أكاديمية كاوست
KAUST ACADEMY

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Reproducible Machine Learning Pipelines for CMS TICL

Dhiyaa Madkhali, Felice Pantaleo, Wahid Redjeb

dhiyaa.madkhali@kaust.edu.sa, felice.pantaleo@cern.ch, wahid.redjeb@cern.ch

Abstract

We present **ticl-ml-pipelines**, a configuration-driven framework that converts CMS TICL NanoAOD inputs into validated machine-learning datasets and traceable training runs. The framework combines typed configuration, event-local ROOT loading, aligned Sim-to-Reco and Reco-to-Sim filtering, deterministic event-level splitting, application-owned preprocessing, executable feature contracts, reusable trainers, model artifacts, and MLflow provenance. The final project phase corrected the energy-regression representation from one row per reconstructed Trackster to one row per matched simulated particle. In a held-out diagnostic, this change improved R^2 from 0.565 to 0.931 and reduced RMSE from 661.4 to 264.6 GeV. An independently reproduced four-file demonstration produced 317 matched-particle rows with $R^2 = 0.945$, RMSE of 231.1 GeV, and MAE of 160.2 GeV. These values demonstrate the pipeline mechanics rather than production detector performance.The local suite reports 141 passed tests.

Date: August 24, 2026

Code: https://gitlab.cern.ch/dmadkhal/ticl-ml-pipelines

Contents

1 Introduction 2

2 Related Work 2

3 Methodology 3

3.1 System Overview . . . 3

3.2 Layered Configuration . . . 3

3.3 Event-Local Data Loading . . . 3

3.4 Bidirectional Matching . . . 4

3.5 Deterministic Splitting . . . 4

3.6 Application Preprocessing . . . 4

3.7 Feature Validation and Training . . . 4

3.8 Experiment Tracking . . . 5

4 Results 5

4.1 Test-Suite Verification . . . 5

4.2 Regression Row-Semantics Diagnostic . . . 5

4.3 Four-File Demonstration . . . 5

4.4 Changes Completed after Report 4 . . . 6

5 Discussion 6

6 Limitations and Future Work 7

7 Conclusion 8

References 8

1 Introduction

The Compact Muon Solenoid (CMS) experiment is preparing the High Granularity Calorimeter for the high-luminosity phase of the Large Hadron Collider. Its reconstruction chain includes the TICL framework, which builds higher-level objects from calorimeter information and records associations between simulated and reconstructed objects. These data are suitable for machine learning, but they are not immediately model-ready. Associations are stored as variable-length arrays, indices are local to each collision event, and one event may later expand into several training examples.

The central challenge is therefore not only to train a model, but also to preserve the scientific meaning of the data from the physical input branches to the final evaluation metric. A reliable pipeline must identify:

1. which physical collections and branches were processed;
2. which Sim-to-Reco and Reco-to-Sim associations were retained;
3. which event owns each train, validation, or test assignment;
4. what one model-ready row represents; and
5. which configuration, feature contract, implementation, and artifacts produced a result.

This project addresses these requirements through **ticl-ml-pipelines**, a reusable framework that separates framework-owned contracts from application-owned scientific choices. The framework owns configuration resolution, loading, matching orchestration, split provenance, feature validation, trainer discovery, and experiment recording. Applications own their physics selections, model-ready row representation, features, targets, models, metrics, artifacts, and promotion policies.

The completed project provides:

- layered and typed configuration resolved before ROOT processing;
- event-local Arrow data with aligned bidirectional associations;
- deterministic splitting before application row expansion;
- preprocessing for PID/PFN and particle-level energy regression;
- executable `FeatureSpec` validation for all dataset splits;
- reusable trainers with deterministic seeds, evaluation, artifacts, and application-specific champion rules; and
- consolidated user and configuration documentation for project hand-off.

The evaluation in this report focuses on framework correctness and a small energy-regression demonstration. It does not claim a production PID/PFN benchmark or final detector-performance result.

2 Related Work

TICL was developed to reconstruct particles and showers in an imaging calorimeter by combining local calorimeter objects into higher-level candidates [1]. The TICLNanoValidation project provides reference examples for reading TICL NanoAOD collections and reconstructing association structures [2].

The present project builds on these ideas but focuses on a reusable machine-learning boundary: configuration resolution, typed intermediate data, deterministic split ownership, application-defined row semantics, training contracts, and experiment provenance.

Apache Arrow and Apache Parquet provide the columnar interchange layer between pipeline stages [3, 4]. Arrow preserves nested list structures without forcing them into rectangular arrays, while Parquet provides a portable on-disk representation for train, validation, and test datasets.

PyTorch supplies the model-training layer [5], while MLflow provides experiment records for resolved configuration, metrics, lineage, and model artifacts [6]. The distinctive contribution of this project is the integration of these tools around TICL-specific event and association semantics.

Compared with notebook-specific workflows, the framework makes the interpretation of the data explicit and repeatable. Compared with a generic training framework, it preserves source-file identity, event-local indices, reciprocal matching information, and preprocessing versions required for detector-data applications.

3 Methodology

3.1 System Overview

The final execution path is:

Layered YAML → ROOT loading → Arrow/Parquet → Bidirectional matching → Event splitting and preprocessing → FeatureSpec validation → Registered trainer → Metrics and model artifacts.

Each transition is a defined and validated hand-off. This design allows PID/PFN preparation and energy regression to share the same framework while retaining different scientific row definitions and model implementations.

3.2 Layered Configuration

A run is assembled by deep-merging four configuration layers:

1. **Base configuration:** reusable framework defaults, including thread, split, and training settings.
2. **Offline adapter:** physical collection names, branch templates, data types, and association maps.
3. **Application configuration:** requested fields, matching cuts, dataset identity, preprocessing, trainer, and experiment names.
4. **Runtime overrides:** temporary settings such as thread count, multithreading, and Parquet chunk size.

Nested mappings are preserved during the merge, while later scalar and list values replace earlier values. The merged configuration is converted into typed `RunConfig` objects before ROOT processing begins.

3.3 Event-Local Data Loading

The loader produces one Arrow row per collision event. Before application-specific fields are added, the stable contract contains 14 columns covering:

- source identity: `file`, `run`, `luminosityBlock`, and `event`;
- reconstructed Tracksters, simulated Tracksters, LayerClusters, and Trackster-to-cluster membership; and
- both association directions, represented by aligned `Index`, `Score`, and `SharedEnergy` arrays.

Configured fields are appended with stable prefixes such as `Trackster_`, `SimTrackster_`, and `Cluster_`. The loader reconstructs nested associations from the flat NanoAOD arrays and validates that indices, scores, and shared energies remain positionally aligned.

3.4 Bidirectional Matching

The matching stage independently filters Sim-to-Reco and Reco-to-Sim associations. When a candidate fails a cut, the index, score, and shared-energy values at that position are removed together. Event rows and unrelated columns remain unchanged.

The stage does not recompute detector scores, reorder candidates, or force a unique best match. Matching expressions are treated as trusted configuration because the current implementation evaluates them as Python expressions.

Independent Monte Carlo files may repeat the same `(run, luminosityBlock, event)` tuple. Source-file identity is therefore included in event provenance.

3.5 Deterministic Splitting

Each complete event is assigned to train, validation, or test before the application expands it into model-ready rows. Assignment uses BLAKE2b over the dataset identifier, split seed, source file, run, luminosity block, and event.

This approach ensures that:

- all examples derived from one event remain in the same split;
- file reordering does not change event ownership; and
- independent files containing repeated event numbers remain distinct.

3.6 Application Preprocessing

Applications define the meaning of one model-ready row:

- **PID:** one reciprocal simulated/reconstructed pair with particle labels and Trackster-level features;
- **PID-PFN:** one reciprocal pair with a fixed $50 \times 10 \times 7$ cluster tensor; and
- **Energy regression version 2:** one simulated particle whose reciprocal Trackster raw energies are summed, while the contributing `reco_indices` are retained.

The regression preprocessing version was increased from 1 to 2 when the row meaning changed. Training rejects datasets carrying the older metadata rather than silently accepting an incompatible representation.

3.7 Feature Validation and Training

Each trainer declares a versioned `FeatureSpec` containing the required Arrow columns, exact data types, target column, and expected preprocessing metadata. The runner validates the train, validation, and test splits before model code is executed.

A trainer implements four required methods:

- `feature_spec()` declares the input contract;
- `train(train, validation, context)` fits and returns a model;
- `evaluate(model, test, context)` returns numeric test metrics; and
- `save(model, directory)` writes model artifacts.

An optional `is_champion(metrics)` method places promotion policy with the application. Regression uses an R^2 threshold, while PID uses grouped accuracy.

3.8 Experiment Tracking

The training runner records the resolved configuration, `FeatureSpec`, dataset lineage, trainer identity, random seeds, training metrics, held-out test metrics, champion decision, and model artifacts in MLflow.

Replay verifies that the recorded trainer still resolves to the expected implementation and compares reproduced metrics using an absolute tolerance of 10^{-6}. Replay currently depends on the recorded dataset path, and a completed replay of every legacy run has not been verified.

4 Results

4.1 Test-Suite Verification

The final repository verification ran the complete local test suite and a focused regression suite. Table 1 separates passing tests, environment-dependent skips, and the two stale failures present at Report 4.

Table 1 Recorded test-suite progression.

Checkpoint	Passed	Skipped	Failed
Report 3	92	4	0
Report 4	115	5	2
Final verification	**141**	**0**	**0**

The final run also emitted five non-failing warnings. The focused regression suite reported 14 passed tests.

4.2 Regression Row-Semantics Diagnostic

The initial energy-regression representation used one row per reconstructed Trackster. When several Tracksters were associated with the same simulated particle, this duplicated the target energy and changed the meaning of the learning problem.

A held-out SWAN diagnostic compared that representation with one row per simulated particle, using the sum of reciprocal Trackster raw energies as the input.

Table 2 Held-out row-semantics diagnostic. This is separate from the four-file demonstration in Table 3.

Row definition	RMSE (GeV)	R^2
One row per Trackster	661.419	0.564858
One row per simulated particle	**264.634**	**0.930599**

The particle-level representation reduced RMSE by 396.785 GeV and increased R^2 by 0.365741 on the same held-out split. This result directly motivated regression preprocessing version 2.

4.3 Four-File Demonstration

The final pipeline was independently reproduced on four local ROOT samples. Because PyROOT was unavailable in the verification environment, ROOT input was read with uproot and Awkward.

The repository matching and preprocessing stages were used downstream, and all 310 events passed the repository event validator.

Table 3 Independently reproduced four-file pipeline demonstration.

Quantity	Value
Events	310
Simulated objects	320
Reconstructed Tracksters	12,844
Sim-to-Reco links, before → after	12,842 → 466
Reco-to-Sim links, before → after	12,842 → 12,825
Matched-particle rows	317
RMSE	231.088 GeV
MAE	160.242 GeV
R^2	0.945229

The fitted linear relationship was:

$$E_{\text{target}} = 1.382020979\, E_{\text{raw}} + 125.818555\ \text{GeV}. \tag{1}$$

This result is a quantified framework demonstration. It is not a production detector-performance benchmark, a PID/PFN benchmark, or a complete PyROOT/RDataFrame validation.

4.4 Changes Completed after Report 4

The final project phase delivered the following changes:

- corrected regression preprocessing to one row per matched simulated particle;
- increased the regression preprocessing version from 1 to 2;
- added a reusable linear-regression command-line workflow;
- replaced the global promotion threshold with trainer-owned champion policies;
- simplified the linear trainer from 246 to 169 lines;
- retained raw MSE in GeV^2, RMSE in GeV, R^2, slope, and intercept;
- repaired stale toy paths, optional dependency behavior, and regression configuration tests; and
- rewrote the README and configuration reference and removed the obsolete training notebook.

5 Discussion

The main project result is the traceable relationship between data semantics and model metrics. A resolved configuration identifies the physical inputs; the loader reconstructs event-local structures; matching preserves positional alignment; deterministic splitting prevents rows from one event crossing dataset boundaries; `FeatureSpec` rejects incompatible inputs; and the experiment record stores the implementation identity and resulting artifacts.

The regression correction demonstrates why these contracts matter. The one-row-per-Trackster baseline was mechanically valid but scientifically ambiguous because several rows could repeat the same simulated target. Aggregating reciprocal Trackster energies once per simulated particle aligned the identities of the input and target.

The improvement from $R^2 = 0.565$ to $R^2 = 0.931$ in the held-out diagnostic indicates that the original row semantics weakened the learning problem. This value should not be compared directly with the four-file result of $R^2 = 0.945$, which came from a separate reproduction and dataset composition.

The four-file demonstration confirms that the corrected representation can flow through matching, preprocessing, Parquet hand-offs, training, and evaluation with internally consistent counts and metrics. Its value is architectural rather than a claim of production physics readiness. The sample is too small for detector-performance conclusions, and the verification environment used uproot and Awkward instead of the intended PyROOT input path.

The framework/application boundary also improved maintainability. A small linear trainer and the larger PID/PFN application use the same orchestration and validation interfaces without forcing the framework to understand their scientific row definitions. Moving champion selection into the trainer also removed an inappropriate global accuracy rule and allowed each application to use metrics appropriate to its objective.

Finally, the increase from 92 passing tests at Report 3 to 138 at the final hand-off reflects broader coverage of configuration, schemas, splitting, training, artifacts, replay safeguards, and examples. The two failures recorded at Report 4 were stale test-path problems rather than model failures, but correcting them was necessary to restore a trustworthy runnable suite.

6 Limitations and Future Work

The completed framework has several important limitations.

- **No production PID/PFN benchmark.** Current PID/PFN evidence is limited to synthetic or focused smoke tests. Future work should use representative production data and report class-wise efficiency, purity, confusion matrices, calibration, and uncertainty estimates.
- **Limited regression scale.** The reported regression demonstration contains four files and 317 matched-particle rows. Larger studies should compare several baselines and report energy-binned response, residuals, and statistical and systematic uncertainties.
- **CERN runtime not validated in this report.** Full ROOT/RDataFrame execution, EOS access, Kerberos authentication, CERN MLflow publication, and Kubernetes CI require the supported CERN Linux environment.
- **Memory scaling.** ROOT data are materialized before Parquet chunking, training concatenates each complete split in memory, and lineage conversion creates an additional pandas copy. Streaming and bounded-memory processing should be evaluated.
- **Tracking cannot be fully disabled.** The `training.mlflow.enabled` field is parsed and documented but is not yet a complete disable switch. A genuine local or no-tracking mode should be implemented.
- **Replay portability.** Replay depends on the recorded dataset path and uses one absolute metric tolerance of 10^{-6}. Content-addressed datasets and per-metric tolerances would make replay more durable.
- **Trusted matching expressions.** Matching cuts are evaluated as Python expressions and must therefore come from trusted configuration. A restricted expression language would provide a safer boundary.
- **Production-specific maintenance.** The fixed PID geometry lookup must remain versioned with its production context. ONNX export should move to the newer `torch.export` path, and missing-association behavior should be made explicit.

The recommended next steps are:

1. reproduce the complete data path in the supported CERN environment;
2. benchmark PID/PFN and regression applications at representative scale;
3. quantify statistical and systematic uncertainties; and
4. harden streaming, tracking, expression evaluation, and replay policies.

7 Conclusion

This project developed **ticl-ml-pipelines**, a reproducible framework for transforming CMS TICL NanoAOD data into validated machine-learning datasets and traceable training runs. The completed system combines layered configuration, event-local data loading, aligned bidirectional matching, deterministic event-level splitting, application-defined preprocessing, executable feature contracts, reusable trainers, model artifacts, and MLflow experiment tracking.

The final project phase corrected the energy-regression representation from one row per reconstructed Trackster to one row per matched simulated particle. This correction improved the held-out diagnostic from $R^2 = 0.565$ and an RMSE of 661.4 GeV to $R^2 = 0.931$ and an RMSE of 264.6 GeV. An independent four-file demonstration produced 317 matched-particle rows with $R^2 = 0.945$, an RMSE of 231.1 GeV, and an MAE of 160.2 GeV. These results validate the pipeline mechanics but should not be interpreted as production detector-performance benchmarks.

The final local test suite reported 141 passed tests. The framework is therefore ready for project hand-off and further scientific development. The next milestone is production-scale validation using representative datasets and the supported CERN execution environment. "'

References

[1] Alessio Di Pilato et al. "Reconstruction in an Imaging Calorimeter for HL-LHC". In: *arXiv preprint arXiv:2004.10027* (2020). arXiv: 2004.10027 [physics.ins-det]. URL: https://arxiv.org/abs/2004.10027.

[2] Wahid Redjeb. *TICLNanoValidation*. GitHub repository. Accessed 2026-08-24. URL: https://github.com/waredjeb/TICLNanoValidation.

[3] Apache Software Foundation. *Apache Arrow: A Cross-Language Columnar Data Format*. Accessed 2026-08-24. URL: https://arrow.apache.org/.

[4] Apache Software Foundation. *Apache Parquet: An Open Source, Column-Oriented Data File Format*. Accessed 2026-08-24. URL: https://parquet.apache.org/.

[5] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[6] Matei Zaharia et al. "Accelerating the Machine Learning Lifecycle with MLflow". In: *IEEE Data Engineering Bulletin* 41.4 (2018), pp. 39–45.