Machine Learning for Particle Physics: Event Reconstruction from Detector Data

Dmitry Bylinkin

Basic Research of Artificial Intelligence Laboratory, MIPT

July 2026

Roadmap



- A practical recipe: what to build, how to check it, how to make it better.
- Architectures and case studies.
- Minimal theory.

One event, many questions

A detector event is a cloud of hits. The same input feeds many tasks.

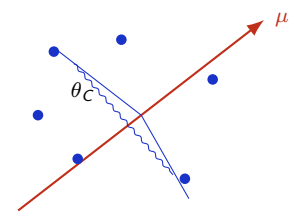
Task	ML type
Direction	regression
Energy	regression
Particle ID	classification
Noise tagging	per-hit classification
Track finding	instance segmentation
New physics	anomaly detection

Running example: muon track reconstruction

A relativistic muon emits Cherenkov light at a fixed angle

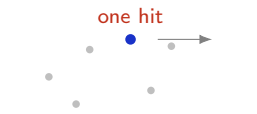
$$\cos\theta_C = \frac{1}{n\beta}.$$

Task: recover the unit direction $\hat{v} \in \mathbb{S}^2$. No magnetic field.
Metrics: angular resolution, distance error



wavefront hits a subset of sensors

A hit as a feature vector



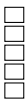
An event is a set of hits of variable length:

$$E = \{x_1, \ldots, x_N\}.$$

- Space-time features are centered using the earliest hit and scaled;
- Charges are normalized as $\tilde{q}_i = \log_{10}(1 + q_i)$;
- The data is augmented.

Event is neither an image nor a fixed vector

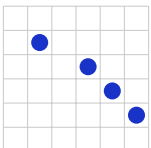
MLP



? order

✗ N varies

CNN



✗ ∼99% empty

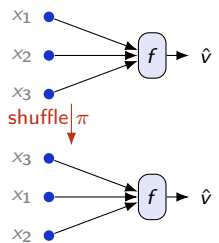
Point cloud



✓ natural

We need an architecture built for *sets of points*.

The key symmetry: permutation invariance



same set, relabeled ⇒ same output

Hits are unordered:

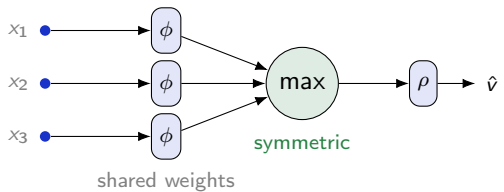
$$f(\{x_1, \ldots, x_N\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(N)}\}).$$

We need an architecture that respects permutations.

Step 1: PointNet

Standard loss function for NPP models is

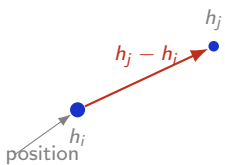
$$\mathcal{L} = 1 - \langle \nu, \hat{\nu} \rangle.$$



✓ Simple, fast, linear in N;

✗ Track geometry is lost.

Step 2: EdgeConv



Graph: hit = node, edges to k nearest neighbors.

$$h_i' = \max_{j \in \mathcal{N}(i)} \mathrm{MLP}\Big(\underbrace{h_i}_{\text{global}} \,\|\, \underbrace{h_j - h_i}_{\text{local}} \Big)$$

$$\hat{y} = R(\{h_i\})$$

- h_i is a latent vector, corresponding to point.
- $(h_j - h_i)$ carries the local structure.
- max over neighbors keeps invariance.

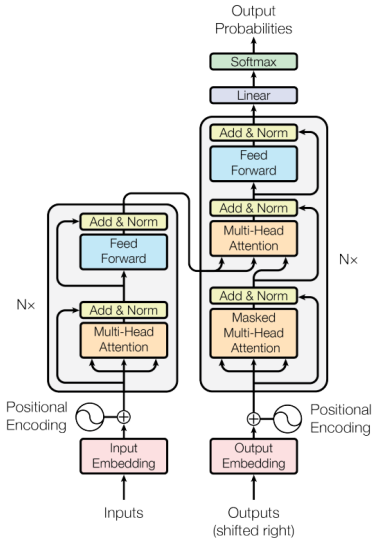
Step 2: remark

- **GraphNeT** – GNNs purpose-built for NPP problems. (2210.12194)
- **GravNet** – GNN for irregular detector geometry. (arXiv:1902.07987)

Watch out

Very deep GNNs suffer oversmoothing, their nodes' features become indistinguishable.

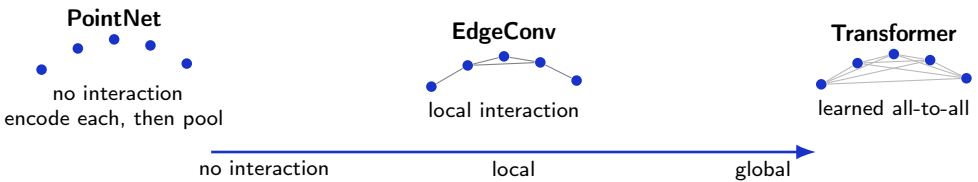
Step 3: Transformer



- Each hit is processed by feed-forward layers;
- Each hit looks at all hits with learnable weights;

Three steps, side by side

	Interaction	Cost	Invariant?	Best when
PointNet	none	$\mathcal{O}(N)$	✓	fast baseline
EdgeConv	local (kNN)	$\mathcal{O}(Nk)$	✓	geometry matters
Transformer	global (all-pairs)	$\mathcal{O}(N^2)$	✓	global structure, compute available



The Kaggle challenge "Neutrinos in Deep Ice"

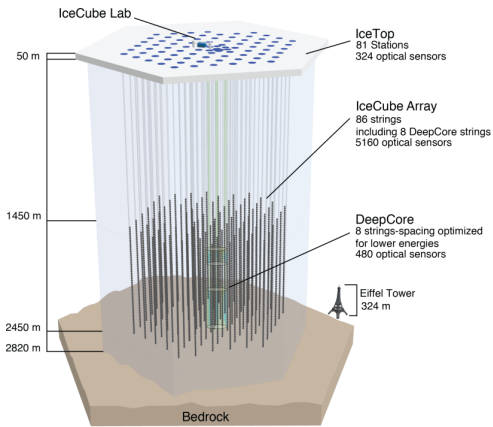
- Jan–Apr 2023; $\sim$900 teams, $>$11,000 submissions, \$50k prizes. (arXiv:2307.15289)
- Data: $\sim$138 million simulated ν interactions. Each event: pulses (t, charge, `sensor_id`) + geometry file.
- Target: angular resolution.

Baseline shipped to participants

DynEdge — a graph network from `GraphNeT`, trained on $<$8% of the data.

The Kaggle challenge "Neutrinos in Deep Ice"

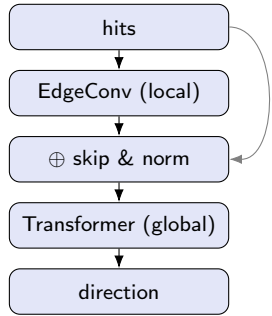
$\sim$5160 optical modules on 86 strings, $\sim 1\,\mathrm{km}^3$ of Antarctic ice, depths $\sim$1.45–2.45 km.
Detects Cherenkov light.



Winning solutions: everyone used transformers

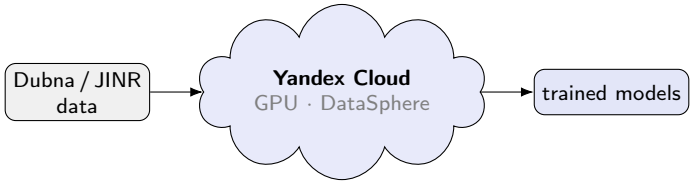
Place	Backbone	Seq. len
1 — "Tito"	EdgeConv + Transformer	up to 6000
2 — "IceMix"	Transformer	768
3 — "GPUs on Ice"	Transformer	3072

All top-3 are transformer-based. (arXiv:2310.15674)



1st place architecture

What we tried: setup

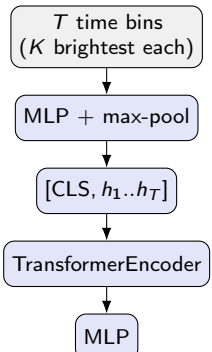


Compute

- Model experiments run on Yandex Cloud GPU compute.
- We used 408227 simulates muon events. PointNet mean angular error is 8.9.

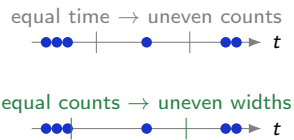
What we tried: VoxelTransformer

- Split the time interval into T equal-duration bins.
- Per bin i: MLP + max-pool $\rightarrow$ single vector h_i;
- $[\,\mathrm{CLS}, h_1, \ldots, h_T\,] \rightarrow$ TransformerEncoder $\rightarrow$ MLP.



What we tried: EqualCountVoxelTransformer

- Sort hits by time, split into T groups of $\sim N/T$ hits.
- To restore the lost time-width info, add two bin-level features per hit: the bin's time *center* and *width* $\Rightarrow$ **7 features** per hit.
- Rest of the architecture is identical.



What we tried: VoxelTransformer and EqualCountVoxelTransformer

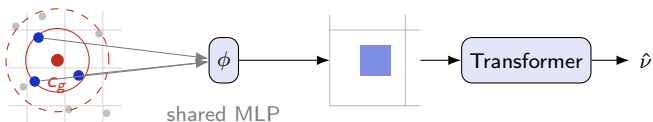
Hits	10-30	30-100	100-300	300-1000	1000+
Baseline	11.6	6.5	4.2	4.3	6.5
VoxelTransformer	13.0	6.4	3.8	3.6	5.9
EqualCountVoxelTransformer	**11.1**	**5.1**	**3.0**	**3.0**	**4.7**

Table 1: Median angular errors (lower is better).

What we tried: a regular space-time grid

Map the unstructured hit cloud onto a regular grid $G_x \times G_y \times G_z \times G_t$ over the event's bounding box.

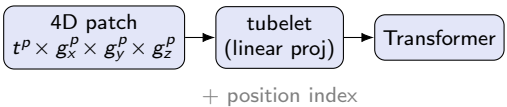
- Take the K nearest hits within radius r, pass their features through a shared MLP;
- Repeat in parallel over several radii $\{r_s\}$ to capture different track scales;
- Output: a dense 4D tensor $\rightarrow$ Transformer.



What we tried: a regular space-time grid

Tokenize the straightened grid with space-time patches.

- Tubelet = small spatial-time window;
- Each tubelet is position-indexed, fed as a sequence to Transformer.



Hits	10-30	30-100	100-300	300-1000	1000+
Baseline	11.6	6.5	4.2	4.3	6.5
VoxelTransformer	13.0	6.4	3.8	3.6	5.9
EqualCountVoxelTransformer	11.1	5.1	3.0	3.0	4.7
Reg-Transformer	**11.0**	**4.8**	**2.5**	**2.3**	**3.1**

Table 2: Median angular errors (lower is better).

What we tried: HitTransformer

Drop binning entirely, each hit is a token.

- Keep the N_{max} brightest hits and order them by time.
- Each hit i: 5 features $\rightarrow$ MLP $\rightarrow$ h_i.
- TransformerEncoder over all $N{+}1$ tokens $\rightarrow$ MLP $\rightarrow$ direction.

Hits	10-30	30-100	100-300	300-1000	1000+
Reg-Transformer	11.0	4.8	2.5	2.3	3.1
HitTransformer	**8.5**	**3.0**	**1.6**	**1.6**	**2.7**

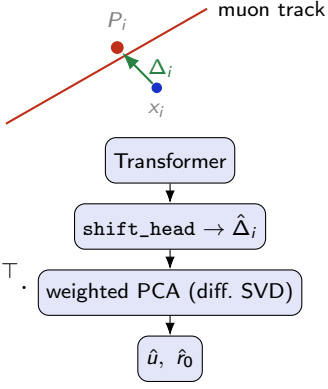
Table 3: Median angular errors (lower is better).

What we tried: MultitaskHit

For each hit x_i, there is an emission point P_i on the track, where the detected photon left the muon. Displacement $\Delta_i = P_i - x_i$ is a target.

- A `shift_head` predicts $\hat{\Delta}_i$ per token;
- A differentiable weighted PCA recovers the track:

$$\hat{r}_0 = \frac{\sum_i w_i \hat{P}_i}{\sum_i w_i}, \quad \hat{u} = \text{top-eigvec} \sum_i w_i (\hat{P}_i - \hat{r}_0)(\hat{P}_i - \hat{r}_0)^\top.$$



$$\text{Loss} \quad \mathcal{L} = \alpha\big(1 - |\cos(\hat{u}, u_{\text{true}})|\big) + \beta \tfrac{1}{N} \sum_i \left\| \hat{\Delta}_i - \Delta_i \right\|^2 + \gamma \left\| \hat{r}_0 - r_0 \right\|^2.$$

What we tried: MultiTaskHit

Method	AngErr°	PosErr m
HitTransformer	5.12	24.50
MultitaskHit	**3.88**	**3.09**

Table 4: Median angular errors and median distance errors.

Hits	10-30	30-100	100-300	300-1000	1000+
HitTransformer	8.5	3.0	1.6	1.6	2.7
MultitaskHit	**7.2**	**2.0**	**0.9**	**0.8**	**1.6**

Table 5: Median angular errors (lower is better).

What we tried: probabilistic MultiTaskHit

Assume $\hat{\Delta}_i \sim \mathcal{N}(\Delta_i^*, \sigma_i^2 I)$ — the model also predicts σ_i^2 per hit. The regression term becomes a negative log-likelihood:

$$\mathcal{L} = \alpha\big(1 - |\cos(\hat{u}, u^*)|\big) + \beta \tfrac{1}{N} \sum_i \Big[\frac{\left\| \hat{\Delta}_i - \Delta_i^* \right\|^2}{2\sigma_i^2} + \tfrac{3}{2} \log \sigma_i^2 \Big] + \gamma \left\| \hat{r}_0 - r_0^* \right\|^2 .$$

Ablation — PCA weights (Gaussian)

Weights	AngErr°	PosErr m
uniform	2.91	2.13
phe $\log(1+q)$	2.78	1.91
mlpw	2.75	1.80

Ablation — noise distribution (phe weights)

Distribution	AngErr°
Normal	2.78
Student (ν=2)	2.60
Cauchy	2.65
Laplace	**2.57**

What we tried: probabilistic MultiTaskHit

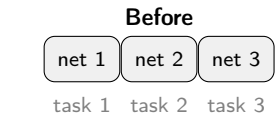
Method	AngErr°	PosErr m
MultitaskHit	3.88	3.09
MultiTaskHit_Prob	**2.54**	**1.60**

Table 6: Median angular errors and median distance errors.

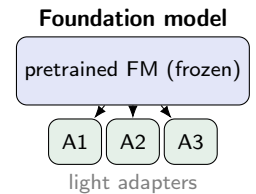
Hits	10-30	30-100	100-300	300-1000	1000+
MultitaskHit	7.2	2.0	0.9	0.8	1.6
MultitaskHit_Prob	**5.6**	**1.1**	**0.4**	**0.4**	**0.9**

Table 7: Median angular errors (lower is better).

From one-task nets to foundation models



train from scratch, once per task

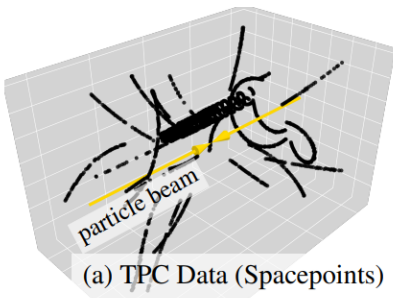


pretrain once, adapt cheaply

FM4NPP: setup

- Data: p+p in the sPHENIX Time Projection Chamber.
- Event = set of spacepoints (x, y, z, q).
- >11 million events; three downstream tasks: **track finding**, **PID**, **noise tagging**.

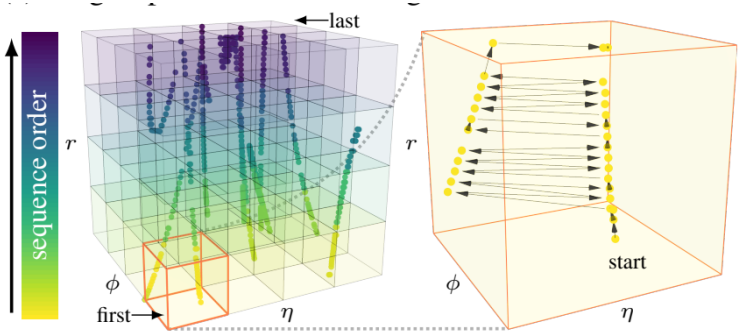
arXiv:2508.14087



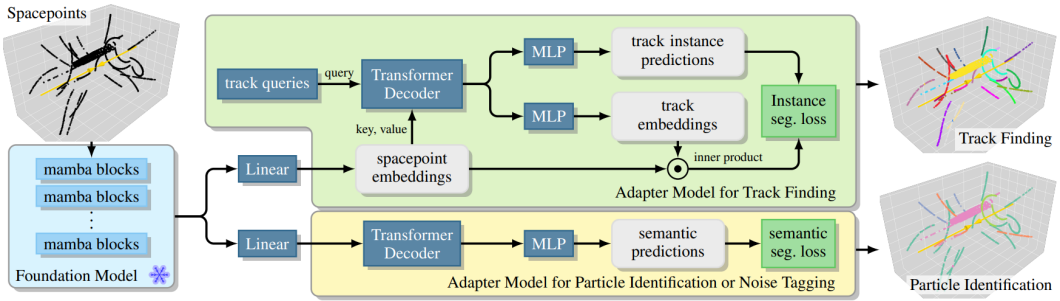
(a) TPC Data (Spacepoints)

FM4NPP: serialization

Hierarchical raster scan: boxes ordered center-outward, within a box sort by radius r.



FM4NPP: pretrain then adapt



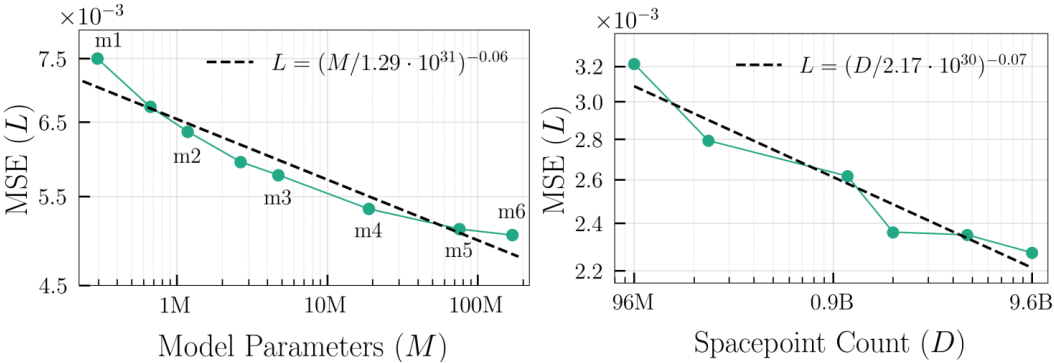
A big model is pretrained once; each task is a lightweight adapter.

FM4NPP: better than specialized models

		Track Finding		
model	#trnbl para.	ARI↑	efficiency↑	purity↑
`EggNet`	0.16M	0.7256	74.19%	<u>75.14%</u>
`Exa.TrkX`	3.86M	<u>0.8765</u>	<u>91.79%</u>	66.42%
`AdapterOnly`	2.39M	0.7243	78.01%	64.54%
`FM4NPP(m6)`	2.39M	**0.9448**	**96.08%**	**93.08%**

FM4NPP beats specialized models on each downstream task.

FM4NPP: scaling laws



FM4NPP scales with the growth of model parameters and training data.

Thank you!

Questions?



@BRAINLABORATORY



@LXSTSVUND