Efficient ML-Assisted Particle Track Reconstruction Designs

Sascha Caron[1,2,*], *Nadezhda* Dobreva[3,**], *Antonio* Ferrer Sánchez[4,5,***], *José D.* Martín-Guerrero[4,5,****], *Uraz* Odyurt[6,2,†], *Roberto* Ruiz de Austri Bazan[7,‡], *Zef* Wolffs[8,2,§], and *Yue* Zhao[9,¶]

[1]High-Energy Physics, Radboud University, Nijmegen, The Netherlands
[2]National Institute for Subatomic Physics (Nikhef), Amsterdam, The Netherlands
[3]Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands
[4]Intelligent Data Analysis Laboratory (IDAL), Department of Electronic Engineering, ETSE-UV, University of Valencia, Valencia, Spain
[5]Valencian Graduate School and Research Network of Artificial Intelligence (ValgrAI), Valencia, Spain
[6]Faculty of Engineering Technology, University of Twente, Enschede, The Netherlands
[7]Instituto de Física Corpuscular, University of Valencia, Valencia, Spain
[8]Institute of Physics, University of Amsterdam, Amsterdam, The Netherlands
[9]SURF, Amsterdam, The Netherlands

Abstract. Track reconstruction is a crucial part of High Energy Physics experiments. Traditional methods for the task, relying on Kalman Filters, scale poorly with detector occupancy. In the context of the upcoming High Luminosity-LHC, solutions based on Machine Learning (ML) and deep learning are very appealing. We investigate the feasibility of training multiple ML architectures to infer track-defining parameters from detector signals, for the application of offline reconstruction. We study and compare three Transformer model designs, as well as a U-Net architecture. We describe in detail the two most promising approaches and benchmark the pipelines for physics performance and inference speed on methodically simplified datasets, generated by the recently developed simulation framework, REDuced VIrtual Detector (REDVID). Our second batch of simplified datasets are derived from the TrackML dataset. Our preliminary results show promise for the application of such deep learning techniques on more realistic data for tracking, as well as efficient elimination of solutions.

1 Introduction

The move from the current Large Hadron Collider (LHC) to its High-Luminosity variant (HL-LHC) is expected to catapult data processing requirements, due to the scale increase of

*e-mail: scaron@nikhef.nl
**e-mail: nadezhda.dobreva@ru.nl
***e-mail: antonio.ferrer-sanchez@uv.es
****e-mail: jose.d.martin@uv.es
†e-mail: uodyurt@nikhef.nl
‡e-mail: rruiz@ific.uv.es
§e-mail: zwolffs@uva.nl
¶e-mail: yue.zhao@surf.nl

event frequency and event hit count. This development will bring new challenges to both the speed and complexity of data processing in tasks such as *particle trajectory reconstruction, a.k.a., tracking*. For instance, increased number of collisions will translate to higher pile-up rate (higher μ value), which in turn will dictate more sophisticated algorithm developments.

Traditional algorithms that are currently in place, e.g., algorithms based on Kalmar filtering, though well-designed, have inherent limitations, rendering these incapable in the face of expected scale and complexity levels. To address such limitations, Machine Learning (ML) is considered to be at the core of the next generation of solutions. ML models are capable of consuming high amounts of data at every instance by design. They excel at learning non-linear relationships and mappings between an input and a corresponding output. Main difficulties with ML model development are arriving at the right design and fulfilling high resource requirements.

We tackle the specific challenge of *tracking* using ML models, and showcase our two most promising pipeline designs, out of four total. We adopt a methodology of increasing problem complexity that enables efficient exploration and evaluation of ML model designs, and we report a number of efficiency metrics for a few simplified problem definitions. Furthermore, we demonstrate the potential of the proposed solutions in terms of computational efficiency as well. Tracking as a task involves two main steps, associating (clustering) hits to respective particles/tracks and formation of a track function that interpolates associated track hits. We specifically focus on the former, which is the more challenging of the two.

2 Background and motivation

In the Large Hadron Collider (LHC), either protons or heavy ions are made to smash into one another in so-called *events*. These events in turn release a plethora of subatomic particles which are detected via *tracking* and *calorimetry*. Sophisticated detectors, such as ALICE [1], ATLAS [2], CMS [3], and LHCb [4], allow us to measure the footprint of individual particles as they travel through space. They are each equipped with dedicated tracking detectors designed to measure the trajectories of charged particles by recording *hits* – the points in which the particles pass through them. The connecting of these hits into physical particle trajectories, i.e., tracking, is a crucial task in HEP experiments, for which a multitude of methods have been developed.

Proposed algorithms are extensively tested and validated using datasets for their expected characteristics, such as tracking reconstruction efficiency and computational efficiency. This is especially the case for ML models. Simulations make for suitable data generators, with FATRAS and ATLFAST being some well-known examples.

Traditional methods to assign hits to track candidates include the Kalman Filter (KF). Both stages of tracking rely on the KF [5], with the track finding phase using a so called Combinatorial Kalman Filter (CKF). The combinatorial nature and inherent sequential execution make these methods scale poorly for the HL-LHC. The currently utilised algorithm is tested on simulated data of events with pile-up of $\mu = 200$ [6], which is similar in complexity to the largest dataset evaluated in this paper. The KF pipeline takes 214.3 HS06 $\times$ seconds for a single event, translating to around 12 seconds CPU-time[1]. An optimised algorithm with tighter track selection in the track finding stage achieves a 7x speed-up, requiring 1.8 seconds of CPU-time per event [6].

[1]The CPU-time is multiplied by the HS06 factor of 17.8 for single-threaded execution. HS06 is a benchmark for measuring CPU performance in HEP. Further information: https://w3.hepix.org/benchmarking/HS06.html

3 Related work

Recent advances in machine learning have led to the exploration of Deep Neural Networks for particle tracking. Graph Neural Networks (GNNs) in particular have emerged as a promising choice and the current state-of-the-art has focused on GNN-based solutions, with [7–10] as the most recent efforts. These methods predict, prune or assign weights to edges between vertices (hits) to construct physical trajectories [11]. A recent successful approach [12, 13] involves constructing a graph by connecting hits from different detector layers that satisfy geometric constraints, then clustering hits of the same track in a learned space using object condensation, and finally regressing the properties of the reconstructed objects. One benchmark for GNNs reports an inference time of 2.2s wall-clock time (including data transfer to GPUs) on the full TrackML events, using an NVIDIA A100 GPU [14]. This approach involves significant pre- and post-processing of the data and reports a TrackML accuracy of about 0.87 – more about this metric will follow in Section 5.

Other approaches consider iterative improvements to traditional algorithms or partial inclusion of ML models, as done, for instance, with "A Common Tracking Software (ACTS)" [15]. *We seek a solution predominantly relying on ML models as its core building block, operating as a single-pass algorithm.* As such, we find the Transformer architecture as a potent candidate to be explored. Research using the Transformer architecture has already been developing with many applications in HEP, e.g., [16–18].

When it comes to datasets for evaluation, the most notable trend is the de facto use of TrackML [19] data. ML model training, however, is computationally expensive, leading researchers to often reduce the data. Most commonly, research only considers data associated with the pixel detector [8, 10]. Some works further reduce noise hits [20, 21] or filter for limited P_T values [22]. The lack of consensus for a common data reduction protocol makes performing direct comparisons between different models challenging.

4 Methods

We investigate four different tracking pipelines but only the two best performing ones are elaborated on in this paper. Both are based on the Transformer model architecture, and make use of its encoder only variant. We also experiment with an encoder-decoder Transformer (EncDec) that autoregressively rebuilds a track given a seed hit, and a U-Net architecture which assigns image pixels to different tracks, and we present their results as well.

The Transformer is a deep learning architecture that enables the modelling of pair-wise relationships among elements in sequential data by leveraging the attention mechanism [23]. It can be used to process sequences with permutation equivariance and work with variable input lengths, which is an advantage for the task of trajectory reconstruction. It consists of two structures: an encoder, which learns latent representations of the input sequence of tokens, and a decoder which auto-regressively generates the output sequence. The attention mechanism allows the model to embed the context of the entire input sequence into the representation of each token. The vanilla attention mechanism is a rather compute-heavy operation, which leads to a quadratic memory and time complexity of the Transformer and can restrict its application to very long sequences. Thus, optimisation techniques to reduce the cost of attention computations have been developed, such as Flash Attention [24].

4.1 Model designs

EncCla

The first model is used as a classifier. It takes as input a sequence of non-discretized hits, all from a single event, and outputs a sequence of class labels, one for each hit. To handle variable input lengths, we use padding up to the maximum number of hits in the current batch. Consequently, masking is used to ensure that the attention mechanism ignores the padding values. The class labels correspond to track IDs, and are defined by discretizing the track parameter space into a fixed number of classes. For that, we bin each parameter using a quantile-based approach, so that each bin contains roughly an equal number of hits. The class labels are created from all unique combinations of track parameter bins. This constructing of track classes a priori can only be done up to a finite granularity and is the main challenge of EncCla as it can hinder clustering in high density environments.

The model consists of an embedding layer, a number of encoder blocks and an output layer. For every dataset, a different set of hyperparameters are used. For the biggest dataset, the model has 6 encoder layers, embedding dimensionality of 128, hidden layer dimensionality of 256, 8 attention heads and dropout of 10%. Moreover, the used track parameters are *phi*, *theta*, *q*, and *p*, binned in 30, 30, 2 and 3 bins respectively, making a total of 5 400 classes.

EncReg

The second proposed approach has the same model structure and input, and also utilizes padding and masking. However, it is a regressor that takes the hit coordinates of a single event and outputs for every hit its corresponding regressed track parameters. A clustering algorithm, HDBSCAN [25], is run on the regressed track parameter space to group hits with similar track parameters. The biggest challenge for EncReg is the discovery of track parameters that sufficiently define a track and can be learned by the model.

A different model is trained for every dataset. For the largest dataset utilised, where the attention computation requires a significant amount of memory and limits the batch size and train time, we train two EncReg models: one with exact attention, and one with Flash attention (EncReg-FA), where measures are taken to improve the memory consumption of the attention computation, and consequently training and hyperparameter tuning are sped-up. We do not consider Flash attention for the other datasets as their power and time consumption is still manageable. For EncReg-FA, we also make use of mixed precision training, which the Flash Attention implementation relies on[2]. EncReg-FA has 7 encoder layers, embedding dimensionality of 128, hidden layer dimensionality of 256, 4 attention heads and dropout of 10%. The regressed track parameters are $sin(phi)$, $cos(phi)$, *theta* and q.

The main advantage of these models is that they perform hit-to-track assignment in one step, i.e., the inference time of the model itself scales sub-quadratically with the number of hits. However, this may not be the case in practice, as the complexity of the model must increase with the number of hits. The potential for fast processing also opens the possibility for these models to be utilised as refiner networks to be added in a second stage of the pipeline. For instance, a refiner which regresses track parameters per cluster, i.e., per reconstructed track, and identifies falsely associated hits, increasing the purity of the cluster.

4.2 Datasets

The datasets used to assess our particle tracking methods are simplified and reduced in complexity, enabling ease of evaluation. They cover a spectrum of scale and track representation detail – by increasing the problem complexity in both dimensions, we can efficiently assess the performance and robustness of candidate tracking algorithms. As such, we consider the following five datasets:

[2]This is a reliance imposed by the utilised framework, PyTorch.

- 10-50 (variable count) linear tracks per event, generated with REDVID,
- 10-50 (variable count) helical tracks per event, generated with REDVID,
- 50-100 (variable count) helical tracks per event, generated with REDVID,
- 10-50 (variable count) tracks per event, extracted from the TrackML dataset,
- 200-500 (variable count) tracks per event, extracted from the TrackML dataset.

The first three datasets are created using the REDVID simulation framework [26]. Track function complexity varies between linear and expanding helical, with the latter representing a simple emulation of charged particles in a magnetic field. The 3D geometric space is defined in cylindrical coordinate system, considering r, θ and z coordinates.

The other two datasets are derived from the previously mentioned TrackML data by selecting a number of tracks at random per event. The 3D hit coordinates are in a Cartesian coordinate system, with the global Z-axis defined along the beam direction [19]. The dataset contains ten values associated with every particle and we make use of four of those as track defining parameters: the charge q, and the initial momentum (in GeV/c) along each global axis (p_x, p_y, p_z). The momenta are transformed from the global into the spherical coordinate system. Another preprocessing step is normalising the data.

5 Results

To measure the *prediction accuracy* of our model design, we consider a custom metric: *FitAccuracy score*. It is essentially identical to the *TrackML score* [27], with a small modification in the case of REDVID-generated datasets. The TrackML score is calculated based on the association of weights to hits in the TrackML data. REDVID simulations do not consider weights, so we consider the weight value of 1 for all hits to arrive at our custom FitAccuracy value. In this fashion, we can have a single comparable scoring for all datasets and model designs. In the definition of the TrackML score and FitAccuracy, reconstructed tracks with four or more hits are considered. At least 50% of a reconstructed track's hits must originate from the same truth particle for that track to be considered for the scoring. The score of a track is the sum of correctly assigned hit weights. Available scoring is provided in Table 1.

Table 1: FitAccuracy scores of the four models per dataset. Note that for EncDec, the model starts out with a seed. These seed hits are not counted towards the accuracy.

	FitAccuracy score				
Dataset	EncDec	EncCla	EncReg	EncReg-FA	U-Net
REDVID - 10-50 linear tracks	93%	93%	97%	-	68%
REDVID - 10-50 helical tracks	85%	93%	92%	-	62%
REDVID - 50-100 helical tracks	85%	88%	85%	-	57%
TrackML - 10-50 tracks	26%	94%	93%	-	-
TrackML - 200-500 tracks	-	78%	70%	67%	-

We consider three additional physics performance metrics (taken from [13]) to further investigate the achieved performance with the EncReg model design. Considering the below definitions, the achieved scores are listed in Table 2.

- Perfect match efficiency ϵ^{perf}: The fraction of reconstructed tracks in which all hits belong to a single particle from the ground truth, normalised to the number of particles.

- LHC-style match efficiency ϵ^{LHC}: The fraction of reconstructed tracks in which at least 75% of the hits are of the same particle, normalised to the number of reconstructed tracks.
- Double majority match efficiency ϵ^{DM}: The fraction of reconstructed tracks in which at least 50% of the hits belong to the same particle and it has less than 50% of its hits outside of the reconstructed track, normalised to the number of particles.

Table 2: The three efficiency scores for the EncReg model given per dataset. Note that the results reported in parentheses correspond to EncReg-FA.

Dataset	ϵ^{perf}	ϵ^{LHC}	ϵ^{DM}
REDVID - 10-50 linear tracks	94%	97%	98%
REDVID - 10-50 helical tracks	78%	94%	96%
REDVID - 50-100 helical tracks	60%	89%	92%
TrackML - 10-50 tracks	78%	91%	97%
TrackML - 200-500 tracks	40% (36%)	75% (72%)	82% (79%)

Beyond physics performance, we also consider the computational effort required. The cost of inference is especially interesting, since tracking is possibly to be deployed in an online (embedded in the data-taking pipeline), or semi-online fashion. Each dataset and model combination will impose a different cost. Table 3 lists the computational cost in terms of CPU-time and GPU-time at inference. HDBSCAN is run fully on the CPU and thus for EncReg the CPU-side refers, for the most part, to the clustering algorithm, while the GPU-side refers to the Transformer regressor. Note that we report the time performance only of the solutions with the highest physics performance. We omit the first iteration in inference loops to avoid cold-start effects. All Transformer models have been trained and evaluated on an NVIDIA A100 GPU with 40 GB HMB2 and 18 CPU cores, on the Snellius supercomputer[3]. As explained in Section 3, since related work apply custom protocols to generate TrackML subsets, a direct comparison in terms of computational effort cannot be made.

6 Conclusion and future work

Considering the results given in Tables 1 and 2 and the computational efficiency values from Table 3, the EncCla and EncReg designs, and generally the Transformer architecture, hold potential for further development. However, as explained, the absence of a common data reduction protocol and presence of pre- and post-processing steps, render comparisons to existing solutions as rough approximations. We believe that following a single-pass design, as seen in our pipelines, would be the most effective way to reduce extra steps and to minimise the computational cost to that of inference alone.

One detail to be aware of is the potential need for larger models when addressing more realistic data. This could be a requirement to deal with the increase in scale and complexity of the problem, which would be the case for all competing designs. On the question of "which of the proposed approaches is to be chosen for further development?", the revealing qualities of the physics performance metrics described in Section 5 are highly valuable. Employment of further denominations of such metrics will be the key in validating different aspects of ML models intended for the task of tracking.

Considering the dramatic reduction in computational resource requirements with the use of Flash Attention, we can conclude that a fully ML-based tracking solution is within reach.

[3]https://www.surf.nl/en/services/snellius-the-national-supercomputer

Table 3: Mean CPU-time and mean GPU-time collections during inference, per event.

Dataset	Model	Infer. (CPU-side)	Infer. (GPU-side)
REDVID - 10-50 linear tracks	EncCla	0.1 ms	4.0 ms
	EncReg	8.3 ms	2.4 ms
REDVID - 10-50 helical tracks	EncCla	0.1 ms	4.1 ms
	EncReg	8.7 ms	2.3 ms
REDVID - 50-100 helical tracks	EncCla	0.1 ms	4.3 ms
	EncReg	18.6 ms	4.1 ms
TrackML - 10-50 tracks	EncCla	0.1 ms	4.0 ms
	EncReg	5.8 ms	2.2 ms
TrackML - 200-500 tracks	EncCla	0.1 ms	7.0 ms
	EncReg	70.5 ms	31.9 ms
	EncReg-FA	72.2 ms	3.6 ms

Acknowledgements

This paper is a collaboration between Radboud University, Nikhef, the University of Twente, SURF, the University of Valencia and the University of Amsterdam. Supporting organisations are: NWO (grant no. 62004546 and grant no. OCENW.GROOT.2019.032), the Dutch national e-infrastructure and the SURF (grant no. EINF-7404), Valencian Community (grant no. CIAICO/2021/184), Valencian Foundation (ValgrAI), Quantum Spain project, Spanish Ministerio de Ciencia e Innovación (grant no. PID2020-113644GB-I00), and the Spanish GVA (grant no. PROMETEO/2022/69).

References

[1] T.A. Collaboration, The ALICE experiment at the CERN LHC, Journal of Instrumentation (2008). 10.1088/1748-0221/3/08/S08002

[2] T.A. Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, Journal of Instrumentation (2008). 10.1088/1748-0221/3/08/S08003

[3] T.C. Collaboration, The CMS experiment at the CERN LHC, Journal of Instrumentation (2008). 10.1088/1748-0221/3/08/S08004

[4] T.L. Collaboration, The LHCb Detector at the LHC, Journal of Instrumentation (2008). 10.1088/1748-0221/3/08/S08005

[5] N. Braun, Combinatorial Kalman filter and high level trigger reconstruction for the Belle II experiment (2019)

[6] T.A. collaboration (ATLAS), Tech. rep., CERN (2019)

[7] J. Duarte, J.R. Vlimant, Graph Neural Networks for Particle Tracking and Reconstruction (2022)

[8] A. Elabd, V. Razavimaleki, S.Y. Huang, J. Duarte, M. Atkinson, et al., Graph neural networks for charged particle tracking on fpgas, Frontiers in Big Data (2022). 10.3389/fdata.2022.828666

[9] M. Mieskolainen, Hypertrack: Neural combinatorics for high energy physics (2023)

[10] D. Murnane, S. Thais, A. Thete, Equivariant graph neural networks for charged particle tracking (2023)

[11] A. Lazar, X. Ju, D. Murnane, P. Calafiura, S. Farrell, et al., Accelerating the inference of the exa.trkx pipeline, Journal of Physics: Conference Series (2023). 10.1088/1742-6596/2438/1/012008

[12] Lieret, Kilian, DeZoort, Gage, An object condensation pipeline for charged particle tracking at the high luminosity lhc, EPJ Web of Conf. (2024). 10.1051/epjconf/202429509004

[13] K. Lieret, G. DeZoort, D. Chatterjee, J. Park, S. Miao, P. Li, High pileup particle tracking with object condensation (2023)

[14] X. Ju, D. Murnane, P. Calafiura, N. Choma, S. Conlon, et al., Performance of a geometric deep learning pipeline for hl-lhc particle tracking, European Physical Journal C (2021). 10.1140/epjc/s10052-021-09675-8

[15] X. Ai, C. Allaire, N. Calace, A. Czirkos, I. Ene, et al., A common tracking software project, Computing and Software for Big Science (2022). 10.1007/s41781-021-00078-8

[16] M. Leigh, D. Sengupta, G. Quétant, J.A. Raine, K. Zoch, T. Golling, PC-JeDi: Diffusion for particle cloud generation in high energy physics, SciPost Phys. (2024). 10.21468/SciPostPhys.16.1.018

[17] L. Builtjes, S. Caron, P. Moskvitina, C. Nellist, R.R. de Austri, R. Verheyen, Z. Zhang, Attention to the strengths of physical interactions: Transformer and graph-based event classification for particle physics experiments (2024)

[18] S. Caron, J.E.G. Navarro, M.M. Llácer, P. Moskvitina, M. Rovers, A.R. Jímenez, R.R. de Austri, Z. Zhang, Universal anomaly detection at the lhc: Transforming optimal classifiers and the ddd method (2024)

[19] S. Amrouche, L. Basara, P. Calafiura, V. Estrade, S. Farrell, et al., The Tracking Machine Learning Challenge: Accuracy Phase, in *The NeurIPS '18 Competition* (2020)

[20] N. Choma, D. Murnane, X. Ju, P. Calafiura, S. Conlon, et al., Track seeding and labelling with embedded-space graph neural networks (2020)

[21] X. Ju, S. Farrell, P. Calafiura, D. Murnane, Prabhat, et al., Graph neural networks for particle reconstruction in high energy physics detectors (2020)

[22] A. Heintz, V. Razavimaleki, J. Duarte, G. DeZoort, I. Ojalvo, et al., Accelerated charged particle tracking with graph neural networks on fpgas (2020)

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is All You Need, in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017)

[24] T. Dao, D. Fu, S. Ermon, A. Rudra, C. Ré, FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, in *Advances in Neural Information Processing Systems* (2022)

[25] R.J.G.B. Campello, D. Moulavi, J. Sander, Density-Based Clustering Based on Hierarchical Density Estimates, in *Advances in Knowledge Discovery and Data Mining* (2013)

[26] U. Odyurt, S.N. Swatman, A.L. Varbanescu, S. Caron, Reduced Simulations for High-Energy Physics, a Middle Ground for Data-Driven Physics Research, in *Computational Science – ICCS 2024* (2024)

[27] Kiehn, Moritz, Amrouche, Sabrina, Calafiura, Paolo, Estrade, Victor, Farrell, Steven, et al., The trackml high-energy physics tracking challenge on kaggle, EPJ Web Conf. (2019). 10.1051/epjconf/201921406037