Equivariant Graph Neural Networks for Charged Particle Tracking

Daniel Murnane[1]**, Savannah Thais**[2]**, Ameya Thete**[3]

[1] Scientific Data Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[2] Data Science Institute, Columbia University, New York, NY 10027, USA
[3] Department of Physics, BITS, Pilani – KK Birla Goa Campus, Zuarinagar, Goa 403726, India

E-mail: `dtmurnane@lbl.gov`

Abstract. Graph neural networks (GNNs) have gained traction in high-energy physics (HEP) for their potential to improve accuracy and scalability. However, their resource-intensive nature and complex operations have motivated the development of symmetry-equivariant architectures. In this work, we introduce EuclidNet, a novel symmetry-equivariant GNN for charged particle tracking. EuclidNet leverages the graph representation of collision events and enforces rotational symmetry with respect to the detector's beamline axis, leading to a more efficient model. We benchmark EuclidNet against the state-of-the-art Interaction Network on the TrackML dataset, which simulates high-pileup conditions expected at the High-Luminosity Large Hadron Collider (HL-LHC). Our results show that EuclidNet achieves near-state-of-the-art performance at small model scales (< 1000 parameters), outperforming the non-equivariant benchmarks. This study paves the way for future investigations into more resource-efficient GNN models for particle tracking in HEP experiments.

1 Introduction

In recent years there has been a sharp increase in the use of graph neural networks (GNNs) for high-energy physics (HEP) analyses [1]. These studies have demonstrated that GNNs have the potential to deliver large improvements in accuracy, and can easily scale to sizable volumes of data. However, many of these architectures either involve a large number of parameters or complex graph operations and convolutions, which make GNNs resource-intensive and time-consuming to deploy. Many real-world datasets, including those from HEP experiments, display known symmetries, which can be used to construct alternatives to computationally expensive unconstrained architectures. By exploiting the inherent symmetry in a given problem, we can restrict the function space of neural networks to relevant candidates by enforcing equivariance with respect to transformations belonging a certain symmetry group. This approach has a two-fold benefit: incorporating equivariance introduces inductive biases into the neural network, and more often than not, equivariant models are more resource-efficient than their non-equivariant counterparts [2, 3].

In this work, we introduce a new architecture of symmetry-equivariant GNNs for charged particle tracking. Using a graph representation of a collision event, we propose EuclidNet, which scalarizes input tracking features to enforce rotational symmetry. In particular, given the detector's symmetry around the beamline (z) axis, we develop a formulation of EuclidNet equivariant to the SO(2) rotation group. Benchmarked against the current state-of-the-art

(SoTA) Interaction Network, EuclidNet achieves near-SoTA performance at small model sizes. We explore the out-of-distribution inference performance of EuclidNet and InteractionNet in order to provide an explanation for the behavior of model performance versus model size. As such, we reveal hints not only as to the upper ceiling on fully-equivariant models, but the symmetries learned by non-equivariant models. The code and models are publicly available at https://github.com/ameya1101/equivariant-tracking.

2 Theory and Background

In this section, we introduce the concept of symmetry-group equivariance and provide a short introduction to the theory of graph neural networks.

2.1 Equivariance

Formally, if $T_g : X \to X$ is a set of transformations on a vector space X for an abstract symmetry group $g \in G$, a function $\phi : X \to Y$ is defined to be equivariant to g if there exists an equivalent set of transformations on the output space $S_g : Y \to Y$ such that:

$$\phi(T_g(\mathbf{x})) = S_g\phi(\mathbf{x}) \tag{2.1}$$

A model is said to be equivariant to a group G if it is composed of functions ϕ that are equivariant to G. In this work, we limit our discussion to rotational equivariance corresponding to the SO(2) group. As an example, let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M)$ be a set of M points embedded in $n-$dimensional space, and $\phi(\mathbf{x}) = \mathbf{y} \in \mathbb{R}^{M \times n}$ be the transformed set of points. For an orthogonal rotation matrix $Q \in \mathbb{R}^{n \times n}$, ϕ is equivariant to rotations if $Q\mathbf{y} = \phi(Q\mathbf{x})$.

2.2 Graph Neural Networks

Consider a graph $\mathcal{G} = (V, E)$ with nodes $v_i \in V$ and edges $e_{ij} \in E$. A graph neural network is a permutation-invariant deep learning architecture that operates on graph-structured data [4]. A GNN commonly consists of multiple layers, with each layer performing the graph convolution operation, which is defined as [5]:

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij}) \tag{2.2}$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \tag{2.3}$$

$$\mathbf{h_i^{l+1}} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i) \tag{2.4}$$

where $\mathbf{h}_i^l \in \mathbb{R}^k$ is the k-th dimensional embedding of node v_i at layer l. a_{ij} are edge attributes. $\mathcal{N}(i)$ is the set of neighbors of node v_i. Finally, ϕ_e and ϕ_h are edge and node operations which are approximated by multi-layer perceptrons (MLPs). $\mathbf{m}_{ij}$ are called messages that are passed between nodes v_i and v_j.

3 Network Architecture

In this section, we describe the architecture of EuclidNet. As described in Figure 1, EuclidNet is constructed by stacking Euclidean Equivariant Blocks (EEB) along with some encoding and decoding layers. The architecture of EuclidNet closely follows that of LorentzNet presented in [6]. In this section, $\phi(a, b, \ldots, f)$ implies that the quantities a through f are concatenated before being passed to ϕ.

Input layer. The inputs to the network are 3D hit positions. For the SO(2) group, scalars are the $z-$coordinate of the hit position, while the 2D coordinates form the vectors. The scalars

are projected to an embedding space using an embedding layer before being passed to the first equivariant block.
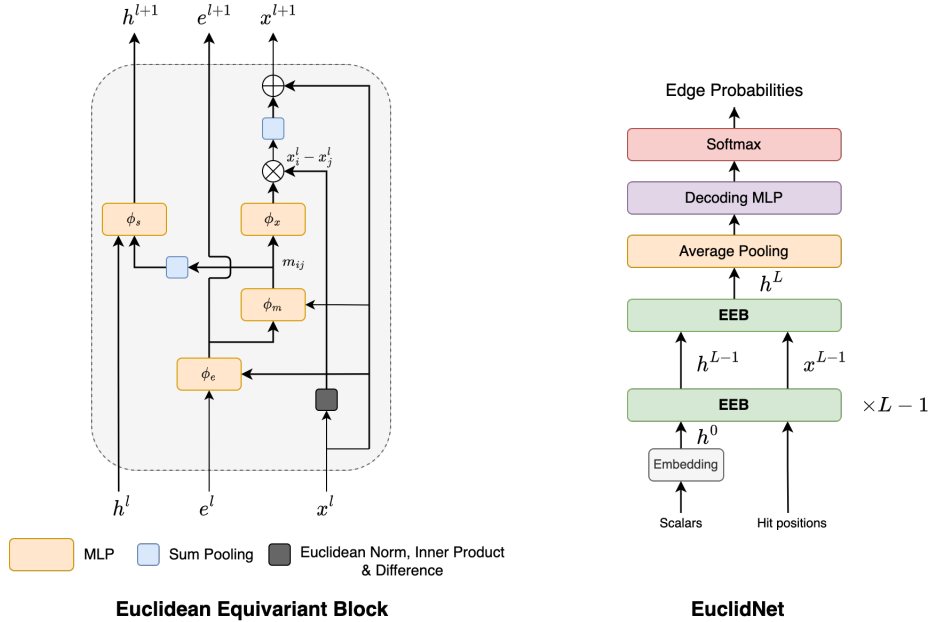

Figure 1: (**left**) The structure of the Euclidean Equivariant Block (EEB). (**right**) The network architecture of EuclidNet.

Euclidean Equivariant Block. Following convention, we use $h^l = (h_1^l, h_2^l, \ldots, h_N^l)$ to denote node embedding scalars and $x^l = (x_1^l, x_2^l, \ldots, x_N^l)$ to denote coordinate embedding vectors in the $l-$th EEB layer. x^0 corresponds to the hit positions and h^0 corresponds to the embedded input of scalar variables. The message m_{ij}^l to be passed is constructed as follows:

$$m_{ij}^l = \phi_m \left(h_i^l, h_j^l, e_{ij}^l, \psi(\|x_i^l - x_j^l\|^2), \psi(\langle x_i^l, x_j^l \rangle) \right) \tag{3.1}$$

$$e_{ij}^l = \phi_e \left(\psi(\|x_i^l - x_j^l\|^2), \psi(\langle x_i^l, x_j^l \rangle), e_{ij}^{l-1} \right) \tag{3.2}$$

where $\phi_m(\cdot)$ is a neural network and $\psi(\cdot) = \text{sgn}(\cdot)\log(|\cdot|+1)$ normalizes large numbers from broad distributions to ease training. The input to ϕ_m also contains the Euclidean dot product $\langle x_i^l, x_j^l \rangle$. The Euclidean distance $\|x_i^l - x_j^l\|^2$ between hits is an important feature and we include it for ease of training. e_{ij}^l is an edge significance weight learnt by an MLP.

We use an Euclidean formulation of the dot product attention from [6] as the aggregation function, which is defined as:

$$x_i^{l+1} = x_i^l + c \sum_{j \in \mathcal{N}(i)} \phi_x(m_{ij}^l) \cdot (x_i^l - x_j^l) \tag{3.3}$$

where $\phi_x(\cdot) \in \mathbb{R}$ is a scalar function modeled by an MLP. The hyperparameter c is introduced to control the scale of the updates. The scalar features, h_i^l, are updated as:

$$h_i^{l+1} = h_i^l + \phi_h \left(h_i^l, \sum_{j \in \mathcal{N}(i)} m_{ij}^l \right) \tag{3.4}$$

Decoding layer. After L stacks of EEBs, we decode the node embedding $h^L = (h_1^L, h_2^L, \ldots, h_N^L)$. Message passing ensures that the information contained in the vector embeddings is propagated to scalars, therefore it is redundant to decode both. A decoding block with three fully connected layers, followed by a softmax function is used to generate a truth score for each track segment.

4 Results

4.1 Dataset

In this study, we test the developed tracking models on the TrackML dataset, which is a simulated set of proton-proton collision events developed for the TrackML Particle Tracking Challenge [7]. Events are generated with 200 pileup interactions on average, simulating the high pileup conditions expected at the HL-LHC. Each event contains three-dimensional hit positions and truth information about the particles that generated them. In this work, we limit our discussion to the pixel layers only which consist of a highly-granular set of four barrel and fourteen endcap layers in the innermost region. Each event's tracker hits are converted to a hitgraph through an edge construction algorithm. In addition to transverse momentum, noise, and same-layer filters to modulate the number of hits, graph edges are also required to satisfy constraints on certain geometrical quantities. In this study, we use the geometric graph construction strategy from [8] to generate graphs, with $p_T^{\min} = 1.5$ GeV for each event in the dataset.

4.2 Experiments

We train EuclidNet and the Interaction Network for three different values of hidden channels: 8, 16, and 32. The results obtained on the TrackML dataset are summarized in Table 1. We evaluate the models using the Area Under the ROC curve (AUC), which is a commonly used metric in classification problems. We also report the number of model parameters, as well as the purity and efficiency of the resulting event graphs.

Table 1: Performance comparison between EuclidNet and the Interaction Network (IN) on the TrackML dataset. The results for EuclidNet and IN are averaged over 5 runs

N_{hidden}	Model	Params	AUC	Efficiency	Purity
8	EuclidNet	967	**0.9913 ± 0.004**	**0.9459 ± 0.022**	**0.7955 ± 0.040**
	InteractionNet	1432	0.9849 ± 0.006	0.9314 ± 0.021	0.7319 ± 0.052
16	EuclidNet	2580	0.9932 ± 0.003	0.9530 ± 0.014	**0.8194 ± 0.033**
	InteractionNet	4392	0.9932 ± 0.004	**0.9575 ± 0.019**	0.8168 ± 0.073
32	EuclidNet	4448	0.9941 ± 0.003	0.9547 ± 0.019	0.9264 ± 0.023
	InteractionNet	6448	**0.9978 ± 0.003**	**0.9785 ± 0.022**	**0.9945 ± 0.043**

For small model sizes, EuclidNet easily outperforms the unconstrained Interaction Network. We see that rotational symmetry is approximately obeyed in the detector geometry as simulated in the TrackML dataset (the "Generic Detector"). Using this symmetry appears, to first order, to produce an accurate pair-classification algorithm. However, given a larger latent space, the performance assuming this symmetry plateaus. At that point, the unconstrained network is more performant. We initially hypothesized that there are non-symmetric components of the simulation, for example material and magnetic field inhomogeneities. To study this possibility, we train both models on a set of rotations of the dataset by $\theta \in [0, \pi/4, \pi/2, 3\pi/4, \pi]$, and run inference of each of these instances across the set of rotated datasets. In this way, we can capture whether the unconstrained network is learning a function specific to that orientation of
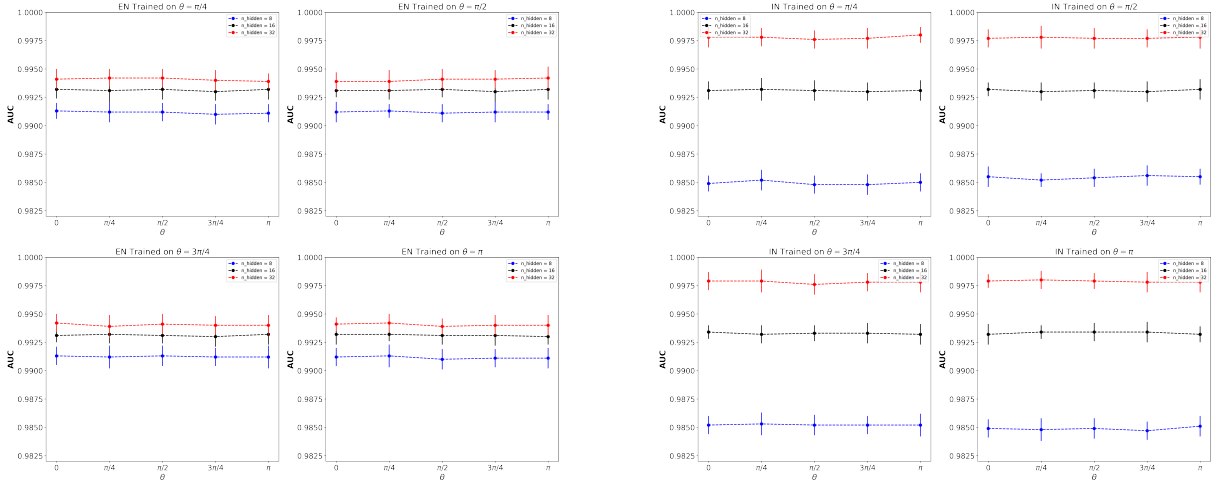
Figure 2: The AUC at inference time plotted as a function of rotations in the input space by an angle θ for models trained on a set of rotations of the dataset by $\theta \in [0, \pi/4, \pi/2, 3\pi/4, \pi]$ for (**left**) EuclidNet and (**right**) the Interaction Network calculated over 5 independent inference runs.

the detector's material and magnetic field. The results of this study are summarized in Figure 2.

In Figure 2, we observe that both the SO(2)-equivariant EuclidNet and the Interaction Network are robust to rotations in input space, irrespective of which set of rotations of the dataset the model was trained on. The general trend observed in Table 1 is also replicated here, with models having a larger latent space producing larger AUC scores. Although these results are inconclusive to fully explain the unconstrained network's superior performance, we posit that they might be a symptom of the dataset's inherent approximate rotational symmetry. If the dataset were only "roughly" rotationally symmetric, then a less expressive EuclidNet would not be able to completely match the approximate symmetry of the event, and would outperform the IN only at latent space sizes that are not sufficient to capture the complete symmetry landscape. However, given a large enough latent space, an unconstrained network such as the IN will have no problems learning the approximate symmetry of the problem. In the future, an accurate study to determine the actual symmetry learnt by both EuclidNet and the IN is required to ascertain the cause for this behaviour.

5 Conclusions and Future Work

In this study, we have presented EuclidNet — a Euclidean rotation-equivariant GNN for the particle tracking problem. The SO(2)-equivariant model offers a marginal improvement (AUC = 0.9913) over the current SoTA benchmark (AUC = 0.9849) at small model scales (< 1000 parameters). However, for the particle tracking problem, we find that an unconstrained model still outperforms an equivariant architecture at larger model sizes. More work is needed to concretely establish the reasons for this result. Possible future directions include studying the problem's equivariance and identifying non-equivariant facets, if they exist, as well as investigating the quality of the learnt symmetry. However, if the dataset inherently contains non-equivariant features, any symmetric model will always underperform. In this case, models which relax the strict constraints imposed by symmetry-following architectures might be able to learn the non-equivariant aspects of the tracking dataset.

Acknowledgements

Appendix A   Training Details

We use the binary cross-entropy loss function for both EuclidNet and the Interaction Network. All models are optimized using the Adam optimizer [9] implemented in PyTorch with a learning rate $\eta = 1 \times 10^{-3}$ and momentum coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$. The learning rate is decayed by 0.3 every 100 epochs. Both are trained on a single NVIDIA A100 GPU each for a 200 epochs using early stopping with a patience of 50 epochs. Total training time for both models is typically 2 hours.

Appendix B   Equivariance Tests

We also test the equivariance of EuclidNet and find that it is indeed equivariant to SO(2) transformations up to floating-point numerical errors. For a given transformation $R \in SO(2)$, we compare $R_\theta \phi(\mathbf{x})$ and $\phi(R_\theta \mathbf{x})$, where $\mathbf{x}$ are the hit coordinates and $\phi(\cdot)$ is a EuclidNet instance. In Figure B1 we plot the relative deviation, defined as

$$\varepsilon(\theta) = \frac{\langle \phi(R_\theta \mathbf{x}) \rangle - \langle R_\theta \phi(\mathbf{x}) \rangle}{\langle R_\theta \phi(\mathbf{x}) \rangle} \tag{B.1}$$

where $\langle \cdot \rangle$ is the mean computed over 500 tracking events in the test dataset. We find the relative deviation from rotations to be $< 10^{-8}$.
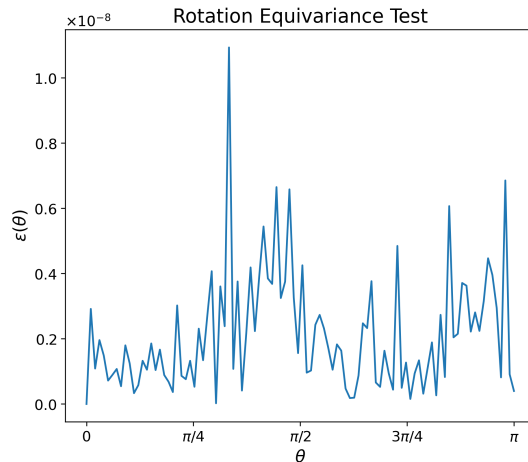

Figure B1: Relative deviation of the output node vector representations to rotations around the beamline (z-axis).

References

[1] Thais S, Calafiura P, Chachamis G, DeZoort G, Duarte J, Ganguly S, Kagan M, Murnane D, Neubauer M S and Terao K 2022 *2022 Snowmass Summer Study* (*Preprint* 2203.12852)

[2] Satorras V G, Hoogeboom E and Welling M 2021 *International Conference on Machine Learning* (PMLR) pp 9323–9332

[3] Bogatskiy A *et al.* 2022 *2022 Snowmass Summer Study* (*Preprint* 2203.06153)

[4] Kipf T N and Welling M 2016 (*Preprint* 1609.02907)

[5] Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 *International conference on machine learning* (PMLR) pp 1263–1272

[6] Gong S, Meng Q, Zhang J, Qu H, Li C, Qian S, Du W, Ma Z M and Liu T Y 2022 *JHEP* **07** 030 (*Preprint* 2201.08187)

[7] Amrouche S *et al.* 2019 *The Tracking Machine Learning challenge : Accuracy phase* (*Preprint* 1904.06778)

[8] DeZoort G, Thais S, Duarte J, Razavimaleki V, Atkinson M, Ojalvo I, Neubauer M and Elmer P 2021 *Comput. Softw. Big Sci.* **5** 26 (*Preprint* 2103.16701)

[9] Kingma D P and Ba J 2014 (*Preprint* 1412.6980)