Transformer-based Approaches to Particle Identification in the T2K SuperFGD

Enhancing neutrino interaction studies through modern Machine Learning methods.

V. Kiseeva[1] on behalf of the T2K

1, Joint Institute for Nuclear Research

kiseevavi@jinr.ru

The T2K experiment

T2K- Tokai to Kamioka (Japan)

A long-baseline neutrino experiment designed to study how neutrinos change flavor as they travel from **J-PARC (Tokai)** to **Super-Kamiokande (Kamioka)**, 295 km away [1].

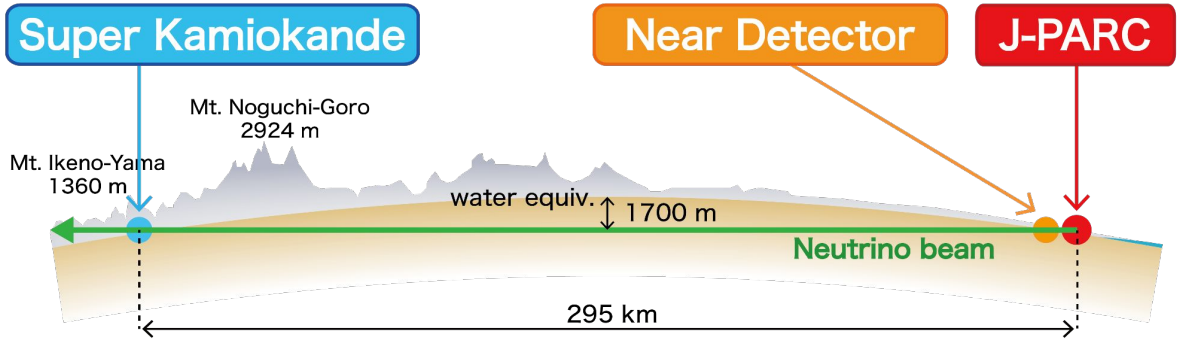


Fig.1 A schematic of a neutrino's journey from the neutrino beamline at J-PARC, through the near detectors (green dot) which are used to determine the properties of the neutrino beam, and then 295 km underneath the main island of Japan to Super-Kamiokande [1].

Neutrino Beam Production

30 GeV proton beam from J-PARC strikes a graphite target to produce π^+ and K^+, which are focused by magnetic horns into a 94 m helium-filled tunnel where they decay in flight **($\pi^+ \rightarrow \mu^+ + \nu_\mu$)** to generate a neutrino beam - or, in reversed magnetic polarity, an antineutrino - beam [2].
After the decay tunnel, all remaining hadrons and muons below 5 GeV/c are absorbed in the beam dump, ensuring that only neutrinos and high-energy muons continue forward.

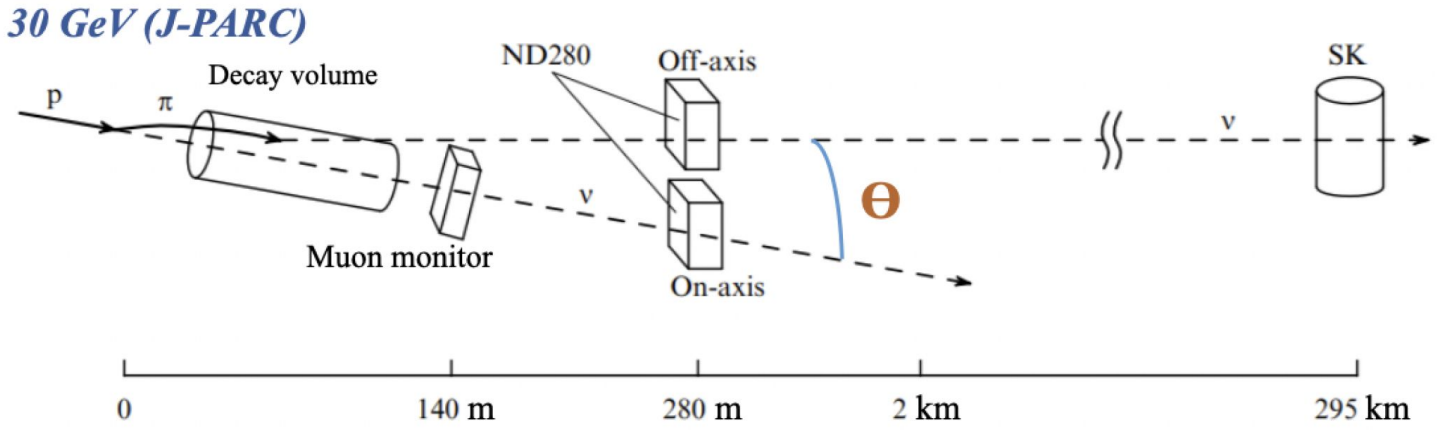


Fig.2 Schematic view of the T2K experiment [2].

Near Detector

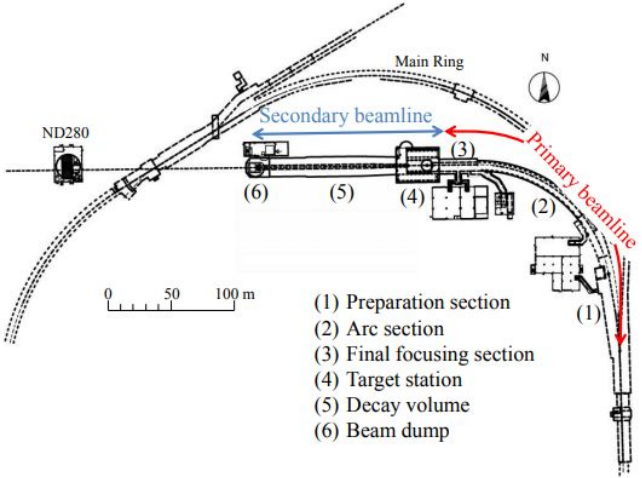


Fig. 3 Overview of the T2K neutrino beamline [1].

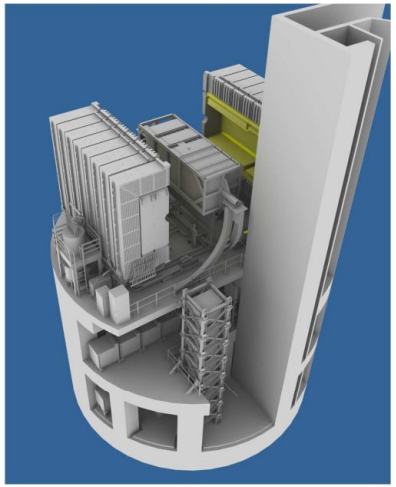


Fig. 4 Near detector complex [1].

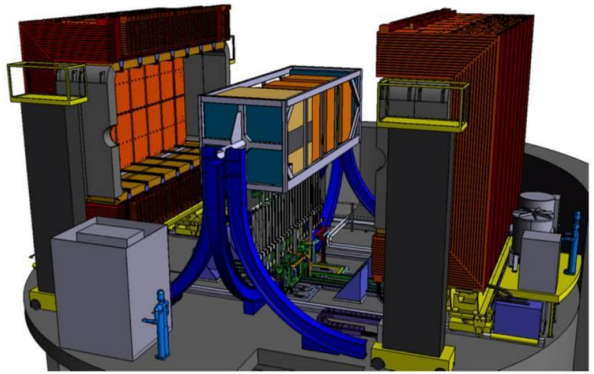


Fig.5 3D Model of the B1 floor of the ND280 pit [3].

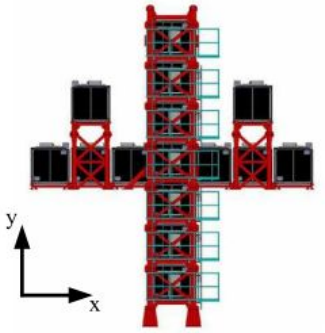


Fig. 6 INGRID on-axis detector [1].

Near detector complex, located 280m from the production target and consists of an on-axis beam monitor and an **off-axis detector (2.5°)**, which measures the **neutrino flux and composition** (flux, energy spectrum, and interaction rates) prior to oscillation toward the far detector.

Super-Fine-Grained Detector

The **Super-Fine-Grained Detector** (SFGD) in the T2K experiment identifies neutrino interactions, and thus their visible final-state particles such as protons, muons, and pions [2].

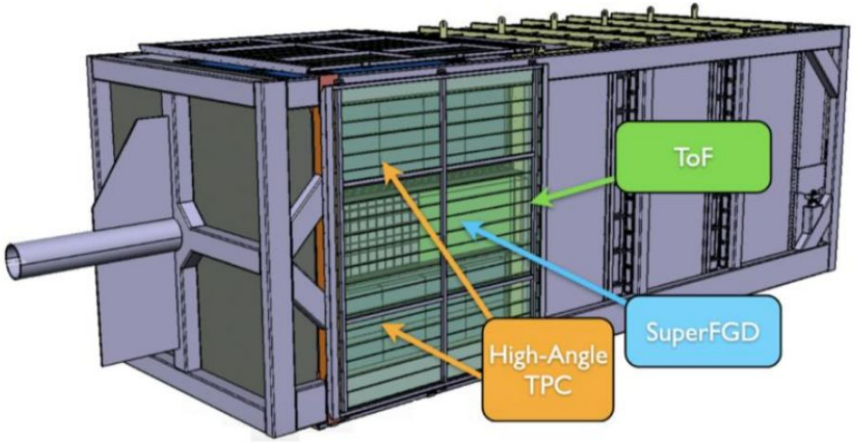


Fig.7 CAD 3D Model of the ND280 upgrade detector. In the upstream part (on the left in the drawing) two High-Angle TPCs with the scintillator detector Super-FGD in the middle.

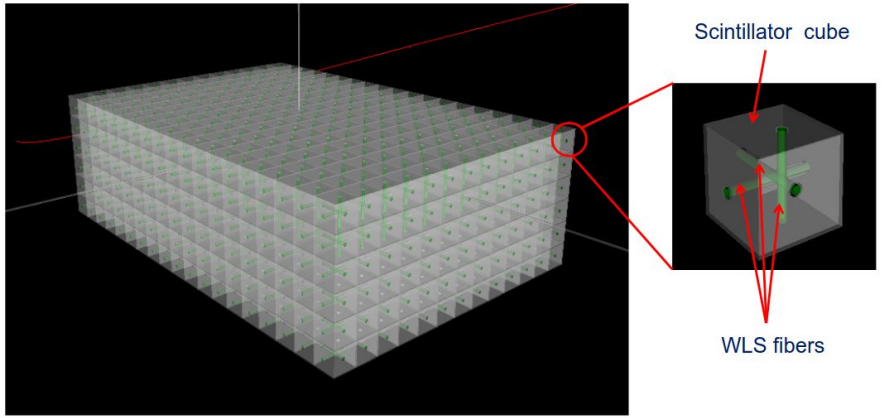


Fig.8 Schematic concept of the SFGD structure. Dimension of the active part is 192 × 184 × 56 cubes, the size of each cube is 1×1×1 cm3 [3].

Monte Carlo Data

after reconstruction

59,578 particle gun events

	TruePID	TrueMomentum	NNodes	NodeOrder	NodePosX	NodePosY	NodePosZ	NodeT	Nodededx	TrkLen	TrkEDepo	TrkDir1	TrkDir2
0	13	183.392853	24	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[562.3175048828125, 551.2816162109375, 541.041...	[-111.6478500366211, -115.0490493774414, -116....	[-1320.262451171875, -1309.0963134765625, -129...	[2407.756591796875, 2407.81689453125, 2407.876...	[165.22618103027344, 155.27743530273438, 476.1...	356.045837	9982.164062	0.654334	-0.231817

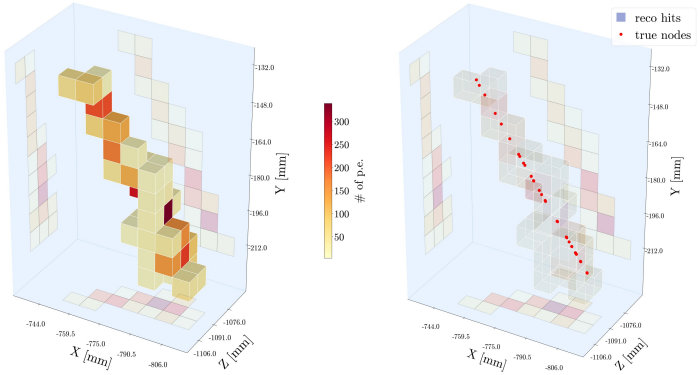


Fig.9 The 3D points of the events are stored in the form of nodes **corresponding with fitted positions after performing the track reconstruction** [4] .

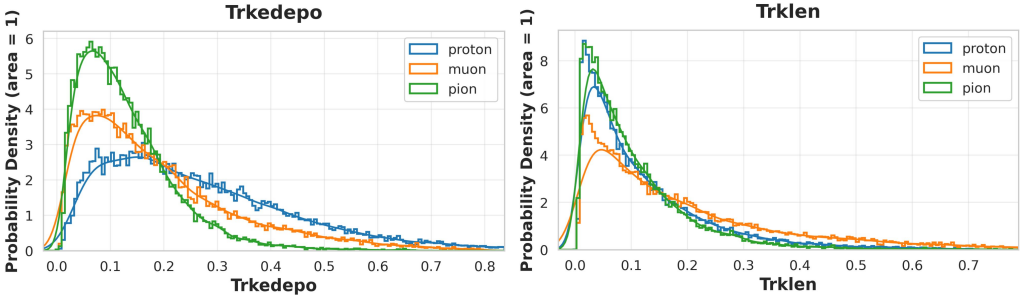


Fig.10 Examples of features distribution [4].

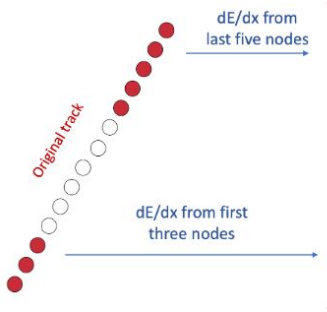
Monte Carlo Data

after reconstruction

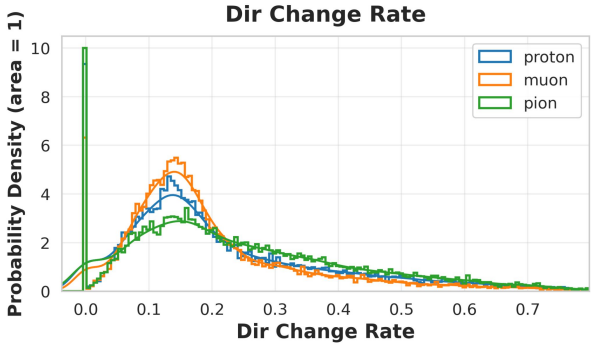
- TruePID: PDG code for particle identification (PID); 2212 (proton), 13 (muon), 211 (pion).
- TrueMomentum: **momentum** in MeV.
- NNodes: **number of nodes** of the event (3D spatial points).
- NodeOrder: **order of the nodes** within the event.
- NodePosX: **array with the coordinates** of the nodes along the **X**-axis (in mm).
- NodePosY: array with the coordinates of the nodes along the **Y**-axis (in mm).
- NodePosZ: array with the coordinates of the nodes along the **Z**-axis (in mm).
- NodeT: array with the timestamps of the nodes (in ms).
- Nodededx: array with **energy deposits of the nodes** (dE/dx).
- TrkLen: **length of the track** (in mm).
- TrkEDepo: **total track energy deposition** (in arbitrary unit).
- TrkDir1: **track direction**, **polar** angle (in degrees).
- TrkDir2: track direction, **azimuth angle** (in degrees).

Feature engineering

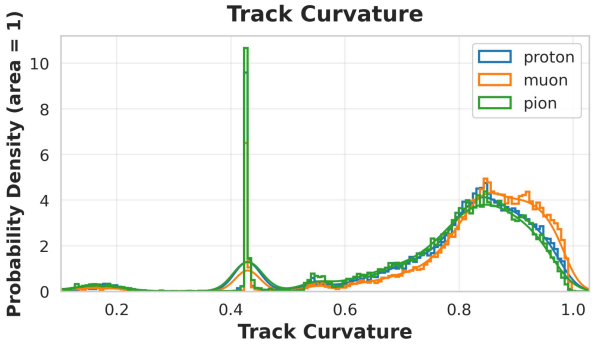
1-8. node_dE/dx_i
Nodes dE/dx
Keeps nodes dE/dx information



9. dEdx_mean
Average energy deposit
Avg. energy loss per unit length



10. track_curvature
Measures bending
Helps detect pions vs. muons in a magnetic field



node_distance (sequential)
Distance between points
Differentiates dense vs. sparse tracks

11. node_distance_max
Max distance between points
Differentiates dense vs. sparse tracks

12. node_distance_std
Avg. distance between points
Differentiates dense vs. sparse tracks

13. node_distance_mean
Std distance between points
Differentiates dense vs. sparse tracks

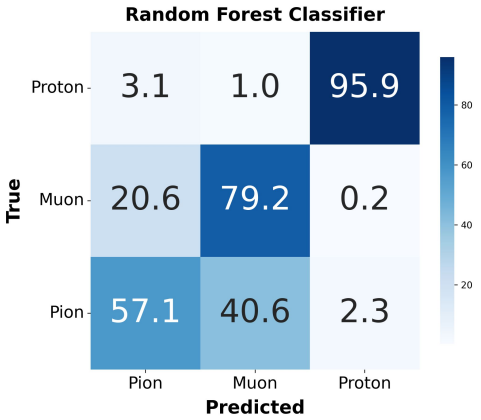
14. dEdx_std
The standard deviation of energy deposit
Variation in energy loss, detects interactions or noise

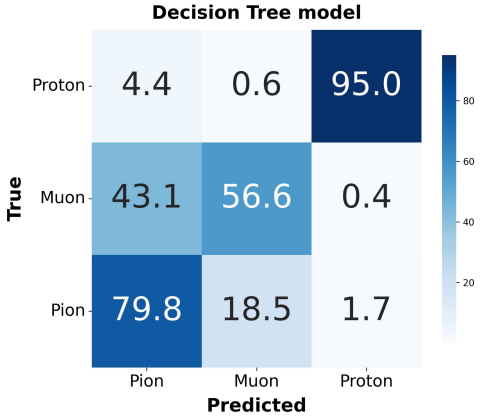
15. directional_change_rate
Frequency of turning angles
Helps separate linear vs. curved tracks

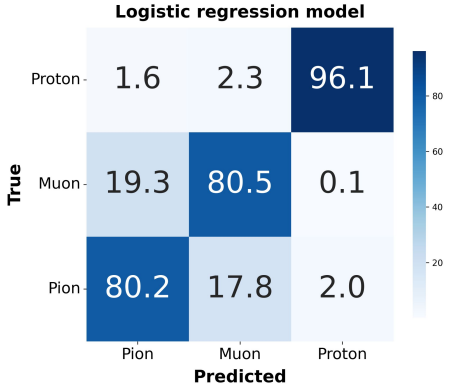
16. dEdx_min_max
Min energy deposit divided by max energy deposit

+Initial features **(20 scalar and 5 sequential in total)**

Baseline Models

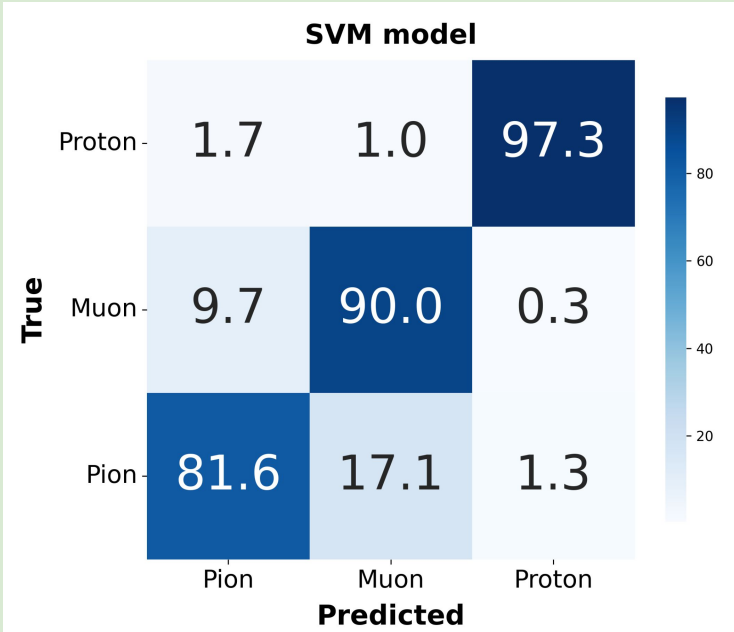






	precision	recall	f1-score
Proton	0.9869	0.9732	0.9800
Muon	0.8533	0.9001	0.8760
Pion	0.8559	0.8156	0.8353

Best Baseline Model

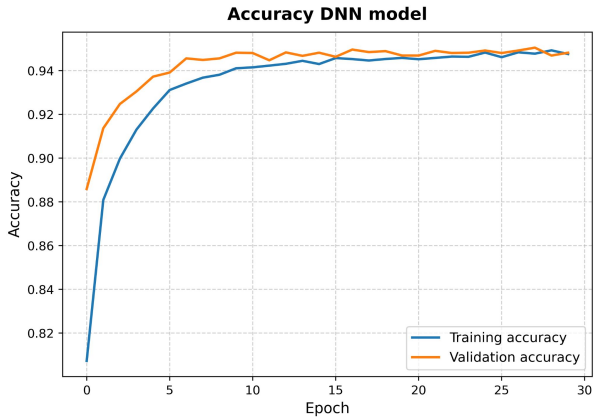


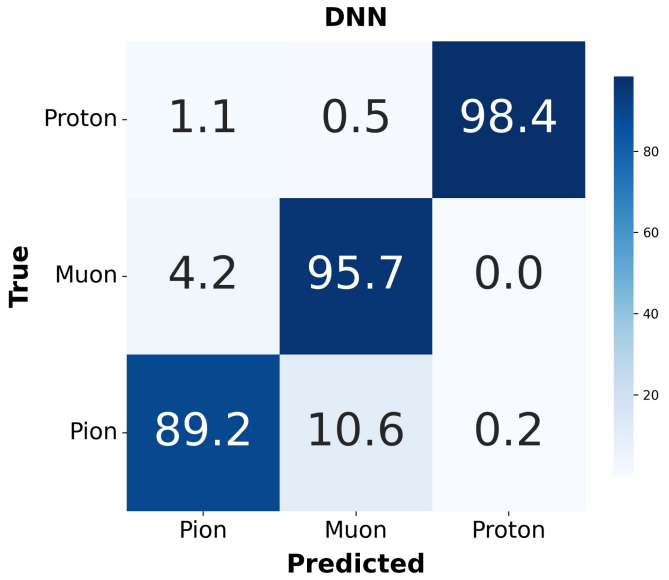
DNN (MLP) Model

Pre-processing:
Feature engineering →
All features are standardized (mean=0, std=1)

Baseline:
Input →
Dense(64, ReLU) →
Dropout(0.1)→
BatchNorm →
Dense(32, ReLU) →
Dense(3, Softmax)

Params: ~4K | Epochs: 30 | Optimizer: Adam |
Loss: Categorical CE





Transformer Model

Pre-train

[seq_input, mask_input]

▼

Encoder
— Dense(64)
— PositionalEnc
— Transformer blocks

▼

TimeDistributed(Dense(5))

▼

(MSE)

Sequential Branch

[seq_input, mask_input]

▼

Dense(64) (batch, 180, 5) → (batch, 180, 64)

PositionalEnc (+ 180×64 sin\cos)

Transformer Block × N (3 times)
1) MultiHeadAttention
2) FeedForward

Attention pooling MHA

seq_repr(64)

Scalar Branch

[scalar_input]

▼

Dense(64) →
Dropout(0,3) →
Dense(32)

▼

scalar_repr(32)

Concatenate

Concatenate

Dense(64, ReLU) → Dropout(0.3)

▼

Dense(3, softmax) → probability

1. Pre-processing

Input format: Monte-Carlo events.

Creating new variables:

node_distance (the distance between neighboring nodes).

dir_change_rate (average angular change of direction per unit length).

track_curvature (the curvature of the track in the magnetic field).

Scalar features: compute per-event aggregates (mean, std, max etc.).

Divide features into **sequential features** and **global scalar features**

Normalization: scale features per-feature (MinMaxScaler and standardize)

Padding & mask: pad sequences to max L = 180 nodes; produce boolean mask

Labels & split: TruePID→{0,1,2} → one-hot; split 80% train / 20% test, fix imbalance.

Pre-training phase (self-supervised)

Idea:
Train the encoder to reconstruct masked (20%) node features - forces it to learn local geometry and energy-loss (dE) correlations.

Architecture:

- Encoder: Dense(64) → PositionalEncoding → 3 × TransformerBlock(MHA + FFN)
- Reconstruction head: TimeDistributed(Dense(5))

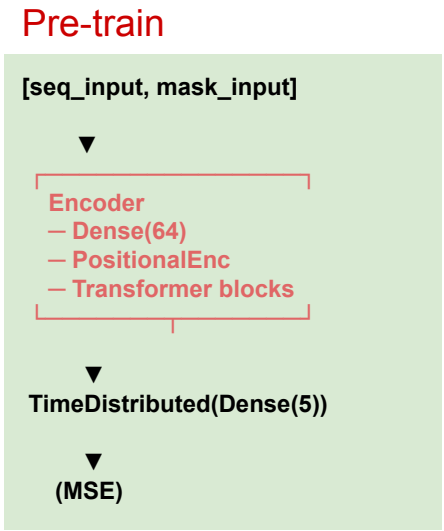
Input: (batch, 180, 5) — sequential detector data with padding mask.

Loss: Mean Squared Error (MSE) computed only for masked nodes.

Optimizer: Adam(lr=1e-3)

Epochs: 10

Output: Pre-trained encoder weights (encoder_recon_pretrained.weights.h5)



2.Reconstruction Pre-training

Simplified example of the pre-train process:

X0 event:

1	1	1	1
2	2	2	2
3	3	3	3
0	0	0	0
0	0	0	0

padding

…

pad_mask = [1,1,1,0,0,0]

X0 masked:

1	1	1	1
-1	-1	-1	-1
3	3	3	3
0	0	0	0
0	0	0	0

masked

padding

…

pretrain_mask = [False, True, False, False, False, False]

Prediction

1.8	2.1	2.0	1.9

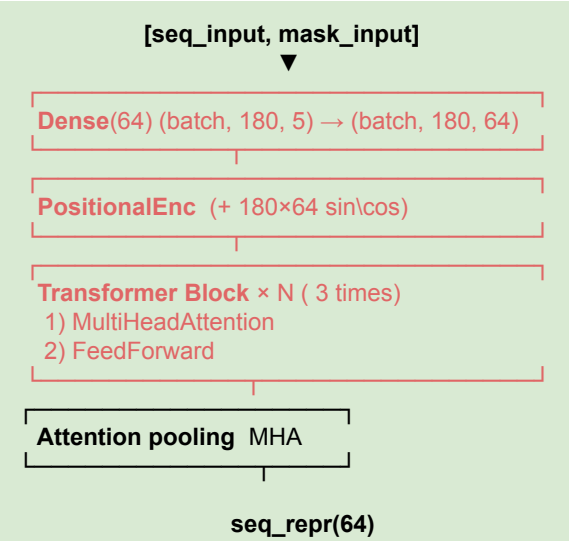
loss = mse(y_true, y_pred) * mask

For X[0] → MSE = 0.015

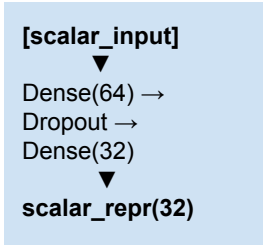
Total loss = ∑ loss / n

3.Train Model

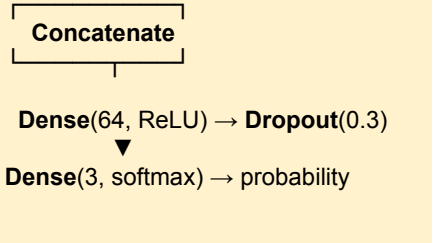
Sequential Branch



Scalar Branch



Concatenate



Transformer Model

Pre-processing

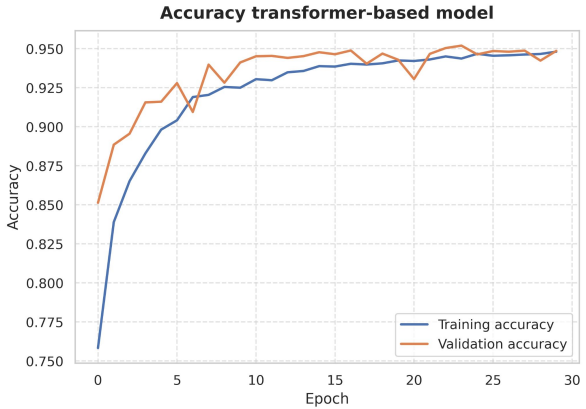
- Raw event → node sequence (≤180 nodes).
- Pad to length 180 + boolean mask.
- Compute scalar event features (mean, std, max, counts).
- Standardize all numeric features (mean=0, std=1).
- Split 80% train / 20% test (preserve class balance).
- Labels → one-hot.

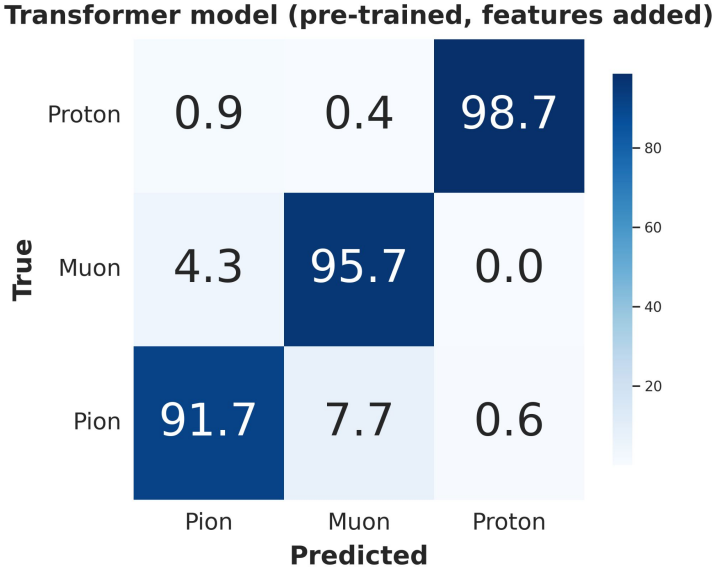
Pre-train (self-supervised reconstruction)

- Task: mask node features → reconstruct them (MSE).
- Encoder: Dense(64) → PositionalEncoding → 3× TransformerBlock (MHA + FFN).
- Head: TimeDistributed(Dense(5)) (per-node reconstruction).
- Train: Adam(1e-3), MSE, ~10 epochs; save encoder weights.
- Goal: learn robust node embeddings (geometry & dE patterns).

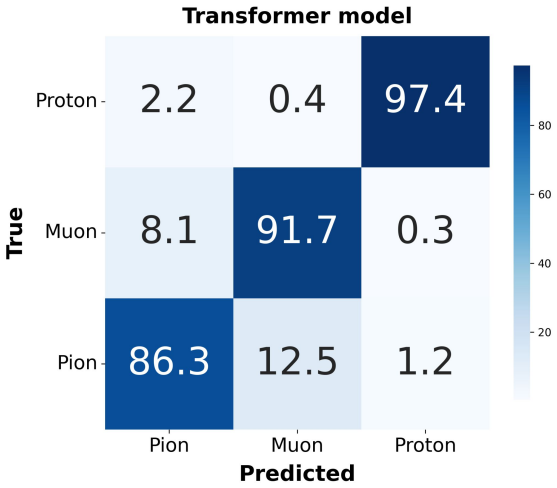
Train (classification)

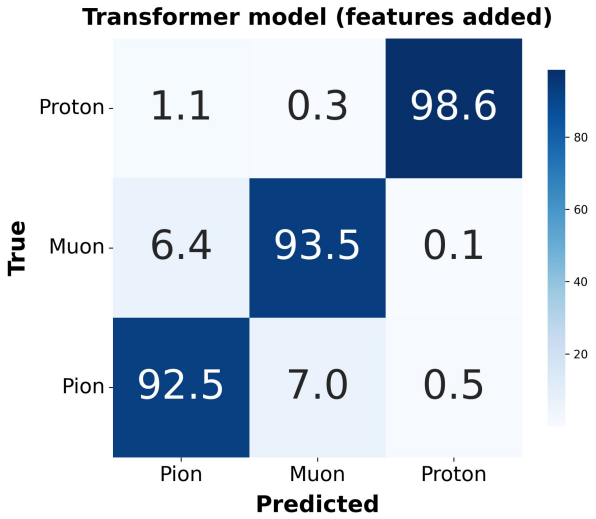
- Use pretrained encoder → token embeddings (batch,180,64).
- Attention Pooling → seq_repr (batch,64).
- Scalar branch: Dense(64) → Dropout(0.3) → Dense(32) → scalar_repr.
- Classifier: concat(seq_repr, scalar_repr) → Dense(64) → Dropout → Dense(3, softmax).
- Loss/opt: CategoricalCrossentropy(label_smoothing=0.1), Adam(1e-3, weight_decay=1e-5).
- Callbacks: EarlyStopping, ModelCheckpoint (monitor val_accuracy).
- Metrics: accuracy, per-class normalized confusion matrix.

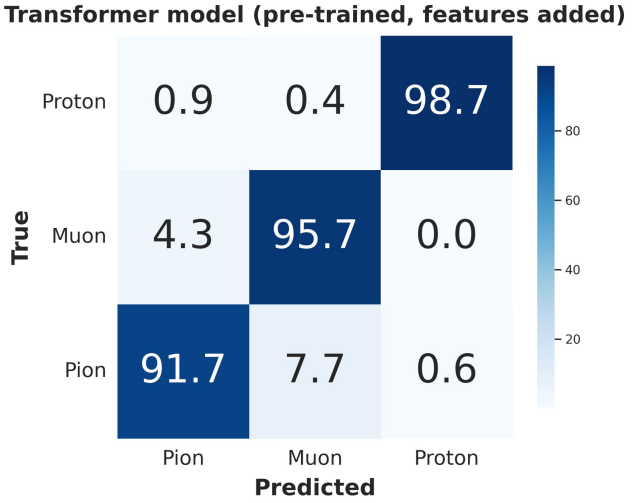




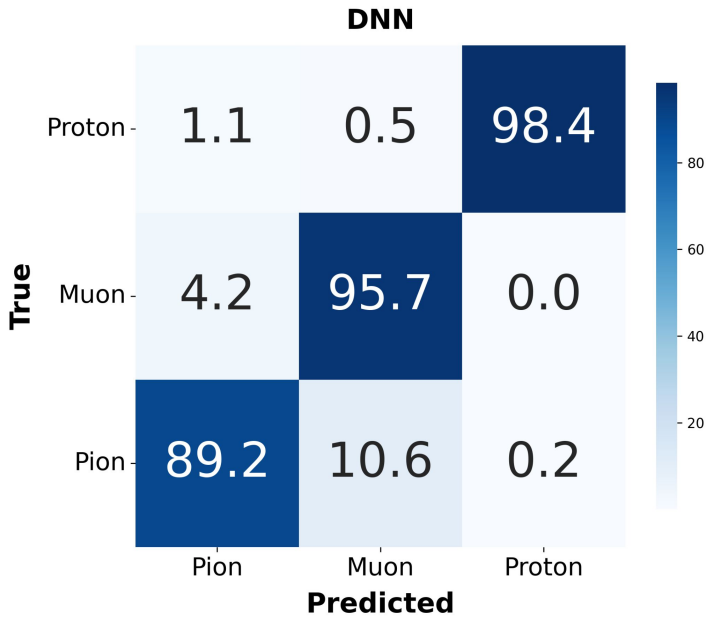
Transformer Models

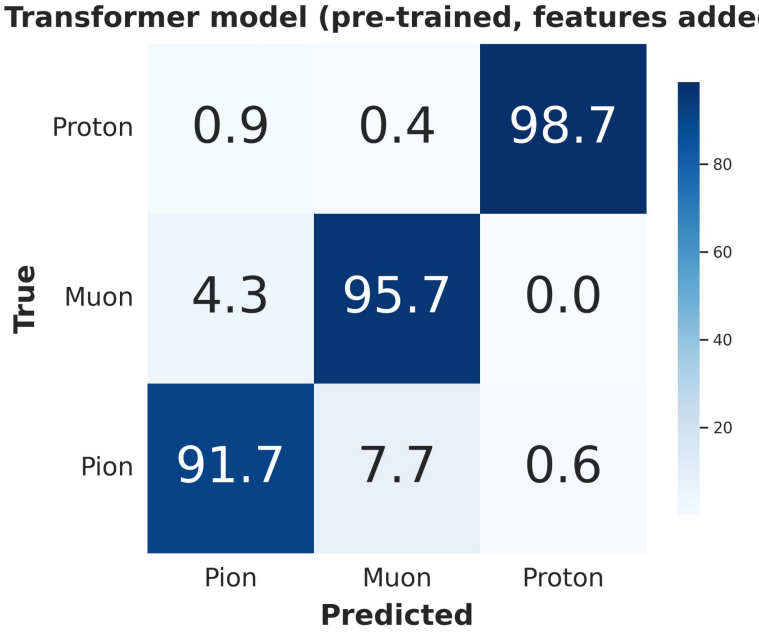






Compare Models





Conclusion

Transformers provided a **modest improvement**, but they handle sequences and node order, **generalize better** after pretraining, and are **expected to produce more stable predictions** on real data since they use all track information, exploiting global node-to-node correlations unlike Graph Neural Networks which are limited to local neighborhood aggregation.

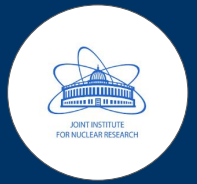
References

[1] K. Abe et al., (T2K Collaboration), The T2K experiment. Nucl. Instrum. Methods Phys. Res. Sect. A 659(1), 106–135 (2011). https://doi.org/10.1016/j.nima.2011.06.067

[3] T2K ND280 Upgrade – Technical Design Report, arXiv:1901.03750v2 [physics.ins-det], 14 Oct 2020, doi:10.48550/arXiv.1901.03750

[2] A. Dergacheva, et al, Current Status of the Novel 3D SuperFGD Detector for the T2K Experiment, Physics 2023, 5(3), 690–703, doi:10.3390/physics5030046

[4] S. Alonso Monsalve, T2K Masterclass: AI and Machine Learning in Neutrino Experiments, T2K Collaboration, https://t2k.org/young/young-doc/young-masterclass/masterclassAI.



Thank you!

Enhancing neutrino interaction studies through modern Machine Learning methods.

V. Kiseeva on behalf of the T2K

Backup

Transformer Model

Pre-train

[seq_input, mask_input]

▼

Encoder
— Dense(64)
— PositionalEnc
— Transformer blocks

▼

TimeDistributed(Dense(5))

▼

(MSE)

Sequential Branch

[seq_input, mask_input]

▼

Dense(64) (batch, 180, 5) → (batch, 180, 64)

PositionalEnc (+ 180×64 sin\cos)

Transformer Block × N (3 times)
1) MultiHeadAttention
2) FeedForward:
Dense(128, ReLU)
Dense(64)

Attention pooling MHA, (batch, 180, 64)→ (batch, 64)

seq_repr(64)

Scalar Branch

[scalar_input]

▼

Dense(64) → Dropout → Dense(32)

▼

scalar_repr(32)

Concatenate

Concatenate → vector 96

Dense(64, ReLU) → Dropout(0.3)

▼

Dense(3, softmax) → probability

Transformer Model

Pre-train

[seq_input, mask_input]

▼

Encoder
— Dense(64)
— PositionalEnc
— Transformer blocks

▼

TimeDistributed(Dense(5))

▼

(MSE)

Sequential Branch

[seq_input, mask_input]

▼

Dense(64) (batch, 180, 5) → (batch, 180, 64)

PositionalEnc add 180×64 sin\cos to every node

Transformer Block × N (3 times)
1) MultiHeadAttention:
Attention: Query·Key^T / √d → softmax, Σ (w * Value)
2) FeedForward:
Dense(128, ReLU)
Dense(64)

Attention pooling compute how strongly each node embedding (from the encoder) should contribute to the global event representation.
Using MHA, (batch, 180, 64)→ (batch, 64)

seq_repr(64)

Scalar Branch

[scalar_input]

▼

Dense(64) → Dropout → Dense(32)

▼

scalar_repr(32)

Concatenate

Concatenate(seq_repr(64), scalar_repr(32))
→ vector 96

Dense(64, ReLU) → **Dropout**(0.3)

▼

Dense(3, softmax) → probability: [p_pion, p_muon, p_proton]

Node Distance

Distance between consecutive nodes along the track:

$$d_i = \sqrt{((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2)}$$

Per-event statistics: mean, std, max

Intuition: captures step size and local energy loss density.

Direction Change Rate

How much the track direction changes between consecutive segments:

$$\Delta\theta_i = \arccos\left[((p_{i+1} - p_i) \cdot (p_i - p_{i-1})) / (|p_{i+1} - p_i| \cdot |p_i - p_{i-1}|) \right]$$

$$\text{Direction change rate} = (1 / L) \Sigma \Delta\theta_i$$

Intuition: larger angular changes → more scattering → likely muon or pion, not straight proton.

Track Curvature

Bending of the trajectory (important in a magnetic field):

$$\kappa = |v \times a| / |v|^3$$

- **v** — local velocity vector between nodes
- **a** — change of v (discrete derivative)

Intuition:

- High curvature → low-momentum charged particles
- Near-zero curvature → straight high-energy tracks

All variables were standardized (mean = 0, std = 1) and combined with global features for classification.

Transformer Model

Sequential Branch

[seq_input, mask_input]

▼

Dense(64) (batch, 180, 5) → (batch, 180, 64)

PositionalEnc add 180×64 sin\cos to every node

Transformer Block × N (3 times)
1) MultiHeadAttention:
Query = Dense(64), Key = Dense(64), Value = Dense(64),
Attention: Q·K^T / √d → softmax, Σ (w * Value)
2) FeedForward:
Dense(128, ReLU)
Dense(64)

Attention pooling compute how strongly each node embedding (from the encoder) should contribute to the global event representation. Using MHA, (batch, 180, 64)→ (batch, 64)

seq_repr(64)

Scalar Branch

[scalar_input]

▼

Dense(64) → Dropout → Dense(32)

▼

scalar_repr(32)

Concatenate

Concatenate(seq_repr(64), scalar_repr(32))
→ vector 96

Dense(64, ReLU) → **Dropout**(0.3)

▼

Dense(3, softmax) → probability: [p_pion, p_muon, p_proton]

Multi-Head Attention (MHA) operation

Multi-Head Attention computes how strongly each node embedding xi (from the encoder) should contribute to the global event representation.

Mathematical formulation

For each head(h=1,... ,H):

$$Q^{(h)} = qW_Q^{(h)}, \quad K^{(h)} = XW_K^{(h)}, \quad V^{(h)} = XW_V^{(h)}$$

$$A^{(h)} = \text{softmax}\left(\frac{Q^{(h)}\,K^{(h)^T}}{\sqrt{d_k}}\right), \qquad O^{(h)} = A^{(h)}\,V^{(h)}$$

Concatenate:

$$O = \text{Concat}\left(O^{(1)}, \ldots, O^{(H)}\right), \qquad \text{seq\_repr} = O\,W_O$$

Category	Description
Dataset	~60 000 Monte Carlo *particle gun* events
Event content	Number of nodes, 3D node coordinates (X, Y, Z), node energy deposits, total energy deposition, track length, polar (θ) and azimuthal (φ) angles
Derived variables	
Inter-node distances	Mean, standard deviation, and maximum
• Energy loss per unit	Mean (dE/dx) along the track