Trigger Detection for the sPHENIX Experiment via Bipartite Graph Networks with Set Transformer

Tingting Xuan[1], Giorgian Borca-Tasciuc[1], Yimin Zhu[1], Yu Sun[2], Cameron Dean[3], Zhaozhong Shi[3], and Dantong Yu[4]✉

[1] Stony Brook University, Stony Brook, New York, USA
{tingting.xuan,giorgian.borca-tasciuc,yimin.zhu}@stonybrook.edu
[2] Sunrise Technology Inc., Stony Brook, New York, USA yu.sun@sunriseaitech.com
[3] Los Alamos National Laboratory, New Mexico, USA
{ctdean,zhaozhongshi}@lanl.gov
[4] New Jersey Institute of Technology, Newark, New Jersey, USA
dantong.yu@njit.edu

Abstract. Trigger (interesting events) detection is crucial to high-energy and nuclear physics experiments because it improves data acquisition efficiency. It also plays a vital role in facilitating the downstream offline data analysis process. The sPHENIX detector, located at the Relativistic Heavy Ion Collider in Brookhaven National Laboratory, is one of the largest nuclear physics experiments on a world scale and is optimized to detect physics processes involving charm and beauty quarks. These particles are produced in collisions involving two proton beams, two gold nuclei beams, or a combination of the two and give critical insights into the formation of the early universe. This paper presents a model architecture for trigger detection with geometric information from two fast silicon detectors. Transverse momentum is introduced as an intermediate feature from physics heuristics. We also prove its importance through our training experiments. Each event consists of tracks and can be viewed as a graph. A bipartite graph neural network is integrated with the attention mechanism to design a binary classification model. Compared with the state-of-the-art algorithm for trigger detection, our model is parsimonious and increases the accuracy and the AUC score by more than 15%.

Keywords: Graph Neural Networks · Event Detection · Physics-Aware Machine Learning.

1 Introduction

sPHENIX is a high-energy nuclear physics experiment under construction at Brookhaven National Laboratory and situated on the Relativistic Heavy Ion Collider (RHIC). The goal of sPHENIX is to probe the initial moments after the Big Bang by studying quark-gluon plasma, a state of matter where atomic nuclei melt under extremely hot and dense conditions.

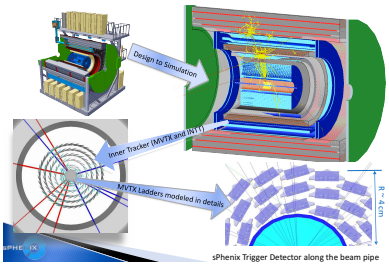

Fig. 1: sPHENIX Track Detector and Trigger Design.

The sPHENIX detector shown in Figure 1 consists of several subdetectors for collecting a wide range of patterns of physics events. The two subsystems closest to the collision point are of most interest to this study. They are the MAPS vertex detector (MVTX) and intermediate tracker (INTT). The MVTX detector provides vertexing and tracking with high precision, while the INTT provides tracking with a resolution capable of determining the individual beam crossings at RHIC. sPHENIX also uses an outer calorimetry system to measure the energy of particles in the detector at a low speed of 15 kHz, limited by the readout electronics. As the collision rate for protons at RHIC is approximately 2 MHz, the calorimeter system does not work with the online setting and is not considered in this paper.

Heavy Flavor decays that we attempt to detect exhibit several prominent characteristics with a wide value range that overlaps with background events. The complex pattern and non-trivial decision boundary between heavy flavor delay and background events provide an ideal playground to apply ML techniques. The particles of interest decay on short time scales, typically a few nanoseconds or less. These particles may travel several millimeters at the speed of light before they decay. Physicists often extrapolate Particle tracks in the detector space to determine whether the tracks coincide with the beam collision point.

A tremendous volume of data is produced during collider experiments, but only a tiny fraction of the data needs to be selected due to the rarity of the targeted events. For example, an event that includes a charm quark typically occurs once for every 50 background events [1,21] while a beauty quark occurs once for roughly 1000 background events [2]. Collider experiments require a trigger system to reduce data in real-time and resolve the big data problem that is impractical for any data processing facility [11]. The triggers make decisions to keep or discard an event in situ and significantly reduce the data volume that needs to be retained for physics experiments. Our paper brings forth significant impacts to physics experiments by shifting many offline analysis tasks into an online setting and significantly shortening the latency between experiment and scientific discovery.

Before experiments start, we rely on simulation data to train and test our model. The simulated data is expected to match real data at a high level and has been extensively validated from previous physics experiments.

Contributions

- In this paper, we design a highly effective graph pooling/distributing mechanism for graph-level classification and prediction.
- Our model does not demand any pre-existing graph topology. Instead, it employs a set transformer to combine local and global features into each network node, enables effective knowledge exchange among network nodes, and supports local and global-scale graph learning.
- We incorporate well-known physics analysis into the multi-task neural network architecture and explicitly inference a crucial physics property in nuclear physics experiments, i.e., transverse momentum. This physics-driven learning improves our model accuracy and AUC score (Area Under Curve) by about 15%.

2 Related Work

The domain of experimental physics has a history of utilizing Machine Learning (ML) in physics-related tasks like particle identification, event selection, and reconstruction. Neural Networks have been used in these experiments [3,35,16] at first and were replaced by Boosted Decision Trees [40]. Recently Neural Networks and their modern implementation of Deep Learning (DL) regain their popularity in physics because of their superior ability to automatically learn effective features for many tasks and their outstanding performance [26].

Convolutional Neural Network (CNN) [25] is the most commonly used architecture with DL in the field of particle physics. It has proven its success in many tasks such as jet identification [9,20,23], particle identification [12,19,24], energy regression [12,15,36], and fast simulation [12,31,33]. The majority of CNN architecture models take a fixed-grid input tensor to represent the detector of an array of sensors. More efforts attempt to explore other alternative DL architectures for a better representation of physics: Recurrent Neural Network [14], recursive networks [28], Graph Neural Network (GNN) [37], and DeepSets [24] for the particle jet identification tasks. GNN is also applied to the classification tasks in neutrino physics [13]. Garnet [36] is distance-weighted graph network that can efficiently detect irregular pattern of sparse data. Transformer model architecture [43] shows its success in many applications [10,17,39]. Taking the data cloud as a set, Set Transformer [27] utilizes attention mechanism to learn interactions between elements with EdgeConv that is permutation equivariant and fits the set property.

Our previous works using ML to solve the same trigger detection problem appear in [45,44]. Beyond these efforts, no other studies are addressing the same issue. Jet taggers address a similar situation of tagging tracks with particles. Nevertheless, Jet taggers rely on the different physics properties collected from

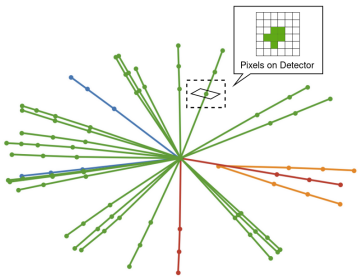

Fig. 2: An Example of Trigger Event. In trigger events, particles decay into two or more different particles soon after the collision. Lines represent the trajectories of the particles. Green ones come from the center of the collision. Blue, red, and orange tracks start from the position where the decay happens.

calorimeter detectors and they are interested in different target particles. Jet tagger belongs to offline analysis, and the readout speed from the calorimeter is much slower than the event rate, invalidating their applications in the online use case. Several ML methods for top tagging are discussed in [22]. We gain insights from the model design of existing tagging algorithms, especially those supporting the particle cloud [36,37,38]. Our final goal is to apply the algorithm to an online data-driven trigger system in an end-to-end fashion [29,44].

3 Problem Definition

Figure 2 schematically illustrates the trigger problem. The input of the physics event consists of a set of tracks and is represented as a matrix $X \in \mathcal{R}^{n \times d}$, where n is the total number of tracks, and d is the dimensionality of the features of each track. Tracks are treated as vertices in a graph. The goal is to determine whether this graph corresponds to a trigger event and triggers the data acquisition system to retain the readouts from the entire detector for future studies.

The commonly adopted GNN-based trigger prediction model attempts to perform end-to-end prediction from the raw hits that are the coordinates of the detector pixels where a particle traverses the detector. This domain-agnostic approach does not offer any insight into why an event becomes a trigger and results in inferior performance. Domain scientists require physics-aware reasoning and interpretation by explicitly incorporating physics models and properties. Since collecting advanced physics properties by detector requires sophisticated detectors and incurs considerable latency compared to the fast geometric detector, it is not feasible to use advanced physics properties in an online data acquisition environment. To incorporate the advanced physics properties, we must regress them onto the available geometric data. Our ultimate goal is to predict triggers while retaining the interpretability and rationality of intermediate tasks by replicating offline physics analysis workflows.

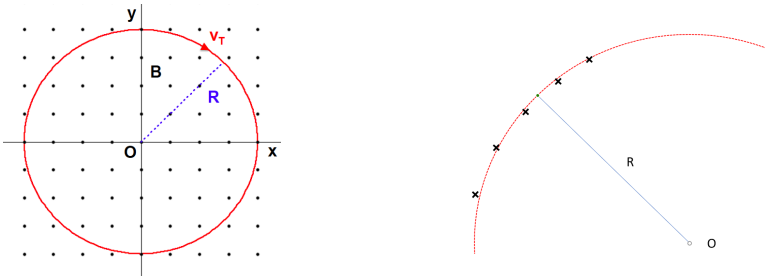

Fig. 3: The left figure shows that a positively charged particle will undergo a circular motion clockwise with a radius R in the uniform magnetic field B along the $+z$ direction. The right figure shows an example track with a fitted circle. The black cross markers represent five hits on the example track; the red dashed curve approximate a particle track and is the fitted circle with a radius R.

4 Transverse Momentum Estimation

Transverse momentum, as part of the kinematics of particles, is crucial for studying particle dynamics in high-energy and nuclear physics experiments. The transverse momentum of a charged particle can be estimated with the knowledge of the magnetic field where it travels and the radius of its curved path in the magnetic field. The magnetic field in the sPhenix detector is fixed and can be precomputed; once we redefine the triggering task on the graph of tracks instead of hits, this transverse momentum is accessible for individual tracks and correlated with the radius of the particle's curved path (tracks) calculated using the geometry information of the hits on the track.

4.1 Physics Relation between Transverse Momentum and Track Curvature

In particle collider experiments, we typically choose the cylindrical coordinates to describe the particle momentum $\vec{p} = (p_T, p_z)$ for simplicity. It is conventional to choose the beam direction as the z-axis. Here, p_T is called the transverse momentum, an analog of track radius R in the cylindrical coordinates in the transverse direction. p_z is called the longitudinal momentum, an analog of z in the cylindrical coordinates in the transverse direction.

The sPHENIX experiment uses a solenoid magnet with the field aligning with the beam direction in the z-axis. The left figure in Figure 3 shows a positively charged particle moving clockwise under a magnetic field.

For a charged particle with charge q traveling across the magnetic field, a Lorentz force acts on the charged particle. In our case, B, defined as the magnetic field strength of the sPHENIX solenoidial magnet, is along the z direction. The Lorentz force maximizes in the transverse direction. The velocity vector $\vec{v}$ is decomposed like the momentum $\vec{v} = (v_T, v_z)$. When the charge of a particle is e,

combined with the equation for circular motion, the magnitude of the Lorentz force that points radially inward, is given by:

$$F = m\frac{v_T^2}{R} = ev_T B.$$

The momentum is given by $p_T = mv_T$. Hence, we get

$$p_T = eBR. \tag{1}$$

All particles tracked in the detector have a unit charge of one electron volt (eV) and so, if we change units so that the momentum is measured in GeV/c where c is the speed of light, then equation 1 becomes

$$p_T = 0.3BR, \tag{2}$$

where the magnetic field is measured in Tesla (T). The detailed derivation is beyond this paper. Equation 2 shows that the crucial physics property of particle momentum is proportional to the track radius and guides us to integrate this physics insight into the ML-based detection.

4.2 Track Curvature Fitting

A track consists of a list of hits tracking particles traversing detector layers. The transverse momentum is highly correlated with the detected curvature of particle tracks. We fit a circle to those hits to approximate the momentum and calculate its radius. Here, regarding the direction of the magnetic field, we only need to consider the x-axis and y-axis. The image on the right in Figure 3 shows an example track with its fitted circle.

A circle is represented by the following formula: $x^2+y^2+\beta_1 x+\beta_2 y+\beta_3 = 0$. Given a track of k_T hits $T = \{(x_1, y_1), (x_2, y_2), ..., (x_{k_T}, y_{k_T})\}$, we define a linear system that consists of k_T equations for these hits and attempt to derive the circle's coefficients $\beta = [\beta_1, \beta_2, \beta_3]^T$. To get the best circle approximation, we use the least-squares (LS) optimization to solve the linear regression equation and extract the β coefficients:

$$\beta = (A^T A)^{-1} A^T B.$$

Here $A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ ... & & \\ x_{k_T} & y_{k_T} & 1 \end{bmatrix}$, $B = [-x_1^2 - y_1^2, -x_2^2 - y_2^2, ..., -x_{k_T}^2 - y_{k_T}^2]^T$. With the optimized coefficients for the fitted circle, the circle radius is as follows:

$$R = \frac{1}{2}\sqrt{\beta_1^2 + \beta_2^2 - 4\beta_3}. \tag{3}$$

4.3 Momentum Estimation

The momentum can be calculated by Equation 2 using the estimated radius. However, this has the drawback that at least three hits are required for the track to estimate the radius. We propose the second method based on a feed-forward (FF) neural network to predict the transverse momentum of the given track and its LS-estimated radius R. We compare the estimation accuracy in Section 6.2 and evavalue their effect on trigger detection in Section 6.3.

5 Bipartite Graph Networks with Set Transformer for Trigger Detection

To resolve the trigger detection problem, we attempt to label events based on their entire tracks. The previous work [45] builds an affinity matrix of tracks and forms a graph where each node represents a track. Given a track graph, our algorithm first applies GNN to learn local embeddings and uses various pooling methods to aggregate and label the event. In this paper, we discard the common practice of building the fine-scale affinity matrix (graph) among hits and directly apply physical analysis to guide our neural network architecture. Determining whether an event is a trigger event involves three types of interactions:

1. Local track-to-track interactions, such as determining whether two tracks share the same origin vertex.
2. Track-to-global interactions, such as determining the collision vertex of the event and potential secondary vertices of decaying.
3. Global-to-track interactions, such as comparing each track's origin with the collision vertex of the event.

To incorporate various interactions, our neural network model uses set attention mechanisms to facilitate local track-to-track information flow, accumulate track information into aggregators to facilitate local track-to-global information flow, and updating the track embeddings based on the aggregators to facilitate global-to-local information flow. Thereby, the bipartite GNN architecture performs the local track-to-global and global-to-local information flow.

Our model consists of several Set Encoder with Bipartite Aggregator (SEBA) Blocks. The SEBA Blocks update our track embeddings and exchange information between tracks. SEBA Blocks contain a Set Attention Block, a Bipartite Aggregation module, and a Feed-Forward (FF) network for transformation, as shown in Figure 5. After several SEBA Blocks, we use some aggregation functions as the readout functions to obtain the global representation for the whole set and feed the representation into a FF network to get the final output. Figure 4 shows the entire architecture of our model.

5.1 Set Attention Blocks

We use the Set Attention Blocks designed by Lee in [27]. The module applies a self-attention mechanism to every element in the input set to enable the model

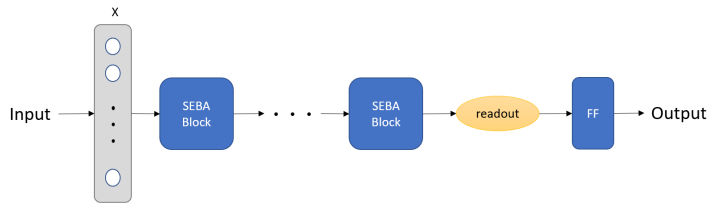

Fig. 4: Bipartite Graph Networks with Set Transformer Model Architectures.

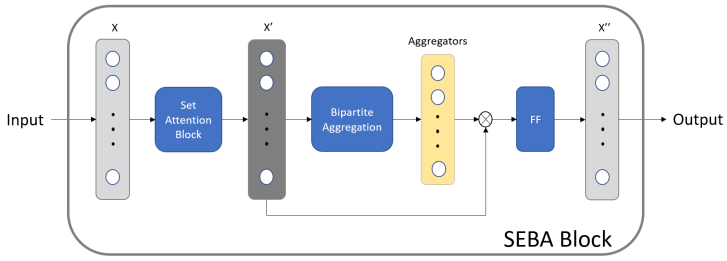

Fig. 5: Set Encoder with Bipartite Aggregator (SEBA) Blocks.

to encode pairwise- or higher-order interactions between the elements in the set. From the physics perspective, physicists often need to analyze the interactions between tracks and which tracks are from the same origin points. The self-attention mechanism neatly follows this physics practice.

Set Attention Blocks use Multi-head attention that Vaswani introduced originally in [43]. It packs a set of queries into a matrix Q. The keys and values are also stacked into matrices K and V.

For a single attention function, We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head attention projects each of the Q, K, V matrices onto h different matrices separately, and applies an attention function to each of these h projections. The multiple heads allow the model to jointly attend to information from different representations subspaces at different positions.

$$\text{Multihead}(Q, K, V) = \text{Concat}(O_1, ..., O_h)W^O,$$

where $O_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and W_i^Q, W_i^K, W_i^V, W^O are learnable parameters.

Given the matrix $X \in \mathcal{R}^{n \times d}$ which represents a set of d-dimensional vectors. Set Attention Blocks applies a multi-head attention function with $Q = K =$

$V = X$ and serves as a mapping function using the following equation:

$$SAB(X) = \text{LayerNorm}(H + \text{rFF}(H)),$$

where $H = \text{LayerNorm}(X + \text{Multihead}(X, X, X))$, rFF is a row-wise FF layer applied to the input matrix and LayerNorm is the layer normalization [8].

5.2 Bipartite Aggregation

The idea of bipartite aggregation comes from GarNet architecture in [36]. GarNet is a distance-weighted graph network that partitions nodes into two groups: regular nodes of input elements and k aggregators, with both groups sharing the same node space. The original GarNet uses FF networks to measure the 'distance' between the input elements and these aggregators and aggregates information from elements to aggregators using a distance-weighted potential function. Instead of using the potential function and physical positions to interpret the relationship between track nodes and aggregators, we use the neural network to learn the affinity scores dynamically among network nodes.

The trigger decision is a graph-level prediction and requires local and global pooling. Experiments in [30] show that sophisticated pooling algorithms have no significant advantages over simple global pooling. The problem arises from the one-way rigid pooling mechanism that aggregates limited, sometimes biased information from the lower layer to the higher layer while ignoring the reverse information flow for distributing the aggregator information back to the low-level graph nodes for adjustment and adaptation. We propose an iterative message passing between two sides of networks, i.e., two-way gathering (from track nodes to aggregators)/scattering (from aggregators to track nodes) operations to resolve the missing link and inflexibility caused by the standard pooling algorithm.

Figure 6 shows a complete bipartite graph that consists of two type of nodes: track nodes $X = \{\vec{x}_1, ..., \vec{x}_n\}$ and aggregators $A = \{\vec{a}_1, ..., \vec{a}_k\}$. Our track graph uses aggregators to gather/disseminate information, where each track node is connected to each aggregator. The aggregators in our problem setting are closely related to two physics notions: the primary vertex where a physics event of collision happens and the secondary vertices where particles decay.

Each aggregator gathers information from all elements via the edges. The embedding of the edge between the ith element and the jth aggregator is

$$\vec{e}_{ij} = s_{ij}\vec{x}_i, \text{where } S = \sigma(\text{FF}_{\text{agg}}(X))$$

In the above equation, $E = \{e_{ij}\} \in \mathcal{R}^{n \times k \times d}$ represents the message from track nodes to aggregators, and $S = \{s_{ij}\} \in \mathcal{R}_{+}^{n \times k}$ is the score matrix between track nodes and aggregations nodes. FF_{agg} is a feed-forward neural network that assigns k weights to each element in X. The FF_{agg} acts as a gate function to the k aggregators, with each output weigh s_{ij} determining node i's contribution to the j-th aggregators, and σ is an activation function. After all information flows through the neural network gates in FF_{agg}, each aggregator performs a readout

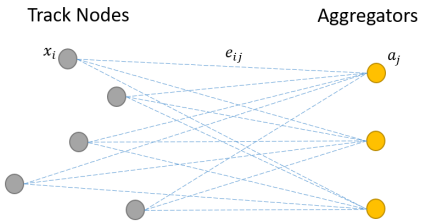

Fig. 6: Bipartite Graph formed by Elements and Aggregators.

function on its gathered node information from tracks. We define the aggregator function as follows:

$$A = \text{readout}(E), \text{where } A \in \mathcal{R}^{k \times 2d}.$$

In our model, we concatenate the result of the mean and maximum readout functions and assign them to each aggregator.

Given the fixed number of aggregators, once we aggregate the information from elements to aggregators, we treat the k aggregators as a global graph embedding vector $g \in R^{k \times 2d}$. Then we concatenate this global vector with each track node vector to implement the by-pass connection in Figure 5 and then apply another feed-forward network to update each node vector. This by-pass design allows us to scale the network stack into deep layers for complex event detections.

$$x_i' = \text{FF}_{\text{Node}}(\text{Concat}(x_i, g))$$

The bipartite aggregators first extract the local information to a global vector. Then the global information is fed back to each element by concatenation. It generates an information cycle from local to global, then back to local. Considering also the Set Attention Block that allows the pairwise information exchange between elements, by repeating our SEBA block several times, the intra-set relationship is well explored by our model.

6 Experiment Results

6.1 Dataset and Experiment Settings

Our experiment utilizes a simulation dataset[5]. We simulate the physics process with the PYTHIA8.3 package [41] and the GEANT4 simulation toolkit [5,7]. PYTHIA8 is a software package simulating QCD processes in small collisions systems and has been widely validated at many colliders [6,32,42]. GEANT4 is a package that simulates the passage of particles and radiation through matter.

[5] Some example data files can be found at https://github.com/sPHENIX-Collaboration/HFMLTrigger.

Table 1: Performance of Momentum Regression Models. MLP-n indicates an MLP with 4 layers and hidden dimensions of size n. Denote Pearson's R as P's R and Spearman's R as S's R.

	All tracks			Tracks with at least 3 hits		
Method	R^2	P's R	S's R	R^2	P's R	S's R
LS	-116.18	0.0997	0.9376	-117.40	0.0999	0.9618
MLP-8	0.9071	0.9524	0.9534	0.9071	0.9524	0.9534
MLP-256	0.9100	0.9540	0.9564	0.9100	0.9540	0.9564

For this project, only the three-layer MVTX and two-layer INTT detectors are simulated because they are the only detectors capable of operation fast enough to achieve the goal of extremely high speed online data analysis and decision making. It has been extensively validated from previous physics experiments that a high-level agreement exists between the simulation data and real data [4]. Our model will be used to filter data in the real experiment that will start taking data in February 2023. The simulation data supporting the findings of this study are available from the corresponding author upon reasonable request[6].

The input vector for each track consists of coordinates of five hits in each detector layer, the length of each track segment, the angle between segments sequentially, and the total length of the track. The number of tracks per event varies from several to dozens. The coordinates of the geometric center of all the hits in the graph are calculated as complementary features to ease the downstream learning task. We also use the LS-estimated radius as another feature. All of the experiments, unless otherwise noted, use $1,000,000$ training samples, $400,000$ validation samples, and $400,000$ test samples. All models use the same pre-split training, validation, and test data sets, to ensure no information leakage and fair comparison. We adopt the Adam optimizer with a decayed learning rate from $1e-4$ to $1e-5$ in 50 epochs for all the training experiments. Experiments are run on various GPU architectures, including NVIDIA Titan RTX, A5000, and A6000. All baselines and our model are implemented using PyTorch [34] and PyTorch Geometric [18]. The code is publicly available on GitHub. [7].

6.2 Transverse Momentum Estimation

We compare two methods for estimating the transverse momentum. The transverse momentum p_T is linearly proportional to the radius of a track R, as shown in Equation 2. Here, we use a constant magnetic field with $B = 1.4$ T in our dataset settings. We estimate the radius R using the LS fitting described in Section 4. If there are not enough hits to estimate the radius using LS fitting, we set the estimated radius to be zero.

[6] Please contact yu.sun@sunriseaitech.com for data access.

[7] https://github.com/Sunrise-AI-Tech/ECML2022-TriggerDetectionForTheSPHENIXExperimentViaBipartiteGraphNetworksWithSetTransformer

Table 2: Comparison to Baseline Models with Estimated Radius.

	with LS-radius			without radius		
Model	#Parameters	Accuracy	AUC	#Parameters	Accuracy	AUC
Set Transformer	300,802	84.17%	90.61%	300,418	69.80%	76.25%
GarNet	284,210	90.14%	96.56%	284,066	75.06%	82.03%
PN+SAGPool	780,934	86.25%	92.91%	780,678	69.22%	77.18%
BGN-ST	355,042	**92.18%**	**97.68%**	354,786	**76.45%**	**83.61%**

We make several observations on the p_T estimation results in Table 1. The LS estimated p_T achieves a high degree of correlation with the true p_T as measured by Spearman's R. However, the low value of Pearson's R and the highly negative coefficient value of determination indicate a poor linear correlation between the two. We hypothesize that this is due to outliers because some tracks occasionally produce an impossibly large estimated p_T. Using an MLP on the track to refine the LS p_T results seems to be highly effective, with all three correlation coefficients indicating that the models are highly predictive of the true p_T even with a small neural network. Noticeably, however, for tracks with at least three hits, the LS method outperforms the MLP method for Spearman's R. This might explain similar performance on triggering when using the LS-estimated p_T and the MLP-estimated p_T shown in Table 3.

6.3 Trigger Detection

Baselines. We compare our model with the ParticleNet(PN)+SAGPool method proposed in [45]. We also use Set Transformer and GarNet as our baselines because they are also well-suited to the problem of classifying a set of tracks. For Set Transformer, we use hidden dimension of 128 and four attention heads. For GarNet, we set hidden dimension of 64 and sixteen aggregators. For BGN-ST, we also use hidden dimension of 64 and sixteen aggregators. We use two-layer neural network architecture for all three models. The PN+SAGPool model has two stages. The first stage uses PN to generate an affinity matrix and track embeddings. Three edge-convolutional layers are used, with the hidden dimensions of 64, 128, and 64, respectively. All three edge-convolutional layers use 15 nearest neighbors when updating the node embeddings. The second stage includes the Sagpool layer aggregating the embeddings to perform trigger prediction. Sagpool uses three hierarchical pooling layers with a pooling ratio of 0.75.

Table 2 compares the performance between BGN-ST and the baseline models. From Table 2, we observe that BGN-ST outperforms all other methods by a significant margin. It usually takes no more than one day to train the Set Transformer, GarNet and BGN-ST on a single GPU card. The baseline PN+SAGPool has more parameters than ours. It takes two to three days to train the baseline model PN+SAGPool.

Effect of Transverse Momentum Estimation. Table 2 shows the performance comparison between different models with or without radius. From the table, we

Table 3: Comparison of BGN-ST with LS-Estimated Radius and MLP-Refined Radius. A three-layer model with sixteen aggregators is used in the experiment.

	LS		MLP	
Hidden dim	Accuracy	AUC	Accuracy	AUC
32	91.52%	97.33%	91.48%	97.31%
64	92.18%	97.68%	92.23%	97.73%
128	**92.44%**	**97.82%**	**92.49%**	**97.86%**

Table 4: Hyperparameter Grid Search for BGN-ST

Hyperparameter	Range
Hidden dim	32, 64, 128, 256
#Aggregators	8, 16, 32
#Layers	2, 3
Activations	ReLU, Tanh, Potential, Softmax

observe both accuracy and AUC jump by 15% when the radius is added to all these four models under the same model setting.

Effect of Refining Transverse Momentum Estimate with an MLP. From Table 3, it is clear that further refining the momentum with an MLP trained to predict the momentum from the track and the LS-estimated radius does not yield any tangible improvement in the model performance. This applies to both the smaller and larger models.

Ablation Study of Hyperparameters. We perform a grid search on hyperparameters in Table 4 to find the best setting for trigger detection with BGN-ST.

We compare activation functions for aggregators in Table 4 in a two-layer BGN-ST with 64 hidden dimensions and 16 aggregators. The table shows that Softmax has the highest accuracy and AUC score among all choices.

We also undertook some other ablation studies. Figure 7 shows the accuracy comparison for different hidden dimensions, the number of aggregators, and layers. A larger model tends to perform better, but the number of parameters also increases exponentially. Our best performance is using a three-layer model with 256 hidden dimensions and 32 aggregators. The best accuracy for the test dataset is 92.52%, and the best AUC score is 97.86%.

7 Conclusions

This paper details a novel Bipartite GNN architecture with a set transformer that uses the set attention mechanism to enhance the tracking with event features and ease the modeling of particle interactions in physics. Our model architecture benefits the pairwise interactions between tracks and allows a two-way scattering and gathering for effective information exchange and adaptive graph pooling.

Table 5: Ablation Study of Activations

Activation	Accuracy	AUC
ReLU	90.74%	96.87%
Tanh	90.19%	96.58%
Potential	90.41%	96.75%
Softmax	**92.18**%	**97.68**%

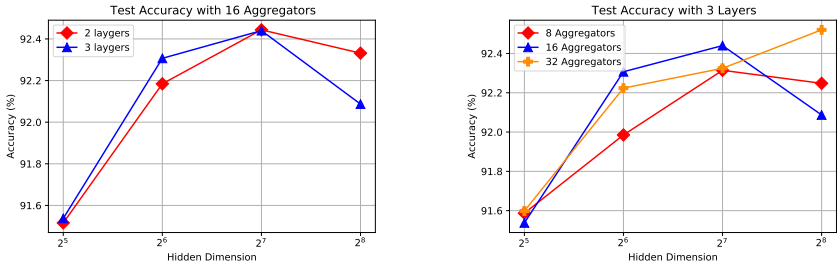

Fig. 7: Accuracy performance in respect to hidden dimension for two/three-layer models and different number of aggregators.

We empirically validate that BGN-ST outperforms all selected state-of-the-art methods. The paper adopts the physics-aware concept and introduces explicit physics properties such as transverse momentum. As a result, we improve the model accuracy and AUC score by about 15%.

References

1. Aaij, R., et al.: Prompt charm production in pp collisions at $\sqrt{s} = 7$ TeV. Nuclear Physics B **871**(1), 1–20 (2013)
2. Aaij, R., et al.: Measurement of the b-quark production cross section in 7 and 13 TeV pp collisions. Physical review letters **118**(5), 052002 (2017)
3. Abreu, P., Adam, W., Adye, T., Agasi, E., Alekseev, G., Algeri, A., Allen, P., Almehed, S., Alvsvaag, S., Amaldi, U., et al.: Classification of the hadronic decays of the Z0 into b- and c-quark pairs using a Neural Network. Physics Letters B **295**(3-4), 383–395 (1992)
4. Adare, A., Afanasiev, S., Aidala, C., Ajitanand, N., Akiba, Y., Akimoto, R., Alexander, J., Aoki, K., Apadula, N., Asano, H., et al.: An upgrade proposal from the phenix collaboration. arXiv preprint arXiv:1501.06197 (2015)
5. Agostinelli, S., et al.: Geant4: A simulation toolkit. Nucl. Instrum. Meth. **A506**, 250 (2003)
6. Aguilar, M.R., Chang, Z., Elayavalli, R.K., Fatemi, R., He, Y., Ji, Y., Kalinkin, D., Kelsey, M., Mooney, I., Verkest, V.: Pythia 8 underlying event tune for rhic energies. Physical Review D **105**(1) (2022)
7. Allison, J., Amako, K., Apostolakis, J., Araujo, H., Dubois, P., et al.: Geant4 developments and applications. IEEE Trans.Nucl.Sci. **53**, 270 (2006)
8. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)

9. Baldi, P., Bauer, K., Eng, C., Sadowski, P., Whiteson, D.: Jet substructure classification in high-energy physics with deep neural networks. Physical Review D **93**(9), 094034 (2016)
10. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
11. Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., Zdeborová, L.: Machine learning and the physical sciences. Reviews of Modern Physics **91**(4), 045002 (2019)
12. Carminati, F., Khattak, G., Pierini, M., Vallecorsafa, S., Farbin, A., Hooberman, B., Wei, W., Zhang, M., Pacela, B., Vitorial, M.S., et al.: Calorimetry with deep learning: particle classification, energy regression, and simulation for high-energy physics. In: Workshop on deep learning for physical sciences (DLPS 2017), NIPS
13. Choma, N., Monti, F., Gerhardt, L., Palczewski, T., Ronaghi, Z., Prabhat, P., Bhimji, W., Bronstein, M.M., Klein, S.R., Bruna, J.: Graph Neural Networks for IceCube signal classification. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 386–391. IEEE (2018)
14. collaboration, C., et al.: CMS Phase 1 heavy flavour identification performance and developments. CMS Detector Performance Summary CMS-DP-2017-013 (2017)
15. De Oliveira, L., Nachman, B., Paganini, M.: Electromagnetic showers beyond shower shapes. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **951**, 162879
16. Denby, B.: Neural networks and cellular automata in experimental high energy physics. Computer Physics Communications **49**(3), 429–448 (1988)
17. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
18. Fey, M., Lenssen, J.E.: Fast Graph Representation Learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds
19. Guest, D., Cranmer, K., Whiteson, D.: Deep learning and its application to LHC physics. Annual Review of Nuclear and Particle Science **68**, 161–181 (2018)
20. Guest, D., Collado, J., Baldi, P., Hsu, S.C., Urban, G., Whiteson, D.: Jet flavor classification in high-energy physics with deep neural networks. Physical Review D **94**(11), 112002 (2016)
21. Jackson, P., et al.: Measurement of the total cross section from elastic scattering in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector. Physics Letters. Section B: Nuclear, Elementary Particle and High-Energy Physics **761**, 158–178 (2016)
22. Kasieczka, G., Plehn, T., Butter, A., Cranmer, K., Debnath, D., Dillon, B.M., Fairbairn, M., Faroughy, D.A., Fedorko, W., Gay, C., et al.: The machine learning landscape of top taggers. SciPost Physics **7**(1), 014 (2019)
23. Komiske, P.T., Metodiev, E.M., Schwartz, M.D.: Deep learning in color: towards automated quark/gluon jet discrimination. Journal of High Energy Physics **2017**(1), 1–23 (2017)
24. Komiske, P.T., Metodiev, E.M., Thaler, J.: Energy flow networks: deep sets for particle jets. Journal of High Energy Physics **2019**(1), 1–46 (2019)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**
26. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444
27. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: International Conference on Machine Learning. pp. 3744–3753. PMLR (2019)

28. Louppe, G., Cho, K., Becot, C., Cranmer, K.: QCD-aware recursive neural networks for jet physics. Journal of High Energy Physics **2019**(1), 1–23 (2019)
29. Mahesh, C., Dona, K., Miller, D.W., Chen, Y.: Towards an interpretable data-driven trigger system for high-throughput physics facilities. arXiv preprint arXiv:2104.06622 (2021)
30. Mesquita, D., Souza, A., Kaski, S.: Rethinking pooling in graph neural networks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 2220–2231 (2020)
31. de Oliveira, L., Paganini, M., Nachman, B.: Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. Computing and Software for Big Science **1**(1), 1–24 (2017)
32. P. Skands, S. Carrazza, J.R.: Tuning PYTHIA 8.1: the Monash 2013 Tune. The European Physical Journal C **74**(3024) (2014)
33. Paganini, M., de Oliveira, L., Nachman, B.: CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with Generative Adversarial Networks. Physical Review D **97**(1), 014021 (2018)
34. Paszke, A., et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035 (2019)
35. Peterson, C.: Track finding with neural networks. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **279**(3), 537–545 (1989)
36. Qasim, S.R., Kieseler, J., Iiyama, Y., Pierini, M.: Learning representations of irregular particle-detector geometry with distance-weighted graph networks. The European Physical Journal C **79**(7), 1–11 (2019)
37. Qu, H., Gouskos, L.: Jet tagging via particle clouds. Physical Review D **101**(5), 056019 (2020)
38. Qu, H., Li, C., Qian, S.: Particle transformer for jet tagging. arXiv preprint arXiv:2202.03772 (2022)
39. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019)
40. Roe, B.P., Yang, H.J., Zhu, J., Liu, Y., Stancu, I., McGregor, G.: Boosted decision trees as an alternative to artificial neural networks for particle identification. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **543**(2-3), 577–584 (2005)
41. Sjöstrand, T., Mrenna, S., Skands, P.: A brief introduction to PYTHIA 8.1. Computer Physics Communications **178**(11), 852–867 (June 2008)
42. Skands, P.: Tuning Monte Carlo generators: The Perugia tunes. Physical Review D **82**(7) (2010)
43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
44. Xuan, T., Zhu, Y., Durao, F., Sun, Y.: End-To-End Online sPHENIX Trigger Detection Pipeline. In: Machine Learning and the Physical Sciences Workshop at the 35th Conference on Neural Information Processing Systems (2021)
45. Zhu, Y., Xuan, T., Borca-Tasciuc, G., Sun, Y.: A new sPHENIX heavy quark trigger algorithm based on Graph Neutral Networks. In: Machine Learning and the Physical Sciences Workshop at the 35th Conference on Neural Information Processing Systems (2021)