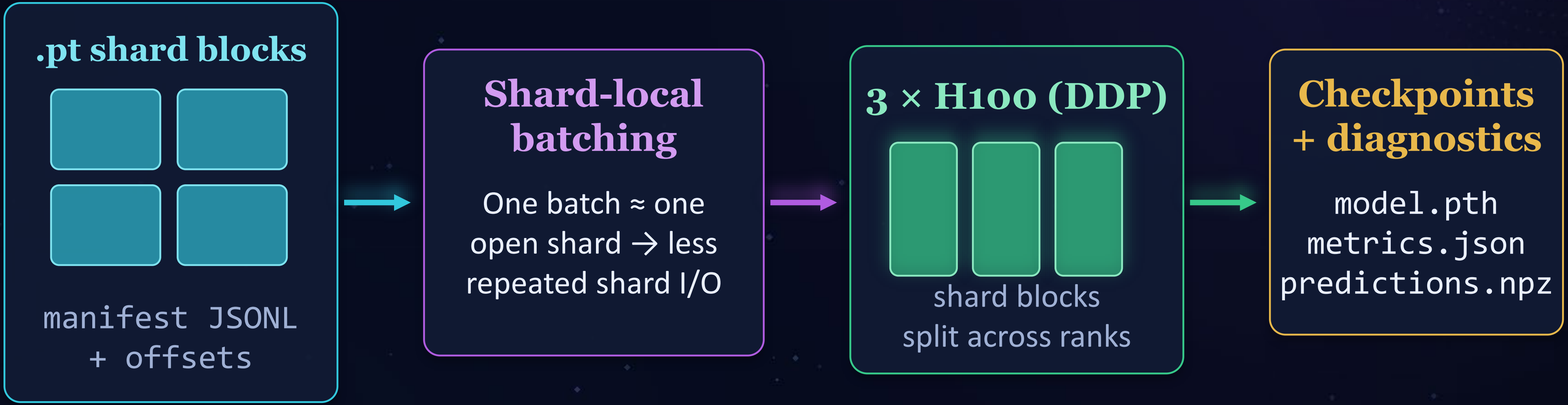
`train.py (Distributed Data Parallel(DDP), checkpoints, metrics)`

→ **predictions + diagnostics**

Rise combinations, residual windows, balanced / stratified views, and feature selection **are all handled by the manifest / Dataset / config layers — never by re-reading ROOT.**

ROOT I/O stops here

H100-Friendly Training System



Baseline configuration

- hidden=512
- heads=4
- layers=3
- batch=128
- max_len=1024
- feat=[8,9,10,16]
- cosine loss

SECTION 3

Direction Reconstruction Validation

$\cos\theta$ between predicted and reference directions — in simulation (MC) and in real solar-neutrino candidate data.

3

MC Electrons

Data information

Sample
MC electron events (held-out test set)

Reference
MC truth direction

Check
$\cos\theta$ (predicted, truth) distribution

Rising toward +1 indicates successful direction reconstruction.

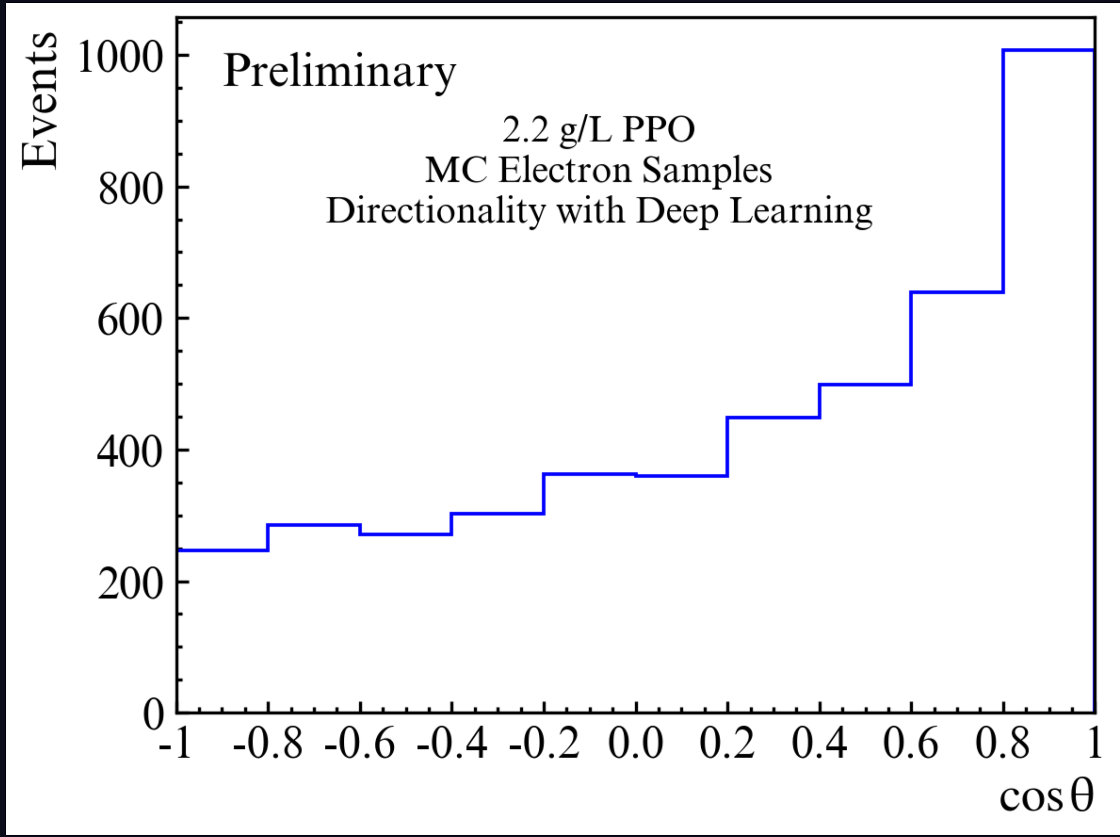


Fig. MC electrons: predictions align with MC truth as cosθ → 1.

Real IBD Candidates

Data information

Sample

Real anti-neutrino candidates from SNO+, detected via inverse beta decay (IBD: $\bar{\nu}_e + p \rightarrow e^{+} + n$).

Reference

Outward radial direction (center → vertex)

Check

$\cos\theta_{pred \cdot center}$ distribution

Flat response indicates no direction bias.

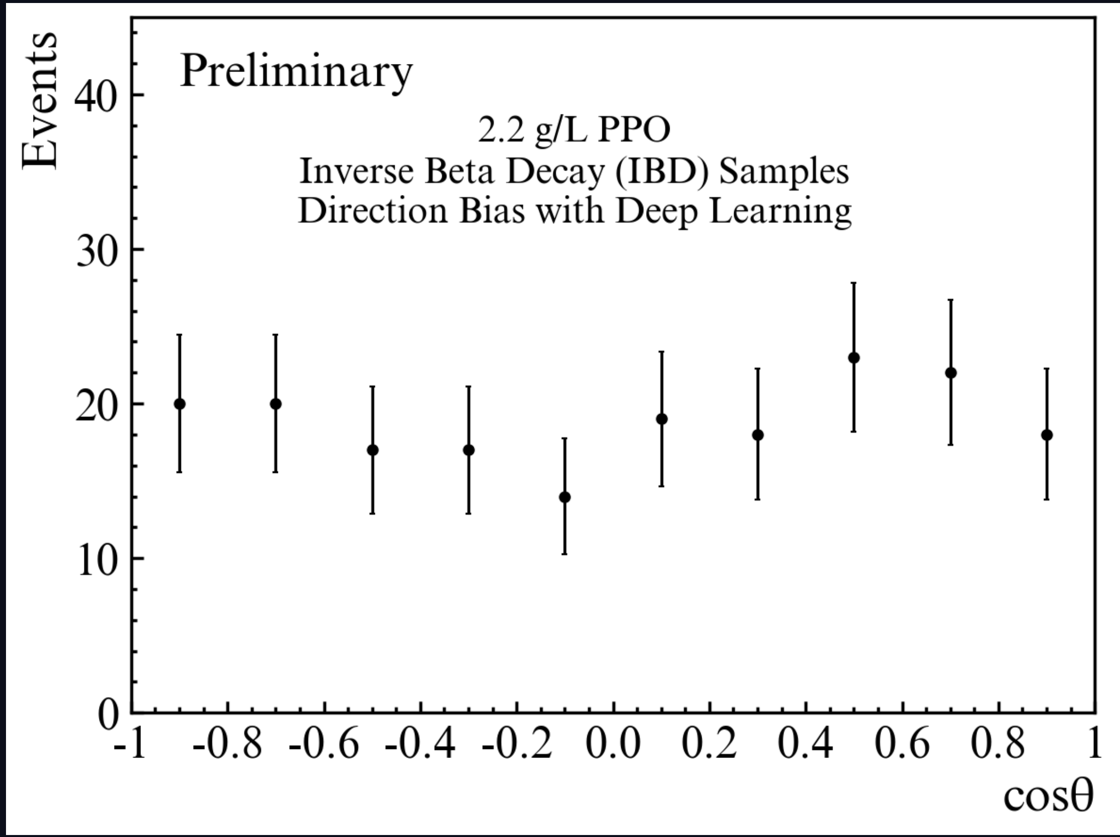


Fig. IBD samples: flat cosθ response checks for direction bias.

Real Solar Neutrino Candidates

Data information

Sample

Real solar neutrino candidates

Reference

Expected solar direction at event time

Check

$\cos\theta$ (predicted, solar) distribution

Forward excess near $\cos\theta = 1$ indicates directional consistency with the expected solar direction.

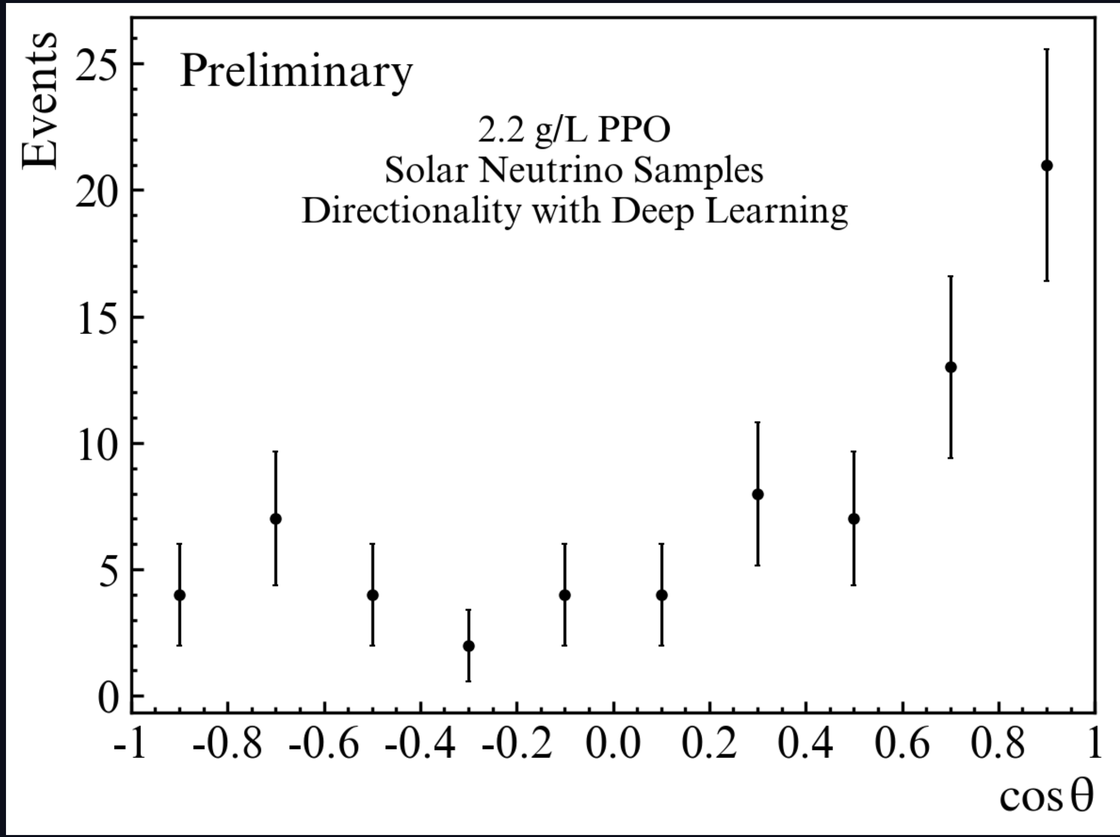


Fig. Solar neutrino samples: forward excess follows the expected Sun direction.

CONCLUSIONS

Summary

Reproducible Pipeline

ATHANOR provides a ROOT → tensor → training pipeline. Fixed 17D caches decouple expensive ROOT preprocessing from GPU training.

Dual-Temporal Architecture

Combines PMT geometry, hit timing, self-attention, an adaptive time gate, and PMA pooling over an unordered hit cloud.

$\cos\theta$ Validation

Direction reconstruction validated with cos θ distributions in MC (truth) and real data (expected solar direction).

Toward robust direction reconstruction in high-light-yield liquid scintillator detectors.

Designed to study directional information under high-scintillation conditions — without overclaiming explicit Cherenkov / scintillation separation.

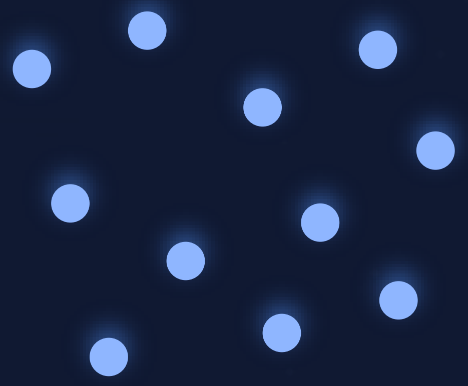
Thanks

BACKUP

Permutation-Aware Hit Modeling

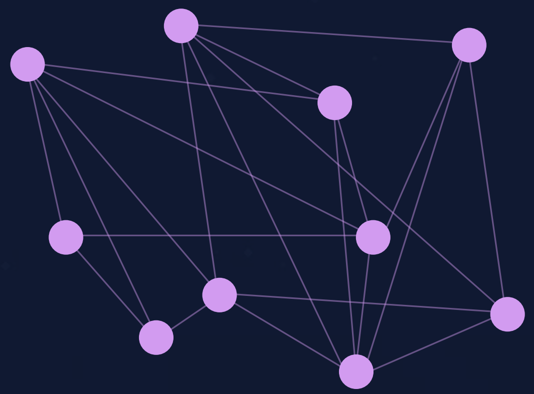
Instead of imposing an image grid, the model treats each event as a masked PMT-hit point cloud.

Unordered hits → tokens



variable hit count
padding + mask → fixed N

Self-attention



long-range hit–hit correlations

Pooling with Multi-head Attention → event vector

1 vector per event

aggregates variable valid-hit tokens

No image grid — order-invariant point-cloud modeling.

Hit-Residual Token Pruning

Physical Motivation

In a high-light-yield liquid scintillator, most PMT hits originate from late-arriving isotropic scintillation photons. Directional information is carried by a small subset of early, prompt hits.

Pruning Criterion

Each token carries ***F12***: the hit-time residual relative to the ScintFitter reconstructed vertex:

$$r_i^{sf} = t_i^{hit} - t_i^{expected}(\hat{x}_{sf})$$

Tokens with $|r_i^{sf}| > \tau$ are removed before the Transformer encoder.

Design

- $\tau \in \{20, 10, 5\}$ ns — $\tau = 20$ ns is the preprocessing baseline
- Applied at training (MC) and validation; τ may differ between MC and real data
- No trainable parameters; hit mask updated in sync
- Applied at Dataset layer — no ROOT reprocessing required
- $N \to N' < N$, cutting attention cost from $O(N^2)$ $\to O(N'^2)$

Interpretation Note

Emphasizes early-timing structure — not an explicit Cherenkov/scintillation separation.