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
- The AI/ML working group
- Overview of the group activities:
    1. SFGD Momentum reconstruction and PID with a BDT (TMVA)
    2. Global ND280 PID with BDT (XGBoost)
    3. Identify EM shower with PointNet
    4. 2D+3D CNN for e/γ classification
    5. Other projects using ND280 data (Omnifold, Normalizing Flows)
- Detailed project: PID in the SFGD with a Transformer

The T2K experiment & its Near Detector

The T2K experiment & its Near Detector

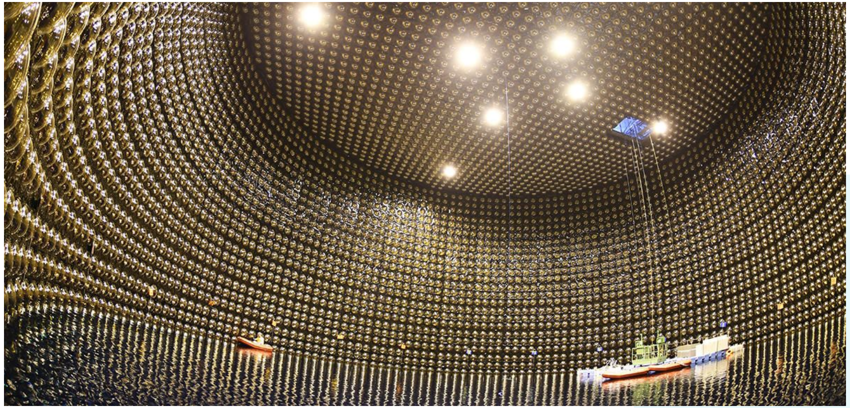
particle accelerator to create neutrino beam

The T2K experiment & its Near Detector

near detector: ND280

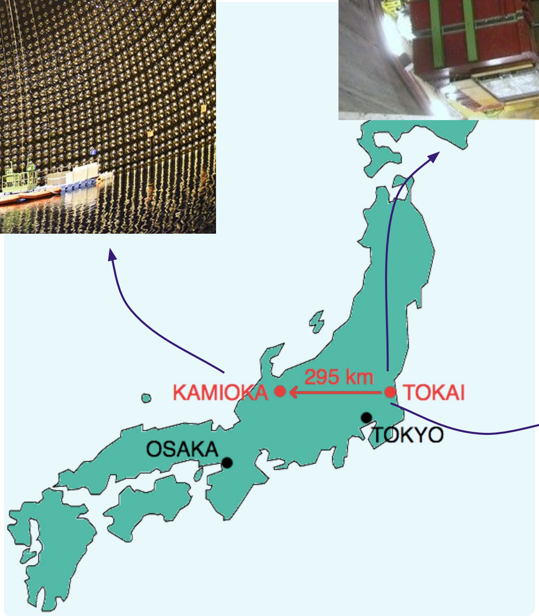
particle accelerator to create neutrino beam

The T2K experiment & its Near Detector



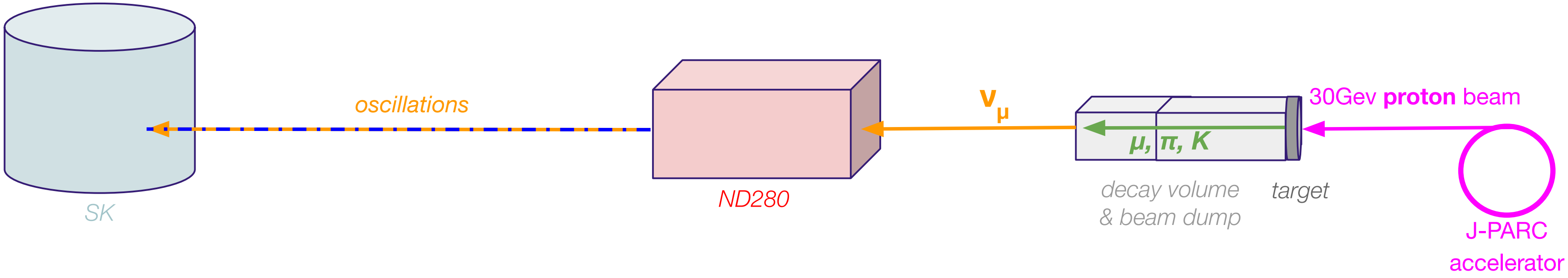
far detector: Super-Kamiokande

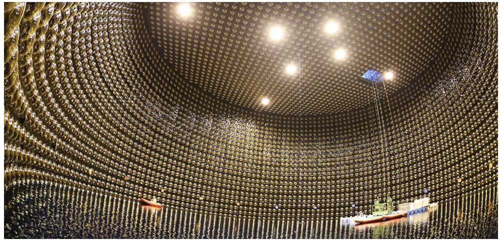
near detector: ND280



particle accelerator to
create neutrino beam

The T2K experiment & its Near Detector





Super-Kamiokande

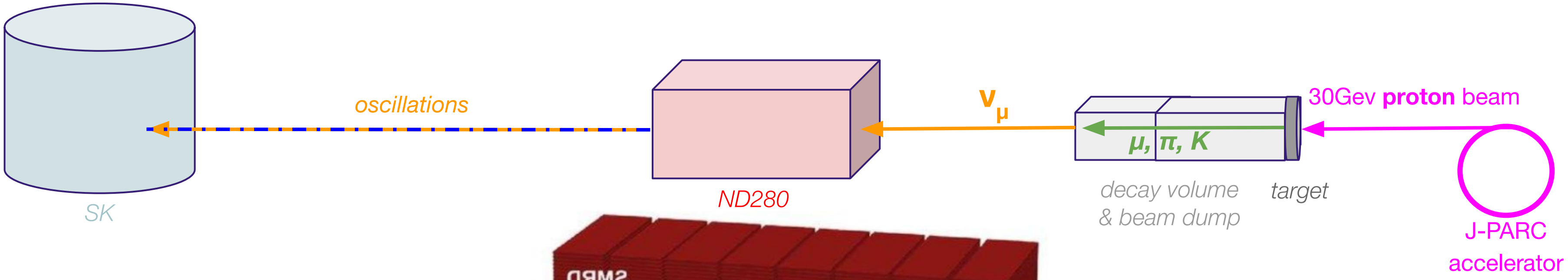
295km

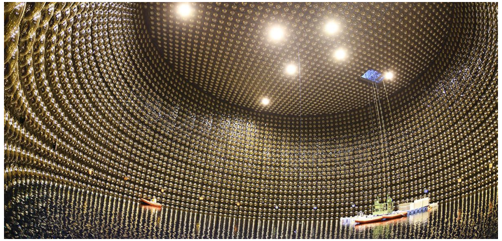
ND280

280m

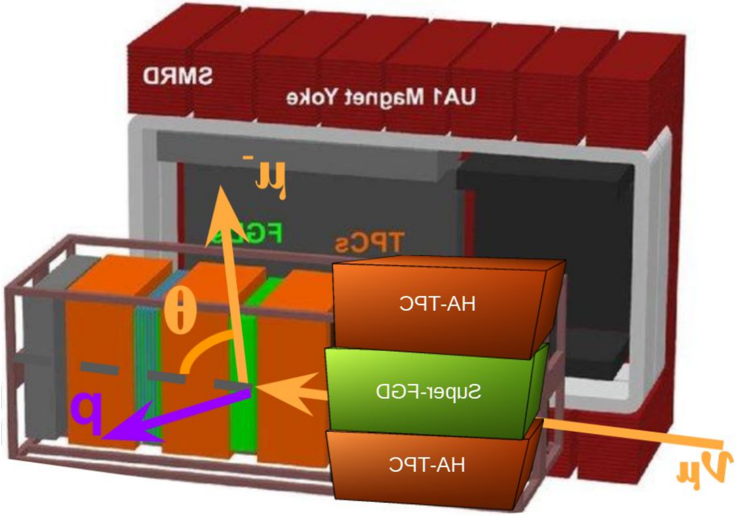
neutrino beam

The T2K experiment & its Near Detector



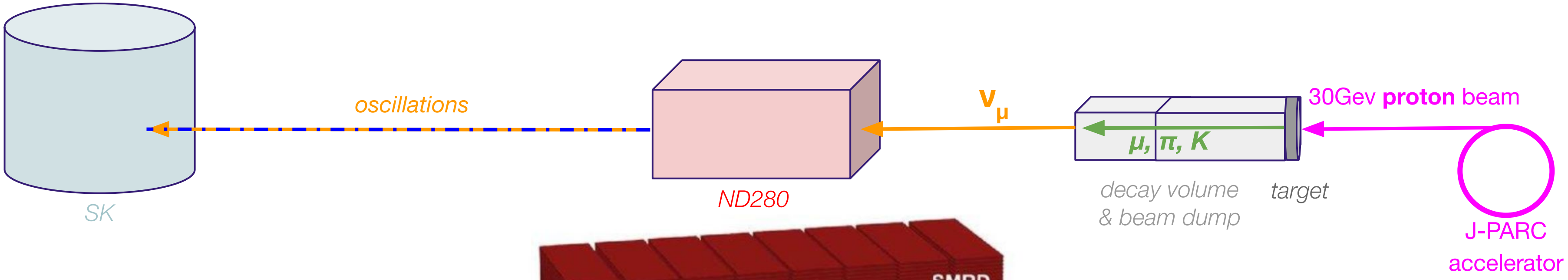


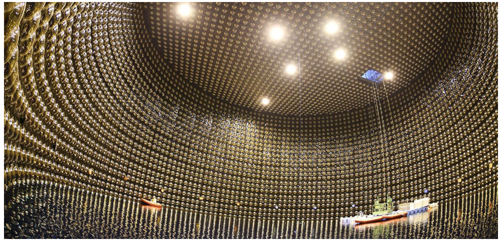
Super-Kamiokande



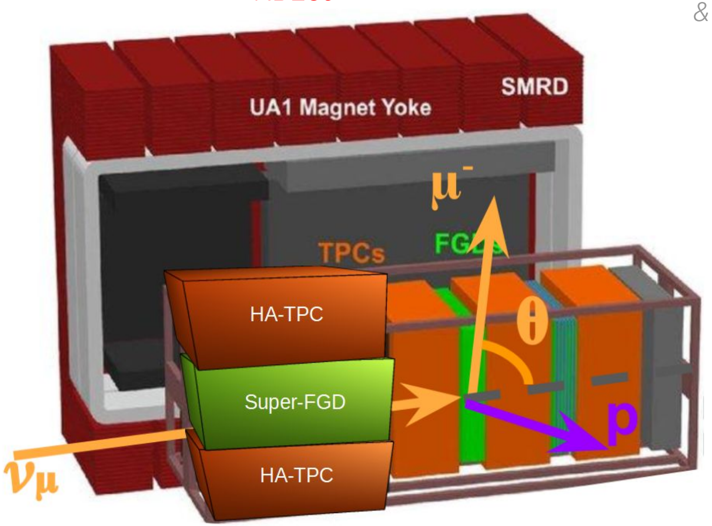
neutrino beam

The T2K experiment & its Near Detector



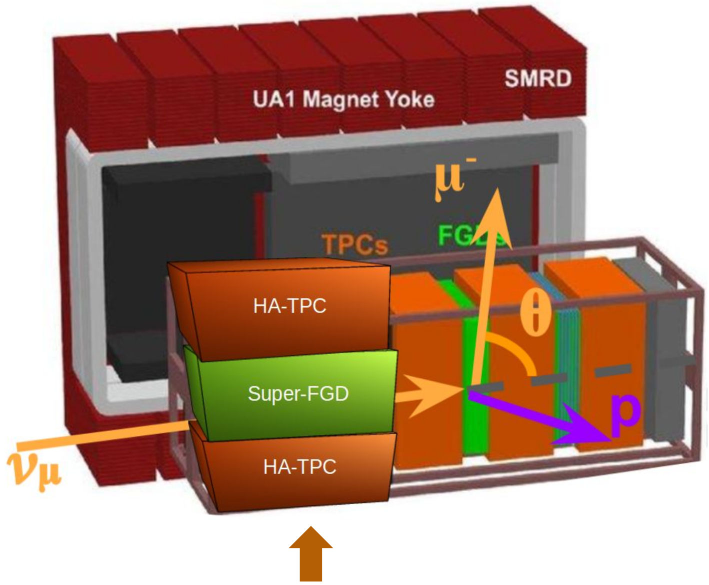


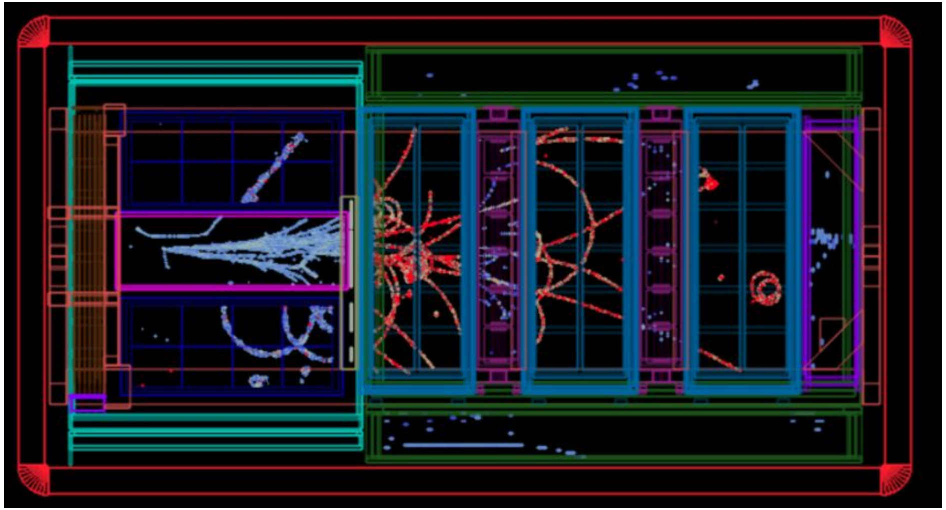
Super-Kamiokande



neutrino beam

The T2K experiment & its Near Detector

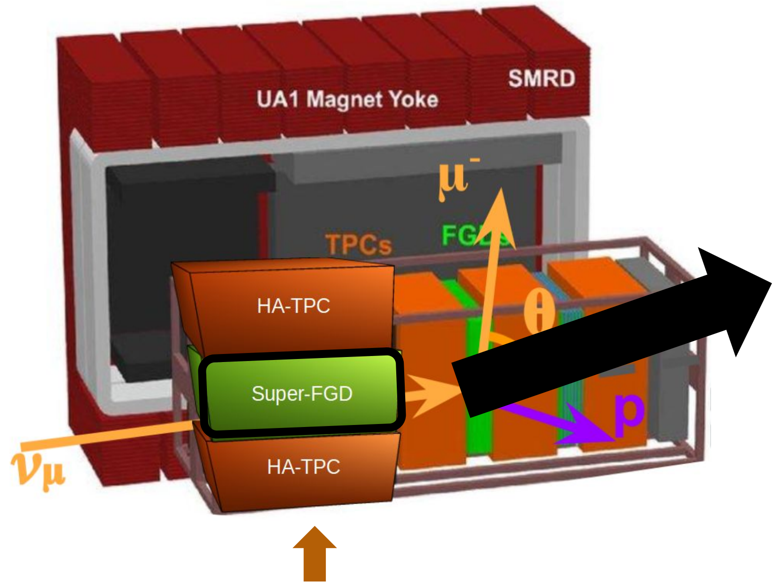




ND280 Upgrade installed last year!

& data taking since end of 2024

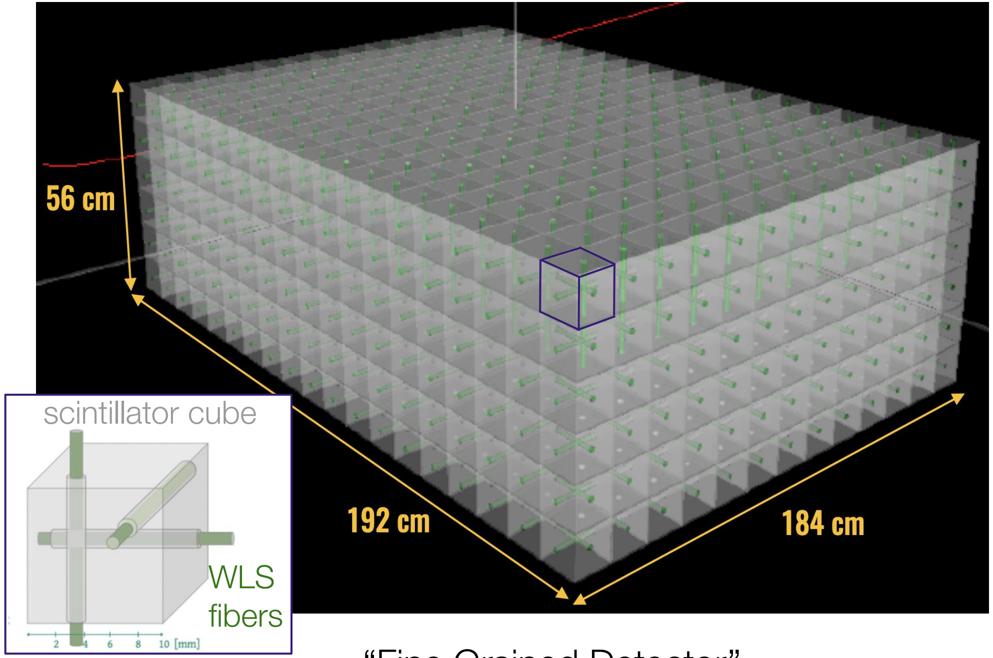
The T2K experiment & its Near Detector



ND280 Upgrade installed last year!

& data taking since end of 2024

The Super-FGD: a Scintillator Detector



~2M cubes

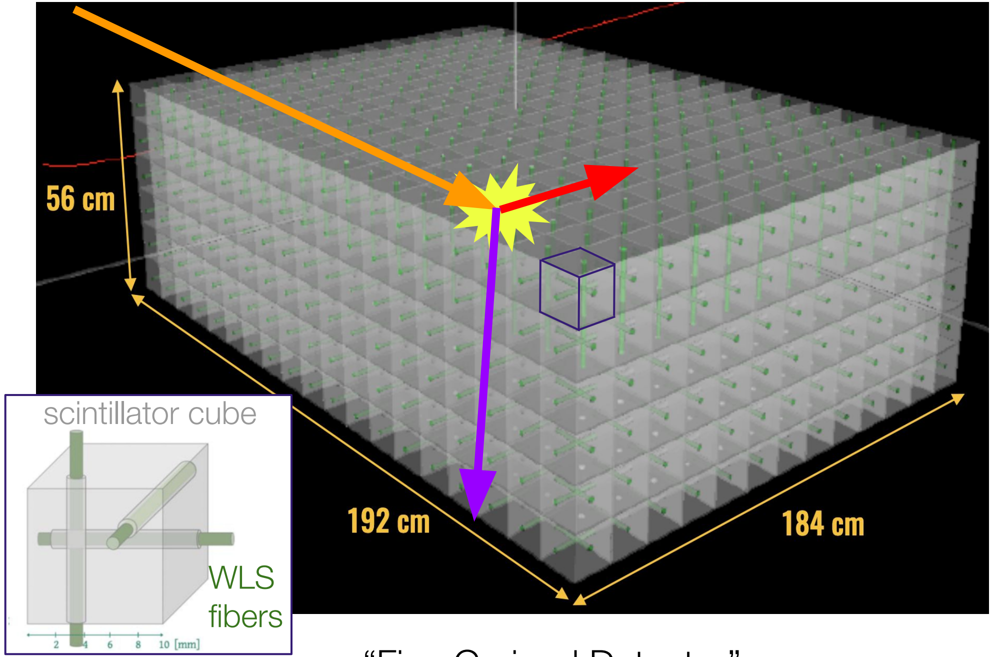
"Fine Grained Detector"

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: a Scintillator Detector



~2M cubes

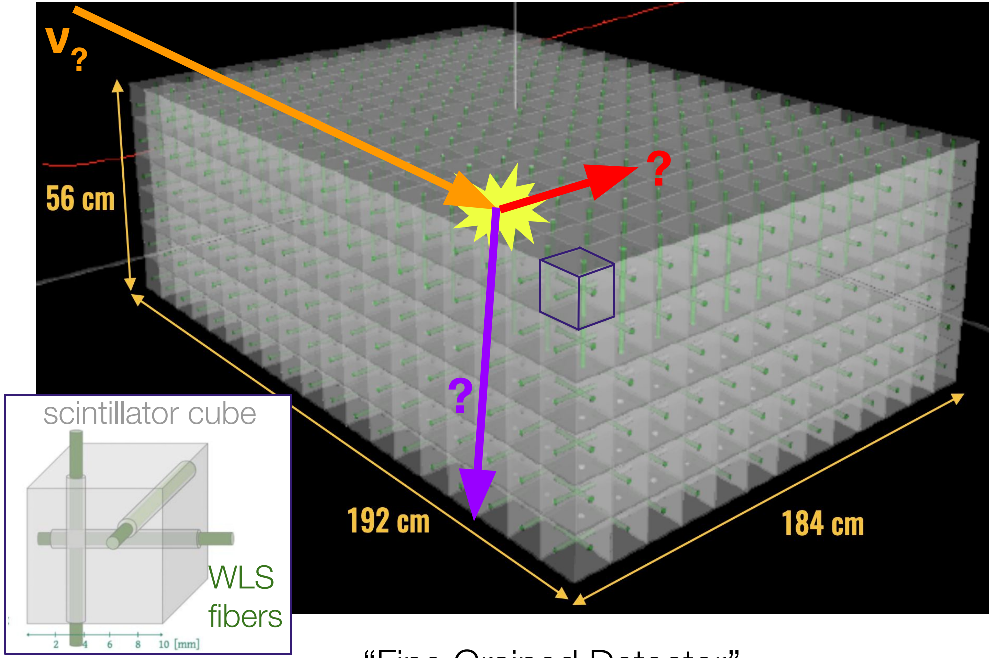
“Fine Grained Detector”

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

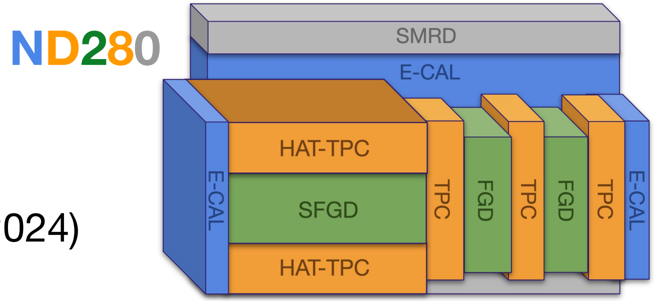
The Super-FGD: a Scintillator Detector



~2M cubes

“Fine Grained Detector”

The ND280 AI/ML working group



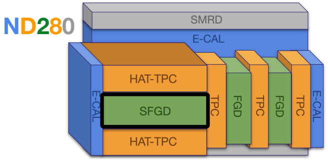
- Recent working group dedicated to AI/ML methods (since Nov. 2024)
- Convener is Saùl Alonso Monsalve (ETH Zurich)
- ~ 10 active analysers
- In different part of the experiment, various task focused around the Near Detector:

★ ***reconstruction***: vertex activity, track fitting, momentum reco ⟶ 3 projects

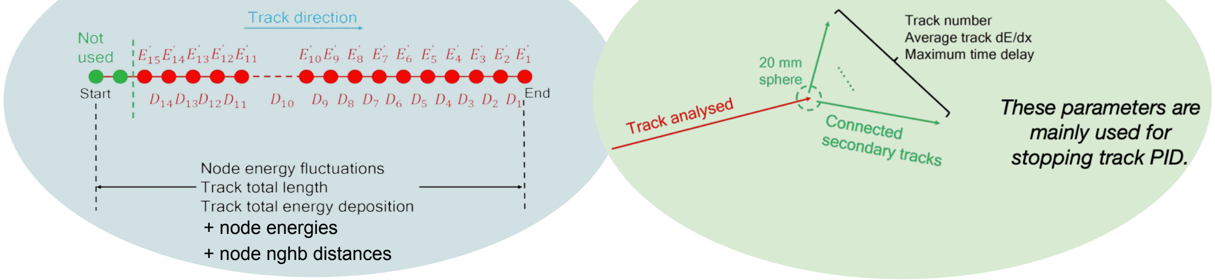
★ ***analysis***: PIDs, unfolding, modelling posterior systematics ⟶ 7 projects

★ ***simulation***: cross-section sampling ⟶ 1 project

1. SFGD Momentum reconstruction & PID with BDT (TMVA)



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



These parameters are mainly used for stopping track PID.

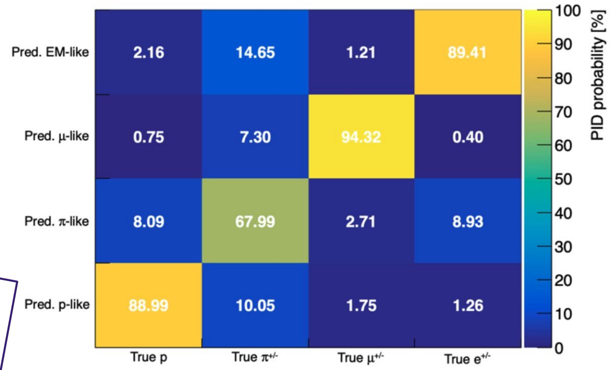
BDT HP: 5 to tune (done by hand):

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

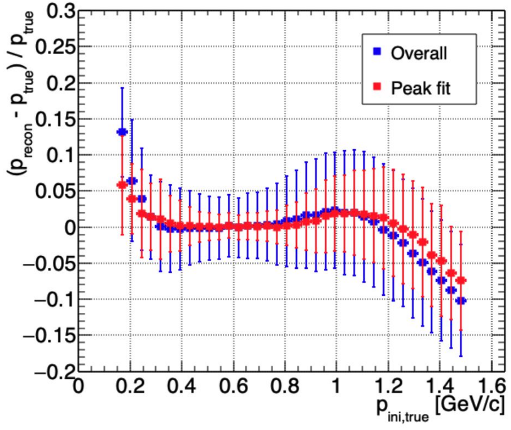
same for regression & classification

Training:
- on particle-gun MC data (i.e. 1 particle /event): p, $\pi^{\pm}$, $\mu^{\pm}$, $e^{\pm}$
- 4 independent PID classifier & 3 independent momentum regressors

Test: classification results



momentum resolution:

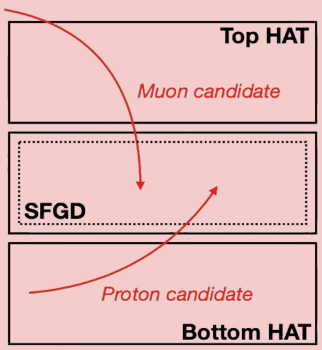


Currently in use in official T2K analysis software

1. SFGD Momentum reconstruction & PID with BDT (TMVA)

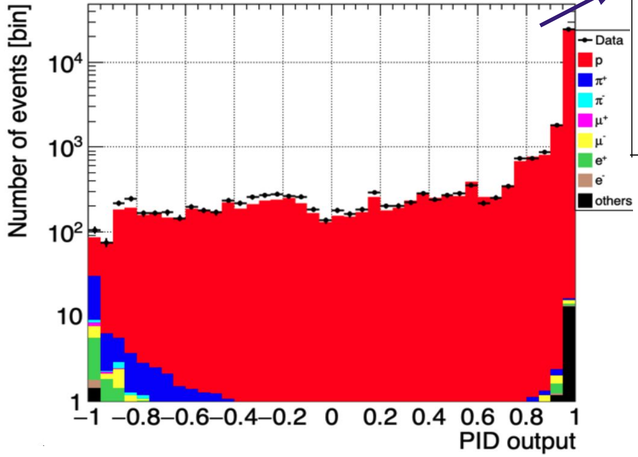
BDT Systematics:

- for real application, apply BDT to MC & data ⇒ check if have similar performances!
- If not → need to evaluate the difference
- Propage this difference as systematic source in the analysis
- Evaluation of systematics: **use stopping control sample**



stopping control sample

PID efficiency:

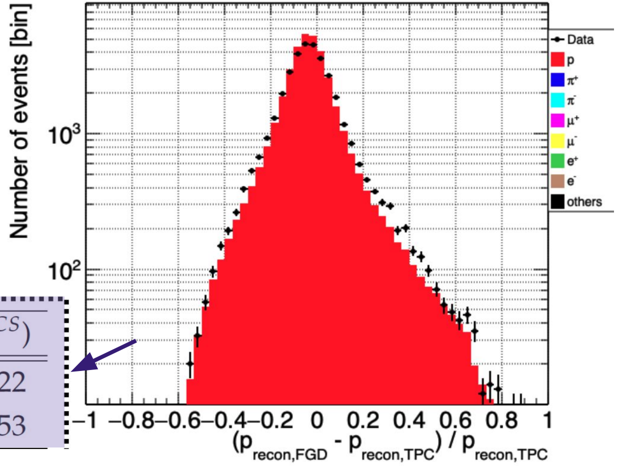


Control Sample	Data / MC
Proton	0.99 ± 0.00
Pion (π^+)	$\mathbf{2.17 \pm 0.06}$
Muon (μ^-)	1.32 ± 0.05
Electron / Positron	1.41 ± 0.15

those values are propagated as systematics

Control Sample	Data / MC (r^{CS})
Proton	1.2034 ± 0.0222
Muon (μ^-)	0.8840 ± 0.0153

Momentum resolution:



1. SFGD Momentum reconstruction & PID with BDT (TMVA)

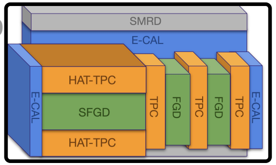
Questions

- is the momentum regression used in SFG Recon?
- are the systematics currently propagated/where/what is the plan otherwise?
- in previous table, no pi-, no mu+ why
- controle sample: for each particle type? ID only with hats/tpcs? from outside events?
- 4 independent PID classifier & 3 independent momentum regressors
  but CM was classifier with 4 particles: just for the result plots?
-
- results data/mc FGD1,2 from technote → no results yet from SFGD? what should I say (control sample image is with SFGD): which are in use (eg for syst propa)

- 2 slides for this project because actively used in many analysis ⇒ good example

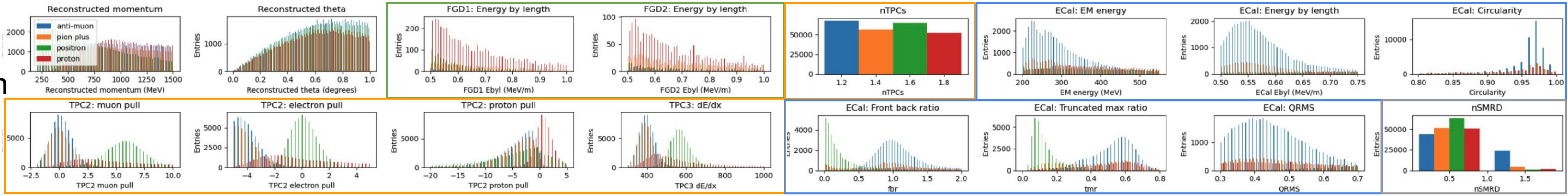
2. Global ND280 PID with BDT (XGBoost)

ND280



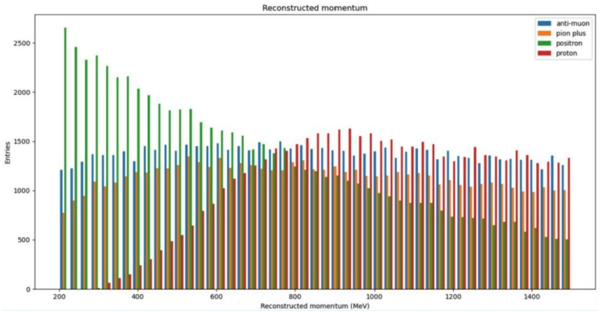
Goal: use inputs from many ND280 sub-detectors to get a global PID tool

Inputs:
16 variables from 4 of ND280 sub-detectors

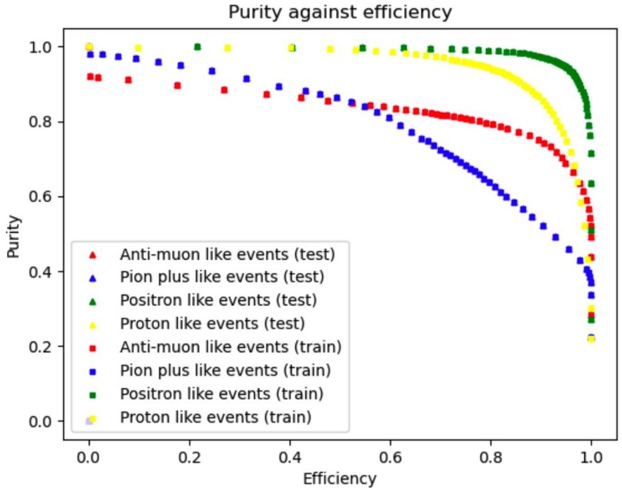


Training:
- on particle-gun MC data: p, π^+, μ^+, e^+
- starting position in FGD1 (later: SFGD)
- use HYPEROPT for the HPO

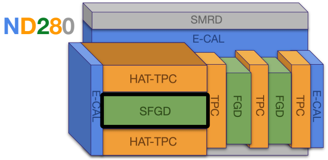
Preprocessing: re-weight each events to have uniform reconstructed momentum to avoid direct momentum dependence



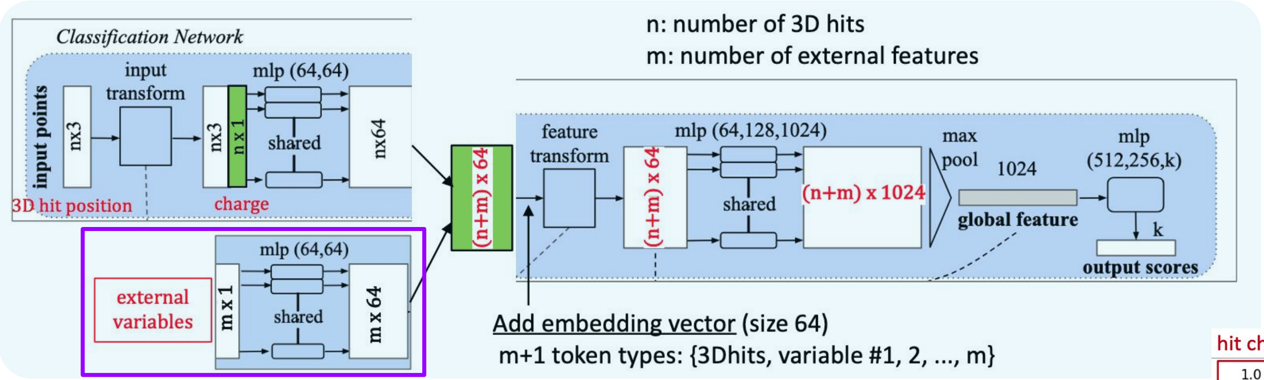
Results:



3. Identify EM shower in SFGD: PointNet



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



Goal:

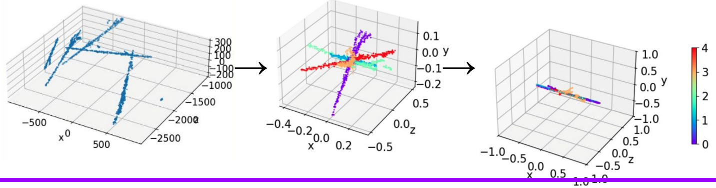
distinguish EM shower-like (e,γ) particle from non EM ones (μ,π,p)

Inputs:

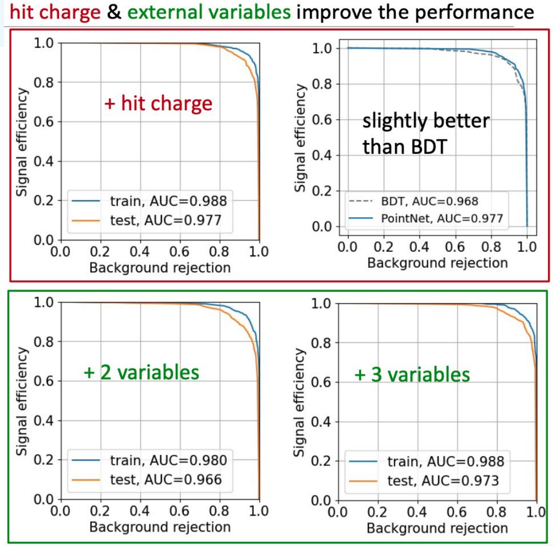
pgun of e^-, μ^- for now

Preprocessing:

- center and align showed better perf



Test:



External features: adding global feature of the event increase performances!
will be tested: add all BDT variables as external features

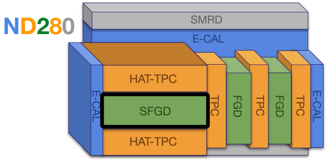
- total charge in the event
- total number of hits
- look for 3rd variable

3. Identify EM shower in SFGD: PointNet

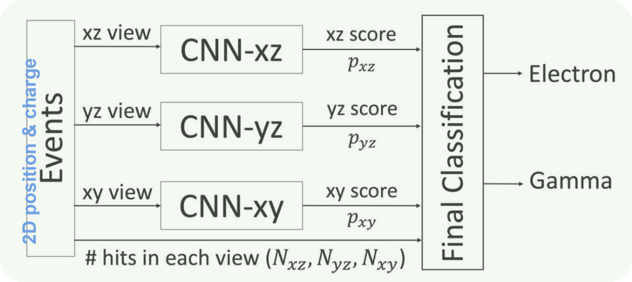
Question:

- segmentation network not used right? (from archi image) why don't we need it here
- how manageable is it to rm the 512 pt limit and make it changeable?
- plot updates with all eff,pur on same one?
- perf update with all BDT var?

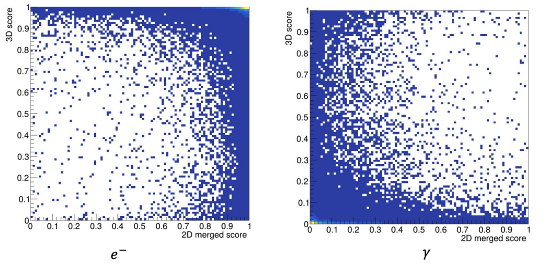
4. 2D+3D CNN for e/γ classification



Architecture: experimental combination of 2D CNNs (*ResNet50*) + sparse 3D CNN (*SSCN*)



& Sparse Submanifold convolutional networks ⇒ $$(final\ score) = \frac{(2D\ score) + (3D\ score)}{2}$$

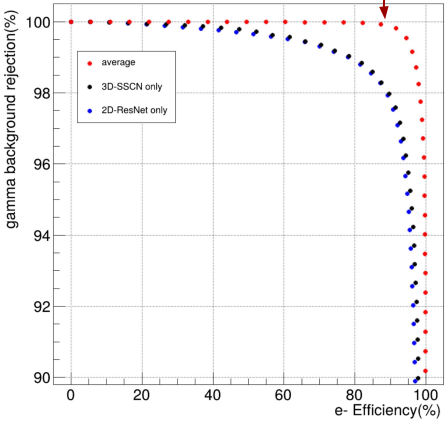


2D vs 3D scores: not much correlations

Training: - particle-gun MC data: e⁻ and γ with 400k training event
- true info used to precut some events

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**

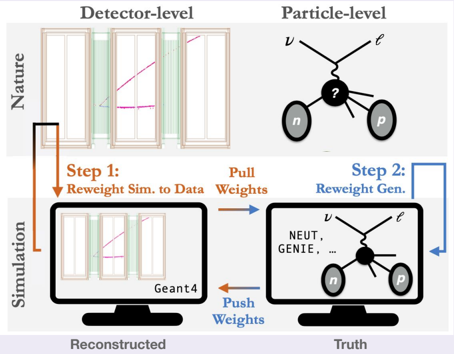


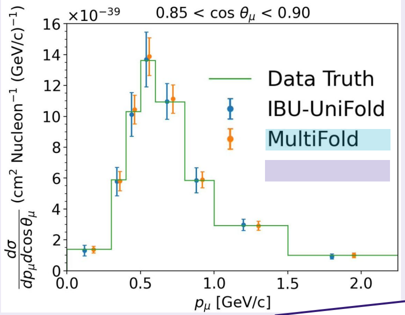
e/γ events looks similar because of
$\gamma \rightarrow e^+e^-$ creating similar shower of particles
⇒ **challenge is to distinguish them**

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

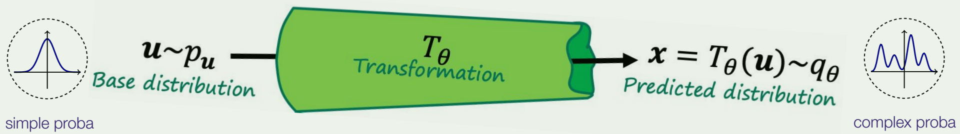
Omnifold:

unbinned method to unfold ND280 data faster

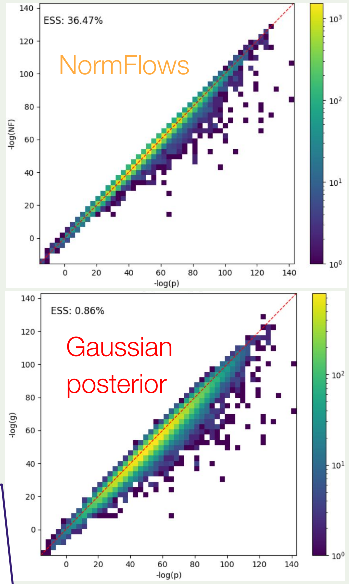




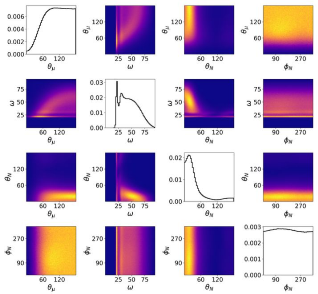
Normalizing flows:

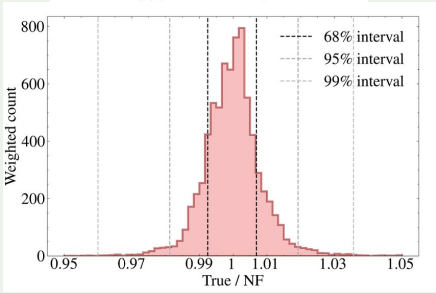


Modeling posterior systematics



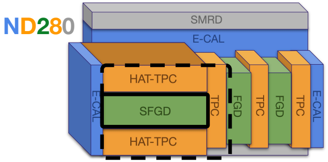
Efficient CCQE cross-section sampling





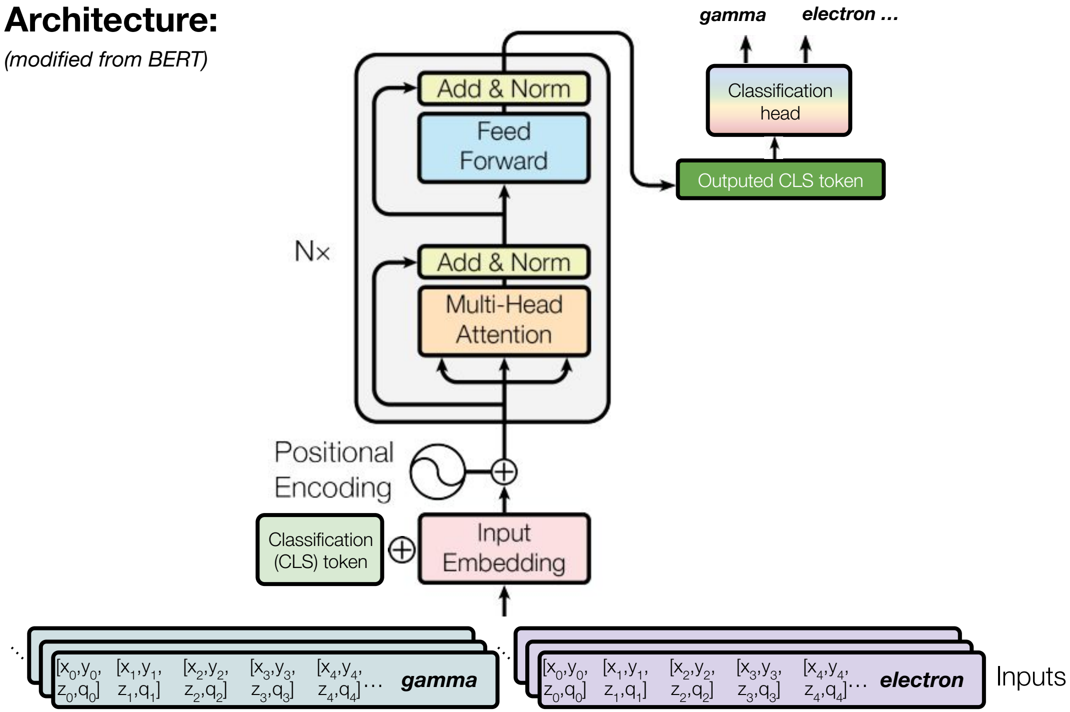
Discussions to incorporate these in T2K softwares

Detailed project: PID in the SFGD with a Transformer

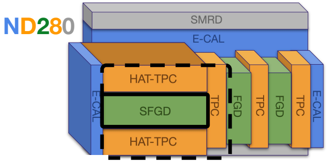


Architecture:

(modified from BERT)

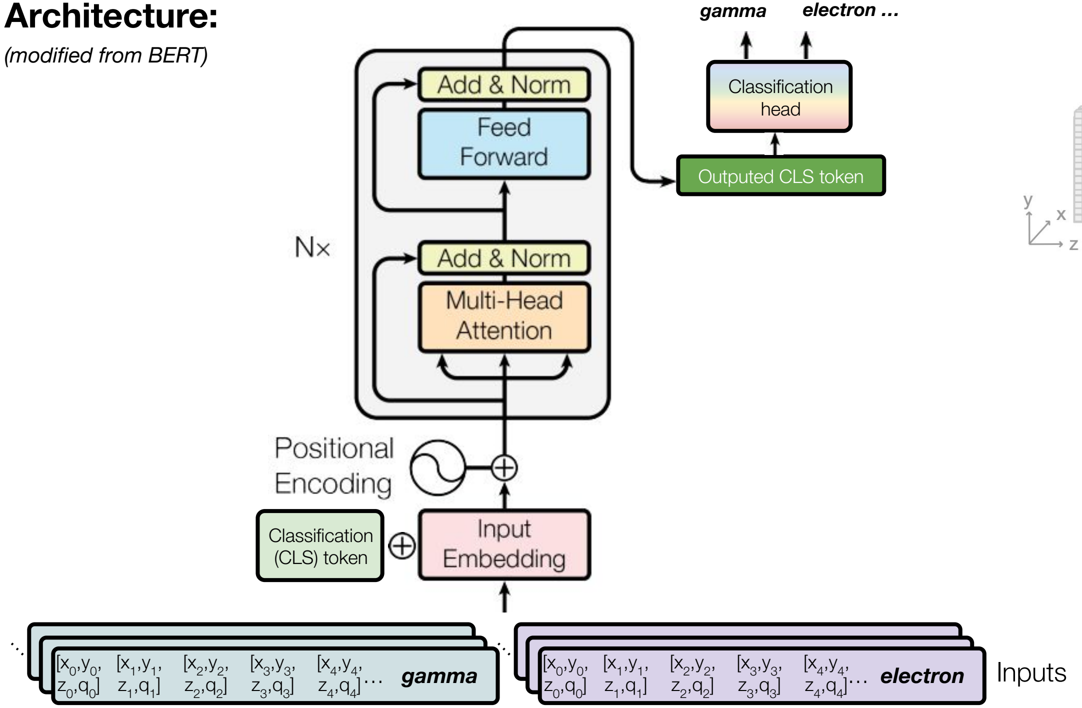


Detailed project: PID in the SFGD with a Transformer

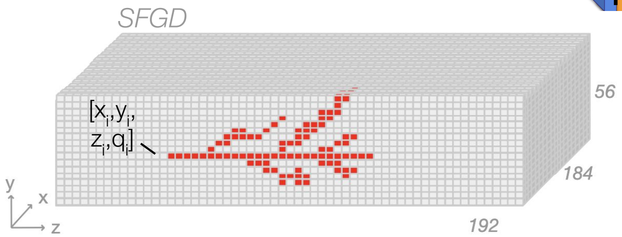


Architecture:

(modified from BERT)



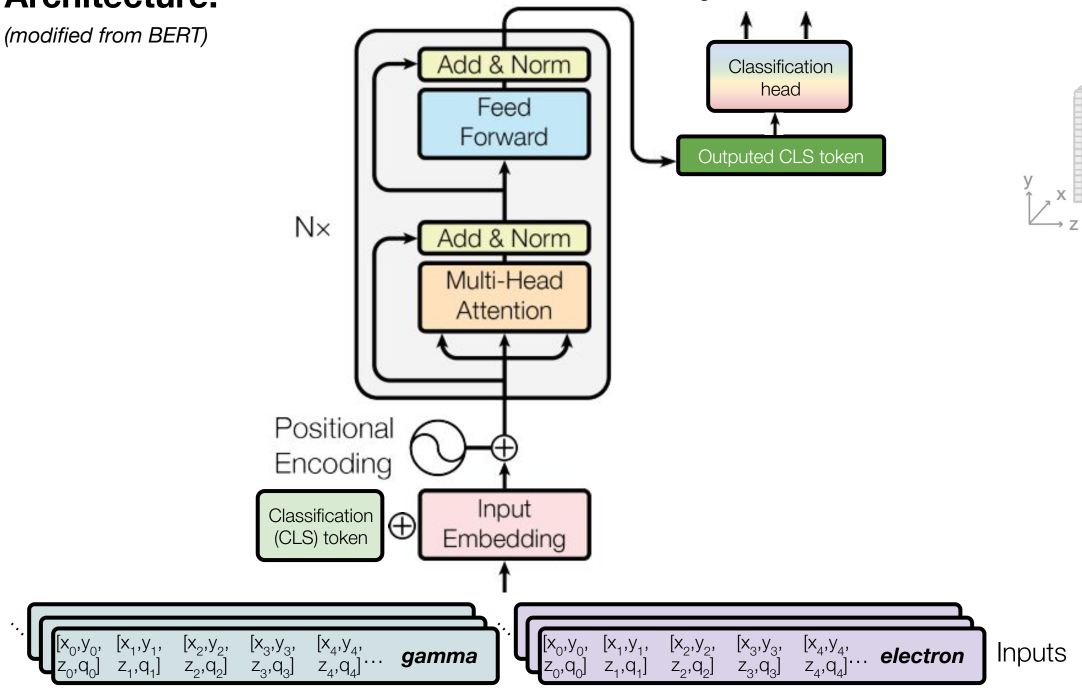
Inputs:



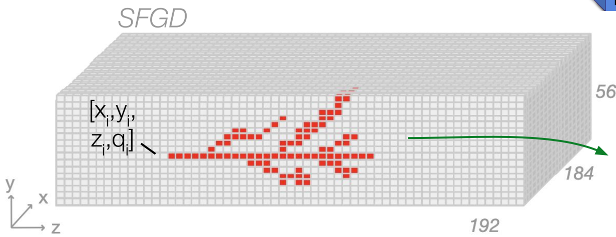
Detailed project: PID in the SFGD with a Transformer

Architecture:

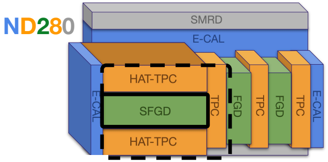
(modified from BERT)



Inputs:



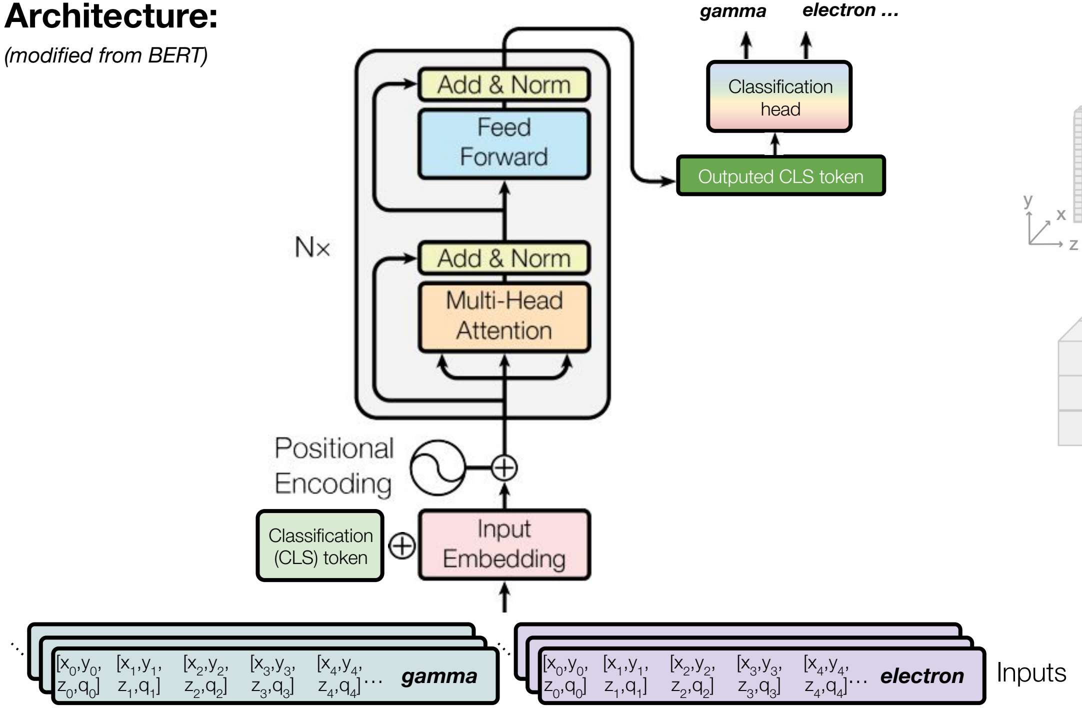
Too long sequences of hits: use Vision Transformer principle



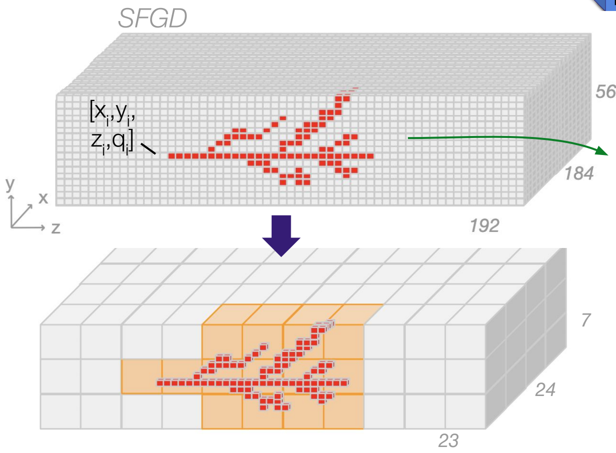
Detailed project: PID in the SFGD with a Transformer

Architecture:

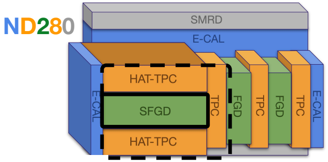
(modified from BERT)



Inputs:



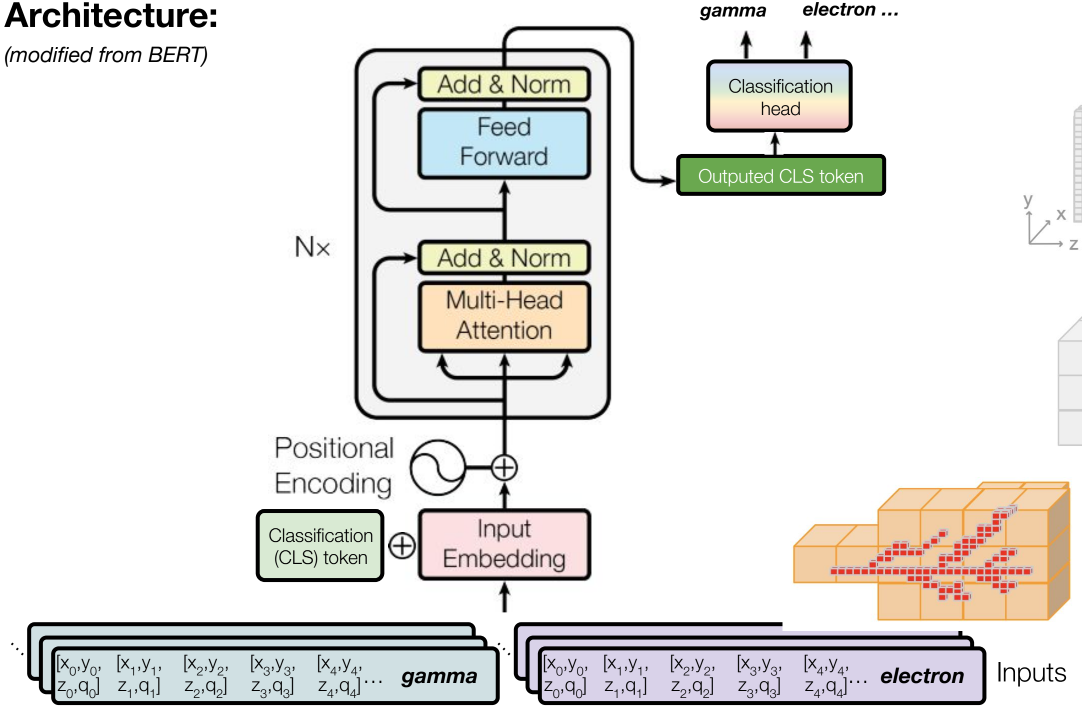
Too long sequences of hits: use Vision Transformer principle



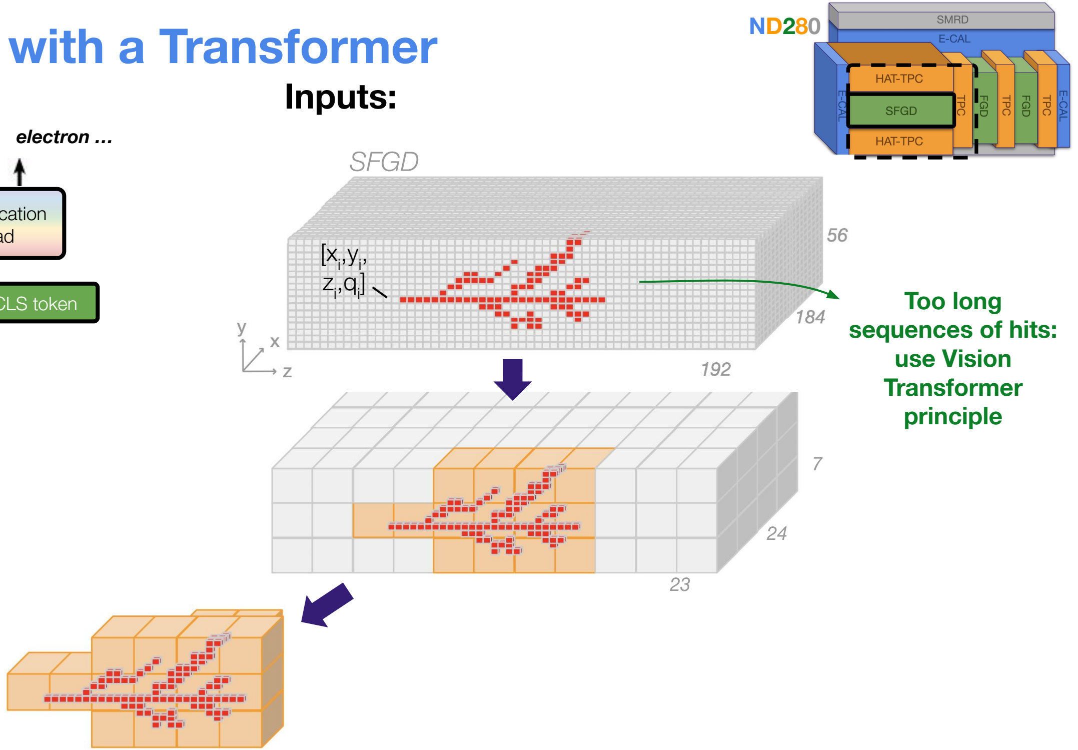
Detailed project: PID in the SFGD with a Transformer

Architecture:

(modified from BERT)

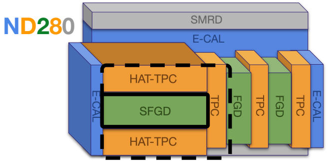


Inputs:



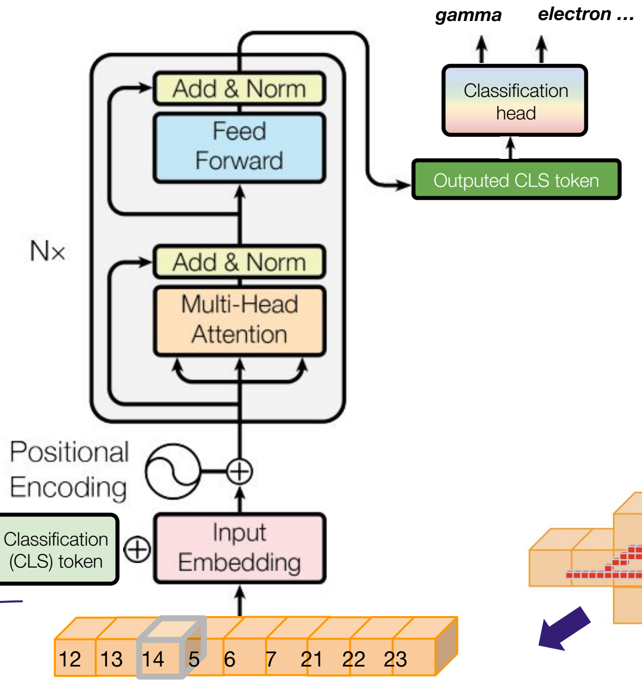
Too long sequences of hits: use Vision Transformer principle

Detailed project: PID in the SFGD with a Transformer



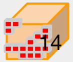
Architecture:

(modified from BERT)

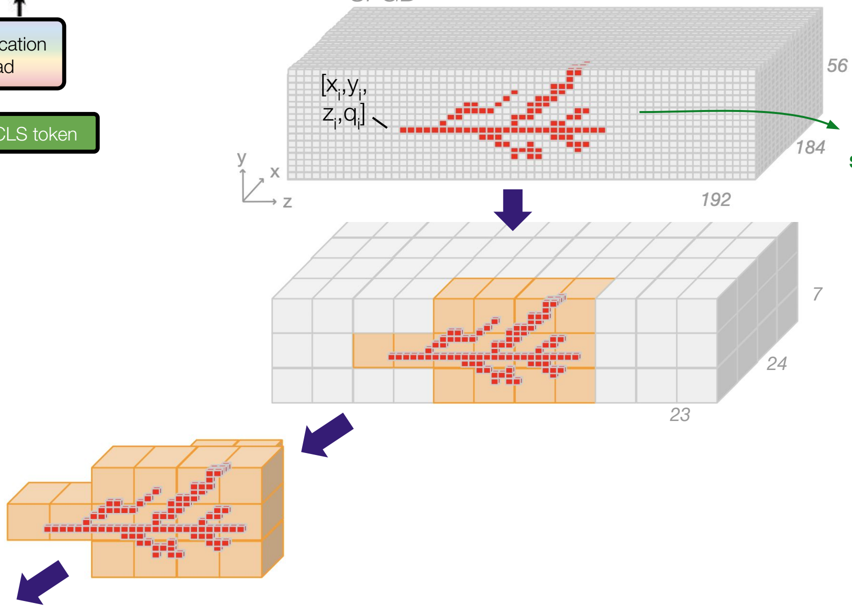


positional encoding of the parent cubes in the parent grid

each parent cube holds the info of its child voxels: relative positions & charge

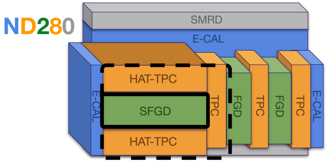


Inputs:



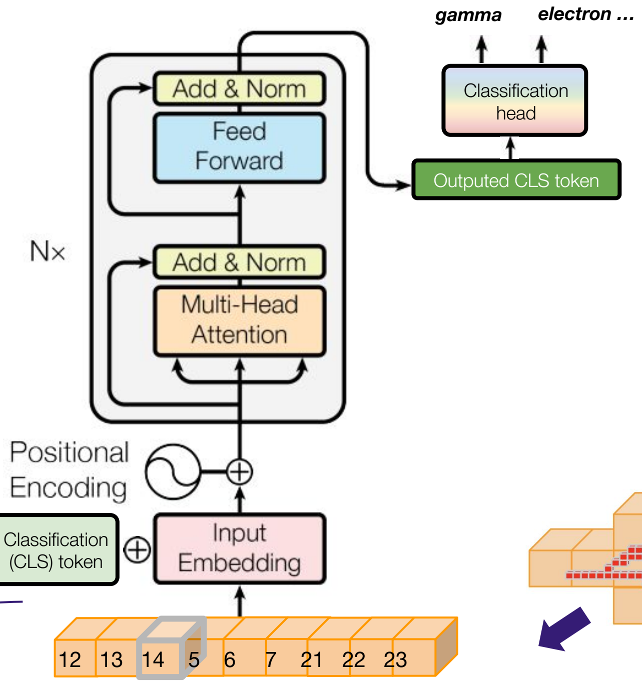
Too long sequences of hits: use Vision Transformer principle

Detailed project: PID in the SFGD with a Transformer



Architecture:

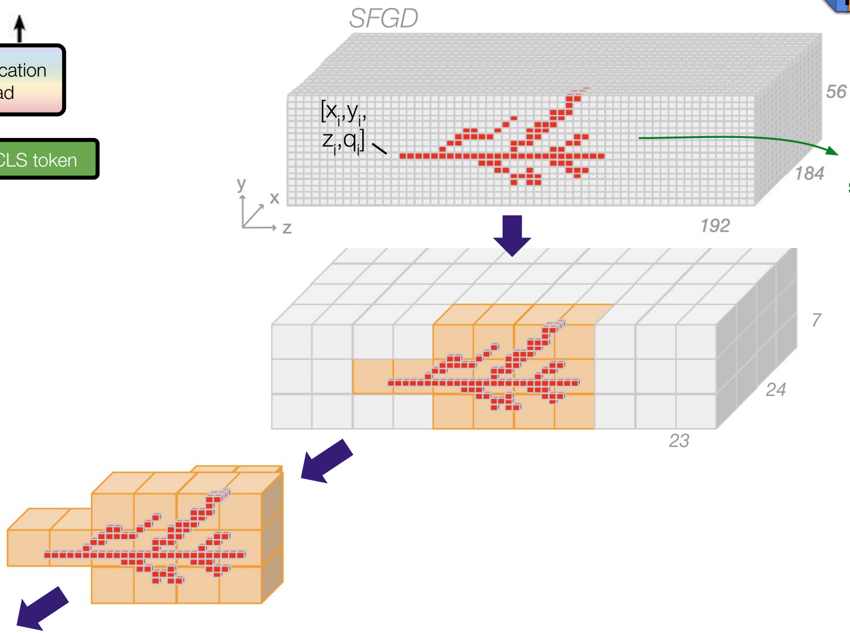
(modified from BERT)



positional encoding of the parent cubes in the parent grid

each parent cube holds the info of its child voxels: relative positions & charge

Inputs:



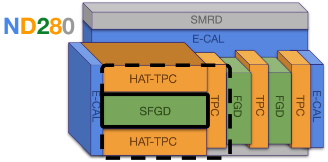
Too long sequences of hits: use Vision Transformer principle

Training: - particle-gun MC data: e^-, γ, π^-, μ^-

- either along 1 direction or isotropic

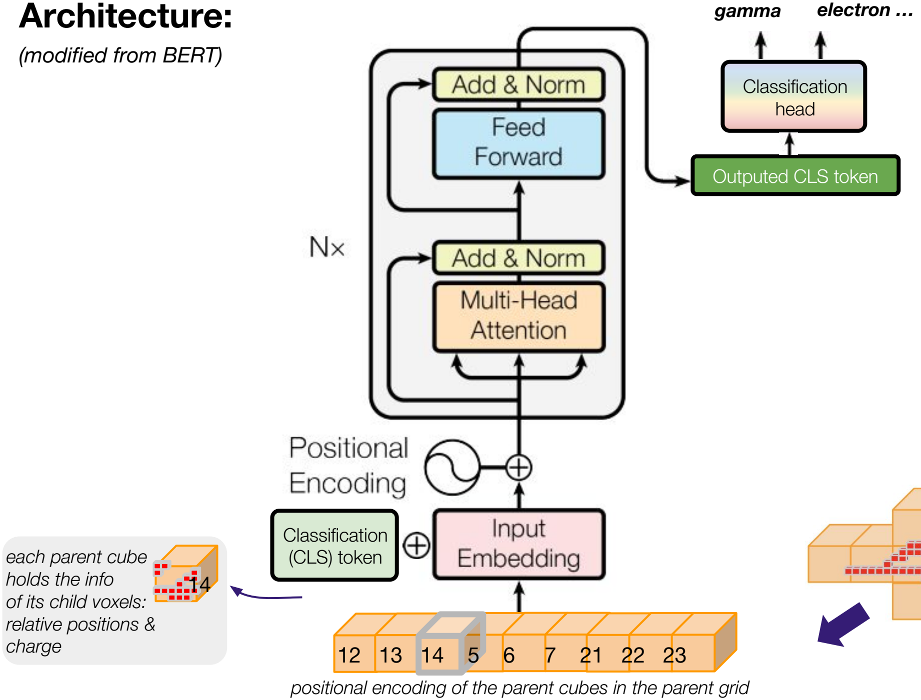
'z-pgun' *'iso-pgun'*

Detailed project: PID in the SFGD with a Transformer

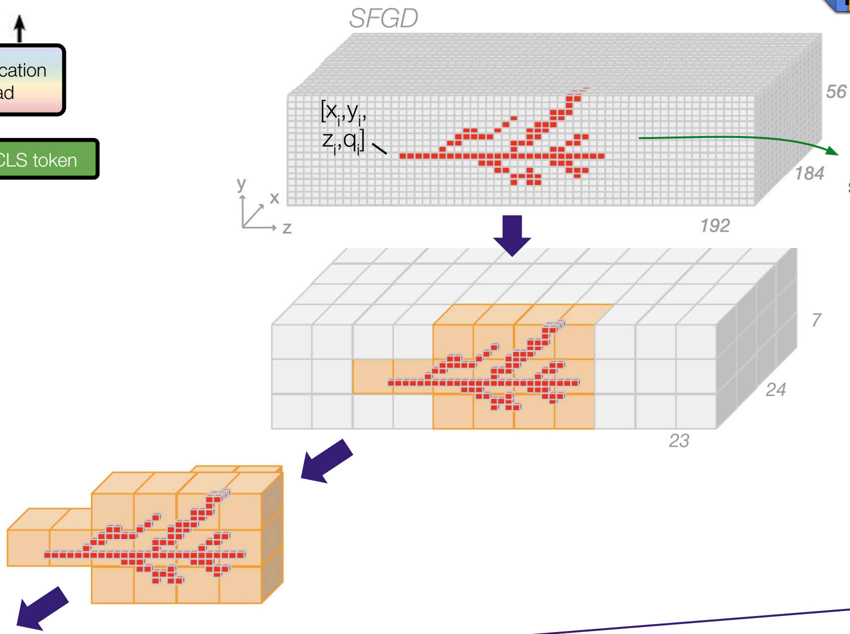


Architecture:

(modified from BERT)



Inputs:



Too long sequences of hits: use Vision Transformer principle

Training: - particle-gun MC data: e^-, γ, π^-, μ^-

- either along 1 direction or isotropic

'z-pgun' *'iso-pgun'*

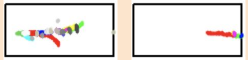
- architecture converted in C++ (*Libtorch*)
- saved weights from training

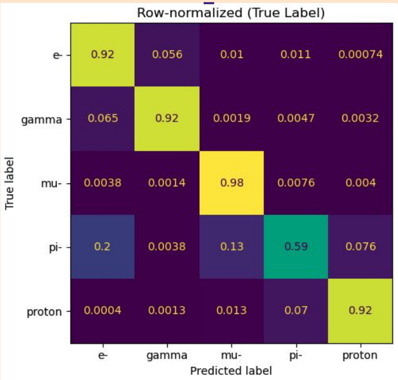
➡ 1rst complex ML model implemented in analysis soft!

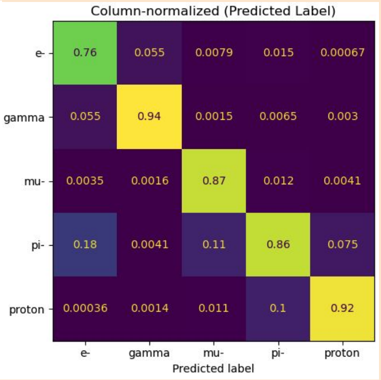
Detailed project: PID in the SFGD with a Transformer

Test:

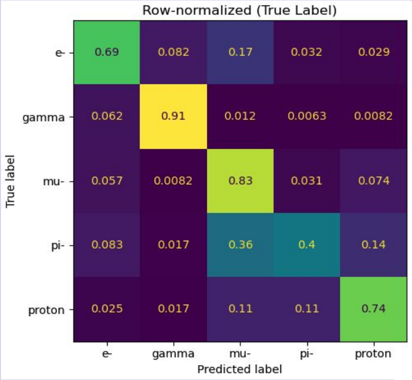
☺️ ***z-pgun*** *(train&test)*

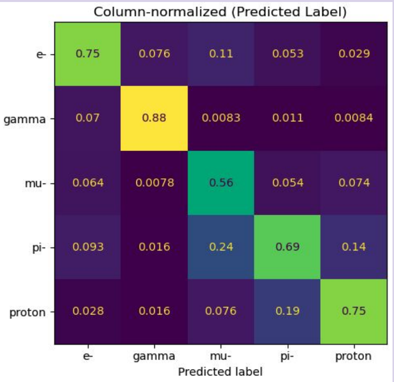






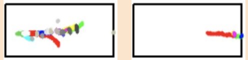
😐 ***iso-pgun*** *(train&test)*





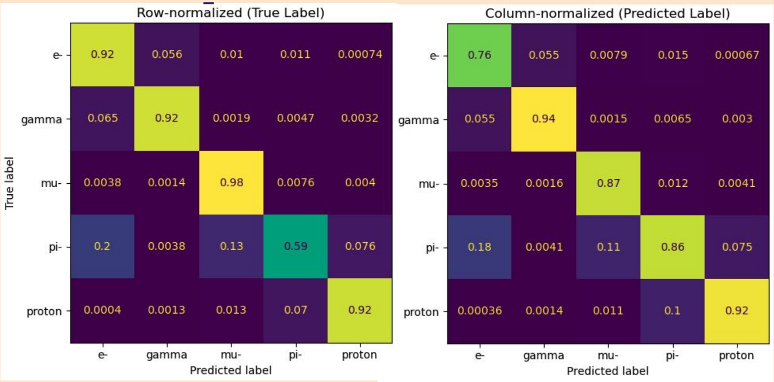
Detailed project: PID in the SFGD with a Transformer

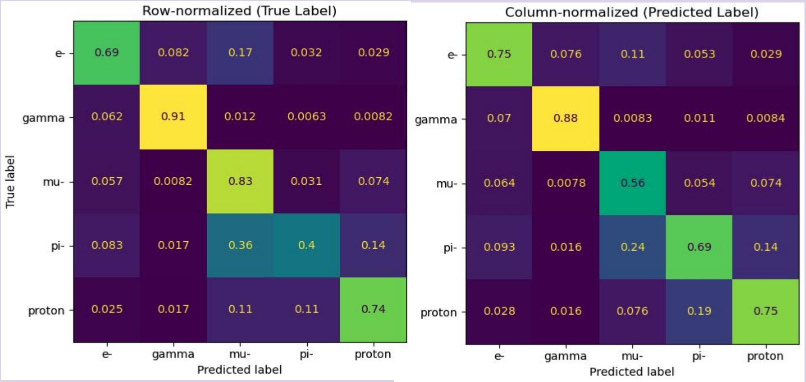
☺️ ***z-pgun*** *(train&test)*



😐 ***iso-pgun*** *(train&test)*

Test:

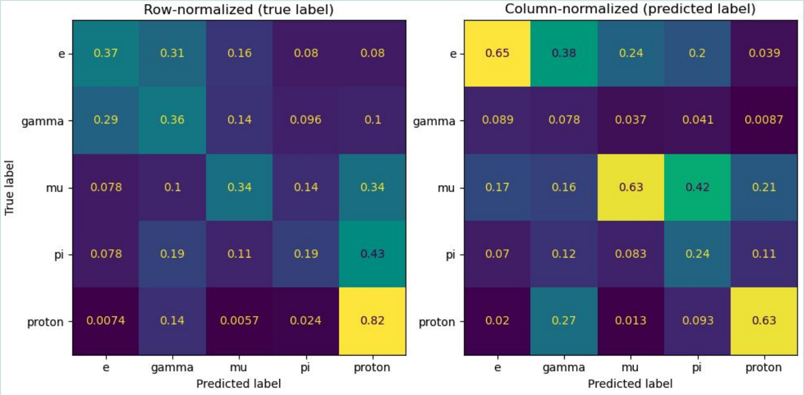




Test on neutrino interactions:

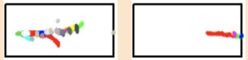
(iso-pgun training)

☹️

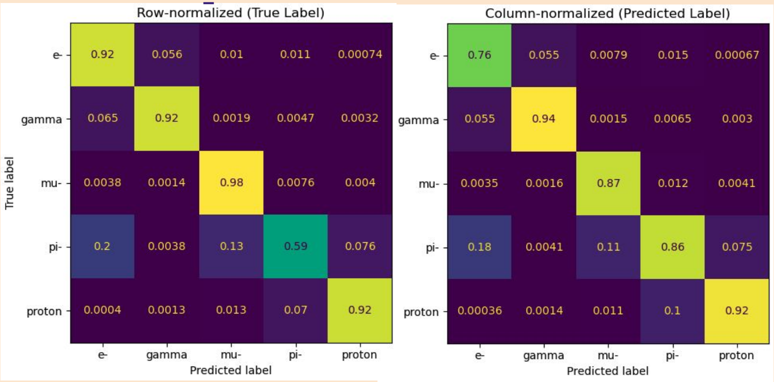


Detailed project: PID in the SFGD with a Transformer

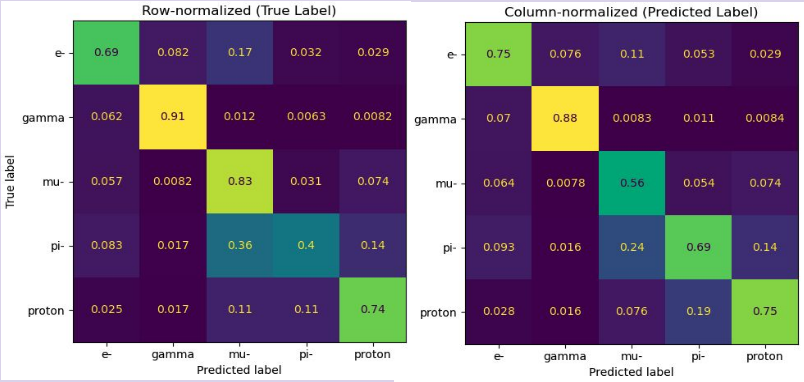
☺️ ***z-pgun*** *(train&test)*



Test:



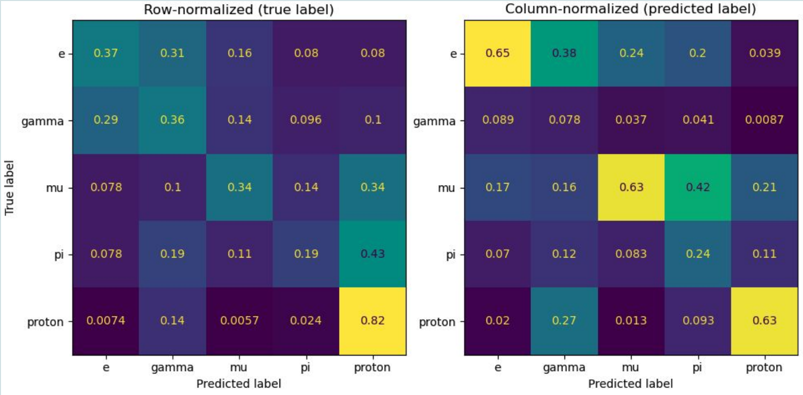
😐 ***iso-pgun*** *(train&test)*



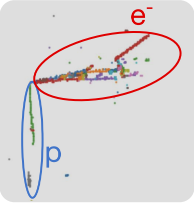
Test on neutrino interactions:

(iso-pgun training)

☹️



On-going improvements:



- refine pattern recognition before PID (if bad, then model cannot succeed)
- use time of each hits
- add e^+, π^+, μ^+ to training
- add extra token to the input sequence:
  - total nhits
  - total charge
  - pulls from nghb detectors (HAT/TPCs)

Summary

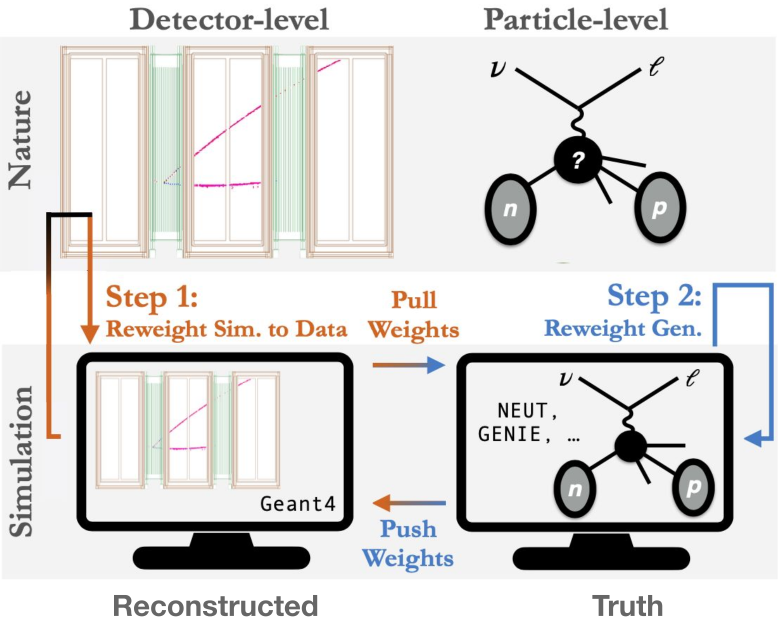
- Increasing number of ML projects at T2K ND
- In all stages of the experiment: more project in the analysis part, then reconstruction then simulation
- Methods are starting to be used in the official software (BDT, Transformer), paving the way for the other ones
- Challenges to come:
  - more friendly integration possibilities for these methods
  - strategy for systematic propagation

Back-up

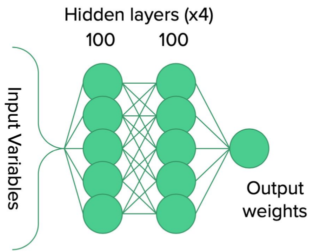
ref. https://arxiv.org/pdf/2504.06857v1

Unfolding of ND280 data: Omnifold

Working principle:



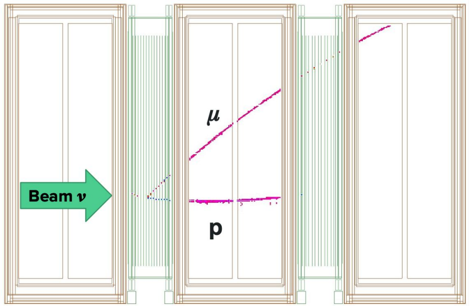
Archi:



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



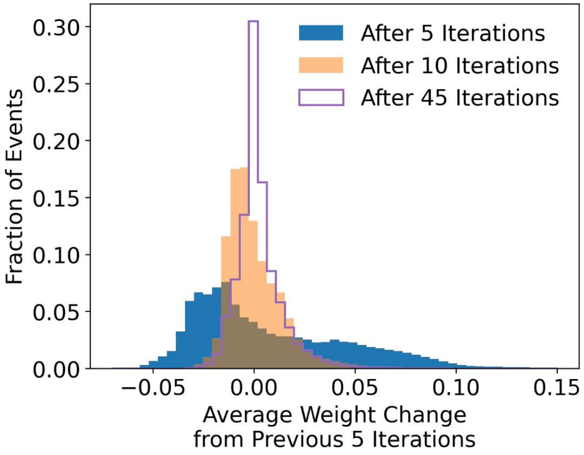
- *for test:* create fake dataset with a BeRPA-based modification to the true interaction rates

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)

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

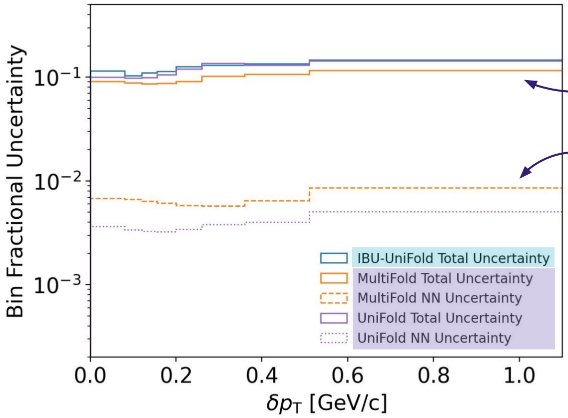
conventional-like method: IBU-UniFold, Binned UniFold

Omnifold variations (inputs choice): UniFold, MultiFold, OmniFold

	χ^2			
Method	$(p_\mu, \cos\theta_\mu)$ DoF=58	δp_T DoF=8	$\delta\alpha_T$ DoF=8	$\delta\phi_T$ DoF=8
Prior	298.2	2.3	5.9	4.9
IBU-UniFold	2.1	0.2	0.4	0.1
Binned UniFold	21.4	1.4	0.9	0.5
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

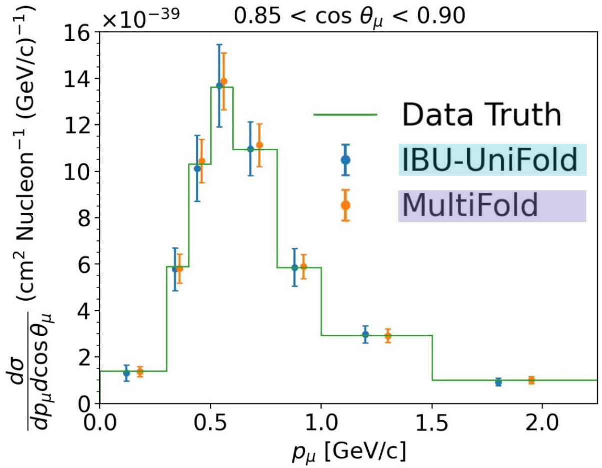
	Triangular Discriminator			
Method	$(p_\mu, \cos\theta_\mu)$	δp_T	$\delta\alpha_T$	$\delta\phi_T$
Prior	545.6	27.5	31.2	26.7
IBU-UniFold	17.1	1.9	3.4	0.8
Binned UniFold	29.9	2.8	6.0	1.9
UniFold	17.3	5.7	5.8	1.7
MultiFold	2.7	0.7	0.6	0.6
OmniFold	9.4	1.7	3.0	2.1

Uncertainties:



total

from NN stochasticness



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

Modeling posterior systematic: Normalizing Flows

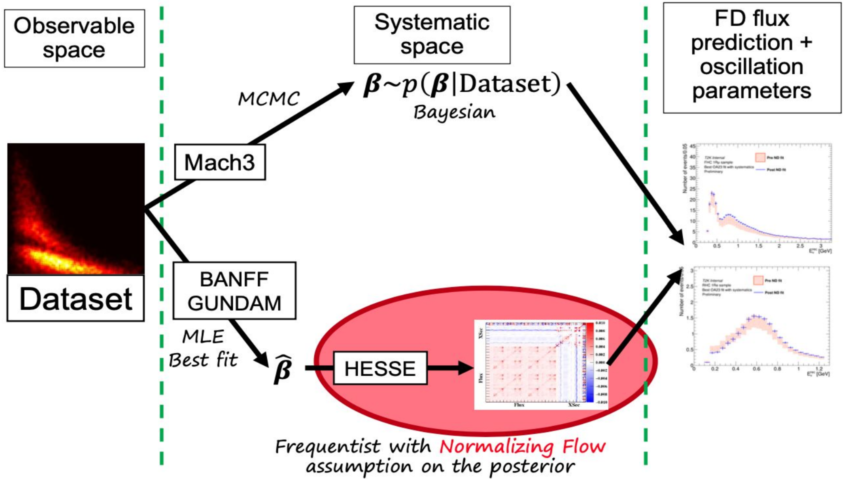
ND fit is a constrain on systematic uncertainties using ND280 observations

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

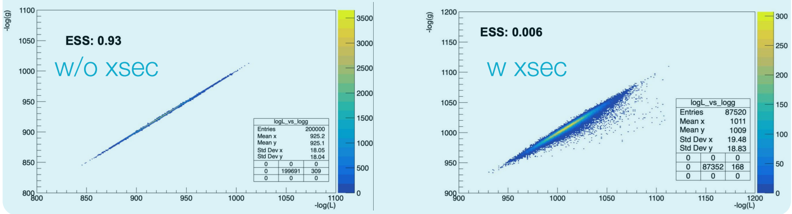
Goal: Learn the posterior probability distribution of neutrino flux binned in neutrino energy

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)



true posterior vs gaussian approx:

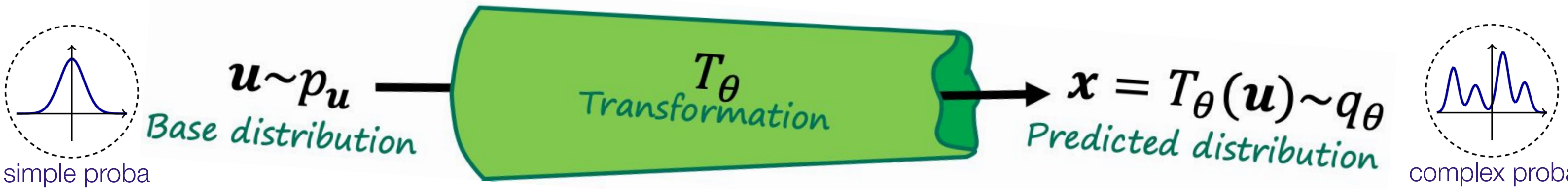


Where ML comes in:

replace *GUNDAM* gaussian approximation by something more complex to capture non-gaussianities in xsec params

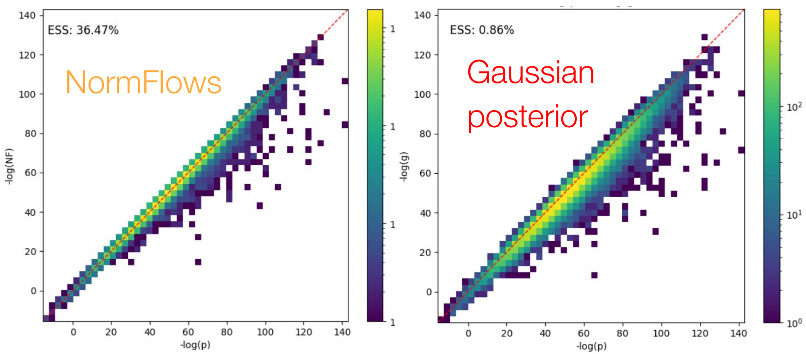
Modeling posterior systematic: Normalizing Flows

NormFlows:

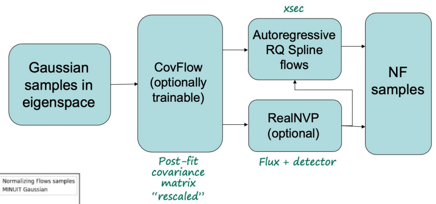


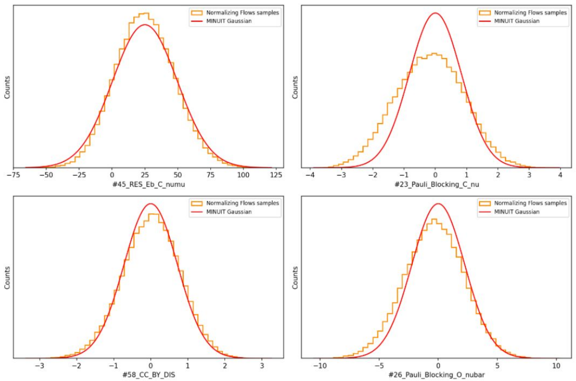
↳ straighforward to: sample $x = T_\theta(u) \sim q_\theta$ with $u \sim p_u$ & evaluate the proba $q_\theta(x) = p_u(u) \,|\det\left(J_{T_\theta}(u)\right)|^{-1}$

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



Archi: 5 RQ-NSF splines, MANN-parametrized flows





Fast method:
10M sample /day
vs ?

Modeling posterior systematic: Normalizing Flows

Questions:

- running time comparison with non-ML method?
- is loss func the KDL or derived from it?

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

CCQE cross-section sampling: Normalizing Flows

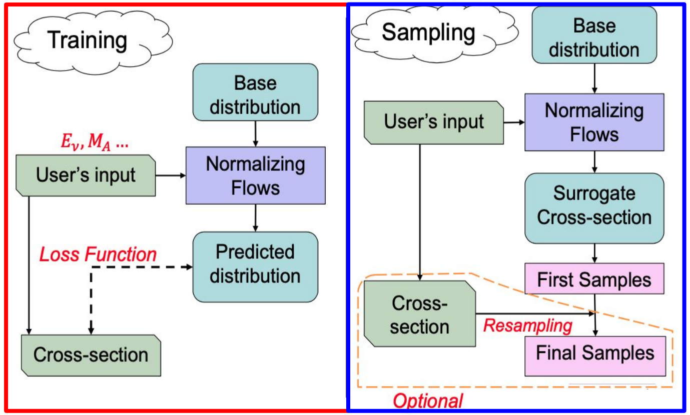
Goal: efficient MC sampling for CCQE exclusive cross-section of neutrino-nucleus (^{12}C)

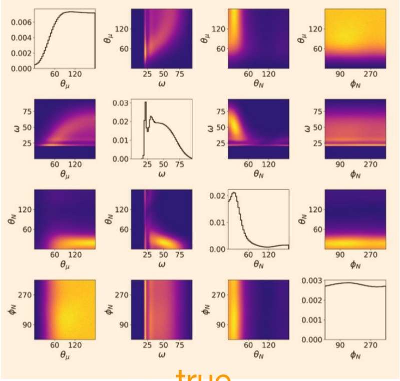
Conventional method:
use sophisticated nuclear models $\frac{d^6\sigma}{dE_\nu d\omega d\Omega_\mu d\Omega_N} \propto L^{\mu\nu}W_{\mu\nu} \rightarrow$ long computations

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

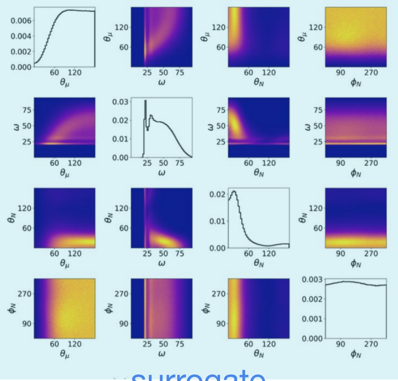
Train: to model 1p1h i.e. $p(\theta_\mu, \omega, \theta_N, \phi_N \mid E_\nu, \alpha)$ on 2 shells for many E_ν

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

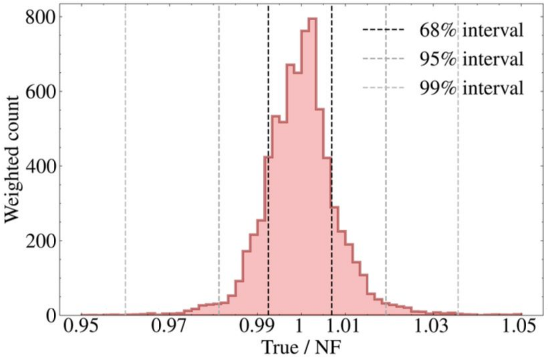




true



surrogate



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

Next:
2p2h will be more complicated because higher dimension