An Object Condensation Pipeline for Charged Particle Tracking at the High Luminosity LHC

Kilian Lieret[1,2,*] and *Gage* DeZoort[1,**]

[1]Princeton University
[2]IRIS-HEP

Abstract. Recent work has demonstrated that graph neural networks (GNNs) trained for charged particle tracking can match the performance of traditional algorithms while improving scalability to prepare for the High Luminosity LHC experiment. Most approaches are based on the edge classification (EC) paradigm, wherein tracker hits are connected by edges, and a GNN is trained to prune edges, resulting in a collection of connected components representing tracks. These connected components are usually collected by a clustering algorithm and the resulting hit clusters are passed to downstream modules that may assess track quality or fit track parameters. In this work, we consider an alternative approach based on object condensation (OC), a multi-objective learning framework designed to cluster points belonging to an arbitrary number of objects, in this context tracks, and regress the properties of each object. We demonstrate that OC shows very promising results when applied to the pixel detector of the trackML dataset and can, in some cases, recover tracks that are not reconstructable when relying on the output of an EC alone. The results have been obtained with a modular and extensible open-source implementation that allows us to efficiently train and evaluate the performance of various OC architectures and related approaches.

1 Introduction

The exploration of tracking algorithms based on graph neural networks (GNNs) is motivated by the poor computational scaling of combinatorial Kalman filter (CKF) algorithms with pileup [1]. In recent years, many approaches have been developed around the edge classification paradigm (see e.g. Ref. [2]), in which GNNs are designed to predict whether or not edges drawn between tracker hits represent physical trajectories. These architectures have been shown to demonstrate excellent physics performance and, importantly, scalability with respect to pileup [3]. In the majority of these approaches, tracks are rendered from edge-weighted graphs directly, either by a graph walk algorithm, spatial clustering, or simply collecting connected components.

This work instead explores a learned track rendering stage based on object condensation (OC), a multi-loss training scheme we use to cluster hits belonging to the same track in a learned clustering space. Employing only a very lightweight edge classifying network

*e-mail: kl5685@princeton.edu
**e-mail: jdezoort@princeton.edu

(without any message passing), we show that the OC approach is able to deliver excellent performance when applied to simulated high-pileup TrackML pixel detector events. We also show that the algorithm is able to reconstruct tracks with missing edges, which is not possible when relying on the output of an edge classifier alone. Therefore, this algorithm could also be used at the end of an EC pipeline, leveraging both node and edge embeddings to resolve ambiguities that may exist at the output of the edge classifier.

2 Dataset and input features

This study is performed using the TrackML dataset [4, 5] that simulates the worst-case HL-LHC pileup conditions ($\langle\mu\rangle = 200$) in a generic tracking detector geometry[1]. Our studies are limited to the innermost pixel detector layers, including 4 barrel layers and 7 layers in each endcap. In each pixel tracker event, we embed hits as graph nodes with 14 features, including:

- The cylindrical coordinates r, ϕ, z of the hits, where z corresponds to the line of the colliding proton-proton pairs; the corresponding pseudorapidity η and the conformal tracking coordinates u and v [6] are also included.
- The hit's charge fraction (sum of charge in the cluster divided by the number of activated channels), as well as variables describing the shape and orientation of the cluster introduced in Ref. [7]. For the latter, we closely follow the implementation in Ref. [2].

3 Tracking metrics

We study several tracking definitions to evaluate the performance of our pipeline. They are defined with respect to target populations of tracks; for example, we commonly report the tracking efficiency on *reconstructable particles* that produce at least three hits and have $|\eta| < 4.0$. The efficiencies are defined with respect to various matching criteria between reconstructed tracks and truth tracks. For a given reconstructed track c, we define the *majority particle* π_c as the particle with the largest number of hits within c (and choose a random one if this condition applies to multiple particles). We write $\#c$ for the number of hits in c and $\#\pi_c$ for the number of hits of π_c anywhere. We define the *majority fraction* f_c^{in} as the number of hits of π_c within c divided by $\#c$ and the *majority outside fraction* f_c^{out} as the number of hits of π_c outside of c divided by $\#\pi_c$.

- **Perfect match efficiency** ($\epsilon^{\text{perfect}}$): the number of reconstructed tracks with $\#c > 3$, π_c reconstructable, $f_c^{\text{in}} = f_c^{\text{out}} = 1$ normalized over the number of reconstructable particles.
- **LHC-style match efficiency** (ϵ^{LHC}): the number of reconstructed tracks with $\#c > 3$, π_c reconstructable, $f_c > 75\%$ normalized to the number of clusters with reconstructable π_c. Note that duplicates, wherein multiple reconstructed tracks match to one particle, are possible with this definition.
- **Double majority match efficiency** (ϵ^{DM}): the number of reconstructed tracks with $\#c > 3$, π_c reconstructable, $f_c^{\text{in}} > 50\%$, and $f_c^{\text{out}} > 50\%$ normalized to the number of reconstructable particles. This definition produces unique cluster-track assignments.

We also define the fake rate based on the double majority metric as the number of reconstructed tracks with $\#c > 3$ and π_c reconstructable that do not satisfy the double majority criterion normalized to the number of clusters with reconstructable π_c.

[1]We use the version corresponding to throughput phase of the TrackML challenge, which is hosted at Codalab [5].

As we are mostly interested in high-p_{T} tracks, we also consider these metrics with an additional $p_{\mathrm{T}} > 0.9\,\mathrm{GeV}$ threshold applied to particles and majority particles in the definition of the metrics. The corresponding metrics are denoted $\epsilon^{\mathrm{DM}}_{p_{\mathrm{T}}>0.9}$, $\epsilon^{\mathrm{perfect}}_{p_{\mathrm{T}}>0.9}$, and $\epsilon^{\mathrm{LHC}}_{p_{\mathrm{T}}>0.9}$.

4 Graph construction

The initial graph is constructed by connecting hits on different detector layers that satisfy a series of geometric constraints and pass a classifier threshold.

4.1 Edges based on geometric constraints

This procedure is nearly identical to the geometric graph construction procedure described in Ref. [8], but without applying any cuts based on truth information. Edges between nodes i and j are selected based on the following geometric variables:

$$z_0 := z_i - r_i \frac{z_j - z_i}{r_j - r_i} \quad \phi_{\mathrm{slope}} := \frac{\phi_j - \phi_i}{r_j - r_i}, \quad \text{and} \quad \Delta R := \sqrt{(\eta_j - \eta_i)^2 + (\phi_j - \phi_i)^2},$$

Here we require candidate edges to satisfy $z_0 < 197.4\,\mathrm{mm}$, $\phi_{\mathrm{slope}} < 0.001825/\mathrm{mm}$, and $\Delta R < 1.797$. The cutoff points were optimized to maximize the fraction of reconstructable track edges appearing in the graph, while simultaneously minimizing the number of un-physical edges constructed. In contrast to Ref. [8], no barrel intersection cut is applied. Note that the performance of the graph construction can be translated to an approximate upper bound for the performance of the pipeline downstream. Assuming that the pipeline can build tracks exactly out of those hits that are connected, that is, assuming a pipeline with perfect edge classification followed by identifying tracks as connected components of the resulting edge subgraph, we obtain $\epsilon^{\mathrm{DM}}_{p_{\mathrm{T}}>0.9} \leq_{\mathrm{EC}} 97.4\%$ and $\epsilon^{\mathrm{perfect}}_{p_{\mathrm{T}}>0.9} \leq_{\mathrm{EC}} 84.0\%$.

We provide four initial edge features based on the coordinates of the two hits involved: Δr, $\Delta\phi$, Δz, and ΔR. The resulting graphs are denoted $\mathcal{G} = (X, R_a, I)$, where $X = (x_i)_{i=1,\dots,N} \in \mathbb{R}^{N\times 14}$ are the node features, $I \in \mathbb{N}^{2\times N_{\mathrm{edges}}}$ is the list of edges in coordinate format, and $R_a = (e_{ij})_{(i,j)\in I}$ with $e_{ij} \in \mathbb{R}^4$ are the edge features[2]. We also define truth labels $l_i \in \{0, 1, ..., N_{\mathrm{t}}\}$ (where N_{t} is the number of particles in the graph) indicating the hit is noise ($l_i = 0$) or belongs to track t ($l_i = t$, $1 \leq t \leq N_{\mathrm{t}}$); in this work, we do not consider shared hits between tracks. The truth label y_{ij} indicates whether an edge connects two non-noise hits of the same particle ($l_i = l_j > 0$, $(i, j) \in I$). The geometric cuts alone achieve a purity of $N^{\mathrm{built}}_{\mathrm{true}}/N^{\mathrm{built}}_{\mathrm{total}} = 4.5\%$ at 2.8×10^6 edges per graph.

4.2 Edge filtering

We then apply a lightweight edge classifier to reduce the number of false edges. For this, we train a fully connected neural network (FCNN) ϕ that takes node and edge features as inputs. The node and edge features described in section 2 and subsection 4.1 are concatenated, $z^{(0)}_{ij} = [x_i, x_j, e_{ij}]$, $(i, j) \in I$, and embedded into a 256-dimensional space by a fully connected layer: $z^{(1)}_{ij} = W^{(1)} z^{(0)}_{ij}$, with learnable weights $W^{(1)} \in \mathbb{R}^{256\times(14+14+4)}$. We then apply a fully connected network of five hidden layers of width 256 with ReLU activations and residual connections of the form $z^{(\ell+1)}_{ij} = \sqrt{\beta}\, W^{(\ell+1)}\, \mathrm{ReLU}(z^{(\ell)_{ij}}) + \sqrt{1-\beta}\, z^{(\ell)}_{ij}$, where $l = 1, \dots, 5$, $(i, j) \in I$, and $\beta = 0.4$.

[2]while edge attributes and features are vectors of length N_{edges}, they are indexed by double indices $(i, j) \in I$ (abbreviated to ij in subscript) for notational convenience

To obtain an edge weight, we apply the logistic activation function σ: $w_{ij} = \sigma(W^{(7)}\,\mathrm{ReLU}(z_{ij}^{(6)})) \in (0,1)^{\mathrm{N_{edges}}}$, where $W^{(7)} \in \mathbb{R}^{1\times 256}$. This output is trained with binary cross entropy loss to classify whether an edge connects two hits of the same particle. As we are more interested in tracks with a high value of p_T, we exclude true edges connecting hits of particles with $p_\mathrm{T} < 0.9\,\mathrm{GeV}$ from the loss, i.e.,

$$\ell_{\mathrm{EF}}(y,w) := -\frac{1}{N_{\mathrm{edges}}}\sum_{(i,j)\in I}\Big(\delta_{(p_\mathrm{T}>0.9)}\,y_{ij}\log w_{ij} + (1-y_{ij})\log(1-w_{ij})\Big),$$

$$\text{where}\quad \delta_{(p_\mathrm{T}>0.9)} := \begin{cases}0 & l_i = 0 \vee p_T^{l_i} < 0.9\,\mathrm{GeV},\\ 1 & \text{else}\end{cases}, \tag{1}$$

and $p_\mathrm{T}^{l_i}$ denotes the p_T of the particle belonging to hit i.

The classifier achieves a ROC AUC of 93.3% when evaluated on all tracks and 99.8% when evaluated on all tracks of interest. Here, *tracks of interest* refers to tracks with $p_\mathrm{T} > 0.9\,\mathrm{GeV}$ and the additional constraints described in section 3. To find an appropriate threshold, we calculate the upper bounds to $\epsilon^{\mathrm{DM}}_{p_\mathrm{T}>0.9}$ for the subgraphs satisfying $w_{ij} < w_{\mathrm{thld}}$ for all edges. Based on Figure 1a, we set $w_{\mathrm{thld}} = 0.03$, resulting in TPR = 48.3%, TPR (tracks of interest) = 98.5%, FPR = 1.1%. The approximate upper bounds for this threshold are $\epsilon^{\mathrm{DM}}_{p_\mathrm{T}>0.9} \leq_{\mathrm{EC}} 97.7\%$, $\epsilon^{\mathrm{perfect}}_{p_\mathrm{T}>0.9} \leq_{\mathrm{EC}} 92.1\%$. The purity of the graphs is 68% at 89×10^3 edges per graph.

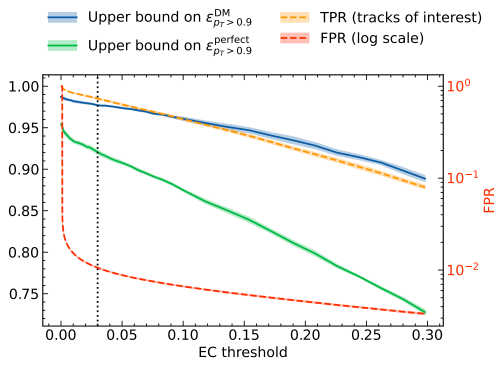
We can also establish a "lower bound" on the performance of the pipeline by reconstructing tracks directly based on this stage. For this, we identify tracks with connected components of the aforementioned subgraphs (though with a stricter value of w_{thld}) and calculate the efficiencies. A scan over w_{thld} is shown in Figure 1b and shows a maximum of $\epsilon^{\mathrm{DM}}_{p_\mathrm{T}>0.9}$ at $w_{\mathrm{thld}} = 0.31$ with $\epsilon^{\mathrm{DM}}_{p_\mathrm{T}>0.9} = 78.9\%$, $\epsilon^{\mathrm{LHC}}_{p_\mathrm{T}>0.9} = 77.1\%$, and $\epsilon^{\mathrm{perfect}}_{p_\mathrm{T}>0.9} = 41.1\%$. However, it should be noted that more elaborate probabilistic schemes to build tracks based on edge scores might surpass these numbers slightly.

5 Object condensation

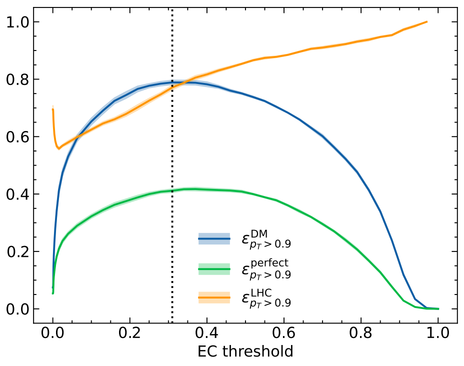
Our architecture extends traditional edge classification pipelines with an additional step based object condensation (OC) [9], a set of truth definitions and loss functions designed to cluster hits belonging to the same object and regress the properties of the reconstructed objects. OC has been extensively validated in applications to calorimetry [9–11], but its applications to tracking have to-date been relatively unexplored.

5.1 Loss functions

For each hit, the OC network predicts condensation strength $\beta_i \in \mathbb{R}$ and clustering coordinates $c_i \in \mathbb{R}^{d_c}$. During training, the highest-β_i hit in each track is dubbed the track's condensation point; the goal of OC is to cluster hits around their track's condensation point in the learned clustering coordinate space. The condensation strength of a track t is that of its condensation point, i.e. $\beta^{(t)} = \max_{\{i|l_i=t\}}\beta_i$. The condensation strength predicted for each hit is used to calculate an un-physical "charge" defined by $q_i = \operatorname{arctanh}^2\beta_i + q_{\min}$ (here, $q_{\min}$ is treated as a hyperparameter). The charge corresponding to a track's condensation point is denoted $q^{(t)}$, located at the position $c^{(t)}$. During training, the condensation points for each track are used to define attractive and repulsive losses designed to produce well-separated clusters of hits



(a) Upper efficiency bounds



(b) Lower efficiency bounds

Figure 1: Performance of the edge classifier applied during edge construction. The left plot shows the true positive and false positive rates together with approximate upper bounds on the recoverable performance based on a perfect EC. The right plot shows the achievable tracking performance by applying a cut on the edge classifier and identifying tracks with connected components of the resulting subgraph. Note that $\epsilon^{\mathrm{DM}}_{p_{\mathrm{T}}>0.9}$ is not strictly monotonous because it is normalized to the number of reconstructed tracks rather than particles.

belonging to the same track in the c_i coordinates:

$$L_{\mathrm{V}} := \frac{1}{N}\sum_{i=1}^{N} q_i \sum_{t=1}^{N_t}\left(\delta_{(l_i=t)} V_t^{\mathrm{att}}(c_i) + s_{\mathrm{rep}}(1-\delta_{(l_i=t)}) V_t^{\mathrm{rep}}(c_i)\right) \tag{2}$$

Here, $\delta_{(l_i=t)}$ is 1 when the node's track label is t and 0 otherwise, and s_{rep} is a hyperparameter. The potential functions are a quadratic attractive loss and a repulsive hinge loss:

$$V_t^{\mathrm{att}}(c) := \delta_{(p_{\mathrm{T}}>0.9)}\, q^{(t)}\, \|c^{(t)} - c\|^2, \quad V_t^{\mathrm{rep}}(c) := q^{(t)} \max(0, 1 - \|c^{(t)} - c\|), \tag{3}$$

where $\delta_{(p_{\mathrm{T}}>0.9)}$ (defined as in Equation 1) excludes hits from noise or low-p_{T} particles from the attraction. An additional loss term L_β is designed to encourage a unique condensation point for each track and suppress the condensation strengths of noise hits:

$$L_\beta := \frac{1}{N_t}\sum_{t=1}^{N_t}(1-\beta^{(t)}) + s_{\mathrm{B}} \frac{\sum_{i=1}^{N} \beta_i\, \delta_{(l_i=0)}}{\sum_{i=1}^{N} \delta_{(l_i=0)}}. \tag{4}$$

Here, s_B is a hyperparameter that controls the strength of noise suppression. All loss terms are finally combined as $L := L_V + s_\beta L_\beta$. For the results in this paper, we choose $s_{\mathrm{rep}} = 0.6$, $s_\beta = 0.004$, $q_{\mathrm{min}} = 0.34$, and $s_{\mathrm{B}} = 0.09$. To reduce the memory footprint of the loss functions, the graphs are split in 32 sectors during training.

5.2 Model

The GNN that is doing the heavy lifting of this tracking pipeline is built from *interaction network* layers [12] with residual connections in the node updates. Node and edge features are first encoded, $x_i^{(1)} = W_{\mathrm{node}}^{\mathrm{enc}} x_i$, $e_{ij}^{(1)} = W_{\mathrm{edge}}^{\mathrm{enc}} e_{ij}$, where $(i, j) \in I$, $W_{\mathrm{node}}^{\mathrm{enc}} \in \mathbb{R}^{192\times 14}$, $W_{\mathrm{edge}}^{\mathrm{enc}} \in$

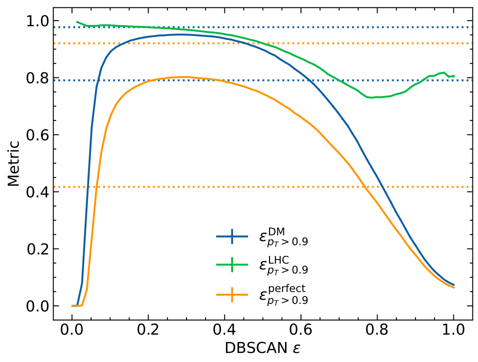


Figure 2: Optimizing the DBSCAN hyperparameters to obtain the maximum performance. The dashed lines are the upper and lower bounds after the EF step for reference (see Figure 1).

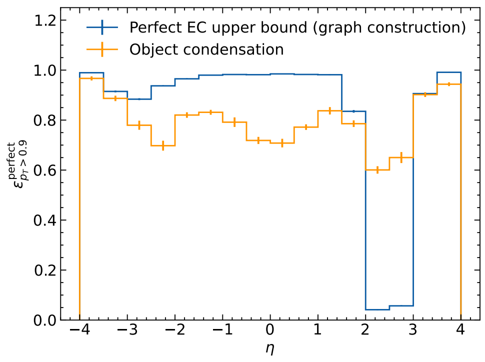


Figure 3: An OC pipeline outperforms a perfect EC when edges between the barrel and the right endcap have been removed.

$\mathbb{R}^{192\times4}$. Then, five iterations of message passing are performed with

$$\begin{aligned} e_{ij}^{(k+1)} &= (\Phi^{(k+1)} \circ \mathrm{ReLU})\Big(\big[x_i^{(k)}, x_j^{(k)}, e_{ij}^{(k)}\big]\Big), \\ x_i^{(k+1)} &= \sqrt{\beta}\,\Psi^{(k+1)}\Big(\big[\mathrm{ReLU}(x_i^{(k)}), \textstyle\sum_{j\in\mathcal{N}_i} e_{ij}^{(k+1)}\big]\Big) + \sqrt{1-\beta}\,x_i^{(k)}. \end{aligned} \tag{5}$$

Here, Φ and Ψ are FCNNs with ReLU activations and a layer width of 192 and one hidden layer. β has been chosen to be 0.2. Finally, the outputs are decoded as $c_i = W_{\mathrm{c}}^{\mathrm{dec}}\,\mathrm{ReLU}(x^{(6)})$ (clustering coordinates), $\beta_i = \sigma(W_\beta^{\mathrm{dec}}\,\mathrm{ReLU}(x^{(6)}))$ (condensation likelihoods), where σ is the logistic function, and $W_{\mathrm{c}}^{\mathrm{dec}} \in \mathbb{R}^{192\times24}$, $W_\beta^{\mathrm{dec}} \in \mathbb{R}^{192\times1}$. The total number of parameters of this model is 1.9×10^6.

5.3 Postprocessing and results

Hit clusters produced in the OC clustering space must be rendered by a downstream algorithm. For this, we use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), an iterative clustering algorithm that has two parameters, ϵ (defining the size of the neighborhood of a point that is considered when merging clusters), and k (minimum number of points within a neighborhood for the points to be considered a *core point*) [13]. For this application, $k = 1$ is optimal; maximizing $\epsilon^{\mathrm{DM}}_{p_\mathrm{T}>0.9}$ vs ϵ yields $\epsilon = 0.279$ (see Figure 2). With this, we obtain $\epsilon^{\mathrm{DM}}_{p_\mathrm{T}>0.9} = 95\%$, $\epsilon^{\mathrm{LHC}}_{p_\mathrm{T}>0.9} = 97\%$, $\epsilon^{\mathrm{perfect}}_{p_\mathrm{T}>0.9} = 80\%$ and $f_{p_\mathrm{T}>0.9} = 1.7\%$. All metrics are presented vs p_T and vs η in Figure 4.

In a side study, we have also tested the ability of the OC network to reconstruct tracks with missing edges after graph construction or edge filtering. For this, all edges between the barrel and the right endcap have been removed after graph construction, limiting the upper bound for $\epsilon^{\mathrm{perfect}}_{p_\mathrm{T}>0.9}$ for a perfect EC to almost zero for tracks with $2 < \eta < 3$. However, as OC is using edges only as a means to exchange information, it is not subject to this upper bound. Indeed, the OC pipeline achieves $\epsilon^{\mathrm{perfect}}_{p_\mathrm{T}>0.9} = 60\%$ in this region. This is shown in Figure 3.

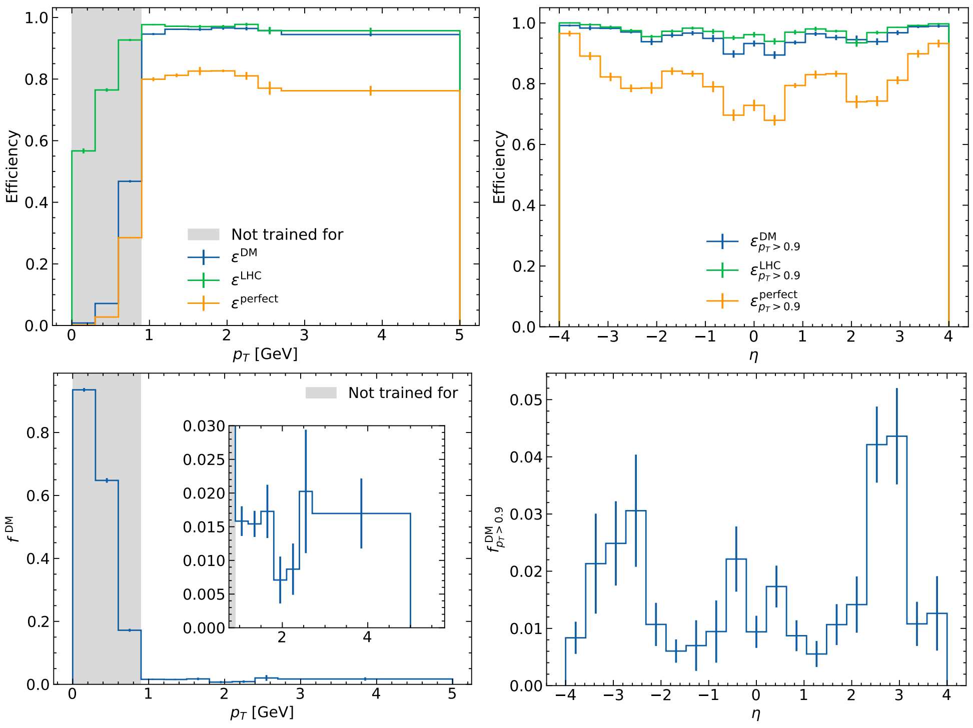


Figure 4: Tracking performance in bins of p_{T} and η.

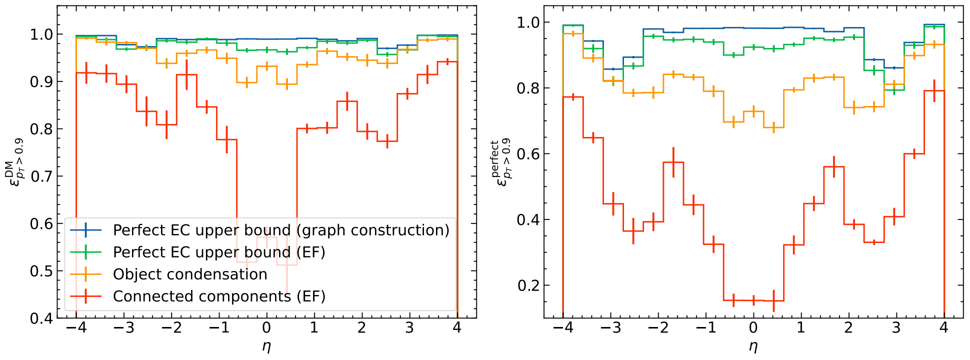


Figure 5: Comparing the performance of the OC pipeline with the upper and lower bounds introduced in this paper.

6 Summary

This paper presents the first GNN-based charged particle tracking pipeline that uses the OC approach to reconstruct tracks in the worst-case pileup conditions expected at the HL-LHC. Our pipeline shows excellent performance with respect to several metrics when applied to the pixel detector of the TrackML dataset. We also demonstrate that OC approach can join partial tracks that are not connected by any of the edges used for message passing, allowing it to outperform algorithms that solely rely on the output of an EC in certain scenarios. This suggests that the use of OC at the output stage of EC-based pipelines may lead to a boost in performance. Future applications of OC may also allow for the regression of track parameters, for example transverse momentum, as part of an architecture capable of rendering tracks and preliminary fits in one shot. The incorporation of track physics may well lead to a more robust model.

All results were produced with the open-source project [14] that implements various OC tracking architectures in a modular and extensible Python package.

References

[1] G. Cerati, P. Elmer, S. Krutelyov, S. Lantz, M. Lefebvre, K. McDermott, D. Riley, M. Tadel, P. Wittich, F. Würthwein et al., EPJ Web of Conferences **127**, 00010 (2016)

[2] X. Ju, D. Murnane, P. Calafiura, N. Choma, S. Conlon, S. Farrell, Y. Xu, M. Spiropulu, J.R. Vlimant, A. Aurisano et al., The European Physical Journal C **81**, 876 (2021), `htttps://doi.org/10.1140/epjc/s10052-021-09675-8`

[3] A. Lazar, X. Ju, D. Murnane, P. Calafiura, S. Farrell, Y. Xu, M. Spiropulu, J.R. Vlimant, G. Cerati, L. Gray et al., Journal of Physics: Conference Series **2438**, 012008 (2023)

[4] S. Amrouche, L. Basara, P. Calafiura, V. Estrade, S. Farrell, D.R. Ferreira, L. Finnie, N. Finnie, C. Germain, V.V. Gligorov et al., *The Tracking Machine Learning challenge : Accuracy phase*, in *The NeurIPS '18 Competition*, edited by S. Escalera, R. Herbrich (Springer International Publishing, Cham, 2020), pp. 231–264, `https://doi.org/10.1007/978-3-030-29135-8_9`

[5] S. Amrouche, L. Basara, P. Calafiura, D. Emeliyanov, V. Estrade, S. Farrell, C. Germain, V.V. Gligorov, T. Golling, S. Gorbunov et al., arXiv:2105.01160 [cs.LG] (2021), submitted to Computing and Software for Big Science, `https://doi.org/10.48550/arXiv.2105.01160`

[6] M. Hansroul, H. Jeremie, D. Savard, Nucl. Instrum. Methods Phys. Res., A **270**, 498 (1988)

[7] P.J. Fox, S. Huang, J. Isaacson, X. Ju, B. Nachman, Journal of Instrumentation **16**, P05001 (2021), `2012.04533`

[8] G. DeZoort, S. Thais, J. Duarte, V. Razavimaleki, M. Atkinson, I. Ojalvo, M. Neubauer, P. Elmer, Computing and Software for Big Science **5** (2021)

[9] J. Kieseler, The European Physical Journal C **80** (2020)

[10] S.R. Qasim, K. Long, J. Kieseler, M. Pierini, R. Nawaz, arXiv:2106.01832 [physics.ins-det] (2021), https://doi.org/10.48550/arXiv.2106.01832

[11] S.R. Qasim, N. Chernyavskaya, J. Kieseler, K. Long, O. Viazlo, M. Pierini, R. Nawaz, Eur. Phys. J. C **82**, 753 (2022), `https://doi.org/10.1140/epjc/s10052-022-10665-7`

[12] P.W. Battaglia, R. Pascanu, M. Lai, D. Rezende, K. Kavukcuoglu (2016), preprint at https://arxiv.org/abs/1612.00222, `1612.00222`

[13] M. Ester, H.P. Kriegel, J. Sander, X. Xu et al., *A density-based algorithm for discovering clusters in large spatial databases with noise.*, in *kdd* (1996), Vol. 96, pp. 226–231

[14] K. Lieret, G. DeZoort, *gnn_tracking: An open-source GNN tracking project*, https://github.com/gnn-tracking/gnn_tracking