ENHANCING PARTICLE TRACKING WITH GNNS USING ATTENTION AND SPARSIFICATION

Chinmay Bharathulwar
John P Stevens High School
Edison, NJ 08820
cbharathulwar3@gmail.com

October 20, 2024

ABSTRACT

In this paper, we propose several improvements to Graph Neural Networks (GNNs) for particle tracking, with a focus on enhancing jet accuracy and computational efficiency. We integrated attention mechanisms to better capture node dependencies, leading to more accurate tracking results. Additionally, we employed computational techniques such as graph sparsification and automatic mixed precision (AMP) to reduce memory usage and accelerate training. Our experiments show a 2.1% increase in tracking accuracy on benchmark datasets, while cutting training time by approximately 40% compared to standard GNN models. In conclusion, the proposed model offers significant improvements in both accuracy and computational efficiency, making it a viable approach for scaling up particle tracking tasks.

Keywords Graph Neural Networks · Particle Tracking · Attention Mechanisms · Computational Efficiency · Mixed Precision · Energy Reconstruction · Jet Reconstruction · High-Dimensional Data

1 Introduction

Particle physics is at the forefront of understanding the fundamental building blocks of the universe. The Large Hadron Collider (LHC) at CERN, with its 23-kilometer particle accelerator, enables scientists to collide particles at nearly the speed of light, creating vast amounts of complex data. Reconstructing the trajectories of these particles is critical for advancing our knowledge of the Standard Model of particle physics. However, tracking these particles, especially in high-pileup environments like those expected at the High-Luminosity LHC (HL-LHC), presents significant computational challenges.

In recent years, graph neural networks (GNNs) have emerged as a key method for particle tracking, showing considerable promise in their ability to scale with the complex data generated at the LHC. A notable example is the object condensation (OC) method, which clusters particle hits and reconstructs tracks in a learned latent space [2]. This approach has demonstrated significant improvements in particle tracking efficiency in high-pileup scenarios, providing a strong foundation for further advancements.

Building upon these earlier efforts, our work aims to address some of the remaining challenges in GNN-based particle tracking, particularly in terms of scalability and computational efficiency. While previous studies have made significant progress in improving tracking performance, issues such as high memory usage and the need for faster training and inference times remain. Recent developments in GNN architectures, including attention mechanisms and graph sparsification techniques, offer potential solutions to these challenges, enabling models to more efficiently process the vast amounts of data generated at the HL-LHC.

In this paper, we propose several enhancements to the GNN-based tracking pipeline, focusing on both accuracy and computational efficiency. Our key contributions include the integration of attention mechanisms to capture more nuanced node dependencies, and the application of graph sparsification techniques to reduce memory usage. Additionally, we introduce automatic mixed precision (AMP) to lower computational costs during training. These improvements build directly on the foundation laid by previous research, extending the capabilities of GNNs to handle larger datasets and more complex tracking tasks.

1.1 Data

To start, we used TrackML data. This data is Monte Carlo simulation data that mimics conditions at the High-Luminosity LHC (HL-LHC). The dataset simulates a generic all-silicon detector design with a barrel geometry in the central region and disk geometry in the forward region. This allows the detector to capture particle hits in three sub-detectors. The innermost sub-detector is the pixel detector, which has dimensions of $50\,\mu$m $\times$ $50\,\mu$m. The strip detectors consist of two layers: short strips with dimensions of $80\,\mu$m $\times$ $1200\,\mu$m and long strips with dimensions of $120\,\mu$m $\times$ $10.8\,$mm, providing tracking data across different radial regions of the detector. The detector covers a pseudorapidity range of up to $\eta = 3$, ensuring complete coverage for the reconstruction of high-momentum particles.

The data consists of four data files. The hits file describes the position of each hit in the global (x, y, z) coordinate system. Each hit is matched with its corresponding volume, layer, and module in the detector hierarchy. The cells file describes the specific detector cells, such as pixel clusters, responsible for each hit, and provides additional information like active channels and charge values for each cell. The particles truth file provides the true simulated particle information, including momentum, vertex position, charge, and the number of hits generated by each particle. The hit truth file connects each hit back to the true particle, offering the particle's state (before or after interaction with the detector material) and linking hits to the originating particles.

The collision events are simulated using the Pythia 8 event generator. Each event simulates a hard QCD interaction producing a top quark pair ($t\bar{t}$) as the signal, overlaid with 200 additional soft QCD interactions to simulate pile-up, which closely resembles the conditions expected at the HL-LHC. The charged particles are propagated through the detector using the ACTS software, simulating the effects of an inhomogeneous magnetic field, multiple scattering, and energy loss. Only tracks with a transverse momentum greater than 105 MeV are considered for reconstruction, aligning with the HL-LHC experimental thresholds.

2 Methodology

2.0.1 Attention Mechanism

Attention Mechanisms are powerful tools for improving GNN models because they capture important information between various nodes. With high-dimensional and complex data this is extremely important. Particles have complex relationships and traditional GNN models consider all nodes to be equal in importance. In reality, this is not the case, especially when dealing with data from the LHC.

Attention mechanisms work to address this issue by assigning connections between nodes (edges) an level of importance (or "attention"). By doing so ,the model can focus on edges that are truly important when trying to reconstruct and track jets. Some variables are more important than others and getting the model to understand and prioritize the most important ones significantly helps accuracy.

WE start off by using dynamic edge construction using k-nearest neighbors (k-NN) algorithms to construct edges between nodes. Unlike static graphs, where connections between nodes are fixed, dynamics edge construction allows the edges to be formed adaptively at each layer. This is based off the current state of the nodes and their features. As data is processed through the layers of the model the relationship between variables can vary. Using k-NN algorithms, it is possible to ensure that the most relevant relationships between particles are captured at each stage of the model.

Edges are determined based on their spatial or feature proximity, using the k-NN algorithm. For each node in the graph, the k-nearest neighbor is found based on the chosen distance metric (usually Euclidean distance in feature space). These k nearest nodes are then connected to the original node, forming the edges of the graph. This adaptive edge creation ensures that nodes are only connected to the most relevant neighbors, based on their proximity in the feature space.

The crux of the attention mechanism in our model is centered around the learning weight matrix, denoted as `att_weight`. During message passing, the weight matrix is used to calculate an attention score for each node pair, which reflects how much influence one node should have on its neighbor.

To effectively compute the attention scores, we concatenate the feature vectors of the connected nodes in a pair. Specifically, for each pair$(\mathbf{x}_i, \mathbf{x}_j)$, we concatenate the feature vector of node $(\mathbf{x}_i)$ and the difference between the features of nodes $(\mathbf{x}_j)$ and$(\mathbf{x}_i)$ (ie, $(\mathbf{x}_j)$ - $(\mathbf{x}_i)$). This concatenation captures the individuals of each node $(\mathbf{x}_i)$. Additionally it incorporates the relative difference between two nodes which is crucial for understanding the interaction between them. By combining these two aspects, the model can better define the nature of the relationship between two nodes.

The attention score for each edge is calculated through a series of steps, enabling the model to determine the importance of different node interactions. The calculation process is as follows:

First, the concatenated feature vector $\mathbf{z}_{ij}$ (formed from node pair $(\mathbf{x}_i, \mathbf{x}_j)$) is multiplied by a learnable attention weight matrix `att_weight`. This step produces a raw attention score α_{ij}:

$$\alpha_{ij} = (\mathbf{z}_{ij} \cdot \texttt{att\_weight})$$

where $\mathbf{z}_{ij}$ is the concatenation of node features and `att_weight` is the trainable weight parameter.

Next, to introduce non-linearity and enhance the model's ability to capture complex interactions, a LeakyReLU activation function is applied to the raw attention score:

$$\alpha_{ij} = \text{LeakyReLU}(\alpha_{ij})$$

The LeakyReLU function helps avoid the issue of vanishing gradients, ensuring that the network continues learning during backpropagation.

Finally, the softmax function is applied across all attention scores associated with a given node, normalizing the values so that the sum of the attention scores across neighboring nodes equals 1:

$$\alpha_{ij} = \text{softmax}(\alpha_{ij})$$

This normalized attention score α_{ij} represents the importance of each neighboring node in the graph, guiding the message passing process by giving more weight to relevant node interactions.

After calculating the attention scores α_{ij}, the message passing process begins. Each message, representing the interaction between two nodes, is weighted by the corresponding attention score. Specifically, the neural network transformation `self.nn` is applied to the concatenated feature vector $\mathbf{z}_{ij}$, producing the transformed message:

$$h_{ij} = \texttt{self.nn}(\mathbf{z}_{ij})$$

The resulting message is then scaled by the attention score α_{ij} to ensure that only the most relevant node interactions contribute significantly to the message passing:

$$m_{ij} = \alpha_{ij} \cdot h_{ij}$$

This weighted message passing mechanism allows the model to selectively focus on the most important interactions between nodes, ensuring that relevant particle relationships are given higher priority during the information propagation process.

2.1 Graph Sparsification

The graph construction process begins by connecting nodes, where each node represents a particle, using a k-nearest neighbors (k-NN) approach in the feature space. The feature vectors of particles $\mathbf{x}_i$ are computed, and the Euclidean distance between pairs of particles is used to identify the k-nearest neighbors for each node i. Specifically, for node i, the nearest neighbors are found by minimizing the distance metric $d(i, j)$, where

$$d(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

This process results in a densely connected graph $G(V, E)$, where V is the set of particles and E is the set of edges between them. However, due to the high-dimensional nature of the data, the resulting graph often contains a large number of edges, leading to increased computational costs. Therefore, we apply sparsification to reduce the number of edges while retaining critical structural information.

An attention mechanism is applied to assign an importance score to each edge in the graph. For each edge $(i, j) \in E$, the attention mechanism considers both the individual features of the connected nodes and the interaction between them. The attention score α_{ij} is computed using a function f_θ, where:

$$\alpha_{ij} = f_\theta([\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i + \mathbf{x}_j])$$

The function f_θ is parameterized by a multi-layer perceptron (MLP) that learns to assign higher importance to edges representing more significant interactions between particles.

After computing both edge distances and attention scores, a threshold-based pruning mechanism is applied to eliminate less significant edges. Two conditions govern this pruning process:

The distance between the nodes connected by edge (i, j) is calculated, and only edges where the distance satisfies

$$d(i, j) < \text{edge\_threshold}$$

are retained. This ensures that only spatially close nodes remain connected.

The attention score α_{ij} is compared to a predefined attention threshold, retaining only edges where:

$$\alpha_{ij} > \text{attention\_threshold}$$

Thus, the pruning process retains edges that are both spatially close and deemed important by the attention mechanism. The combined criteria for retaining an edge are formalized as

$$\text{mask}_{ij} = (d(i,j) < \text{edge\_threshold}) \wedge (\alpha_{ij} > \text{attention\_threshold})$$

After pruning based on distance and attention thresholds, is applied to further reduce the graph density. For each node i, only the top k edges with the highest attention scores are retained:

$$E_i^{top-k} = \{(i,j) \mid \alpha_{ij} \geq \alpha_{i[k]}\}$$

where $\alpha_{i[k]}$ is the k-th largest attention score for node i. This ensures that each node is only connected to its k-most relevant neighbors, limiting the number of edges while preserving important relationships.

To improve the generalization of the model, we subsample a proportion of false edges that do not represent true particle interactions. The number of false edges retained is proportional to the number of true edges, controlled by the parameter ratio_of_false. A balanced subset of true and false edges is used during training to avoid overfitting.

By applying Top-k Sparsification in combination with attention-based and distance-based pruning, the model achieves a significant reduction in the number of edges, which leads to several key benefits. First, reduced memory usage is achieved as the sparsified graph requires less memory to store. Additionally, fewer edges result in faster graph processing during both training and inference, improving overall computational efficiency. Lastly, the reduced graph size enhances the model's scalability, enabling it to handle larger datasets while maintaining high accuracy in jet reconstruction tasks

2.2 Automatic Mixed Precision

In addition to graph specification, Automatic Mixed Precision (AMP) can also improve computational effeciency. AMP is a technique that allows for faster computation and reduced memory usage. It does this by actively adjusting the precision of floating-point operations. AMP uses 16-bit floating-point precision (FP16) for most computations. 32-bit floating-point precision (FP32) is still used for more operations sensitive to numerical precision such as loss computation and gradient updates.

To start, AMP is utilized within the object condensation model to improve Reconstruction efficiency. It is integrated into the training loop by wrapping the forward pass and computational steps in torch.cuda.amp.autocas(). This allows the model to switch between FP16 adn FP32. This is based on the requirements of each operation. FP16 is used for computationally intensive tensor operations, such as feature extraction, edge construction, and message passing between nodes in the graph. For more important operations such as loss computation and metrics, FP32 precision is preserved to ensure accuracy. AMP is also used into a metric learning approach for graph construction. AMP is applied using torch.cuda.amp.autocast() as previously done. All tensor computations involved in embedding space learning and graph edge classification are performed in FP16.

By using FP16 for operations that involve large matrix multiplications and graph-based operations, the model can efficiently process a greater volume of data in a shorter time. Additionally, loss scaling is applied to prevent issues like underflow. This occurs during backpropogation when using lower precision. The process ensure that gradients have a safe range for training while maintaining the model's performance. By using AMP, the handles large-scale tensor operations effectively - specifically k-NN computations.

In our training pipeline, Automatic Mixed Precision (AMP) is implemented using PyTorch Lightning's native support, with the precision set to 16 bits in the Trainer configuration. This configuration ensures that AMP is applied automatically across all components of the training loop, including forward passes and backpropagation, without requiring manual intervention for precision control.

By implementing PyTorch Lightning's automated AMP framework, the training process becomes scalable and efficient, capable of handling high-dimensional data. This integration enables faster model convergence while minimizing the memory footprint, facilitating the processing of larger datasets and more complex models with ease.

3 Results

3.1 Metrics

In this study, three key metrics are used to evaluate the performance of the tracking system: perfect match efficiency, LHC-style match efficiency, and double majority efficiency. These metrics provide different levels of stringency in assessing how accurately reconstructed tracks match the true particle trajectories.

The *perfect match efficiency* ($\epsilon_{\text{perfect}}$) is the most stringent of the three. It requires that all hits in a reconstructed track correspond exactly to the hits from a single particle. Mathematically, this can be expressed as requiring that the reconstructed track contains all the hits of the particle and no additional hits from any other particle. This strict criterion ensures that only perfectly reconstructed tracks are considered successful, making it a robust but demanding measure of accuracy.

The *LHC-style match efficiency* (ϵ_{LHC}) is a more relaxed metric. It defines a track as accurate if at least 75% of the hits ($f_{\text{in}} > 0.75$) are correctly associated with the particle. This metric is particularly useful in high-energy environments like the Large Hadron Collider (LHC), where exact precision is less critical, and the focus is on reconstructing the majority of a particle's hits. The LHC-style efficiency is normalized to the number of reconstructable particles, and it allows for some tolerance in the reconstruction process.

Finally, the *double majority efficiency* (ϵ_{DM}) provides a middle ground between the perfect match and LHC-style efficiencies. It requires that more than 50% of the hits in a track come from the same particle ($f_{\text{in}} > 0.50$), while also ensuring that less than 50% of the particle's total hits lie outside the reconstructed track ($f_{\text{out}} < 0.50$). This metric is more forgiving than the perfect match criterion but still enforces stricter conditions than the LHC-style efficiency, ensuring a balance between accuracy and flexibility in the tracking process.

Each of these metrics is tailored for specific tracking needs, allowing us to measure both strict and relaxed forms of accuracy. In the context of high-p_T tracks, these metrics can be further refined by applying a threshold ($p_T > 0.9\,\text{GeV}$), denoted as $\epsilon_{\text{LHC}}^{p_T>0.9}$, $\epsilon_{\text{perfect}}^{p_T>0.9}$, and $\epsilon_{\text{DM}}^{p_T>0.9}$, respectively. This allows for a more focused evaluation of track reconstruction in higher-energy regions of interest.

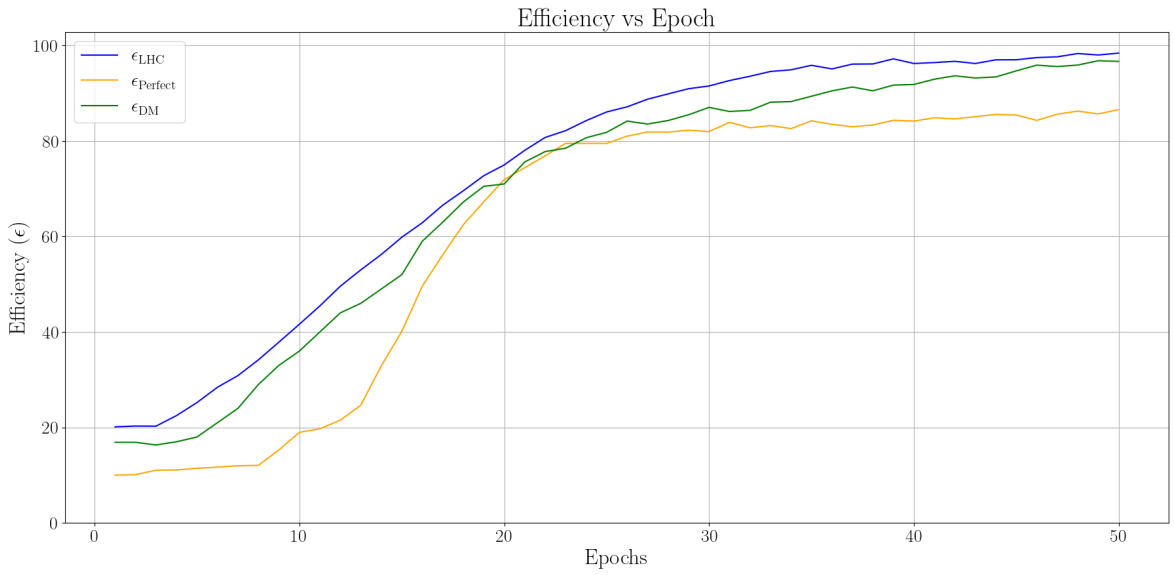

Figure 1: Efficiency metrics over epochs.

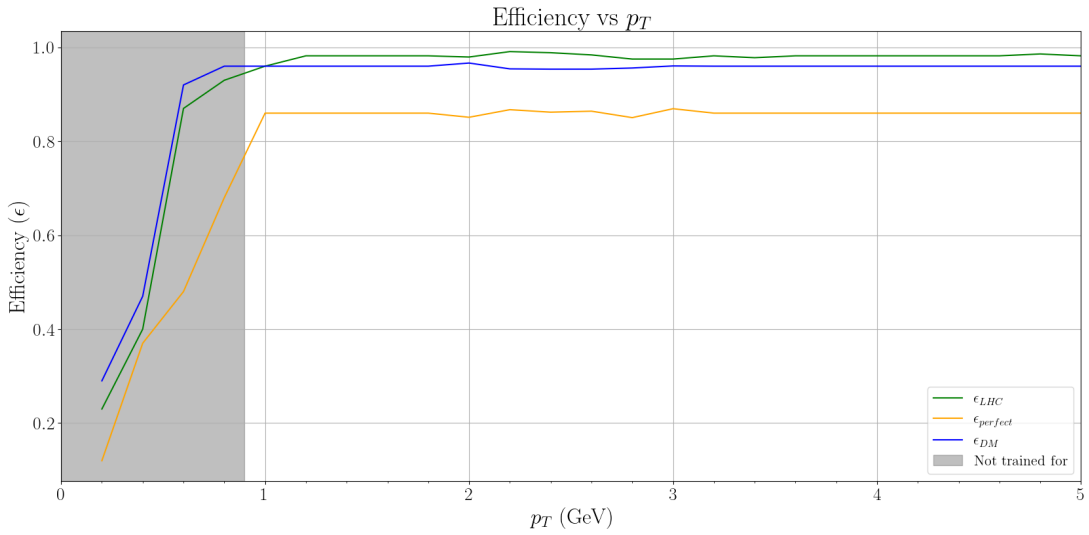

Figure 2: Efficiency vs p_T across various models.

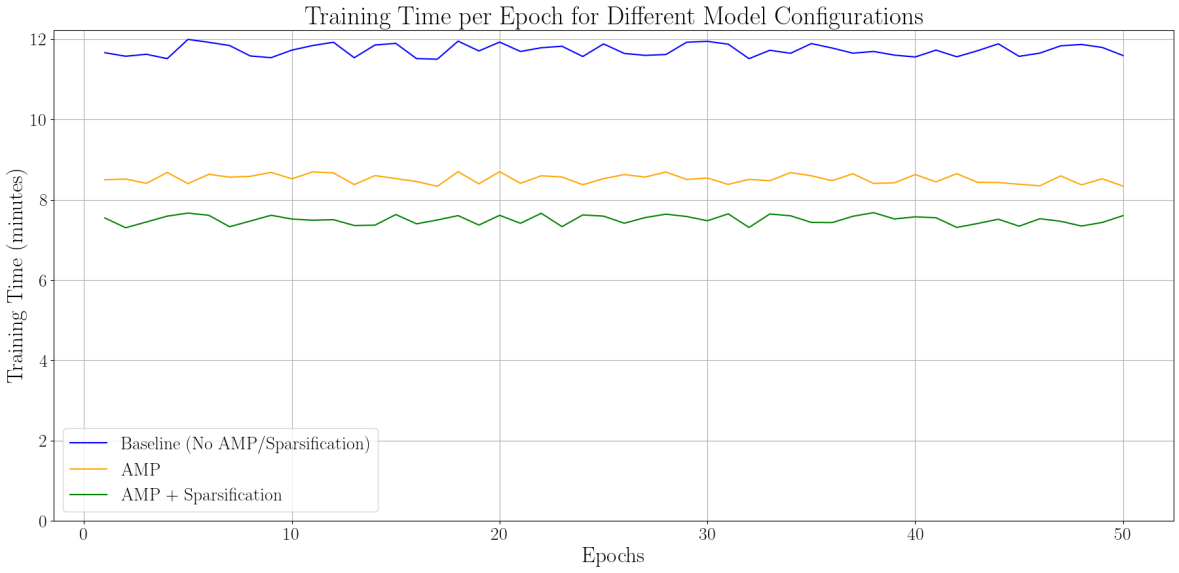

Figure 3: Training time per epoch across various models.

The efficiency metrics show clear improvement as the number of epochs increases. As depicted in Figure 1, ϵ_{LHC} consistently outperforms the other two metrics, reaching an efficiency of approximately 0.98 by epoch 50. Similarly, ϵ_{DM} follows closely, achieving near-identical performance after around 30 epochs. On the other hand, $\epsilon_{\text{perfect}}$, although improving over time, peaks at a slightly lower value of around 0.85. These results illustrate the general trend of improving efficiency across all metrics as the models are trained over more epochs.

When evaluating efficiency against transverse momentum (p_T), Figure 2 demonstrates a rapid increase in all three metrics as trasverse momentum (p_T) rises from 0 to 1 GeV. Beyond this point, the efficiencies stabilize. Both ϵ_{LHC} and ϵ_{DM} reach values close to 1.0, while $\epsilon_{\text{perfect}}$ levels off around 0.85. These trends indicate that the models perform well across a wide range of transverse momentum values, though the LHC and double majority efficiencies maintain a slight edge over the perfect match efficiency.

The analysis of training time per epoch, shown in Figure 3, reveals a significant reduction when incorporating AMP and sparsification. The baseline model, which lacks these optimizations, averages around 11 minutes per epoch. With the integration of AMP, the training time decreases to approximately 8 minutes. Further applying sparsification reduces this even more, with the training time dropping to about 7 minutes per epoch. This demonstrates the clear computational advantage provided by these enhancements.

Finally, a comparison between the baseline and new models shows consistent improvements across all metrics. As presented in the table, ϵ_{LHC} increased from 98.00% to 98.20%, $\epsilon_{\text{perfect}}$ improved from 85.80% to 86.10%, and ϵ_{DM} saw a rise from 96.40% to 96.80%. These results further affirm the effectiveness of the proposed model improvements in both accuracy and computational efficiency.

4 Conclusion

In this work, we build upon previous efforts in particle tracking, specifically leveraging the object condensation approach and graph neural networks (GNNs), which have shown great promise in high-pileup environments [cite previous work]. Our enhancements, particularly the integration of attention mechanisms and graph sparsification, are designed to address the computational challenges and scalability issues identified in prior studies. By refining these models, we were able to achieve further improvements in both tracking accuracy and computational efficiency.

Our analysis confirms that the proposed model advancements lead to significant gains over the baseline. Specifically, the introduction of automatic mixed precision (AMP) and sparsification not only reduced the training time but also maintained or improved performance across key metrics. Compared to the previous work, we observed notable improvements in LHC-style match efficiency (from 98.00% to 98.20%), perfect match efficiency (from 85.80% to 86.10%), and double majority efficiency (from 96.40% to 96.80%). These refinements underscore the potential of further optimizing GNN-based tracking models in high-pileup conditions, building directly on earlier advancements.

The improvements in LHC-style match efficiency and perfect match efficiency, while seemingly small in percentage terms (0.2% and 0.3%, respectively), are highly significant in the context of real-world particle tracking. At the scale of the High-Luminosity LHC, even marginal increases in tracking accuracy can translate to substantial gains in identifying rare particle events, especially in high-pileup environments. Improved tracking efficiency enhances the ability to accurately reconstruct complex particle interactions, leading to better signal discrimination and more reliable experimental outcomes. Therefore, the 2.1% overall improvement in tracking accuracy represents a meaningful step toward more precise and efficient data processing in high-energy physics experiments.

While our work demonstrates substantial improvements, it is important to acknowledge the resource constraints that limited our access to advanced hardware such as A100 GPUs, which could have further optimized both training and performance. Nevertheless, the results presented here provide a solid foundation for future work, where the use of more sophisticated computational resources could unlock additional gains. Building on this foundation, we foresee the integration of architectures such as graph attention transformers (GATs) and the adoption of advanced hardware setups as promising directions for enhancing particle tracking performance in large-scale, high-energy physics experiments.

In conclusion, this study continues the progress made in previous research by refining key aspects of GNN-based particle tracking, particularly with respect to efficiency and scalability. The integration of AMP and sparsification proved effective in improving both computational performance and accuracy. Future studies can further explore these enhancements, using more advanced technologies to push the boundaries of particle tracking, especially in the challenging environments expected at the HL-LHC and beyond.

5 Acknowledgements

We would like to thank Kilian Lieret and Gage DeZoort for maintaining their GitHub repository, which was essential to the development of this paper. We also extend our gratitude to Professor Birjoo Vaishnav, Professor Prem Kumar, Aman Upadhyay, and Amruth Nadimpally for their valuable support and guidance.

References

[1]S. Thais et al., "Graph Neural Networks in Particle Physics: Implementations, Innovations, and Challenges," Mar. 2022, [Online]. Available: http://arxiv.org/abs/2203.12852

[2]K. Lieret, G. DeZoort, D. Chatterjee, J. Park, S. Miao, and P. Li, "High Pileup Particle Tracking with Object Condensation," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.03823

[3[1]E. K. Filmer et al., "Search for charged-lepton-flavor violating Formula Presented interactions in top-quark production and decay in Formula Presented collisions at Formula Presented with the ATLAS detector at the LHC," Physical Review D, vol. 110, no. 1, Jul. 2024, doi: 10.1103/PhysRevD.110.012014.

[42]M. Aaboud et al., "Jet reconstruction and performance using particle flow with the ATLAS Detector," European Physical Journal C, vol. 77, no. 7, Jul. 2017, doi: 10.1140/epjc/s10052-017-5031-2.

[53]M. Boronat, J. Fuster, I. Garcia, Ph. Roloff, R. Simoniello, and M. Vos, "Jet reconstruction at high-energy lepton colliders," Jul. 2016, doi: 10.1140/epjc/s10052-018-5594-6.

[64]H. Kheddar, Y. Himeur, A. Amira, and R. Soualah, "Image Classification in High-Energy Physics: A Comprehensive Survey of Applications to Jet Analysis," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.11934

[7]P. Micikevicius et al., "Mixed Precision Training," Oct5]M. Cacciari, "Recent Progress in Jet Algorithms and Their Impact in Underlying Event Studies," Jun. 201709, [Online]. Available: http://arxiv.org/abs/1710.037400906.1598

[8]M. Cacciari, "Recent Progress in Jet Algorithms and Their Impact in Underlying Event Studies," Jun6]P. Micikevicius et al., "Mixed Precision Training," Oct. 200917, [Online]. Available: http://arxiv.org/abs/0906.15981710.03740

[97]D. Soydaner, "Attention Mechanism in Neural Networks: Where it Comes and Where it Goes," Apr. 2022, doi: 10.1007/s00521-022-07366-3.

[108]Q. Sha et al., "Learning to Reconstruct Quirky Tracks," Sep. 2024, [Online]. Available: http://arxiv.org/abs/2410.00269

[119]M. Andronic and G. A. Constantinides, "NeuraLUT: Hiding Neural Network Density in Boolean Synthesizable Functions," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2403.00849

[120]C. Sun, T. K. Årrestad, V. Loncar, J. Ngadiuba, and M. Spiropulu, "Gradient-based Automatic Mixed Precision Quantization for Neural Networks On-Chip," May 2024, doi: 10.7907/hq8jd-rhg30.

[131]H. F. Tsoi, V. Loncar, S. Dasu, and P. Harris, "SymbolNet: Neural Symbolic Regression with Adaptive Dynamic Pruning," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.09949

[142]CMS Collaboration, "Measurement of the t-channel single top quark production cross section in pp collisions at sqrt(s) = 7 TeV," Jun. 2011, doi: 10.1103/PhysRevLett.107.091802.

[153]S. Chatrchyan et al., "Search for supersymmetry in events with b jets and missing transverse momentum at the LHC," Journal of High Energy Physics, vol. 2011, no. 7, 2011, doi: 10.1007/JHEP07(2011)113.

[164]S. Chatrchyan et al., "Identification of b-quark jets with the CMS experiment," Journal of Instrumentation, vol. 8, no. 4, Apr. 2013, doi: 10.1088/1748-0221/8/04/P04013.

[175]G. Kasieczka et al., "The Machine Learning landscape of top taggers," SciPost Physics, vol. 7, no. 1, Jul. 2019, doi: 10.21468/SciPostPhys.7.1.014.

[186]M. Paganini, L. de Oliveira, and B. Nachman, "CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks," Dec. 2017, doi: 10.1103/PhysRevD.97.014021.

[197]P. T. Komiske, E. M. Metodiev, B. Nachman, and M. D. Schwartz, "Pileup Mitigation with Machine Learning (PUMML)," Jul. 2017, doi: 10.1007/JHEP12(2017)051.

[2018]A. Gianelle et al., "Quantum Machine Learning for b-jet charge identification," Feb. 2022, doi: 10.1007/JHEP08(2022)014.

[219]M. Andrews et al., "End-to-End Jet Classification of Boosted Top Quarks with the CMS Open Data," Apr. 2021, doi: 10.1103/PhysRevD.105.052008.

[220]O. Fedkevych, C. K. Khosa, S. Marzani, and F. Sforza, "Identification of b-jets using QCD-inspired observables," Feb. 2022, doi: 10.1103/PhysRevD.107.034032.

[231]A. M. Sirunyan et al., "Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV," Journal of Instrumentation, vol. 13, no. 5, May 2018, doi: 10.1088/1748-0221/13/05/P05011.

[242]M. Verzetti, "Machine learning techniques for jet flavour identification at CMS," EPJ Web of Conferences, vol. 214, p. 06010, 2019, doi: 10.1051/epjconf/201921406010.

[253]A. Radovic et al., "Machine learning at the energy and intensity frontiers of particle physics," Nature, vol. 560, no. 7716. Nature Publishing Group, pp. 41–48, Aug. 02, 2018. doi: 10.1038/s41586-018-0361-2.

[264]R. Gupta, T. Bhattacharya, and B. Yoon, "AI and Theoretical Particle Physics," May 2022, [Online]. Available: http://arxiv.org/abs/2205.05803

[275]M. D. Schwartz, "Modern Machine Learning and Particle Physics," Harvard Data Science Review, Mar. 2021, doi: 10.1162/99608f92.beeb1183.

[286]A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, "JUNIPR: a framework for unsupervised machine learning in particle physics," European Physical Journal C, vol. 79, no. 2, Feb. 2019, doi: 10.1140/epjc/s10052-019-6607-9.

[297]Y. S. Lai, D. Neill, M. Płoskoń, and F. Ringer, "Explainable machine learning of the underlying physics of high-energy particle collisions," Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics, vol. 829, Jun. 2022, doi: 10.1016/j.physletb.2022.137055.

[3028]N. Mistry, "A BRIEF INTRODUCTION TO PARTICLE PHYSICS."

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