Heterogeneous Graph Neural Network for Identifying Hadronically Decayed Tau Leptons at the High Luminosity LHC

Andris Huang,[a] **Xiangyang Ju,**[b] **Jacob Lyons,**[a] **Daniel Murnane,**[b] **Mariel Pettee,**[c] **and Landon Reed**[d]

[a] *Physics Department, University of California, Berkeley, CA 94720, USA*
[b] *Scientific Data Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[c] *Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[d] *Physics Department, University of Minnesota Duluth, Duluth, MN 55812, USA*

E-mail: andrewhz@berkeley.edu, xju@lbl.gov, jacoblyons98@berkeley.edu, dtmurnane@lbl.gov, mpettee@lbl.gov, reed0658@d.umn.edu

Abstract: We present a new algorithm that identifies reconstructed jets originating from hadronic decays of tau leptons against those from quarks or gluons. No tau lepton reconstruction algorithm is used. Instead, the algorithm represents jets as heterogeneous graphs with tracks and energy clusters as nodes and trains a Graph Neural Network to identify tau jets from other jets. Different attributed graph representations and different GNN architectures are explored. We propose to use differential track and energy cluster information as node features and a heterogeneous sequentially-biased encoding for the inputs to final graph-level classification.

Keywords: tau lepton, graph neural network, heterogeneous graph, LHC

Contents

1 Introduction **1**

2 Simulation **2**

3 Methods **3**
- 3.1 Graph Neural Networks 3
- 3.2 Homogeneous Graph Representations 7
- 3.3 Heterogeneous Representations 7

4 Results **9**
- 4.1 Homogeneous Graph Representations 9
- 4.2 Heterogeneous Encoders 10

5 Discussions **11**
- 5.1 Impact of Fixed-Length Inputs 11
- 5.2 Message Passing Steps 12
- 5.3 Influence of the Sequential Relation 13
- 5.4 Effects of Pileup 13

6 Conclusions and Outlook **14**

1 Introduction

Tau leptons are important in many physics programs in ATLAS [1] and CMS [2]. Examples are the measurements of the Higgs boson [3–14] and the search for additional Higgs boson [15–19]. These analyses highly depend on the efficiency and accuracy of the tau reconstruction and identification. To this end, both ATLAS and CMS combined the tracking information reconstructed from hits recorded by their inner detectors with the energy cluster information measured by their calorimeters. ATLAS developed a *Tau Particle Flow* method. Firstly, the kinematics of charged and neural pions are reconstructed, respectively. Then both pieces of information are used as inputs to train a Boost Decision Tree (BDT) to distinguish τ leptons [20]. CMS developed a deep neural network, specifically the Convolutional Neural Network (CNN), which takes as inputs the physics-inspired features combined with hidden features learned from energy deposits in the calorimeter and outputs a multi-class score that discriminates τ leptons against jets, electrons, and muons [21]. In addition, ATLAS used a Recurrent Neural Network (RNN) to identify hadronic tau τ_{had} decays [22]. The relational information between tracks and energy clusters explored by ATLAS and CMS is mostly based on physics inspirations, serving as inputs to the τ_{had}

identification algorithms. We propose to use Graph Neural Network to learn the relational information between tracks and towers for the τ_{had} identification.

Tau lepton decays hadronically 65% of the time and leptonically 25% of the time. In the hadronic decay mode, tau leptons primarily decay to one charged pion (70%) or to three charged pions (21%), and the majority (68%) include one or more neutral pions. Therefore, their experimental signature corresponds to a jet with one or three tracks in the detector. The former signature is called *one prong* and the latter *three prongs*. Neutrinos from the hadronic tau lepton decay may be inferred from the missing transfer momentum but cannot be reconstructed. This paper focuses on hadronic tau decays.

Recent years have seen many successful applications of treating proton-proton (pp) collision events as graphs and using Graph Neural Networks (GNN) to identify physics objects in particle physics. Ref [23] reviews general GNN applications in particle physics, and Ref [24] focuses on applications in particle tracking and reconstruction. Graphs can naturally represent collision events at different levels depending on the objective of the task. At a high-level, graphs can represent the collision events with reconstructed physics objects as nodes. At a lower level, graphs can represent individual physics objects with detector-level objects such as tracks and energy clusters/towers as nodes. In both cases, graph edges may or may not exist, and nodes are of different types, making those graphs heterogeneous. Heterogenous graphs can be treated as homogeneous graphs by selecting a subset of common node and edge features at the cost of information loss or by padding node features with zeros at the cost of computations. We represent heterogeneous GNNS that treat different node and edge types differently.

This work is organized as follows. Section 2 describes the simulated dataset, followed by Section 3 explaining different graph representations and GNN architectures. Numerical results are shown in Section 4. And we discuss our findings in Section 5 and present conclusions in Section 6.

2 Simulation

Proton-proton collisions are simulated with Pythia 8.302 [25, 26] at a center-of-mass-energy of $\sqrt{s} = 13$ TeV. The detector response is simulated by Delphes 3 [27] with the ATLAS detector configuration, and an average of 200 additional pp collisions emulating the collision density at the High Luminosity LHC are added to each event. Delphes uses the particle-flow reconstruction algorithms to produce *particle-flow tracks* and *particle-flow towers*, which then serve as inputs to reconstruct jets and missing transverse energy. Jets are reconstructed with the anti-k_t [28] algorithm with a radius of 0.4 using FastJet 3.3.2 [29, 30]. As no τ reconstruction algorithm is used, all jets with $p_{\text{T}} > 30$ GeV and $|\eta| < 2.7$ are treated as τ_{had} candidates.

Genuine τ_{had} leptons are simulated by the $pp \to \gamma^* \to \tau\tau$ processes, and fake τ_{had} candidates by jets from the Quantum chromodynamics (QCD) processes. Note that the on-shell Z boson production is excluded to avoid possible biases on the jet kinematic variables. Both are generated by Pythia 8.302, which is also used for parton shower, hadronization, and τ lepton decay. The A14 [31] set of tuned parameters and the NNPDF2.3 LO [32]

set of parton distribution functions with $\alpha_s = 0.13$ are used. All tau leptons were forced to decay hadronically to maximize the generation efficiency. In addition, the invariant mass of two tau leptons ($m^*_{\tau\tau}$) is set to be in the range from 60 GeV to 7000 GeV. To populate generated jet p_{T} spectrum towards high values, the QCD events use a biased phase space sampling which is compensated by a continuously decreasing weight for the event.

We generate 964,000 ditau events and 885,000 QCD events, of which 80% are used for training, 10% for validation, and 10% for testing. A jet is matched to a true τ lepton if the angular distance ΔR [*] between the two objects is less than 0.2. We label a jet as a real τ_{had} if a jet is matched to a true τ lepton; otherwise, a QCD jet. With that, there are 134,000 true 1-Prong τ_{had} jets, 75,000 true 3-Prong τ_{had} jets (total 209,000 true τ_{had} jets), and 2,800,000 QCD jets in the generated events. Given the imbalances of the true τ_{had} jets and QCD jets, we assign a larger weight to true tau jets in the loss function so that the total weights of the two classes are close.

Inputs to the neural networks for the tau identification are the detector-level track parameters ($p_{\mathrm{T}}^{\mathrm{track}}$, η^{track}, ϕ^{track}, d_0, z_0) and tower kinematic variables ($E_{\mathrm{T}}^{\mathrm{tower}}$, η^{tower}, ϕ^{tower}). The d_0 and z_0 are the transverse and longitudinal impact parameters with respect to the collision point. Figure 1 shows the kinematic variables of the tracks and towers. We applied a log transformation to the $p_{\mathrm{T}}^{\mathrm{track}}$ and $E_{\mathrm{T}}^{\mathrm{tower}}$ and normalized all input features between -1 and 1. In addition, the jet kinematic variables, namely the jet p_{T}, η, and ϕ, are also used. And their distributions are shown in Figure 2.

Jets from the hadronically decayed tau leptons have geometrically different track and tower distributions than those from QCD. As shown in Figure 3, tracks are more concentrated on the jet axis for τ_{had} jet than QCD jets. Therefore, we divide the regions encompassing the jet axis according to the distance ΔR away from the jet axis. We define the *core region* as $\Delta R < 0.1$, the *isolated (central) region* $0.1 \leq \Delta R < 0.2$, and the *outer region* $0.2 \leq \Delta R < 0.4$.

This further motivates us to calculate physics-inspired high-level variables defined in Table 1. Figure 4 compares the distributions of the high-level variables among the 1-Prong τ_{had}, the 3-Prong τ_{had}, and the QCD jets. Although the 1-Prong τ_{had} and 3-Prong τ_{had} decay modes differ, their input features fed to the neural network are the same, as listed in Table 2. The same model can be applied to both decay modes; therefore, we trained one model for both decay modes to gain statistics.

3 Methods

3.1 Graph Neural Networks

The GNN models we are exploring contain three discrete modules: Graph Encoder, Message Passing, and Graph Decoder, as pictured in Figure 5. Each module contains three basic neural networks: node networks, edge networks, and global networks. These basic neural networks can be any network architecture. In our studies, they are two layers of Multi-Layer Perceptrons (MLPs), each with 64 units.

[*] ΔR is the Euclidean distance in the transverse plane, defined as $\sqrt{\Delta\eta^2 + \Delta\phi^2}$

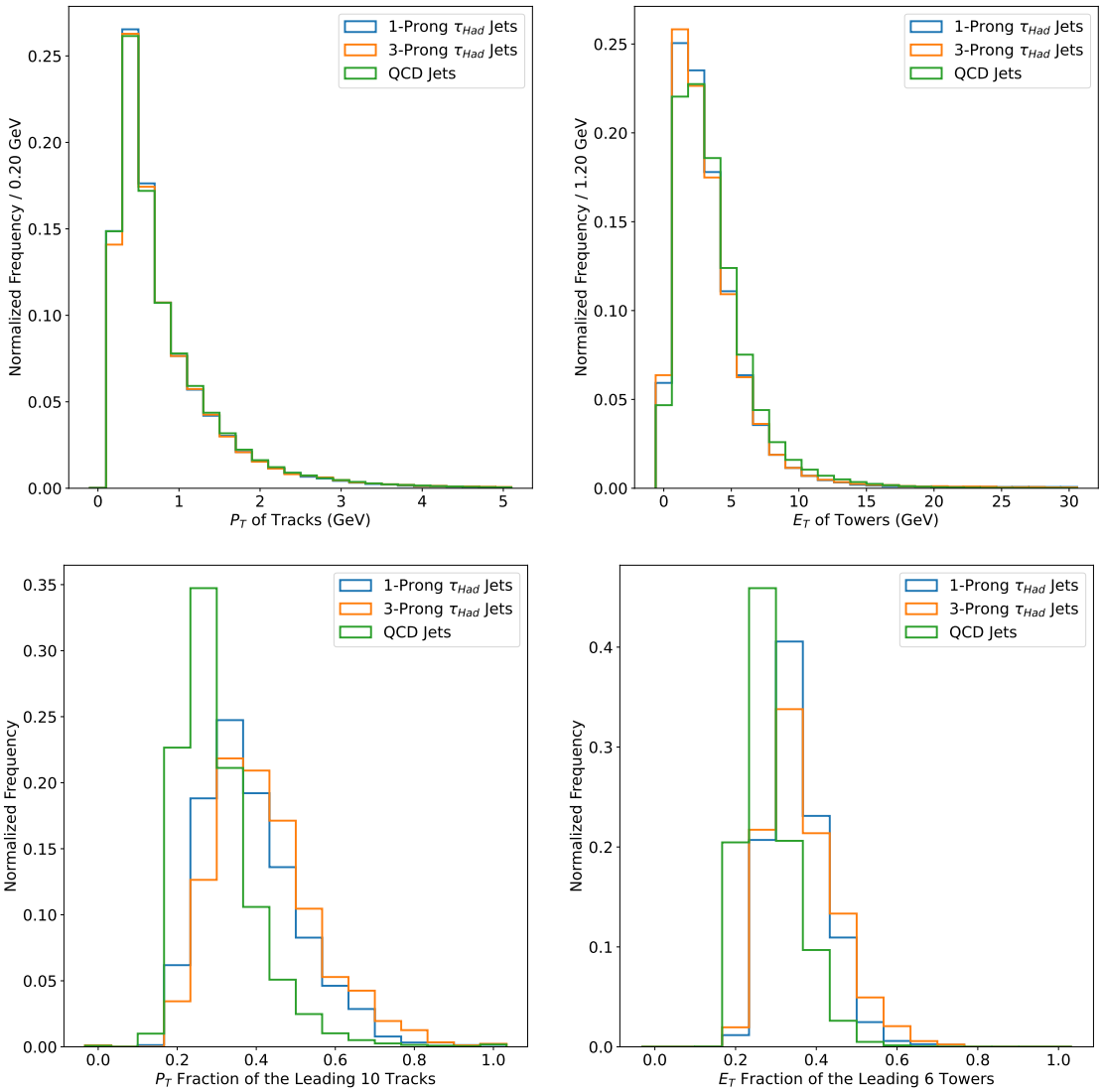

Figure 1. Comparison of the p_{T} and E_{T} of 1-Prong τ_{had} jets, 3-Prong τ_{had} jets, and background (QCD jets). All distributions are normalized to the same area. The fraction of p_{T} (E_{T}) of the 10 (6) leading tracks (towers) is shown in the bottom rows.

Graph Encoder first updates input node features v_i through a node network ϕ^v: $v_i' = \phi^v(v_i)$, where i loops over all nodes. Then, the graph encoder creates edge features using the concatenated features of the two connected nodes through an edge network: $e_k' = \phi^e(v_{r_k}, v_{s_k})$, where v_{r_k} and v_{s_k} are the previously updated node features of the two nodes connected to the edge in question and k loops over all edges. Finally, the graph-level attributes are updated independently through a graph-level network: $u' = \phi^u(u)$. After going through the graph encoder, the input graph becomes an attributed graph with latent node, edge, and graph-level features, denoted as H_0 in Figure 5.

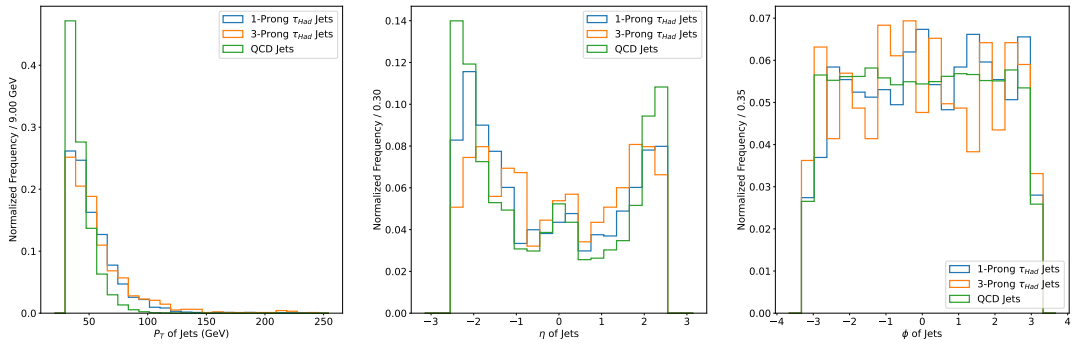

Figure 2. Comparison of the jet p_{T}, η, and ϕ of 1-Prong τ_{had} jets, 3-Prong τ_{had} jets, and background (QCD jets). All distributions are normalized to the same area.

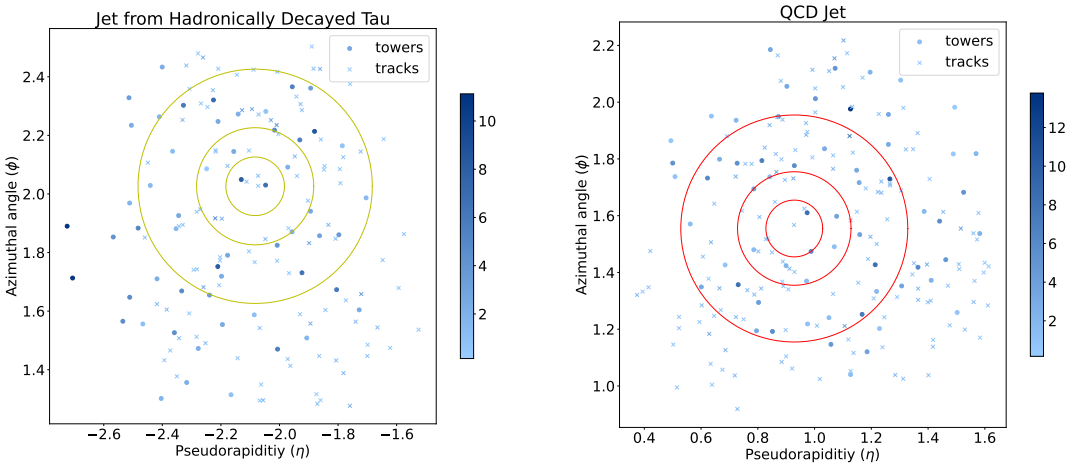

Figure 3. τ_{had} candidate jet vs. QCD jet, with the color of tracks and towers corresponding to p_{T} and E_T respectively

Message Passing is designed to exchange information among the three-level graph attributes. Each message passing step updates the edge-, node-, and graph-level features sequentially. First, edge features at the t-th message passing step are updated:

$$e_k^t = \phi^e(e_k^0, e_k^{t-1}, v_{r_k}^{t-1}, v_{s_k}^{t-1}, u^{t-1})$$

where e_k^0 are the edge features after the encoder, e_k^{t-1} are the edge features at the previous message passing step. Similarly, $v_{r_k}^{t-1}$ and $v_{s_k}^{t-1}$ are the node features and u^{t-1} are the global features. Then node features are updated.

$$v_i^t = \phi^v(v_i^0, v_i^{t-1}, \bar{e}_i^t, u^{t-1})$$

where $\bar{e}_i^{t-1}$ is the aggregated edge features at the t-th message passing step. Finally, global features are updated:

$$u^t = \phi^u(u^0, u^{t-1}, \bar{e}_i^t, \bar{v}^t)$$

Table 1. Definitions of high-level variables for hadronic τ_{had} identification.

Name	Symbol	Definition
Leading Track Momentum Fraction	f_{track}	Transverse momentum of highest p_{T} track associated with jet divided by the sum of transverse energy found in the core region of the jet
Track Radius	R_{track}	p_{T} weighted distance of tracks associated with tau candidate from tau candidate direction
Track Mass	m_{track}	Invariant mass calculated from the sum of the four-momenta of all tracks in the core and isolation region (assuming pion mass for each track)
Number of Isolated Tracks	$N_{\text{track}}^{\text{iso}}$	Number of tracks in the isolated region of jet line line line line
Maximum ΔR	ΔR_{track}	Maximum distance between a core track associated with the tau candidate and the tau candidate direction

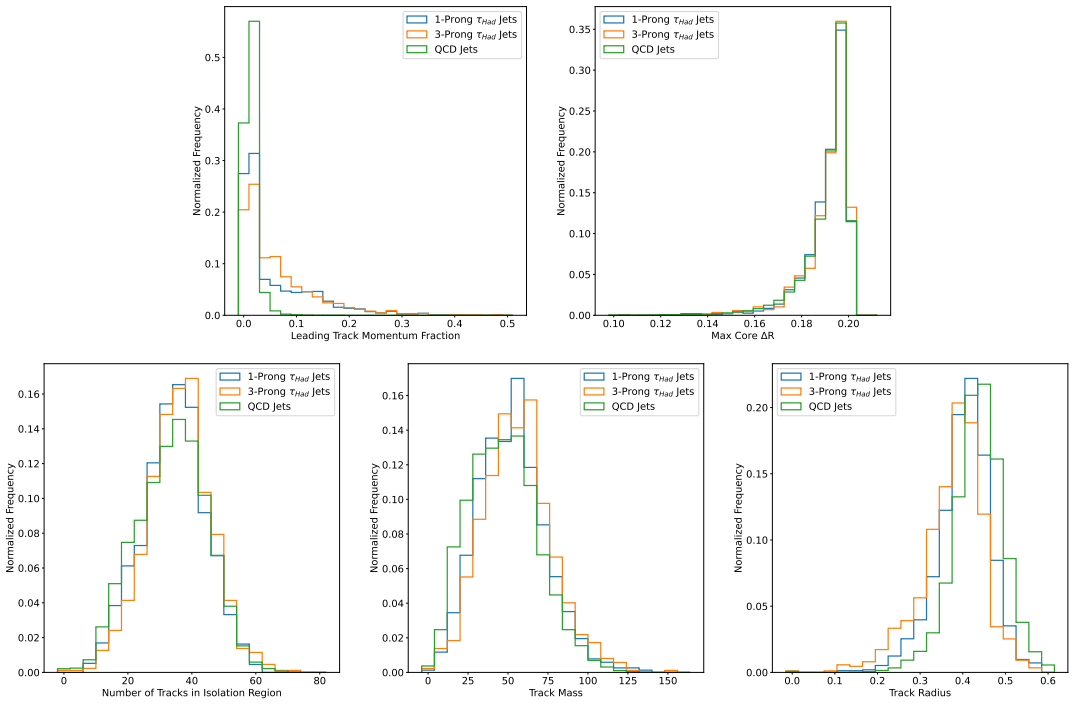

Figure 4. Comparison of distributions of high-level variables for 1-Prong τ_{had} jets, 3-Prong τ_{had} jets, and background (QCD jets). All distributions are normalized to the same area.

where $\bar{v}^t$ is the aggregated node features at the t-th message passing step.

Graph Decoder updates node, edge, and graph-level features independently. In the τ_{had} identification algorithm, Graph Decoder applies neural networks on the graph-level features to get a classification score.

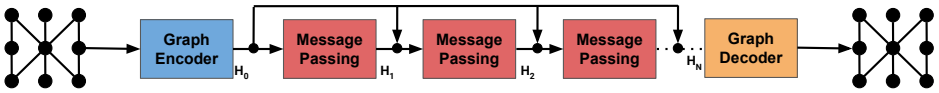

Figure 5. A general graph neural network architecture.

3.2 Homogeneous Graph Representations

Each jet is represented as a fully-connected graph in which nodes are tracks or towers, and edges are connections between every two nodes. We explore different graph attribute assignments. The baseline is to assign node features with only the track or tower kinematic variables, denoted as G_1. Then we add the $p_{\mathrm{T}}^{\mathrm{jet}}$ to all nodes, G_2, and jet kinematics to graph-level attributes, G_3. Instead of using the absolute angular variables (η and ϕ) of tracks and towers, we use the relative angular variables to the track/tower nodes: $\eta^{\mathrm{track}} - \eta^{\mathrm{jet}}$ and $\phi^{\mathrm{track}} - \phi^{\mathrm{jet}}$, leading to configurations of G_4, G_5, and G_6. Physics-inspired edge features, as used in Ref [33], were added to the graphs as inputs. But no improvement was observed. Hence, no edge features are used in the input graphs.

Table 2 summarizes different graph attribute assignments explored. To compare the performance of each graph representation, we use the same homogeneous GNN model, of which the detail is in Section 3.1. Note that tower node features are padded with zeros to make the graphs homogeneous. The padding is not needed for heterogeneous GNNs.

Table 2. Different graph representations for jets. All graphs are full-connected graphs with different node and graph-level attributes. Track node features and tower node features are harmonized by padding the tower node features with zeros. G_1, G_2, and G_4 graph types do not have global (i.e. graph-level) attributes. See Table 1 for high-level variable definitions.

	G_1	G_2
Track nodes	$p_{\mathrm{T}}^{\mathrm{track}}$, η^{track}, ϕ^{track}, d_0, z_0	$p_{\mathrm{T}}^{\mathrm{track}}$, η^{track}, ϕ^{track}, d_0, z_0, $p_{\mathrm{T}}^{\mathrm{jet}}$
Tower nodes	$E_{\mathrm{T}}^{\mathrm{tower}}$, η^{tower}, ϕ^{tower}, 0, 0	$E_{\mathrm{T}}^{\mathrm{tower}}$, η^{tower}, ϕ^{tower}, 0, 0, $p_{\mathrm{T}}^{\mathrm{jet}}$
Global	None	None
	G_3	G_4
Track nodes	$p_{\mathrm{T}}^{\mathrm{track}}$, η^{track}, ϕ^{track}, d_0, z_0, $p_{\mathrm{T}}^{\mathrm{jet}}$	$p_{\mathrm{T}}^{\mathrm{track}}$, $\eta^{\mathrm{track}} - \eta^{\mathrm{jet}}$, $\phi^{\mathrm{track}} - \phi^{\mathrm{jet}}$, d_0, z_0, $p_{\mathrm{T}}^{\mathrm{jet}}$
Tower nodes	$E_{\mathrm{T}}^{\mathrm{tower}}$, η^{tower}, ϕ^{tower}, 0, 0 , $p_{\mathrm{T}}^{\mathrm{jet}}$	$E_{\mathrm{T}}^{\mathrm{tower}}$, $\eta^{\mathrm{tower}} - \eta^{\mathrm{jet}}$, $\phi^{\mathrm{tower}} - \phi^{\mathrm{jet}}$, 0, 0, $p_{\mathrm{T}}^{\mathrm{jet}}$
Global	$p_{\mathrm{T}}^{\mathrm{jet}}$, η^{jet}, ϕ^{jet}	None
	G_5	G_6
Track nodes	$p_{\mathrm{T}}^{\mathrm{track}}$, $\eta^{\mathrm{track}} - \eta^{\mathrm{jet}}$, $\phi^{\mathrm{track}} - \phi^{\mathrm{jet}}$, d_0, z_0, $p_{\mathrm{T}}^{\mathrm{jet}}$	$p_{\mathrm{T}}^{\mathrm{track}}$, $\eta^{\mathrm{track}} - \eta^{\mathrm{jet}}$, $\phi^{\mathrm{track}} - \phi^{\mathrm{jet}}$, d_0, z_0, $p_{\mathrm{T}}^{\mathrm{jet}}$
Tower nodes	$E_{\mathrm{T}}^{\mathrm{tower}}$, $\eta^{\mathrm{tower}} - \eta^{\mathrm{jet}}$, $\phi^{\mathrm{tower}} - \phi^{\mathrm{jet}}$, 0, 0, $p_{\mathrm{T}}^{\mathrm{jet}}$	$E_{\mathrm{T}}^{\mathrm{tower}}$, $\eta^{\mathrm{tower}} - \eta^{\mathrm{jet}}$, $\phi^{\mathrm{tower}} - \phi^{\mathrm{jet}}$, 0, 0, $p_{\mathrm{T}}^{\mathrm{jet}}$
Global	$p_{\mathrm{T}}^{\mathrm{jet}}$, η^{jet}, ϕ^{jet}	$p_{\mathrm{T}}^{\mathrm{jet}}$, η^{jet}, ϕ^{jet}, High-level Variables

3.3 Heterogeneous Representations

The differences between homogeneous GNNs and heterogeneous GNNs lie in whether the basic neural networks treat different node and edge types differently. For homogeneous GNNs, all nodes and edges are updated by the same neural networks. However, for heterogeneous GNNs, different neural networks are applied to different node types and edge types to perform encoding, message passing, or decoding. In our study, we mainly

explore three different heterogeneous graph encoders: the Heterogeneous Node Encoder, the Heterogeneous Node and Edge Encoder, and the Recurrent Encoder.

The homogeneous GNN model uses the *Homogeneous Encoder*, which treats all tracks and towers as the same type of objects and applies the same basic MLPs to each. The *Heterogeneous Node Encoder* treats the nodes as two types of objects, namely the tracks and towers, and applies two separate neural networks to each. The updated nodes are then passed into the same aggregation functions and edge networks. The *Heterogeneous Node and Edge Encoder* utilizes three distinct edge update functions corresponding to the three different edges in the graph, in addition to the two separate node update functions in the *Heterogeneous Node Encoder*. An illustration of this architecture is shown in Figure 6.

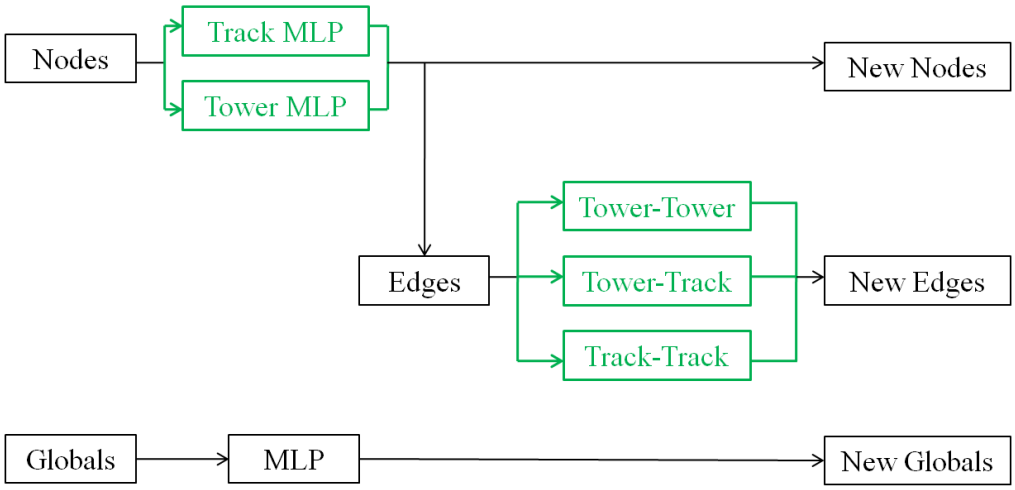

Figure 6. Architecture of the *Heterogeneous Node and Edge Encoder*.

The *Recurrent Encoder* is inspired from the Recurrent Neural Network architecture [22]. Instead of a permutation-invariant encoding, this model encodes the tracks and towers separately as sequences, using two MLP embedding layers followed by two Long Short-Term Memory (LSTM) layers [34], as illustrated in Figure 7. All of the layers use 32 hidden units. The sequential encodings require a fixed-length input, and hence we only use the first ten tracks and six towers inside the jets, ordered descendingly by their P_T and E_T values. In addition, the edges are not used in this encoding, and the global attributes are updated only based on the node and graph-level attributes. Hence, we do not use a *Message Passing* module and directly pass the attributed graphs into a *Graph Decoder* module.

In order to study the importance of the sequential relation used in the *Recurrent Encoder*, we also explore the *Attention Encoder* with and without Positional Encodings. The *Attention Encoder* has the same architecture as the *Recurrent Encoder*, except that the two LSTM layers are replaced with two Multi-Head Attention layers [35] with eight heads and embedding dimension of 32 units, where the first layer is a self-attention layer among the corresponding nodes, and the second layer conducts attention from the embedded graph-level attributes to the nodes. We then evaluate the importance of the sequential

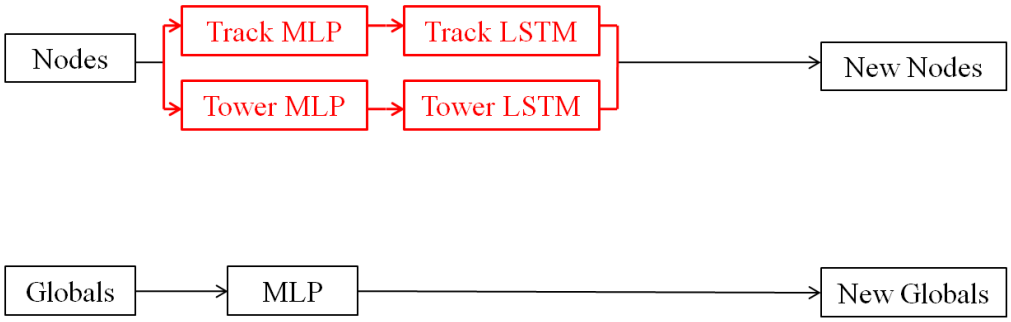

Figure 7. Architecture of the *Recurrent Encoder*.

relation by adding Positional Encodings to the inputs before passing them into the MLPs. Table 3 summarizes those GNN architectures.

Models	Parameters	Input Graph	Message Passing Steps	AUC	Rejection at 75% Efficiency
Homogeneous Encoder	112,961	G_5	1	0.9851	239.7
Heterogeneous Node Encoder	130,049	G_5	1	0.9864	339.6
Heterogeneous Node and Edge Encoder	154,689	G_5	1	0.9886	405.2
Recurrent Encoder	62,801	G_6	0	0.9932	4616.7
Attention Encoder with Positional Encoding	154,329	G_6	0	0.9926	1897.3

Table 3. Summary of different GNN architectures.

4 Results

We evaluate the performance of different GNN models by examining the number of background QCD jets rejected at different signal τ_{had} selection efficiencies. Four working points at 45%, 60%, 75%, and 95% signal selection efficiencies are highlighted in the plots as points.

4.1 Homogeneous Graph Representations

The performance of fully-connected, homogeneous graph networks acting on different graph representations is shown in Figure 8.

By comparing the solid curves (G_1 through G_3), i.e., the models with absolute positions, we can see that the models gain a noticeable improvement when the jet-level variables are added as inputs, partially due to the separation power of the $p_{\text{T}}^{\text{jet}}$. Although the η^{jet} and ϕ^{jet} cannot separate the τ_{had} jet from the QCD jets, changing the η and ϕ of the jet constituents to values relative to η^{jet} and ϕ^{jet} results in better performance as concluded by comparing G_2 with G_4 and G_3 with G_5. The higher rejection power of G_4 at all working points indicates that using relative positions improves the model performance. By comparing the G_4 with the G_5, we can see that the model gains further improvement when jet-level variables are added as graph-level features even though the jet positions are already encoded in the nodes as relative positions. Therefore, the jet-level variables

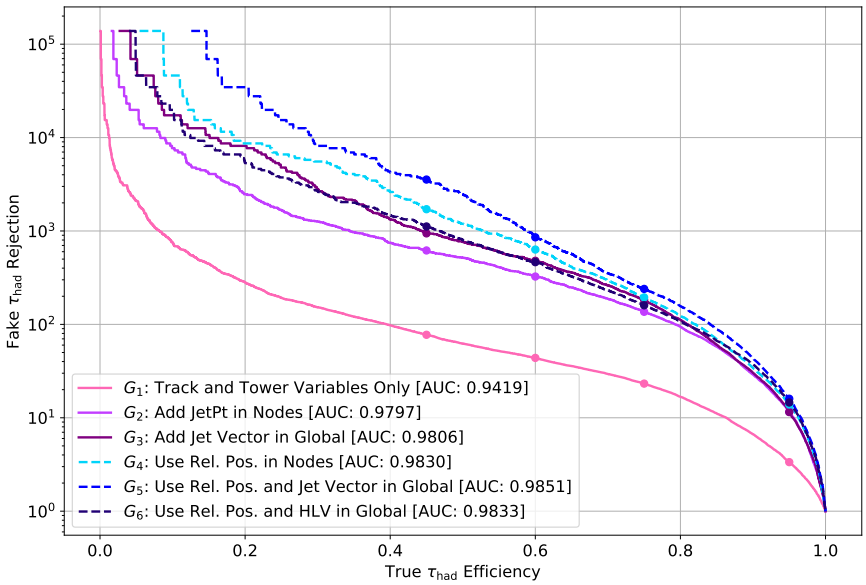

Figure 8. The number of background QCD jets rejected vs. signal τ_{had} selection efficiency for fully-connected, homogeneous graph network with different graph attributes. Curves with solid lines are graphs constructed with absolute track and tower positions as node inputs (G_1 through G_3), while curves with dashed lines are for graphs with relative positions (G_4 through G_6). Inputs for each configuration are listed in Table 2.

deemed essential for the GNN models. However, it is worth noting that adding the derived high-level variables did not improve the GNN performance.

4.2 Heterogeneous Encoders

All models use the input graph representation G_5 specified in Table 2, except for the *Recurrent Encoder*, which uses G_6 and only the first 10 tracks and 6 towers as node inputs. The models with *Message Passing* modules all use one message-passing step in this section. The performance of models with different encoders is presented in Figure 9.

It can be seen that the *Heterogeneous Node Encoder* rejects more background jets for higher efficiencies than the *Homogeneous Encoder* but rejects fewer background jets for lower efficiencies. Specifically, the *Heterogeneous Node Encoder* rejects 100 more background jets (41.7%) than the *Homogeneous Encoder* at 75% efficiency while rejecting 1705 fewer background jets (48.0%) at 45% efficiency. Compared to the *Heterogeneous Node Encoder*, the *Heterogeneous Node and Edge Encoder* rejects more background jets for all efficiencies, indicating that handling different types of edges using different MLPs improves the performance across all signal efficiency. It can also be seen that the *Heterogeneous Node and Edge Encoder* has a better rejection power than the *Homogeneous Encoder* for higher efficiencies and similar rejection power for lower efficiencies, which yields a better AUC value and hence an indication of better overall performance. Noticeably, the *Recurrent*

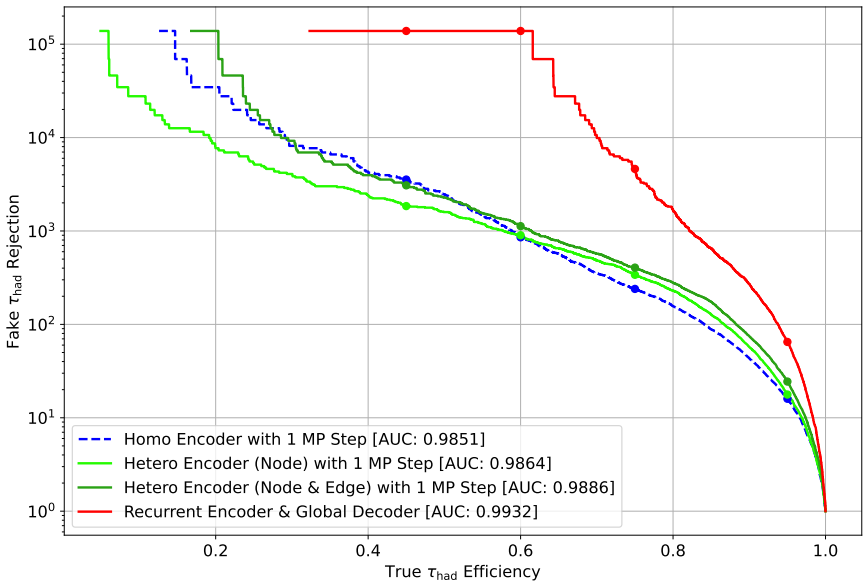

Figure 9. Number of background QCD jets rejected vs signal τ_{had} selection efficiency for homogeneous Graph Neural Network and heterogeneous Graph Neural Network. The dashed blue curve represents the homogeneous model presented as G_5 in Figure 8. The solid curves represent models with heterogeneous encoders.

Encoder has significantly better rejection power than all other models, and potential explanations of this behavior are discussed in the following sections.

In addition, to check if the improvement of the heterogeneous encoder is a result of more trainable parameters, we also trained a *Homogeneous Encoder* model with 178,241 parameters, comparable to the size of *Heterogeneous Node and Edge Encoder*. No significant improvement of the *Homogeneous Encoder* is observed; the new AUC value is 0.9843, close to the original value and lower than that of *Heterogeneous Node and Edge Encoder*.

We also compares our results with the Particle Flow Network (PFN) model, a performant deep learning model for tagging top quarks using particle-level information [36, 37]. Although we use detector-level tracking and cluster information whilst the top quark taggers use particle-level hadron information, we take the PFN model and adjust the model size to be comparable with the *Recurrent Encoder* model. The comparison is shown in Table 4. Compared to the PFN model, the *Recurrent Encoder* provides more than a factor of two rejection powers for a signal efficiency of 75%.

5 Discussions

5.1 Impact of Fixed-Length Inputs

Since the *Recurrent Encoder* only accepts fixed-length input, we explore the impact of fixed-length inputs for other encoders by setting a cutoff in the number of nodes used,

Models	Parameters	AUC	Rejection at 75% Efficiency
Particle Flow Network	63,442	0.9907	1775.6
Heterogeneous Node and Edge Encoder	154,689	0.9886	405.2
Recurrent Encoder	62,801	0.9932	4616.7

Table 4. Comparison with Particle Flow Network in Ref. [36].

i.e., for a given input graph, changing the input node features from all tracks and towers associated with the jet to only the first 10 tracks and 6 towers, the same as the *Recurrent Encoder*. Comparing the *Heterogeneous Node and Edge Encoder* model with G_5 graph configuration, we observed that the model with cutoff yields a slightly worse rejection power than the model with all nodes, for all working points, where the AUC value is decreased from 0.9886 to 0.9876, and the rejection at 75% efficiency decreased from 448 to 373. The other models also exhibit similar behavior. It is worth noting that applying the cutoff reduces the number of input nodes, thereby reducing the computation time in training from 16 hours to 12 hours.

5.2 Message Passing Steps

Due to the sequential relation, the LSTM layer embeds the t^{th} node in the sequence by following the equation

$$o_t = \sigma(W_{io}x_t + W_{ho}h_{t-1} + b_o),$$

where o_t is the output for the t^{th} node, σ is the sigmoid function, x_t is the input of the t^{th} node, h_{t-1} is the hidden state of the $(t-1)^{\text{th}}$ node, and $W_{io},\ W_{ho},\ b_o$ are matrices for trainable parameters [38]. This relation indicates that the embedding of the t^{th} node is obtained by aggregating the hidden embedding of all the previous $t-1$ nodes *recursively*.

In comparison, the permutation-invariant encoders update the nodes v_i in each message-passing step by

$$v_i' = \phi^v(v_i, u, \bar{e_i}'),$$

where v_i' is the updated node embedding, ϕ^v is the node update function, v_i is the node input, u is the graph-level feature, and $\bar{e_i}'$ is the aggregated edge variable obtained by aggregating

$$e_k' = \phi^e(e_k, v_{r_k}, v_{s_k}, u),$$

where e_k' is the updated edge feature obtained by applying the edge updating function on the *original* edge embedding e_k, the sender and receiver nodes v_{r_k} and v_{s_k} connected to edge k, and the graph-level feature u [39]. In this framework, if only one message-passing step is used, even though the graph is fully connected, each node only aggregates the messages generated from "old" neighbors. A larger number of message-passing steps is needed for a node to aggregate messages sent along a path with a length greater than one.

We conducted the study using nine message-passing steps to ensure that each node can be updated by aggregating the messages along a path of length ten to match the cutoff

length of the *Recurrent Encoder*. The result shows that the model with nine message-passing steps outperforms the model with one message-passing step for all working points; the rejection at 75% efficiency is increased from 405 to 449, although the AUC values remain unchanged. It is worth noting that the performance of the model with more message-passing steps is still worse than the *Recurrent Encoder*.

5.3 Influence of the Sequential Relation

One major difference between the *Recurrent Encoder* and other permutation-invariant encoders is the sequential encoding used by LSTM. We explore the influence of the sequential encoding by comparing the performance of the *Attention Encoder* with and without the Positional Encoding [35].

We find that the *Attention Encoder* with Positional Encoding outperforms the model without Positional Encoding for all working points. The AUC value and rejection power at 75% efficiency of the model with Position Encoding are 0.9926 and 1987, higher than the model without Position Encoding (0.9921 and 1799, respectively). Therefore, the sequential relation is likely beneficial for identifying τ_{had} jets. However, since the difference in the performance is not significant, the sequential relation may not be the only factor that determines the good performance of the *Recurrent Encoder*.

In addition, we further explore the contribution of each node in the sequence by examining the learned attention scores. The score is obtained from the second Multi-Head Attention layer, where the attention is conducted when aggregating node features to global attributes. In other words, the attention score represents the contribution of each node feature to the aggregated global attributes, which are then used for classification. We here extracted the attention score from a sample of 10,000 τ_{had} and QCD jets to visualize the contribution of nodes.

Figure 10 shows that the nodes with different p_{T}, at different sequence positions, contribute to the aggregated global attributes with different attention scores. On the one hand, the high p_{T} tracks are scored higher than those low p_{T} tracks, and the 2nd and 3rd highest p_{T} tracks are considered more critical for τ_{had} than QCD jets. However, the pattern is much less distinguishable for energy towers. And on the other hand, energy towers in the core regions are systematically scored higher than the central and outer regions, as shown in the bottom part of Figure 10. However, the score distributions for tracks do not show a clear pattern in the η and ϕ plane.

5.4 Effects of Pileup

In this section, we explore the effects of additional interactions resulting from pileup by examining models trained with a low-pileup dataset and applied to a high-pileup dataset and vice versa. The low-pileup dataset often leads to fewer tracks and towers inside the jets than the high-pileup dataset, as shown in Figure 11.

The resulting AUC values for the *Heterogeneous Node and Edge Encoder* and *Recurrent Encoder* models are shown in Table 5. Both models see a significant downgrade in the inference AUC values when the testing dataset differs from the training data, indicating poor generalizations of these models. The model trained with a high pileup dataset is

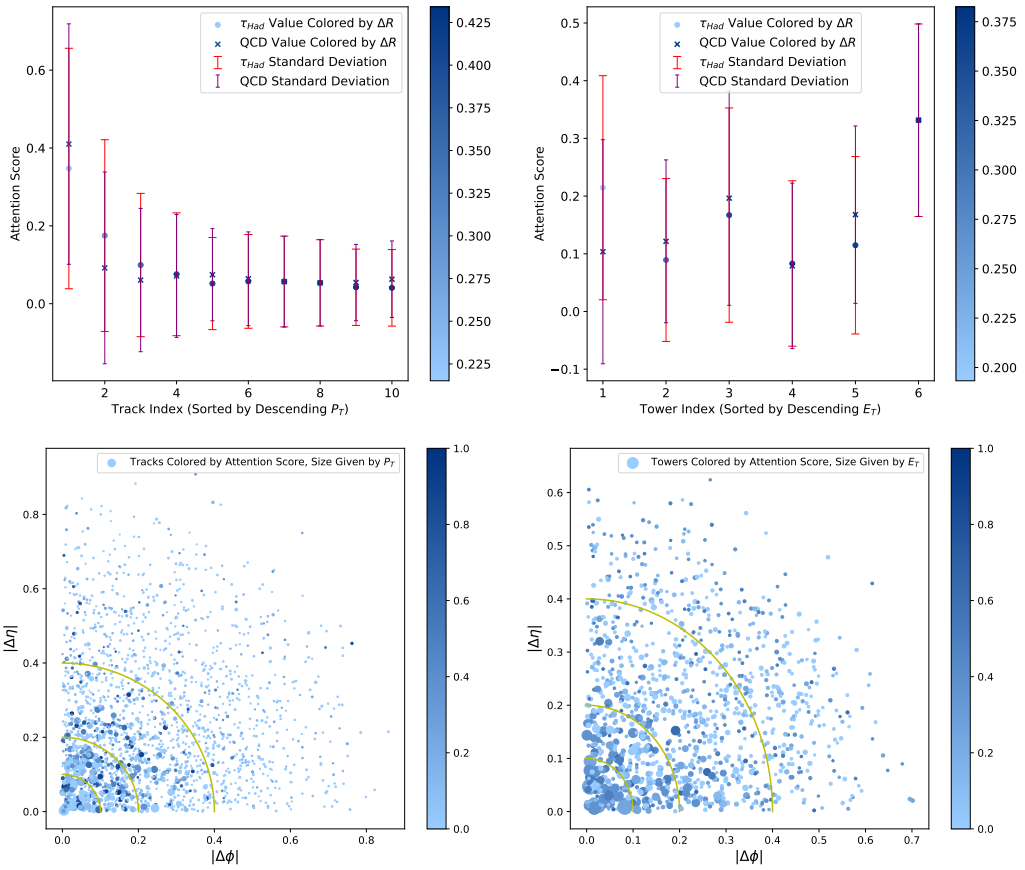

Figure 10. Top row: The mean attention scores of the tracks (left) and towers (right) for 10,000 jets with standard deviations. Bottom: The attention scores of tracks (left) and towers (right) for 200 tau jets; the axes represent the absolute difference in coordinates from the jet axis. Each dot represents a track or tower whose colors represent the attention score, and sizes represent the corresponding P_T or E_T values.

better generalized than that trained with a low pileup dataset. The pileup dependence can be mitigated by training these models with a broader range of pileup conditions or with dedicated pileup suppression techniques, such as the Pileup Mitigation with Machine Learning [40].

6 Conclusions and Outlook

In this study, we present the results of using a heterogeneous Graph Neural Network to identify τ_{had} jet against QCD jets for the HL-LHC. After examining various graph representations and heterogeneous encoders, we found that the jet-level information is essential for model's performance as adding the jet-level features to input graphs improves the model's separation power. While exploring the heterogeneity of the detector data and its integration into the Graph Neural Network, we found utilizing the *Heterogeneous Node and Edge Encoder* architecture results in increased QCD jet rejections in regions of high

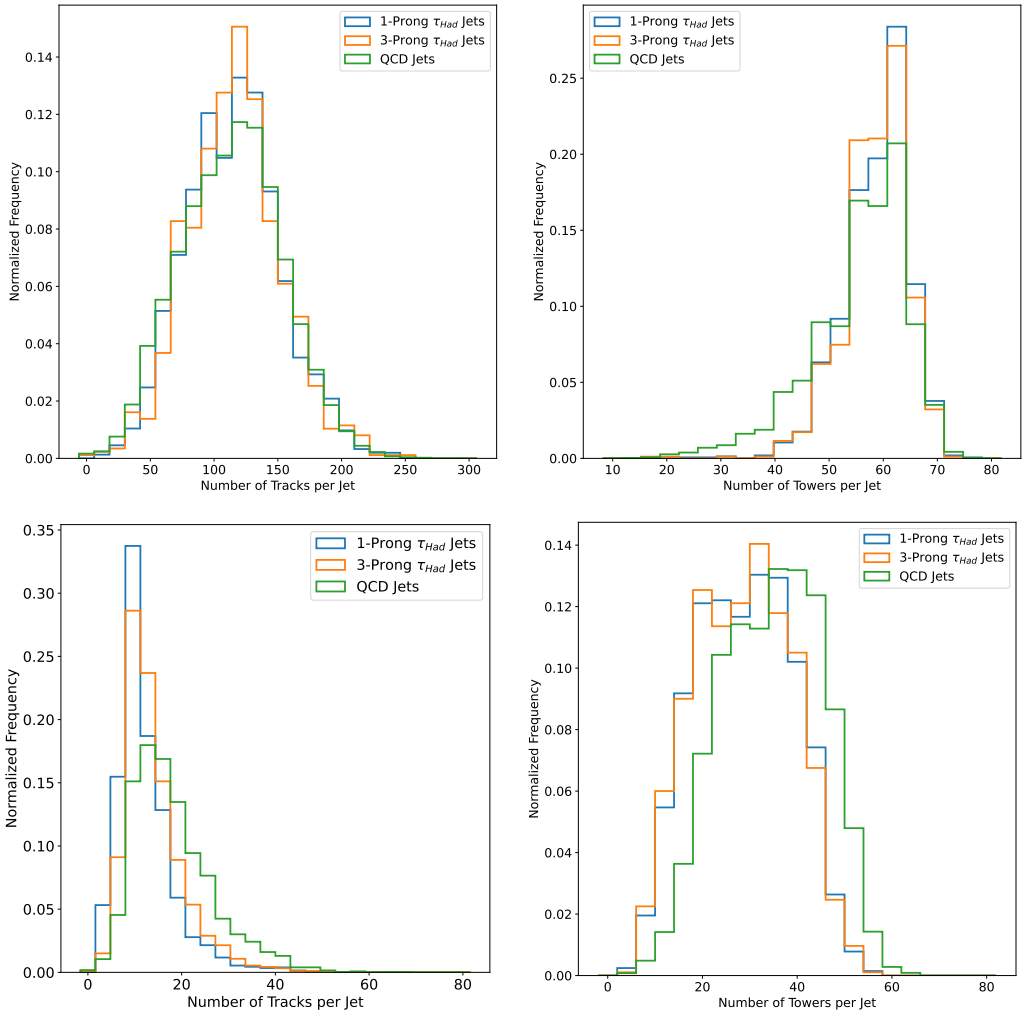

Figure 11. Comparison of the number of tracks and towers of 1-Prong τ_{had} jets, 3-Prong τ_{had} jets, and background (QCD jets). The top row represents the distributions for the $\mu = 200$ dataset, and the bottom row represents the distributions for the $\mu = 40$ dataset. All distributions are normalized to the same area.

τ_{had} efficiency and comparable rejections in regions of lower τ_{had} efficiency, outperforming the *Homogeneous Encoder* architecture overall. Additionally, we observed that sequential encoding outperforms permutation-invariant encodings because high p_{T} tracks and energy clusters in the core region are more important and need to be treated differently.

Acknowledgments

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

Table 5. AUC values for models trained on a high pileup dataset and applied to a low pileup dataset, where μ denotes the average additional proton-proton collisions. The inferences are all conducted on the $\mu = 200$ dataset.

Model	Training Dataset	Inference Dataset	AUC	Rejection at 75% Efficiency
Heterogeneous Node and Edge Encoder	$\mu = 200$	$\mu = 200$	0.9886	448.5
	$\mu = 200$	$\mu = 40$	**0.9804**	**107.3**
	$\mu = 40$	$\mu = 200$	**0.9614**	**32.8**
	$\mu = 40$	$\mu = 40$	0.9900	568.2
Recurrent Encoder	$\mu = 200$	$\mu = 200$	0.9932	4616.7
	$\mu = 200$	$\mu = 40$	**0.9862**	**1033.3**
	$\mu = 40$	$\mu = 200$	**0.9683**	**160.7**
	$\mu = 40$	$\mu = 40$	0.9928	487.1

References

[1] **ATLAS** Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, *JINST* **3** (2008) S08003.

[2] **CMS** Collaboration, *The CMS Experiment at the CERN LHC*, *JINST* **3** (2008) S08004.

[3] **ATLAS** Collaboration, *Evidence for the Higgs-boson Yukawa coupling to tau leptons with the ATLAS detector*, *JHEP* **04** (2015) 117, [arXiv:1501.04943].

[4] **ATLAS** Collaboration, *Search for the Standard Model Higgs boson produced in association with a vector boson and decaying into a tau pair in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector*, *Phys. Rev. D* **93** (2016), no. 9 092005, [arXiv:1511.08352].

[5] **ATLAS** Collaboration, *Test of cp invariance in vector-boson fusion production of the higgs boson using the optimal observable method in the ditau decay channel with the atlas detector*, *Eur. Phys. J. C* **76** (2016), no. 12 658, [arXiv:1602.04516].

[6] **ATLAS** Collaboration, *Cross-section measurements of the Higgs boson decaying into a pair of τ-leptons in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *Phys. Rev. D* **99** (2019) 072001, [arXiv:1811.08856].

[7] **ATLAS** Collaboration, *Test of cp invariance in vector-boson fusion production of the higgs boson in the ditau channel in proton-proton collisions at 13 tev with the atlas detector*, *Phys. Lett. B* **805** (2020) 135426, [arXiv:2002.05315].

[8] **ATLAS** Collaboration, *Measurements of Higgs boson production cross-sections in the $H \to \tau^+\tau^-$ decay channel in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, arXiv:2201.08269.

[9] **CMS** Collaboration, *Observation of the Higgs boson decay to a pair of τ leptons with the CMS detector*, *Phys. Lett. B* **779** (2018) 283–316, [arXiv:1708.00373].

[10] **CMS** Collaboration, *Analysis of the CP structure of the Yukawa coupling between the Higgs boson and τ leptons in proton-proton collisions at $\sqrt{s} = 13$ TeV*, *JHEP* **06** (2022) 012, [arXiv:2110.04836].

[11] **CMS** Collaboration, *Measurement of the inclusive and differential Higgs boson production cross sections in the decay mode to a pair of τ leptons in pp collisions at $\sqrt{s} = 13$ TeV*, *Phys. Rev. Lett.* **128** (2022), no. 8 081805, [arXiv:2107.11486].

[12] **CMS** Collaboration, *Measurement of the Higgs boson production rate in association with top quarks in final states with electrons, muons, and hadronically decaying tau leptons at $\sqrt{s} = 13$ TeV*, *Eur. Phys. J. C* **81** (2021), no. 4 378, [arXiv:2011.03652].

[13] **CMS** Collaboration, *Measurements of Higgs boson production in the decay channel with a pair of τ leptons in proton-proton collisions at $\sqrt{s} = 13$ TeV*, arXiv:2204.12957.

[14] **CMS** Collaboration, *Constraints on anomalous Higgs boson couplings to vector bosons and fermions from the production of Higgs bosons using the $\tau\tau$ final state*, arXiv:2205.05120.

[15] **ATLAS** Collaboration, *A search for high-mass resonances decaying to $\tau^+\tau^-$ in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector*, *Phys. Lett. B* **719** (2013) 242–260, [arXiv:1210.6604].

[16] **ATLAS** Collaboration, *Search for additional heavy neutral Higgs and gauge bosons in the ditau final state produced in 36 /fb of pp collisions at 13 TeV with the ATLAS detector*, *JHEP* **01** (2018) 055, [arXiv:1709.07242].

[17] **CMS** Collaboration, *Search for additional neutral MSSM Higgs bosons in the $\tau\tau$ final state in proton-proton collisions at $\sqrt{s} = 13$ TeV*, *JHEP* **09** (2018) 007, [arXiv:1803.06553].

[18] **CMS** Collaboration, *Search for a heavy pseudoscalar Higgs boson decaying into a 125 GeV Higgs boson and a Z boson in final states with two tau and two light leptons at $\sqrt{s} = 13$ TeV*, *JHEP* **03** (2020) 065, [arXiv:1910.11634].

[19] **CMS** Collaboration, *Search for lepton flavour violating decays of a neutral heavy Higgs boson to $\mu\tau$ and $e\tau$ in proton-proton collisions at $\sqrt{s} = 13$ TeV*, *JHEP* **03** (2020) 103, [arXiv:1911.10267].

[20] **ATLAS** Collaboration, *Reconstruction of hadronic decay products of tau leptons with the ATLAS experiment*, *Eur. Phys. J. C* **76** (2016), no. 5 295, [arXiv:1512.05955].

[21] **CMS** Collaboration, *Identification of hadronic tau lepton decays using a deep neural network*, *JINST* **17** (2022) P07023, [arXiv:2201.08458].

[22] **ATLAS** Collaboration, *Identification of hadronic tau lepton decays using neural networks in the atlas experiment*, tech. rep., Report No. ATL-PHYS-PUB-2019-033, 2019, https://cds.cern.ch/record/2688062, 2019.

[23] J. Shlomi, P. Battaglia, and J.-R. Vlimant, *Graph neural networks in particle physics*, *Machine Learning: Science and Technology* **2** (Jan, 2021) 021001, [arXiv:2007.13681].

[24] J. Duarte and J.-R. Vlimant, *Graph Neural Networks for Particle Tracking and Reconstruction*, arXiv:2012.01249.

[25] T. Sjöstrand, *The PYTHIA Event Generator: Past, Present and Future*, *Comput. Phys. Commun.* **246** (2020) 106910, [arXiv:1907.09874].

[26] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, et al., *An Introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159–177, [arXiv:1410.3012].

[27] **DELPHES 3** Collaboration, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **02** (2014) 057, [arXiv:1307.6346].

[28] M. Cacciari, G. P. Salam, and G. Soyez, *The anti-k_t jet clustering algorithm*, *JHEP* **04** (2008) 063, [arXiv:0802.1189].

[29] M. Cacciari, G. P. Salam, and G. Soyez, *FastJet user manual*, *Eur.Phys.J.* **C72** (2012) 1896, [arXiv:1111.6097].

[30] M. Cacciari, G. P. Salam, and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896, [arXiv:1111.6097].

[31] ATLAS Collaboration, *ATLAS Pythia 8 tunes to 7 TeV data*, .

[32] **NNPDF** Collaboration, *Parton distributions for the LHC Run II*, *JHEP* **04** (2015) 040, [arXiv:1410.8849].

[33] H. Qu, C. Li, and S. Qian, *Particle Transformer for Jet Tagging*, arXiv:2202.03772.

[34] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, et al., *Attention is all you need*, *Advances in neural information processing systems* **30** (2017).

[36] P. T. Komiske, E. M. Metodiev, and J. Thaler, *Energy flow networks: deep sets for particle jets*, *Journal of High Energy Physics* **2019** (2019), no. 1 1–46.

[37] A. Butter et al., *The Machine Learning Landscape of Top Taggers*, *SciPost Phys.* **7** (2019) 014, [arXiv:1902.09914].

[38] W. Zaremba, I. Sutskever, and O. Vinyals, *Recurrent neural network regularization*, *arXiv preprint arXiv:1409.2329* (2014).

[39] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, et al., *Relational inductive biases, deep learning, and graph networks*, arXiv:1806.01261.

[40] P. T. Komiske, E. M. Metodiev, B. Nachman, and M. D. Schwartz, *Pileup Mitigation with Machine Learning (PUMML)*, *JHEP* **12** (2017) 051, [arXiv:1707.08600].