AI and Machine Learning Applications at the Near Detector of the T2K Experiment

Anaëlle Chalumeau
for the T2K collaboration & ND280 AI/ML working group

EPS-HEP Marseille, 07/07/2025
↳T16 - AI for HEP

Overview

- The T2K experiment and its Near Detector
- Overview of the AI/ML activities:

1. Momentum reconstruction and PID with a BDT (TMVA)
2. Global ND280 PID with BDT (XGBoost)
3. Identify EM shower with PointNet
4. 2D+3D CNNs for e/γ classification
5. PID with a Transformer
6. Other projects using ND280 data (Omnifold, Normalizing Flows)

The T2K experiment & its Near Detector

The T2K experiment & its Near Detector

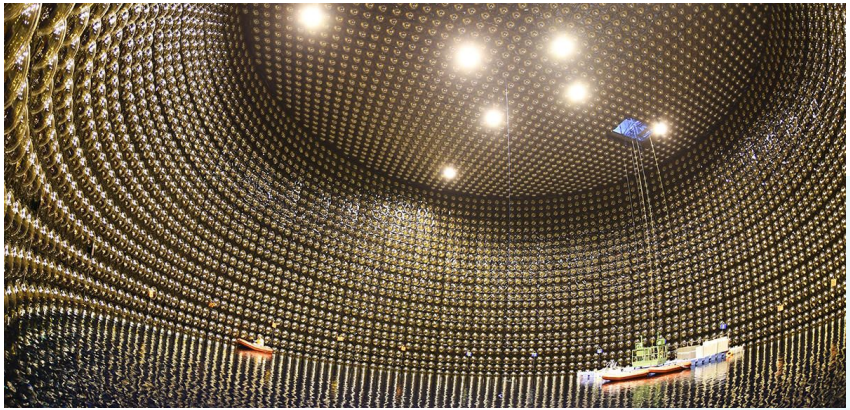
particle accelerator to create neutrino beam

The T2K experiment & its Near Detector

near detector: ND280

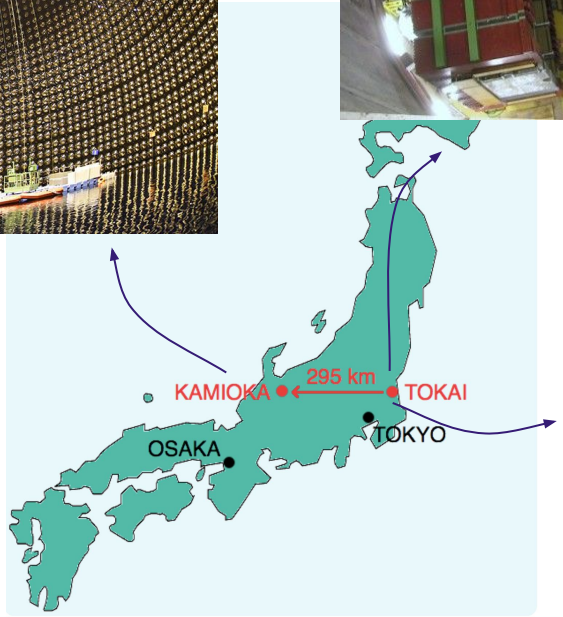
particle accelerator to create neutrino beam

The T2K experiment & its Near Detector



far detector: Super-Kamiokande

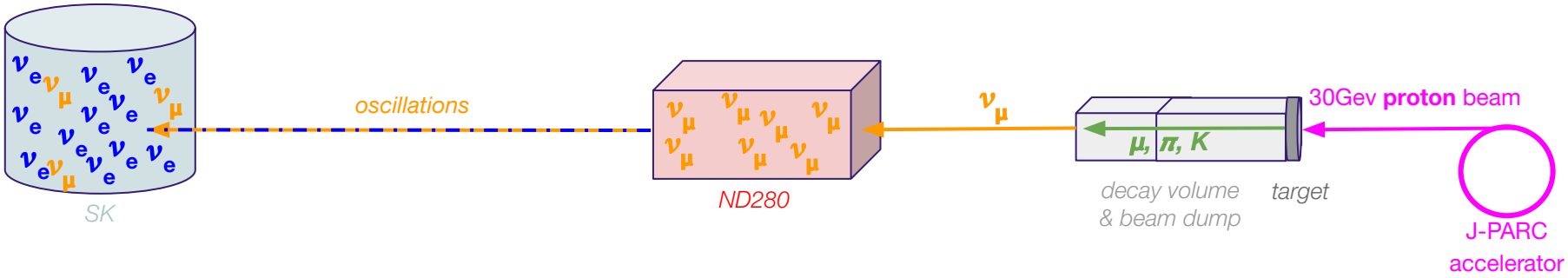
near detector: ND280

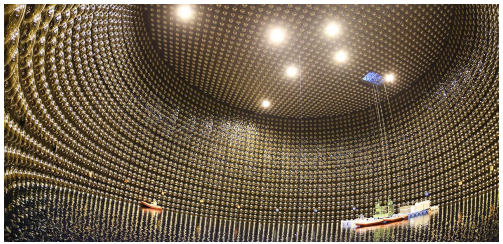


particle accelerator to create neutrino beam

The T2K experiment & its Near Detector

goal: measure neutrino oscillation parameters





Super-Kamiokande

295km

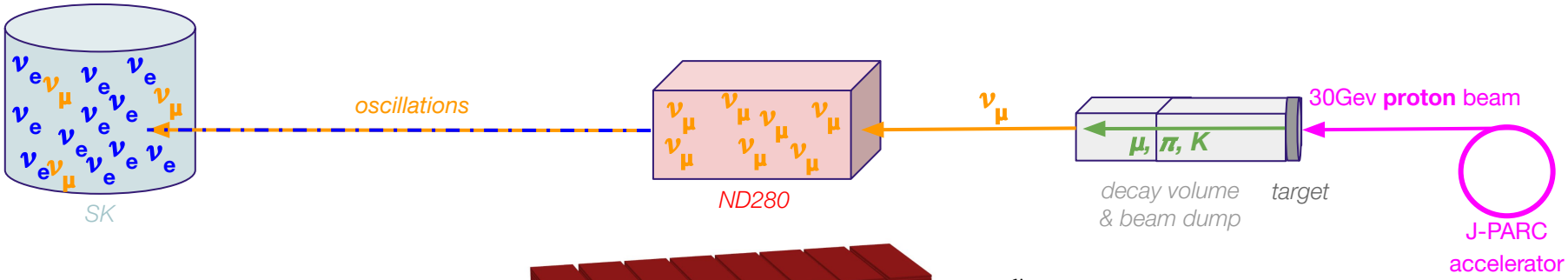
ND280

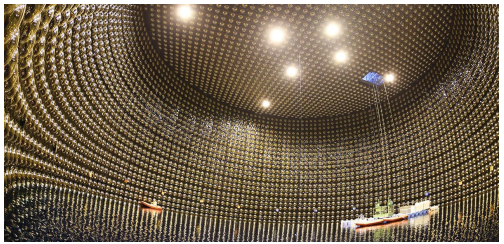
280m

neutrino beam

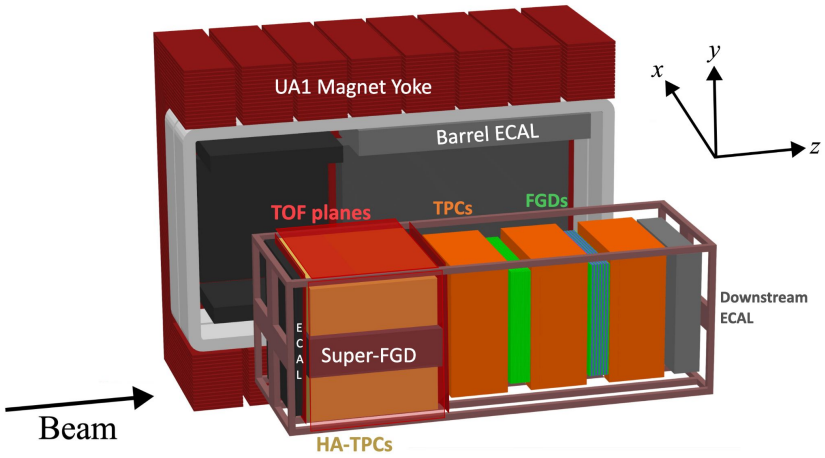
The T2K experiment & its Near Detector

goal: measure neutrino oscillation parameters



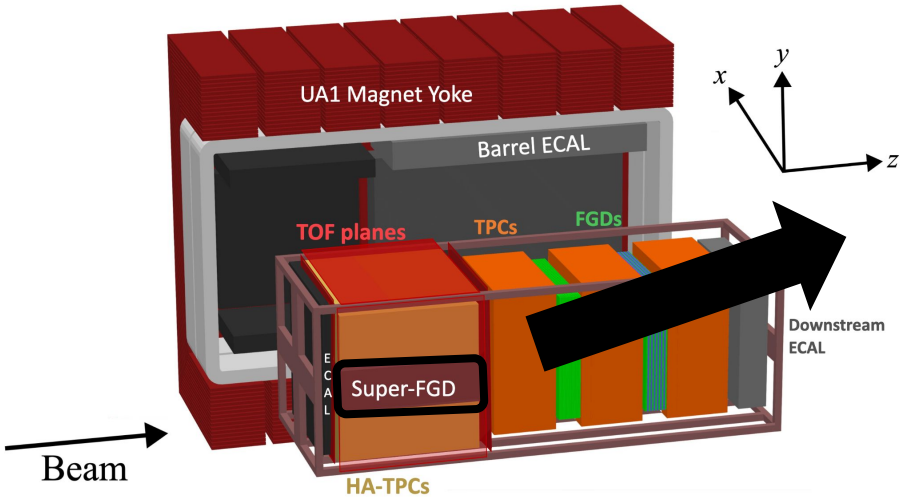


Super-Kamiokande



neutrino beam

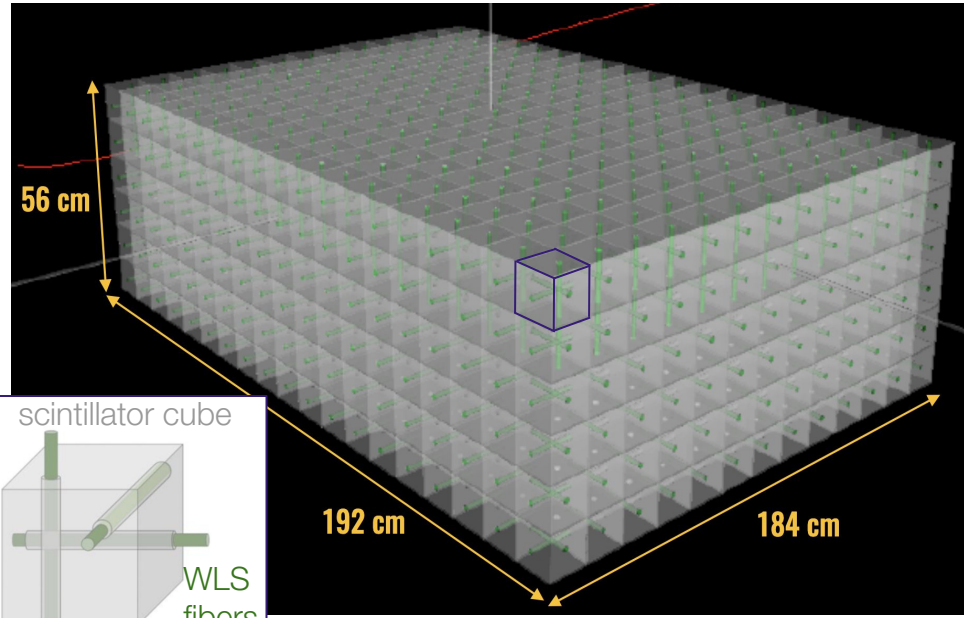
The T2K experiment & its Near Detector



ND280 Upgrade installed last year!

& data taking since mid-2024

The Super-FGD (SFGD): a Scintillator Detector



~2M cubes

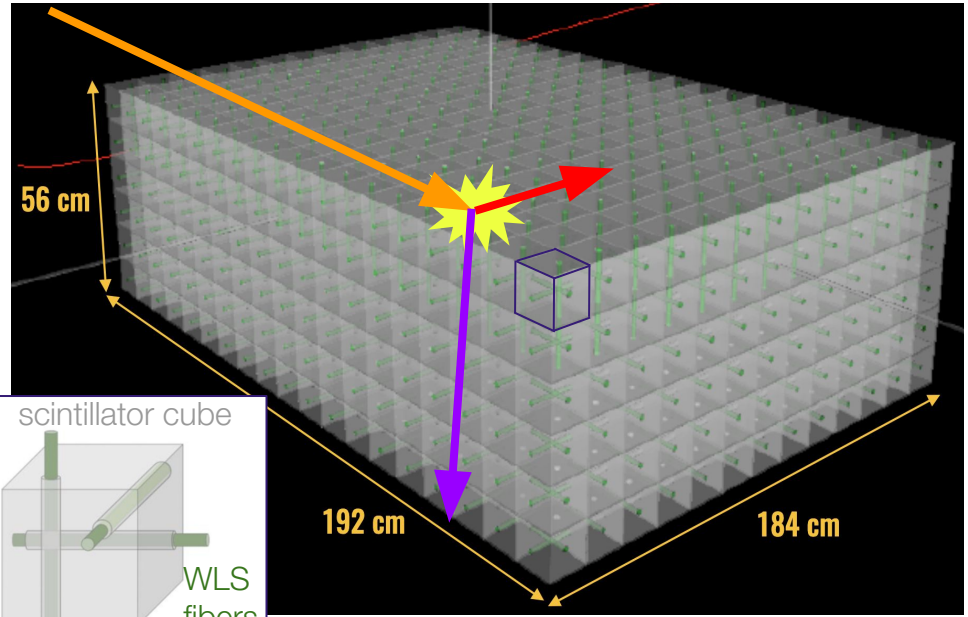
"Fine Grained Detector"

The T2K experiment & its Near Detector

New detector technology

⇒ need new tools to identify the particle types (PID) from neutrino interaction using charge deposition in the detector

The Super-FGD (SFGD): a Scintillator Detector



~2M cubes

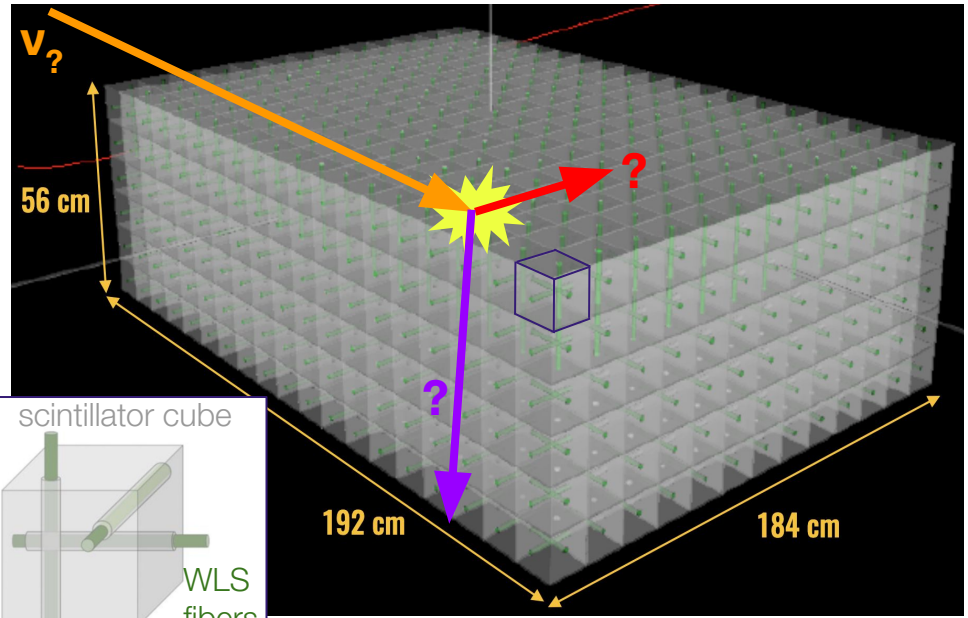
"Fine Grained Detector"

The T2K experiment & its Near Detector

New detector technology

⇒ need new tools to identify the particle types (PID) from neutrino interaction using charge deposition in the detector

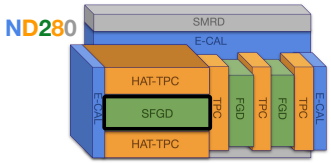
The Super-FGD (SFGD): a Scintillator Detector



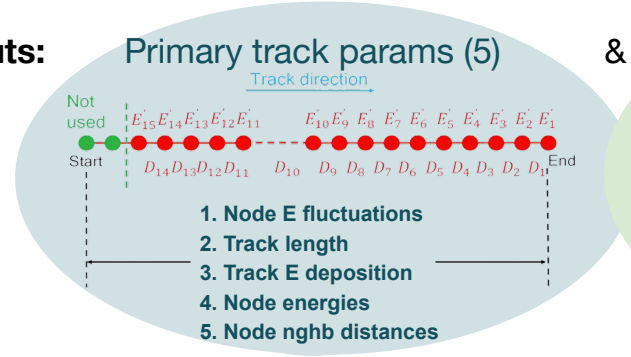
~2M cubes

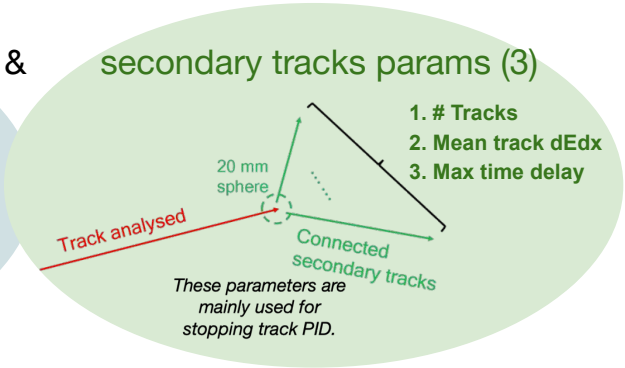
"Fine Grained Detector"

1. Momentum reconstruction & PID with BDT (TMVA)



Inputs: Primary track params (5) & secondary tracks params (3)





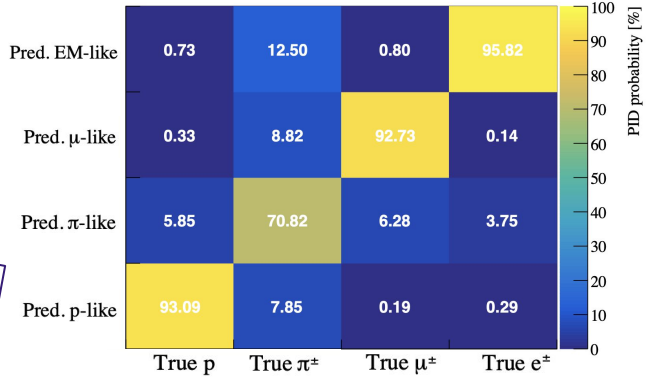
HPO: 5 to tune (done by hand):

Hyperparameter	Value
Decision tree number M	2000
Division point number K at each node	20
Decision tree maximum depth D_{max}	3
Shrinkage ν	0.05
Stochastic boosting fraction f	0.5

same for regression & classification

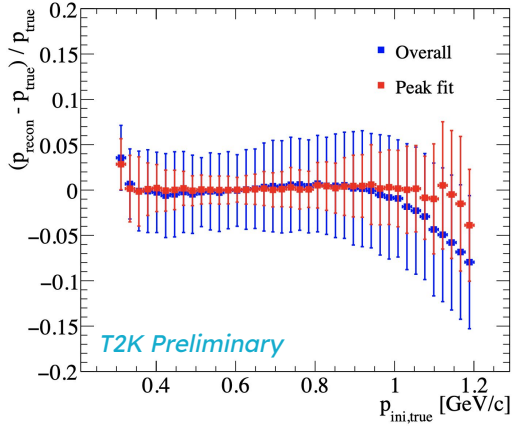
Training: - on particle-gun MC data (i.e. 1 particle /event): p, $\pi^{\pm}$, $\mu^{\pm}$, $e^{\pm}$ with 2M/type
- distinct classifiers & momentum regressors

Results: classification results

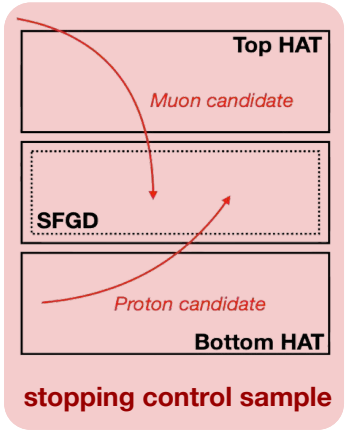


Fully integrated into T2K analysis level software

momentum resolution:

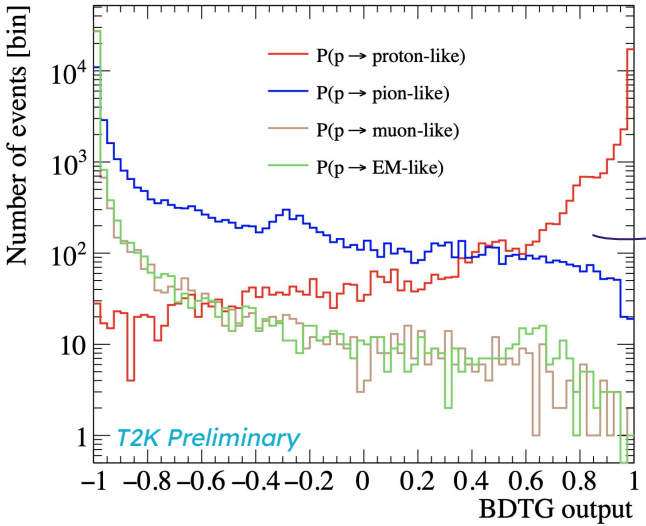


1. Momentum reconstruction & PID with BDT (TMVA)



BDT Systematics:

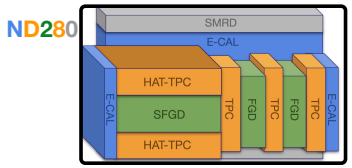
- Apply the BDT to **control samples** for both MC and data
- **Stopping control sample**: tracks that enter and stop in the SFGD
- Evaluate the difference and propagate it as a systematic uncertainty in analyses



+ same plot for control sample data

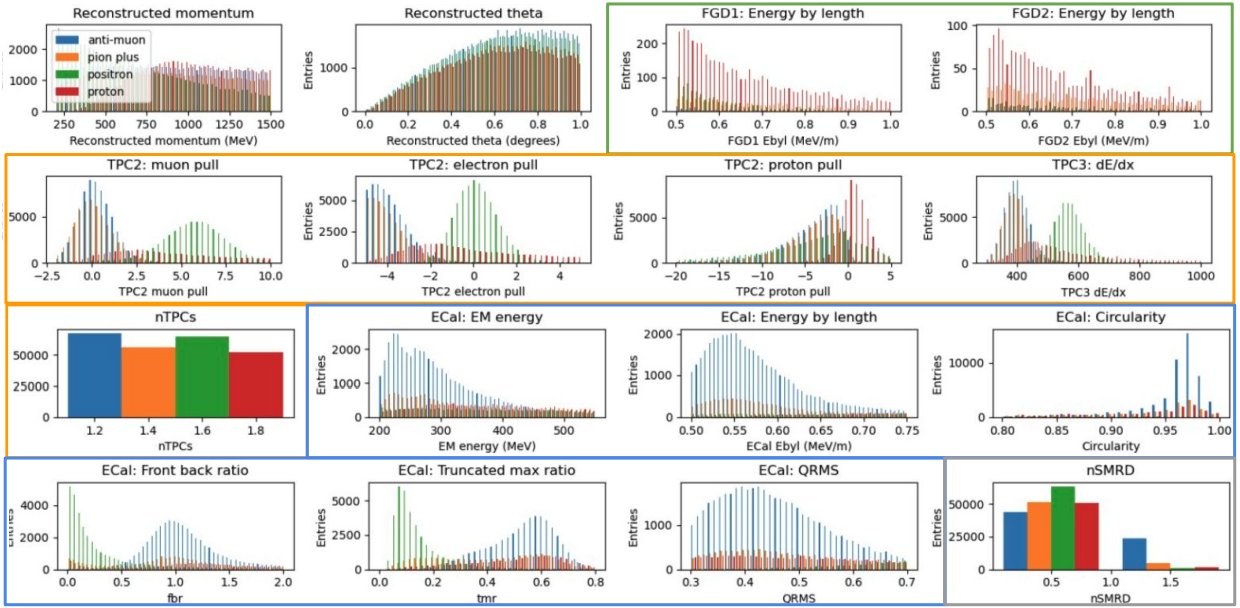
ratio between MC/data is propagated as a systematic uncertainty

2. Global ND280 PID with BDT (XGBoost)

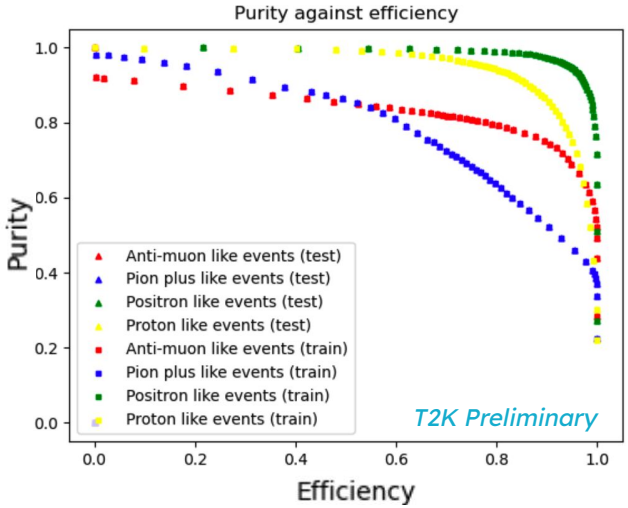


Goal: use inputs from many ND280 sub-detectors to get a global PID tool

Inputs: 16 variables from 4 of ND280 sub-detectors

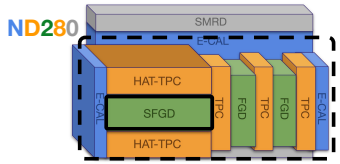


Results:

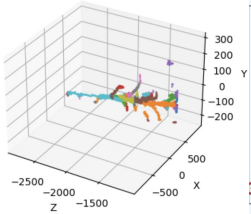


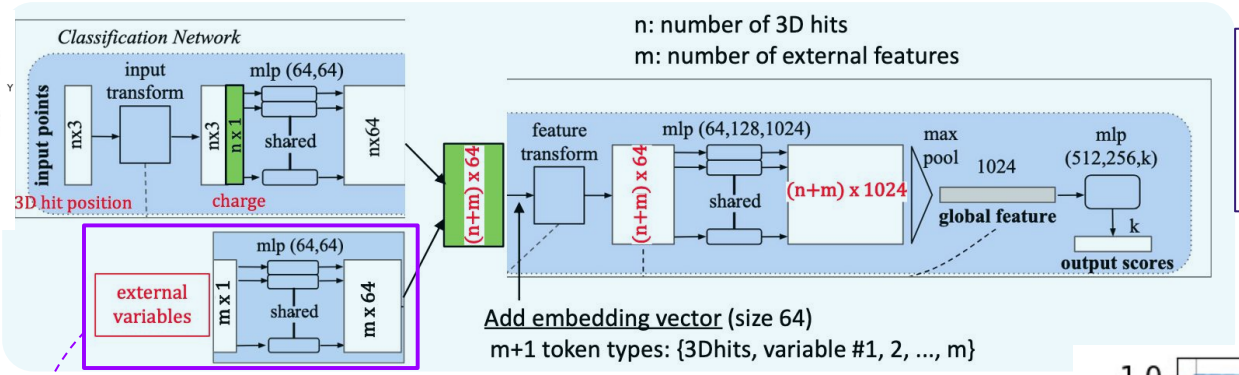
Training: - on particle-gun MC data: p, π^+, μ^+, e^+ with 250k events
- HPO

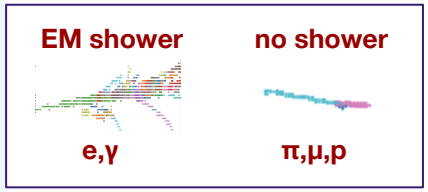
3. Identify EM shower with PointNet



Architecture: PointNet (DNN for 3D point cloud data) with modifications







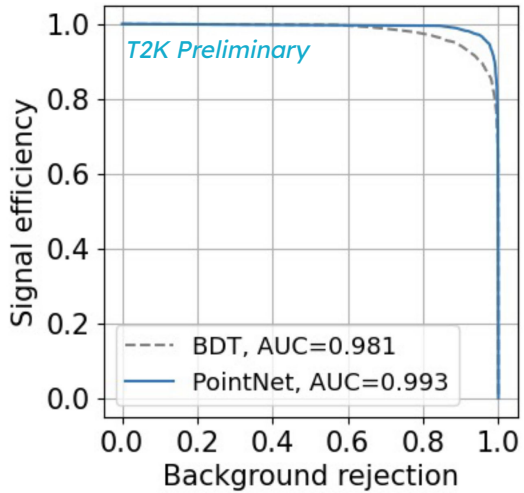
Goal: distinguish EM shower-like (e,γ) particle from non EM ones (μ,π,p)

Training data: pgun MC data of e^-, μ^- with 8000 patterns + data augmentation

External features: adding general feature of the event increase performances!

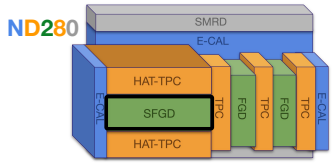
- SFGD variables (similar to BDT)
- TPC/HAT/ECAL variables
- size of the shower

Results:

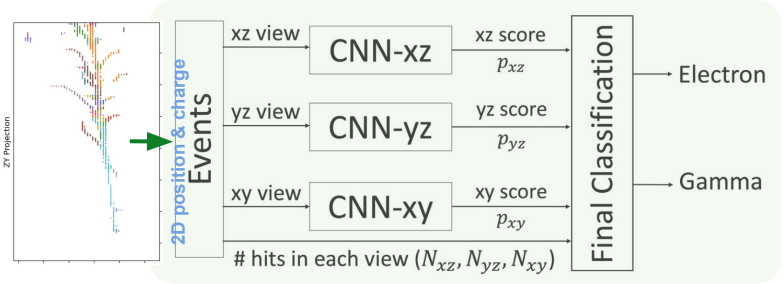


work of Hikaru Tanigawa

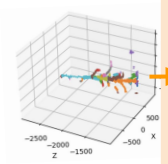
4. 2D+3D CNNs for e/γ classification



Architecture: experimental combination of 2D CNNs *ResNet50* + sparse 3D CNN *SSCNN*

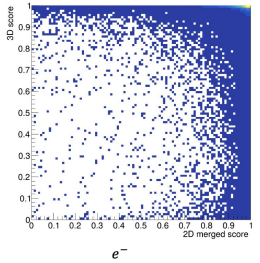


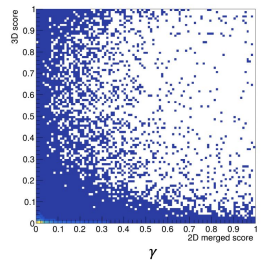
&



Sparse Submanifold Convolutional Network

$$\Rightarrow (final\ score) = \frac{(2D\ score) + (3D\ score)}{2}$$





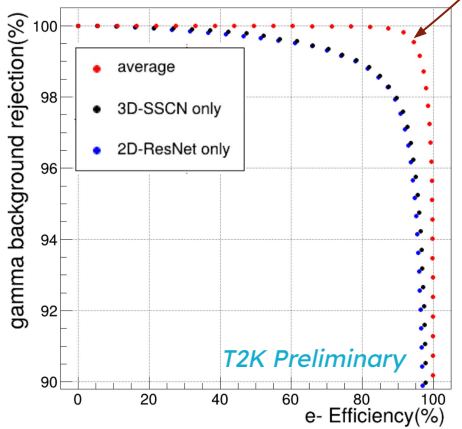
2D vs 3D scores: not much correlations

Training: pgun MC data of e^- and γ with 400k training event

Results:

model	e^- efficiency	γ rejection
3D-SSCN	95.5	95.3
3D-ResNet	94.9	95.8
2D-ResNet	95.3	94.4
3DSS+3DRes	96.1	96.4
3DSS+2DRes	**98.3**	**97.9**

T2K Preliminary

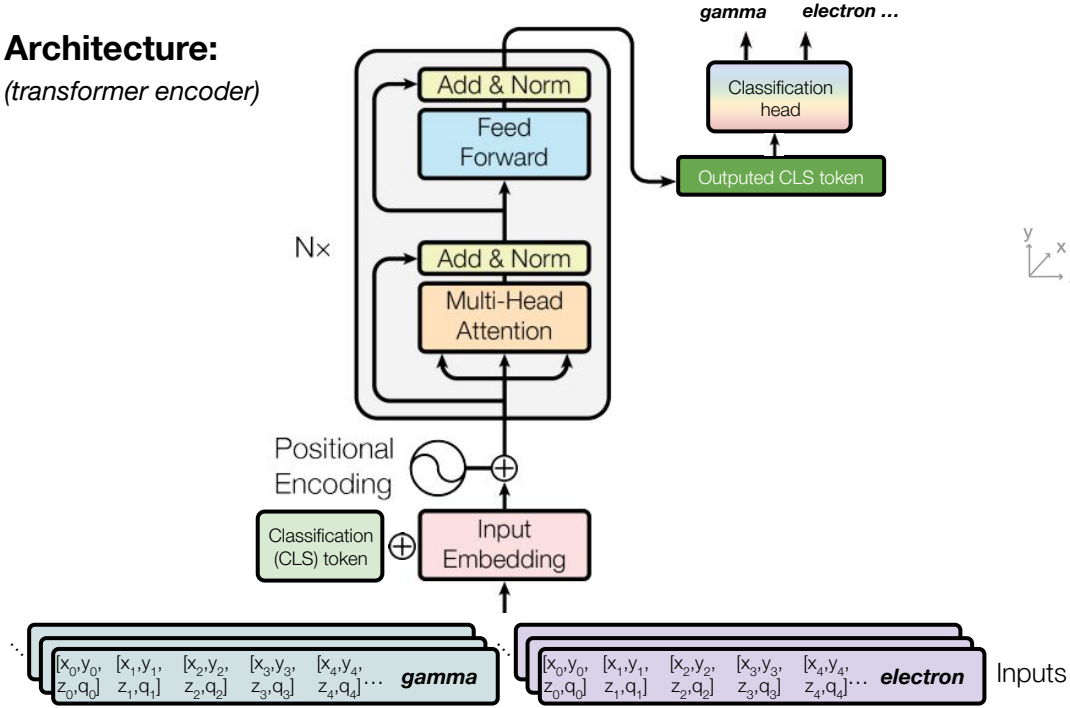


e/γ events looks similar because of γ→e^+e^- creating similar shower of particles
⇒ **challenge to distinguish them**

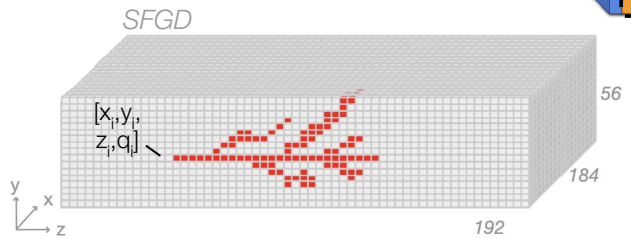
5. PID with a Transformer

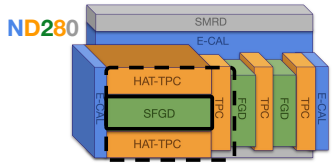
Architecture:

(transformer encoder)



Inputs:

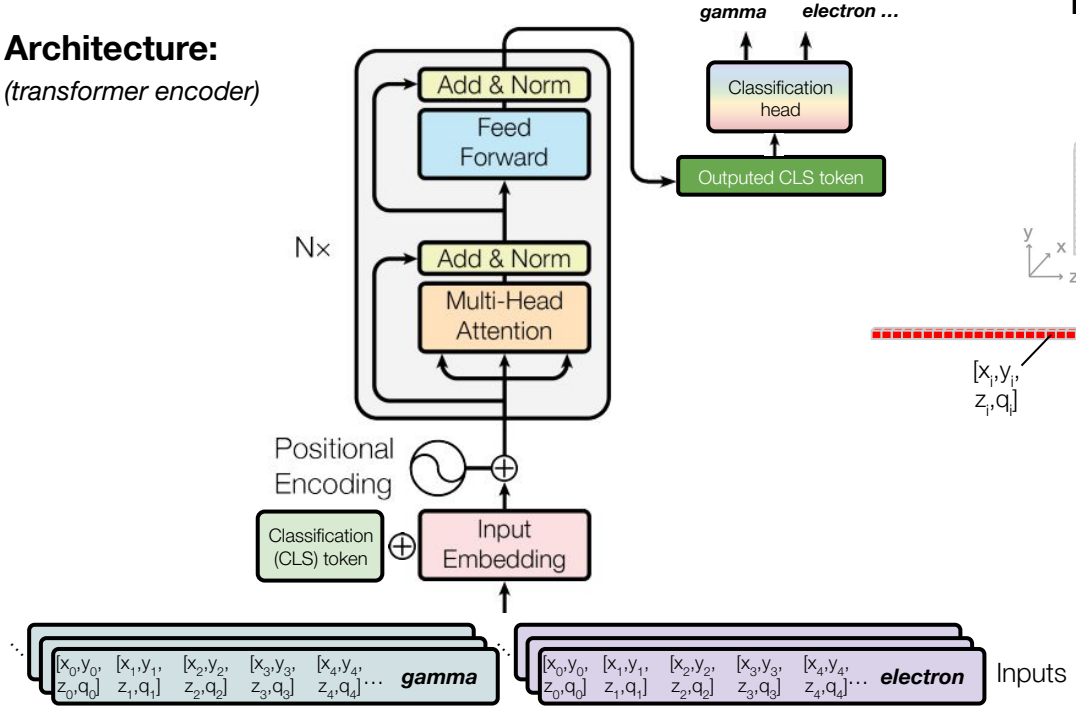




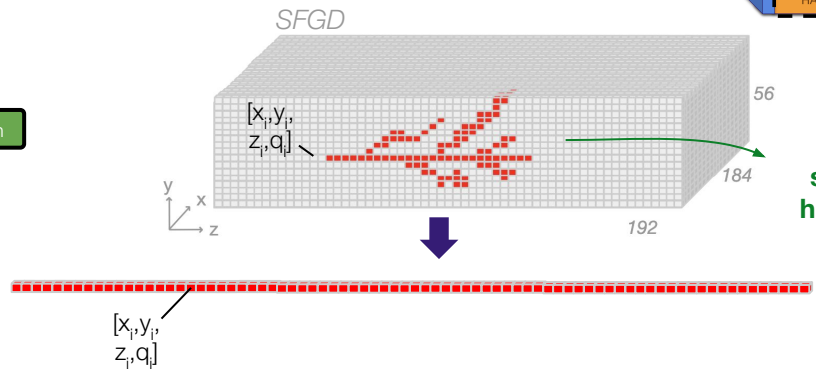
5. PID with a Transformer

Architecture:

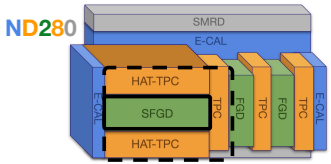
(transformer encoder)



Inputs:



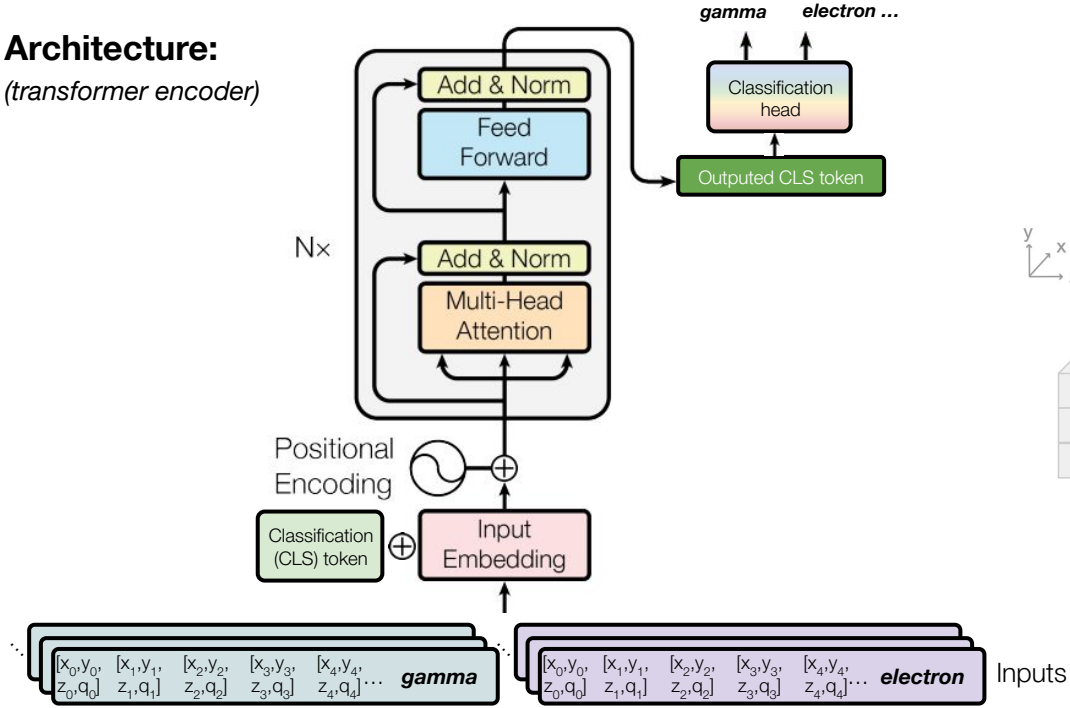
Too long sequences of hits: use Vision Transformer principle



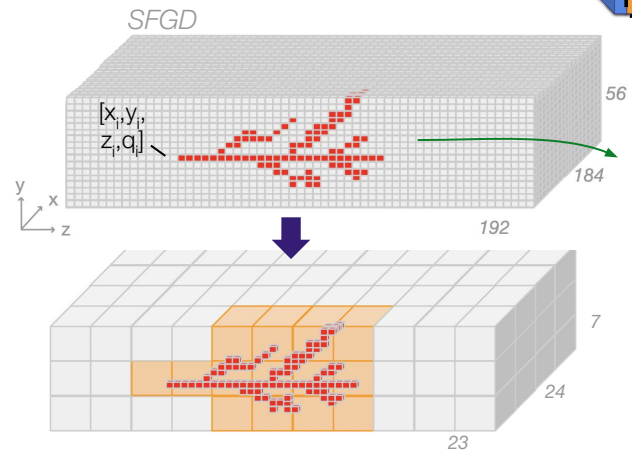
5. PID with a Transformer

Architecture:

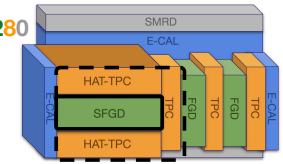
(transformer encoder)



Inputs:



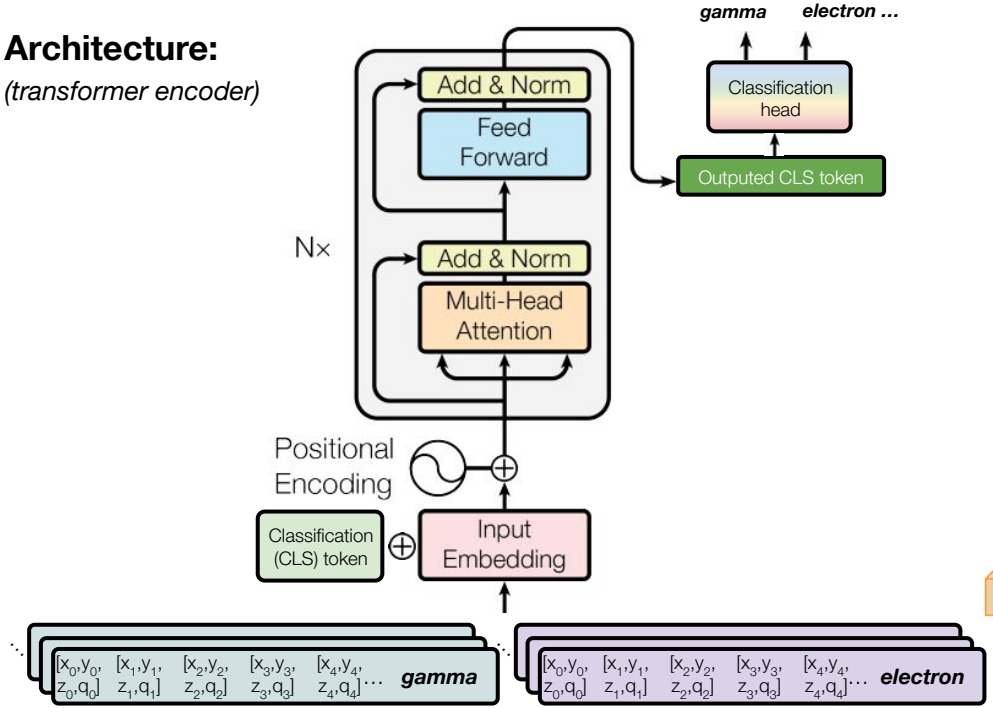
Too long sequences of hits: use Vision Transformer principle



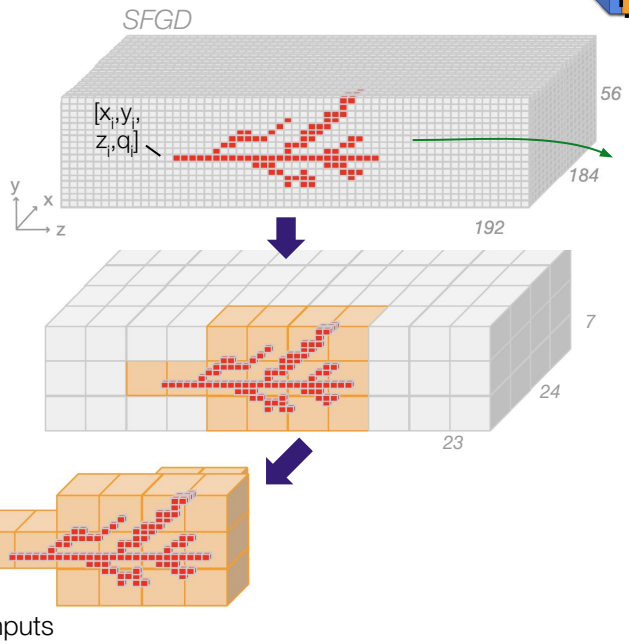
5. PID with a Transformer

Architecture:

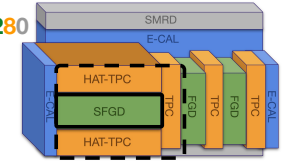
(transformer encoder)



Inputs:



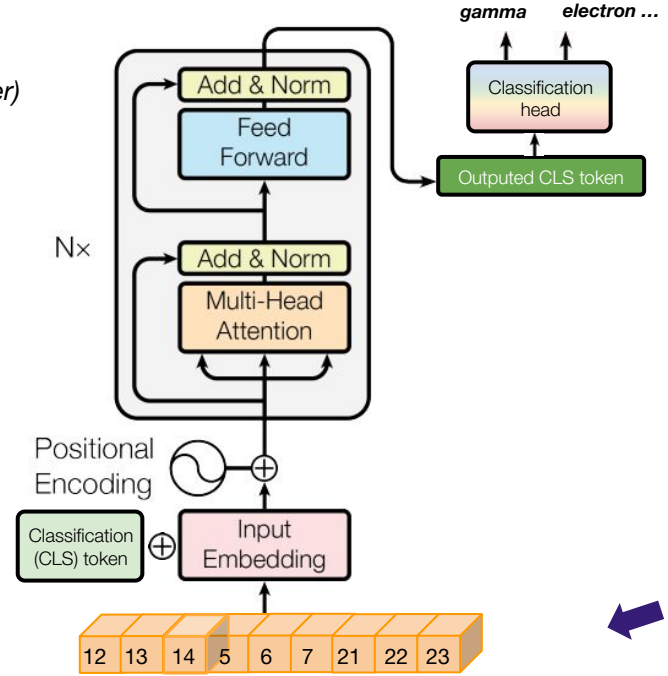
Too long sequences of hits: use Vision Transformer principle



5. PID with a Transformer

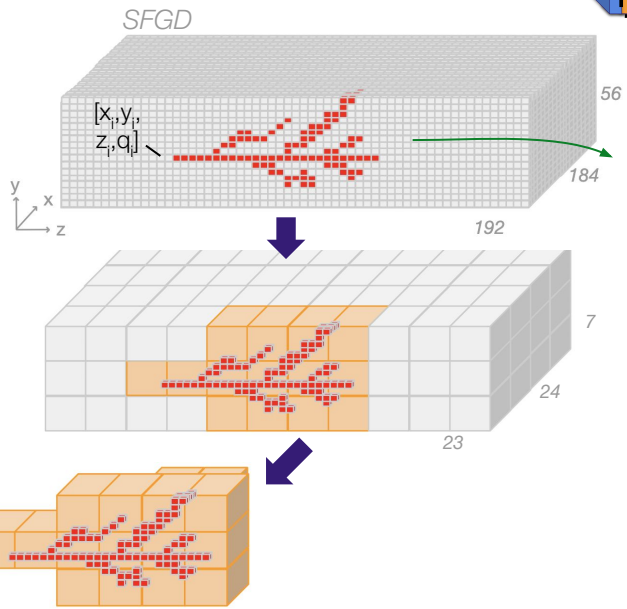
Architecture:

(transformer encoder)

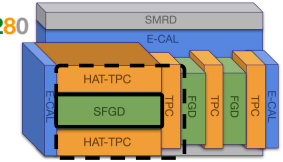


positional encoding of the parent cubes in the parent grid

Inputs:



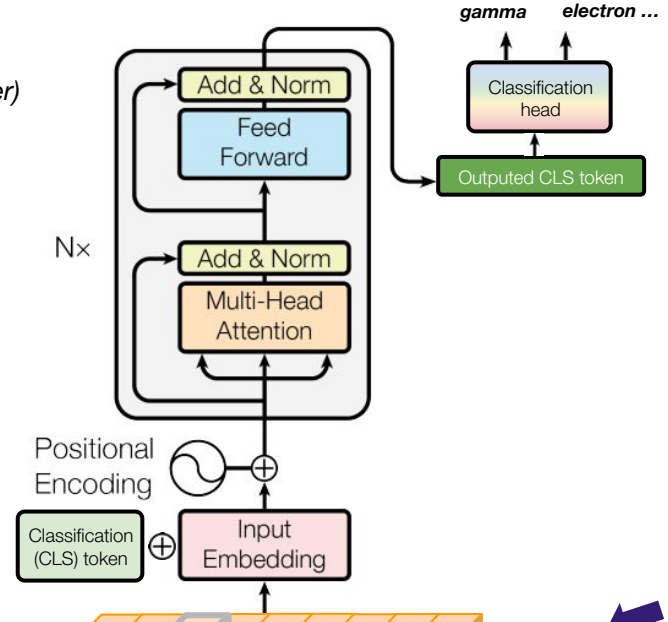
Too long sequences of hits: use Vision Transformer principle



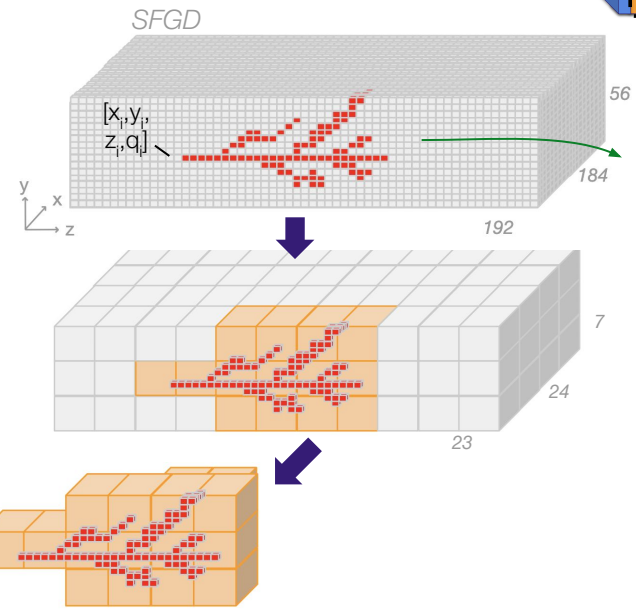
5. PID with a Transformer

Architecture:

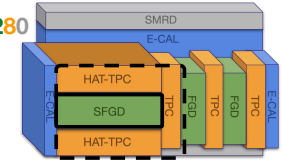
(transformer encoder)



Inputs:

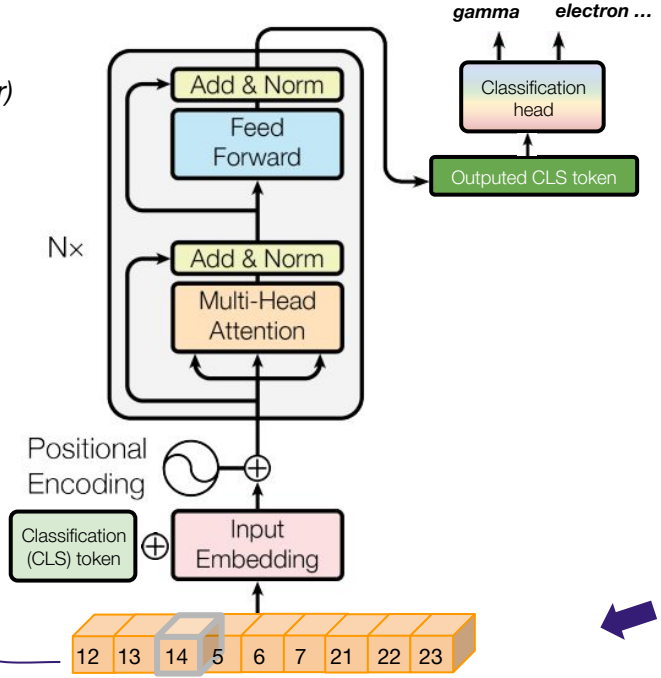


Too long sequences of hits: use Vision Transformer principle

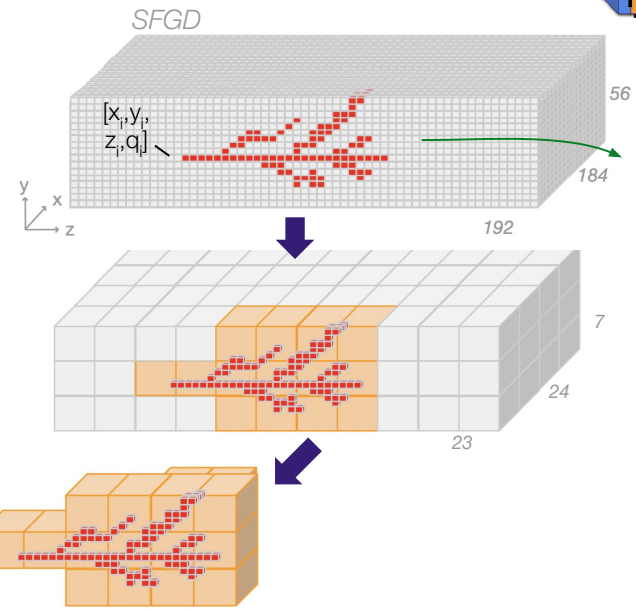


5. PID with a Transformer

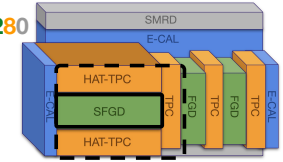
Architecture:
(transformer encoder)



Inputs:



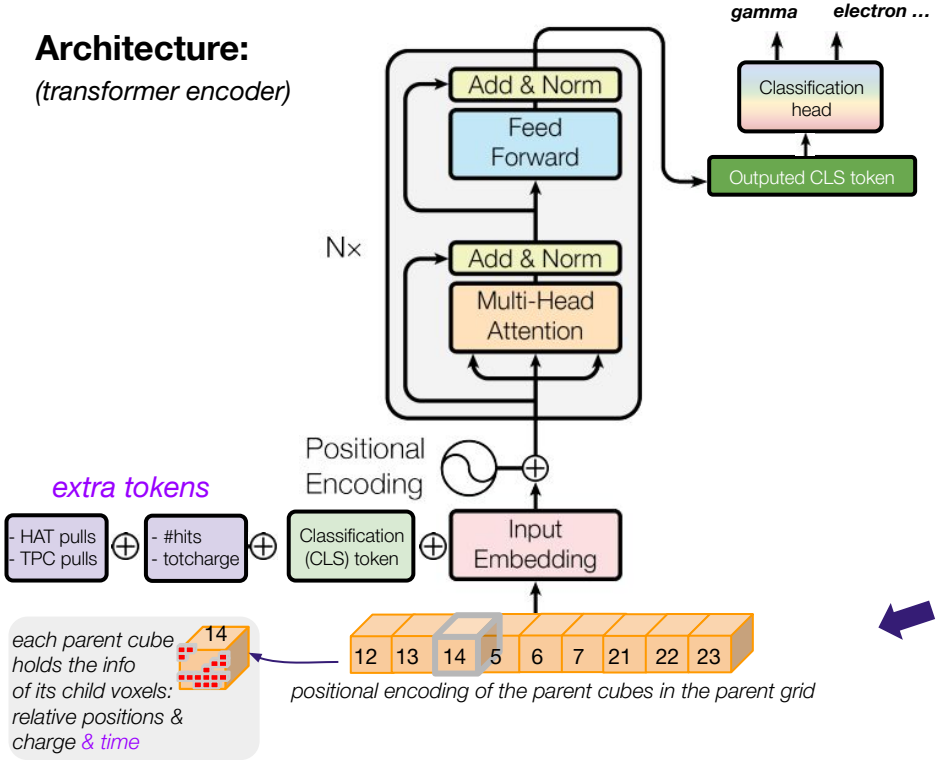
Too long sequences of hits: use Vision Transformer principle



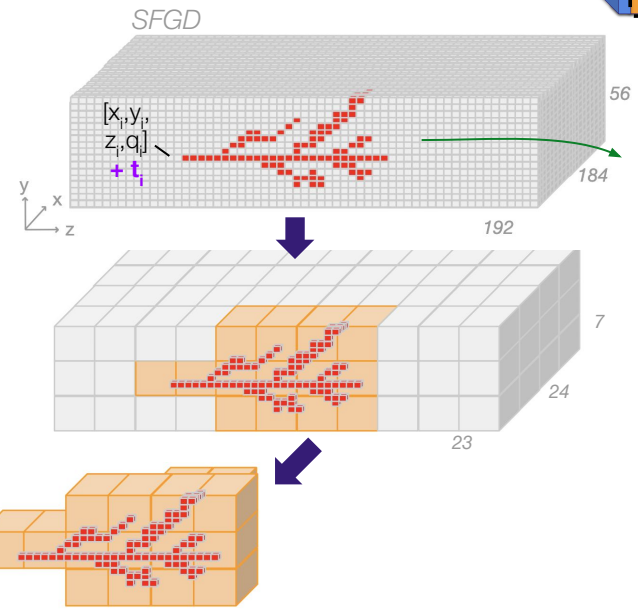
Training: a) pgun: e^-, γ, μ^-, π^-, p, 200k/part, only SFGD data

5. PID with a Transformer

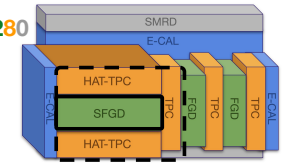
Architecture:
(transformer encoder)



Inputs:



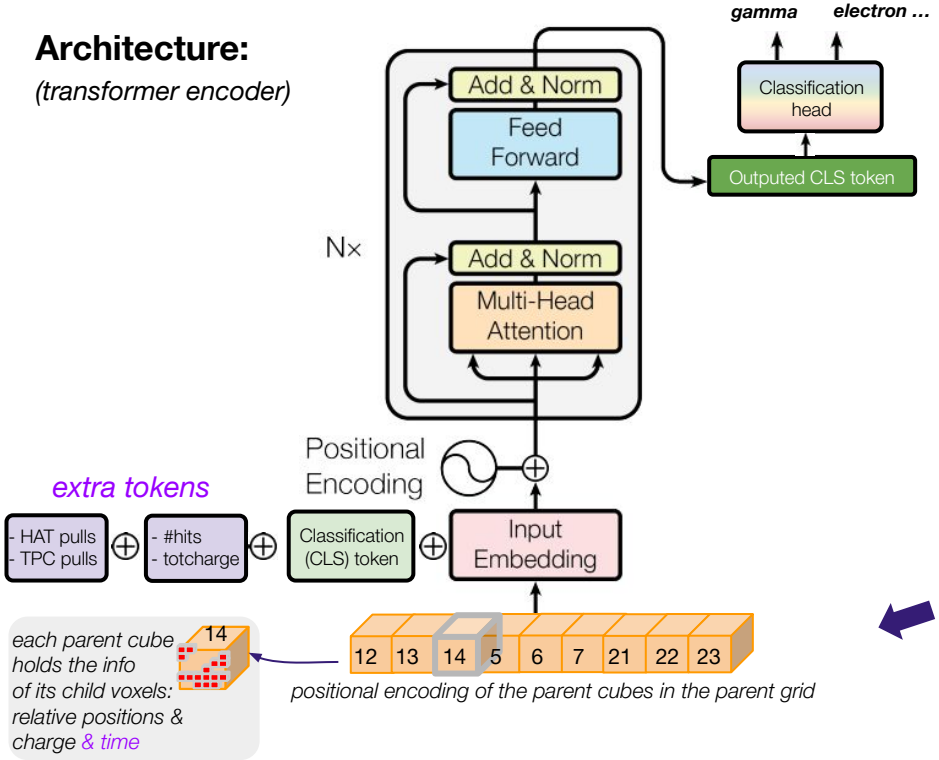
Too long sequences of hits: use Vision Transformer principle



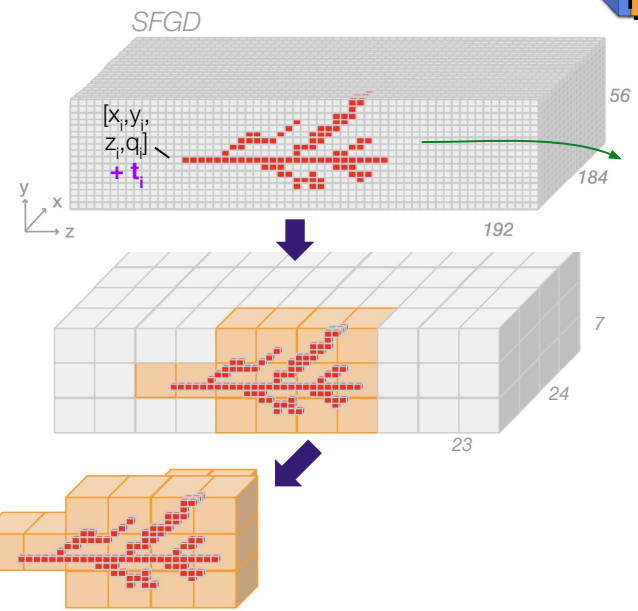
Training: a) pgun: e^-, γ, μ^-, π^-, p, 200k/part, only SFGD data
b) pgun: $e^{+/-}$, γ, $\mu^{+/-}$, $\pi^{+/-}$, p, 1M/part
+ time info + extra tokens (dEdX from nghb TPCs)

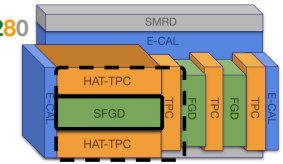
5. PID with a Transformer

Architecture:
(transformer encoder)



Inputs:





Too long sequences of hits: use Vision Transformer principle

- architecture converted in C++ (*Libtorch*)
- saved weights from training to be loaded in C++

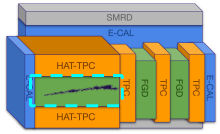
➡ 1rst complex ML model implemented in analysis soft!

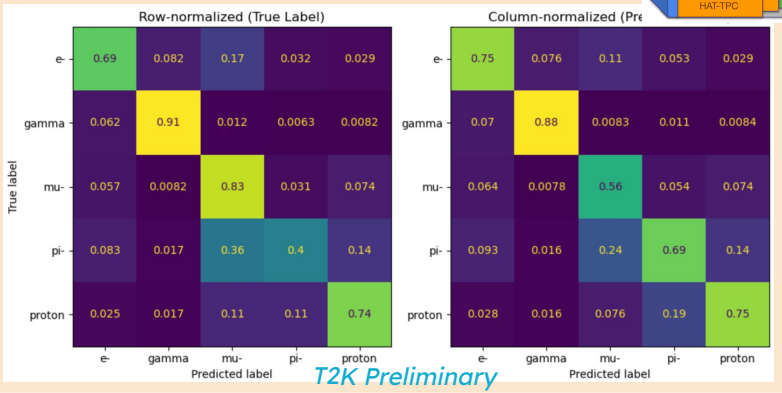
Training: a) pgun: e^-, γ, μ^-, π^-, p, 200k/part, only SFGD data
b) pgun: $e^{+/-}$, γ, $\mu^{+/-}$, $\pi^{+/-}$, p, 1M/part
+ time info + extra tokens (dEdX from nghb TPCs)

5. PID with a Transformer

Test:

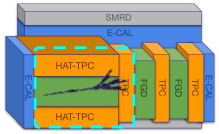
☺ *training a)*

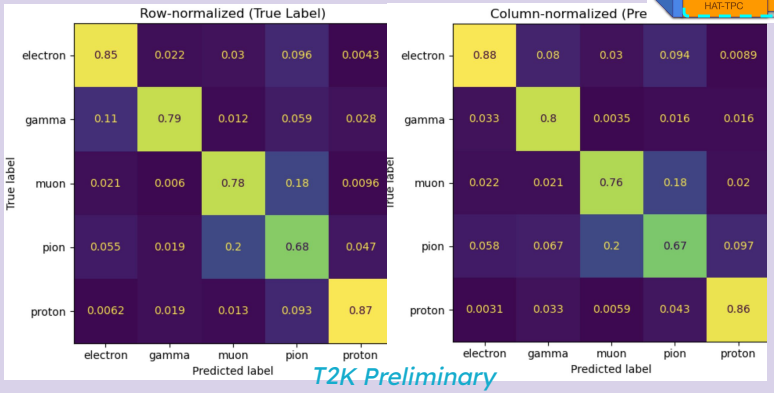




T2K Preliminary

☺ *training b)*



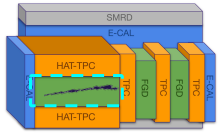


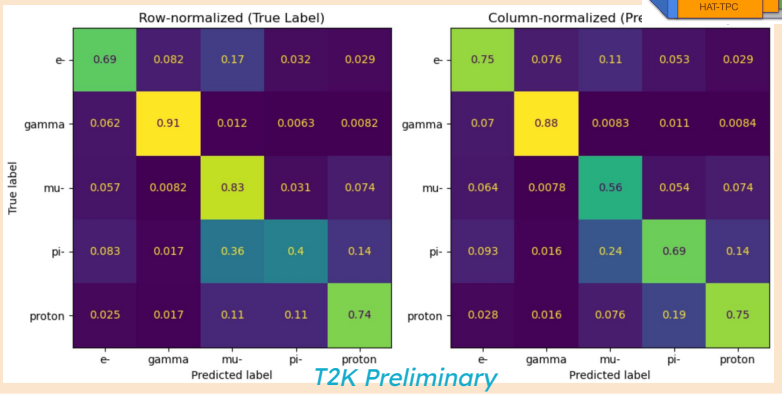
T2K Preliminary

5. PID with a Transformer

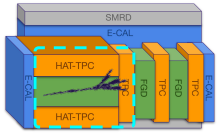
Test:

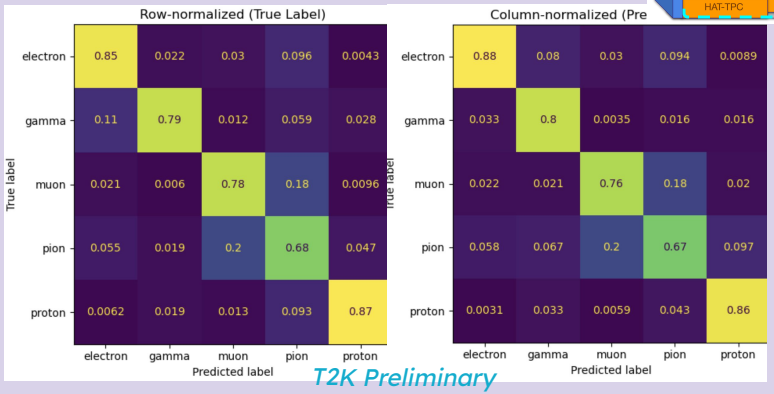
☺ *training a)*



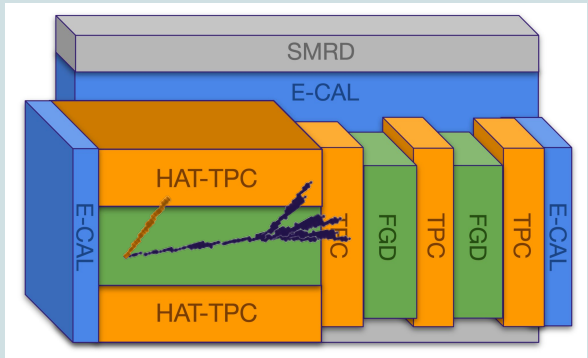


☺ *training b)*





Test on simulated neutrino interactions:

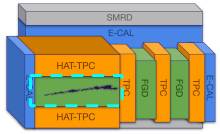


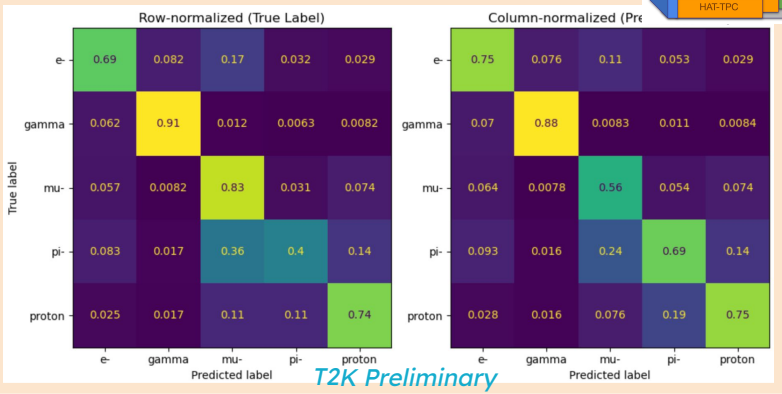
need good pattern recognition first!

5. PID with a Transformer

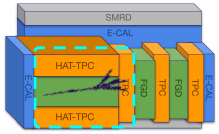
Test:

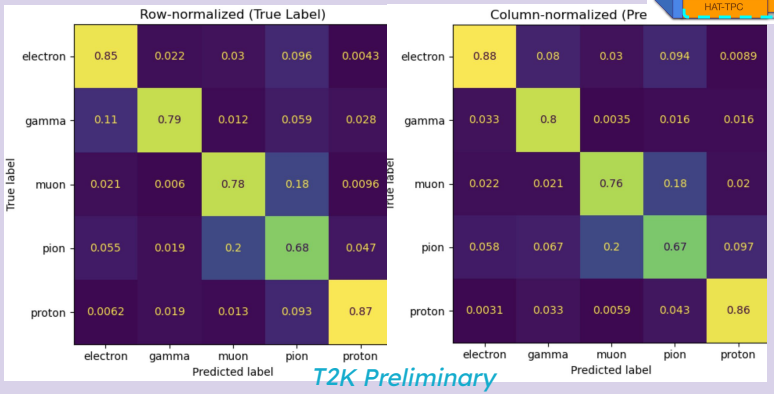
☺ *training a)*



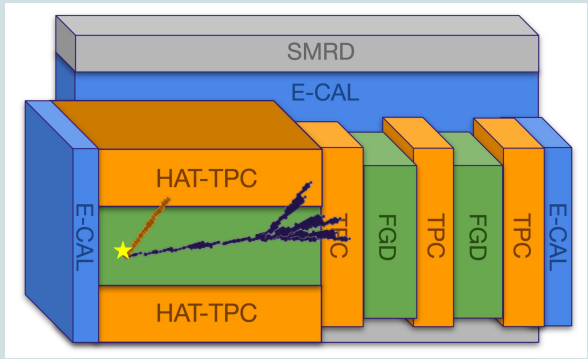


☺ *training b)*





Test on simulated neutrino interactions:

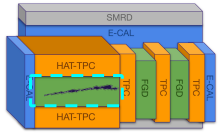


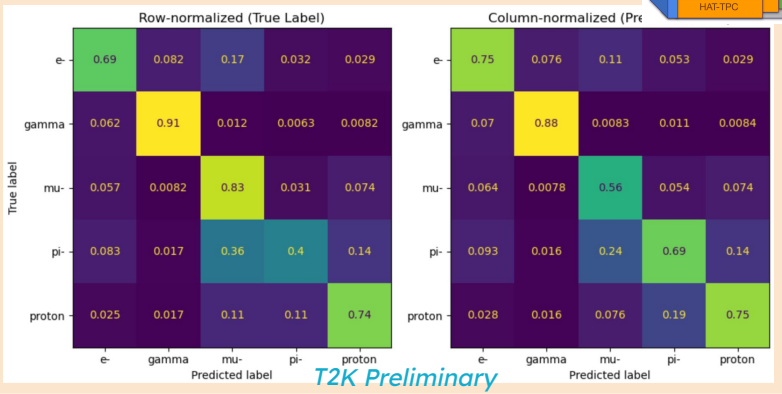
need good pattern recognition first!

5. PID with a Transformer

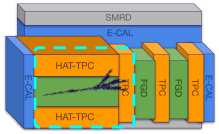
Test:

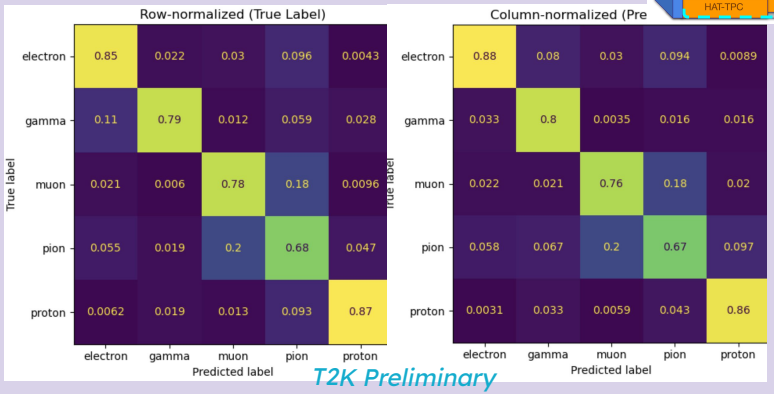
☺ *training a)*



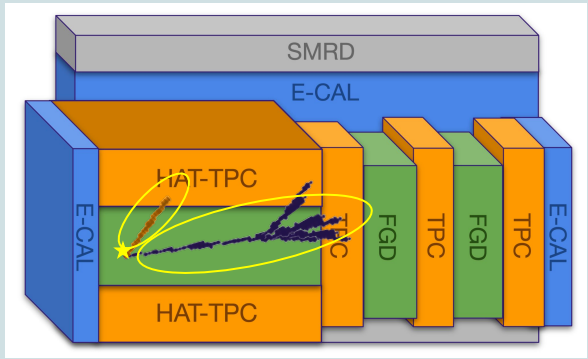


☺ *training b)*





Test on simulated neutrino interactions:

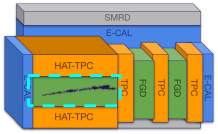


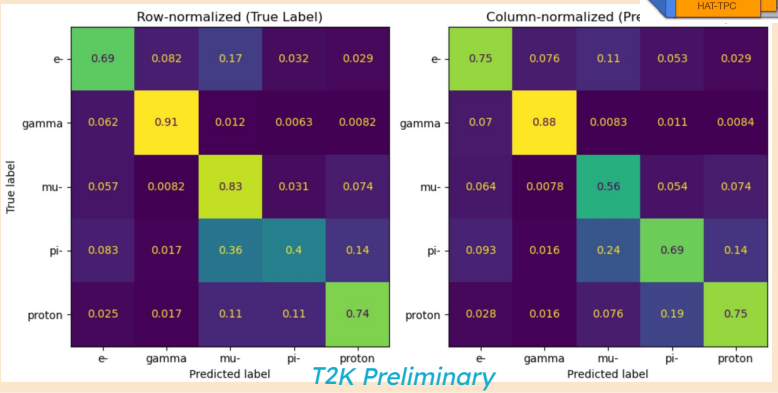
need good pattern recognition first!

5. PID with a Transformer

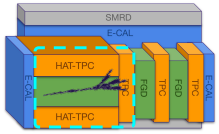
Test:

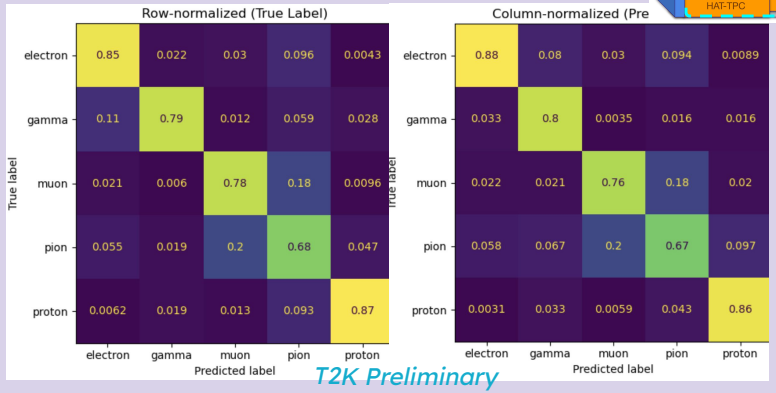
☺ *training a)*





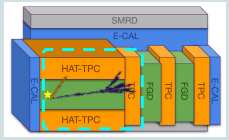
☺ *training b)*

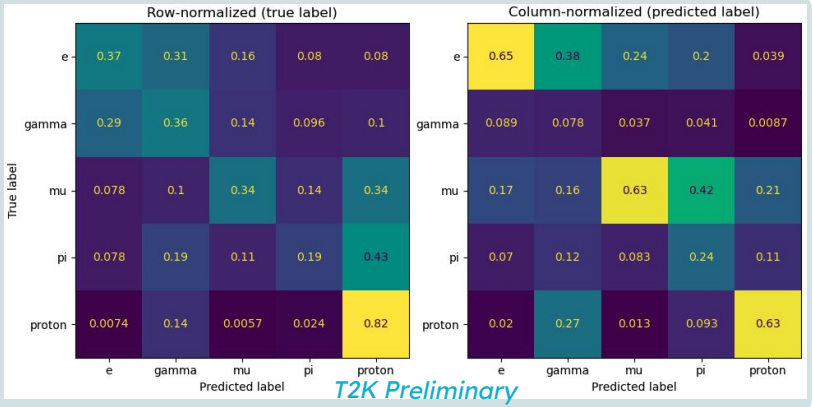




Test on simulated neutrino interactions:

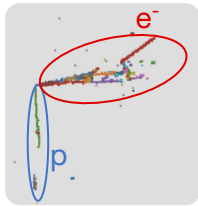
(training a)





On-going improvements:

- refine pattern recognition before PID (if bad, then model cannot succeed)
- test *training b)* on neutrino interactions
- add info from other detectors

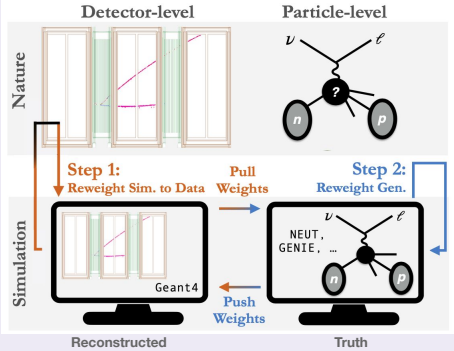


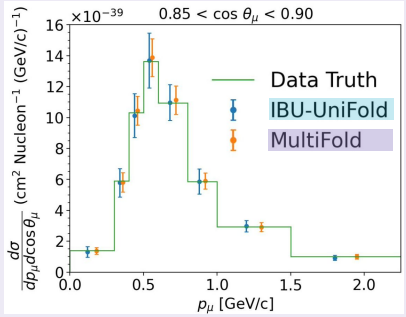
6. Other projects using ND280 data (Omnifold, Normalizing Flows)

Omnifold:

work of Roger Huang

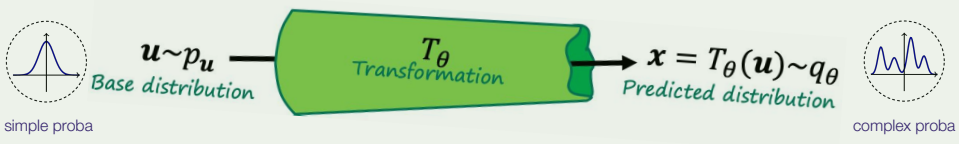
unbinned method to unfold ND280 data faster



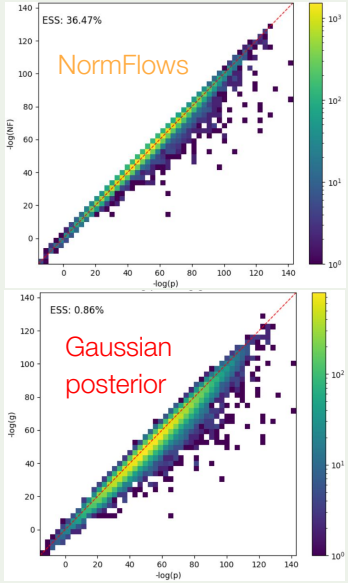


Normalizing flows:

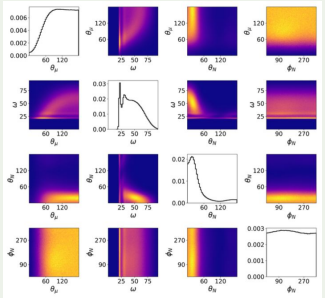
work of Mathias El Baz

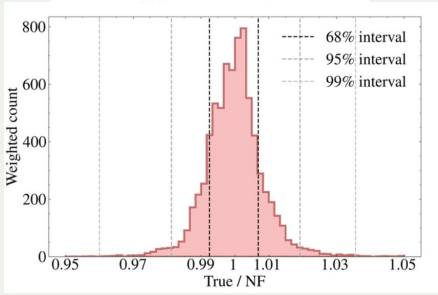


Modeling posterior systematics



Efficient CCQE cross-section sampling





Summary

- **Increasing number of ML projects** at T2K ND
- In all stages of the experiment: **more project in the analysis** part, then reconstruction then simulation
- Methods are **starting to be used in official softwares** (BDT, Transformer), paving the way for the other ones

- **Strengths:**
  - High accuracy in classification/regression tasks
  - Learns complex features from raw or high-dimensional data
  - Efficient at scale for large datasets (once trained)

- **Challenges**:
  - More friendly integration possibilities into C++
  - Systematic uncertainty: need robust strategies

- **Limitations**:
  - Sensitivity to simulation mismodeling → need careful validation on real data
  - Training variability (random seeds, small data) can cause instability in results
  - Interpretability of deep networks is limited

Thank you!

Back-up

3. Identify EM shower in SFGD: PointNet

External features:

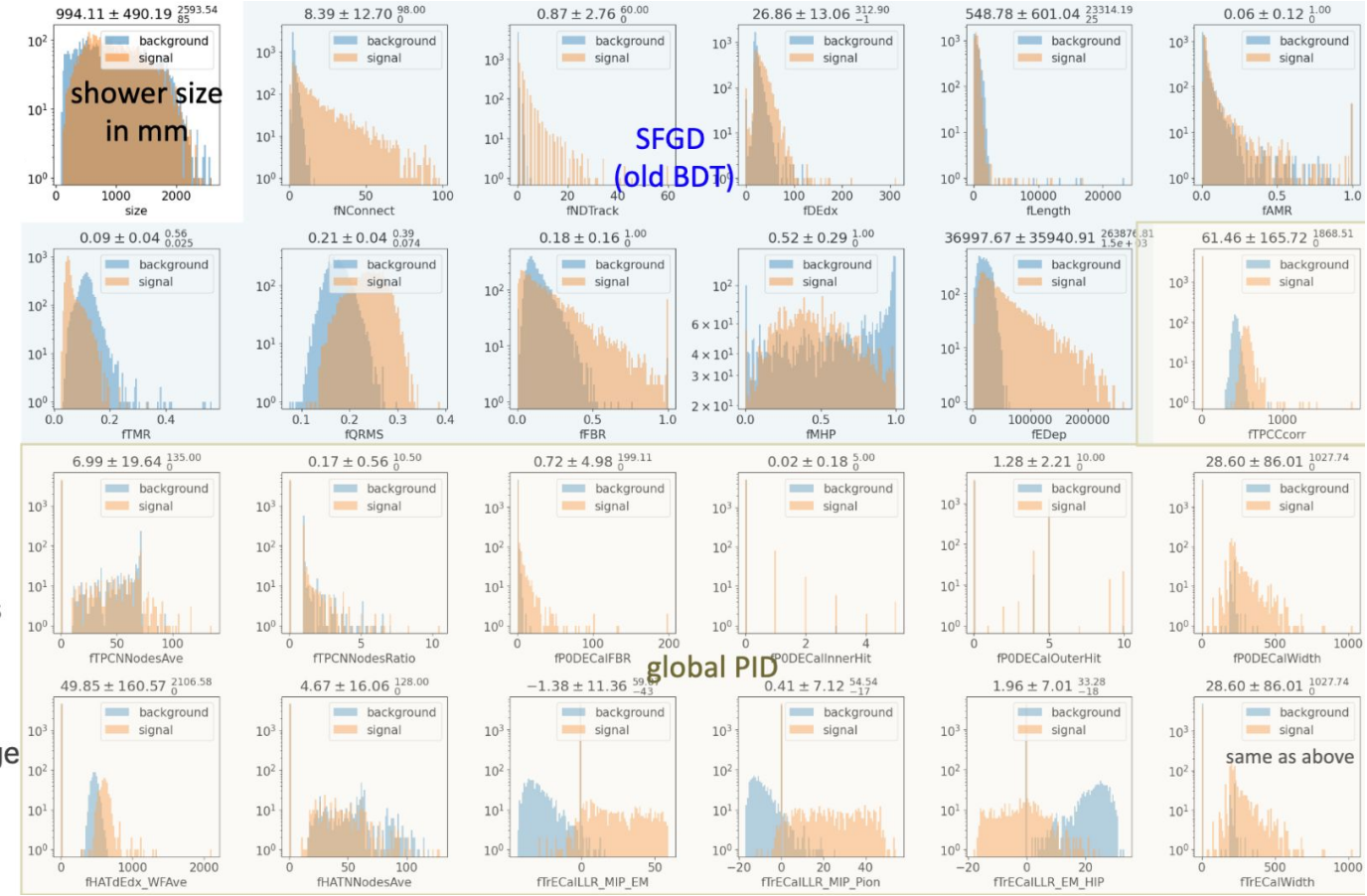
shower size
- added by HT

SFGD features: 10
- from old BDT

nonSFG features: 13
- from global PID
- mostly empty for the contained cones
  - 85-90% are “0”

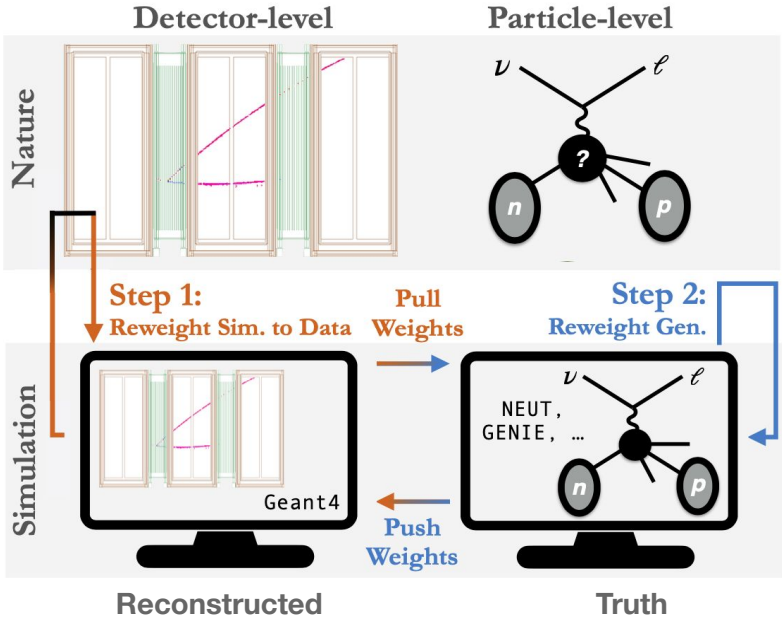
Distributes in wide range

→ Need scaling

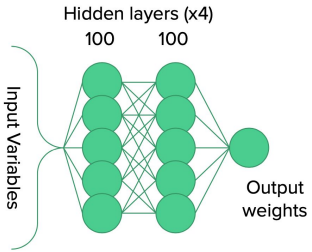


Unfolding of ND280 data: Omnifold

Working principle:



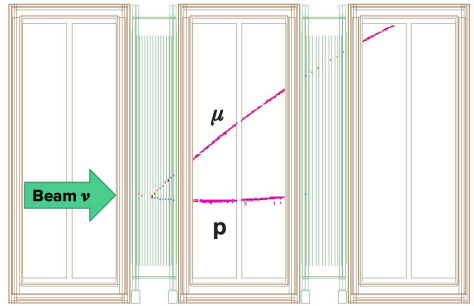
Archi:



1NVIDIA A100 on a NERSC Perlmutter node: takes < 30 min to run 15 OmniFold iterations on one set of data/MC

Data:

- 1.2M simulated ND280 evts ≃ 20k measured evts
- with π^+ and leading *p* kinematics



- *for test:* create fake dataset with a BeRPA-based modification to the true interaction rates

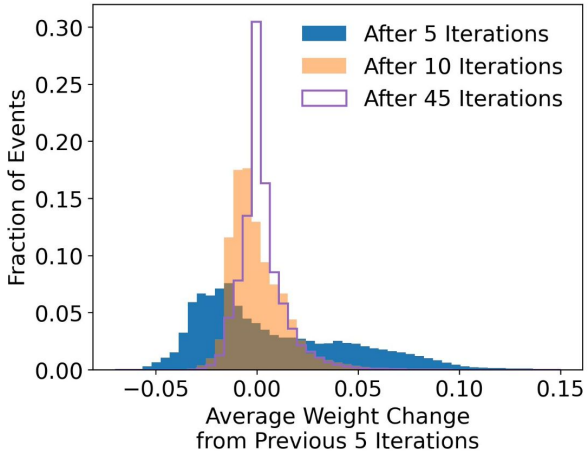
Inputs:

- kinematic observables (p_μ, $\cos\theta_\mu$, p_p, δp_T, $\delta\alpha_T$, $\delta\phi_T$)
- detector sample ID
- interaction topology (CC0π0p, CC0π1p, CC0πNp, CC1π, CCother)

ref. https://arxiv.org/pdf/2504.06857v1

Unfolding of ND280 data: Omnifold

Test:



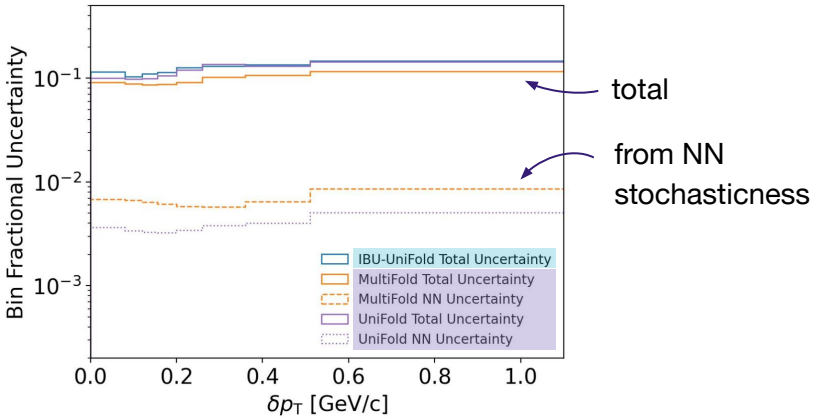
Comparision with conventional-like unfoldings:

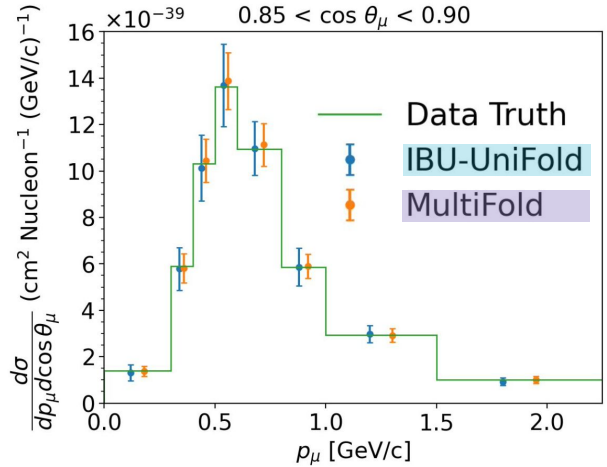
- not straightforward since Omnifold is unbinned
- use Omnifold in a way that it is mathematically equivalent to IBU (Iterative Bayesian Unfolding): inputs limited to bin indices

	Method	χ^2 $(p_\mu, \cos\theta_\mu)$ DoF=58	χ^2 δp_T DoF=8	χ^2 $\delta\alpha_T$ DoF=8	χ^2 $\delta\phi_T$ DoF=8
	Prior	298.2	2.3	5.9	4.9
conventional -like method	IBU-UniFold	2.1	0.2	0.4	0.1
	Binned UniFold	21.4	1.4	0.9	0.5
Omnifold variations (inputs choice)	UniFold	27.1	1.1	0.6	1.1
	MultiFold	3.1	0.3	0.2	0.3
	OmniFold	10.0	0.8	1.1	0.4

	Method	Triangular Discriminator $(p_\mu, \cos\theta_\mu)$	δp_T	$\delta\alpha_T$	$\delta\phi_T$
	Prior	545.6	27.5	31.2	26.7
conventional -like method	IBU-UniFold	17.1	1.9	3.4	0.8
	Binned UniFold	29.9	2.8	6.0	1.9
Omnifold variations (inputs choice)	UniFold	17.3	5.7	5.8	1.7
	MultiFold	2.7	0.7	0.6	0.6
	OmniFold	9.4	1.7	3.0	2.1

Uncertainties:





ref. https://arxiv.org/pdf/2312.02045

Modeling posterior systematic: Normalizing Flows

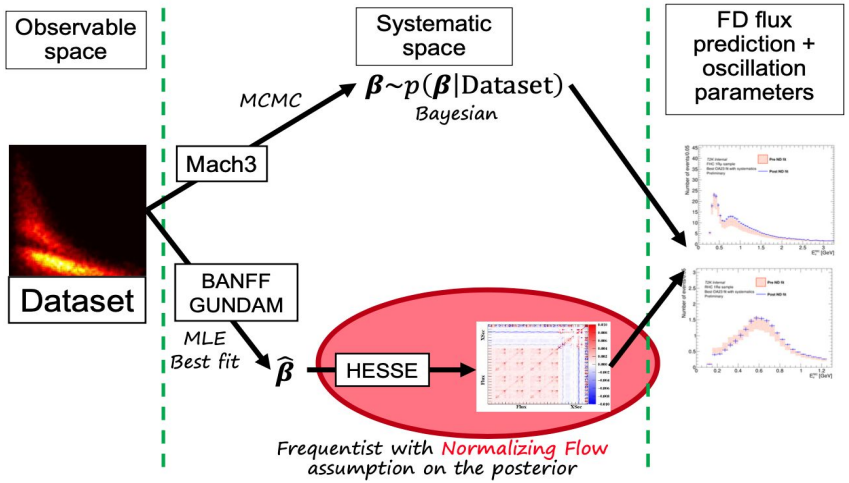
ND fit is a constrain on systematic uncertainties using ND280 observations

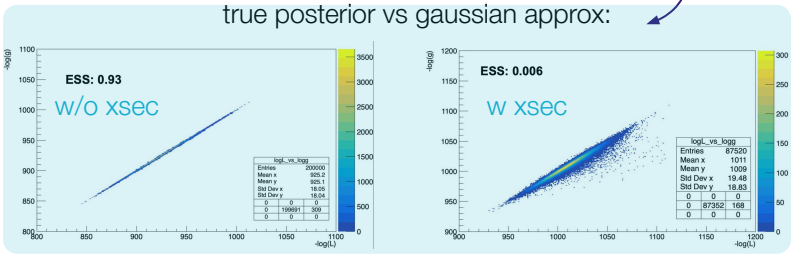
Context: ND280 likelihood of systematics depends on >700 variables (from flux, detector and xsec uncertainties)

Goal: Learn the posterior probability distribution of neutrino flux binned in neutrino energy

Conventional methods:

- Semi-frequentist (*GUNDAM*): gaussian assumption on the posterior, get best-fit params from MLE
  + analytical
  - miss xsec non-gaussianities
- Bayesian (*Mach3*): sample from the posterior using MCMC
  + capture non-gaussianties
  - pt cloud estimation (not analytical)





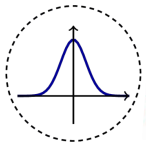
Where ML comes in:

replace *GUNDAM* gaussian approximation
by something more complex
to capture non-gaussianities of the posterior distribution

ref. https://arxiv.org/pdf/2312.02045

Modeling posterior systematic: Normalizing Flows

NormFlows:

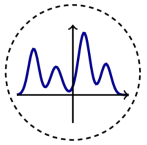


simple proba

$u \sim p_u$ Base distribution

T_θ Transformation

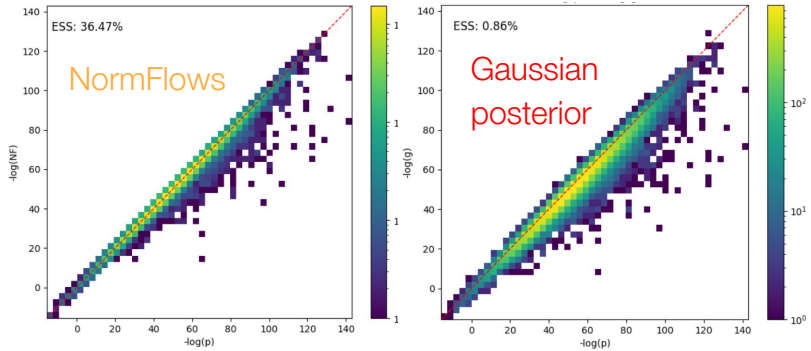
$x = T_\theta(u) \sim q_\theta$ Predicted distribution



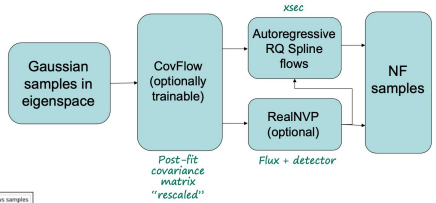
complex proba

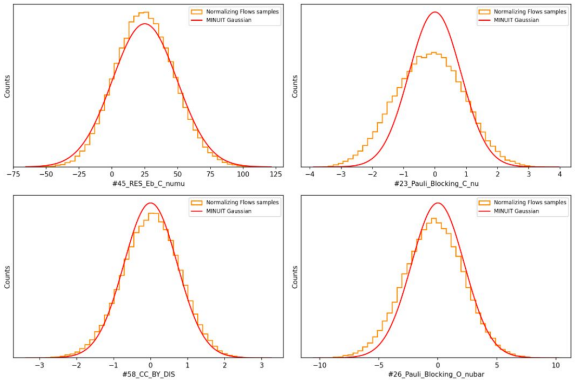
↳ straighforward to: sample $x = T_\theta(u) \sim q_\theta$ with $u \sim p_u$ & evaluate the proba $q_\theta(x) = p_u(u) \,|\det\left(J_{T_\theta}(u)\right)|^{-1}$

Test: on full sets of systematics (OA 2022 config) to learn the 59 xsec probas conditioned on the 652 flux + detector systematics



Archi: RQ-NSF for xsec dimensions & Real-NVP for flux+detector ones





Fast method:
10M sample /day
vs ?

ref. https://arxiv.org/pdf/2502.14452

CCQE cross-section sampling: Normalizing Flows

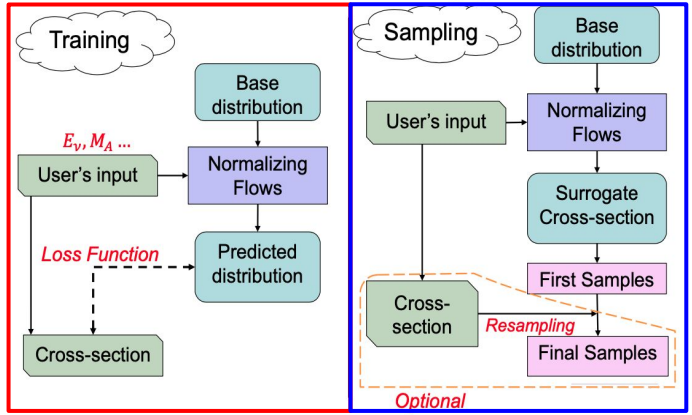
Goal: efficient MC sampling for CCQE exclusive cross-section of neutrino-nucleus (^{12}C)
(for Mean Field xsec, historically to complex to sample from)

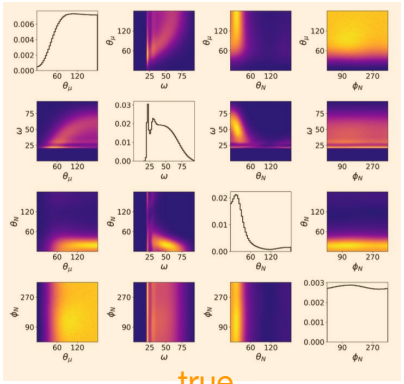
Conventional method:
use sophisticated nuclear models $\frac{d^6\sigma}{dE_\nu d\omega d\Omega_\mu d\Omega_N} \propto L^{\mu\nu} W_{\mu\nu}$ → long computations

ML method: same Normalizing Flow architecture as project 5
(slightly different loss function derived from KLD)

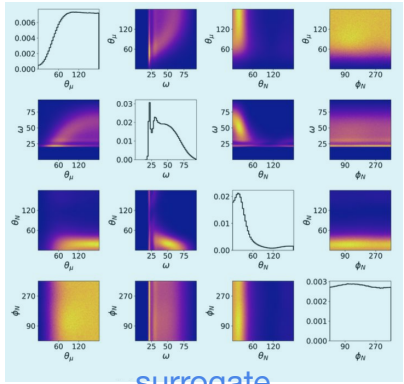
Train: to model 1p1h i.e. $p(\theta_\mu, \omega, \theta_N, \phi_N \mid E_\nu, \alpha)$ on 2 shells for many E_ν

Test: sample e.g for (600 MeV, 1s shell)

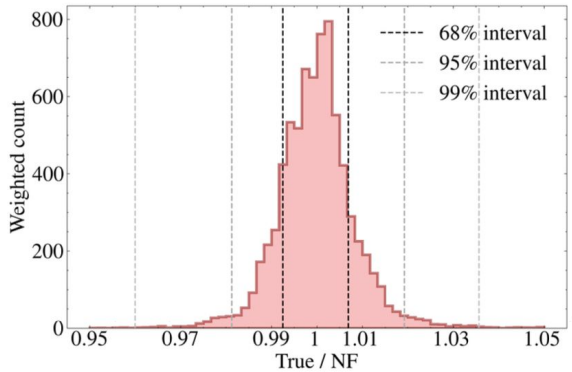




true



surrogate



Fast method:
1M sample /25min/CPU
vs 1 day/CPU