Hierarchical Graph Neural Networks for Particle Track Reconstruction

Ryan Liu[1, 2], Paolo Calafiura[1], Steven Farrell[1], Xiangyang Ju[1], Daniel Thomas Murnane[1], Tuan Minh Pham[1]

[1]Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, USA
[2]University of California, 366 Physics North, MC 7300, Berkeley, CA 94720, USA

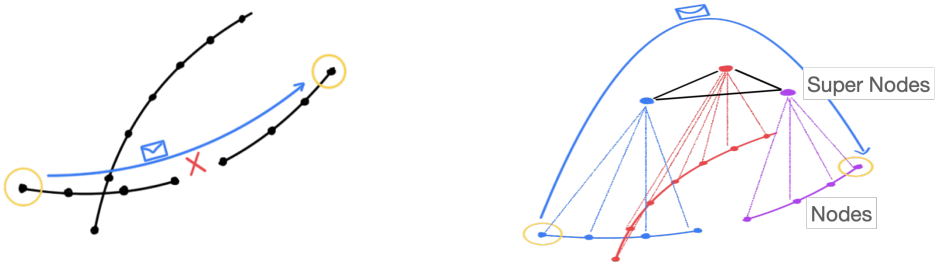
E-mail: ryanliu@lbl.gov

Abstract. We introduce a novel variant of GNN for particle tracking—called Hierarchical Graph Neural Network (HGNN). The architecture creates a set of higher-level representations which correspond to tracks and assigns spacepoints to these tracks, allowing disconnected spacepoints to be assigned to the same track, as well as multiple tracks to share the same spacepoint. We propose a novel learnable pooling algorithm called GMPool to generate these higher-level representations called "super-nodes", as well as a new loss function designed for tracking problems and HGNN specifically. On a standard tracking problem, we show that, compared with previous ML-based tracking algorithms, the HGNN has better tracking efficiency performance, better robustness against inefficient input graphs, and better convergence compared with traditional GNNs.

1. Introduction

In the upcoming High Luminosity Phase of the Large Hadron Collider (HL-LHC) [1, 2], the average number of inelastic proton-proton collisions per bunch $\langle\mu\rangle$ (pile-up) is expected to reach 200 in the new silicon-only Inner Tracker (ITk). This will pose a significant challenge in track reconstruction due to the limited computational resources [3]. Since charged particle reconstruction ("particle tracking") dominates the CPU resources dedicated to event offline reconstruction, a new and efficient algorithm for event reconstruction becomes an urgent need. The HEP.TrkX project [4] and its successor the Exa.TrkX project [5] have studied Graph Neural Networks (GNNs) for charged particle tracking, and excellent performance on the TrackML dataset [6] has been demonstrated in Refs. [7, 8] and more recently on ITk simulation, referred to as GNN4ITk [9].

However, despite the success of GNN-based tracking algorithms, there is much in these techniques that can be improved. In particular, GNN tracking suffers from two types of errors: (1) **broken tracks** (one true track split into multiple segments) and (2) **merged tracks** (a track contains spacepoints of multiple particles). In its nature, the GNN4ITk tracking pipeline prototype [9] is a process of reducing the number of edges; starting from a graph constructed for example by a multi-layer perceptron (MLP) embedding model, filter MLP and GNN edge classifiers are applied to filter out fake edges (i.e. connecting two spacepoints of distinct particles). Thus, broken tracks are more difficult to remove than merged tracks since they can only be resolved by including more edges during the graph construction stage. As such, the pipeline is very sensitive to the efficiency of the graph constructed. Furthermore, the nature of



(a) Flat GNN (b) Hierarchical GNN

Figure 1: The HGNN can not only shorten the distance between two nodes and effectively enlarge the receptive field but also pass messages between disconnected components

message-passing neural networks [10] utilized in the GNN4ITk pipeline, precludes the passing of information between disconnected components, such as the two ends of a broken track. Broken tracks not only limit the performance of edge-cut-based algorithms but also inhibit the full capability of the message-passing mechanism.

In this paper, we present a novel machine learning model called Hierarchical Graph Neural Network (HGNN) [1] for particle tracking to address the aforementioned problems. Similar to the pooling operation often used in Convolutional Neural Networks (CNN), the HGNN pools nodes into clusters called "super-nodes" to enlarge the "receptive field" of nodes to resolve the problem that a "flat" GNN cannot pass messages between disconnected components. Unlike the case of image processing where pooled pixels are already arranged on a 2D grid, the pooled super-nodes cannot use a graph induced by the original graph since disconnected components will remain disconnected. Thus we propose to utilize a K-nearest-neighbors (KNN) algorithm to build the super-graph among super-nodes to facilitate message passing between super-nodes. Furthermore, the HGNN offers us a new approach to track building, as defining a bipartite matching between nodes (spacepoints) and super-nodes (tracks). We measure the performance of this matching procedure against several baselines and show that it can not only recover broken tracks, but also produces fewer fakes tracks from merging.

2. Related Work

2.1. The GNN4ITk Pipeline for Charged Particle Tracking

The GNN4ITk pipeline [8, 9] aims to accelerate particle tracking by utilizing geometric deep learning models. The pipeline as implemented can be divided into four steps: firstly, graph construction takes place to build a graph on the input point-cloud. With one possible construction technique, an MLP is trained to embed spacepoints into a high-dimensional space such that spacepoints belonging to the same particle gets closer in space; a fixed radius graph is then built and passed to a "filter" MLP. The filter takes in spacepoint doublets and prunes the graph down by a $O(10)$ factor in the number of edges. A graph neural network is used to prune the graph further down. Finally, the tracks are built by running a connected components algorithm on the pruned graphs, and ambiguities are resolved by a walk-through algorithm based on topological sorting.

2.2. Graph Pooling Algorithms

As discussed in section 1, the pooling algorithm is a crucial piece of the HGNN architecture. Graph pooling has long been studied in the context of graph neural networks as generating graph

[1] The code now available on github

representations require some global pooling operation. Ying *et al.* introduced DiffPool [11], which pools the graph by aggregating nodes according to weights generated by a GNN. DiffPool pools the graph to a fixed number of super-nodes, and the pooled graph has a dense adjacency matrix. Lee *et al.* proposed SAGPool [12], which pools a graph by selecting top-k rank nodes and uses the subgraph induced. However, SAGPool does not support soft assignment, i.e. assigning a node to multiple super-nodes. The granularity is completely defined by the hyperparameter k and thus also pools to a fixed number of super-nodes. Diehl proposed EdgePool [13], which greedily merges nodes according to edge scores. It is capable of generating a graph that is sparse and variable in size. These pooling algorithms and their features are presented in table 1, along with our proposed pooling technique, described in section 3.1.

Table 1: Graph Pooling Algorithms

Tracking Goal	Feature	DiffPool	SAGPool	EdgePool	GMPool (ours)
Subquadratic scaling	Sparse	✗	✓	✓	✓
End-to-end trainable	Differentiable	✓	✓	✓	✓
Variable event size	Adaptive number of clusters	✗	✗	✓	✓
Many hits to many particles relationship	Soft assignment	✓	✗	✗	✓

2.3. Hierarchical Graph Neural Networks

Hierarchical structures of graph neural networks have been studied in the context of many graph learning problems; some of them utilize deterministic pooling algorithms or take advantage of preexisting structures to efficiently create the hierarchy [14, 15, 16, 17, 18], while the others [19, 20, 21] create the hierarchy in a learnable fashion. Compared with solely graph pooling operations [11], by retaining both pooled and original representations one has the capability of simultaneously performing node predictions and learning cluster-level information. Furthermore, as shown in [20], introducing hierarchical structures can solve the long-existing problem of the incapability of capturing long-range interactions in graphs. Empirical results also show that Hierarchical GNNs have better convergence and training stability compared with traditional flat GNNs.

3. Model Architecture

In order to build the model, there are several challenges that must be tackled, namely, pooling the graph, message passing in the hierarchical graph, and designing a loss function for such a model. In the following section, we introduce our proposed methods for each of them.

3.1. Gaussian Mixture Pooling

In order to provide the features in table 1, we propose a method that leverages the connected components algorithm and Gaussian Mixture Model. The algorithm takes a set of node embeddings as input. The embeddings are then used to calculate edge-wise similarities defined as $s_{ij} = \tanh^{-1}(\vec{v}_i \cdot \vec{v}_j)$. We hypothesize that the graph consists of two types of edges, in-cluster edges and out-of-cluster edges. Then, given the distribution of node similarities, we fit a Gaussian Mixture Model (GMM) to obtain the estimation of the in-cluster and out-of-cluster distributions $p_{in}(s)$ and $p_{out}(s)$. An example distribution is plotted in fig. 3b. We then solve for s_{cut} by $\ln(p_{in}(s_{cut})) - \ln(p_{out}(s_{cut})) = r$, where r is a hyperparameter defining the resolution

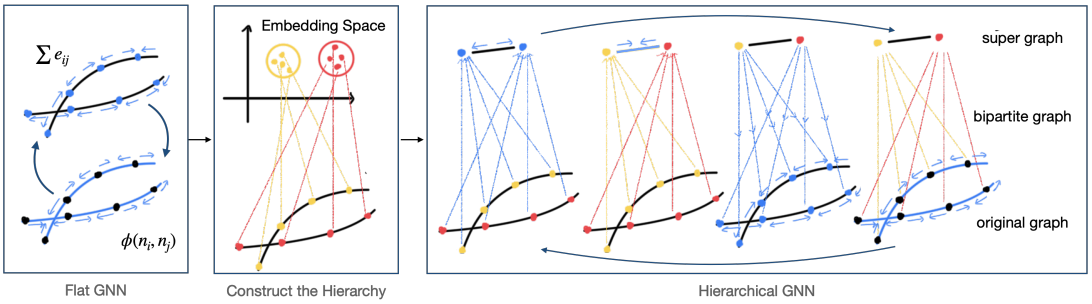
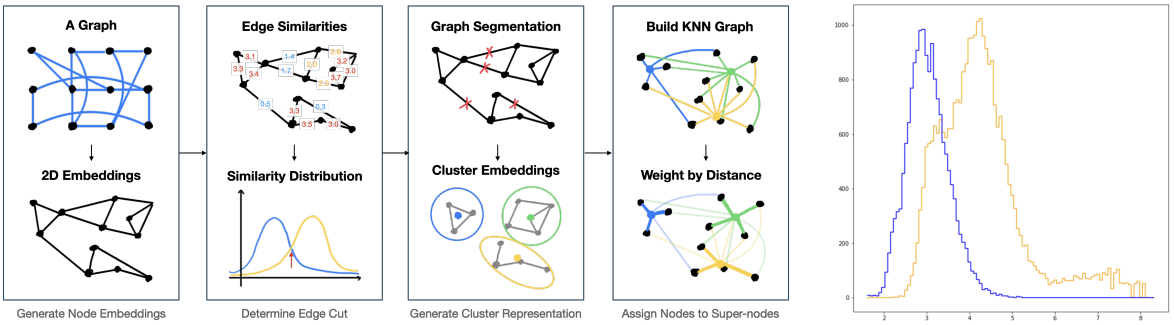


Figure 2: A schematic overview of the HGNN architecture. A flat GNN encoder is used to transform features and embed spacepoints. A pooling algorithm (GMPool) follows to build the hierarchy using the embedded vectors. Finally, hierarchical message passing is applied iteratively to obtain final representations of both nodes and super-nodes.



(a) (b)

Figure 3: (a): schematic overview of the GMPool algorithm. (b): Distribution of edge similarities. Edges connecting spacepoints of the same particle are colored in yellow and otherwise blue.

of the pooling algorithm. The s_{cut} value that gives the best separation of in- and out-of-cluster Gaussians is chosen, and edges with scores below this value are cut. The connected components algorithm follows, and the components C_α of the cut graph are regarded as super-nodes.

To construct super-edges, first super-node embeddings are defined as the centroid of each of the connected components in the embedding space, i.e. $\vec{V}_\alpha = \frac{\vec{V}'_\alpha}{\|\vec{V}'_\alpha\|_{L_2}}$ where $\vec{V}'_\alpha = \frac{1}{N(C_\alpha)} \sum_{i \in C_\alpha} \vec{v}_i$ To connect nodes with super-nodes, similar to the method used in [22], we maintain the sparsity by constructing the bipartite graph with the k-nearest neighbors algorithm. The differentiability can be restored by weighting each of the edges according to the distance in the embedding space, i.e. and $w_{i\alpha} = \frac{\exp(v_i \cdot V_\alpha)}{\sum_{\alpha \in \mathcal{N}(i)} \exp(v_i \cdot V_\alpha)}$ Finally, node features are aggregated to be super-node features according to the graph weights. The super-graph construction is identical except that the k-nearest neighbor search has the same source and destination set. Thanks to its edge-cut nature, the GMPool has sub-quadratic time complexity and runs in milliseconds on our graphs.

3.2. Hierarchical Message Passing Mechanism

In general, it is possible to stack arbitrarily many pooling layers to obtain a hierarchy of arbitrary height. However, the nature of tracking problems suggests that a spacepoint-particle hierarchy will be sufficient for tracking problems. Thus, the pooling layer in this work is kept to be of two levels. For each of the nodes, we update it by aggregating adjacent edge features, super-nodes

features weighted by bipartite graph weights, and its own features. For each of the super-nodes, it is updated by aggregating super-edge features weighted by super graph weights, node features weighted by bipartite graph weights, and its own features. For edges and super-edges, their update rule is identical to the one used in interaction networks.

3.3. Bipartite Classification Loss

At this point, the architecture of HGNN is possible to train on traditional tasks such as node-embedding thanks to GMPool's differentiability. This feature is useful for apples-to-apples comparisons between flat and hierarchical GNNs under the same training regimes. However, to exploit the full potential of the HGNN, we propose a new training regime for it specifically. The most natural way of doing track labeling with HGNN is to use super-nodes as track candidates. For each of the spacepoint-track pairs (bipartite edges), a score is produced to determine if it belongs to a specific track. A maximum-weight bipartite matching algorithm is used to match tracks to super-nodes to define the "truth" for each of the bipartite edges. The loss is given by the binary cross-entropy loss defined by the matched truth. An auxiliary hinge embedding loss is also used for the first warm-up epochs to help the embedding space stably initialize.

4. Results

4.1. Dataset

In this paper, the dataset used to report the performance of HGNN is that of the TrackML Challenge[6]. The TrackML dataset contains events of simulated proton-proton collisions at $\sqrt{s} = 14$TeV with pile-up $\langle\mu\rangle = 200$. Details can be found in [6]. The HGNN has been evaluated in two scenarios; the first scenario is called TrackML-full and contains 2200 filter-processed events, each with approximately $O(7k)$ particles and $O(120k)$ spacepoints. In addition to that, an extensive test of robustness has been done on Bipartite Classifiers, using a simplified dataset TrackML-1GeV. We take the subgraph induced by removing any track below $p_T = 1$GeV. Such an event typically consists of $O(1k)$ particles and $O(10k)$ spacepoints.

4.2. Evaluation

The evaluation metric is tracking efficiency and purity. A particle is matched to a track candidate if **(1)**: the track candidate contains more than 50% of the spacepoints left by the particle and **(2)**: more than 50% of the spacepoints in the track candidate are left by the particle. A track is called reconstructable if it **(1)** left more than 5 spacepoints in the detector and **(2)** has $p_T \geq 1$GeV. The tracking efficiency and fake rate (FR) are thus defined as:

$$\text{Eff} := \frac{N(\text{matched}, \text{reconstructable})}{N(\text{reconstructable})} \qquad \text{FR} := 1 - \frac{N(\text{matched})}{N(\text{track candidates})}$$

4.3. Experiments

We evaluate four models on the TrackML-full dataset. **(1)**: Embedding Flat GNN (E-GNN), **(2)**: Embedding Hierarchical GNN (E-HGNN), **(3)**: Bipartite Classifier Hierarchical GNN (BC-HGNN), **(4)**: Edge Classifier Flat GNN (EC-GNN). The first two serve for apples-to-apples comparisons between flat and hierarchical GNNs - the loss function is the same as the hinge embedding loss used for the metric learning graph construction; tracks candidates are selected by applying a spatial clustering algorithm (H-DBSCAN). The third model represents the state-of-the-art hierarchical GNN for particle tracking; the last one is identical to the GNN4ITk pipeline, and serves as a baseline. The performance of a truth-level connected-components (Truth-CC) track builder are also reported; this takes in filter-processed graphs and prunes them down with ground truth. It is a measure of the graph quality and also an upper bound of edge classifier flat GNN performance. The timing results are obtained on a single Nvidia A100 GPU. To test

robustness against edge inefficiency, we remove 0%, 20%, 30%, and 40% of the edges and train the Bipartite Classifier model to compare it with the Truth-CC.

Table 2: TrackML-Full experiment results. Comparison between embedding models shows that hierarchical structure can enhance the expressiveness of GNNs. Comparing Bipartite Classifiers with the Truth CC, we can see that Bipartite Classifiers can recover some of the tracks that cannot be reconstructed by edge-based GNNs[2]. The timing results also show that HGNN scales to large input graphs of HL-LHC events competitively with other embedding GNNs

Models	E-GNN	E-HGNN	BC-HGNN	EC-GNN	Truth-CC
Efficiency	94.61%	95.60%	**97.86%**	96.35%	97.75%
Fake Rate	47.31%	47.45%	**36.71%**	55.58 %	57.67%
Time (sec.)	2.17	2.64	1.07	**0.22**	0.07

Table 3: TrackML-1GeV extensive robustness test results. We can see that Bipartite Classifiers (BC) are very robust against inefficiencies, whereas edge-based GNN's performance is strongly influenced by missing edges.

Percent Edge Removed	0%	10%	20%	30%	40%	50%
BC Efficiency	98.55%	98.39%	97.68%	96.63%	95.10%	92.79%
BC Fake Rate	1.23%	1.55%	2.13%	3.10%	4.75%	7.31%
Truth-CC Efficiency	98.72%	96.21%	92.31%	85.81%	77.26%	64.81%
Truth-CC Fake Rate	5.87%	15.53%	24.40%	33.48%	42.99%	53.12%

5. Conclusion

In this paper, we introduced a novel graph neural network called a hierarchical graph neural network. We also proposed a new learnable pooling algorithm called GMPool to construct the hierarchy. The architecture successfully resolved the issues of GNN being incapable of capturing long-range interactions and the GNN particle tracking pipeline being sensitive to graphs' efficiency. Creating higher-level representations both shortens the distance between distant nodes in graphs and offers new methods of building track candidates. Empirical results demonstrate that Hierarchical GNNs have superior performance compared with flat GNNs. The hierarchical GNN is available at https://github.com/ryanliu30/HierarchicalGNN and has been integrated into the common framework of the GNN4ITk pipeline [23].

6. Acknowledgements

This research was supported in part by: the U.S. Department of Energy's Office of Science, Office of High Energy Physics, under Contracts No. DE-AC02-05CH11231 (CompHEP Exa.TrkX). This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.

[2] The fake rates given here are not intended as final track building performance, but rather as a model-to-model comparison. A walkthrough algorithm such as in the GNN4ITk pipeline, as well as track cleaning and ambiguity resolution can greatly reduce final fake rates.

References

[1] Apollinari G, Béjar A I, Brüning O, Fessia P, Lamont M, Rossi L and Tavian L 2017 High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1 Tech. rep. Geneva URL https://cds.cern.ch/record/2284929

[2] Lyndon E and Philip B 2008 *Journal of Instrumentation* **3** S08001 URL https://dx.doi.org/10.1088/1748-0221/3/08/S08001

[3] Collaboration A 2022 ATLAS Software and Computing HL-LHC Roadmap Tech. rep. CERN Geneva URL https://cds.cern.ch/record/2802918

[4] Farrell S, Calafiura P, Mudigonda M, Prabhat, Anderson D, Vlimant J R, Zheng S, Bendavid J, Spiropulu M, Cerati G, Gray L, Kowalkowski J, Spentzouris P and Tsaris A 2018 Novel deep learning methods for track reconstruction URL https://arxiv.org/abs/1810.06111

[5] Ju X, Farrell S, Calafiura P, Murnane D, Prabhat, Gray L, Klijnsma T, Pedro K, Cerati G, Kowalkowski J, Perdue G, Spentzouris P, Tran N, Vlimant J R, Zlokapa A, Pata J, Spiropulu M, An S, Aurisano A, Hewes J, Tsaris A, Terao K and Usher T 2020 Graph neural networks for particle reconstruction in high energy physics detectors URL https://arxiv.org/abs/2003.11603

[6] Kiehn M, Amrouche S, Calafiura P, Estrade V, Farrell S, Germain C, Gligorov V, Golling T, Gray H, Guyon I, Hushchyn M, Innocente V, Moyse E, Rousseau D, Salzburger A, Ustyuzhanin A, Vlimant J R and Yilnaz Y 2019 *EPJ Web of Conferences* **214** 06037

[7] Biscarat C, Caillou S, Rougier C, Stark J and Zahreddine J 2021 *EPJ Web of Conferences* **251** 03047 URL https://doi.org/10.1051%2Fepjconf%2F202125103047

[8] Ju X, Murnane D, Calafiura P, Choma N, Conlon S, Farrell S, Xu Y, Spiropulu M, Vlimant J R, Aurisano A, Hewes J, Cerati G, Gray L, Klijnsma T, Kowalkowski J, Atkinson M, Neubauer M, DeZoort G, Thais S, Chauhan A, Schuy A, Hsu S C, Ballow A and Lazar A 2021 *The European Physical Journal C* **81** URL https://doi.org/10.1140%2Fepjc%2Fs10052-021-09675-8

[9] Caillou S, Calafiura P, Farrell S A, Ju X, Murnane D T, Rougier C, Stark J and Vallier A 2022 ATLAS ITk Track Reconstruction with a GNN-based pipeline *Connecting The Dots* (Princeton, United States) URL https://hal.archives-ouvertes.fr/hal-03793565

[10] Battaglia P, Pascanu R, Lai M, Jimenez Rezende D *et al.* 2016 *Advances in neural information processing systems* **29**

[11] Ying R, You J, Morris C, Ren X, Hamilton W L and Leskovec J 2018 Hierarchical graph representation learning with differentiable pooling URL https://arxiv.org/abs/1806.08804

[12] Lee J, Lee I and Kang J 2019 Self-attention graph pooling *International conference on machine learning* (PMLR) pp 3734–3743

[13] Diehl F 2019 *arXiv preprint arXiv:1905.10990*

[14] Zhang Z, Zhuang F, Zhu H, Shi Z, Xiong H and He Q 2020 *Proceedings of the AAAI Conference on Artificial Intelligence* **34** 9612–9619 URL https://ojs.aaai.org/index.php/AAAI/article/view/6508

[15] Li Z, Shen X, Jiao Y, Pan X, Zou P, Meng X, Yao C and Bu J 2020 Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications *2020 IEEE 36th International Conference on Data Engineering (ICDE)* pp 1677–1688

[16] Guille A and Attali H 2022 Document classification with hierarchical graph neural networks *18th International Workshop on Mining and Learning with Graphs* URL https://openreview.net/forum?id=ribpghC_s0

[17] Xia Y, Xia C Q, Pan X and Shen H B 2021 *Nucleic Acids Research* **49** e51–e51 ISSN 0305-1048 (*Preprint* https://academic.oup.com/nar/article-pdf/49/9/e51/38000370/gkab044.pdf) URL https://doi.org/10.1093/nar/gkab044

[18] Zhong Z, Li C T and Pang J 2022 *Data Mining and Knowledge Discovery* 1–28

[19] Gao H and Ji S 2019 Graph u-nets *international conference on machine learning* (PMLR) pp 2083–2092

[20] Rampášek L and Wolf G 2021 Hierarchical graph neural nets can capture long-range interactions *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)* (IEEE) pp 1–6

[21] Chen C, Li K, Wei W, Zhou J T and Zeng Z 2021 *IEEE Transactions on Circuits and Systems for Video Technology* **32** 240–252

[22] Qasim S R, Kieseler J, Iiyama Y and Pierini M 2019 *The European Physical Journal C* **79** 1–11

[23] Atkinson M J, Caillou S, Clafiura P, Collard C, Farrell S A, Huth B, Ju X, Liu R, Minh Pham T, Murnane D c a, Neubauer M, Rougier C, Stark J, Torres H and Vallier A gnn4itk URL https://github.com/GNN4ITkTeam/CommonFramework

Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors

Xiangyang Ju, Steven Farrell, Paolo Calafiura, Daniel Murnane, Prabhat
Lawrence Berkeley National Laboratory
Berkeley, CA
xju@lbl.gov

Lindsey Gray, Thomas Klijnsma, Kevin Pedro, Giuseppe Cerati,
Jim Kowalkowski, Gabriel Perdue, Panagiotis Spentzouris, Nhan Tran
Fermi National Accelerator Laboratory
Batavia, IL

Jean-Roch Vlimant, Alexander Zlokapa, Joosep Pata, Maria Spiropulu
California Institute of Technology
Pasadena, CA

Sitong An
CERN, Geneva, Switzerland &
Carnegie Mellon University, Pittsburgh, PA

Adam Aurisano, V Hewes
University of Cincinnati
Cincinnati, OH

Aristeidis Tsaris
Oak Ridge National Laboratory
Oak Ridge, TN

Kazuhiro Terao, Tracy Usher
SLAC National Accelerator Laboratory
Menlo Park, CA

Abstract

Pattern recognition problems in high energy physics are notably different from traditional machine learning applications in computer vision. Reconstruction algorithms identify and measure the kinematic properties of particles produced in high energy collisions and recorded with complex detector systems. Two critical applications are the reconstruction of charged particle trajectories in tracking detectors and the reconstruction of particle showers in calorimeters. These two problems have unique challenges and characteristics, but both have high dimensionality, high degree of sparsity, and complex geometric layouts. Graph Neural Networks (GNNs) are a relatively new class of deep learning architectures which can deal with such data effectively, allowing scientists to incorporate domain knowledge in a graph structure and learn powerful representations leveraging that structure to identify patterns of interest. In this work we demonstrate the applicability of GNNs to these two diverse particle reconstruction problems.

1 Introduction

The reconstruction of particle collision events in high energy physics experiments such as those at the Large Hadron Collider [9] involves challenging pattern recognition tasks. Particle detectors such as ATLAS [1] and CMS [6] are 40m long, 25m diameter instruments with complex geometry and sparse

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

high dimensional data. Specialized detector sub-systems and algorithms are used to reconstruct the different types and properties of particles produced in collisions. For example, charged particle trajectories are reconstructed from spacepoint measurements ("hits") in tracking detectors, and particle showers are reconstructed from clusters in calorimeters. The upgraded High-Luminosity LHC [2], expected to begin operation in 2026, will deliver increased collision data rates and volumes to the experiments, presenting challenges for current reconstruction solutions.

The traditional approach to particle track reconstruction utilizes combinatorial search algorithms guided by a Kalman Filter. These algorithms are highly tuned for physics performance in today's LHC conditions, but are inherently sequential and scale poorly to the expected HL-LHC conditions with $\mathcal{O}(10^4)$ particles and $\mathcal{O}(10^5)$ hits in each event. The expected challenges of deploying the traditional tracking solutions to HL-LHC data motivated the formation of the HEP.TrkX project to investigate potential new solutions with modern deep learning techniques [11, 10].

In this paper we present our work to apply Graph Neural Networks (GNNs) to the particle track and shower reconstruction problems. GNNs were first introduced in [16] and have been applied to a growing variety of problems including social networks, knowledge graphs, recommender systems, and 3D shape analysis [19, 5]. They were first studied for particle tracking applications in [10] and were also studied for the problem of particle and event classification in [3, 13, 15, 7].

2 Methodology

For both tracking and calorimeter cluster problems, we define a graph representation of the data using individual detector measurements as nodes and then constructing edges between nodes with heuristics based on domain knowledge. The GNN models used are based on the Interaction Networks architecture [4]. The primary task of the GNN is to associate detector elements together by classifying the edges of the graph.

2.1 Tracking

For track finding, we consider only tracks and hits in the barrel region of the detector. The graph is constructed so that the nodes are the hits recorded by the detector and the edges are connections of the hits between adjacent detector layers that pass a pre-defined filter that is tuned to be efficient for tracks resulting from high transverse momentum particles. In the input graphs, node features are the three cylindrical coordinates (r, ϕ, z) and edge features are the difference of the coordinates $(\Delta\eta, \Delta\phi)$. The edge labels are 1 if two hits come from the same track, and 0 otherwise.

The GNN architecture has three components: an encoder which transforms input node and edge features into their latent representations, a graph module which performs message passing to update latent features, and an output module which computes edge classification scores. A diagram of the architecture is shown in figure 1. The encoder uses two fully-connected 2-layer networks for transforming node and edge features, respectively. The initial latent features of the nodes and edges are collectively named H_0. The graph module is applied recursively to the latent features. At each iteration i the initial features H_0 are concatenated onto the current features H_i. This shortcut connection was empirically found to improve model performance. The graph module also uses two fully-connected 2-layer networks, one which computes updated edge features and one which computes updated node features using aggregated incoming edge features. After N iterations of the graph module, the output module takes the last latent features H_N and uses a 2-layer fully-connected network to produce classification scores for every edge. All fully-connected layers use a hidden size of 128 and ReLU activation functions, except the final layer of the output module which uses sigmoid activation. We found that using $N = 8$ graph iterations gave the best model performance.

2.2 Calorimeter clustering

Similar methodologies to those in tracking can be employed to identify energy deposits that should be clustered together to form physically meaningful objects. In fact, with some minor modifications the same variety of edge classification networks used in the tracking problems described above can be immediately applied to the problem of calorimetry. If instead of requiring the final output graph to be a collection of tracks, we allow the output graph to be a mesh on a point cloud and label those edges, an energy cluster can be identified. Moreover, instead of simply being 'true' or 'false' edges

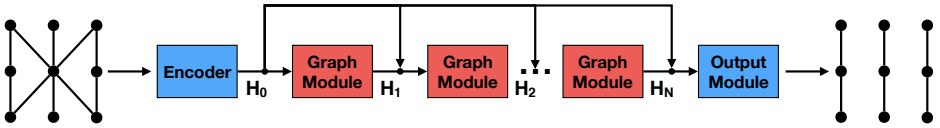


Figure 1: The Graph Neural Network architecture used for tracking.

the particle type of the edge can also be encoded and inferred. This can be achieved with a graph neural network using architectures similar to those demonstrated for tracking as well as networks where the graph is determined dynamically [14]. Here we will focus on the static graph networks and demonstrate results for future calorimeters in particle physics experiments [8].

In particular, we have studied the application of message passing networks to the task of calorimeter clustering, yielding initial promising results. The calorimeter clustering problem is very similar to the tracking problem except that there may be more than two true edges connected to an input node. We cast the task of calorimeter clustering as an operation on an initial static graph generated with a simple algorithm like k-Nearest-Neighbours (kNN), passing messages to generate features for classifying those edges as true or false. Here we are using kNN as stand-in for a lightweight reconstruction algorithm as a first pass to generate a graph on the data. The parameter k was chosen such that there was at least one true edge between all hits in the same truth-level cluster after applying the algorithm. Smaller k results in lower clustering efficiency, depending on the use of noise suppression k can be in the range of 8-24. In particular, these networks use the 'EdgeConv' operator defined in [18], and it was found that concatenating the intermediate hidden states in the output stage improved the rate of model convergence by about a factor of two compared to using no such shortcuts. A diagram of the GNN architecture used for calorimeter clustering is shown in figure 2.

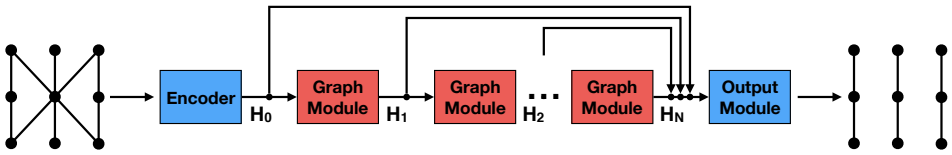


Figure 2: The Graph Neural Network architecture used for calorimeter clustering.

3 Results

The tracking results are based on the TrackML challenge data [17] generated by the ACTS framework [12]. This dataset simulates the very dense environment in the HL-LHC with 200 interactions per bunch crossing on average.

The GNN is trained on an NVIDIA V100 GPU for about 2 epochs in about two hours, resulting in the performance showed in figure 3. With a threshold of 0.5 on the GNN output, the edge efficiency, defined as the ratio of the number of true edges passing the threshold over the number of total true edges, reaches 95.9%, and the purity, defined as the ratio of the number of true edges passing the threshold over the number of total edges passing the threshold, is 95.7%. Guided by the GNN outputs, a simple algorithm is used to reconstruct track candidates. The algorithm makes iterative visits to all hits from inside to outside and reconstructs a best track candidate for the hit in question. Each hit is used only by one track so no ambiguity resolving is needed. This step is called "Connecting The Dots" (CTD). Using the GNN and CTD together reconstructs about 95% of true tracks that can be reconstructed in the graph across the transverse momentum range from 100 MeV to 5 GeV beyond which lacks statistics.

Ongoing work in reconstructing tracks with GNNs includes extending the method to whole detector data and improving the performance of the CTD post-processing algorithm to recover lost efficiency.

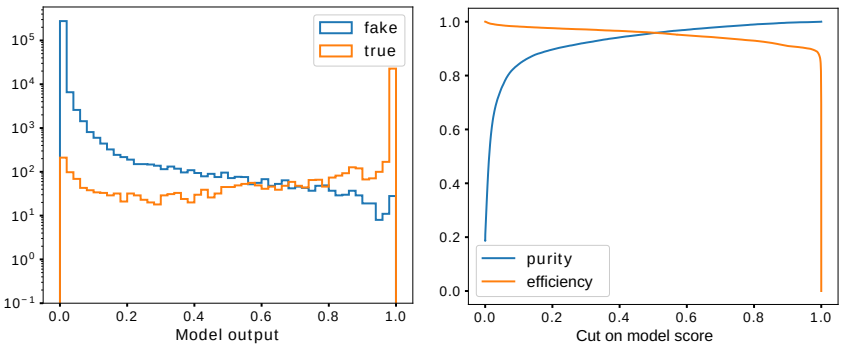


Figure 3: Training results of the Graph Neural Network. Left: The distribution of the edge scores (aka model output) predicted by GNN for true edges that connect the nodes that come from the same track, and for fake edges that do not. Right: The edge purity and efficiency as a function of different cuts on the model score. The definition of efficiency and purity can be found in the text.

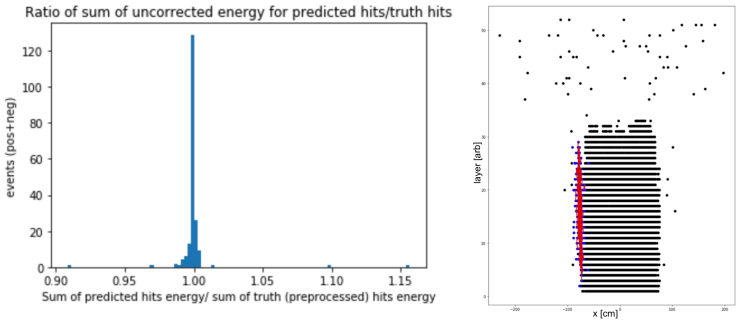


Figure 4: Left: The ratio, per event, for photons of total collected calorimeter energy deposits connected by predicted edges to the energy collected by the associations from ground truth. Right: The event display of a single photon showing the predicted edges (red) and underlying ground-truth nodes (blue) in addition to the energy deposits from noise (black).

In the context of calorimetry, we have achieved results separately for muon, photon, and pion energy deposits in the CMS High-Granularity Calorimeter (HGCal). Each variety of particle deposits energy in the calorimeter in a qualitatively different way, with different expected fluctuations in their energy deposition patterns. Pion showers in particular are the most difficult since they exhibit large variability in their shower transverse profile as a function of the shower depth within the calorimeter. In each case, with examples for photon in figure 4 and pion in figure 5, we have observed excellent performance for correctly associating energy together using the predictions of these networks. For muons we found 99% efficiency with 90% purity, photons we are able to attain 99% efficiency and purity, and for pions we are able to attain better than 90% purity and efficiency. The purity of muons is driven by the large amount of noise hits and edges present in the training sample. All of these measurements are made in dedicated single particle samples for each type of particle, the next step of these tests are to move to variable multi-particle final states such as the decays of τ leptons and then multi-particle jets created in LHC physics events.

This indicates great potential for discovering GNN architectures which can scale to very large number of edges and that can handle multiple high energy particle physics reconstruction tasks. This, in turn, would allow computing centers for high energy physics to focus on certain types of acceleration and better determine where to spend resources and effort in order to become as efficient as possible.

Ongoing work for GNN applications in calorimetry includes studies on how to reconstruct multiple particle types simultaneously using new network architectures which can assign categories to edges. In addition, we are exploring how to better deal with overlapping showers and fractional assignment of calorimeter hit energy into cluster, both of which will be necessary to achieve the best performance

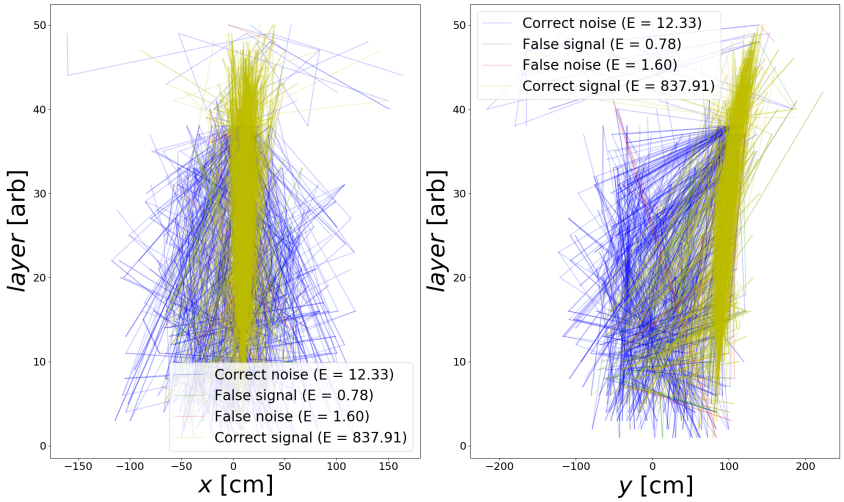


Figure 5: x and y projections of edge classification in a pion shower within the CMS HGCal. The input graph is derived using kNN. The vertical axis in each case is the calorimeter layer number. A score cut of 0.5 is used to identify true edges in this case and edges are labelled according to being true positives (yellow), true negatives (blue), false positives (green), and false negatives (red).

for the HGCal. Finally, explorations into deploying these networks for Liquid Argon Time Projection Chambers are in their initial stages.

4 Conclusion

We have demonstrated that Graph Neural Networks on Point Clouds are suitable for both tracking and calorimetry in high energy physics, having promising physics performance and good scalability. For the track finding problem, the GNNs combined with a simple connecting-the-dot algorithm results in a relative efficiency of over 95% for all particles. Ongoing work is recovering the inefficiency introduced by each selection. For the calorimeter clustering problem, we have found that very similar graph network architectures yield promising solutions. In the individual clustering problems used for testing so far we have found excellent energy collection efficiency, as well as efficiencies and purities better than 90% even in the most difficult scenarios. The next step will be to connect the dots as in the tracking algorithms and derive useful physics quantities from the collections of connected calorimeter energy deposits.

Acknowledgments

We are grateful to Javier Duarte, Phillip Harris, and Jim Hirschauer for the useful discussions. This research was supported in part by the Office of Science, Office of High Energy Physics, of the US Department of Energy under Contracts No. DE-AC02-05CH11231 and No. DE-AC02-07CH11359, FNAL LDRD 2019.017.

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

Part of this work was conducted at "iBanks", the AI GPU cluster at Caltech. We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of "iBanks".

L.G., T.K., K.P., and N.T. are partially supported by Fermilab LDRD L2019.017: "Graph Neural Networks for Accelerating Calorimetry and Event Reconstruction".

S.A. is supported by the Marie Skłodowska-Curie Innovative Training Network Fellowship of the European Commission's Horizon 2020 Programme under contract number 765710 INSIGHTS.

Software Availability

The software and the documentation needed to reproduce the results of this article are available at https://github.com/exatrkx/exatrkx-neurips19

References

[1] G. Aad et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *JINST* 3 (2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003.

[2] G. Apollinari et al. "High Luminosity Large Hadron Collider HL-LHC". In: *CERN Yellow Report* 5 (2015), pp. 1–19. DOI: 10.5170/CERN-2015-005.1. arXiv: 1705.08830 [physics.acc-ph].

[3] J. Arjona Martínez et al. "Pileup mitigation at the Large Hadron Collider with graph neural networks". In: *Eur. Phys. J. Plus* 134.7 (2019), p. 333. DOI: 10.1140/epjp/i2019-12710-3. arXiv: 1810.07988 [hep-ph].

[4] Peter W. Battaglia et al. "Interaction Networks for Learning about Objects, Relations and Physics". In: *CoRR* abs/1612.00222 (2016). arXiv: 1612.00222.

[5] Michael M. Bronstein et al. "Geometric deep learning: going beyond Euclidean data". In: *CoRR* abs/1611.08097 (2016). arXiv: 1611.08097.

[6] S. Chatrchyan et al. "The CMS experiment at the CERN LHC". In: *JINST* 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[7] N. Choma et al. "Graph Neural Networks for IceCube Signal Classification". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dec. 2018, pp. 386–391. DOI: 10.1109/ICMLA.2018.00064.

[8] CMS Collaboration. *The Phase-2 Upgrade of the CMS Endcap Calorimeter*. Tech. rep. CERN-LHCC-2017-023. CMS-TDR-019. Technical Design Report of the endcap calorimeter for the Phase-2 upgrade of the CMS experiment, in view of the HL-LHC run. Geneva: CERN, Nov. 2017. URL: https://cds.cern.ch/record/2293646.

[9] Lyndon Evans and Philip Bryant. "LHC Machine". In: *JINST* 3 (2008), S08001. DOI: 10.1088/1748-0221/3/08/S08001.

[10] Steven Farrell et al. "Novel deep learning methods for track reconstruction". In: *4th International Workshop Connecting The Dots 2018 (CTD2018) Seattle, Washington, USA, March 20-22, 2018*. 2018. arXiv: 1810.06111 [hep-ex].

[11] Steven Farrell et al. "The HEP.TrkX Project: deep neural networks for HL-LHC online and offline tracking". In: *EPJ Web Conf.* 150 (2017), p. 00003. DOI: 10.1051/epjconf/201715000003.

[12] C Gumpert et al. "ACTS: from ATLAS software towards a common track reconstruction software". In: *Journal of Physics: Conference Series* 898 (Oct. 2017), p. 042011. DOI: 10.1088/1742-6596/898/4/042011.

[13] Isaac Henrion et al. "Neural message passing for jet physics". In: (2017). URL: https://dl4physicalsciences.github.io/files/nips_dlps_2017_29.pdf.

[14] Shah Rukh Qasim et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks". In: *Eur. Phys. J.* C79.7 (2019), p. 608. DOI: 10.1140/epjc/s10052-019-7113-9. arXiv: 1902.07987 [physics.data-an].

[15] Huilin Qu and Loukas Gouskos. "ParticleNet: Jet Tagging via Particle Clouds". In: (2019). arXiv: 1902.08570 [hep-ph].

[16] F. Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (Jan. 2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[17] *TrackML Particle Tracking Challenge*. URL: https://www.kaggle.com/c/trackml-particle-identification.

[18] Yue Wang et al. "Dynamic Graph CNN for Learning on Point Clouds". In: (2018). arXiv: 1801.07829.

[19] Jie Zhou et al. "Graph Neural Networks: A Review of Methods and Applications". In: abs/1612.00222 (2019). arXiv: 1812.08434.