End-To-End Online sPHENIX Trigger Detection Pipeline

Tingting Xuan
Department of Applied Mathematics & Statistics
Stony Brook University
Stony Brook, NY 11794
tingting.xuan@stonybrook.edu

Yimin Zhu
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794
yimzhu@cs.stonybrook.edu

Fedrik Durao
Sunrise Technology Inc.
Stony Brook, NY 11794
fedrik.durao@sunriseaitech.com

Yu Sun
Sunrise Technology Inc.
Stony Brook, NY 11794
yu.sun@sunriseaitech.com

Abstract

This paper provides a comprehensive end-to-end pipeline to classify triggers verse background events, make online decisions to filter signal data, and enable the intelligent trigger system for efficient data collection in the sPHENIX Data Acquisition System(DAQ). The pipeline starts with the coordinates of pixel hits that are lightened by passing particles in the detector, applies three-stages of event processing (hits clustering, track reconstruction, and trigger detection), and finally, labels all processed events with the binary tag of trigger versus background events. The whole pipeline consists of deterministic algorithms such as clustering pixels to reduce event size, tracking reconstruction to predict candidate edges, and advanced graph neural network-based models for recognizing the entire jet pattern. In particular, we apply the Message-Passing Graph Neural Network to predict links between hits and reconstruct tracks and a hierarchical pooling algorithm (DiffPool) to make the graph-level trigger detection. We attain an impressive performance ($\geq 70\%$ accuracy) for trigger detection with only 3200 neuron weights in the end-to-end pipeline.

1 Introduction

sPHENIX, introduced in Adare et al. [2015], is a new physics experiment that is under construction at the Relativistic Heavy Ion Collider (RHIC) facility in Brookhaven National Laboratory and enables an extremely rich jet and beauty quarkonia physics program to address fundamental questions about the nature of the strongly coupled quark-gluon plasma discovered at RHIC. It will operate in two years at much higher collision rates and greater instrumental precision than prior RHIC experiments. The sPHENIX tracking system includes a MAPS-based vertex detector (MVTX) in the innermost detector layer to take the snapshot of particle jets immediately after collision. We envision a run plan for 2022–2023 consisting of 22 weeks of Au+Au at 50 kHz and extended periods of p+p and p(d)+Au running at a 10 MHz collision rate. With the planned collision rate, sPHENIX will generate and process detector data stream continuously in real-time at the speed of multiple terabits per second. The data acquisition into disks and storage systems at this rate is cost-prohibitive. Furthermore, the vast data volume will incur a significant burden in data management, catalog, and off-line analysis and lengthens the path from data collection to science discovery.

The hardware trigger system used in the Large Hadron Collider (LHC) often sets thresholds on observables related to the total measured energy, which is mentioned in **??**. The hardware can not

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

capture the majority of interesting decays with medium kinetic energy as in the sPHENIX experiment. The demanding data selection to retain only less than 1% sPHENIX readouts and discard as many background events as possible necessitates sophisticated trigger algorithms using machine learning to recognize decaying patterns and make selection decisions in real-time.

Guest et al. [2018] summarizes recent efforts in using advanced statistical analysis, machine learning, and deep learning methods in event analysis in LHC particle physics experiments. Most of these efforts focus on removing noise, reconstructing tracks between different detectors, and identifying whether tracks are generated by electron, photon, or τ lepton in Kazeev [2020]. These algorithms gain extreme success in identifying low-level (local-level) physics properties, such as hit identification and tracking reconstruction, such as Ju et al. [2021] and DeZoort et al. [2021]. There are multiple attempts on event selection and other high-level physics tasks in Casa and Menardi [2018] and Baldi et al. [2014]. Based on our experiments, many machine learning and deep neural networks experience significant challenges in identifying high-level physics properties and deteriorate so rapidly that they can hardly outperform simple models. We attribute this finding to multiple reasons. There is some mismatch between the selected algorithm and available event data content, types, and structures. These algorithms treat these physics tasks in isolation and focus on improving each task independently and lack the co-design and integrated solutions that replicate the established physics processing workflows.

There are two main challenges to design and implement an online event selection algorithm. The first is to undertake this task end-to-end with limited information and raw input available at the moment of decision making. The second one is to design a neural network that is compatible with detector readout and capable of learning a wide spectrum of physics properties from low-level hits to the high-level trigger. We overcome these two difficulties with a three-stage pipeline and attain an impressive performance in online trigger prediction. The pipeline has three stages: 1) clustering the pixel readings into hits, 2) connecting the hits that belong to the identical particles to reconstruct event graphs, and 3) making the graph level decisions. The main contributions for this paper are as follows:

1. Our event selection pipeline only needs the MVTX fast detector readouts and undertakes multiple tasks that usually appear in offline data analysis and require accessing the reconstructed event data from various (slow) detectors that are not available during real-time data taking. Our work confirms the feasibility of moving these offline analysis tasks to online systems for intelligent data collection.
2. The pipeline is highly effective in reducing events readouts in all three stages:the clustering algorithm removes redundant hits, the tracking algorithm eliminates noisy hits with no meaningful connections to other hits, and the event tagging throws out 90% background events while retaining 37% signals, a $3.7\times$ increase in efficiency compared to random event selection.
3. The graph neural network (GNN) contains a few thousand parameters, only requiring a few thousand multiply-accumulates and can be deployed in hardware accelerators, such as field-programmable gate arrays (FPGAs) and graphics processing units (GPUs) via TensorRT.

2 End-to-End Physics Trigger Detection Pipeline

Our pipeline consists of three stages that reflect the interaction between particles and detectors and is consistent with the standard physics analysis workflow. During sPHENIX experiments, the generated particles from events fly through the detectors and activate a sequence of pixels in the layers of detectors. The detector readout contains the coordinates of these impinged pixels. Because silicon detectors have the advantage of high-speed readout, it becomes the standard choice in the core of detectors and enables a fast online triggering system. The disadvantage of this type of detector is that readouts only contain simple geometric information and require a sophisticated multi-stage processing pipeline to extract physical properties ranging from the low-level hits and tracks, to the high-level jets and tags incrementally. This pipeline must bootstrap the high-level physics from the lower-level features and recover a spectrum of physical properties from the detected hits. It also needs to reflect the offline physics processing and derive the trustworthy physics event model.

Pixels Clustering The experiment data from the detector and simulated data have the following features for each lightened pixel: the layer id, z index, ϕ of the cylindrical coordinates, and the three-dimensional Cartesian coordinates. A particle might impinge a cluster of pixels when it

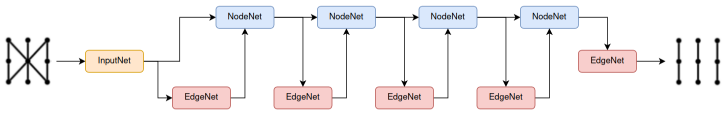

Figure 1: Graph Neural Network for Tracking

flies through a detector with different energies and momenta. These pixels must be grouped and represented by a cluster medoid. We use a deterministic algorithm to segment the contiguous pixels into separate clusters. Many traditional clustering algorithms might incur the $O(n^2)$ time complexity for calculating pair-wise distance for clustering. We cannot use these algorithms because of their time complexity. Instead, we take advantage of the fact that the pixels activated by the same particle must be adjacent in the detector space and design a connected component algorithm with a hash table that hashes pixel indices to memory locations for the efficient search for neighboring pixels. The clustering algorithm has a moderate time complexity of $O(n)$ and can be easily parallelized for the highly efficient execution that only incurs a fixed number of clock cycles in an FPGA-based hardware platform with available $O(n)$ processors.

Track Reconstruction Once we clean up the hits and remove redundant pixels in each event, we reconstruct the tracks by connecting the hits across different detector layers to recover the trajectories of particles. The upcoming sPHENIX detector contains three layers of interleaved silicon strips and generates tracks with three to six hits. By connecting hits in the order of a hit's interaction with detector layers, we recover the trajectory of particles. We model the initial input pixels as the cloud of hits and dynamically construct a graph with nodes (hits) and edges (track segments) to represent a collision event and perform information propagation and convolution with the constructed graphs. We adopt the Message-Passing Graph Neural Network proposed by Farrell et al. [2018] and divide the track reconstruction into two steps. The first step is to identify compatible hits belonging to the consecutive layers with similar angular coordinates and detector axial positions and connect the compatible hits into candidate tracks. To prevent excessive connections, we apply the geometric constraints for possible tracks, down select the candidate edges between two hits, and create the initial event graphs $G = (V, E)$, where V is the hit set, and E is the set of candidate edges. After selecting the edge candidates, the tracking problem is simplified to a binary edge classification problem (link prediction) to predict real track segments. In the second step, we iteratively apply the powerful graph convolution operations to propagate hit information among nodes via candidate edges and enrich each hit with multi-hop neighborhood information. Ultimately, we use a Multilayer Perceptron (MLP) on two end nodes (hits) of each candidate edge and predict whether the candidate edge belongs to the ground truth tracks. The GNN ensures that the graph-level information is learned during the information diffusion processing, and the pairwise link prediction is performed in the context of the entire event graph.

Trigger Prediction With DiffPool For the final step, we perform graph-level prediction based on the graph topology (tracking) learned in the previous stage and identify triggering events with characteristic decays. For graph-level trigger detection, a pooling method is needed to aggregate an arbitrary number of nodes and edges into a graph embedding with fixed dimensionality. The event tracks have a tree structure where the tree root represents the primary vertex, and each branch represents a decay (secondary vertex), and thereby can be aggregated by the bottom-up hierarchical pooling method from Ying et al. [2018] that preserves the original event structure and generates event-level embeddings. The differentiable graph pooling method has some resemblance to the pooling mechanism in Convolutional Neural Network(CNN). The pooling mechanism is straightforward in images with a regular mesh structure and only needs to aggregate all pixels in a square patch based on the maximum, summation, or average. Defining a good pooling operator on a graph structure is much more challenging. The commonly used global pooling directly aggregates all nodes to one vector to represent the whole graph. The global pooling might incur significant information loss on critical physical properties, particularly in our experiment condition where the primary vertex is close to the secondary vertex and the geometric track information of the ordinary and trigger events appear similar. DiffPool performs pooling operations layer by layer in Figure 2 and learns the soft assignment matrix between the nodes on a lower layer to nodes in an upper layer until the number of nodes falls below a threshold. The top layer outputs the aggregation of node embeddings that

represent the primary or secondary vertices. This learning-based pooling ensures that GNN extracts the event's cascaded decay patterns and exports the likelihood of a trigger event.

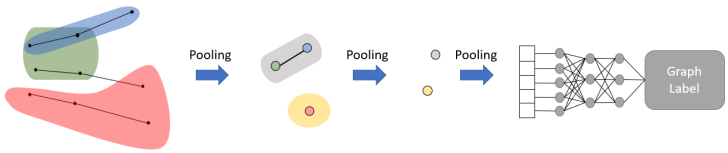

Figure 2: DiffPool Hierachical Pooling

3 Experiment Results

Experiment Dataset The dataset, provided by Huang [2018], contains simulated sPHENIX events with p+p collision at 200 GeV center-of-mass energy. We only use the readouts of the MVTX detector because the MVTX has no dependency on other detector components and is fast enough for the detector to readout individual events without concerning the impacts of background events that take place before the current one. For more details about sPHENIX events, Please refer to Adare et al. [2015]. There are three categories of events in the dataset: events with no-decay, decay but not trigger, and triggers. We further label events with no-decay and decay but not trigger as "negative" and label trigger events as "positive". The input data contains the 3D Cartesian coordinates of all hits. The ground truth from the simulation contains real tracks between hits for tracking and trigger flags for all events. The target information is only accessible in the objective function during the training stage and has never been used in the inference stage.

Tracking We sampled the events and built a balanced training data of trigger events and other events. Table 1 shows that the GNNs have impressive performance with more than 90% accuracy in identifying tracks. Furthermore, the accuracy is highly dependant on the number of layers and hidden dimensions of the MLPs in Graph Neural Networks. A larger number of parameters might lead to higher accuracy while incurring more considerable variance in prediction, more complex parameter space, longer training time, and higher costs for inference compared with the case of a lower dimensions. To trade-off between parameter space and inference latency, we chose hidden dimensionality to be 8, four MLP layers, and four iterations of message passing across all hits before the final output layer of link prediction between the selected pairs of hits.

Table 1: Tracking Graph Neural Network Performance

Hidden dim	MLP layers	Number of Trainable Parameters	Edge Classification Accuracy
8	2	457	93.2%
8	4	745	94.1%
16	2	1681	94.89%
16	4	2769	96.1%
32	2	6433	95.5%
32	4	10657	96.3%

Trigger Detection We use 500,000 trigger events and 500,000 non-trigger events as training sets and 20K events for validation to induce the trigger prediction model. Table 2 shows that the accuracy for different hidden dimensions stays around 70%. We attribute this finding to the fact that the simple geometrical information of hits only requires a small dimensionality for the hidden representation. During the inference, we use testing data with one percent of trigger events. Table 2 also shows the model preserves 37% of triggers when we set the threshold to reject 90% of background events.

Table 2: DiffPool Trigger Detection Performance

Hidden dim	Number of Trainable Parameters	Accuracy	AUC	Efficiency	Purity
8	2441	70.33%	76.84%	37.87%	3.83%
16	5937	69.73%	76.23%	37.30%	3.78%
32	17153	70.11%	76.55%	37.08%	3.75%

4 Conclusion, Limitation and Future Work

The data generated by the sPHENIX experiment outpaces the existing data acquisition systems and necessitates intelligent event tagging and filtering. This paper addresses this challenge and provides an end-to-end solution for the online sPHENIX trigger detection. A tremendous effort focuses on offline reconstruction and analysis once experiment data is collected and transferred to data centers and clouds, such as a trigger algorithm based on GNN by Zhu et al. [2021]. Our endeavor is one of the first attempts to shift many offline tasks, such as noise removal, event reconstruction, and tagging, to the early stage of data collection, providing proof-of-concept for software-based intelligent triggers for future physics experiments and expediting the turn-around from experiment to science discovery. We also deploy this pipeline on FPGA in Xuan et al. [2021]

5 Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Office of Nuclear Physics, under Award Number DE-SC0019518.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results... [No]
   (a) Did you state the full set of assumptions of all theoretical results?
   (b) Did you include complete proofs of all theoretical results?
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyper parameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects... [No]
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable?
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable?
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation?

References

A. Adare, S. Afanasiev, C. Aidala, N. Ajitanand, Y. Akiba, R. Akimoto, J. Alexander, K. Aoki, N. Apadula, H. Asano, et al. An upgrade proposal from the phenix collaboration. *arXiv preprint arXiv:1501.06197*, 2015.

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

A. Casa and G. Menardi. Nonparametric semisupervised classification for signal detection in high energy physics. *arXiv preprint arXiv:1809.02977*, 2018.

G. DeZoort, S. Thais, J. Duarte, V. Razavimaleki, M. Atkinson, I. Ojalvo, M. Neubauer, and P. Elmer. Charged particle tracking via edge-classifying interaction networks. 3 2021.

S. Farrell, P. Calafiura, M. Mudigonda, D. Anderson, J.-R. Vlimant, S. Zheng, J. Bendavid, M. Spiropulu, G. Cerati, L. Gray, et al. Novel deep learning methods for track reconstruction. *arXiv preprint arXiv:1810.06111*, 2018.

D. Guest, K. Cranmer, and D. Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68:161–181, 2018.

J. Huang. sphenix machine learning open data set for tracking and heavy flavor physics. `https://github.com/sPHENIX-Collaboration/HFMLTrigger`, 2018.

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

N. Kazeev. *Machine Learning for particle identification in the LHCb detector*. PhD thesis, Yandex School of Data Analysis (RU), 2020.

T. Xuan, Y. Zhu, F. Durao, and Y. Sun. High performance fpga embedded system for machine learning based tracking and trigger in sphenix and eic. *Journal of Instrumentation*, 2021. Submitted.

R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

Y. Zhu, T. Xuan, G. Borca-Tasciuc, and Y. Sun. A new sphenix heavy quark trigger algorithm based on graph neutral networks. *Machine Learning and the Physical Sciences Workshop at the 35th Conference on Neural Information Processing Systems*, 2021.