



Graph attention neural network for clustering particle tracks by events in the SPD experiment

S. Omelanchuk^{1,a} , Y. Talochka¹ , N. Voytishin¹ , G. Ososkov¹

¹ Meshcheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research, Joliot-Curie Str. 6, Dubna 141980, Moscow Reg., Russia

Received: 21 March 2026 / Accepted: 23 June 2026
© The Author(s) 2026

Abstract A Graph Attention Neural Network (GANN) for particle track clustering by events in time slices obtained at the Spin Physics Detector (SPD) of the Nuclotron-based Ion Collider fAcility (NICA) is presented. A novel approach is applied to time slice processing at high particle multiplicity and pile-up. The model is trained using a hierarchical approach and evaluated on time slices generated by the Monte Carlo technique according to the SPD experiment configuration. Additionally, the model is trained and evaluated on time slices generated using the TrackML dataset to analyze its stability. The model architecture includes a graph encoder and an edge classifier, both of which use the message-passing principle and iteratively structured approach. The GANN exhibits high performance in track clustering by events for the SPD dataset, achieving 99% accuracy and 95% precision and recall.

1 Introduction

The Spin Physics Detector (SPD) is a future multipurpose experiment at the NICA collider (JINR, Dubna), aimed to study the nucleon spin structure and other non-perturbative QCD phenomena [1] in polarized proton-proton and nucleus-nucleus collisions. Under high-luminosity conditions, NICA is designed to reach up to $\sim 10^{32} \text{ cm}^{-2} \text{ s}^{-1}$ for pp collisions and even higher for ion runs.

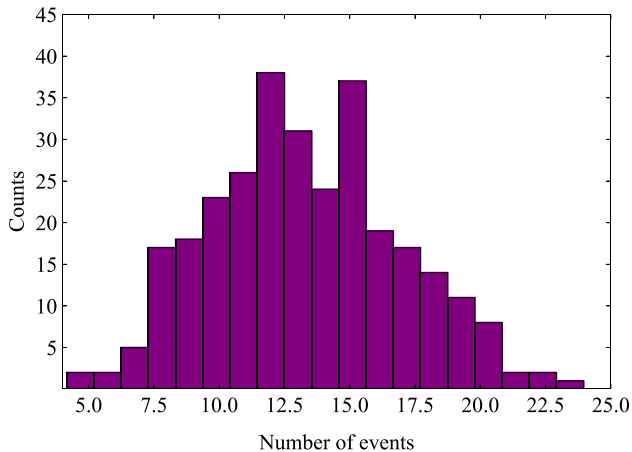
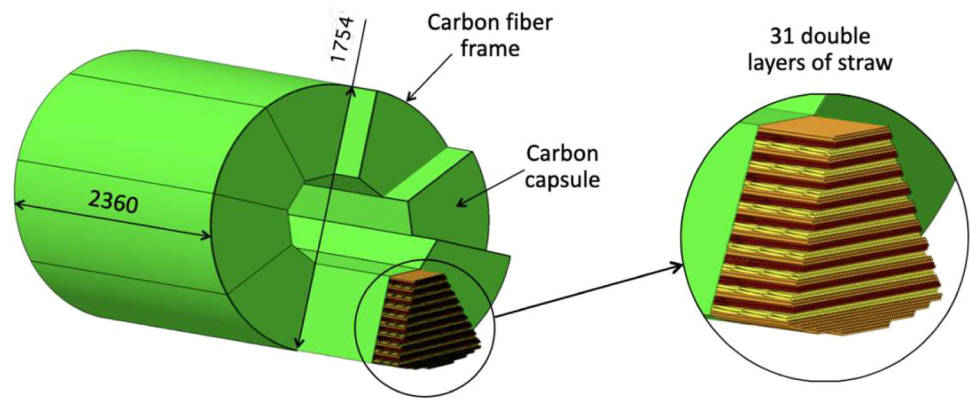
The experimental conditions of SPD are unique. The cornerstone of the experiment is the use of polarized beams. NICA will accelerate and collide polarized protons, deuterons and light nuclei. This allows for measuring spin asymmetries in the production of various particles (pions, kaons, hyperons, D-mesons, etc.). The collisions will occur at a center-of-mass energy of $\sqrt{s} = 10\text{--}27 \text{ GeV}$ (for proton-

proton collisions) [2]. For the precise track reconstruction of secondary vertices (to identify heavy hadrons, e.g., charm mesons) and the measurement of hyperon polarization, a sophisticated experimental setup containing a strong magnetic field, a vertex detector, a tracker, Cherenkov detectors, calorimeters, and a muon system needs to be constructed.

The main challenges for the SPD experiment are the high luminosity and the resulting data rate. At the designed luminosity level, the expected event rate for pp collisions is $\sim 50 \text{ kHz}$. Considering the estimation that a raw event produces $\sim 0.1\text{--}1 \text{ MB}$, the raw data rate is expected at an order of $\sim 100 \text{ TB/s}$, which is impossible to store or fully process online. Along with the fact that the readout electronics will operate in continuous mode with no hardware trigger, software-based online event selection is required. This combination of a triggerless DAQ system with a software filter is called the SPD Online Filter [1].

The usage of an online filter is justified by the fact that it is impossible to store the full stream of raw data coming at the expected rates. This stream should be limited to typical $\sim 10 \text{ GB/s}$, which is the upper limit for large systems. On the other hand, only a small fraction of collisions contain signals of interest, and, under high pile-up conditions, the online rejection of low-significance events is needed to avoid swamping storage with minimum-bias data. The detector front-end electronics digitize signals continuously and ship data to compute nodes with timestamps. In this case, instead of event-by-event readout, the DAQ is designed to operate on time slices (also called “frames” or “integration intervals”) [3]. A time slice is meant to contain $\sim 1\text{--}10 \text{ ms}$ of real-time data including up to 40 overlapped collisions (events). Each time slice is processed by a large farm of processors (FPGA/GPU/CPU) that run reconstruction algorithms online to disentangle individual events and select the most interesting ones.

^a e-mail: omelancuksavelij@gmail.com (corresponding author)

Fig. 1 SPD straw tracker geometry [1]**Fig. 2** Distribution of the number of events in SPD time slices

Disentangling events in a time slice requires online reconstruction with precision normally done offline, which is a major computing challenge, feasible with modern parallel architectures (GPUs/FPGAs). For this purpose, a two-stage procedure is proposed [4]. On the first stage, a recurrent neural network is used to find all tracks in a time slice, i.e., link all hits belonging to the same track, and get rid of the abundance of fake hits [5]. Once a clean time slice with reconstructed tracks is obtained, event disentangling can be finalized by using a neural network. At this stage, the reconstructed tracks must be grouped into separate events by clustering feature vectors. Previously, a Siamese neural network demonstrated accuracy, recall, and precision metrics of 58%, 65%, and 57%, respectively, for this task under conditions of a non-fixed number of events per time slice [4]. In this paper, we present a fast and precise event separation algorithm based on the GANN [6].

2 Datasets

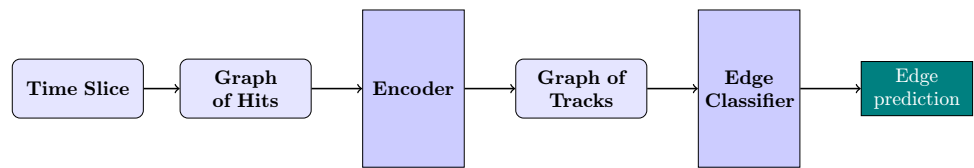
We utilized a dataset that includes a set of time slices obtained using a custom generator for SPD [7]. The SPD tracker system consists of 31 stations along the radial direction, as shown in Fig. 1. We used the geometric parameters of the SPD straw tracker presented in Ref. [4] to correctly compare the performance of the neural networks in this study with the performance of the deep Siamese neural network. Each time slice includes a set of primary particle interaction events.

The number of events per time slice is distributed according to the Poisson distribution (see Fig. 2). In this study, we used time slices with an average number of events $\bar{N}_{ev} = 15$ for training and evaluating the model. The number of tracks within a single event is distributed uniformly from 2 to 6. The dataset stores 3D points (hits) representing particle interactions with the detector material in each event, data linking hits associated with the same track, and data linking tracks associated with the same event. To analyze the stability of the model under study, we additionally utilized the TrackML dataset [8]. This dataset provides simulated measurements of a silicon-based tracking detector with a generic LHC-like geometry, consisting of multiple barrel and end-cap layers. The data include highly curved trajectories of particles with low transverse momentum values (< 200 MeV). The main differences between two datasets are presented in Table 1.

Additionally, we included drift time intervals for each track in both datasets to take into account the overlap in time between tracks that cannot be resolved at the hardware stage. Such intervals are generated uniformly according to the formulas:

Table 1 Comparison of the SPD and TrackML datasets

	Number of hits per track	Vertex smearing (mm)	Radial range (mm)	Spiral	Equidistant hits
SPD tracks	(28, 35)	(−300, 300)	(75, 850)	Low	Yes
TrackML tracks	(3, 20)	(−350, 350)	(50, 1000)	High	No

Fig. 3 Graph Neural Network architecture

$$t_d^i = \frac{i}{N_{ev} - 1}, \quad \Delta t_d = \frac{t_d^{r+1} - t_d^0}{2} - \varepsilon, \quad t_{L/R}^i = t_d^i \mp \Delta t_d,$$

where $i = 1, \dots, N_{ev} - 1$ is the event index, N_{ev} is the number of events in the time slice, t_d^i is the center of the drift time interval for the i -th event, Δt_d is the unified detector error for all events, and t_L^i and t_R^i are the left and right bounds of the drift time interval for the i -th event, r is the overlap parameter, i.e., the event overlap within the drift time intervals for both time directions, and $\varepsilon \sim 10^{-8}$ is a small parameter. The parameter r allows us to control the drift time resolution in the datasets. A larger r leads to the lower time resolution of the detectors.

3 Model

We used the GNN model based on the architecture presented in Ref. [9], where it is applied for enhanced tracking. The architecture diagram is shown in Fig. 3. The diagrams of the main GNN components, the Graph Encoder (GE) and the Edge Classifier (EC), are shown in Figs. 4 and 5. The main idea of our approach is to encode all tracks of a time slice into supernodes (tracks) of a supergraph (graph of tracks), to build connections between such supernodes that can be assigned to the same cluster (taking into account their drift intervals), and perform edge classification. The Cartesian coordinates (x, y, z) of the hits were used as node features for the GE. The model was built and evaluated using the PyTorch framework [10]. The architecture of the GE proposed in this study consists of four main components:

- **Initial graph layer** designed as a Graph Attention (GAT) layer [11] with layer sizes [3, 64], which transforms input nodes (hits) into their latent representations and calculates initial attention coefficients between hits.
- **Convolution graph layer** designed as a GAT layer with layer sizes [64, 64], which aggregates hit features 6 times until convergence is achieved.
- **Pooling layer**, which aggregates hits into supernodes by averaging the feature vectors of hits belonging to the same track.
- **Final linear layer** designed as an MLP [12] with layer sizes [64, 128, 128, 64, 32], which reduces the final dimensionality of the latent supernode representation.

After the encoder stage, supernodes are connected to each other in a supergraph according to the drift-time overlap principle, i.e., two nodes with indices i and j are connected if $[t_L^i, t_R^i] \cap [t_L^j, t_R^j] \neq \emptyset$. Edge labels take two values: 0 and 1, where 1 indicates that two nodes (tracks) correspond to the same cluster (event), marking the edge as true, whereas 0 denotes a false edge. The initial edge purity, i.e., the average ratio of the number of true edges to the total number of edges in a supergraph, is 9.2% in the SPD dataset at $\overline{N}_{ev} = 15$, $r = 5$. The EC evaluates the edge scores, which are compared with the ground-truth labels of edges for the loss function calculation. The EC is designed on the basis of the Graph Isomorphism Network (GIN) architecture [13], where node hidden representations and edge scores are updated via an MLP. The architecture of the EC proposed in this study consists of four main components:

- **Initial edge classifier** designed as an MLP with layer sizes [64, 128, 128, 1], which evaluates attention coefficients according to the formula:

$$\alpha_{ij}^0 = \sigma \left(\text{MLP}(h_i^0 || h_j^0) \right),$$

where h_i^0 is the feature vector of the i -th node (track) at the initial iteration, α_{ij}^0 is the initial attention coefficient for the edge connecting the i -th and j -th nodes (tracks), σ is the sigmoid function.

- **Aggregation layer** designed as a message-passing mechanism with a weighted sum as the aggregation function according to the formula:

$$m_i^l = \sum_{k \in \mathcal{N}(i)} \alpha_{ik}^l h_k^l,$$

where m_i^l is the weighted message to the i -th node (track) from its set of neighbors $\mathcal{N}(i)$ at the l -th iteration, whereas h_k^l and α_{ik}^l are the feature vector of the i -th node (track) and the attention coefficient for the edge connecting the i -th and k -th nodes (tracks) at the l -th iteration, respectively.

- **Node embedding updater** designed as an MLP with layer sizes [64, 128, 128, 32], which updates the node latent representation using the message and its own feature vector as

$$h_i^{l+1} = \text{MLP}(m_i^l || h_i^l),$$

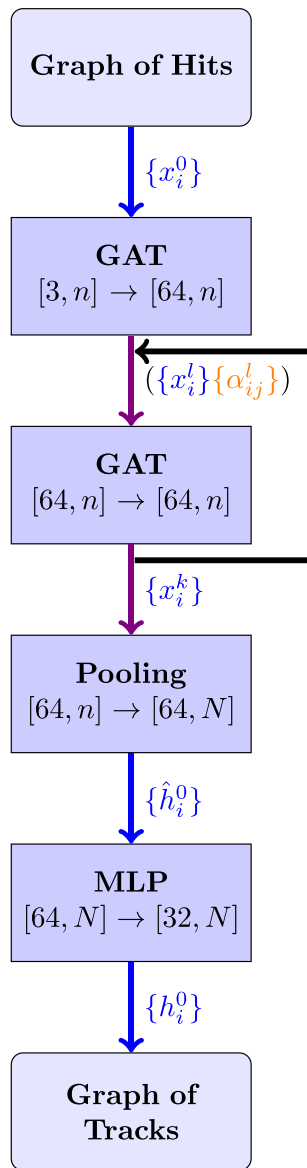


Fig. 4 Graph Encoder architecture

where h_i^{l+1} is the updated feature vector of the i -th node (track).

- **Edge classifier** designed as an MLP with layer sizes $[65, 128, 128, 1]$, which predicts edge labels (attention coefficient) using the updated node and edge embeddings:

$$\alpha_{ij}^{l+1} = \sigma \left(\text{MLP}(h_i^{l+1} || h_j^{l+1} || \alpha_{ij}^l) \right),$$

where α_{ij}^{l+1} is the updated attention coefficient for the edge connecting the i -th and j -th nodes (tracks).

The last three blocks were repeated $n = 6$ times to achieve the convergence of node embeddings, labels and embeddings of edges.

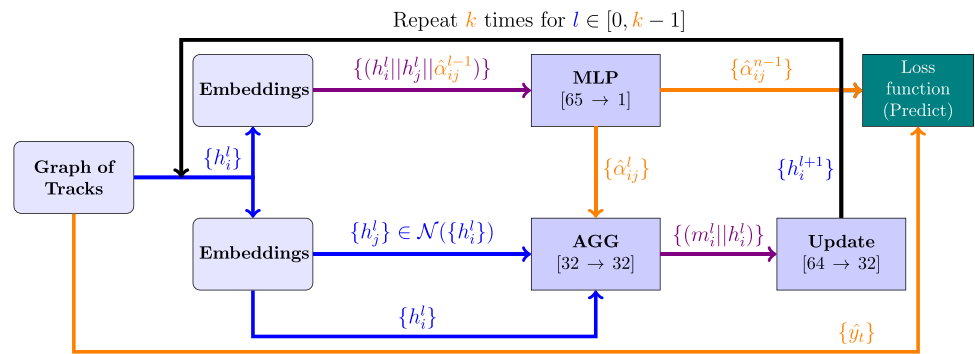
4 Hierarchical training

We applied a hierarchical training approach to our model. The main idea of this approach is to start training the model with simple examples and then progressively increase their complexity. The overlap parameter r controls the drift time resolution of the detectors and is the only free parameter in this study that affects the dataset complexity. Thus, the hierarchical training process starts at $r = 2$ and continues, with the value of r incremented by one at each subsequent stage, until $r = 5$ is reached, which corresponds to the highest pileup level in the experiment. We used the Balanced FoCal loss (BFC) [14] as the loss function with the AdamW optimizer [15] for training. Using the BFC loss enables us to adjust the precision-recall balance. The number of training epochs was set to 100, whereas the batch size was set to 1.

5 Results and discussion

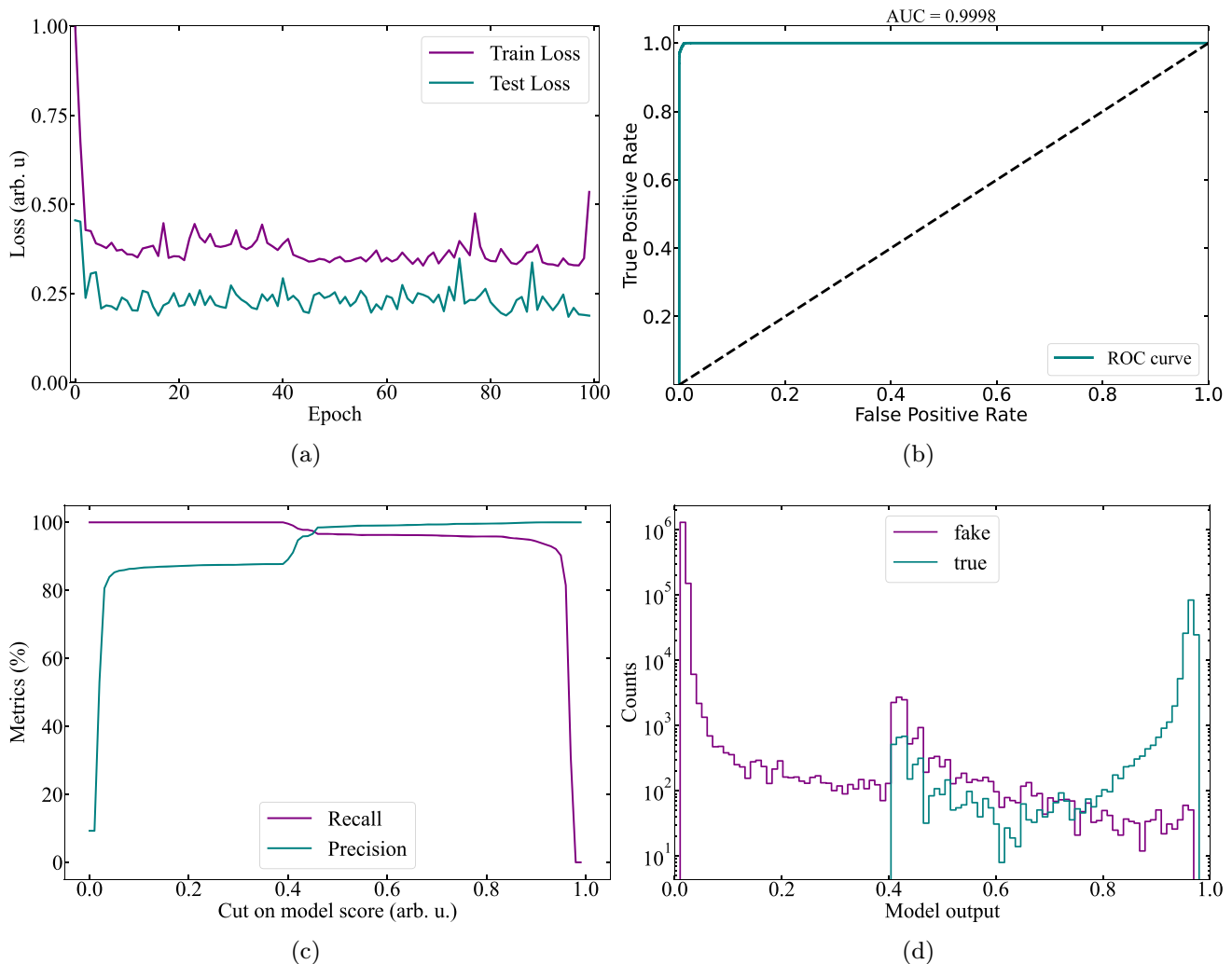
A model trained on time slices with a certain overlap parameter r is henceforth denoted as GANN r . Training starts from the dataset with $r = 2$ to obtain the initial model, GANN2, for hierarchical training. Then, GANN2 is trained on the dataset with $r = 3$ to produce GANN3, and so on. For this study, we trained four models with different r values using the SPD and TrackML datasets. The models are named using a prefix corresponding to their training dataset: ‘SPD-’ for models trained on the SPD dataset and ‘TrackML-’ for those trained on the TrackML dataset. One epoch of GANN r training, involving the processing of 800 time slices on an NVIDIA V100 GPU, takes approximately from 3.5 to 5.2 min for overlap parameter r values ranging from 2 to 5. Finally, we produced two models that perform edge classification on the SPD and TrackML datasets with parameters $\overline{N}_{ev} = 15, r = 5$, denoted as SPD-GANN5 and TrackML-GANN5, respectively.

The training results for the SPD-GANN5 model are presented in Fig. 6. The dependencies of the loss functions, evaluated for the training and test samples, on the number of epochs are shown in Fig. 6a. As seen, the losses are reduced by approximately half by the end of the training process. The Receiver Operating Characteristic (ROC) curve, with an Area Under the Curve (AUC) reaching 0.9998, is shown in Fig. 6b. Such a high AUC value and the almost triangular shape of the ROC curve indicate the high performance of the model in edge classification. Figure 6c presents the recall and precision as functions of the model score. The recall is defined as the ratio of the number of true edges with scores exceeding the threshold (cut on the model score) to the total number of true edges, whereas the precision is defined as the ratio of the number of true edges with scores exceeding the threshold to the total number of edges with scores exceed-

Fig. 5 Edge Classifier architecture

ing the threshold. These metrics achieve 97% at the model score threshold where they are equal. The distributions of true and false edge scores predicted by SPD-GANN5 are shown in Fig. 6d. The true edges exhibit a pronounced peak at high model scores (0.9–1.0), while the false edges dominate in the lower score range (0.0–0.1), with minimal overlap between two distributions. A sharp drop of the distribution of model

scores for true edges around 0.4 indicates that the model was successfully trained to avoid mistakes in true edges definition. This model feature becomes more pronounced during hierarchical training. Subsequently, the score threshold for the SPD-GANN5 model was set to 0.4, which allows us to retain nearly all true edges.

**Fig. 6** Training results for the SPD-GANN5 model: **a** loss function applied on the training and test samples; **b** ROC curve with the AUC value; **c** recall and precision of edge classification as functions of the cut value applied to the edge scores; **d** edge score distributions for true and false edges

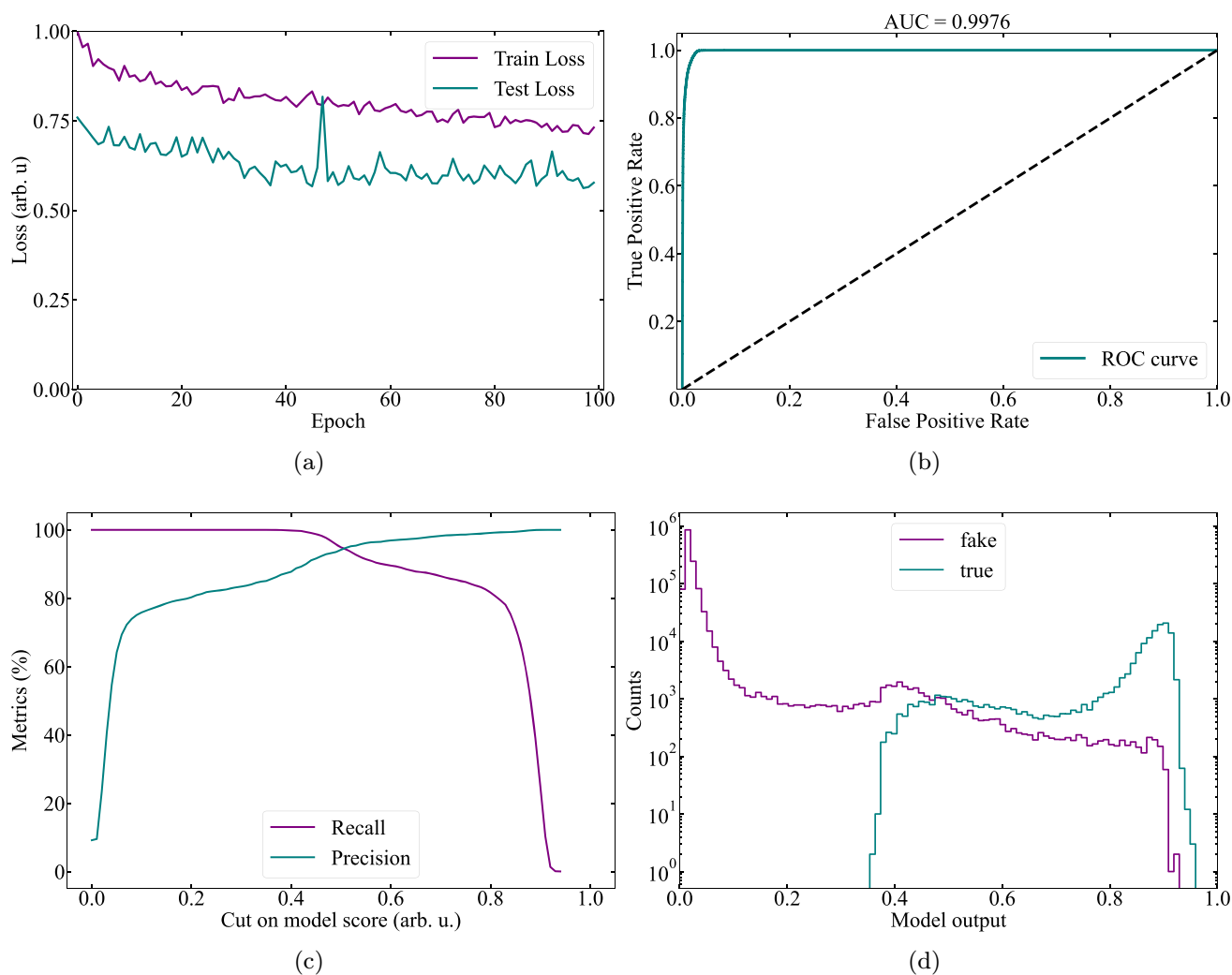


Fig. 7 Training results for the TrackML-GANN5 model: **a** loss function on the training and test samples; **b** ROC curve with the AUC value; **c** recall and precision of edge classification as functions of the cut value applied to the edge scores; **d** edge score distributions for true and false edges

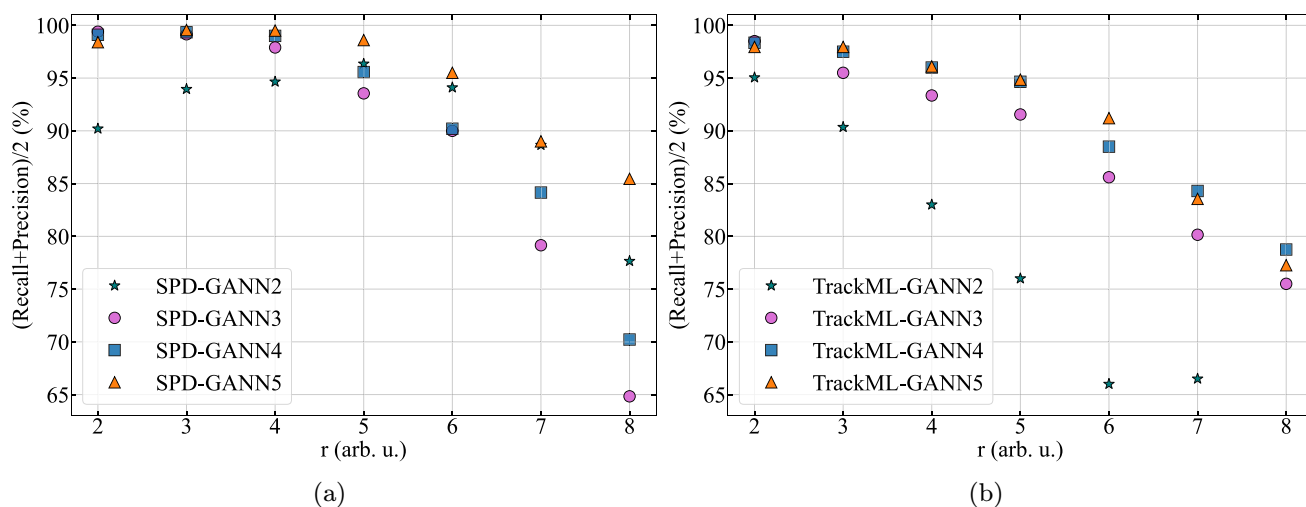


Fig. 8 Dependence of average classification metrics of the SPD-GANNr **(a)** and TrackML-GANNr **(b)** models on the overlap parameter r

To analyze the stability of the proposed approach, GANNr was trained from scratch on the TrackML dataset using the same hierarchical procedure. The resulting model, TrackML-GANN5, was then compared with the SPD-GANN5 model. The training results for TrackML-GANN5 are presented in Fig. 7. The dynamics of the training and test loss functions are shown in Fig. 7a. The losses are reduced by a factor of approximately 1.3 by the end of training. This drop is comparable in magnitude to that of SPD-GANN5 (see Fig. 6a). The ROC curve, with an AUC value reaching 0.9976, is shown in Fig. 7b. As seen, TrackML-GANN5 also exhibits a high AUC value and a ROC curve that approaches the optimal shape. The dependencies of the recall and precision on the model score are presented in Fig. 7c. These metrics achieve 94% at the point where they are equal. Figure 7d illustrates the edge score distributions for true and false edges predicted by TrackML-GANN5. Similar to the SPD case, the distribution for true edges shows a sharp drop around 0.35. This value was chosen as the score threshold for the TrackML-GANN5 model to ensure that all true edges are retained. Consequently, we observed the high performance of our model trained on the TrackML dataset, which contains more spiral tracks with non-equidistant steps between hits, demonstrating the stability of the proposed architecture.

The dependencies of the average metrics of the GANNr models on the overlap parameter r trained on both the SPD and TrackML datasets are presented in Fig. 8, where the advantages of the hierarchical training approach are shown. We additionally evaluated the average metrics at $r = 6, 7$, and 8 to analyze the stability of the models. As seen, the model performance strongly decreases with increasing the overlap parameter r . However, the point at which degradation starts can be shifted using the post-training model. A comparison of the training results clearly demonstrates that SPD-GANN5 achieves better stability when varying r than the TrackML-GANN5 model. In particular, it is a consequence of the complexity of TrackML tracks compared to SPD tracks. The edge purity, i.e., the average ratio of the number of true edges to the total number of edges, in the SPD data after processing by SPD-GANN5 is 91.4% at a model score threshold of 0.4, whereas the edge purity for the TrackML dataset after processing with TrackML-GANN5 is 85.1% at a model score threshold of 0.35.

To obtain final clustering results, it is necessary to eliminate a small number of false edges remaining between clusters after model processing and extract densely interconnected groups of nodes (tracks). We used a modified Louvain method [16] to remove retained false edges in the sparse supergraphs produced by the model. This approach is a precise method with no free parameters due to its iterative structure and strict convergence. However, for the same reasons, it is computationally expensive. Thus, using this method allows us to demonstrate the maximum performance of the proposed

approach. The final recall, precision, and accuracy metrics for particle track clustering by events, i.e., after applying the SPD-GANN5 model and the Louvain method, for SPD time slices at $r = 5$ were obtained as 95.6%, 94.7% and 99.3%, respectively.

6 Conclusions

A Graph Neural Network with an attention mechanism for the clustering of particle tracks by events in the SPD experiment of the NICA accelerator complex was presented. The proposed GNN-based method performs non-binary classification and demonstrates enhanced performance compared to the previous Siamese Neural Network, achieving 99% accuracy and 95% precision and recall in track clustering. The processing speed of the trained model on an Nvidia V100 Tesla GPU is 3.5 time slices per second at $\overline{N}_{ev} = 15$, $r = 5$. Our approach allows processing time slices with a random number of events. Hierarchical training enables learning the GANN to operate effectively at the temporal resolution expected in the experiment. Furthermore, the stability of the architecture was confirmed by testing on the TrackML dataset, showing that the model scales well to complex geometries with non-equidistant hit spacing. The approach presented in this study can be applied to any experiment where track classification by events is an offline post-processing task. The implementation is available in a GitHub repository [17].

Acknowledgements The authors would like to thank the HybriLIT group of JINR as all the computations were held on the basis of the HybriLIT heterogeneous computing platform (LIT, JINR).

Funding The authors received no financial support for the research of this article.

Data Availability Statement This manuscript has associated data in a data repository. [Author's comment: The datasets generated during the current study are available in the folders SPD_15Event and ML_15Event in repository [17]].

Code Availability Statement This manuscript has associated code/software in a data repository. [Author's comment: The software generated during the current study is available in the folder TrackGANN in repository [17]].

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes

were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Funded by SCOAP³.

References

1. V. Abazov et al., Technical design report of the spin physics detector at NICA. *Nat. Sci. Rev.* **1**, 1 (2024). [arXiv:2404.08317](https://arxiv.org/abs/2404.08317)
2. A. Arbuzov, A. Bacchetta, M. Butenschoen, F.G. Celiberto, U. D'Alesio, M. Deka, I. Denisenko, M. Echevarria, A. Efremov, N. Ivanov, A. Guskov, A. Karpishkov, Y. Klopov, B. Kniehl, A. Kotzinian, S. Kumano, J. Lansberg, K.-F. Liu, F. Murgia, O. Teryaev, On the physics potential to study the gluon content of proton and deuteron at NICA SPD. *Prog. Part. Nucl. Phys.* **119**, 103858 (2021). <https://doi.org/10.1016/j.ppnp.2021.103858>
3. S. Ondřej, M. Bodlák, V. Frolov, S. Huber, M. Jandek, V. Jarý, I. Konorov, A. Květoň, D. Levit, J. Nový, D. Steffen, J. Tomsa, M. Virius, The continuously running iFDAQ of the compass experiment. *EPJ Web Conf.* **214**, 01032 (2019). <https://doi.org/10.1051/epjconf/201921401032>
4. M. Borisov, P. Goncharov, G. Ososkov, D. Rusov, Unraveling time-slices of events in the SPD experiment. *Phys. Part. Nucl.* **55**, 453–455 (2024). <https://doi.org/10.1134/S1063779624030237>
5. D. Rusov, P. Goncharov, A. Zhemchugov, G. Ososkov, Deep tracking for the SPD experiment. *Phys. Part. Nucl. Lett.* **20**(5), 1180–1182 (2023). <https://doi.org/10.1134/S1547477123050655>
6. Y. Talochka, G. Ososkov, N. Voytishin, Graph neural network with attention and two-stage aggregation for particle track reconstruction in the TPC MPD of the NICA accelerator complex. *Comput. Softw. Big Sci.* **9**(1), 22 (2025). <https://doi.org/10.1007/s41781-025-00153-4>
7. D. Baranov, S. Mitsyn, P. Goncharov, G. Ososkov, The Particle Track Reconstruction based on deep Neural networks. *Eur. Phys. J. Web Conf. EDP* **214**, 06018 (2019). <https://doi.org/10.1051/epjconf/201921406018>. [arXiv:1812.03859](https://arxiv.org/abs/1812.03859)
8. T. Golling, S. Amrouche, M. Kiehn, P. Calafiura, S. Farrell, H. Gray, V. Estrade, C. Germain, V. Gligorov, I. Guyon, M. Hushchyn, V. Innocente, A. Salzburger, E. Moyse, D. Rousseau, Y. Yilmaz, J.-R. Vlimant, TrackML: a tracking machine learning challenge, in: *Proceedings of the 39th International Conference on High Energy Physics* (2019), p. 159. <https://doi.org/10.22323/1.340.0159>
9. R. Liu, P. Calafiura, S. Farrell, X. Ju, D.T. Murnane, T.M. Pham, Hierarchical Graph Neural Networks for Particle Track Reconstruction, in: *21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI meets Reality* (2023). [arXiv:2303.01640](https://arxiv.org/abs/2303.01640)
10. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, *arXiv e-prints* (2019). [arXiv:1912.01703](https://arxiv.org/abs/1912.01703). <https://doi.org/10.48550/arXiv.1912.01703>
11. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph Attention Networks, *arXiv e-prints* (2017). [arXiv:1710.10903](https://arxiv.org/abs/1710.10903). <https://doi.org/10.48550/arXiv.1710.10903>
12. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536 (1986). <https://doi.org/10.1038/323533a0>
13. K. Xu, W. Hu, J. Leskovec, S. Jegelka, How Powerful are Graph Neural Networks?, *arXiv e-prints* (2018). [arXiv:1810.00826](https://arxiv.org/abs/1810.00826). <https://doi.org/10.48550/arXiv.1810.00826>
14. J. Singh, C. Beeche, Z. Shi, O. Beale, B. Rosin, J. Leader, J. Pu, Batch-balanced focal loss: a hybrid solution to class imbalance in deep learning. *J. Med. Imaging* **10**(5), 051809 (2023). <https://doi.org/10.1117/1.JMI.10.5.051809>
15. I. Loshchilov, F. Hutter, Decoupled Weight Decay Regularization, *arXiv e-prints* (2017). [arXiv:1711.05101](https://arxiv.org/abs/1711.05101). <https://doi.org/10.48550/arXiv.1711.05101>
16. J. Zhang, J. Fei, X. Song, J. Feng, An improved Louvain algorithm for community detection. *Math. Probl. Eng.* **2021**, 1–14 (2021). <https://doi.org/10.1155/2021/1485592>
17. S. Omelianchuk, TrackGANN, <https://github.com/SavelyOm/TrackGANN>, gitHub repository (2026)