# Graph Attention Interaction Networks and Particle Tracking at LDMX

**Majd Ghrear** **Elizabeth Berzin** **Lauren Tompkins**
Department of Physics, Stanford University
Stanford, CA 94305
{mghrear,eberzin,laurenat}@stanford.edu

**Matt Graham**
SLAC National Accelerator Laboratory
Menlo Park, CA 94025
mgraham@slac.stanford.edu

## Abstract

We present a dedicated investigation of graph neural network (GNN)-based particle tracking in small telescope-style detectors, using LDMX as a case study. This detector topology simplifies pipeline components (e.g., graph construction), providing a clean arena to isolate and evaluate GNN performance. We compare Interaction Networks (IN) and a new attention-augmented variant, Graph Attention Interaction Networks (GAIN), against a Combinatorial Kalman Filter (CKF) baseline.We find that GNNs improve tracking efficiencies over CKF, while CKF retains the lowest fake rates. In addition, GAIN consistently outperforms IN across all metrics.

## 1 Introduction

The Light Dark Matter eXperiment (LDMX) [4] is a planned fixed-target electron-beam experiment searching for sub-GeV thermal relic dark matter (DM) via a "missing-momentum" measurement: the incoming electron is measured upstream of the target and the outgoing (recoiling) electron is measured downstream, with DM production inferred from an unexplained loss of momentum. This relies on the tagging and recoil trackers, which measure the electron before and after the target, respectively. Both are telescope detectors, a series of thin silicon strip tracking planes along the beam direction. Electrons traversing the tracking planes deposit energy through ionization. These signals are clustered into space-points that trace the electron's trajectory, from which its momentum is inferred. Fig. 1 illustrates this setup alongside a simulation of multiple electron tracks in the tagging tracker. Identifying space-points belonging to the same track is a key challenge in particle tracking.

Traditional particle tracking algorithms are based on the Combinatorial Kalman Filter (CKF) [20, 12, 3]. CKF-based methods perform robustly across many applications, including complex detector topologies at the Large Hadron Collider (LHC) [11, 1]. More recently, Graph Neural Network (GNN)–based tracking has attracted interest [14, 13, 17, 6, 9, 19, 10]. GNNs operate on graph-structured data, iteratively passing and aggregating information along edges. They are well suited to tasks where relationships matter, such as linking hits into tracks. Here, GNNs provide a relational inductive bias by hard-coding assumptions aligned with detector geometry and track formation.

This work studies GNN-based tracking in LDMX. Prior work has focused on LHC experiments with more complex geometries, higher pileup, and more complicated events, where GNNs have not consistently outperformed CKF. Our telescope-style detectors differ sufficiently to warrant dedicated study and provide an opportunity to better isolate and evaluate GNN performance, since other

Conference on Physics and AI at Stanford University (PAI 2026).

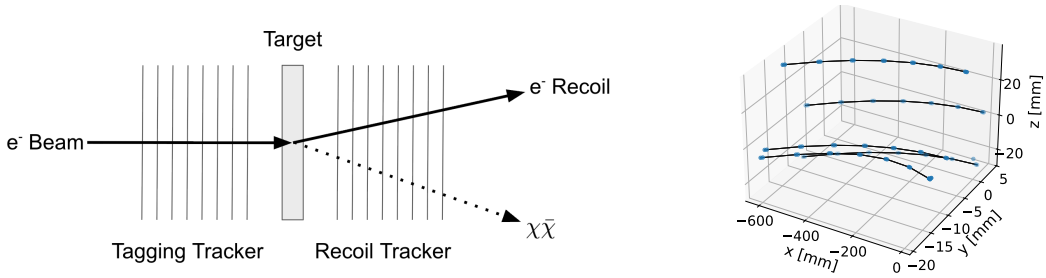


Figure 1: Left: LDMX tagging tracker, target, and recoil tracker. The missing momentum strategy seeks to infer DM ($\chi\bar{\chi}$) production via unexplained missing momentum in the recoiling electron. Right: Five simulated electron tracks in the tagging tracker; ionization deposits are clustered into space-points, with edges connecting space-points from the same parent particle.

steps (e.g., graph construction) are substantially simplified. In addition, we propose an update to the Interaction Network (IN) [5], a physics-motivated GNN architecture that has been widely adopted in GNN-based particle tracking. The IN is updated by incorporating multi-head self-attention ideas from Graph Attention Networks (GAT) [21]; the result is denoted as the Graph Attention Interaction Network (GAIN). GAINs are similar to INs but can focus on relevant interactions while down-weighting irrelevant ones, and multi-headed attention enables larger models, increasing the network's expressive power. Early results are promising, and because INs are broadly applicable, these developments may prove useful beyond particle physics.

## 2 Models

### 2.1 Interaction networks

INs were originally developed to reason about objects and interactions in physical systems [5]. Their inputs are node attributes $V = \{\mathbf{v}_i\}_{i=1}^{N_v}, \mathbf{v}_i \in \mathbb{R}^{F^v}$, external effect vectors $X = \{\mathbf{x}_i\}_{i=1}^{N_v}, \mathbf{x}_i \in \mathbb{R}^{F^x}$, and edges $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1}^{N_e}$, where $\mathbf{e}_k \in \mathbb{R}^{F^e}$ are edge attributes and $r_k$ ($s_k$) denote receiver (sender) indices. For each edge ($k$), an interaction term is formed by concatenating receiver, sender, and edge attributes: $\mathbf{b}_k = \mathbf{v}_{r_k} \| \mathbf{v}_{s_k} \| \mathbf{e}_k$. A common relational model $f_r : \mathbb{R}^{2F^v+F^e} \to \mathbb{R}^{F^{e'}}$ then computes an effect vector $\mathbf{c}_k = f_r(\mathbf{b}_k)$. For each node ($i$), incoming effect vectors are summed to form the aggregated effect $\bar{\mathbf{c}}_i$. The node feature vector ($\mathbf{v}_i$), external effect vector ($\mathbf{x}_i$), and aggregated effect vector ($\bar{\mathbf{c}}_i$) are concatenated into an object vector $\mathbf{o}_i = \mathbf{v}_i \| \mathbf{x}_i \| \bar{\mathbf{c}}_i$. Lastly, a shared object model $f_o : \mathbb{R}^{F^v+F^x+F^{e'}} \to \mathbb{R}^{F^{v'}}$ predicts how the interactions influence each object. Both $f_r$ and $f_o$ are multilayer perceptrons (MLPs) responsible for relational and object reasoning, respectively.

Adapting INs to particle tracking, $V$ is the event's space-points and $E$ is the edges between them. Although $X$ is often omitted, it can encode context such as magnetic-field vectors associated with $V$. Here, the relational and object-reasoning steps correspond to updating edge and node embeddings, respectively. IN layers can be stacked, and track candidates are typically constructed by classifying the resulting edge embeddings or by clustering the resulting node embeddings

### 2.2 Graph attention interaction network layer

This section introduces the building-block layer for Graph Attention Interaction Networks. The idea is to replace the IN's relational reasoning and effect aggregation steps with a multi-headed self-attention mechanism, similar to [21]. This injects graph structure into the attention mechanism, it has access to full relational information (sender, receiver, and edge feature vectors) and influences both edge and node embedding updates. We describe a GAIN layer with $H$ heads below.

The layer takes the same inputs as INs and uses the same initial concatenation to form interaction terms, $\mathbf{b}_k = \mathbf{v}_{r_k} \| \mathbf{v}_{s_k} \| \mathbf{e}_k$. For each head $h \in \{1, \ldots, H\}$, an independent relational model $f_r^h : \mathbb{R}^{2F^v+F^e} \to \mathbb{R}^{F^{e'}}$ computes a raw effect vector $\mathbf{c}_k^h = f_r^h(\mathbf{b}_k)$, and a shared attention function $a^h : \mathbb{R}^{F^{e'}} \to \mathbb{R}$ assigns an unnormalized attention score $e_k^h = a^h(\mathbf{c}_k^h) = \mathrm{LeakyReLU}(\mathbf{a}^h \cdot \mathbf{c}_k^h)$,

with learnable $\mathbf{a}^h \in \mathbb{R}^{F^{e'}}$. Attention scores are normalized across the incoming neighborhood of the receiver. let $\mathcal{N}_i$ denote the the set of edge indices such that node $i$ is the receiver, $\mathcal{N}_i = \{k \in \{1, ..., N_e\} | r_k = i\}$. The normalized attention scores are then $\alpha_k^h = \text{softmax}(e_k^h) = \frac{\exp(e_k^h)}{\sum_{j \in \mathcal{N}_i} \exp(e_j^h)}$, where $i$ is the index of the receiver to edge $k$. The weighted effect vectors, $\alpha_k^h \mathbf{c}_k^h$, are concatenated over all heads to produce the updated edge feature vectors, $E' = \{(\|_{h=1}^H \alpha_k^h \cdot \mathbf{c}_k^h, r_k, s_k)\}_{k=1}^{N_e}$.

For each node ($i$) and head ($h$), weighted incoming effects are summed to form $\bar{\mathbf{c}}_i^h = \sum_{j \in \mathcal{N}_i} \alpha_j^h \cdot \mathbf{c}_j^h$. The aggregated effect vectors are the concatenated over all heads, $\bar{\mathbf{c}}_i = \|_{h=1}^H \bar{\mathbf{c}}_i^h$. An object vector is then obtained by concatenating the node-level vectors $\mathbf{o}_i = \mathbf{v}_i \| \mathbf{x}_i \| \bar{\mathbf{c}}_i$. Finally, an object model $f_o : \mathbb{R}^{F^v + HF^{e'} + F^x} \to \mathbb{R}^{F^{v'}}$, computes $\mathbf{v}_i' = f_o(\mathbf{o}_i)$. The updated nodes are $V' = \{\mathbf{v}_i'\}_{i=1}^{N_v} = \{f_o(\mathbf{v}_i \| \mathbf{x}_i \| \bar{\mathbf{c}}_i)\}_{i=1}^{N_v}$. A PyTorch Geometric [15] implementation of GAIN is available here: `https://github.com/mghrear/GAIN`.

Prior work incorporates continuous edge features into GAT-style propagation by using them to parameterize pairwise aggregation weights that are adapted across layers [16]. EGAT [22] conditions attention on edge information and updates edge representations via line-graph-style constructions. RGAT [8] targets typed graphs, applying attention within or across relation types to compute relation-aware message weights. GAIN instead builds on INs: we use relational models to compute interaction embeddings from each sender–receiver–edge tuple, apply multi-head attention over incoming interactions to weight their contributions, and update node states with a shared object model. This naturally supports continuous edge features and helps down-weight spurious candidate edges.

# 3 Application to LDMX tracking

## 3.1 Simulation and datasets

We use simulated LDMX events generated with the public `LDMX-SW` framework[4], largely based on `GEANT4`[2]. All simulations use an $8.0$ GeV incident electron beam, and we consider two scenarios:

**Scenario 1** we generate "signal" events where dark matter is produced in the target via a mediator ($A'$). This case is challenging because the $A'$ takes most of the electron's momentum [7], leaving a highly deflected recoil that produces less hits. The task is to reconstruct the recoil electron track in the recoil tracker. The training set includes samples generated at several mediator masses, $m_{A'} = 1.0, 0.1, 0.01, 0.001$ GeV, with 159203, 39727, 99297, and 38069 simulated events, respectively. We reserve 20% of the training sample for validation. For testing, we focus on the most difficult case, $m_{A'} = 1.0$ GeV and use a test set with 372452 simulated events.

**Scenario 2** we generate an inclusive sample with 1–5 beam electrons per event (30,000 events per multiplicity), split into 70% training, 15% validation, and 15% testing. The task is to determine the electron multiplicity by counting reconstructed tracks in the Tagging tracker.

## 3.2 Pipeline and architecture

We use a typical three-stage GNN-based tracking pipeline. (1) Graph construction: For each event, space-points are vertices with features given by their Cartesian coordinates. Next, candidate edges are selected. For LHC applications, this is non-trivial: fully connected graphs are infeasible, so an edge-selection algorithm is required, and its quality strongly impacts downstream performance. Here, lower occupancy and simpler geometry simplify graph construction. We demonstrate two approaches: fully connected graphs for scenario 1 and in scenario 2 we connect space-points in neighboring tracking planes. Edge features are displacement vectors from sender to receiver. (2) Edge labeling: A GNN performs binary classification on edges, where label 1 denotes a true connection (same track) and label 0 denotes otherwise. (3) Track segmentation: we keep edges with score $\geq 0.5$ for scenario 1 and $\geq 0.9$ for scenario 2, then tracks are formed as connected components in the pruned graph.

To compare IN and GAIN fairly, we match architectures such that the edge/node feature expansions are equivalent and the parameter counts are similar. External effects ($X$) are omitted, and the relational/object models are 3-layer ReLU MLPs. Scenario 1: Both models use 3 message-passing layers with node/edge feature expansion $3 \to 12 \to 24 \to 24$. GAIN uses 3 heads and hidden width 40 (47,741 params); IN uses hidden width 58 (48,203 params). Scenario 2: Both use larger

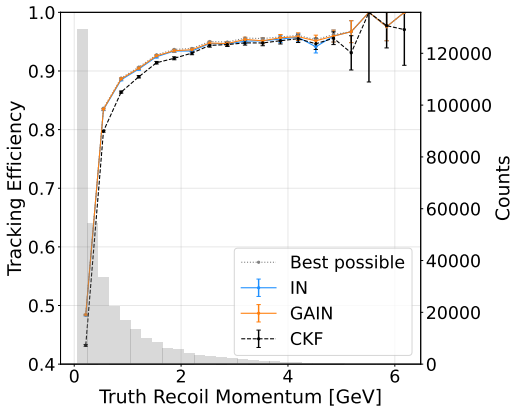


Figure 2: Tracking efficiency versus truth recoil momentum for GAIN, IN, and CKF. The dotted line denotes the best achievable performance. The gray histogram (right axis) shows the event distribution.

Table 1: Performance metrics (mean $\pm$ std.) for GAIN and IN, evaluated over 5 model initializations.

| Model | BCE Loss ($\times 10^{-2}$) | Brier ($\times 10^{-3}$) | ECE ($\times 10^{-3}$) | AUC-ROC |
|---|---|---|---|---|
| GAIN | $4.56 \pm 0.82$ | $13.2 \pm 0.25$ | $1.67 \pm 0.25$ | $0.99831 \pm 0.00007$ |
| IN | $6.01 \pm 0.42$ | $17.4 \pm 1.35$ | $2.80 \pm 0.71$ | $0.99714 \pm 0.00042$ |

models with expansion $3 \to 64 \to 128 \to 128$. GAIN uses 8 heads and hidden width 50 (384,761 params); IN uses hidden width 128 (389,121 params). In all cases, a final relational model maps concatenated source, target, and edge embeddings to a sigmoid edge score. Full implementation details are available here: `https://github.com/mghrear/GAIN`.

### 3.3 Results

Both IN and GAIN are optimized with `Adam` [18] at lr$= 5 \times 10^{-3}$ for 200 epochs, using a StepLR scheduler with $\gamma = 0.9$ every 5 epochs. We save the model with the lowest validation BCE loss. To match prior LDMX tracking studies, reconstructed tracks in the recoil tracker must have $\geq 7$ hits. For the tagging tracker this hit requirement is reduced to $\geq 6$. A reconstructed track is matched to a truth track if they share at least $50\%$ of hits in both directions (reco-in-truth and truth-in-reco); otherwise it is classified as fake. The tracking efficiency is the number of matched tracks divided by the number of simulated tracks. All results use the test sets from Sect. 3.1.

**Scenario 1** We evaluate tracking efficiency versus recoil momentum for CKF, IN, and GAIN (Fig. 2). Both GNNs outperform CKF, with efficiency approaching the best possible performance set by the fraction of truth tracks passing the 7-hit requirement. Overall fake rates are $1.5 \times 10^{-4}$ (CKF), $4.11 \times 10^{-3}$ (GAIN), and $4.35 \times 10^{-3}$ (IN). To compare IN and GAIN as edge classifiers, Table 1 reports the mean$\pm$std of their binary cross entropy (BCE) loss, brier score, expected calibration error (ECE), and AUC-ROC. GAIN improves upon IN across all metrics.

**Scenario 2** GAIN (IN) achieves BCE 0.0155 (0.0273) with AUC–ROC 0.9998 (0.9994). Tracking gains are strongest at high multiplicities: in 5-electron events, the GAIN (IN) reconstructed track count is $90.9\%$ ($77.4\%$) the truth count, with $1.77\%$ ($6.66\%$) fake rate. Reducing (increasing) GAIN's heads by one gives BCE $0.0183$ ($0.0137$); an analogous IN change to node/edge feature expansion at matched parameter count yields $0.0384$ ($0.0238$). Adding two message-passing layers by repeating the final layer yields a 5-layer GAIN with BCE $0.0119$, while the matched IN variant attains $0.0351$. Under this depth-extension scheme and at comparable parameter counts, IN is more sensitive to increased depth, while GAIN remains stable and benefits from additional layers. Fig. 3 shows confusion matrices for the 5-layer GAIN model and CKF, alongside their fake rates. With increasing pileup, GAIN exceeds CKF in efficiency, while CKF has the lowest fake rates.

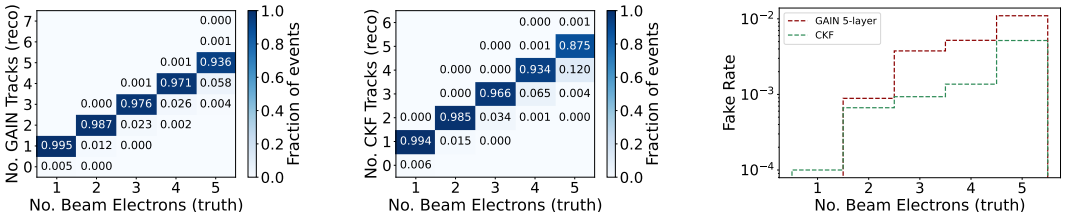


Figure 3: Left and Center: Confusion matrix showing number of beam electrons versus reconstructed tracks for GAIN and CKF. Right: Fake rate versus number of beam electrons for GAIN, IN, and CKF.

## 4 Conclusion

These early results are promising for applying GNN-based particle tracking to telescope-style detectors. Across the scenarios studied, GNNs improve tracking efficiency, while CKF maintains the lowest fake rate. GAIN delivers consistent gains over IN across multiple metrics. This work demonstrates that GNNs can be competitive with CKF and that GAIN can outperform IN. The results also suggest further gains are possible through architectural optimization, as well as improved graph construction and track segmentation, which we leave for future work.

## Acknowledgments and Disclosure of Funding

Contributions from Stanford are supported by the US Department of Energy under grants DE-SC0022083. Elizabeth Berzin is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2146755, and by the Stanford Graduate Fellowship.

## References

[1] M. Aaboud et al. Performance of the ATLAS Track Reconstruction Algorithms in Dense Environments in LHC Run 2. *Eur. Phys. J. C*, 77(10):673, 2017. doi: 10.1140/epjc/s10052-017-5225-7.

[2] S. Agostinelli et al. GEANT4 - A Simulation Toolkit. *Nucl. Instrum. Meth. A*, 506:250–303, 2003. doi: 10.1016/S0168-9002(03)01368-8.

[3] Xiaocong Ai et al. A Common Tracking Software Project. *Comput. Softw. Big Sci.*, 6(1):8, 2022. doi: 10.1007/s41781-021-00078-8.

[4] Torsten Akesson et al. LDMX - The Light Dark Matter eXperiment, 8 2025.

[5] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[6] Catherine Biscarat, Sylvain Caillou, Charline Rougier, Jan Stark, and Jad Zahreddine. Towards a realistic track reconstruction algorithm based on graph neural networks for the HL-LHC. *EPJ Web Conf.*, 251:03047, 2021. doi: 10.1051/epjconf/202125103047.

[7] James D. Bjorken, Rouven Essig, Philip Schuster, and Natalia Toro. New Fixed-Target Experiments to Search for Dark Gauge Forces. *Phys. Rev. D*, 80:075018, 2009. doi: 10.1103/PhysRevD.80.075018.

[8] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. Relational graph attention networks, 2019. URL `https://openreview.net/forum?id=Bklzkh0qFm`.

[9] Sylvain Caillou, Paolo Calafiura, Steven Andrew Farrell, Xiangyang Ju, Daniel Thomas Murnane, Charline Rougier, Jan Stark, and Alexis Vallier. ATLAS ITk Track Reconstruction with a GNN-based pipeline. Technical report, CERN, Geneva, 2022. URL `https://cds.cern.ch/record/2815578`.

[10] Paolo Calafiura, Jay Chan, Loic Delabrouille, and Brandon Wang. EggNet: An Evolving Graph-based Graph Attention Network for Particle Track Reconstruction, 7 2024.

[11] Serguei Chatrchyan et al. Description and Performance of Track and Primary-Vertex Reconstruction with the CMS Tracker. *JINST*, 9(10):P10009, 2014. doi: 10.1088/1748-0221/9/10/P10009.

[12] CMS collaboration et al. Description and performance of track and primary-vertex reconstruction with the cms tracker. *Journal of Instrumentation*, 9(10):P10009–P10009, 2014.

[13] Gage DeZoort, Savannah Thais, Javier Duarte, Vesal Razavimaleki, Markus Atkinson, Isobel Ojalvo, Mark Neubauer, and Peter Elmer. Charged Particle Tracking via Edge-Classifying Interaction Networks. *Comput. Softw. Big Sci.*, 5(1):26, 2021. doi: 10.1007/s41781-021-00073-z.

[14] Javier Duarte and Jean-Roch Vlimant. Graph Neural Networks for Particle Tracking and Reconstruction, 12 2020.

[15] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[16] Liyu Gong and Qiang Cheng. Exploiting edge features in graph neural networks. *arXiv: Learning*, 2018. URL `https://api.semanticscholar.org/CorpusID:59316985`.

[17] Xiangyang Ju et al. Performance of a geometric deep learning pipeline for HL-LHC particle tracking. *Eur. Phys. J. C*, 81(10):876, 2021. doi: 10.1140/epjc/s10052-021-09675-8.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 12 2014.

[19] Kilian Lieret, Gage DeZoort, Devdoot Chatterjee, Jian Park, Siqi Miao, and Pan Li. High Pileup Particle Tracking with Object Condensation. Technical report, 2023. URL `https://cds.cern.ch/record/2898594`. 8 pages, 6 figures, 8th International Connecting The Dots Workshop (Toulouse 2023).

[20] Are Strandlie and Rudolf Frühwirth. Track and vertex reconstruction: From classical to adaptive methods. *Reviews of modern physics*, 82(2):1419–1458, 2010.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

[22] Ziming Wang, Jun Chen, and Haopeng Chen. Egat: Edge-featured graph attention network. In Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2021*, pages 253–264, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86362-3.