Using Graph Neural Networks for Particle Reconstruction

Jason Weitz,* Dmitri Demler,† and Anthony Vizcaíno Aportela‡
University of California San Diego
(Group 3)
(Dated: March 23, 2023)

This paper proposes the use of Graph Neural Networks (GNNs) as an alternative solution to the challenges of track reconstruction in high energy physics experiments. The paper highlights the difficulties of accurately determining particle paths in real-time when detectors capture more information than can be recorded, and when hits are densely packed while also taking uncertainties into account. By leveraging abstract associations between hits, GNNs offer a promising approach to the particle track reconstruction problem. Our application of a GNN is influenced by the use of this neural network architecture as done in the paper "Graph Neural Networks for Particle Reconstruction in High Energy Physics Detectors" (Ju, et al. 2020). This paper focuses on demonstrating the applicability of GNNs to the specific problem of charged particle trajectory reconstruction in tracking detectors and presents a new approach that can help improve accuracy and scalability in high energy physics experiments.

I. INTRODUCTION

The reconstruction of particle collision events in high energy physics experiments, such as those at the Large Hadron Collider (LHC), presents a complex pattern recognition task for particle detectors such as ATLAS and CMS. These detectors capture more information than we have the ability to record in real time. As a result the detectors need to have the ability to choose what information to discard. This is done by partially reconstructing particle events in real time and discriminating based on those results. One of the many sub-tasks involved in this process is track reconstruction, where the paths of particles resulting from collisions are rebuilt from signals detected in a sub-detector called the tracker[1].

Loosely speaking, the tracker is a cylindrical grid of sensors which surrounds the point of collision and records information about the time and location a potential particle may have passed through. The resulting data is a point cloud of particle locations in space, and the task of track reconstruction involves connecting these points to form plausible paths for each particle, while eliminating implausible ones[1].

The challenge of track reconstruction lies in the difficulty of quickly and accurately determining the path of each particle, particularly when the hits are densely packed, while also taking the uncertainties associated with the coarseness of the sub-detector into account. This challenge is further compounded by the upcoming High Luminosity upgrade to the LHC, which is expected to result in an increase in the number of hits per event[1].

Historically, track reconstruction has been approached using the Kalman Filter, a combinatorial search algorithm that has been highly tuned for performance in current LHC conditions. However, this approach is intrinsically sequential and presents scalability challenges, as it is inefficient to parallelize and scale to the expected increase in hits per event[1].

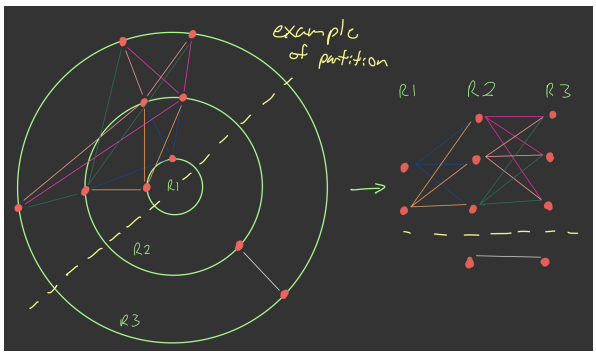

FIG. 1. On the left is a toy representation of how hits may be connected by edges in space. On the right is an isomorphic representation of the same graph. An example of how the data may be divided is represented by the dotted line. Note there are no edges crossing this line.

An alternative solution to this challenge is the use of Graph Neural Networks (GNNs), which can leverage abstract associations between hits and offer a promising approach to the particle track reconstruction problem. GNNs effectively consider this dimensionality, sparsity, and complexity - therefore it is a competent algorithm to use in the case of particle tracking. The task involves demonstrating the applicability of GNNs to the specific problem of charged particle trajectory reconstruction in tracking detectors. Our application of a GNN is influenced by the use of this neural network architecture as done in the paper, "Graph Neural Networks for Particle Reconstruction in High Energy Physics Detectors"[1]. The methods proposed in this paper look to apply this machine learning algorithm to accurately reconstruct the trajectories of charged particles at the LHC.

* jdweitz@ucsd.edu
† ddemler@ucsd.edu
‡ aaportel@ucsd.edu

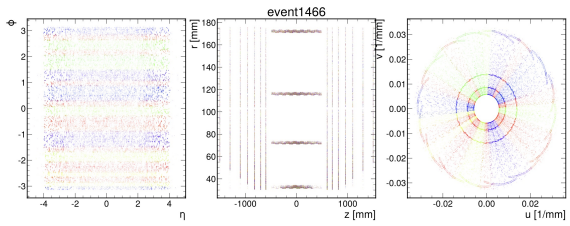

FIG. 2. Visualization for particle hits.

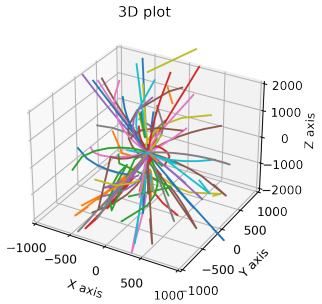

FIG. 3. Visualization of a random sampling of true tracks.

II. DATASET

The dataset utilized in this study is the TrackML Particle Tracking Challenge dataset, which was made available by Kaggle[2]. This dataset was originally used as part of a competition in 2018, with the aim of developing machine learning models that can accurately reconstruct particle tracks from 3D points generated by proton collisions in silicon detectors. The dataset comprises several files, including a `test.zip` file containing 125 events, a `train_{1,2,3,4,5}.zip` file with the full training dataset containing 8850 events split into five files, a `train_sample.zip` file containing the first 100 events from the training dataset, and `detectors.zip` file containing information about the detector geometry.

Each event in the dataset is made up of 3D points for particles generated from collisions caused by protons at the LHC. The training dataset provides the recorded hits, along with the ground truth relationship to particles and the initial parameters of the particles. In contrast, the test dataset only includes the recorded hits. The dataset is provided in a `.csv` format, and each event comprises four files: hits, hit cells, particles, and event ground truth relationship.

The event hit file contains five values for each respective hit, including a `hit_id` that numerically identifies the hit within the event, `x`, `y`, `z` to indicate the position of the hit, `volume_id` to identify the detector group, `layer_id` to identify the detector layer within the group, and `module_id` to identify the detector module within the layer.

The event truth file maps hits to the state of the true particle at the measured hit. Each entry has one mapping for a hit and particle, and includes a `hit_id`, a `particle_id` that identifies the true particle by associating zero values when the hit did not come from a particle, but rather from detector noise, `tx ty tz` for the true coordinates of the hits, `tpx`, `tpy`, `tpz` for the true particle momentum, and `weight`, where the total sum of weights in one event is one and each hit has a weight.

Finally, the event particles file provides values for each particle, including `particle_id`, `vx, vy, vz` for initial position, `px, py, pz` for initial momentum, `q` for particle charge, and `nhits` for the number of hits generated by a particle.

III. METHODS

A. Preprocessing

The dataset provided by Kaggle contains a series of hits in the calorimeter, each represented by three cartesian coordinates denoting their location in space.

However, the use of cartesian coordinates for this purpose is suboptimal as it fails to capture the intrinsic properties of the particles in terms of their momentum and frames of reference. To overcome this limitation, we have incorporated a data preprocessing step that converts the cartesian coordinates into polar-like coordinates (ϕ, η, z) to better comprehend the position of the hits and their relationship with other hits. In particular, we utilize the pseudo-rapidity η as it provides a more accurate characterization of the distribution of highly boosted post-collision particles. By using these polar coordinates, we can conveniently divide the graphs into sectors, which facilitates a detailed analysis of the data, as elaborated in subsequent sections.

The primary challenge in this study has been the conversion of a list of coordinates into a graph that is both descriptive and memory-efficient. A fully connected graph can be naively constructed by labeling all the edges with information encoding the solid angle between points, but this approach scales poorly with the square of the number of nodes. Given that each event in the dataset contains approximately 10^5 hits, constructing a graph for each event in this manner would result in a graph with on the order of 10^{10} edges. Fortunately, physical constraints can be leveraged to greatly reduce the number of edges required for accurate representation of the data.

To generate the training graphs, we adopted three main strategies. First, we imposed a solid angle threshold to remove edges that are unlikely to be related. Specifically, it is unlikely that a hit on one side of the detector will correlate with a hit on the other. Second, we emulated time series by assuming that hits on consecutive rings of the tracker are related to each other in time. Hence, hits on the same ring are likely part of different tracks. Third, we subdivided the detector into cone-shaped regions emanating from the point of collision to

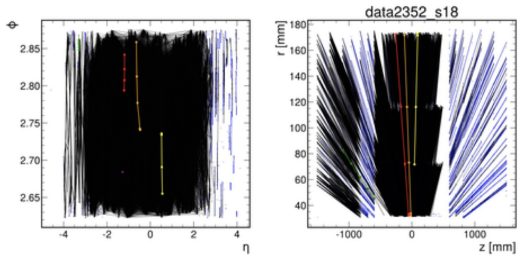

FIG. 4. True particle paths versus all edges.

reduce the number of edges while preserving track information. This can be accomplished with ϕ and η, as high-momentum particles tend to curve less in the magnetic field and can be confined to cone-shaped regions.

To transform the columnar hit data into graph data, we first converted the data from rectilinear to cylindrical coordinates $(x, y, z) \rightarrow (\phi, \eta, z)$. Next, we split the resulting data-frame into smaller data-frames, each localized to a region in the detector. For each data-frame, we defined a fully connected graph whose nodes are labeled by the rows of the data-frame (ϕ, η, z). We then labeled each edge by the difference of the first two columns of adjacent nodes $(\Delta\phi, \Delta\eta)$, and removed any edges in which the solid angle is less than a certain threshold $(\Delta R < \sqrt{\Delta\phi^2 + \Delta\eta^2})$. Finally, we removed any edges between nodes that are within a similar concentric ring of the detector. The resulting pseudo-code can be summarized as follows:

1. Convert data-frame from rectilinear to cylindrical coordinates $(x, y, z) \rightarrow (\phi, \eta, z)$
2. Split data-frame into a list of smaller data-frames, each localized to a region in the detector
3. For each data-frame, define a fully connected graph with nodes labeled by rows of the data-frame (ϕ, η, z)
4. Label each edge by the difference of the first two columns of adjacent nodes $(\Delta\phi, \Delta\eta)$
5. Remove any edges with solid angle less than a certain threshold $(\Delta R < \sqrt{\Delta\phi^2 + \Delta\eta^2})$
6. Remove any edges between nodes that are within a similar concentric ring of the detector

B. Description of Model

In this study, we follow closely the edge classifier model[3] derivative of the original paper. It aims to predict the probability that a given edge is a "true" edge. Specifically, the model takes a list of edges and returns a list of probabilities that describe the likelihood of each edge being part of a track. To achieve this, the model

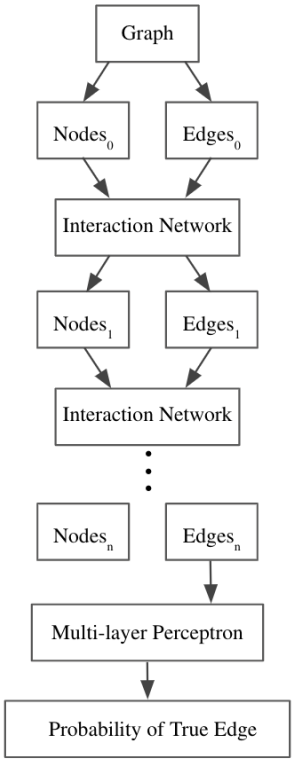

FIG. 5. Process to determine probabilities that edges are true.

first takes in the list of nodes and edge attributes, which are then encoded using a pair of multi-layered perceptrons. The resulting encoding is iteratively connected to an interaction network, which refines the representation through multiple iterations. Finally, the edge attribute encoding is fed into another multi-layered perceptron, and the output layer produces a list of numbers between 0 and 1, representing the probabilities of each edge being part of a track. A visual description of the model is included in figure 5

C. Baseline Comparison

We decided to compare the performance of a combinatoric geometric algorithm[4] with that of the machine learning-based GNN for track reconstruction in particle physics data. We chose this baseline algorithm to provide insight into the trade-offs between different modeling approaches. Ideally, we would have compared our results to the Kalman filter, which is a widely-used algorithm for track reconstruction. However, we found the implementation of the Kalman filter to be challenging due to its complexity.

The combinatoric geometric algorithm we used was a participant in the Kaggle competition from which we collected our data. This algorithm achieved a proficient accuracy and placed 5th in the competition. The algorithm creates a candidate list of possible tracks by selecting combinations of two or three hits between detector lay-

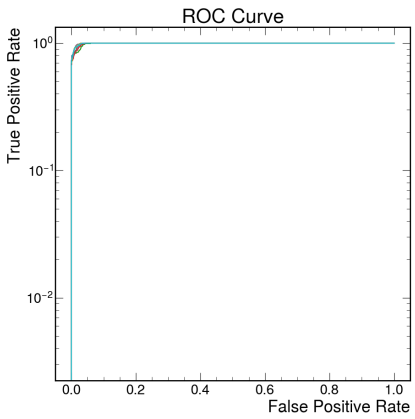

FIG. 6. ROC curve over 10 epochs for the GNN.

ers. The list is then truncated based on the most likely tracks, and the candidate tracks are extended by determining the nearest hit based on an idealized trajectory post-fitting. Notably, this algorithm does not implement much machine learning, making it an attractive baseline for comparison with the GNN.

IV. RESULTS

The implementation of the Github code yielded impressive results, achieving an accuracy of 96.3% already in the first epoch. The accuracy increased to 96.7% after 10 epochs, indicating the model's ability to learn from the data over time. The ROC analysis further validated these results, with an AUC of 97.8% at epoch 1 and a 98.3% AUC at epoch 10.

In contrast, the baseline method that uses a combinatoric geometric algorithm achieved an approximate accuracy of 79%. Notably, this was accomplished with a processing time of 30 seconds per event, while the GNN achieved results from an event in just 0.5 seconds during inference. However, it is important to note that the training time for the GNN was not considered in this comparison, and it may require more computational resources to train the model effectively. Despite this, once trained, the GNN is capable of producing results rapidly and efficiently.

V. CONCLUSION

Machine learning programs have an inherent disadvantage to their mathematical counterparts in initial training time. The mathematical methods don't have a training step and as a result their initial running time is faster than GNN's. However GNN's are faster in more practical situations: when these methods are implemented on the triggers at the LHC they will come pre-trained and the computation time of the GNN can be 60 times faster than the mathematical alternatives.

Despite the need for more memory when training, GNN's show great potential in particle tracking, thanks to their ability to combine the abstraction of CNN's and the feature retention of RNN's. This is particularly important given the enormous amount of data being processed by sensors, with rates of up to 100Tb/s that need to be reduced to 10 Gb/s [5]. Both speed and accuracy are crucial in this context, and GNN's can offer both.

We learned how tricky data processing can be with GNN's and an efficient preprocessing method can exponentially improve a network's computation time. Furthermore, we learned how hard setting up the input of a GNN can be. We think that if we further limit the possible particle tracks by using more proven physical restrictions we could increase time efficiency and accuracy. Furthermore, we think that a good idea is to split the event data into sectors in a more natural way that does not bias particles of a certain energy.

VI. EXTERNAL LINKS

https://github.com/AnthonyAportela/physML_group3

ACKNOWLEDGMENTS

We would like to thank Javier Duarte, Kilian Lieret, and Edwin Steiner.

[1] X. Ju, S. Farrell, P. Calafiura, D. Murnane, L. Gray, T. Klijnsma, K. Pedro, G. Cerati, J. Kowalkowski, G. Perdue, *et al.*, Graph neural networks for particle reconstruction in high energy physics detectors, arXiv preprint arXiv:2003.11603 (2020).

[2] T. P. T. Challenge, Trackml particle tracking challenge dataset, `https://www.kaggle.com/competitions/trackml-particle-identification/data` (2018), accessed on: March 22, 2023.

[3] K. Lieret, gnn-tracking: A library for graph neural networks in object tracking, `https://github.com/gnn-tracking` (2021), accessed on: March 22, 2023.

[4] E. Steiner, Trackml solution, `https://github.com/edwinst/trackml_solution` (2018).

[5] A. M. Deiana, N. Tran, J. Agar, M. Blott, G. Di Guglielmo, J. Duarte, P. Harris, S. Hauck, M. Liu, M. S. Neubauer, *et al.*, Applications and techniques for fast machine learning in science, Frontiers in big Data **5**, 787421 (2022).