TrackML High Energy Physics Tracking Challenge

Kanishk Bakshi [2025280440][1], Mac' O Brien [2025280442][2], and Sebastian Jonathan [2025280072][3]

[1,2,3]Team 5 (Alto), Department of Computer Science and Technology, Tsinghua University
[1]kanishkbakshi@zohomail.cn
[2]mac.w.obrien@gmail.com
[3]pfr25@mails.tsinghua.edu.cn

ABSTRACT

We address the problem of reconstructing charged-particle trajectories from detector hit data. The task is challenging due to high hit density, missing or spurious measurements, and non-linear (helical) particle trajectories in a magnetic field. In this work, we implement and compare three complementary approaches: (1) a Graph Neural Network (GNN) implementation that frames hits and candidate links as a graph to learn global association patterns, (2) Method 1 a hybrid, engineering-focused pipeline that combines heuristic candidate generation with supervised pruning and local fitting, and (3) Method 2 a classical geometric pipeline that builds tracks by iteratively forming pairs, extending to triples, fitting helices, augmenting duplicates, and performing a greedy conflict-resolution assignment using an outlier-density based scoring metric. All three approaches are evaluated on the same hit-level test dataset using. The result shows that method 2 gives the best accuracy among the three approaches.

Introduction

Reconstructing particle trajectories from discrete detector hits is a fundamental task in experimental high-energy and nuclear physics and in any application that requires recovering continuous trajectories from sparse, noisy measurements. A typical detector produces many spatially localized hits per event; tracks correspond to ordered subsets of those hits that lie along approximately helical paths in the presence of a magnetic field. The principal challenges are (i) high combinatorics when proposing candidate associations between hits, (ii) measurement noise and duplicates, (iii) short or highly curved tracks that are difficult to distinguish from random hit clusters, and (iv) the need for computationally efficient algorithms that scale to large numbers of events.

This report documents three distinct approaches we developed and evaluated on the same dataset. The three methods represent different points in the design space between classical geometric processing and modern learned association models:

1. **Graph neural Network:** We cast the hit set and candidate links as a graph and train a GNN to predict link probabilities and to assemble links into track hypotheses. The GNN is intended to learn global context and non-local dependencies that are difficult to encode with hand-designed heuristics. The implementation and training scripts are provided in the attached GNN code.

2. **Method 1 (Hybrid Pipeline):** In this method, we implement a hybrid pipeline that combines engineered candidate-generation heuristics with supervised AI-based pruning models to score and prune hit pairs. Track hypotheses are formed by constructing and traversing adjacency graphs of high-scoring hit pairs, rather than relying on explicit parametric curve fitting. This method prioritizes a compact candidate set and leverages learned components where they provide the largest marginal benefit.

3. **Method 2 (Classical Geometric Pipeline):** Method 2 is a carefully optimized geometric algorithm that proceeds in stages: select promising hit pairs on adjacent layers, extend pairs into triples via geometric extrapolation, fit helices to triples and extend those fits to full tracks, add nearby duplicate hits per layer, and finally perform greedy conflict resolution by repeatedly selecting the highest-scoring track and removing its hits from overlapping candidates. Candidate pruning at early stages uses lightweight supervised scorers; the final track score is computed from an outlier-density model that favors tracks unlikely to appear by random alignment of background hits. Acceleration data structures are used to make neighborhood queries and helix–layer intersection computations efficient.

Goals and Evaluation

The goal of this project is to build a model that can predict particle tracks based on the hit points. This project used track score(accuracy) as an evaluation method, which calculate how much the predicted tracks overlap with the true tracks. Moreover, the time execution of each method is also compared.

Dataset

The dataset used in this work is a large-scale simulated tracking dataset derived from proton–proton collision events in a silicon-based detector representative of those used at the Large Hadron Collider (LHC) at CERN. Each event corresponds to a single collision and contains a set of three-dimensional detector measurements, referred to as hits, produced by charged particles as they propagate through successive detector layers under the influence of a magnetic field. The objective of the track reconstruction problem is to group these hits into particle trajectories such that each hit is uniquely assigned to one reconstructed track within an event. The same dataset is also publicly available through an online machine learning repository.

The data are organized as independent events and were originally distributed in compressed archives of approximately 81 GB, expanding to roughly 400 GB after extraction. Each event is stored as a collection of structured files containing detector measurements and, for training data, complete ground-truth information. The hit data provide the global spatial coordinates of each measurement along with detector volume, layer, and module identifiers, which encode the hierarchical structure of the detector and allow geometry-aware processing.

For supervised training and evaluation, ground-truth annotations associate each hit with the particle that generated it and provide the true particle state at the detector intersection, including momentum information. Additional particle-level data describe the initial kinematic properties of all simulated particles, such as their production vertex, momentum, charge, and the total number of hits they produce. At a finer level, detector response information is available in the form of hit cell data, which record the individual sensor elements contributing to each hit together with their signal amplitudes.

A detailed detector geometry description is provided separately and shared across all events. The detector consists of silicon sensor modules arranged in cylindrical barrel layers and disk-shaped endcaps. Each module is defined by its position, orientation, local coordinate system, and segmentation parameters, enabling precise transformations between local sensor coordinates and global spatial coordinates. This information is essential for accurate geometric extrapolation and track fitting.

Data Preprocessing

All preprocessing steps are performed independently for each event, reflecting the physical independence of collision events. Raw hit coordinates are first loaded together with their associated detector identifiers and geometry information. Detector layers are ordered according to their radial or longitudinal position to impose a consistent traversal order for candidate track construction. Hits originating from non-reconstructible particles, as indicated by ground-truth labels, are excluded from supervised training targets but retained during inference to reflect realistic detector conditions.

For learning-based methods, hit-level features are normalized and augmented with geometry-derived attributes, such as cylindrical coordinates and inter-layer distances. Graph-based representations are constructed by connecting hits across adjacent detector layers using spatial and geometric constraints, forming candidate edges for the Graph Neural Network. Ground-truth hit-to-particle associations are used to generate edge and track-level supervision signals.

For geometry-driven approaches, preprocessing focuses on efficient candidate generation. Spatial indexing structures are built to accelerate neighborhood queries, and detector geometry is used to constrain extrapolation between layers. Duplicate measurements within the same detector layer are handled explicitly during preprocessing to avoid premature fragmentation of track candidates.

Across all methods, preprocessing is designed to balance physical realism, computational efficiency, and compatibility with both learned and classical reconstruction pipelines. The resulting event-wise representations serve as the common input to the Graph Neural Network model, Method 1, and Method 2 evaluated in this report.

Methods

We investigated three approaches to the track reconstruction problem. We begin with a Graph Neural Network (GNN)–based method, which has been widely recognized in the recent literature as an effective approach for learning hit-to-track associations. We then introduce Method 1, a hybrid strategy that combines physics-inspired constraints with machine learning models to balance interpretability and performance. Finally, we present Method 2, a predominantly physics-driven approach that achieves the best overall performance in our experiments; in this method, machine learning is used only for noise suppression through two logistic regression models applied at the pairwise and triplet construction stages.

Graph Neural Network Approach

We formulate the track reconstruction problem as a graph-based edge classification task, where detector hits are represented as nodes and candidate hit-to-hit connections are represented as edges. The goal of the Graph Neural Network (GNN) is to identify which candidate edges correspond to true physical connections between hits originating from the same charged particle trajectory.

1. **Hit representation and normalization:** Each hit is represented using its cylindrical coordinates (r, ϕ, z) derived from the global Cartesian measurements. To ensure numerical stability and consistent scaling across events, the coordinates are normalized by fixed detector-scale constants. These normalized hit features serve as the input to the learning pipeline.

2. **Metric learning and embedding space:** To control the combinatorial explosion of possible hit pairs, we first train a metric embedding network that maps each hit into a low-dimensional latent space. The embedding network consists of a multilayer perceptron with layer normalization and nonlinear activations, followed by ℓ_2 normalization to constrain embeddings to the unit hypersphere. The embedding is trained using a contrastive objective with hard negative mining, encouraging hits from the same particle to be close in embedding space while pushing apart hits from different particles. Only hits associated with reconstructible particles are used to define positive and negative pairs during training.

3. **Graph construction with geometric pruning:** Candidate graphs are constructed independently for each event. To improve scalability, events are partitioned into overlapping azimuthal sectors, and graph construction is performed within each sector. For each sector, k-nearest neighbor (kNN) search is applied in the learned embedding space to propose candidate edges. These edges are then aggressively pruned using detector-aware geometric constraints, including consistency in radial progression, azimuthal separation, detector module identity, and three-dimensional distance thresholds. This two-stage procedure significantly reduces the number of spurious edges while maintaining high recall for true hit connections.

4. **Graph neural network architecture:** The pruned graphs are processed by a deep message-passing GNN designed for edge classification. Node and edge features are first projected into a shared hidden space using linear encoders. The network then applies multiple residual interaction layers, where edge features are updated based on the concatenation of source node features, destination node features, and current edge attributes. Updated edge features are aggregated at destination nodes to update node representations. Residual connections are used for both node and edge updates to stabilize training in deeper architectures. After several message-passing layers, a final multilayer perceptron produces a scalar logit for each edge, representing the probability that the edge connects hits from the same particle.

5. **Training objective:** The GNN is trained as a binary edge classifier using a weighted binary cross-entropy loss. Due to the extreme imbalance between true and false edges, the positive class weight is dynamically computed for each graph to counteract class imbalance and prevent degenerate solutions. Training is performed event-wise and sector-wise to manage memory consumption and allow processing of large events.

6. **Track construction and inference:** At inference time, predicted edge probabilities are thresholded to retain high-confidence connections. The surviving edges are merged across sectors and assembled into tracks by computing connected components of the resulting undirected graph. Each connected component corresponds to a reconstructed track candidate, subject to a minimum hit count requirement. This post-processing step converts local edge predictions into global track hypotheses.

Method 1

Method 1 implements a data-driven geometric approach to track reconstruction that combines explicit spatial constraints with feature-based decision rules inspired by classical machine learning.

Hit representation and feature engineering. Each detector hit is represented by its three-dimensional spatial coordinates (x, y, z). To facilitate geometric reasoning and feature construction, hits are transformed into cylindrical coordinates,

$$r = \sqrt{x^2 + y^2}, \qquad \phi = \arctan 2(y, x), \tag{1}$$

which reflect the radial structure of the detector and the transverse motion of charged particles in a magnetic field. Hits are grouped by detector layer, and layers are processed in a fixed order to enforce directional consistency. From individual hits and short hit sequences, a set of engineered features is computed, including inter-hit distance, angular separation, step length, and local directional change. These features form the basis for data-driven candidate evaluation.

Candidate generation across detector layers. Initial track candidates are generated by connecting hits across adjacent detector layers. A candidate connection between two hits i and j is accepted if it satisfies a set of geometric constraints,

$$\|\mathbf{x}_i - \mathbf{x}_j\| < d_{\max}, \qquad |\phi_i - \phi_j| < \Delta\phi_{\max}, \tag{2}$$

where $d_{\max}$ and $\Delta\phi_{\max}$ are empirically selected thresholds. Rather than being fixed a priori, these thresholds are tuned through exploratory analysis of the training data by inspecting feature distributions for true and false associations. This process mirrors supervised learning workflows in which decision boundaries are chosen to balance recall and precision.

Sequential track construction. Starting from valid hit pairs, track candidates are extended sequentially across subsequent detector layers. At each extension step, the current track direction is estimated using the most recent hits, and candidate next hits are evaluated based on their consistency with this estimate. Consistency is measured through engineered features such as deviation in direction, change in step length, and accumulated angular variation. Only hits satisfying predefined tolerances are appended to the track, ensuring smooth and physically plausible trajectories.

Heuristic scoring and data-driven decisions. During extension, each candidate track is associated with a heuristic score that aggregates local feature consistency across its hits. Tracks exhibiting stable feature behavior, such as small directional changes and uniform step lengths, are favored, while those showing erratic behavior are penalized. Although the scoring function is deterministic, its structure reflects a learned decision process operating on data-derived features, analogous to classical machine learning models that operate on fixed feature vectors.

Pruning and noise suppression. As the number of track candidates grows, pruning rules are applied to suppress noise-induced trajectories. These rules include minimum track length requirements, limits on cumulative directional deviation, and thresholds on average feature consistency. The pruning parameters are tuned using data-driven evaluation on labeled events, in a manner similar to hyperparameter optimization in traditional machine learning pipelines. This pruning stage is critical for controlling combinatorial growth and maintaining computational efficiency.

Conflict resolution and final track selection. After candidate generation and pruning, overlapping or duplicate tracks may remain. Conflicts are resolved using deterministic selection criteria that favor longer tracks and those with higher internal consistency scores. Each hit is assigned to at most one reconstructed track, ensuring a valid final assignment. The resulting set of tracks constitutes the output of Method 1.

Method 2

Method 2 adopts a physics-driven strategy for track reconstruction, in which candidate trajectories are constructed and evaluated primarily through geometric and statistical modeling of charged-particle motion in a magnetic field. Machine learning is used only in a limited capacity for early-stage noise suppression, while the final track selection is performed using a probabilistic scoring model grounded in detector hit densities.

Pair generation and geometric constraints. The reconstruction process begins by identifying promising pairs of hits on adjacent or near-adjacent detector layers. Let $\mathbf{x}_i = (x_i, y_i, z_i)$ and $\mathbf{x}_j = (x_j, y_j, z_j)$ denote two hit positions. Candidate pairs are required to satisfy basic geometric consistency conditions, including monotonic progression in radius or longitudinal coordinate and bounded angular separation,

$$\Delta\phi_{ij} = \left|\phi_i - \phi_j\right| < \Delta\phi_{\max}, \tag{3}$$

where $\phi = \arctan(y/x)$. These constraints strongly reduce combinatorial background while retaining physically plausible track segments.

To further suppress random hit combinations, candidate pairs are filtered using a lightweight logistic regression classifier trained on simple geometric features. This classifier acts only as a fast rejection mechanism and does not attempt to model the full particle dynamics.

Extension to triplets and local helix estimation. Surviving hit pairs are extended to triplets by extrapolating the pair-defined direction to the next detector layer and selecting nearby hits. With three spatial measurements, local track curvature can be estimated. In a uniform magnetic field $\mathbf{B}$ aligned with the detector axis, charged-particle trajectories in the transverse plane follow circular motion with radius

$$R = \frac{p_T}{|q|B}, \tag{4}$$

where p_T is the transverse momentum and q is the particle charge. The triplet thus defines a local helix hypothesis, characterized by curvature $\kappa = 1/R$ and pitch determined by the longitudinal displacement.

A second logistic regression classifier is applied at the triplet level, exploiting curvature consistency and angular alignment to remove remaining spurious combinations. As with the pair classifier, this step serves purely to prune the search space.

Track extension via helix propagation. Each accepted triplet is used to fit a helical trajectory parameterized as

$$x(\phi) = x_0 + R\cos(\phi + \phi_0), \tag{5}$$
$$y(\phi) = y_0 + R\sin(\phi + \phi_0), \tag{6}$$
$$z(\phi) = z_0 + \alpha\phi, \tag{7}$$

where (x_0, y_0, z_0) defines the helix reference point and α controls the pitch. The fitted helix is propagated layer-by-layer through the detector, and at each intersection the nearest compatible hit is added to the candidate track. To improve robustness against noise and local misfits, the helix parameters are repeatedly re-estimated using the most recent hits closest to the current layer.

Duplicate hit recovery. Since a particle may produce multiple measurements on a single detector layer, duplicate hits are recovered after initial track construction. For each traversed layer, hits within a small distance of the fitted helix are added back to the track, ensuring that measurement multiplicity is preserved without introducing significant ambiguity.

Outlier-based track scoring. Rather than explicitly modeling measurement uncertainties, candidate tracks are evaluated using an outlier-density-based scoring function. The underlying assumption is that spurious tracks arise from random hit alignments, which can be approximated by the local hit density on each detector layer. For a track $\mathscr{T}$ consisting of hits $\{h_k\}$, the score is defined as

$$S(\mathscr{T}) = -\sum_k \log \rho_k, \tag{8}$$

where ρ_k denotes the estimated density of background hits in the vicinity of hit h_k on its corresponding layer. Tracks passing through low-density regions are therefore favored, as they are statistically unlikely to be formed by chance.

Greedy conflict resolution. Because track candidates are generated independently, substantial overlap exists between them. Final track assignment is performed using a greedy selection procedure: at each step, the highest-scoring track is accepted, its hits are removed from all other candidates, and affected track scores are updated. This process is repeated until no valid candidates remain. The greedy strategy efficiently resolves conflicts while preserving high-quality, non-overlapping tracks.

Results

This section presents a comparison of the three approaches evaluated in this work: a Graph Neural Network (GNN), a data-driven geometric method (Method 1), and a physics-driven classical tracking method (Method 2). The comparison focuses on reconstruction accuracy and computational efficiency, which together determine the practical usability of each approach for large-scale track reconstruction.

Table 1 summarizes the overall performance of the three methods. The GNN-based approach achieves relatively low reconstruction accuracy under the available computational constraints, with slow training and inference times. While graph neural networks are expressive and theoretically powerful, their performance in this setting is limited by resource availability and sensitivity to local prediction errors, which can lead to fragmented or incorrect tracks.

Method 1 significantly improves reconstruction accuracy, reaching approximately 85.7%. This approach benefits from explicit geometric constraints combined with data-driven, feature-based decision rules. However, the increased complexity of candidate evaluation and pruning leads to relatively high inference time, on the order of ten minutes per event. Training-related steps, such as empirical threshold tuning, also require substantial computational effort.

Method 2 achieves the best overall performance among the evaluated approaches. With reconstruction accuracy close to 90%, it consistently outperforms Method 1 while requiring substantially less inference time, approximately two minutes per event. The improved efficiency arises from the stronger enforcement of physics-inspired geometric constraints, which reduce combinatorial growth and enable more aggressive pruning of unlikely trajectories. Method 2 does not require expensive training procedures, making it both faster and more stable across events.

Table 1. Overall performance comparison of the three reconstruction approaches.

Approach	Accuracy (%)	Training Speed	Inference Speed
GNN	31.29	Slow	Slow
Method 1	85.67	Many hours	$\sim$10 min/event
Method 2	88–90	Fast	$\sim$2 min/event

In addition to aggregate accuracy, manual inspection of reconstructed tracks indicates that Method 2 produces longer and more coherent trajectories, with reduced fragmentation and fewer spurious associations. Method 1 also reconstructs meaningful tracks but exhibits a higher tendency toward fragmentation in dense regions, consistent with its weaker geometric constraints. The GNN-based approach is particularly sensitive to local prediction errors, where a single incorrect edge decision can disrupt an otherwise valid track.

Discussions

Although the GNN approach has shown promising results in the literature, for our purposes, the graph-construction phase proved to be a computationally-intensive bottleneck, as the large graph sizes needed for good results required prohibitive amounts of memory during both training and inference. We tried to scale down the size of the graphs by lowering the number of neighbors that were selected by KNN after the embedding phase, which did save memory, but resulted in poor downstream performance. This is likely because any true edges missed in the initial graph construction phase are unrecoverable during the later GNN edge classification phase—so we were only able to save on memory-usage in the graph construction phase by sacrificing many true samples, ultimately leading to short, fragmented tracks, with low overall performance. In light of these limitations, we decided to move on to more computationally feasible approaches.

A key failure mode observed is the *derailing* problem: a single erroneous decision in the middle of a candidate trajectory can fragment a true track into several pieces or, conversely, merge distinct tracks into one. This failure mode has an outsized effect on final evaluation metrics because tracking performance is sensitive to global continuity; local errors therefore propagate nonlinearly through the reconstruction pipeline.

For the learning-based approach, the principal advantage is scalability and adaptability. The AI-based pruning model learns complex, non-local association patterns that are difficult to encode by hand and can generalize across detector regions and diverse track topologies. However, the approach remains sensitive to training-data coverage and hyperparameter choices. In our work hyperparameter search was constrained by available computational resources, which limited model capacity and, consequently, the achievable recall–precision trade-off. Moreover, purely edge-wise learning exposes the model to the derailing problem: locally plausible edge predictions do not guarantee globally consistent tracks unless explicit global constraints or structured losses are introduced.

The physics-driven geometric approach offers opposite trade-offs. Grounded in deterministic helix propagation and statistical scoring, it provides a robust, interpretable baseline that is fast and reliable in practice. Because the algorithm encodes the dominant physical priors explicitly (layer ordering, curvature constraints, and local density modeling), it achieves high precision with limited tuning. Its principal limitation is an effective performance ceiling dictated by the model assumptions: when detector effects or complex event topologies deviate from the assumed model, additional training data do not offer an immediate remedy. In other words, the method is high-bias but low-variance.

The results demonstrate that while learning-based approaches offer flexibility, classical and physics-driven methods currently provide superior accuracy–efficiency trade-offs for this problem under limited computational resources. The strong performance of Method 2 highlights the effectiveness of explicitly encoding geometric and physical priors in large-scale track reconstruction tasks.

The two-stage supervised pruning (simple classifiers applied at the pair and triplet levels) proved critical in practice: tiny amounts of targeted learning substantially reduce combinatorics without undermining the interpretability of the geometric core. This design retains most of the speed and determinism of the physics-first pipeline while selectively using data-driven components where they are most effective.

Conclusion

We presented and compared three approaches for track reconstruction: a graph neural network that learns hit associations from data, and a hit pairs classification by neural network model, and a physics-first pipeline that constructs and scores helix-based track candidates with minimal learning. The GNN approach is computationally prohibitive. Method 1 approach gives quite good performance but a very slow execution time. The physics-driven approach attains strong, interpretable performance by leveraging geometric constraints and a density-based scoring metric; when paired with lightweight supervised pruning it offers an excellent trade-off between accuracy and runtime. Overall, the physics-driven approach gives the best score among the three aproaches.

Future work

There are several direction which this project can be improved. First, for the learning-based methods, we can explore structured losses and global objective functions that directly penalize derailing-type errors (for example, track-level consistency losses, CRF-like global inference layers, or differentiable clustering losses). Increasing model capacity and reducing runtime through

architecture optimization, quantization, or distributed training will also be important to realize the full potential of learned association.

Second, for the physics-driven pipeline, improvements can come from richer probabilistic modeling of measurement noise and more sophisticated density estimators, together with algorithmic accelerations for nearest-neighbor queries and helix propagation. A promising direction is to hybridize deeper: allow the deterministic pipeline to call learned modules for local uncertainty estimation or context-aware scoring while keeping the high-level helix propagation deterministic.

Third, we note that emerging paradigms such as quantum-enhanced optimization or quantum-inspired algorithms might eventually offer speedups for specific subproblems (for instance, combinatorial assignment or global hypothesis scoring). Realizing such gains will require careful co-design of algorithms and hardware-aware formulations; therefore, these ideas remain longer-term research directions rather than immediate engineering solutions.

Reproducibility

The source code and datset are available at the project repository: `https://workdrive.zoho.in/folder/a1kl822c4ed6c0f3748a8b207f02b64138ef4`. The repository includes environment specifications, training scripts for the GNN, Method 1, Method 2 Geometric pipeline.

References

1. R. Frühwirth, "Application of Kalman filtering to track and vertex fitting," *Nucl. Instrum. Methods Phys. Res. A*, vol. 262, pp. 444–450, 1987. doi:10.1016/0168-9002(87)90887-4.
2. M. Kiehn et al., "The TrackML high-energy physics tracking challenge," *EPJ Web Conf.*, 2019. doi:10.1051/epjconf/201921406037.
3. G. DeZoort, S. Thais, J. Duarte, V. Razavimaleki, M. Atkinson, I. Ojalvo, M. Neubauer, P. Elmer, "Charged particle tracking via edge-classifying interaction networks," *Comput. Softw. Big Sci.*, 2021. doi:10.1007/s41781-021-00073-z.
4. A. Elabd et al., "Graph Neural Networks for Charged Particle Tracking on FPGAs," *Frontiers in Big Data*, vol. 5, p. 828666, 2022. doi:10.3389/fdata.2022.828666.
5. J. Duarte and J.-R. Vlimant, "Graph neural networks for particle tracking and reconstruction," in *Artificial Intelligence for High Energy Physics*, World Scientific, 2022. `https://doi.org/10.1142/9789811234033_0012`.
6. X. Ju et al., "Performance of a geometric deep learning pipeline for HL-LHC particle tracking," *Eur. Phys. J. C*, vol. 81, 876, 2021. doi:10.1140/epjc/s10052-021-09675-8.
7. S. Amrouche et al., "The Tracking Machine Learning Challenge: Throughput phase," 2023. doi:10.1007/s41781-023-00094-w.
8. Y. Iiyama, G. Cerminara, A. Gupta, J. Kiesler, V. Loncar, M. Pierini, et al., "Distance-weighted graph neural networks on FPGAs for real-time particle reconstruction in high energy physics," *Front. Big Data*, 2020. doi:10.3389/fdata.2020.598927.
9. J. Pata, J. Duarte, J.-R. Vlimant, M. Pierini, M. Spiropulu, "MLPF: Efficient machine-learned particle-flow reconstruction using graph neural networks," arXiv:2101.08578, 2021. arXiv:2101.08578.
10. K. Bakshi and K. Srinivasan, "Quantum inspired qubit qutrit neural networks for real time financial forecasting," *Scientific Reports*, vol. 15, article 28711, 2025. doi:10.1038/s41598-025-09475-0.

A Kaggle score

Our Kaggle submission achieved a public score of 0.62788 and a private score of 0.85667, which would place our approach in 6th position on the private leaderboard at the time of evaluation. These scores correspond to Method 1.

Note that the TrackML Kaggle competition concluded several years ago, and the leaderboard is no longer actively maintained. As a result, the reported ranking and scores may not be directly visible on the current competition webpage.

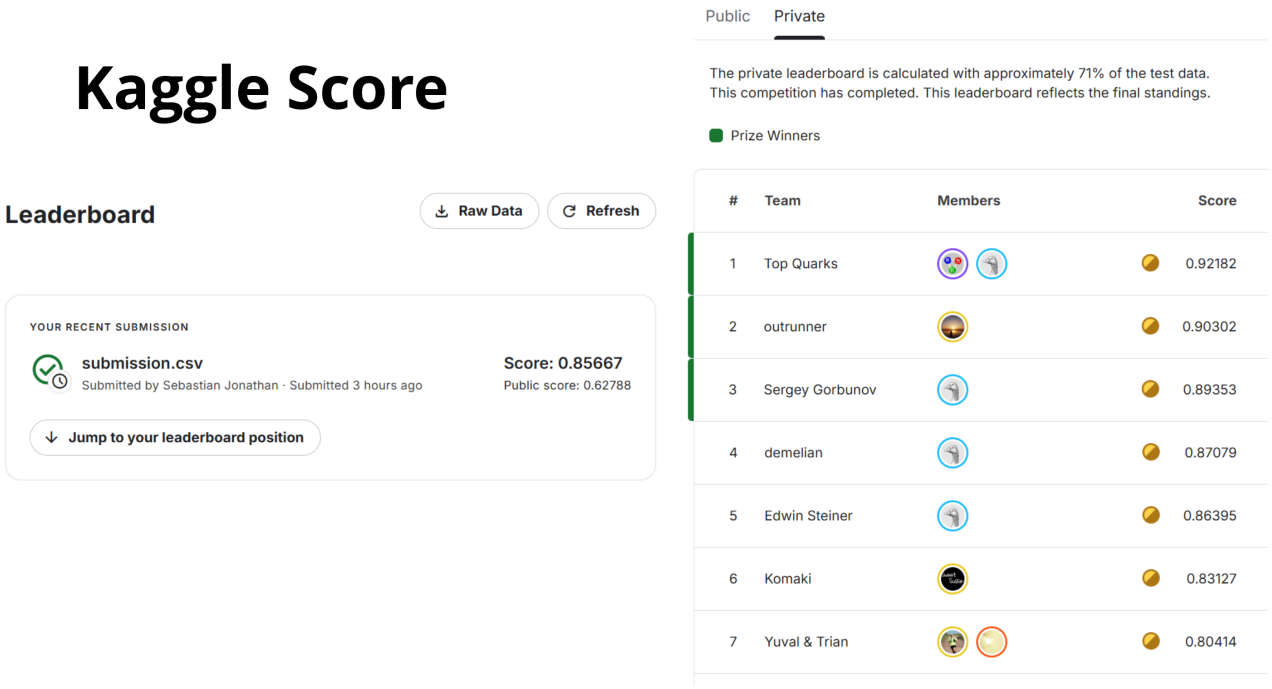

Figure 1. Kaggle Score