Influencer Loss: End-to-end Geometric Representation Learning for Track Reconstruction

Daniel Murnane[1,*]

[1]Lawrence Berkeley National Laboratory, CA USA

Abstract. Significant progress has been made in applying graph neural networks (GNNs) and other geometric ML ideas to the track reconstruction problem. State-of-the-art results are obtained using approaches such as the Exatrkx pipeline, which currently applies separate edge construction, classification and segmentation stages. One can also treat the problem as an object condensation task, and cluster hits into tracks in a single stage, such as in the GravNet architecture. However, condensation with such an architecture may still require non-differentiable operations, and arbitrary post-processing. In this work, I extend the ideas of geometric attention to the task of fully geometric (and therefore fully differentiable) end-to-end track reconstruction in a single step. To realize this goal, I introduce a novel condensation loss function called Influencer Loss, which allows an embedded representation of tracks to be learned in tandem with the most representative hit(s) in each track. This loss has global optima that formally match the task of track reconstruction, namely smooth condensation of tracks to a single point, and I demonstrate this empirically on the TrackML dataset. The model not only significantly outperforms the physics performance of the baseline model, it is up to an order of magnitude faster in inference.

1 Introduction

A large variety of problems within the high energy physics (HEP) domain can be cast as dealing with point cloud structure [1–6]. More generally, problems in chemistry, biology, and robotics are often able to be expressed as point cloud segmentation, classification or regression tasks. Any case where data is not simply tabulated, but instead contains variable-length sets or sequences of data points within a training sample, is most generally represented as a point cloud. Much of the raw data produced in HEP experiments fits this taxonomy, including energy deposits in silicon trackers and calorimetery systems, vectors of particle momenta and abstract representations of hardware, among others. These data are used in a variety of downstream tasks, such as track reconstruction, jet reconstruction, particle isolation and calibration. As such, these tasks are well-described as point cloud tasks, which often require an output at a different scale or granularity than the input. For example, a jet reconstruction may consume a cloud of N energy clusters and return a set of jet-level values.

The majority of efforts in recent years have been spent on constructing the most expressive architecture to consume this data, from RNNs and CNNs [7–10], to GNNs [11–16], to

*e-mail: dtmurnane@lbl.gov

transformers [17, 18], each applying successively fewer restrictions on the input data structure, and therefore more natively capturing it as point cloud-like. However, the question of how to represent point cloud tasks in the training process and loss function has received less attention. Some tasks have intuitive and natural training procedures. For example, jet property regression is naturally trained with a single jet-level error loss. The natural loss function for segmentation of point clouds is less obvious, and it is this problem that I address in this work.

For this, I take as a test case the task of charged particle track finding in a high luminosity collider environment. This task exemplifies many of the challenges of point cloud segmentation: noisy and variable length clouds, large clouds of up to O(1 million) points with correlations between distant points, various scales of segmented objects that must also be classified or regressed downstream. I introduce a simple baseline method for this task, as well as a new loss function called the Influencer Loss, which significantly outperforms the physics and computational performance of the baseline and which, I argue, rigorously parameterises the task of point cloud segmentation.

2 Background & Previous Work

The track finding problem in high energy physics shares many similarities with other tasks such as jet clustering, and can be described as follows: For a given collision event in a particle collider, with N 3-or-4-dimensional energy deposits, each either associated with one or more of the M particles traversing the event or associated with noise, return a set of M track candidates. Each track candidate should be an (ordered or unordered) subset of the N energy deposits. This distinguishes the tracking problem from the classification problem, as each point can belong to zero or one or many instances, and there is a variable number of object instances (particles) across events.

This problem has been traditionally solved as a "navigation" problem, where some starting point is heuristically chosen (e.g. the innermost layer of a detector), a path is seeded by several physically possible points, then walked through according to a model of particle trajectory in that environment (e.g. the helical trajectory of a charged particle in a magnetic field) [19, 20]. More recent efforts treat track finding as a graph traversal problem. In that case, techniques such as graph neural networks can prune physically unlikely path connections in a graph representation of the point cloud, then some graph traversal or segmentation method used to convert the graph into a set of track candidates.

The above methods treat the track finding problem as one composed of sequential points, and while this is the physical basis for the production of a track, there is no reason to impose this constraint on a solution. We can treat the set of points in a particle track as the unordered members of some object that must be detected or "segmented". This task is well-known in the computer vision community, where it is known as instance segmentation. Such segmentation can be easily described in the regularly-spaced 2D grid structure of images, either by drawing bounding boxes [21–25], or by classifying pixels [26, 27]. The boundary of a point cloud instance is harder to define, and previous point cloud segmentation approaches have mostly considered this to be a point-wise instance classification problem [28, 29].

A promising approach to point cloud segmentation called Object Condensation is described in [30]. In this algorithm, points are embedded into a learned space, as well as receiving some learned "charge" value. More strongly charged points attract others, and are also likely to serve as "condensation points" - points that well-represent that object. In inference, points above some condensation score are sorted by decreasing score, with neighbours iteratively found for each point and subsequently removed from the pool of potential neighbours.

While this approach more closely aligns with the task of point cloud object detection than simple clustering in an embedded space, there are several shortcomings related to the misalignment between the inference procedure and the training procedure: a) A hard cut is applied on the condensation score in inference, which requires careful tuning; b) Points are sorted and iteratively sampled, which is not easily parallelisable, and not directly captured in the loss function; c) For computational tractability, only "maximum-scoring" condensation points are considered in training, which can neglect global optima. To address these issues, let us proceed by defining the characteristics of an ideal point cloud object detection loss function:

- For the point cloud x_i, there should be a notion of *representative* points $\mathcal{I}_i$ and *represented* points $\mathcal{U}_i$, where these need not be mutually exclusive sets;
- We would like our loss function to be fully geometric. That is, we would like to capture the representativeness of points entirely by the distance between $\mathcal{I}_i$ and $\mathcal{U}_i$; and
- We would like points in $\mathcal{U}_i$ to condense around exactly one point in $\mathcal{I}_i$ that represents their class.

In the following section we present a loss function that satisfies these simple constraints.

3 Influencer Loss Function

The Influencer Loss is motivated conceptually by the roles of members in a social network. The authors of [31, 32] identify a variety of member types based on their participation in the information flow of a network - their connectivity and whether they are a source or sink of information. Several of these types are shown in **??**. While a more nuanced paradigm might be valuable in future work, here we simplify to two roles: "influencer" nodes and "user" nodes. Influencers are those nodes that represent a class, while user nodes point to influencers as their representative. To handle the notion of directionality, we embed *all* nodes with both an influencer embedding and a user embedding, where directed edges are formed between those user-embedded nodes close to influencer-embedded nodes.

To define the loss function, consider N points $\{x_i\}$ in a track T_a, each embedded into a space $\mathbb{R}^E$ with two different learnable models: a user-embedding $\mathcal{U}(x_i)$ and an influencer-embedding $\mathcal{I}(x_j)$. The **attractive Influencer-User loss** is given by

$$L^+_{\mathcal{UI}}(T_a) = \exp\left(\frac{1}{N}\sum_j \ln\left(\frac{1}{N}\sum_i \Delta^2_{ij}\right)\right), \qquad L^+_{\mathcal{UI}} = \frac{1}{n}\sum_a L^+_{\mathcal{UI}}(T_a), \qquad y_{ij} = 1 \tag{1}$$

For track T_a, this is the arithmetic mean of the distances Δ_{ij} between each of the N^2 pairs of user and influencer embeddings of points in the track. It has minima where all users are close to one or more influencers from the same track. To discourage users being near influencers from other tracks, we introduce a repulsive user-influencer loss contribution $L^-_{\mathcal{UI}}$, and to discourage condensation around more than one influencer per class, we introduce a second repulsive contribution between influencers $L_{\mathcal{I}}$. These are defined by

$$L^-_{\mathcal{UI}} = mean_{ij}(max(0, 1 - \Delta_{ij})), \quad \text{and} \quad L_{\mathcal{I}} = mean_{ij}(max(0, \mathbf{\Delta} - \Delta_{ij})), \quad y_{ij} = 0 \tag{2}$$

We then take this combination as the Influencer Loss $L = L^+_{\mathcal{UI}} + aL^-_{\mathcal{UI}} + bL_{\mathcal{I}}$, where the weights a and b can be used to tune to the desired efficiency-purity rate and efficiency-duplicate rate, respectively.

With this loss function in hand, training requires first finding neighbours between user-influencer pairs within a radius of 1 and influencer-influencer pairs within a radius of $\mathbf{\Delta}$, to

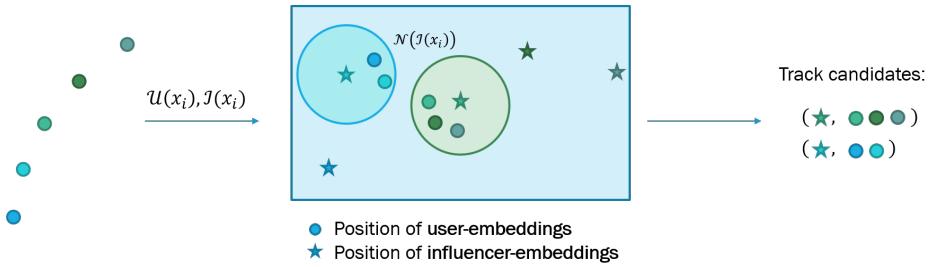


Figure 1: A sketch of the single-step inference of the Influencer Condensation model. Hits are embedded separately as influencers $\mathcal{I}(x_i)$ and users $\mathcal{U}(x_i)$, and a radius nearest neighbor graph is constructed, with users as the database set and influencers as the query set. Influencers with non-empty neighbourhoods are designated the representatives of a track candidate, and the neighbourhood is the track candidate itself.

calculate the repulsive losses. Naively, a loop over tracks is required to calculate the attractive loss. For luminous events, this presents a very expensive step. However, this can be vectorised with a GPU scatter operation, which provides a significant improvement in training time[1].

While this training procedure has extra complexity compared with the naive baseline and the object condensation approaches, inference is much simpler: We perform a fixed radius search for user nodes around each influencer nodes, as sketched in fig. 1. For a fully converged loss, we will obtain the desired case of all users in each class neighbouring exactly one influencer of that class. We examine the performance of the Influencer Loss in the following sections.

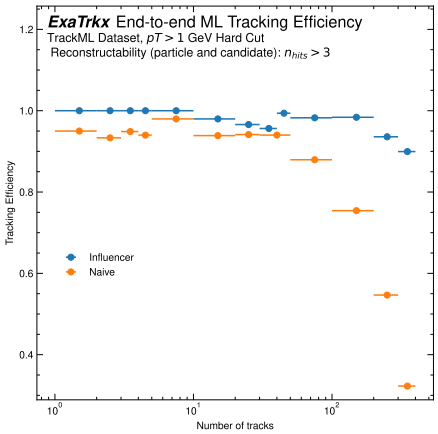
4 Physics Performance

Physics performance is here evaluated on the TrackML dataset. This dataset contains events simulated in a generic, ATLAS-like detector in high-luminosity LHC conditions [33]. While all events are produced with pile-up of $\langle\mu\rangle = 200$, in this work I down-sample to fixed numbers of tracks as a proxy for different luminosities, in order to study the scaling behaviour of the object condensation approaches. Additionally, tracks from particles below 1GeV transverse momentum are removed to simplify the problem.

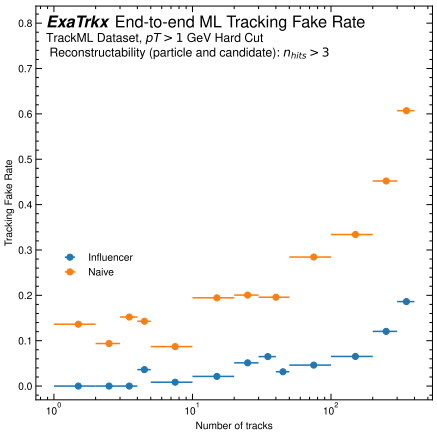
To study the performance of the influencer condensation model, I introduce a naive baseline that captures the end-to-end representative nature of the condensation approach, but does not use the influencer loss. Specifically, the baseline uses a greedy approach to condensation:

1. Embed all hits with a transformer network
2. During training, apply a contrastive hinge loss to hits, as in eq. (2)
3. During inference, perform the following loop over all points:
   (a) Randomly select a point
   (b) If the point is already part of a track candidate, skip it
   (c) Otherwise, make the point a "representative" and find all its neighbors within radius R

[1] Details of this implementation can be found in the project repository github.com/murnanedaniel/InfluencerNet



(a) The track reconstruction efficiency



(b) Tracking fake rate

Figure 2: Physics performance across a proxy of luminosity, obtained by downsampling high-luminosity events. Performance is given as the efficiency and fake rate, as defined in eq. (3).

(d) Propose this representative and its neighborhood as a track candidate

Already, this naive baseline works quite well for low luminosity events (events with less than 100 tracks), as can be seen in fig. 2. The definitions of tracking efficiency and fake rate (FR) are the conventional ones

$$eff = \frac{N_{particles}(\text{matched, reconstructable})}{N_{particles}(\text{reconstructable})} \qquad FR = \frac{N_{tracks}(\text{unmatched})}{N_{tracks}} \tag{3}$$

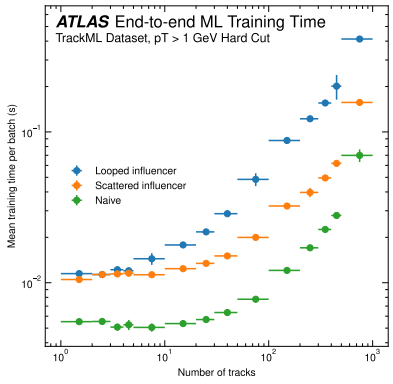
where a particle is reconstructable if it has at least 3 hits, a track candidate must also have at least 3 hits, and a particle is matched if there exists a track candidate such that greater than 50% of its hits belong to the particle.

We see that while the naive baseline struggles to maintain high efficiency as events scale to higher luminosity, the influencer condensation model stays consistently above 90% tracking efficiency. Similarly, the fake rate is uniformly an order of magnitude lower when using the influencer loss, than when using the greedy approach. Some component of the fake candidates can be attributed to the naively random choice of representatives in the baseline.

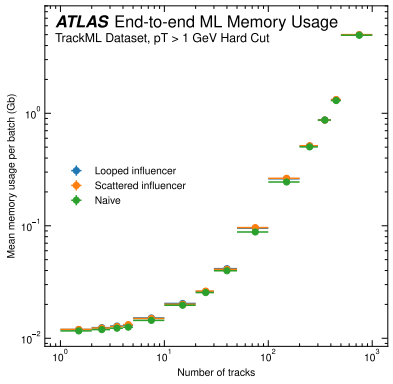
5 Computational Performance

While physics performance indeed appears to at least match (and significantly exceed in some regimes) simpler condensation approaches, the main contribution of the influencer condensation approach is its high alignment in inference with the tracking challenge. As such, there are no arbitrary choices or post-processing steps required, and inference is highly parallelizable. This does come at a cost of training time, as can be seen in the looped influencer performance of fig. 3a. This cost is due to a track-wise fully-connected distance calculation. This can already be improved with the GPU-implemented scatter influencer loss, shown in fig. 3a to reduce the training time to only 1.5× that of the naive baseline.

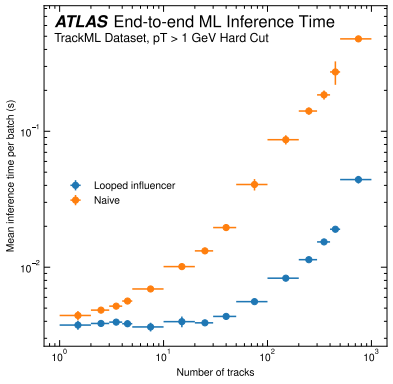
While the backbone architecture to each condensation approach is the same in our study (a vanilla transformer), the influencer model requires embedding into two spaces: influencer and



(a) The mean time required to pass each event through a training step



(b) Peak-memory usage when passing an event through GPU forward and backward passes



(c) The mean time required to pass each event through track candidate building

Figure 3: Computational performance of the Influencer Condensation model and naive baseline model, across both training and inference. Timing and memory usage are obtained across 1000 events, on an Nvidia A100 40Gb GPU.

user space. As such, there is a slight increase in training memory required, as in fig. 3b. However, we see the true value of this model in fig. 3c, where the inference time at low-luminosity is essentially flat, and then scales with the number of tracks, but with an order of magnitude lower latency than the naive benchmark, which relies on sequential post-processing.

6 Summary

This work introduces a novel approach to the tracking challenge, using a version of object condensation called influencer condensation. By amortizing a more complex training proce-

dure that harnesses two embedding spaces for each hit, an influencer embedding and a user embedding, the inference of this condensation model is highly efficient and parallelizable. We show that despite its very low latency, it still outperforms a much slower baseline model in track reconstruction efficiency and fake rate measurements. Tracking efficiency remains above 90% for events containing up to 500 particles. Scaling this model to events of the size expected at the HL-LHC - $O(10k)$ particles - is intended to be the immediate next step of this work, in particular with a view to application on ATLAS HL-LHC Inner Tracker (ITk) events.

Other future research directions include using the representative influencer nodes as focal points of regressing track parameters and other higher-order physics features, which are used for downstream tasks. The results presented here use an expensive loop calculation for the influencer loss function, however shortly prior to writing, a vectorized version of this loss was implemented and significantly faster training is observed. Future work will present results with this implementation. Finally, the inclusion of representative points natively in training opens the possibility of fully differentiable training of a hierarchy of condensed objects, such that representative track-like nodes of condensed hits may be further condensed into representative decay chain-like nodes, and so on. This also presents intriguing directions for pooling and unpooling in generative point cloud architectures.

7 Acknowledgements

This work is supported by the US DoE's Office of Science, under contract DE-AC02-05CH11231 (CompHEP Exa.TrkX) and the Exascale Computing Project (17-SC-20-SC). 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. I am grateful to Paolo Calafiura for presenting this work on my behalf, and Lindsey Gray for pointing out the possibility of vectorising the influencer loss function.

References

[1] H. Qu, L. Gouskos, Physical Review D **101** (2020)

[2] V. Mikuni, F. Canelli, The European Physical Journal Plus **135** (2020)

[3] P.T. Komiske, E.M. Metodiev, J. Thaler, Journal of High Energy Physics **2019** (2019)

[4] C. Shimmin, *Particle convolution for high energy physics* (2021), 2107.02908

[5] Tech. rep., CERN, Geneva (2020), all figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2020-014, https://cds.cern.ch/record/2718948

[6] D. Murnane, *Graph structure from point clouds: Geometric attention is all you need* (2023), 2307.16662

[7] K. Goto, T. Suehara, T. Yoshioka, M. Kurata, H. Nagahara, Y. Nakashima, N. Takemura, M. Iwasaki, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **1047**, 167836 (2023)

[8] Tech. rep., CERN, Geneva (2017), all figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-003, http://cds.cern.ch/record/2255226

[9] J.S.H. Lee, I. Park, I.J. Watson, S. Yang, Journal of the Korean Physical Society **74**, 219 (2019)

[10] R. Acciarri, C. Adams, R. An, J. Asaadi, M. Auger, L. Bagby, B. Baller, G. Barr, M. Bass, F. Bay et al., Journal of Instrumentation **12**, P03011 (2017)

[11] J. Shlomi, P. Battaglia, J.R. Vlimant, Machine Learning: Science and Technology **2**, 021001 (2021)

[12] S. Thais, P. Calafiura, G. Chachamis, G. DeZoort, J. Duarte, S. Ganguly, M. Kagan, D. Murnane, M.S. Neubauer, K. Terao, *Graph neural networks in particle physics: Implementations, innovations, and challenges* (2022), 2203.12852

[13] S.R. Qasim, J. Kieseler, Y. Iiyama, M. Pierini, The European Physical Journal C **79** (2019)

[14] N. Choma, D. Murnane, X. Ju, P. Calafiura, S. Conlon, S. Farrell, Prabhat, G. Cerati, L. Gray, T. Klijnsma et al., *Track seeding and labelling with embedded-space graph neural networks* (2020), 2007.00149

[15] S. Caillou, P. Calafiura, S.A. Farrell, X. Ju, D.T. Murnane, C. Rougier, J. Stark, A. Vallier (ATLAS), Tech. rep., CERN, Geneva (2022), https://cds.cern.ch/record/2815578

[16] A. Elabd, V. Razavimaleki, S.Y. Huang, J. Duarte, M. Atkinson, G. DeZoort, P. Elmer, S. Hauck, J.X. Hu, S.C. Hsu et al., Frontiers in Big Data **5** (2022)

[17] H. Qu, C. Li, S. Qian, *Particle transformer for jet tagging* (2022), 2202.03772

[18] V. Mikuni, F. Canelli, Machine Learning: Science and Technology **2**, 035027 (2021)

[19] K. Yagoh, K. Ogawara, S.i. Iida, *The particle tracking method using the kalman filter*, in *Flow Visualization VI: Proceedings of the Sixth International Symposium on Flow Visualization, October 5–9, 1992, Yokohama, Japan* (Springer, 1992), pp. 838–842

[20] B. Ristic, S. Arulampalam, N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications* (Artech house, 2003)

[21] K. He, X. Zhang, S. Ren, J. Sun, *Deep residual learning for image recognition* (2015), 1512.03385

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, in *Computer Vision – ECCV 2016* (Springer International Publishing, 2016), pp. 21–37, https://doi.org/10.1007%2F978-3-319-46448-0_2

[23] J. Redmon, A. Farhadi, *Yolo9000: Better, faster, stronger* (2016), 1612.08242

[24] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications* (2017), 1704.04861

[25] J. Redmon, A. Farhadi, *Yolov3: An incremental improvement* (2018), 1804.02767

[26] K. He, G. Gkioxari, P. Dollár, R. Girshick, *Mask r-cnn* (2018), 1703.06870

[27] T.Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, *Focal loss for dense object detection* (2018), 1708.02002

[28] C.R. Qi, H. Su, K. Mo, L.J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation* (2017), 1612.00593

[29] H. Zhao, L. Jiang, J. Jia, P. Torr, V. Koltun, *Point transformer* (2021), 2012.09164

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

[31] I. Kim, T.W. Valente, Connections **40**, 129 (2021)

[32] R. Recuero, G. Zago, F. Soares, Social media+ society **5**, 2056305119848745 (2019)

[33] S. Amrouche, L. Basara, P. Calafiura, V. Estrade, S. Farrell, D.R. Ferreira, L. Finnie, N. Finnie, C. Germain, V.V. Gligorov et al., in *The NeurIPS '18 Competition* (Springer International Publishing, 2019), pp. 231–264, https://doi.org/10.1007%2F978-3-030-29135-8_9