Equivariant GNNs for Charged Particle Tracking

IRIS-HEP Fellows Presentations

Ameya Thete[1]

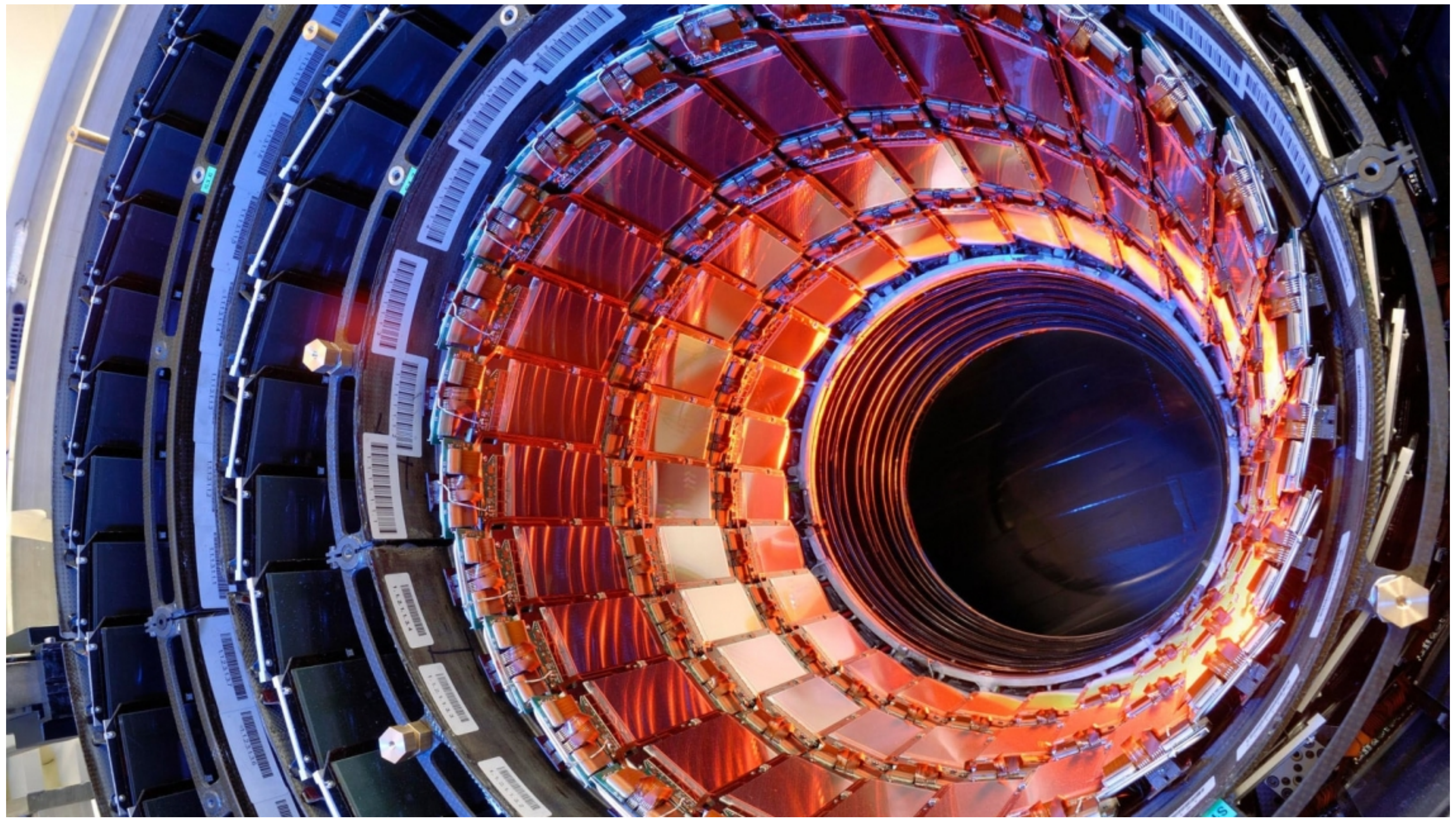
Mentors: Savannah Thais[2], Daniel Murnane[3]

[1]BITS, Pilani - K.K. Birla Goa Campus
[2]Columbia University
[3]Lawrence Berkeley National Laboratory

19 October 2022

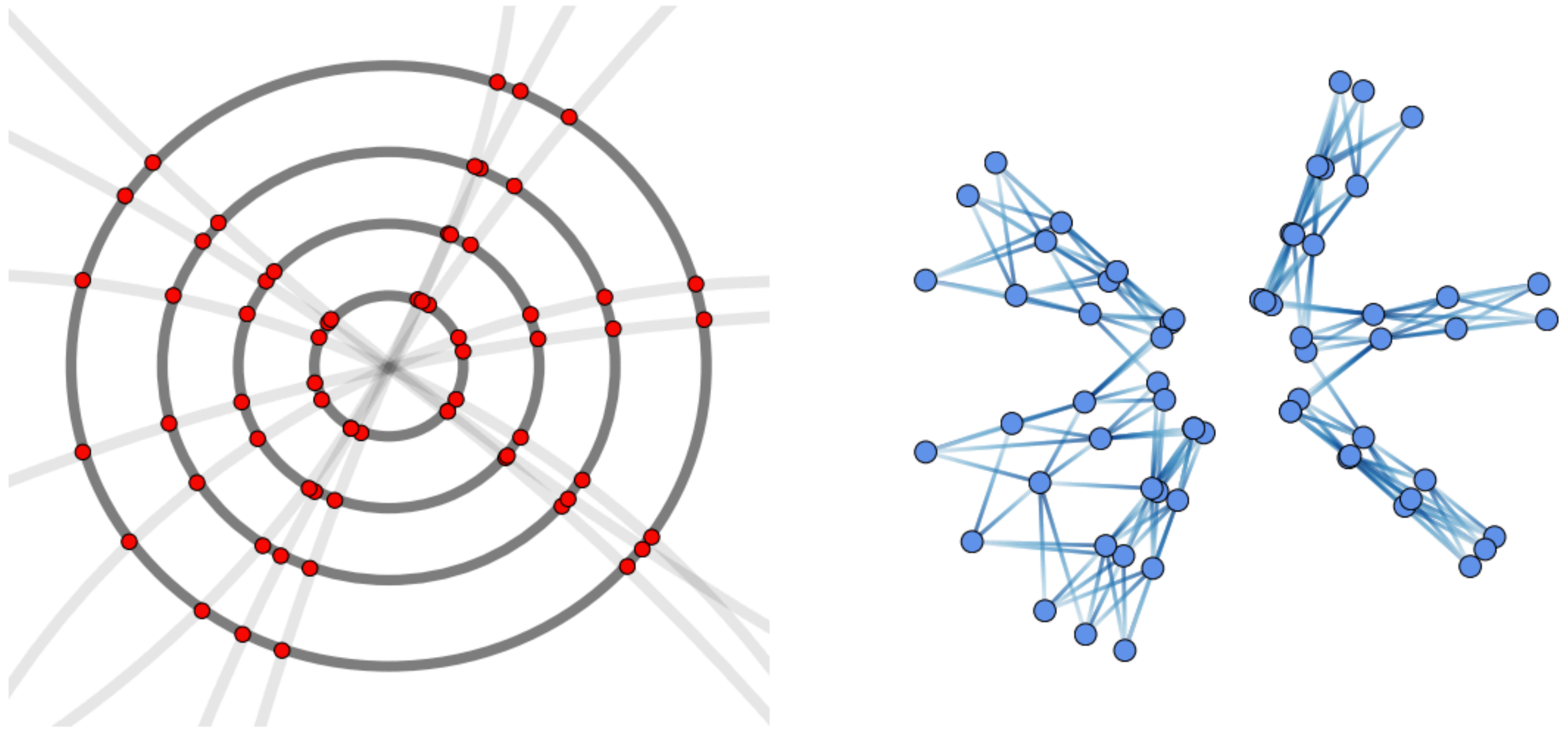
Background

- High-energy collisions produce a huge number of particles; we want to see these particles, which leave "**tracks**"
- Charged particles produced in collisions at the LHC generate energy deposits called "**hits**" in tracker layers
- Track reconstruction involves clustering these hits to trace out the trajectories of these charged particles
- Current track-reconstruction algorithms scale poorly with detector occupancy; unsuitable for HL-LHC applications!


The CMS Tracker showing silicon strip detectors in the barrel module [CERN]

Graph Neural Networks (GNNs)

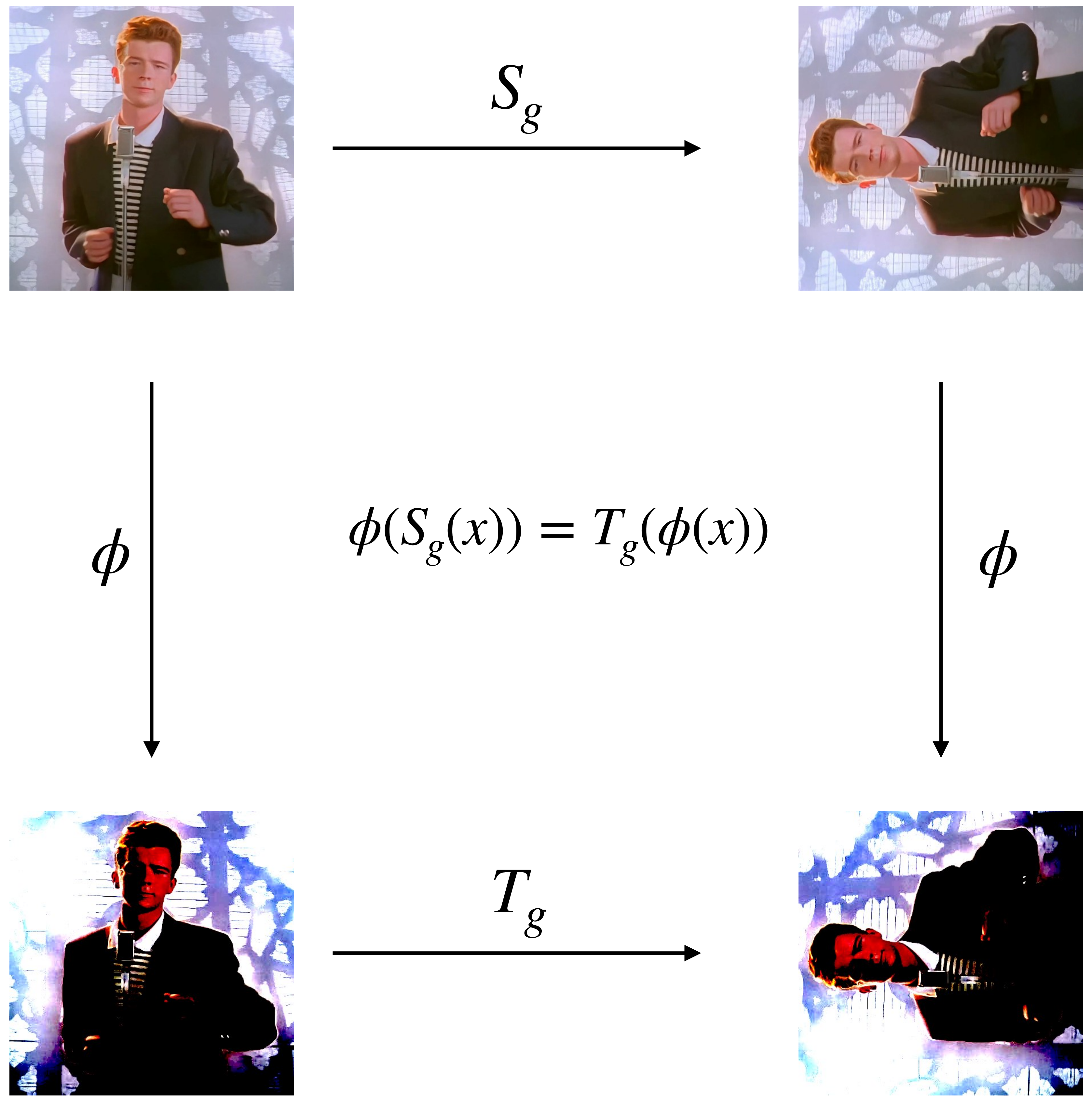
- **Graphs** can capture inherent sparsity and relational structure of most physics data
- Tracking data is naturally represented as graphs by identifying hits as nodes and track segments as edges.
- GNNs are a class of geometric DL algorithms that operate on graphs
- Recent progress with GNNs demonstrates that edge-classifying GNNs are suitable for particle tracking [Exa.TrkX (2019); DeZoort et al. (2021)]



Segments of hits in a tracking detector can be mapped to a graph with the nodes representing the hits and the edges representing potential track segments [Duarte and Vlimant (2022)]

Project Summary

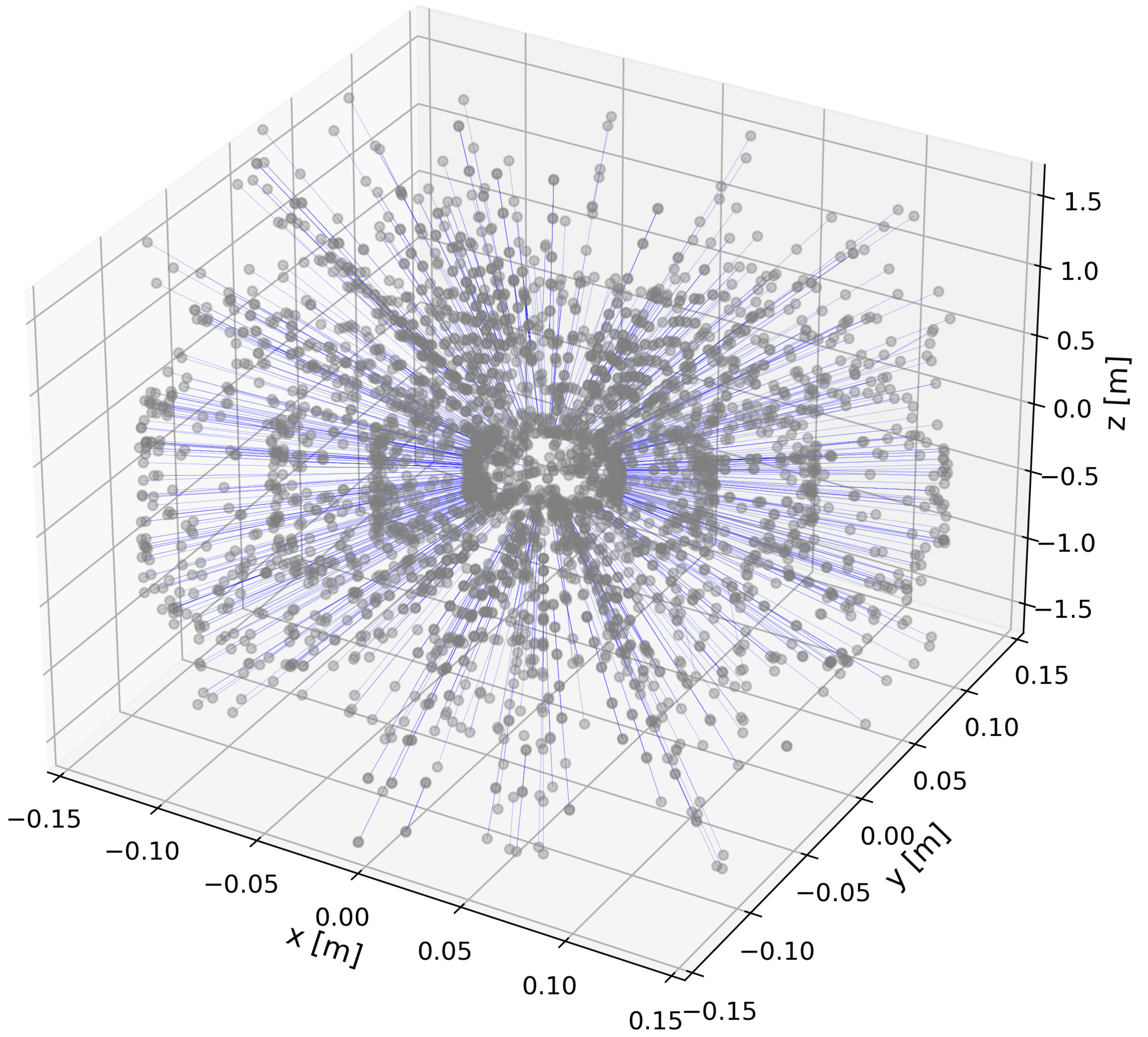
- The primary aim of the project is to study **equivariant formulations** of GNNs for the particle tracking problem.
- Such a formulation would be unaffected by symmetry group transformations of the input space.
- Equivariant models have been shown to outperform non-equivariant counterparts [Satorras et al. (2021); Gong et al. (2022)].
    - For example, Lorentz-equivariant taggers achieve state-of-the-art tagging accuracies and background rejection rates, require fewer training samples and have faster inference times.
    - Constraining the model restricts its hypothesis space; aside from imparting inductive biases, this also has the potential to reduce the number of trainable parameters and inference times.



Example of rotational equivariance on an image with a neural network ϕ and $S_g, T_g \in G$, the group of image rotations.

Dataset

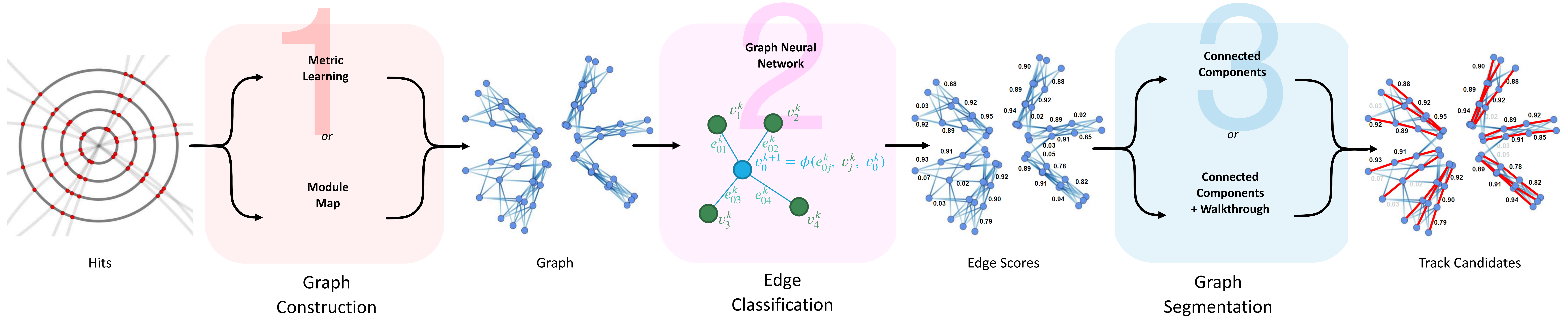
- We use the TrackML dataset: ACTS-simulated $p-p$ collision events with ~200 pileup interactions.
- Each event contains 3D hit position (x, y, z) and truth information about the particle that generated them.
- We only include hits generated in the pixel layers in the innermost region of the tracker.
- 3 filters applied to the dataset:
  - A $p_T^{\min}$ filter to reject hits generated by particles with $p_T < p_T^{\min}$
  - Noise filter to reject noise hits
  - Same-layer filter to ensure only one hit per particle per layer
- Graph constructed by mapping hits to nodes and edges to possible track segments
  - Edge e_{ij} constructed if it satisfies constraints on the geometric quantities $z_0 = z_i - r_i \dfrac{z_j - z_i}{r_j - r_i}$ and $\phi_{\text{slope}} = \dfrac{\phi_j - \phi_i}{r_j - r_i}$ [(r, ϕ, z) are detector coordinates]



An example hitgraph for an event with $p_T^{\min} = 1.5$ GeV. Blue lines denote true track segments.

Tracking Pipeline

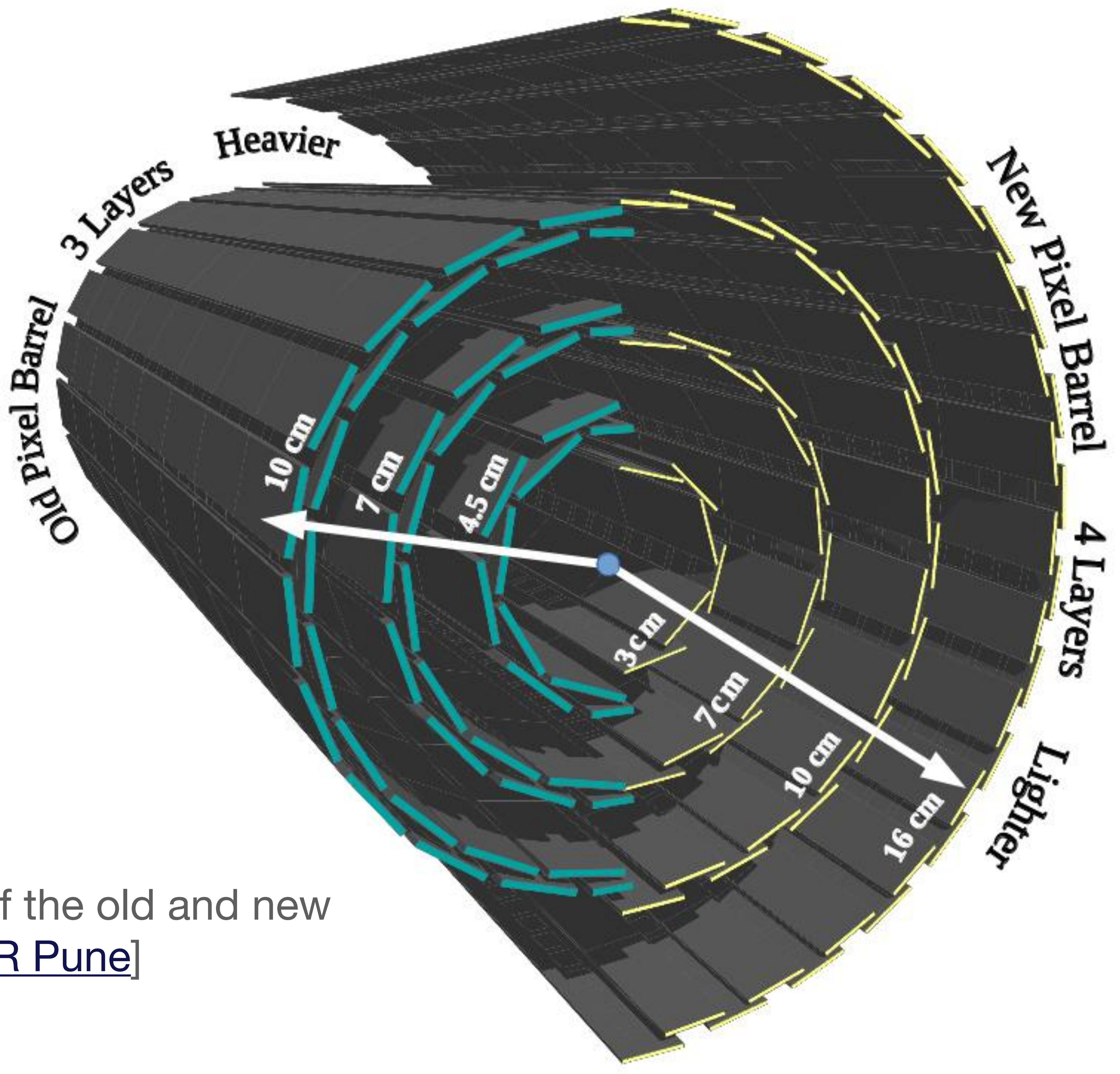
- A typical GNN-based end-to-end tracking pipeline consists of three stages:
  - Graph construction
  - Edge-classification
  - Track building
- Optimisations can be made at any of these three stages.
- In this work, we focus on the **edge-classification** step.
- We use the **geometric graph construction** strategy with a $p_T^{\min} = 1.5$ GeV cut from DeZoort et al (2021).
- Track building results are not computed for this study.



An illustration of the various steps in the tracking pipeline [Murnane (2022)]

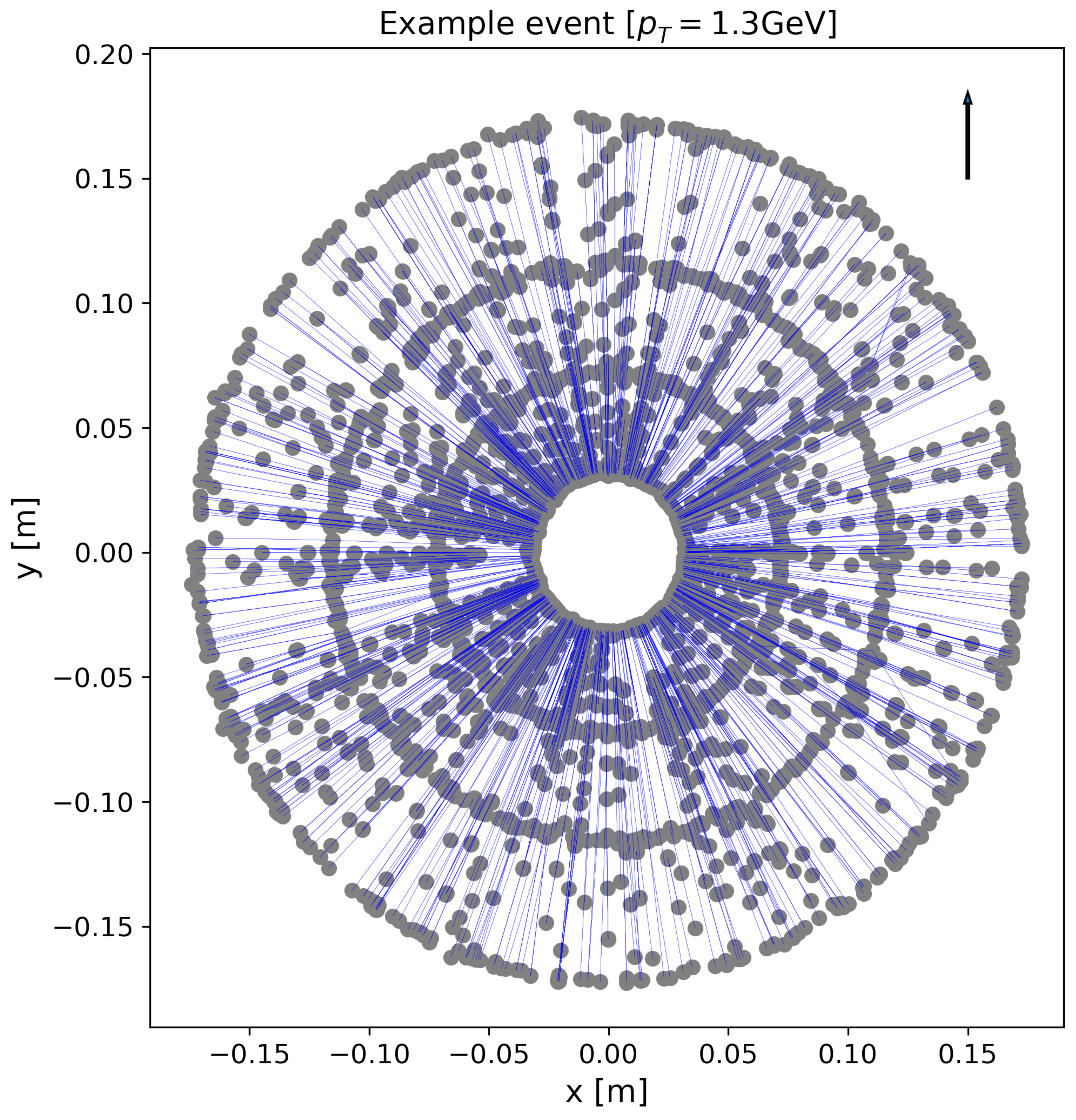
Step 1: A Long, Hard Look

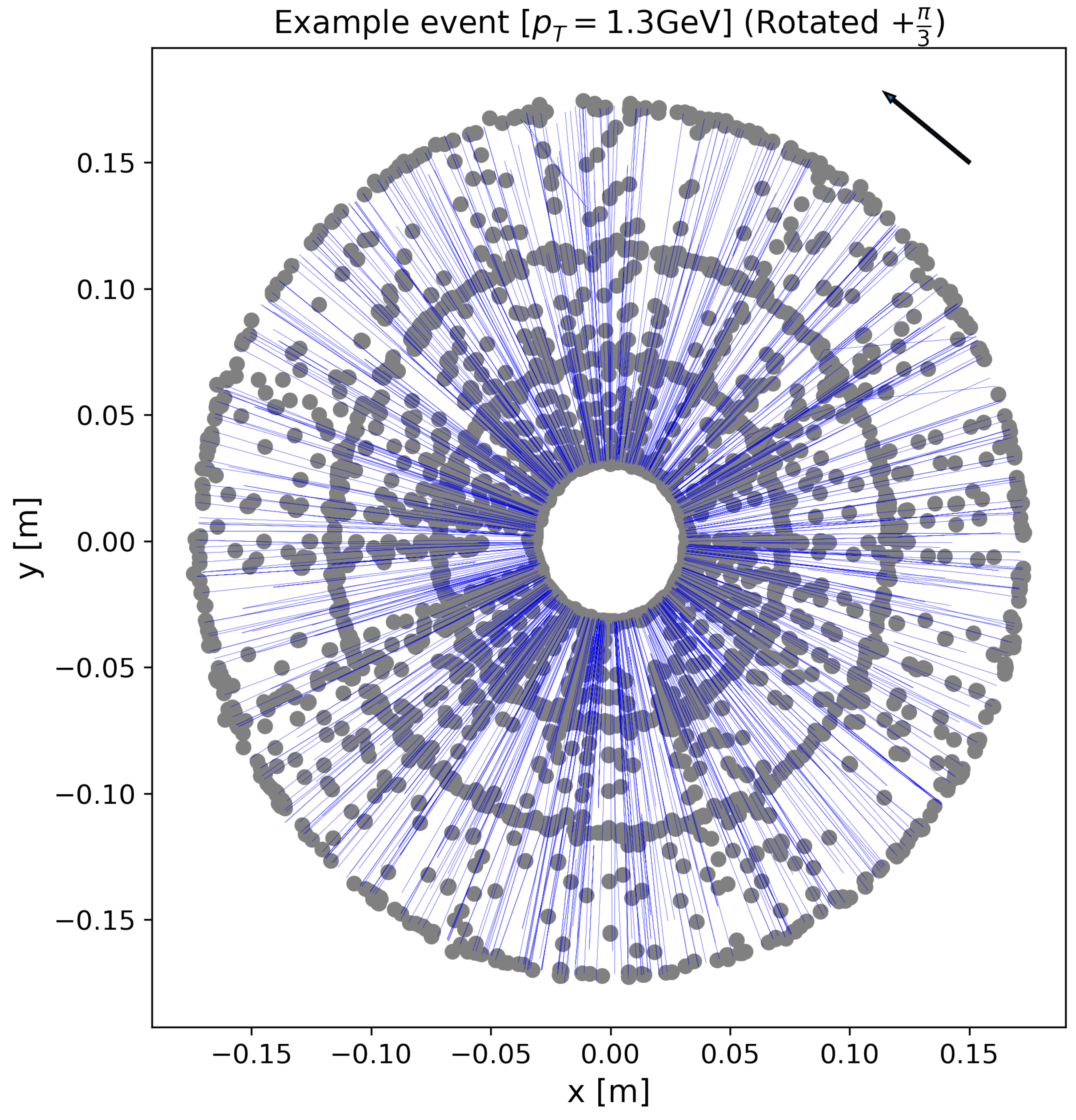
- The first task is to — rather literally — take a hard look at the problem and the dataset.
- As the image suggests, the detector has a cylindrical symmetry (in the x-y plane) — strong case for studying SO(2) group equivariance.
- Now that we've observed a symmetry in the problem, we want to consider models that incorporate this symmetry.
- The choice of symmetry also guides the architecture of the GNN.
- As most physical laws are also equivariant to rotations, we might also want to study the SO(3) symmetry group consisting of orthonormal rotations in Euclidean space.



An engineering design of the barrel layers of the old and new Pixel detector at the CMS Experiment [IISER Pune]

Step 1: A Long, Hard Look





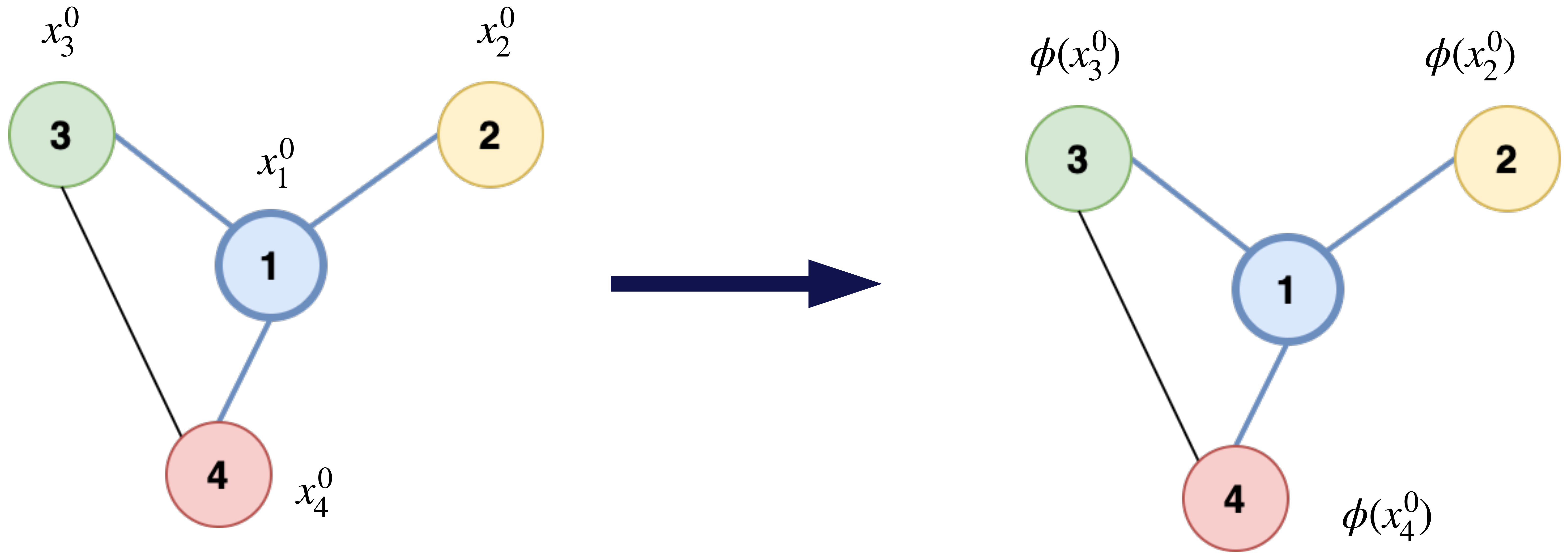
(Left) An example hitgraph for an event with $p_T^{\text{min}} = 1.5$ GeV. (Right) The same hitgraph rotated by +60° about the z-axis. Blue edges denote true track segments (false edges are not shown).

Step 2: Building the GNN

- Once the relevant symmetries have been identified, we can build the corresponding GNN architecture.
- We call the class of GNNs used in this study "**EuclidNet**", given their equivariance to Euclidean symmetry groups.
- The EuclidNet design is inspired by the LorentzNet architecture [Gong et al. (2022)]; implements the message passing formulation (cf. the matrix formulation)
- The input to the GNN is a hitgraph consisting of nodes and edges
  - All nodes have a 3D feature vector: (x, y, z) coordinates.
  - All edges have a 4D feature vector: a displacement vector + distance in detector space.
- Edge features are neither SO(2) nor SO(3) equivariant — cannot be used!

Sidenote: How does one include symmetries?

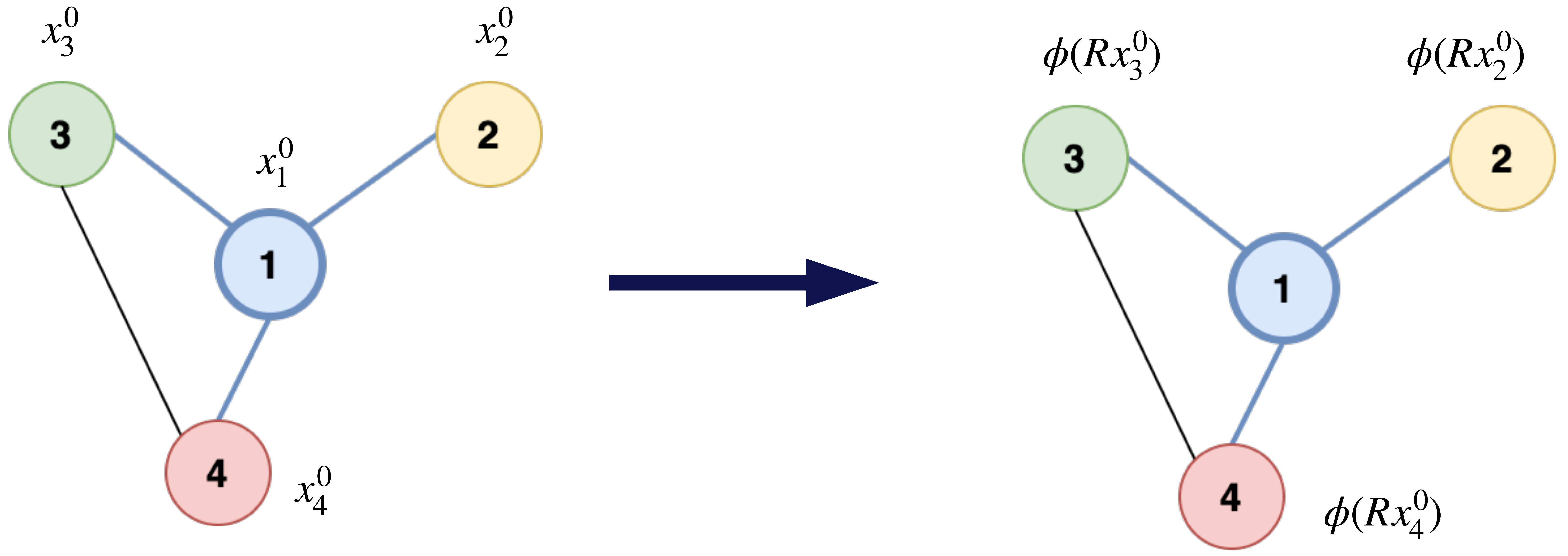
- Let's say we have a graph that we expect is equivariant under Euclidean rotations.
- Given a rotation R, under an arbitrary message passing scheme, the transformation propagates like so:



$$x_1^1 = \sum_i \phi(x_i^0)$$

Sidenote: How does one include symmetries?

- Let's say we have a graph that we expect is equivariant under Euclidean rotations.
- Given a transformation R, under an arbitrary message passing scheme, the transformation propagates like so:

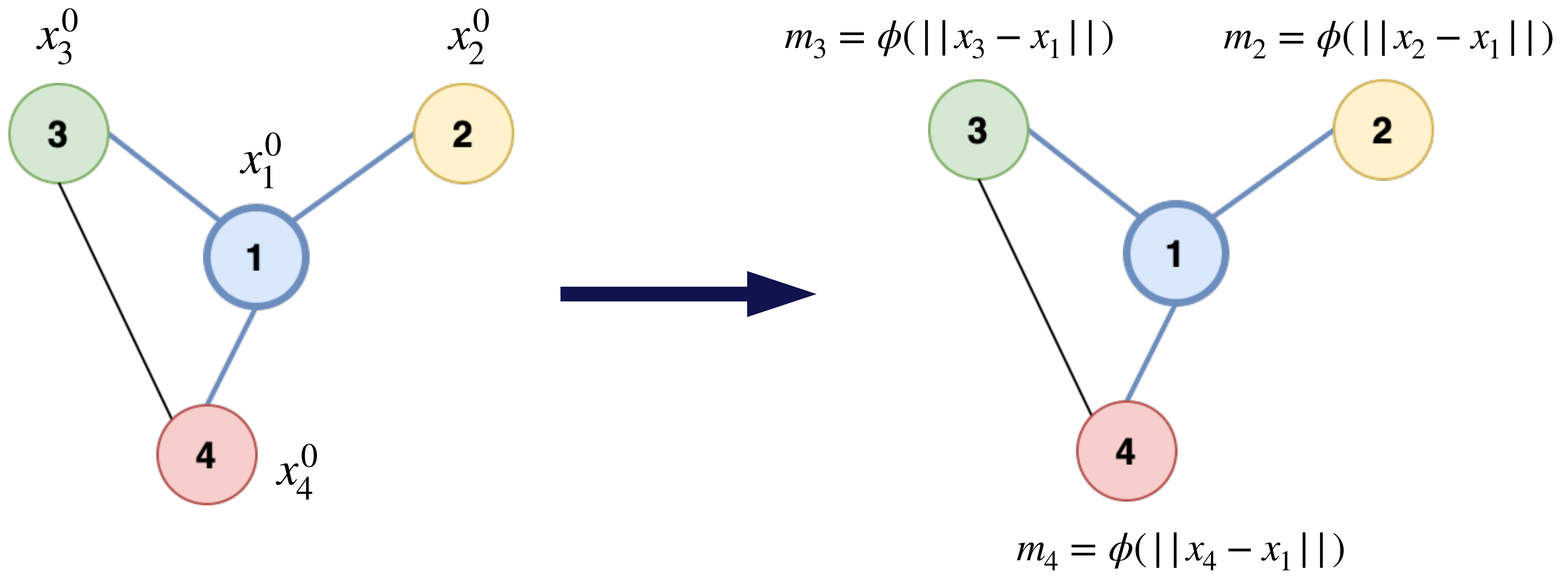


Not Equivariant!

$$\boxed{x_1^{1'} = \sum_i \phi(Rx_i^0)}$$

Sidenote: How does one include symmetries?

- To preserve rotational symmetry, we must choose a specific kind of message passing function.

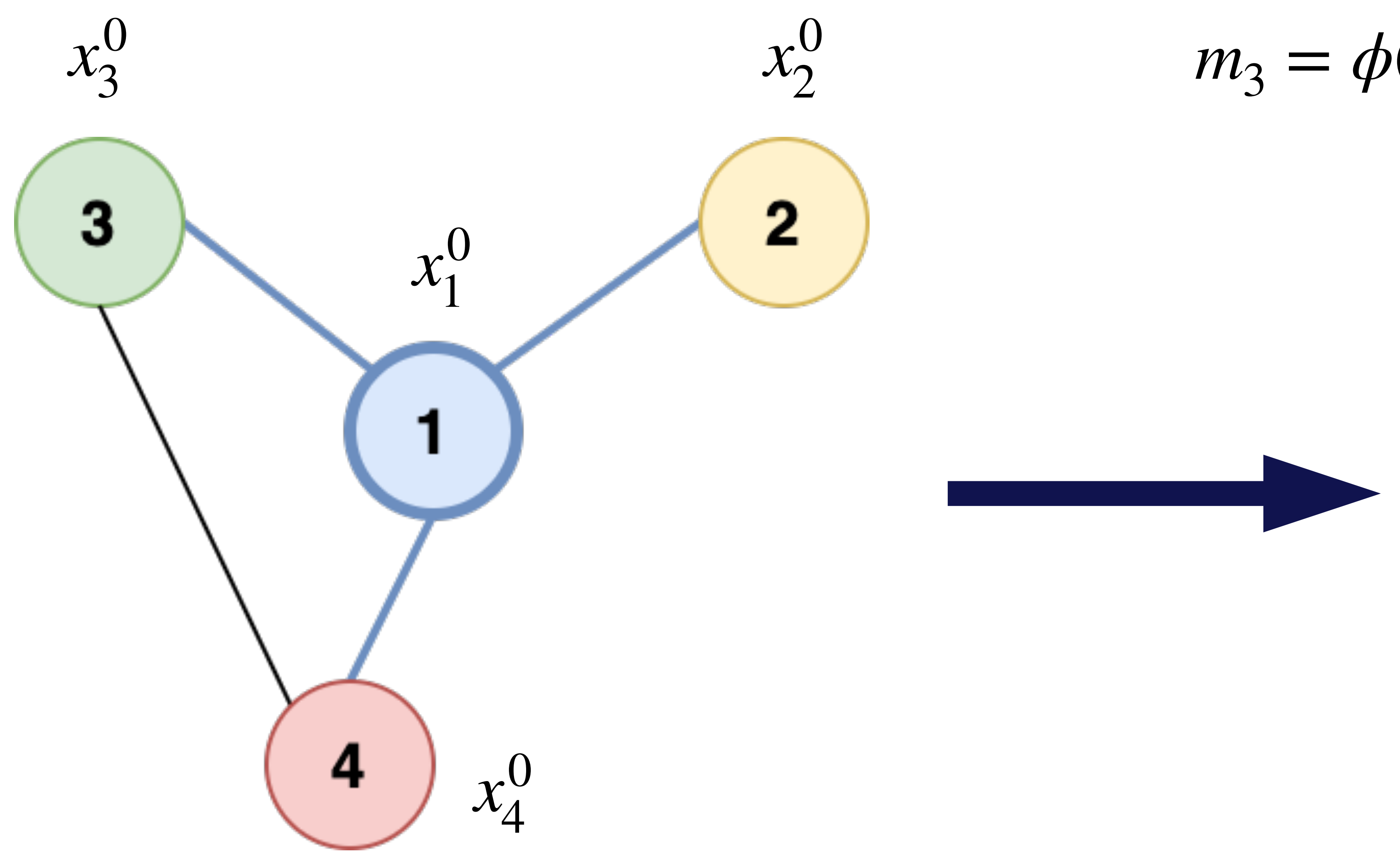


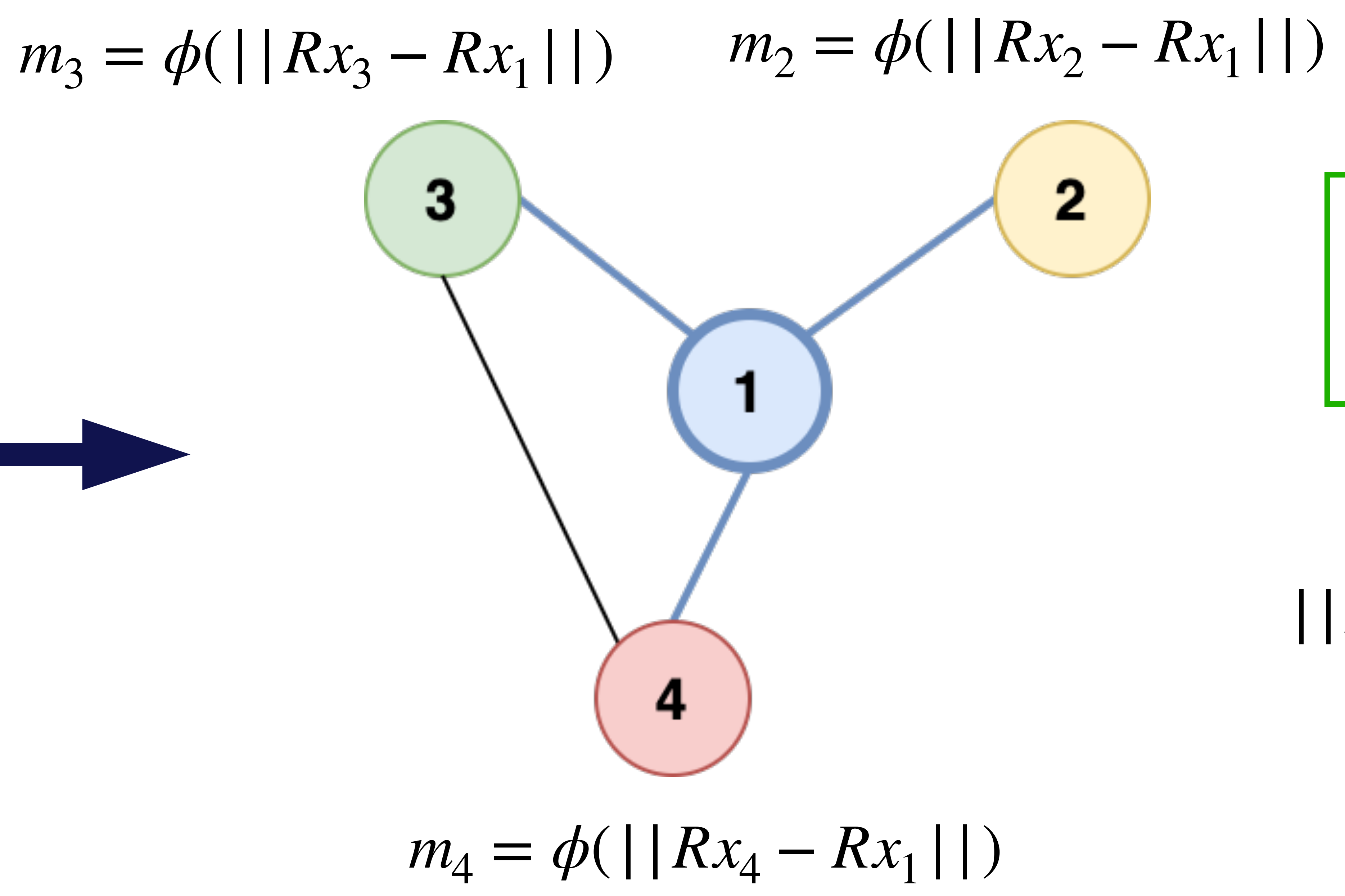
x_3^0
x_2^0
x_1^0
x_4^0
3
2
1
4
$m_3 = \phi(||x_3 - x_1||)$
$m_2 = \phi(||x_2 - x_1||)$
$m_4 = \phi(||x_4 - x_1||)$

$$x_1^1 = x_1^0 + \sum_i \phi(m_i)(x_i - x_1)$$

Sidenote: How does one include symmetries?

- To preserve rotational symmetry, we must choose a specific kind of message passing function.




$m_3 = \phi(||Rx_3 - Rx_1||)$
$m_2 = \phi(||Rx_2 - Rx_1||)$
3
2
1
4
$m_4 = \phi(||Rx_4 - Rx_1||)$

Equivariant!

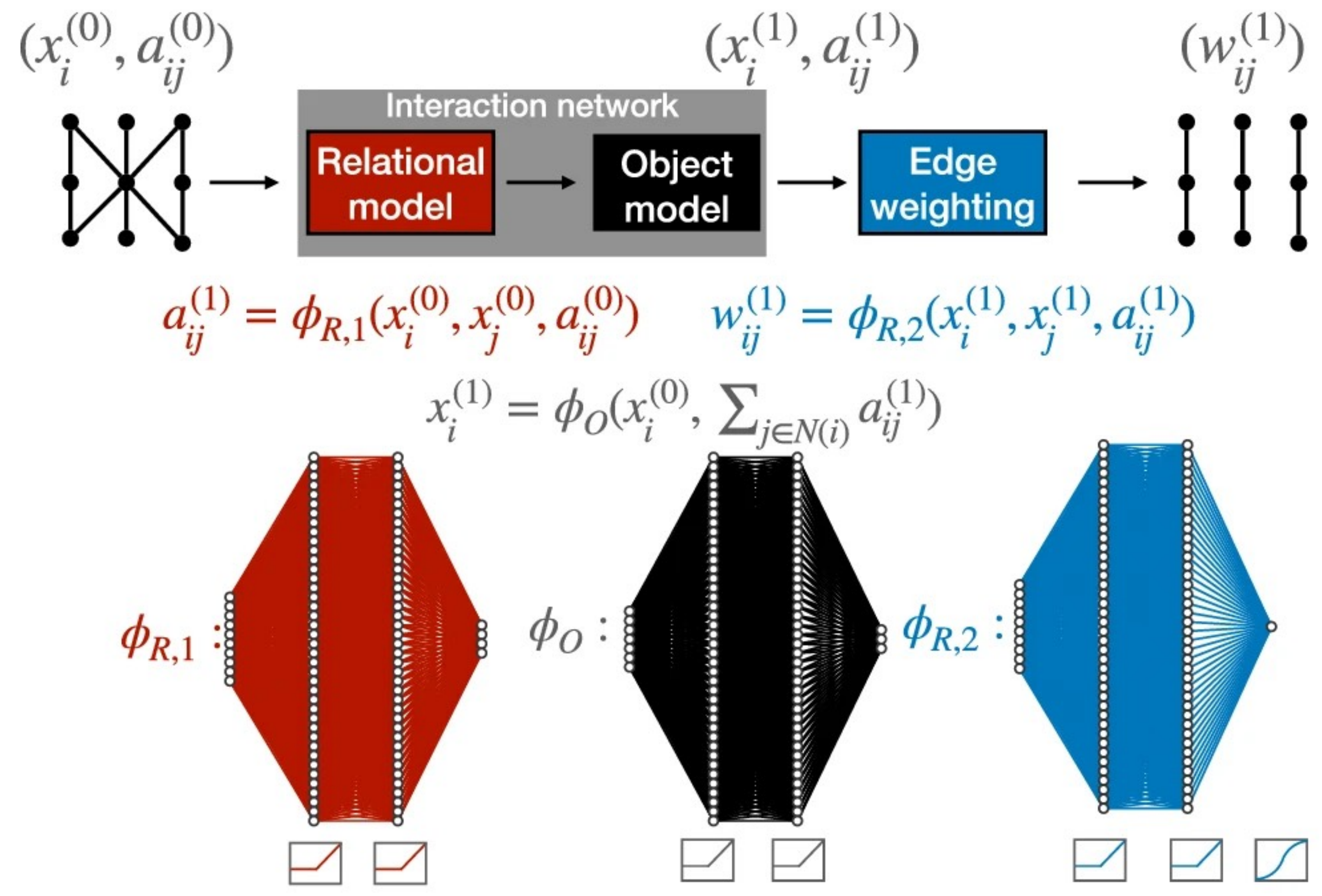
$$\boxed{Rx_1^1 = Rx_1^0 + \sum_i \phi(m_i)(Rx_i - Rx_1)}$$

$$\begin{aligned} ||Rx_i - Rx_j||^2 &= (Rx_i - Rx_j)^T(Rx_i - Rx_j) \\ &= (x_i - x_j)^T R^T R(x_i - x_j) \\ &= ||x_i - x_j||^2 \end{aligned}$$

To enforce equivariance, the exchanged messages must be constructed using invariant quantities!

Step 2.1: The Interaction Network

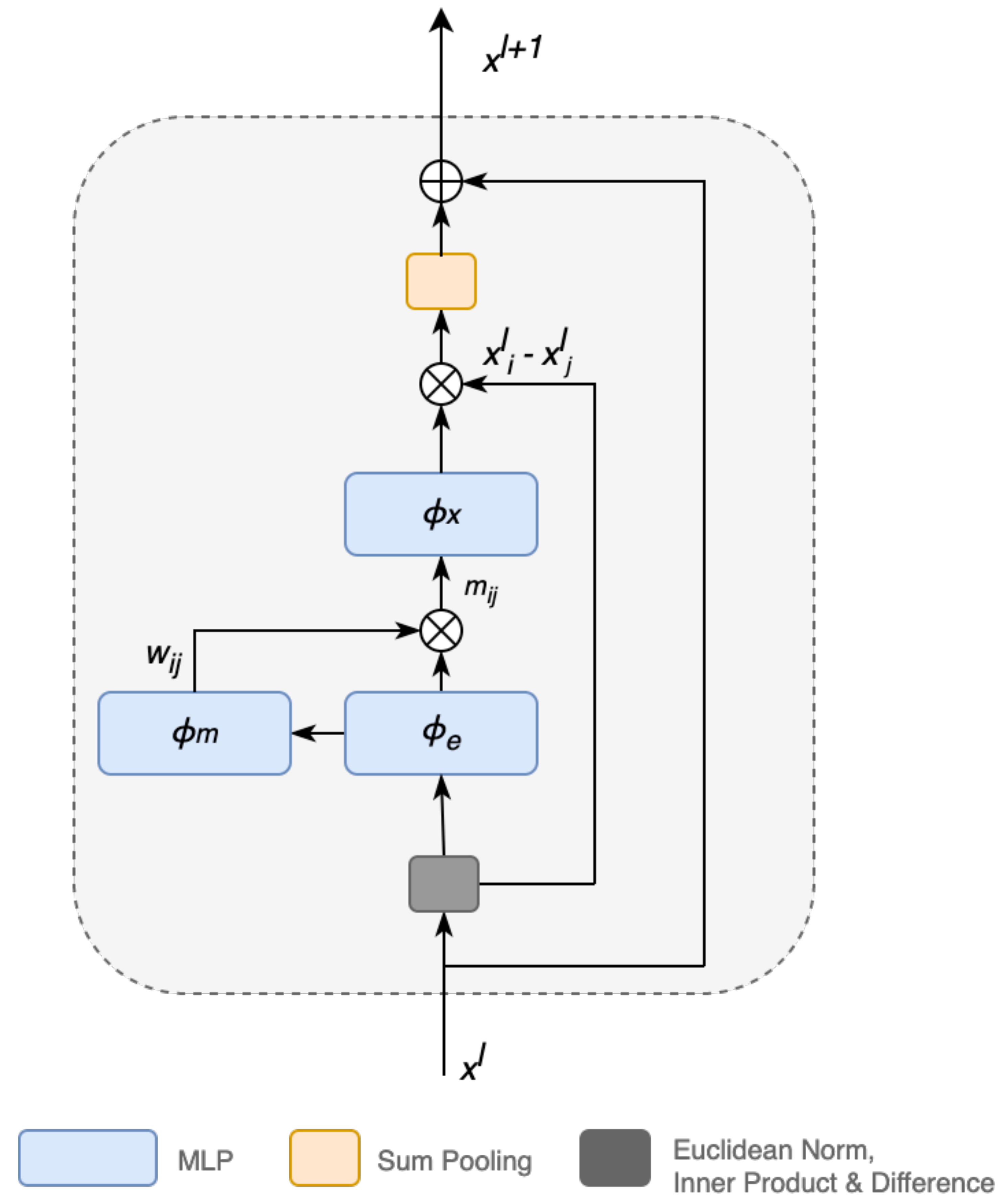
- To compare the performance of our equivariant models, we used the interaction network from [DeZoort et al. (2021)] as our baseline.
- The Interaction Network (IN) is the current state-of-the-art model for the particle tracking problem.
    - Consists of a series of “object” and “relation” reasoning steps
- The IN is an edge-classifying network that produces a probability score for each edge in the hit graph, indicating whether the edge is a track segment (true) or is a frivolous edge (false).



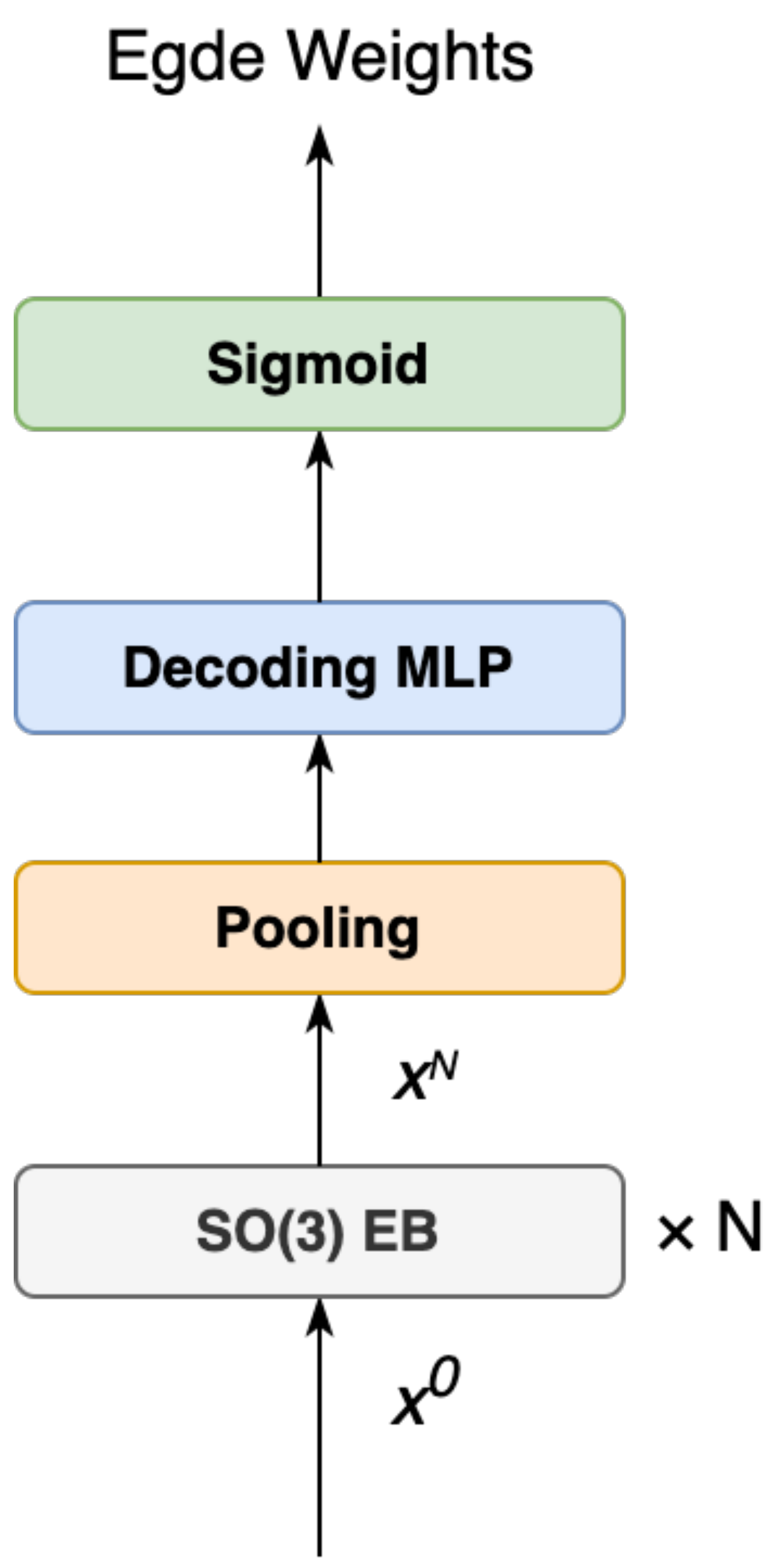
$(x_i^{(0)}, a_{ij}^{(0)})$
$(x_i^{(1)}, a_{ij}^{(1)})$
$(w_{ij}^{(1)})$
Interaction network
Relational model
Object model
Edge weighting
$a_{ij}^{(1)} = \phi_{R,1}(x_i^{(0)}, x_j^{(0)}, a_{ij}^{(0)})$
$w_{ij}^{(1)} = \phi_{R,2}(x_i^{(1)}, x_j^{(1)}, a_{ij}^{(1)})$
$x_i^{(1)} = \phi_O(x_i^{(0)}, \sum_{j\in N(i)} a_{ij}^{(1)})$
$\phi_{R,1}:$
$\phi_O:$
$\phi_{R,2}:$

The complete IN forward pass with the relational and object models approximated as MLPs [DeZoort et al. (2021)]

Step 2.2: The SO(3) EuclidNet



SO(3) Equivariant Block



SO(3) EuclidNet

- The SO(3) EuclidNet takes as input only the node features
- Constructs 2 scalars: inner products between node vectors and euclidean norms.
- These scalars are used to construct the message; since all operations are performed on scalars, the overall network is equivariant.

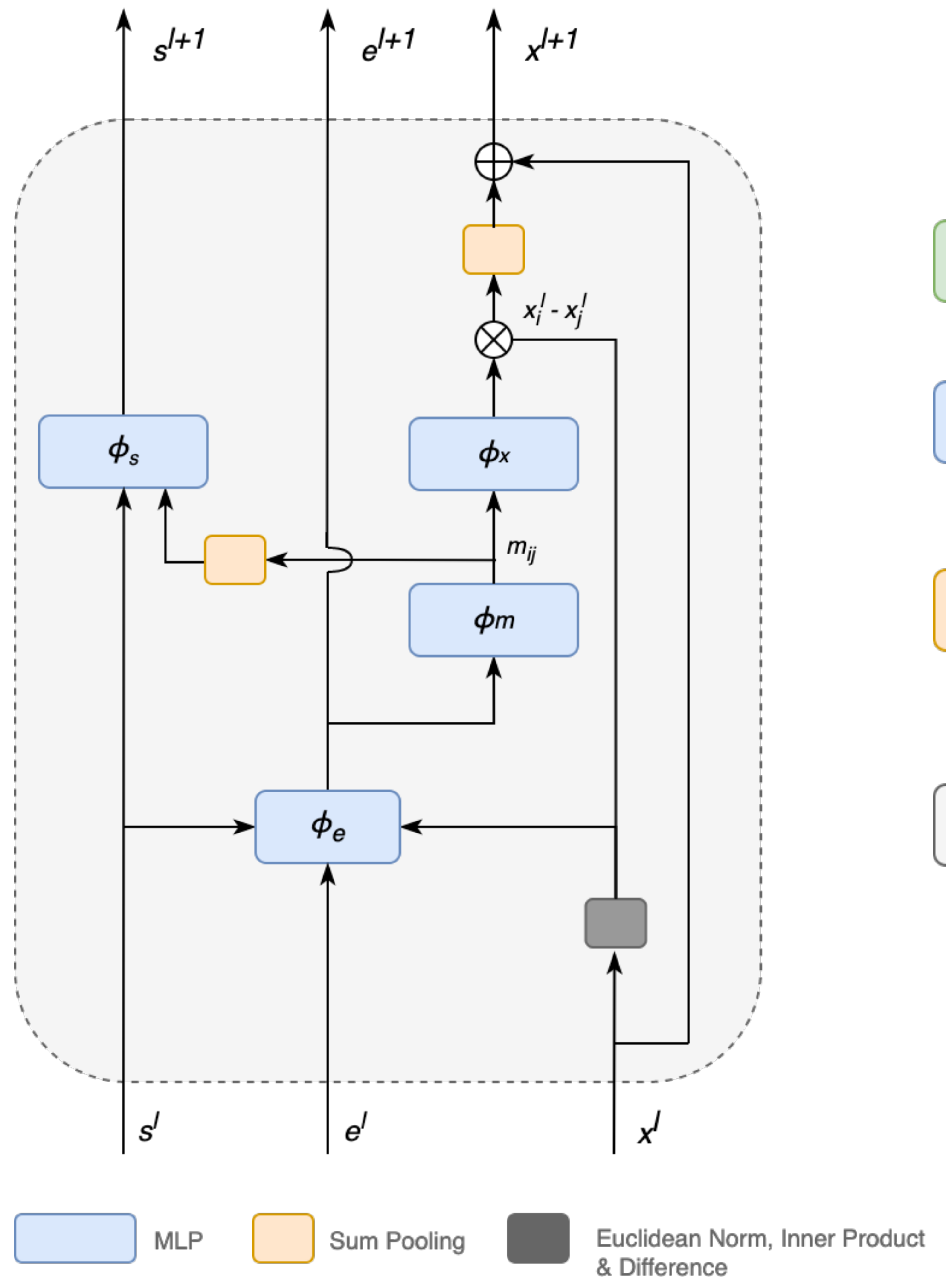
$$m^l_{ij} = \phi_e\left(\psi(\langle x^l_i, x^l_j\rangle),\ \psi(||x^l_i - x^l_j||^2)\right)$$

- $\psi(\cdot) = \mathrm{sgn}(\cdot)\log(|\cdot| + 1)$ normalises large numbers for ease of optimisation.
- The EB block is repeated N times (~3-5), followed by a pooling operation; the result is passed to an MLP which generates a weight/probability score for each edge in the hitgraph.
- **Update operations:**

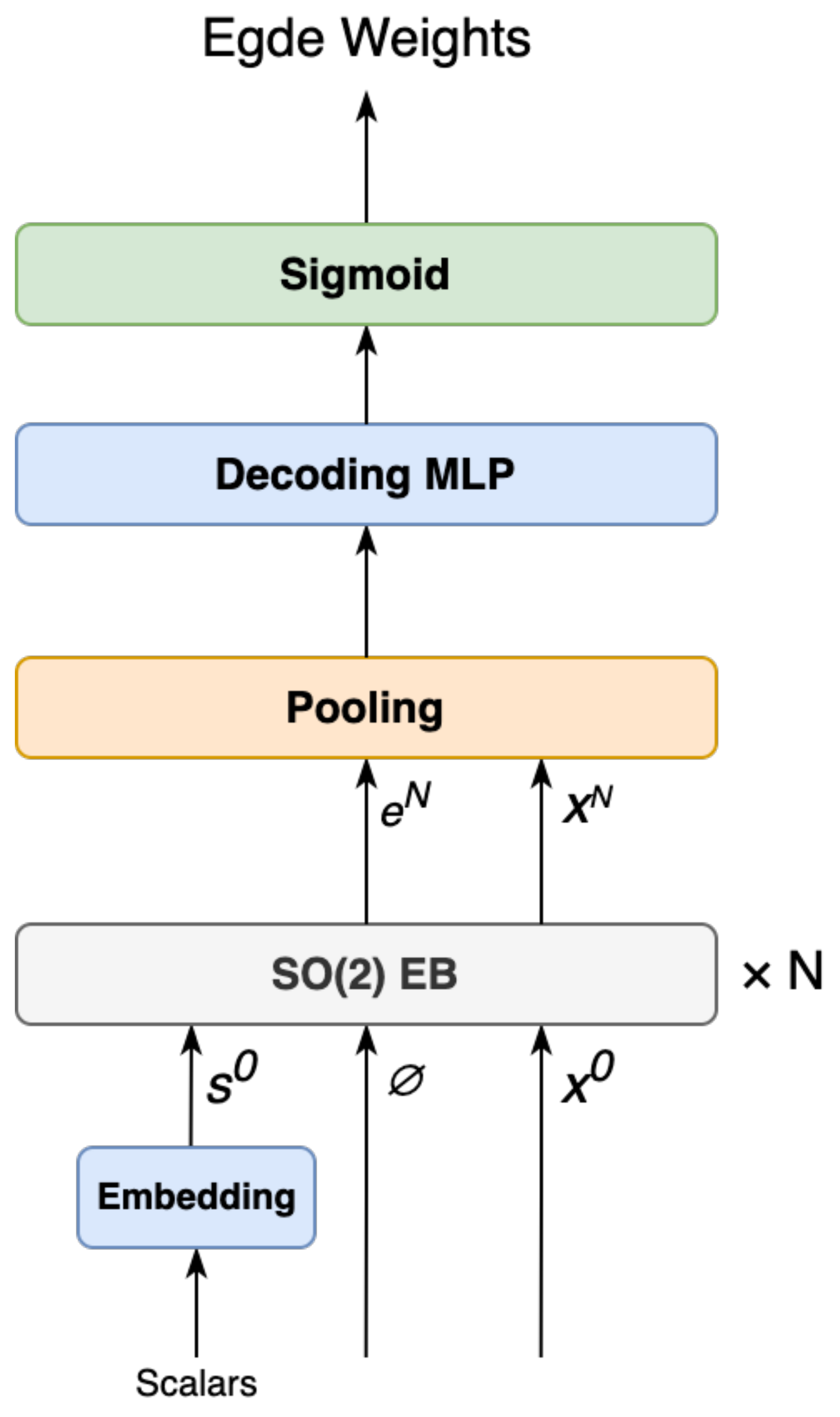
$$x^{l+1}_i = x^l_i + c \sum_{j\in[N]} \phi_x(m^l_{ij}) \cdot (x^l_i - x^l_j)$$

- Depending on which quantities are pooled and passed to the decoding MLP, the output can be **chosen to be equivariant or non-equivariant**.

Step 2.3: The SO(2) EuclidNet



SO(2) Equivariant Block



SO(2) EuclidNet

- The SO(2) EuclidNet takes the node features and the z-coordinate separately as a (scalar) input
- Inner products between node vectors and euclidean norms constructed as before.
  - Additionally, e^l, a dedicated invariant channel is constructed at each iteration.

$$m_{ij}^l = \phi_e\left(\psi(\langle x_i^l, x_j^l\rangle),\ \psi(||x_i^l - x_j^l||^2), s_i^l, s_j^l, e_{ij}^l\right)$$

- We reuse weights across time steps by passing the outputs of one iteration as inputs in the next iteration to the same module.
- **Update operations:**

$$x_i^{l+1} = x_i^l + c\sum_{j\in[N]} \phi_x(m_{ij}^l)\cdot(x_i^l - x_j^l)$$

$$s_i^{l+1} = s_i^l + \phi_s(s_i^l, \sum_{j\in[N]} m_{ij}^l)$$

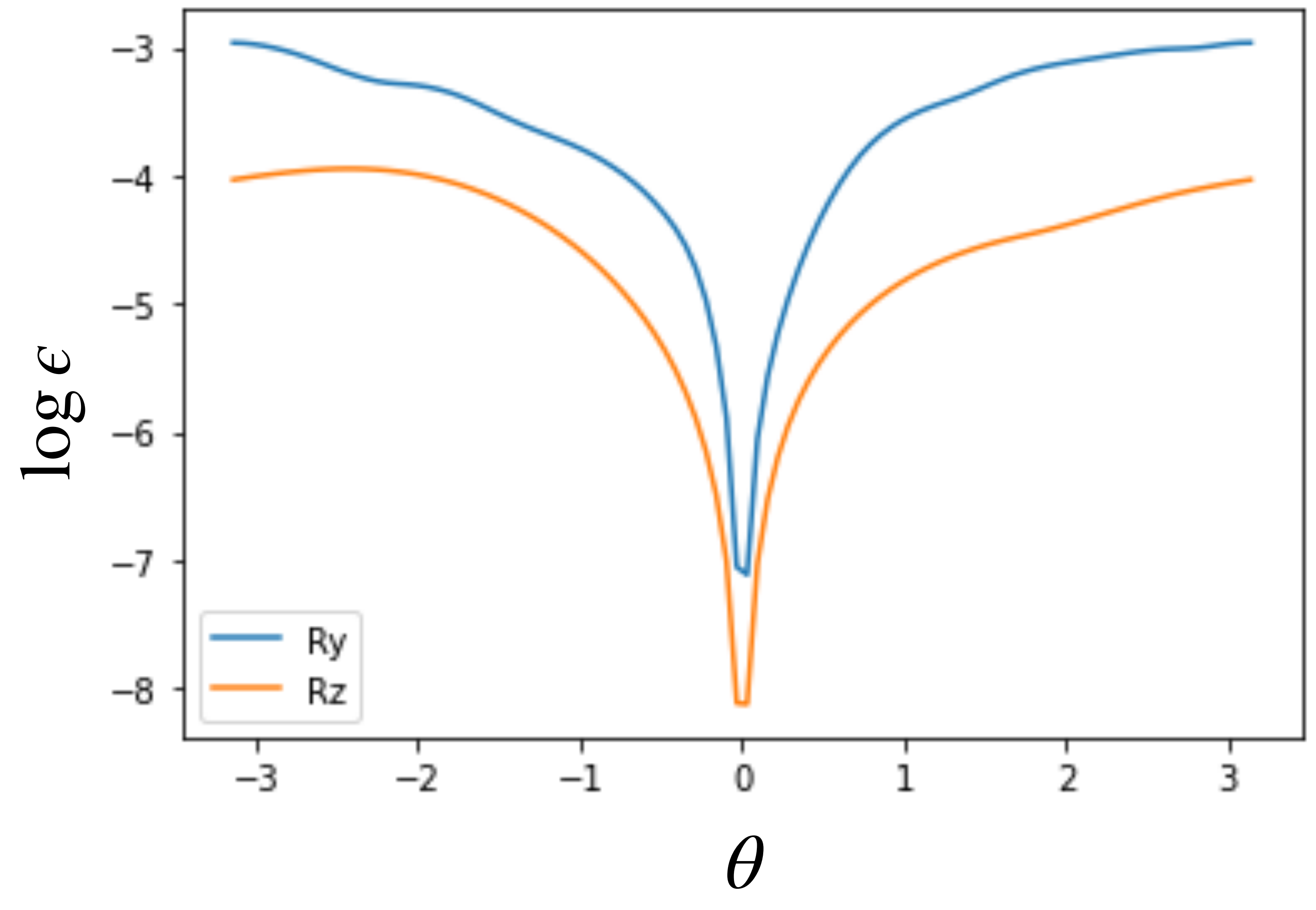
$$e_{ij}^{l+1} = \phi_e(m_{ij}^l)$$

- As in the previous case, the output can be chosen to be equivariant or non-equivariant.

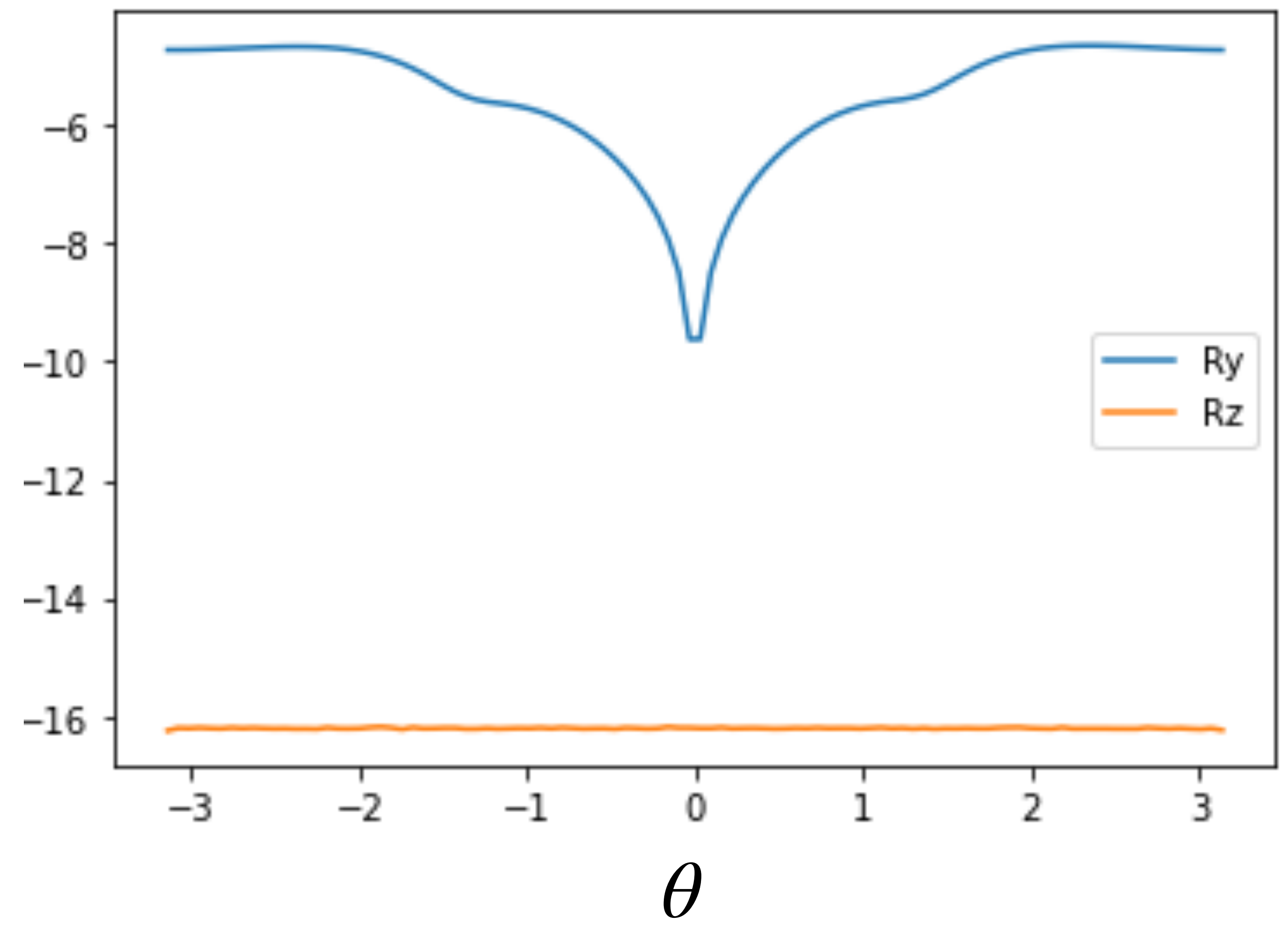
Equivariance Test

- To verify the "baked" symmetry, we rotate the test data by applying a 3D rotation matrix, $R_\theta(\cdot)$ along the z and y axes.
- For a model M, we define an error $\epsilon = |R_\theta(M(x)) - M(R_\theta(x))|$
- Unlike EuclidNet, the Interaction Network is **not robust** to transformations in input space.

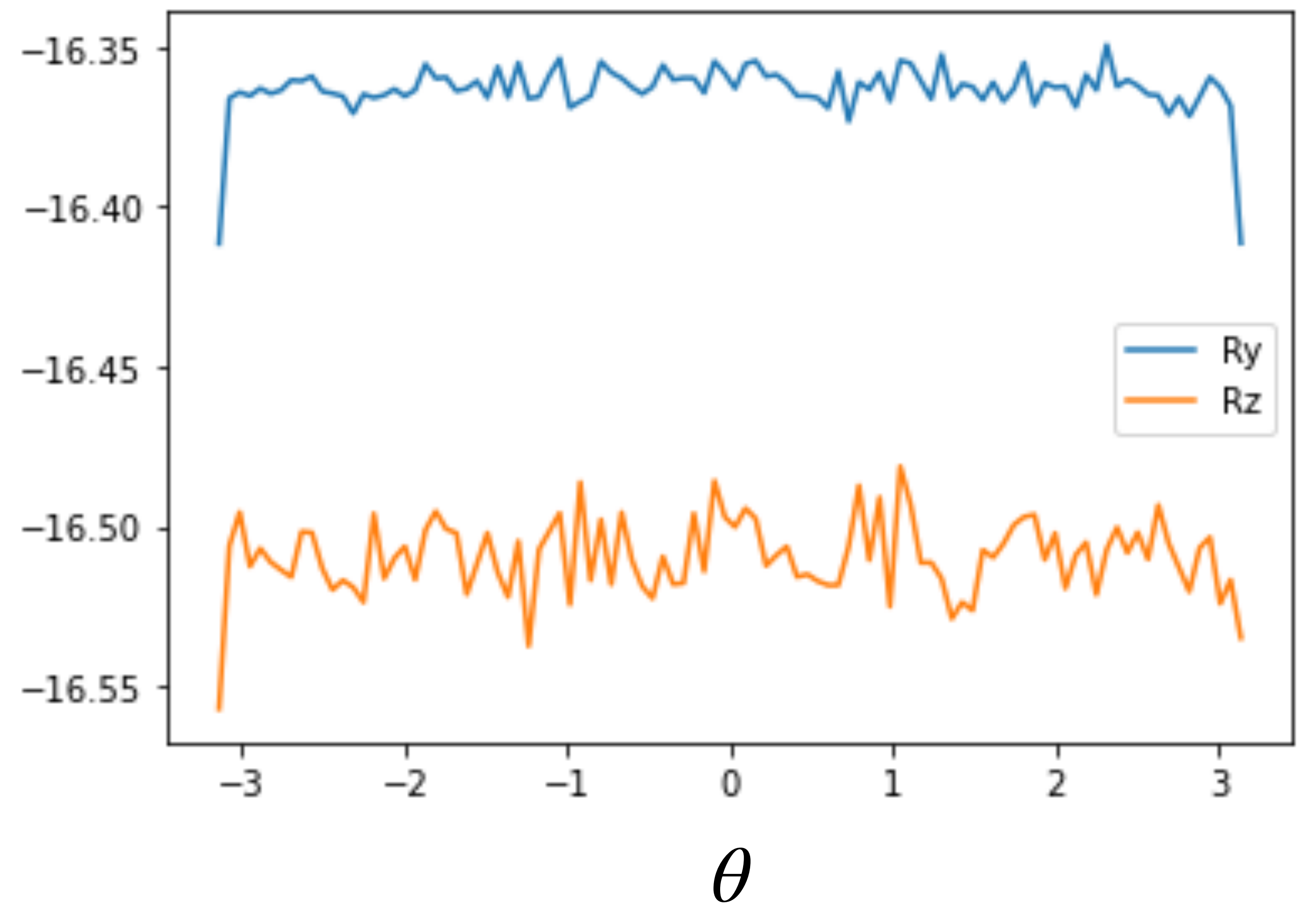
Interaction Network



SO(2) EuclidNet



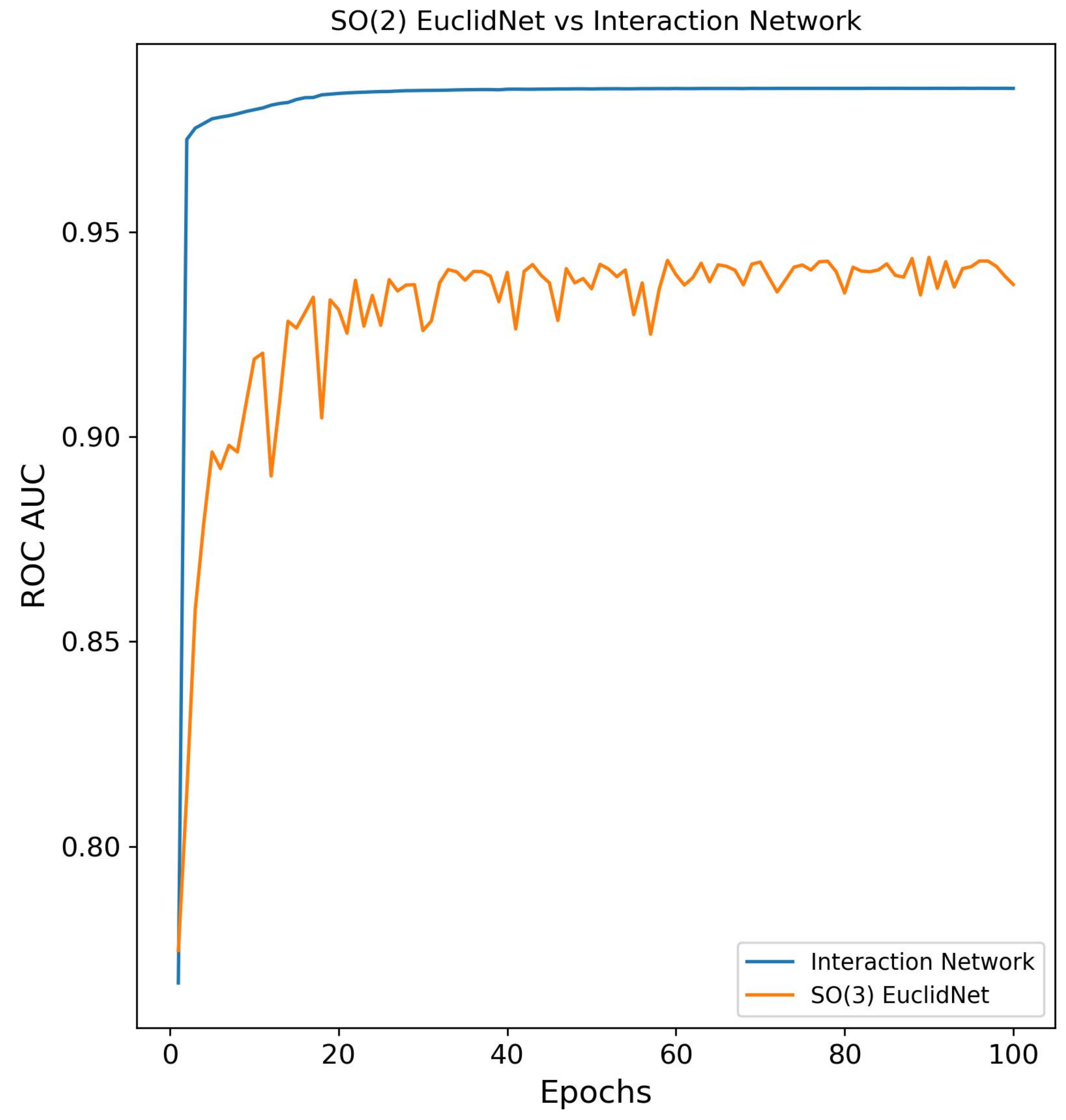
SO(3) EuclidNet

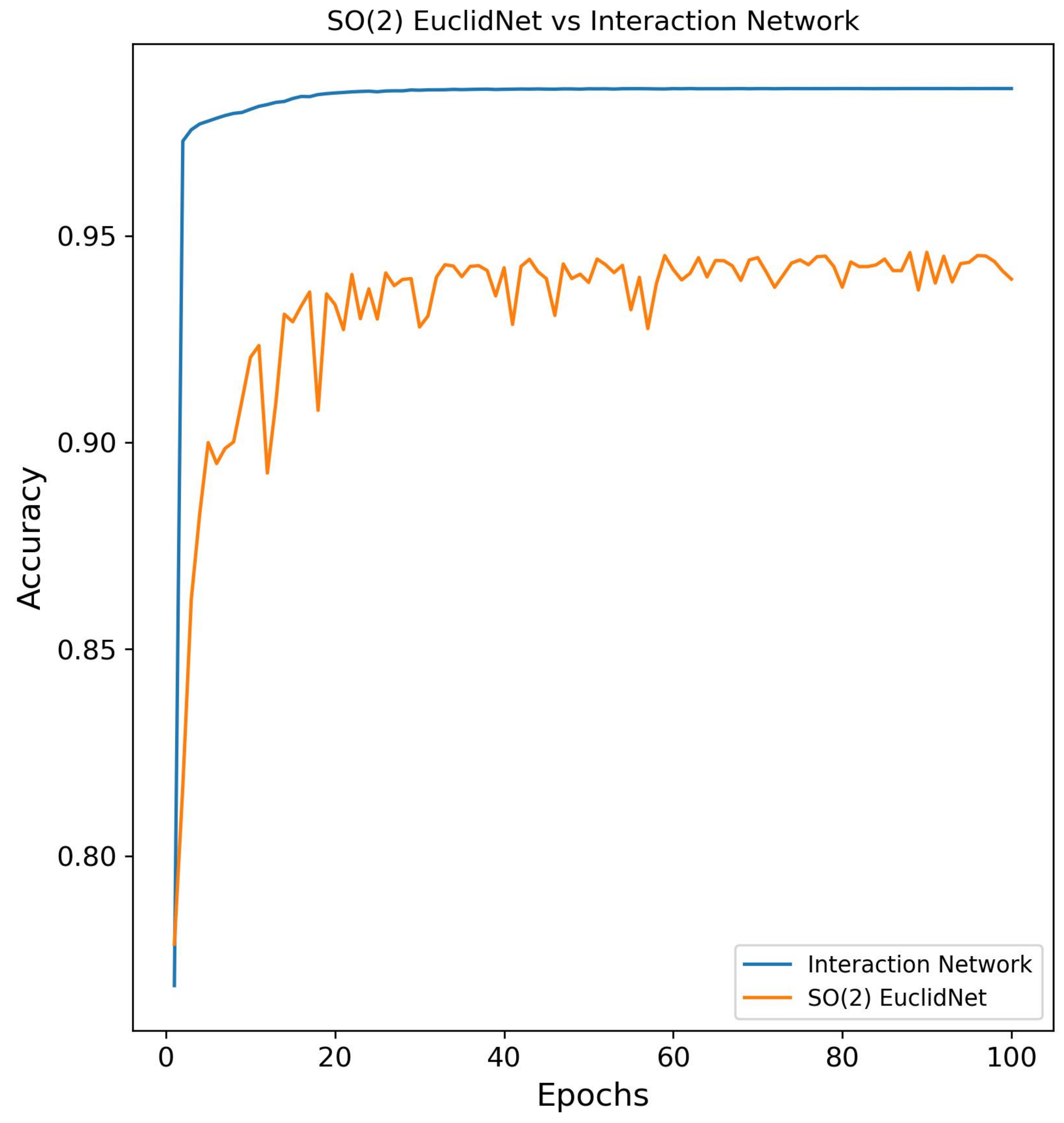


Step 3: Profit???

Not quite.

Results: SO(3) EuclidNet





Training curves for the SO(3) EuclidNet benchmarked against the Interaction Network. Both models were trained for 100 epochs, and had 8 hidden channels.

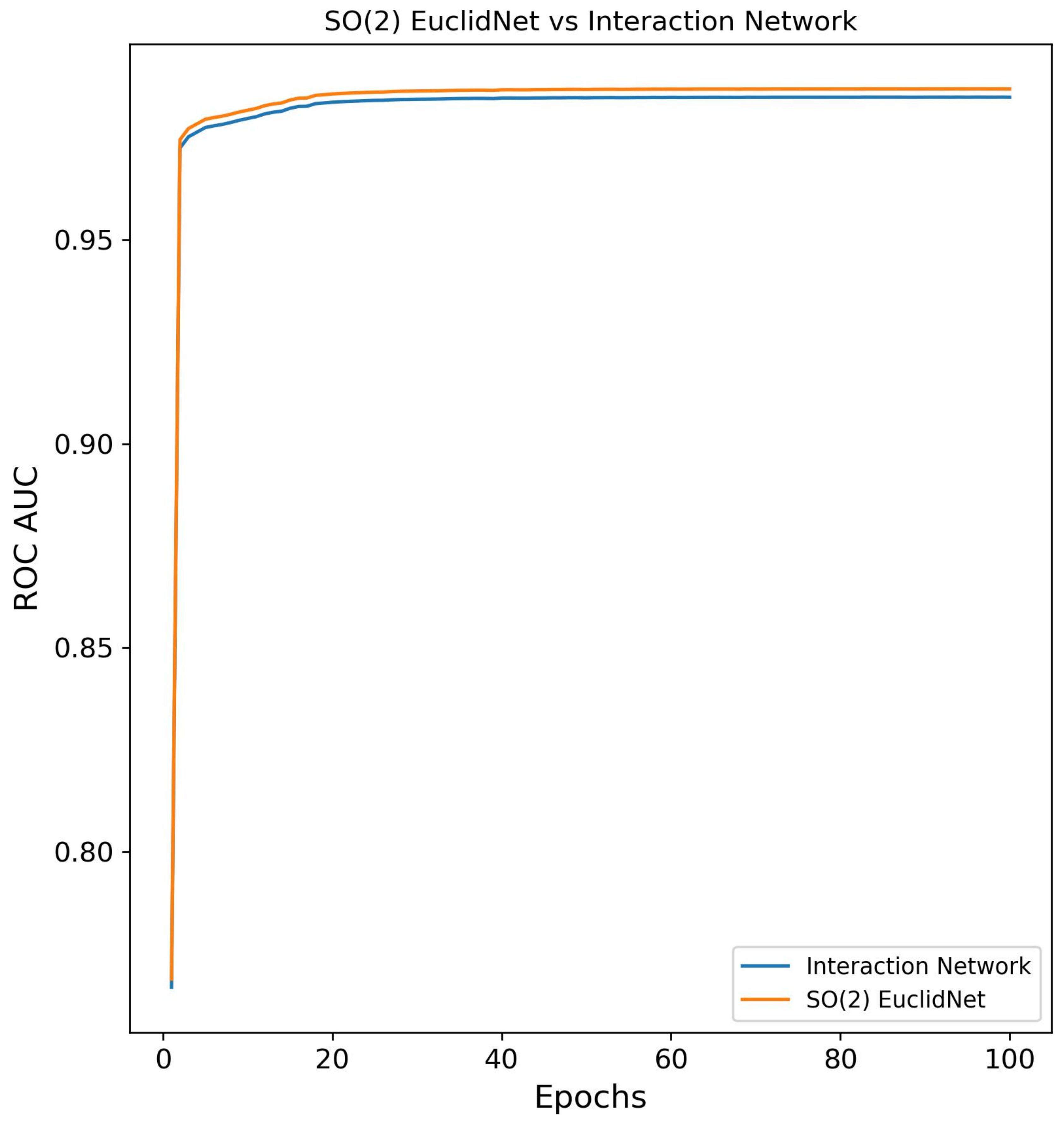
Results: SO(3) EuclidNet

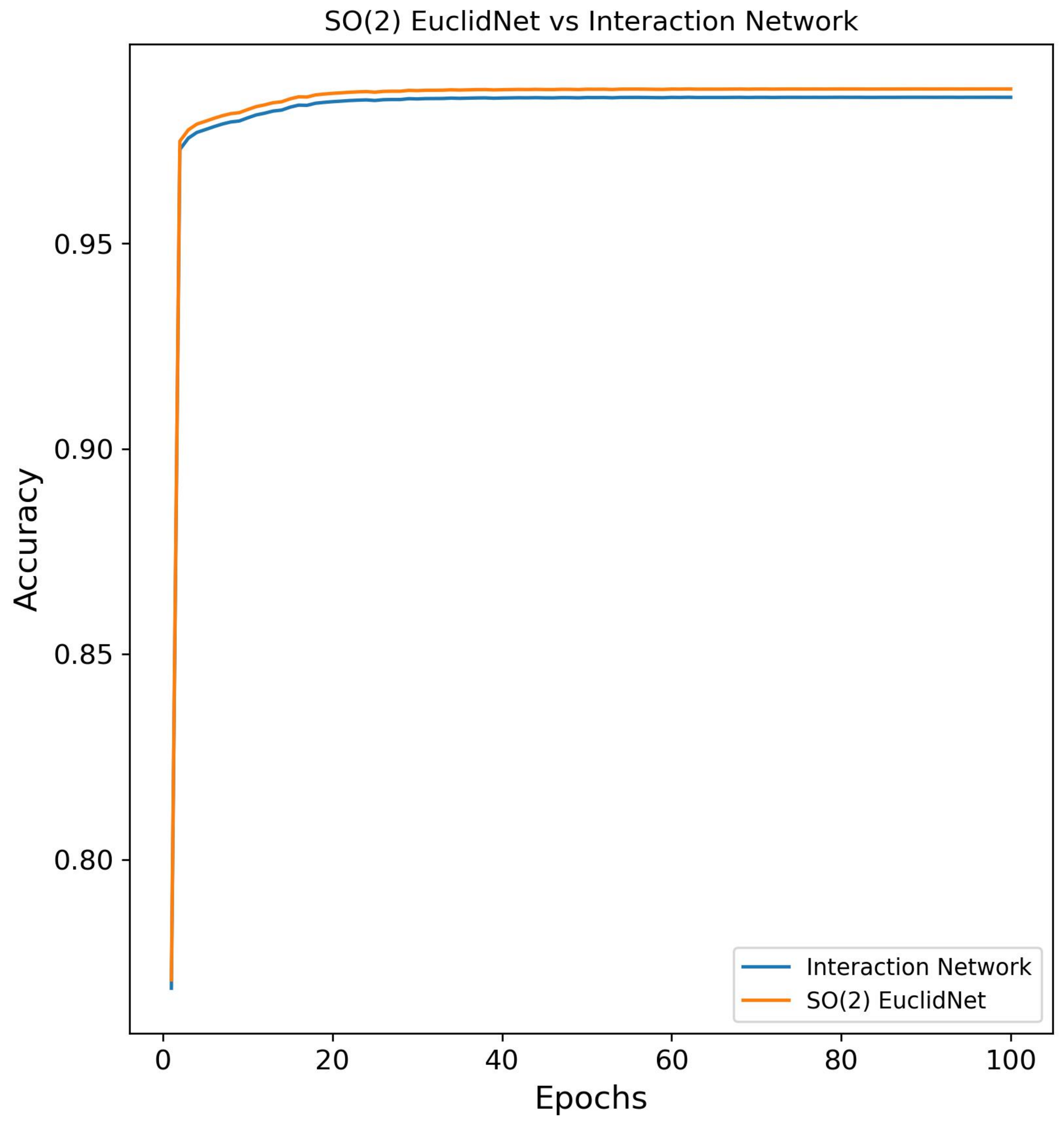
	n_hidden = 8				n_hidden = 16			
Model	**# Parameters**	**AUC**	**Efficiency**	**Purity**	**# Parameters**	**AUC**	**Efficiency**	**Purity**
SO(3) EuclidNet	**780**	0.9439 ± 0.002	0.8684 ± 0.011	**0.9551 ± 0.028**	**2580**	0.9547 ± 0.004	0.8398 ± 0.024	**0.9453 ± 0.041**
Interaction Network	1432	**0.9849 ± 0.006**	**0.9314 ± 0.021**	0.7319 ± 0.052	4392	**0.9932 ± 0.004**	**0.9575 ± 0.019**	0.8168 ± 0.073

SO(3) EuclidNet is unable compete with the baseline — is SO(3) too restrictive?

Perhaps we cannot expect good performance here, SO(2) might be better as it is a slightly more “liberal” symmetry.

Results: SO(2) EuclidNet





Training curves for the SO(3) EuclidNet benchmarked against the Interaction Network. Both models were trained for 100 epochs, and had 8 hidden channels.

Results: SO(2) EuclidNet

	n_hidden = 8				n_hidden = 16			
Model	**# Parameters**	**AUC**	**Efficiency**	**Purity**	**# Parameters**	**AUC**	**Efficiency**	**Purity**
SO(2) EuclidNet	**957**	**0.9913 ± 0.004**	**0.9459 ± 0.022**	**0.7955 ± 0.040**	**1778**	0.9932 ± 0.003	0.9530 ± 0.014	**0.8194 ± 0.033**
Interaction Network	1432	0.9849 ± 0.006	0.9314 ± 0.021	0.7319 ± 0.052	4392	0.9932 ± 0.004	**0.9575 ± 0.019**	0.8168 ± 0.073

Huh. Not as beneficial as we claimed it would be.

SO(2) EuclidNet appears to perform well with a smaller number of hidden channels, however the confidence intervals still overlap.

Are there aspects of the problem that just aren't equivariant? In that case, any symmetric network will always be too restrictive.

Conclusions

- We develop and apply an Euclidean-equivariant GNN to the particle tracking problem.
- Early results indicate that such a formulation offers a marginal improvement over the current state-of-the-art benchmark: the Interaction Network, but the results are still within a standard deviation of each other.
- This result is unlike other reported results from studies of equivariance which demonstrate an unequivocal advantage of equivariant neural networks over standard counterparts.
- To rule out the possibility of either erroneous training or hidden non-equivariant computations in the architecture, we run tests to confirm that our models are robust to symmetry group transformations in the input space.
    - While EuclidNet is stable to such transformations, the benchmark sees a considerable degradation in performance for samples subjected to these transformations.
- More work in the future is needed to concretely establish the possible reasons for this result.
    - We posit that the tracking problem possibly contains non-equivariant aspects which cannot be learnt by a constrained model.
    - Presently, we have no known method to measure the quality of the learnt equivariance. Recent work [Mosklev et al. (2022)] has the potential to evaluate the "degree" to which a network is equivariant.
    - [Finzi et al. (2021)] suggests that most real-world problems benefit more from implementing "soft equivariance", where the symmetry constraints are somewhat relaxed. Future studies could study these implementations which allow the model to learn non-equivariant features in the problem.

Deliverables

- D. Murnane, S. Thais, and A. Thete. Equivariant Graph Neural Networks for Charged Particle Tracking. 21st International Workshop on Advanced Computing Analysis Techniques in Physics Research (ACAT). Poster (upcoming). Oct. 2022. url: https://indi.to/Gh2Fs
- **GitHub repository:** https://github.com/ameya1101/equivariant-tracking

Acknowledgements

This work was supported by IRIS-HEP through the U.S. National Science Foundation (NSF) under Cooperative Agreement OAC-1836650 and by Department of Energy grant DE-SC0007968. 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 using 2022 NERSC award ERCAP-0021226.

Thank you!

And a special thanks to my mentors, Daniel Murnane and Savannah Thais!