THE EUROPEAN
PHYSICAL JOURNAL C

Regular Article - Experimental Physics

Semi-supervised graph neural networks for pileup noise removal

Tianchun Li[1], **Shikun Liu**[1], **Yongbin Feng**[2,a] (ORCID), **Garyfallia Paspalaki**[1], **Nhan V. Tran**[2], **Miaoyuan Liu**[1], **Pan Li**[1,b]

[1] Purdue University, West Lafayette, IN 47907, USA
[2] Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

Abstract The high instantaneous luminosity of the CERN Large Hadron Collider leads to multiple proton–proton interactions in the same or nearby bunch crossings (pileup). Advanced pileup mitigation algorithms are designed to remove this noise from pileup particles and improve the performance of crucial physics observables. This study implements a semi-supervised graph neural network for particle-level pileup noise removal, by identifying individual particles produced from pileup. The graph neural network is firstly trained on charged particles with known labels, which can be obtained from detector measurements on data or simulation, and then inferred on neutral particles for which such labels are missing. This semi-supervised approach does not depend on the neutral particle pileup label information from simulation, and thus allows us to perform training directly on experimental data. The performance of this approach is found to be consistently better than widely-used domain algorithms and comparable to the fully-supervised training using simulation truth information. The study serves as the first attempt at applying semi-supervised learning techniques to pileup mitigation, and opens up a new direction of fully data-driven machine learning pileup mitigation studies.

1 Introduction

The high instantaneous luminosity of the CERN Large Hadron Collider (LHC) enables studies of the deep mysteries of our universe, such as the nature of the Higgs boson [1,2] and dark matter as well as the origin of the matter-antimatter asymmetry [3]. The enormous amount of data coming from increasingly noisy particle collisions, recorded by more com-plex detectors, poses various challenges to data collection and analysis [4–8]. Multiple collisions in the same or nearby proton bunch crossings lead to overlapping particle interactions, referred to as pileup (PU). To achieve the desired physics sensitivity with the LHC data, the noise from PU particles needs to be identified and mitigated effectively in order to identify signals of interest, i.e., those from the primary interaction of interest, often referred to as the leading vertex (LV). The average number of PU interactions during the LHC data-taking period of 2016–2018 is around 30–40 [9,10]. This is expected to increase in future data-taking periods and reaches around 150 for the high luminosity LHC [11]. Improvements in pileup mitigation techniques can therefore have significant effects on the entire current and future LHC program, through performance gains in the reconstruction of all high-level physics objects which in turn are used in nearly all measurements and searches at ATLAS and CMS.

Particles produced from proton–proton (pp) interactions are reconstructed using the hit information in the tracking detectors and the energy deposits in the calorimeters. Due to the excellent performance of the charged particle tracking systems and their reconstruction algorithms, the track and vertex information of charged particles within the tracker acceptance can be precisely determined [12,13]. Most charged particles associated with PU vertices can be identified and removed from the event. This is often referred to as charged hadron subtraction, and its performance can be found in [4,14]. The remaining challenge of the pileup mitigation task falls therefore mostly on neutral particles, including photons and neutral hadrons.

During data-taking between 2009–2012, most of the developments of pileup mitigation algorithms focused on area-based subtractions [15–17], which correct the physics quantities based on the average pileup density per event. While these methods provide unbiased estimations of jet four-momenta, their resolutions usually become worse and only operate at the level of a whole jet object. More advanced

Tianchun Li, Shikun Liu and Yongbin Feng contributed equally to this work.

[a] e-mail: yfeng@fnal.gov

[b] e-mail: pan.li@ece.gatech.edu (corresponding author)

particle-level algorithms have been developed later on, such as SoftKiller [18], Constituent Subtraction [19], and PUPPI [20]. SoftKiller makes use of the fact that particles from PU vertices tend to have lower transverse momentum (p_T) than the particles from the LV, and applies a p_T cut to remove low-p_T ("soft") pileup particles. Constituent Subtraction is generalized from the area-subtraction methods, creates "ghost" constituents according to the average pileup density, and modify the particle p_T based on the nearby "ghosts". PUPPI, on the other hand, makes use of the neighboring particle information and defines a local shape variable α for each particle. Per-particle weights are calculated based on α, and the particle four momenta are rescaled with their corresponding weights. PUPPI has achieved significantly better performance compared with other methods and has been adopted in many LHC analyses [7]. All of these rule-based algorithms developed and described above do not need labeled simulated data for training, but the parameters in these algorithms need to be carefully tuned based on the real data for each experimental setting.

With the recent rapid developments of machine learning (ML) algorithms, studies [21–24] have been performed applying ML techniques to the pileup mitigation task. These ML-based algorithms adopt convolutional neural network [21], gated graph neural network [22,25], and attention-based models [23,24], to learn complex patterns from the training data and have achieved significantly better performance than the classical domain algorithms in simulation studies. Most of these algorithms require a large amount of LV/PU label information of input particles to get sufficiently trained, termed "fully-supervised" methods. However, such label information for neutral particles is difficult to retrieve in the full Geant-based simulations [26] due to the complications of the showering process in the calorimeters, and does not exist in real collision data. The simulation inaccuracy makes it non-trivial to train and deploy these algorithms to the actual experiments. Dedicated model tunings and precise calibrations are often required, bringing in extra work and systematic uncertainties [7,27,28]. Alternative ways include mixing the special low-pileup data with pileup ones, which requires dedicated low-pileup runs to collect such dataset, and also further studies to understand and verify the performances, e.g., comparing the events in two datasets and creating training labels, and performances with models trained on dataset with mislabels.

The goal of our work is to abandon the previous fully-supervised methods as they rely on the label information of neutral particles. Instead, a novel semi-supervised machine-learning technique (SSL) is applied, taking advantage of the fact that the LV/PU labels of charged particles can still be precisely determined with reconstruction-level information of real collision data. Inspired by the success of PUPPI, our key idea is to capture the effects of neighboring particle fea-

tures on the LV/PU estimation of the target particle, which does not strongly depend on whether the target particle is neutral or charged. To achieve this, we first construct a graph connecting particles close to each other in the physical space, and then train a graph neural network (GNN) using exclusively the LV/PU labels of charged particles. The trained GNN is further applied to neutral particles to estimate the probability of each of them being produced from LV or PU. To avoid the label leakage and the potential bias due to the feature shifts from charged to neutral particles, we propose a random masking technique, which can be viewed as a separate and unique contribution to the adopted machine learning technique itself. The GNN mimics PUPPI in the sense that it explores the neighboring particle features to form a *data-driven* local shape variable for pileup mitigation, fully learned from the real experimental data. It not only aggregates the features in a more expressive way than PUPPI, but also avoids the complex manually tuning procedure of PUPPI.

The effectiveness of this SSL approach is carefully studied and confirmed by comparing the performance of a GNN from fully-supervised training, a GNN with the same architecture but from semi-supervised training, and the domain algorithm PUPPI, in the simulations of different processes and different pileup conditions. DELPHES-based [29] simulation samples are used in order to carry out the fully-supervised training, with more details provided in later sections. It has been found that there is no significant performance drop going from fully supervised training to semi-supervised training, and the GNNs achieve better performance than PUPPI in both cases.

The studies in this paper serve as the first attempt of applying a SSL approach to pileup mitigation studies. This approach does not rely on any neutral particle pileup label information from simulations, and therefore the full workflow can be performed directly on real collision data, without concerns about differences between data and simulation or imperfect choices in the truth labeling. Comparisons are made between techniques using simulated data with truth labels. With promising results, it is worthwhile to study and explore similar approaches in more realistic simulations and real collision data in the near future.

Details of our studies and the results are presented in the following sections. Section 2 provides a brief overview of the previous related works. Section 3 describes the details of the simulation setup and the dataset used. Section 4 presents the methodology of the semi-supervised training technique, the network architecture, and the training setup. Section 5 presents the results, with the performance benchmarks of labeling LV/PU particles, and the subsequently reconstructed physics quantities such as observables of hadronic jets and missing transverse momentum. Section 6 discusses the

results and followup studies. Section 7 summaries the paper with an outlook for future developments.

2 Related work

As briefly introduced in Sect. 1, SoftKiller, Constituent Subtraction, and PUPPI are the three currently widely used pileup mitigation algorithms which operate on a per-particle basis. SoftKiller breaks one event into patches and defines a single p_T cut p_T^cut based on the p_T of the hardest particle in each patch $p_{\text{T},i}^\text{max}$:

$$p_\text{T}^\text{cut} = \text{median}_{i \in \text{patches}}\{p_{\text{T},i}^\text{max}\} \tag{1}$$

Particles with p_T lower than p_T^cut will be marked as pileup and removed from the event in the subsequent reconstructions. Compared with previous area-based pileup mitigation algorithms [15–17,19], SoftKiller operates at the individual particle level and brings significant improvements to the hadronic jet observables, such as mass, p_T, and substructure variables. However, on the other hand, making use of only the p_T information will drop lots of other useful information.

Constituent Subtraction is the generalization of area-based pileup correction methods. It converts the pileup energy density ρ to a set of "ghost" particles with transverse momentum p_T^g using:

$$p_\text{T}^g = A^g \times \rho \tag{2}$$

where A^g is a predefined area. A distance metric $\Delta R_{i,k}$ is calculated among particle i and ghost k:

$$\Delta R_{ik} = p_\text{T}^\alpha \sqrt{(\eta_i - \eta_k^g)^2 + (\phi_i - \phi_k^g)^2} \tag{3}$$

where α is a parameter free to choose, and in some experiments set to 0; η is pseudorapidity and ϕ is the azimuthal angle in collider cylindrical space. Each particle's p_T with a certain $\Delta R_{i,k}$ are corrected based on the comparison with $p_{\text{T},k}^g$.

PUPPI makes use of the information from local neighboring particles. The local shape variable α is calculated according to:

$$\alpha_i = \log \sum_j \xi_{ij} \times \theta(R_\text{min} < \Delta R_{ij} < R_0) \tag{4}$$

where $\Delta R_{ij} = \sqrt{(\Delta\eta_{ij})^2 + (\Delta\phi_{ij})^2}$ is the distance between the neighboring particle j and the target particle i in the $\eta - \phi$ space; the sum j is over neighboring particles in the event with $\Delta R < R_0$ and $\Delta R > R_\text{min}$; $\xi_{ij} = p_{\text{T},j}/\Delta R_{ij}$; and θ is the Heaviside step function. The local shape α is computed per particle, and PUPPI weights are assigned to individual particles accordingly, which describes the probability of each particle being produced from LV. The particle four momenta are rescaled based on the PUPPI weights.

Compared with SoftKiller, PUPPI makes better use of the local neighboring features and the α calculation does not depend on target particles' p_T. But the critical part of the PUPPI algorithm is one *ad hoc*, expert-level metric. There are some parameters that require extensive studies and manual tunings, such as the choice of the cone size R_0, the selection of the neighboring particles, and the metric ξ_{ij}, which can be sometimes changed to $(p_{\text{T},j}/\Delta R_{ij})^2$, etc. Hence, more recent efforts have focused on developing machine learning approaches to automatically learn such combinations.

PUMML serves as the first attempt to apply modern deep learning (DL) techniques for pileup mitigation. It treats collision events as images and particles as pixels in the $\eta - \phi$ grids. With a convolutional neural network applied to extract local features, it achieves better performance compared with PUPPI and SoftKiller. However, representing particles as images requires fixed spatial resolution, which in realistic cases depends on the $\eta - \phi$ positions and can vary dramatically for different types of particles. For sparse events with a limited number of particles in certain regions, flat images would also waste computing resources.

Benefiting from the rapid developments in the DL community, deep sets, graph neural networks, and attention mechanisms are introduced in particle physics [30], such as jet flavor tagging [31,32], calorimeter and event reconstruction [33–35], and also pileup mitigation studies [22–24]. Treating each particle as one unit, these DL architectures do not assume regular detector geometry and can explore much more effectively and efficiently the local structures in collision events. Models including PUPPIML [22], ABCNet [23], and PUMA [24] belong to this category and have shown promising results produced on DELPHES-based [29] simulation data.

Applying these similar architectures on more realistic GEANT-based [26] simulations and real collision data is the next major task. However, it is very challenging to apply such ML algorithms to these more realistic scenarios as the proposed models need full supervision, i.e., being trained with a large number of labeled (LV or PU) neutral particles. The neutral particle pileup label information is hard to be recovered in GEANT simulations and does not exist in real collision data. In order to overcome this challenge and bring these powerful DL models into the realistic deployment and usage for the LHC experiments, we explore the idea of semi-supervised learning, where the training is performed on charged particles, whose LV/PU labels can be determined at reconstruction level for both data and simulations, and the trained model is then applied on neutral particles to estimate their LV/PU probabilities. Note that even though we design and apply our own neural network architecture in Sect. 4, the main focus of this study is not the architecture but rather the semi-supervised training method. The same approach can generally be applied to other network architectures as well if they are believed to have better discriminating power.

3 Datasets

For our studies, simulated datasets have been generated of different physical processes under different pileup conditions. In this study, we select three pileup conditions where the numbers of pileup interactions (n_{PU}) are 20, 80, and 140 respectively, and two hard scattering signal processes, $Z(\nu\nu)$+jets and $H(b\bar{b})$+jets. We study these two signal processes because they include important physics signatures which are affected significantly by additional pileup interactions. In the $Z(\nu\nu)$+jets process, the invisibly decaying Z has the detector signature of missing transverse momentum, p_T^{miss}, and reconstructing this quantity with high fidelity is important across a broad range of LHC analyses. In the $H(b\bar{b})$+jets process, jet objects – collimated sprays of many particles – are produced. Furthermore, the substructure of the jet is also very important for a wide array of applications and both jets and jet substructure reconstruction can be affected by the presence of pileup particles. When studying the performance of our algorithm, we use the resolution of the reconstruction of p_T^{miss} and jet p_T and mass as benchmark metrics. The dataset generation follows a similar setup as in [22], with no detector-level effects taken into account. We have also performed some tests on the DELPHES samples generated using its default CMS card configuration, where different resolutions are applied to charged and neutral particles. The effects are found out to be small and most of the results still hold.

For each signal process, 30K hard scattering events and 30M pileup QCD events are generated separately using PYTHIA 8.223 [36] with 4C tune [37]. The 30K signal events are then randomly divided into 3 groups for different pileup conditions, each with 10K signal events. Particles from the hard scattering signal process are overlaid at particle level with the ones from pileup events, which are randomly selected with n_{PU} following a Poisson distribution, centralized at n_{PU} = 20, 80, and 140, respectively. The overlaying process is done through DELPHES 3.3.2 [29]. Along the beam axis direction, the vertices are randomly distributed following the Gaussian distribution with a spread of 5.3 cm. The allowed maximum spread is 25 cm. No spread in the transverse plan of the beam axis is applied. All particles with p_T > 0.5 GeV are kept in the output particle collection. Checks are done by lowering the minimum neutral particle p_T to 0.1 GeV, where no significant differences are found on the jet and p_T^{miss} performances.

4 Methodology

The details of the problem formulation and the semi-supervised training setup are provided in this section.

4.1 Formulating pileup mitigation as a graph-based SSL problem

Graph-based SSL is a widely-used technique in the ML community to handle the case where training samples (labeled) and testing samples (unlabeled) are connected as nodes in a graph [38,39]. More importantly, the graph structure that connects these nodes also indicates a certain level of labeling information. For example, it is widely used to detect social-group labels of individuals in social network analysis [40–42], where two individuals denoted by two nodes are more likely to be connected if they share the same social-group label. Graph-based SSL effectively combines the graph structure, the labels of training samples, and the features of both training and testing samples all together to predict the labels of testing samples with high accuracy.

The pileup mitigation problem can be naturally formulated as a graph-based SSL problem by utilizing the geometric relationship between charged particles and neutral particles, and the labels of charged particles to make predictions over neutral particles. Specifically, as introduced in Sects. 1 and 2, charged particle labels (LV or PU) can be precisely determined in real experiments for the most part, while neutral particle labels remain unknown and need to be inferred. Because of the short range of hadronization and parton showers, and also the larger boosts from higher-p_T energetic particles, charged and neutral particles from the LV tend to be more localized in certain regions of the $\eta - \phi$ space, while particles from PU are more isotropically distributed. Exploration of such local connections between charged and neutral particles helps identify individual particles produced from LV or PU. Therefore, an effective learning procedure of a model should not only leverage the self features of the neutral particle, but also the features of its neighboring particles, in particular the labels of its neighboring charged particles, which the graph-based SSL is by definition designed for. For each event in our study, we view particles as nodes and connect particles with edges if their distance in the $\eta - \phi$ space is small. Note that building graph formulation also naturally fits the sparse nature of the particles located in the geometric space. Alternative ways such as viewing the data as images in the $\eta - \phi$ space with regular pixels and rounding the locations of particles onto those pixels, often suffer from the rounding error and a granularity selection issue.

4.1.1 Unique ML aspects of the pileup mitigation problem

There are also two fundamental differences between the pileup mitigation problem and a traditional graph-based SSL problem to be noted:

(1) **Graph-level generalization.** Pileup mitigation requires graph-level generalization that traditional graph-based

SSL does not need. Traditional graph-based SSL typically adopts only one single graph, e.g., a social network, to connect all training and testing samples. In pileup mitigation, each event forms one graph consisting of both charged and neutral particles. The obtained model is also expected to be generalized across different types of events (graphs). In our studies, multiple events (graphs) are used to train and test the model.

(2) **Particle-level label usage.** The way to use labels in pileup mitigation is fundamentally different from that in traditional graph-based SSL. Traditional graph-based SSL typically assumes that labels of training samples are only used to supervise the model training and not used as input features of the model. However, in pileup mitigation, the labels of charged particles to supervise the model training are also needed to feed the model as features, because they provide very informative information for the inference of neighboring particles. If we use the labels of all charged particles both as the input features of the model and to supervise the model training, the obtained model cannot be applied to the inference of neutral particles as neutral particles do not have such labels. To address such a problem, we propose a random masking strategy, where we randomly mask the charged particles to decide whether their labels are used to supervise the model training or as input features. The detailed masking process will be discussed in the following section.

4.2 Detailed approach

The four steps of the developed approach to train the model are provided in Fig. 1.

(1) Graphs are constructed on an event-by-event basis where each node in the graph is one particle.
(2) A random selection and masking of charged particles are carried out.
(3) A GNN is applied to aggregate neighboring features and update the node representations.
(4) The LV/PU prediction is computed based on the final node representations.

Details are documented in the following subsections. It is worth pointing out that although the GNN takes the entire graph as input, only those selected and masked charged particles will be used to supervise the model training, specifically for computing the loss function, performing backward propagation, and optimizing the model parameters. At the inference stage, the masking procedure is excluded and the inference is conducted on all neutral particles. We explain the details of the approach as follows.

4.2.1 Graph construction

One graph is constructed per event to establish the relations between particles and their neighbors. Particles are viewed as nodes and two particles are connected if their distance in the $\eta - \phi$ space, $\Delta R = \sqrt{(\Delta \eta)^2 + (\Delta \phi)^2}$, is smaller than a certain threshold R_0. A smaller threshold would result in a sparser graph, easier to compute but with less neighbor
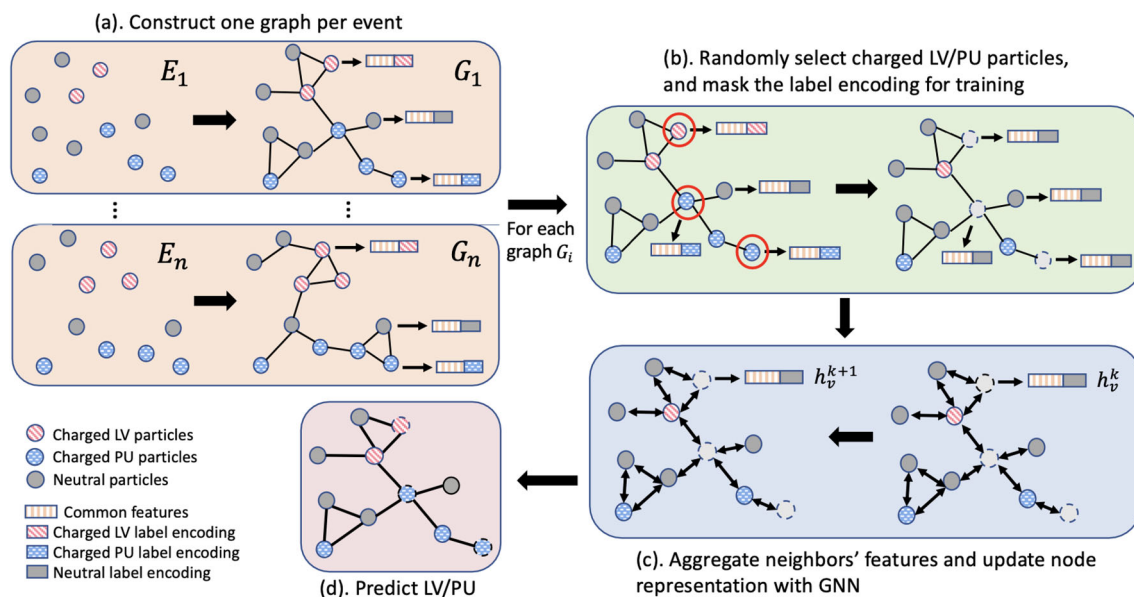

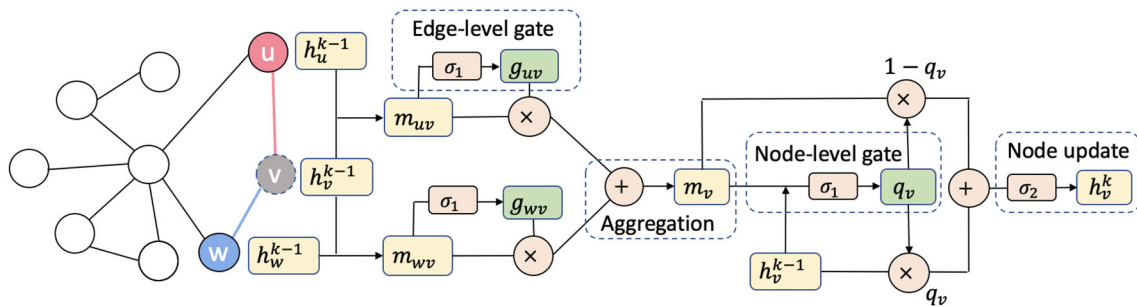**Fig. 1** A diagram illustrating the SSL model training flow

Fig. 2 Node representation h_v update for the kth iteration

Table 1 Message passing and representation updates in the graph neural network

Message formulation:	$m_{uv} = \left[h_u^{k-1}, h_v^{k-1}, \Delta\eta_{uv}, \Delta\phi_{uv}, \Delta R_{uv}, h_g^{k-1} \right]$ (5),
Aggregation:	$m_v = \sum_{u \in N(v)} g_{uv} m_{uv}$, where $g_{uv} = \text{Sigmoid}(W_1 m_{uv} + b_1)$ (6)
Node-level gate:	$q_v = \text{Sigmoid}(W_2[h_v^{k-1}, m_v] + b_2)$ (7)
Node update:	$h_v^k = \text{ReLU}(q_v(W_3 h_v^{k-1} + b_3)) + (1 - q_v)(W_4 m_v + b_4))$, (8)

information, while a larger threshold would result in a denser graph, with more neighbor information but more computing-intensive. $R_0 = 0.4$ is chosen in this study.

4.2.2 Graph neural networks

Graph neural networks (GNN) have shown to be one powerful tool for graph-based SSL [39]. GNN first associates each node with a node representation based on the initial node features and then updates node representations by aggregating and combining with the representations of the neighboring nodes. One widely-used GNN model is GraphSage [43], which takes an average of neighboring node features for node representation update. Several previous works in applying machine learning techniques for pileup mitigation also chose GNN as their models [22,24].

Even though many varieties of GNN models can be applied to our framework in pileup mitigation, we focus on using a variant of the gated GNN model [44]. Since there are certain scenarios where LV particles are surrounded by PU particles, the gated GNN model can automatically learn the gates to control the aggregation of neighboring particles' representations. In contrast, the GraphSage model does not have such control when averaging the representations of the neighbors.

Let h_v^k denote the node v representation at k-th layer. Our gated GNN model, as shown in Fig. 2, is formulated as shown in Table 1: where $\Delta\eta, \Delta\phi, \Delta R$ are the geometric features that characterize two particles' spatial coordinates differences η, ϕ, and distance $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$, and h_g is a global node, which is calculated as the average of all node representations in one graph. The node representations are initialized as particle features that in our studies include the

particle transverse momentum p_T and one-hot label encoding, that is, (1, 0, 0) for PU charged particles, (0, 1, 0) for LV charged particles, (0, 0, 1) for neutral particles and masked charged particles, where the procedure of masking charged particles will be introduced in Sect. 4.2.3. For a target node v, in Eq. (6), $g_{uv} \in [0, 1]$ is a weight learned for each neighboring node $u \in N(v)$ to control the amount of information that is passed to v. In Eq. (7), another gate $q_v \in [0, 1]$ controls the portion between the representation at $(k - 1)$-th layer h_v^{k-1} and the aggregation from the neighbors m_v, when formulating the new node representation h_v^k in Eq. (8). The node representations of the selected particles in the final layer of the GNN are put through a multi-layer perceptron [45] with two hidden layers to make the final prediction.

4.2.3 Masking charged particle and random selection

The primary goal of masking a subset of charged particles is to make the model leverage the labels (LV or PU) of charged particles in two different ways simultaneously. On the one hand, the masked charged particles are used to supervise the model training, with the expectation that the model trained on these charged particles can be applied to infer the labels of neutral particles in the later testing stage. Therefore, the features of these charged particles in the training should mimic the ones of neutral particles, and their LV/PU labels should not be used as the input features. On the other hand, the LV/PU label information of neighboring charged particles serves as important inputs for predicting the labels of target particles. Thus the label information of neighboring particles should be kept in the inputs.

Note that such masking procedure is at risk of breaking the original structure of the data and thus may introduce biases.

Table 2 The first four columns include the mean and variance of four types of particles under different pileup conditions per graph. The last two columns indicate the number of charged particles being randomly selected for training per graph in one epoch

n_{PU}	# Particles (in total)				# Selected particles (for training)	
	Charged LV	Charged PU	Neutral LV	Neutral PU	Charged LV	Charged PU
20	69 ± 34	407 ± 154	43 ± 22	203 ± 80	7	35
80	67 ± 33	1630 ± 310	42 ± 22	813 ± 160	7	126
140	68 ± 34	2846 ± 408	43 ± 22	1423 ± 213	7	224

To reduce such bias, our model only masks a small portion of the charged particles per event. However, masking only a small portion of charged particles for training may not sufficiently leverage the labels. To achieve a better usage of the labels, we propose the *random selection mechanism*. That is, for each event, we perform multiple-time random selections of the charged particles for masking. This guarantees that for each event each time, only a small portion of charged particles are masked and used to supervise the model training, while most of the charged particles of this event can be eventually used to supervise the model after running the model multiple times on this event.

Another practical consideration is regarding the time complexity of model training. Although the random masking strategy guarantees a sufficient usage of the labels and the data, setting the masking portion too small may slow down the training procedure, because a huge time of model running per event is needed to guarantee a good coverage of the masking procedure. To balance the tradeoff, in our experiments we randomly select about 10% of charged particles per event each time. Table 2 includes the numbers of selected charged LV and PU particles per graph per epoch and the total number of charged LV and PU particles per graph. With about tens of training epochs, all charged particles should be selected as training data at least once by random selection. Even though different pileup levels seem to affect the actual numbers of selected particles greatly, experiments show that the model is robust when it is trained on one pileup level and tested on another pileup level.

4.3 Training details and complexity

We also compare our SSL model with a model that has the same architecture but is trained using fully supervised learning (SL), i.e, using the labels of neutral particles without any masking strategy. The SL model needs to be trained and tested over different events, though our SSL model does not need to. To make fair comparisons, for experiments where $n_{PU} = 80$, there are 3000/1000/1000 events for training/validation/testing. When $n_{PU} = 140$, 2000/800/800 events are used for training/validation/testing. For the $n_{PU} = 140$ scenario, there are more particles per event, so the total number of events is reduced to maintain reasonable memory

usage. To save graph construction time for random masking, a masking vector is implemented to efficiently mask the charged particles for training each epoch. The vector can be easily altered to mask another set of training particles without constructing a new graph entirely. During training, the model is trained until convergence, which normally takes about 5 times running over all the events for training. The total number of parameters is around 1300 and can be trained within 6 hours on one NVIDIA Tesla V100 or P100.

In order to reduce the training complexity, we construct graphs by only connecting particles with $\Delta R \leq 0.4$. This restriction in ΔR, in this case, will make the entire graph sparse to reduce the time for graph construction, training, and inference. The graph construction time is approximately 0.1 s per event (per graph) and the inference time is about 30 ms for a graph with $\Delta R \leq 0.4$ and about 50 ms for a graph with $\Delta R \leq 0.8$ at $n_{PU} = 80$. The inference time becomes longer if we increase ΔR when constructing the graph.

5 Results

Experiments are carried out to verify the effectiveness of the model trained via SSL and its ability to be adapted to different n_{PU} levels. The performance with PUPPI, semi-supervised training, and supervised training are compared in this section. Firstly, we examine the performance at particle level, using the receiver operating characteristic (ROC) curves and the area under the ROC curve (AUC) scores trained and tested under different pileup conditions. Then, the performance of physics observables, such as the hadronic jet mass and p_T, and also the missing transverse momentum p_T^{miss}, are studied and compared among the three approaches. Finally, some event display examples are provided, visualizing the differences and improvements of (semi-)supervised results with respect to PUPPI.

5.1 Performance at particle level

Figure 3 shows the ROC curves and Table 3 lists the area under the ROC curve (AUC) score trained and tested in different n_{PU} conditions. These plots show the performance at a per-particle level of the LV and PU labels. In all these cases,
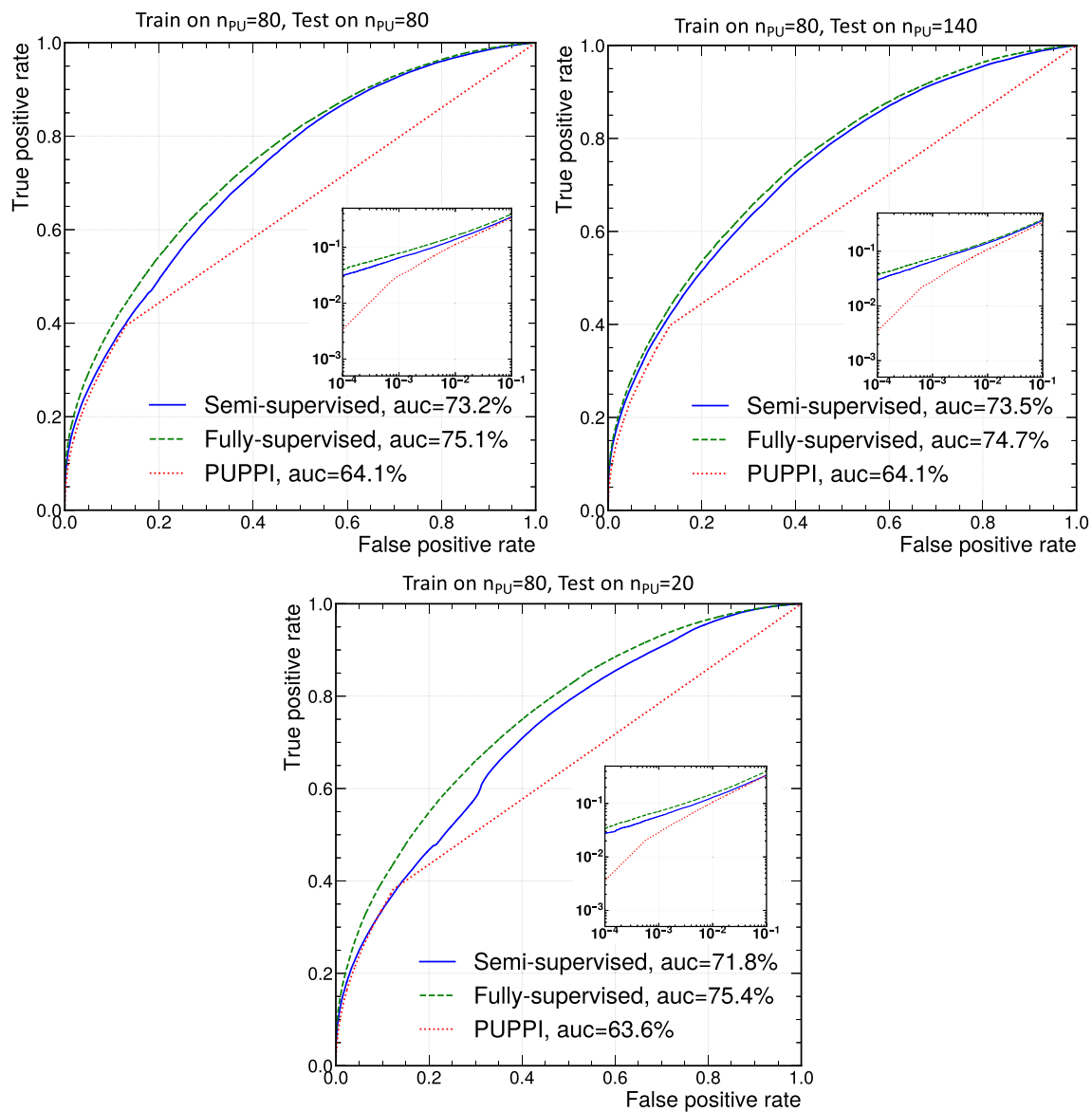
Fig. 3 The ROC curves for the gated GNN on neutral particles under $n_{PU} = 80$ for SSL, fully-supervised learning, and the domain PUPPI algorithm. The small plot inserted under the ROC curve is the log scale of the lower left region of the ROC curve for better visualization

when training and testing at the same pileup level, both SL and SSL outperform PUPPI by around 10%, and the performance decrease from SL to SSL is within a few percent. An inset shows the plot in the log-log scale where smaller false positive rates are important given the much larger number of PU particles compared with the number of LV particles in one event. How the per-particle performance is manifested in physics object performance will be explored below.

When training and testing under different pileup conditions, SL models seem to be more robust to different pileup conditions, i.e., the AUC scores are more consistent across different training and testing conditions, whereas the SSL model is more sensitive to the pileup condition. An interesting observation is that the SSL model performs well when

extrapolating to a higher n_{PU} condition, but not the reverse order. It is interesting to further improve the generalization capability of the model across different pileup conditions. We leave it as a future research direction.

5.2 Performance on jet observables

The GNN model output, which is an N-dimensional array of float numbers between 0 and 1 (N is the total number of particles per event), can be interpreted as the probability of how likely each corresponding particle is produced from the LV. Similar to the approach adopted in the PUPPI algorithm, the four-momenta of all particles are rescaled with the corresponding GNN outputs. The jets are then clustered with the

Table 3 AUC scores (%) of the PUPPI algorithm, supervised (SL) and semi-supervised (SSL) models trained and tested on three different pileup conditions: $n_{PU} =20$, 80, and 140

Training testing	Model type	$n_{PU} = 20$	$n_{PU} = 80$	$n_{PU} = 140$
$n_{PU} = 20$	PUPPI	63.6	64.2	64.1
	SL	76.6	75.5	74.3
	SSL	73.9	70.7	67.6
$n_{PU} = 80$	PUPPI	63.6	64.1	64.1
	SL	75.4	75.1	74.7
	SSL	71.8	73.2	73.5
$n_{PU} = 140$	PUPPI	63.7	64.2	64.1
	SL	69.5	75.2	75.0
	SSL	53.0	71.7	72.9

particle rescaled four-momenta using the anti-kt jet clustering algorithm [46], with the radius parameter R chosen to be 0.7 to be consistent with previous related work [22]. Jets clustered with the generator-level LV particles serve as the ground truth information for comparison.

We study the leading jet in the event with truth p_T above 20 GeV in the H($b\bar{b}$)+jets sample. Because this process is inclusive, the typical jet p_T is approximately 60 GeV, nearly half the Higgs mass. Figure 4 shows the reconstructed jet mass and p_T resolutions with respect to the truth-level jets for the scenario where $n_{PU} = 80$. Resolutions are defined as $q_{83} - q_{14}$, where q_{83} and q_{14} are the 84% and 14% quantiles. We study reconstructed jets which are within $\Delta R = 0.1$ of a truth-level jet. Compared with PUPPI, the bias and resolution of jet masses and p_T clustered with both the semi-supervised and fully-supervised algorithms are significantly smaller. This indicates that the GNN approach does both a better job in predicting the overall aggregate p_T of the jet object, and also its substructure using the jet mass metric. Improvements over PUPPI are comparable to other DL approaches using GNNs [22]. Compared with the SL approach, the performance drop of the SSL approach is relatively small. These

are consistent with the per-particle performance results and show the improvements provided by the SL and SSL models. The results are also consistent across different pileup scenarios.

Figure 5 shows the relative jet mass and jet p_T resolutions as a function of n_{PU}, where better resolutions can be observed across all the tested n_{PU} conditions.

5.3 Performance on missing transverse momentum

We study the missing transverse momentum (p_T^{miss}) resolution performance of our algorithm using Z($\nu\nu$)+jets events. The p_T^{miss} is the negative vector sum of the particles in the event and are calculated with the rescaled four-momenta of all particles. We compare the SL and SSL approaches with the performance of PUPPI and the results are shown in Fig. 6. Compared with PUPPI, the resolution is significantly better ($\sim 20\%$), with some minor deviation in the mean value from zero. This can potentially be due to the SSL misidentifying some LV particles as PU ones, and removing these from the LV collection. The bias nevertheless is small and can be mit-
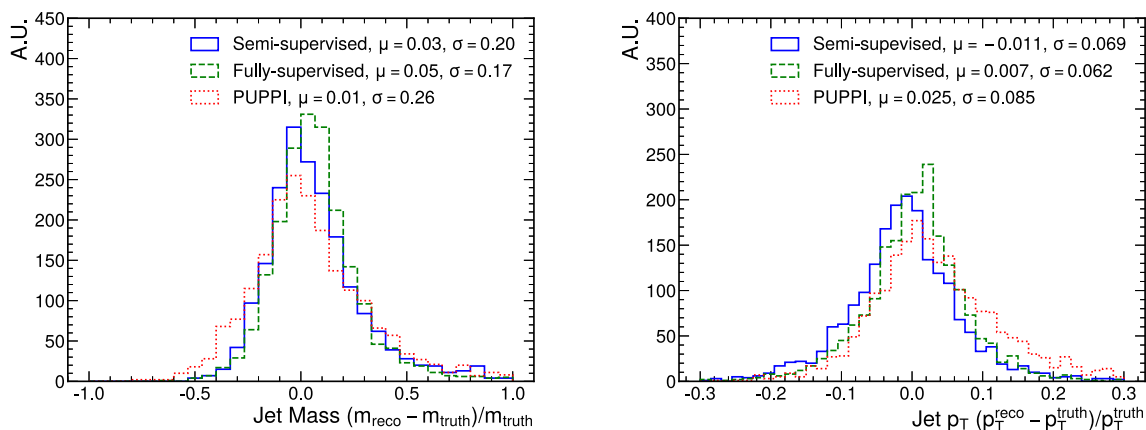

Fig. 4 Performance on jet mass and jet p_T with different pileup mitigation techniques for $n_{PU} = 80$
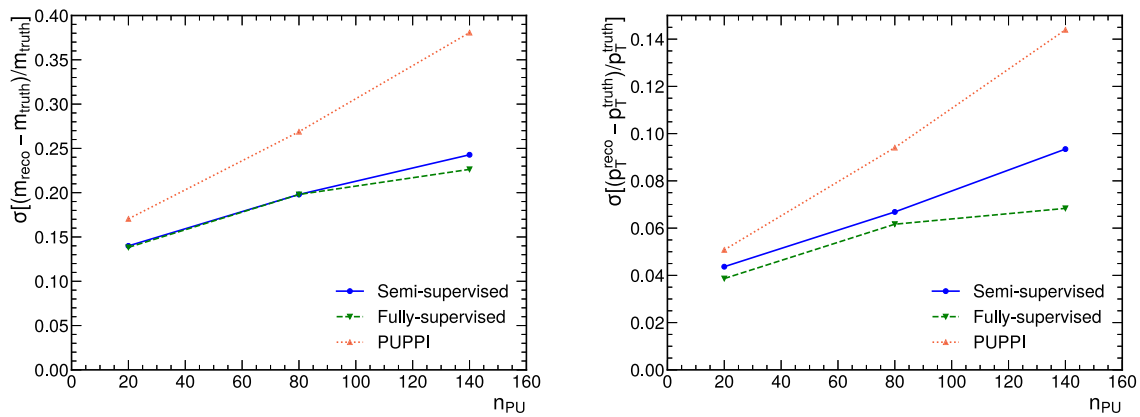
Fig. 5 Resolutions of jet mass (left) and jet p_T (right) as a function of n_{PU}
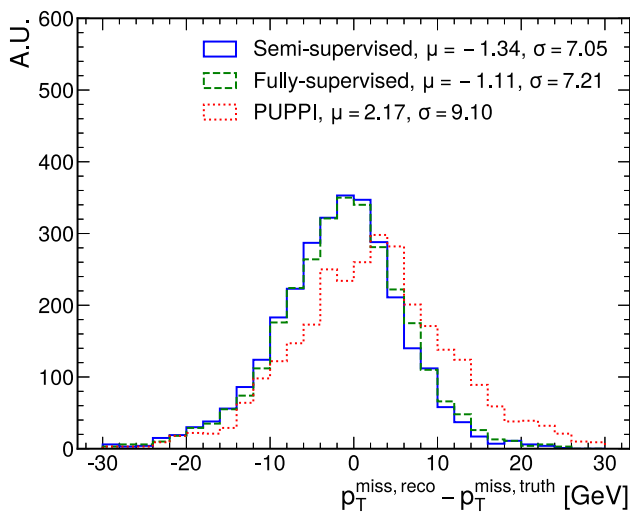

Fig. 6 Resolution of the missing transverse momentum of events for the different pileup mitigation models for $n_{PU} = 80$

igated via offline calibrations. The results are also consistent across different pileup scenarios.

5.4 Event visualization

Figure 7 provides one event visualization of the particle distributions in the $\eta - \phi$ space with different pileup mitigation algorithms: PUPPI (top right), SL (lower left), and SSL (lower right). The marker size scales with the particle p_T. It can be observed that while PUPPI leaves some PU remnants, both SL and SSL models clean the PU particles more efficiently while preserving the LV particles.

In summary, the performances of all these particle-level metrics and physical observables are consistent and show the improvements of the SSL models with respect to the currently widely used domain algorithm PUPPI. Compared with the traditional SL approach, the performance decreases with the novel SSL approach is negligible. However, the SL approach

cannot be directly applied to the real experimental data while the SSL approach can. The trainings and evaluations are currently all performed on the DELPHES-based simulation data, and their effectiveness will be carefully re-examined on the GEANT-based simulation and real collision data in future studies.

6 Discussion on the connections with PUPPI

This section briefly discusses the connections between the Graph (S)SL model and PUPPI, in both the algorithm design and outputs, in order to help provide some insights into what the model learns and where further improvements could be realized.

6.1 Model design and understanding behavior

As briefly mentioned in previous sections, the GNN model architecture is designed to mimic the α calculation in PUPPI, with trainable parameters directly learned from data that can be more expressive and powerful. The similarities and differences between PUPPI and the GNN model are compared in detail here:

(1) *Targets particle self features*. PUPPI does not use particle self features while the GNN model does. In PUPPI, only the neighboring particle features are included when determining PUPPI weights, whereas in the GNN model both the target and the neighboring particle information are used in Eq. (8). This is potentially very useful in some practical cases: for example, for high-p_T particles which are highly likely to be produced from the LV, PUPPI usually requires one additional step to manually assign high weights but the GNN is expected to handle these well automatically.
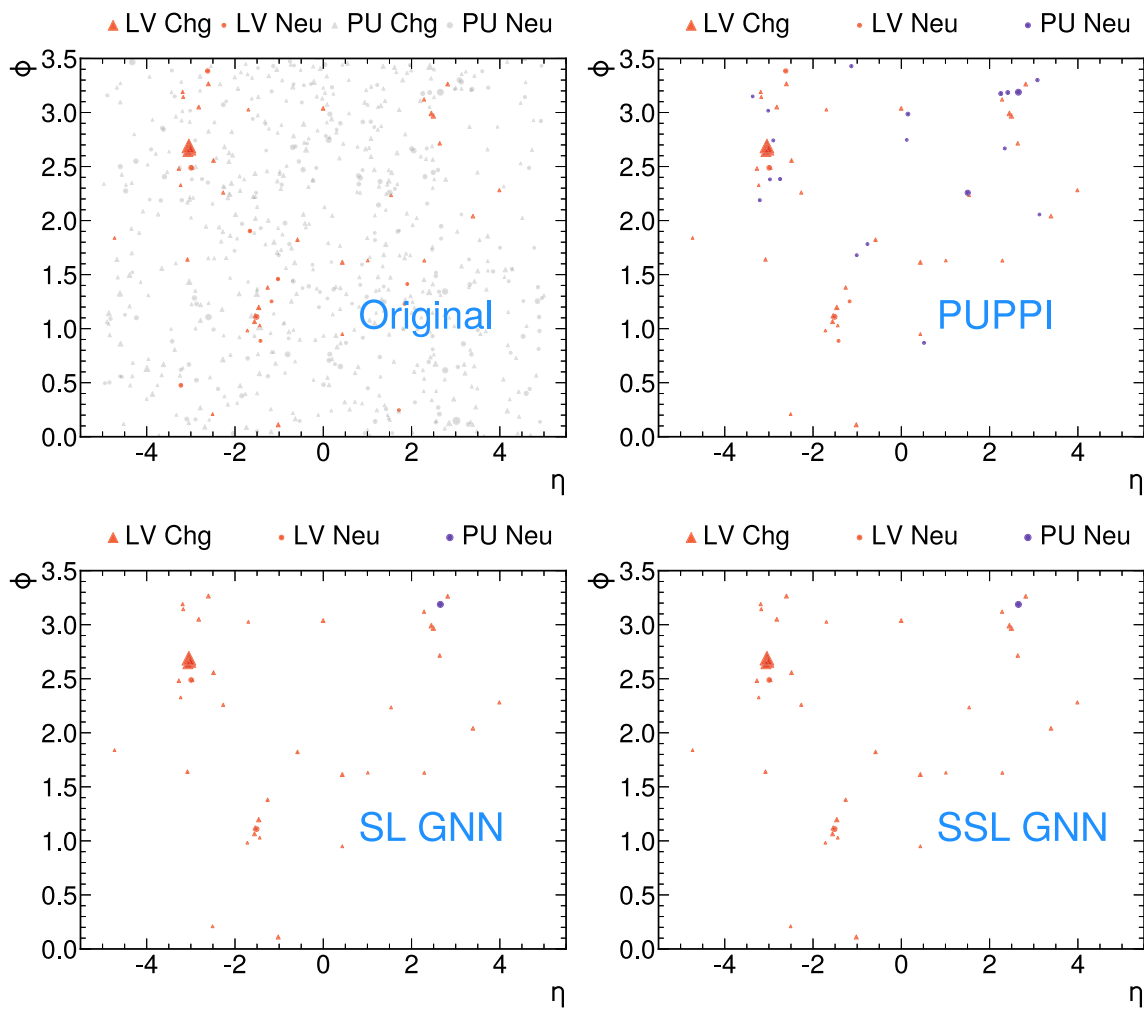
Fig. 7 Some event display examples of the particle distributions in the $\eta - \phi$ space. The upper left plot is using ground truth information. The upper right plot is after applying PUPPI. The lower left and right plots are after applying SL and SSL, respectively

(2) *Selections/Gates to remove noise.* When aggregating information from neighboring particles, different selection criteria can be applied to remove noisy information and keep only the useful ones. Within the tracker acceptance, PUPPI uses all the neighboring charged LV particles. In the GNN model, Eq. (6) does a similar job - the gate $g_{\mu\nu}$ is applied to determine the weight (importance) of the neighboring information, and therefore the noisy information can be reduced.

(3) *Choice of metric.* For the neighboring particles passing the selection, PUPPI utilizes their p_T and the ΔR distance with respect to the target particle, and defines the metric $p_{T,j}/\Delta R_{ij}$. Different metric options, such as $(p_{T,j}/\Delta R_{ij})^2$, or $p_{T,j}^2/\Delta R_{ij}$ were also studied in the PUPPI developments. Claiming what is the best choice is ad-hoc and takes a lot of human labor. This is avoided in the GNN model as more information is the neighboring particles are included in the inputs, such as the p_T, η, $\Delta\eta$, $\Delta\phi$, ΔR, and the metric with more compli-

cated and powerful forms can be learned inside the GNN model.

(4) *Generalization of PUPPI.* The Graph SSL model can also be viewed as a direct generalization of PUPPI because both of them learn or tune their parameters by only using charged particles whose labels are available in real collision data. In contrast, the Graph SL model needs extra labeling information from neutral particles.

Figure 8 shows the p_T-weighted ΔR distribution of the LV (left) and PU (right) particles in the proximity of one truth-level jet in H($b\bar{b}$)+jets events. The ΔR is calculated between the particle direction and the associated jet axis, and the particle p_T is normalized to the truth-level jet p_T and served as the weight for each entry in the two histograms.

From the right plot, it can be observed that PUPPI in general removes around 50% of the PU particles, while the SSL model removes around 75% of the PU particles. From the left plot, compared with PUPPI, in the central region (small ΔR)
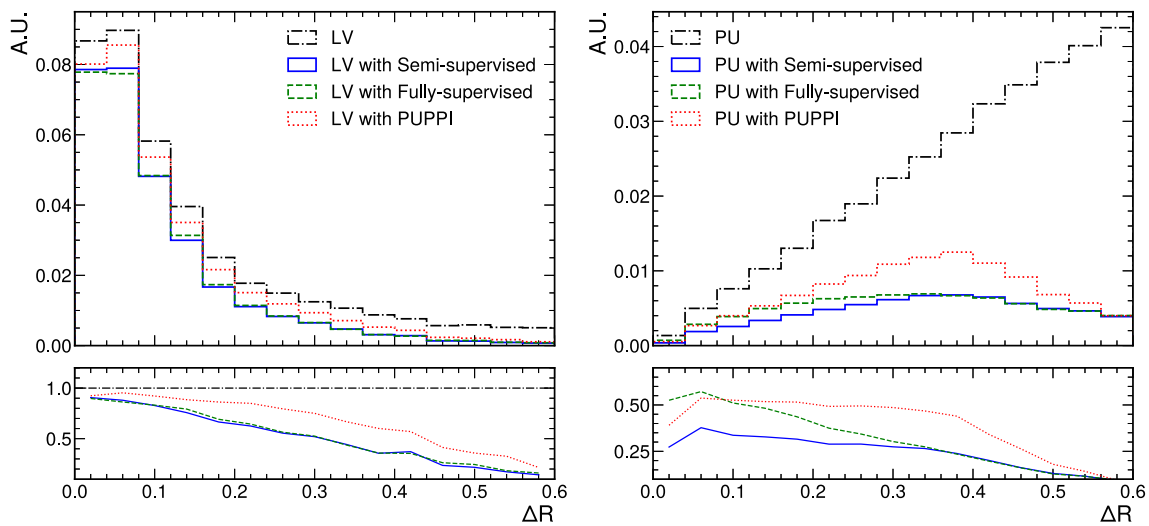
Fig. 8 The p_{T}-weighted ΔR distributions of the LV (left) and PU (right) particles locally to a truth-level jet

where most of the LV particles exist, SL and SSL models keep similar amounts of LV particles as PUPPI. In the region far from the jet axis (large ΔR) where fewer LV particles exist, both SL and SSL models remove more LV particles than PUPPI. From both these plots, it is clear that generally the Graph SL and SSL models are more aggressive in removing particles at the edge of the jets than PUPPI which leads to improved physics performance. However, this also indicates areas of potential further improvements.

6.2 Output comparison

We would also like to further explore the model outputs and directly compare them to PUPPI outputs. Figure 9 shows the outputs of the GNN model (left) and PUPPI weights (right) for neutral particles. For the pileup neutral particles, most of them get a score close to 0. For the LV neutral particles,

a fraction of them get assigned a score close to 1, correctly identified as LV particles, while there are still some assigned a weight close to zero, indicating spaces for future improvements. In general, the Graph SL and SSL models tend to create a more gradual assignment of LV-like vs PU-like where the weights are bunched more towards 0 or 1 whereas PUPPI will say either definitely the particle is PU or else gives a much more uniform probability. The effect is exacerbated in the case of SL vs. SSL where the SL model tends to give some particles a weight closer to 0.5. However, given that the plots are presented with a log y-axis scale, these are generally a small fraction of the overall particles.

Figure 10 shows the GNN model outputs with respect to the PUPPI weights (left) and the neutral particle p_{T}(right). The correlation between GNN outputs and PUPPI weights is not strong, where most of the particles with high PUPPI weights still get relatively small GNN outputs. On the right
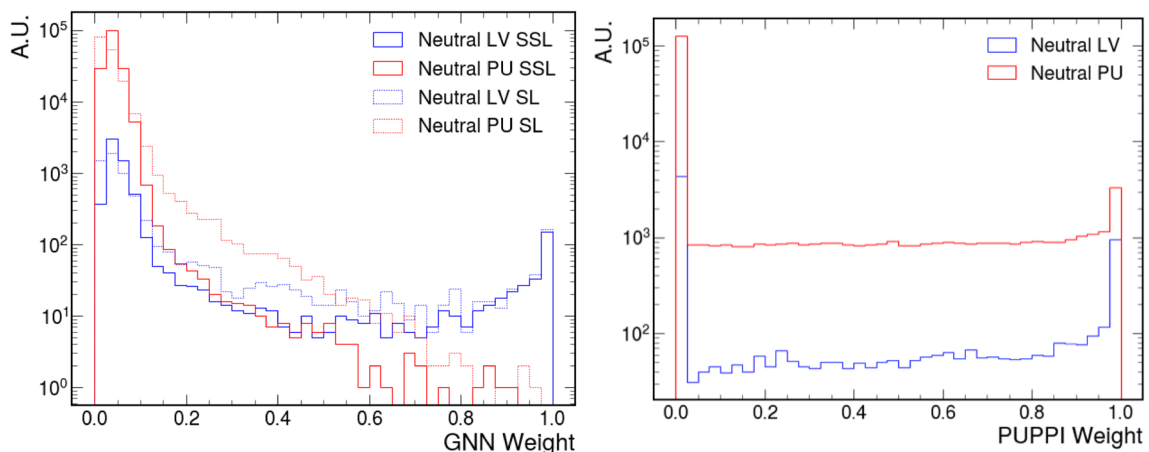
Fig. 9 GNN weight (left) and PUPPI weights (right) on neutral particles
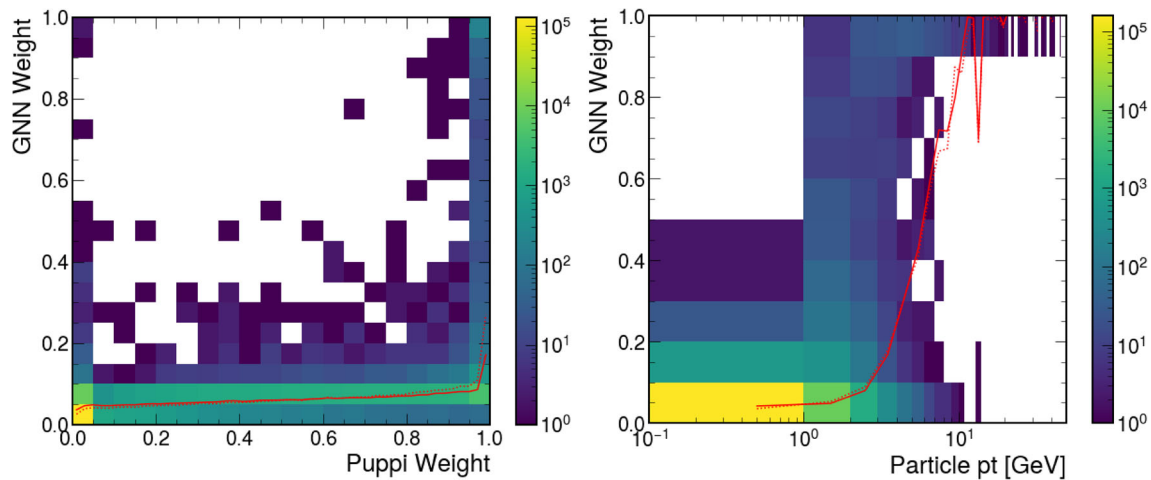
Fig. 10 GNN output with respect to the PUPPI weights (left) and the p_T of neutral particles (right)

plot, for particles with p_T below 1 GeV, most of them get assigned weight close to 0; as p_T goes higher, the weight increases, more likely to be produced from the LV.

In summary, it can be observed that the GNN models tend to be more powerful at exploring the high-dimensional input feature space and more efficient in removing pileup noises, especially in the regions with less LV activities (i.e., large ΔR with respect to the LV jet axes). While the LV particles with high p_T or close to the jet axes are preserved by the GNN model, it is less efficient for keeping the LV information far from the jet axes. This can be further studied and potentially improved in future studies.

7 Summary and outlook

This paper presents the first study of semi-supervised ML techniques with a graph neural network for the pileup mitigation task. The task is cast naturally as a graph learning problem, the training is performed on labeled charged particles, and the inference is evaluated on unlabeled neutral particles. This is performed through a careful feature masking process which trains on charged particles *as if* they were neutral particles. By approaching pileup mitigation as a semi-supervised learning problem, we can train from the data and avoid complicated issues arising from (a) data and simulation differences for soft and hard-to-model physics and (b) labeling neutral particles which is inherently challenging given the relatively poor spatial and energy resolution from detecting neutral particles. Compared with PUPPI, the Graph SSL algorithm is more powerful at removing pileup particles, while maintaining the leading vertex particle information. Improvements are observed at the particle-level LV/PU identification and physics observables such as jet p_T and mass, and p_T^{miss}.

This study serves as a proof of concept, with promising and extensive future studies planned to apply this technique to train directly on real collision data, without any dependence on the ground-truth labeling information. In such cases where the forward region has no tracking information, the momentum and spatial resolutions are expected to be worse than the central ones. We believe that transfer learning techniques can be explored to properly apply the training in the central region to the forward region, and to mitigate the potential larger differences between charged and neutral particles in more realistic scenarios. We show that treating the pileup mitigation task as one that can be machine-learned from data with minimal dependence on simulation is particularly promising and opens up a number of new and interesting challenges for research.

Appendix A: Additional material

We emphasize that the merit of this paper lies in demonstrating the possibility and effectiveness of the semi-supervised learning approach for the pileup mitigation task. Therefore the studies in the main contents of this paper are carried on an idealized configuration. While working on studying the performance of this approach with Geant-based full simulations, we have also verified the performance on DELPHES samples with the resolution effects included. We use the default DELPHES v3.5.0 CMS card, where the momentum resolutions for charged particles and neutral ons are different. We followed the same strategy to train and test the model (The main architecture is the same and the semi-supervised approach is unchanged, with more convolutional operations and more depth in the GNN part to capture more features). Figure 11 shows the ROC curve of the test performances on the neutral particles and the jet mass distributions on the $Z(\nu\nu) + $ jets samples. As can be seen, the performance with the semi-supervised training approach does not degrade significantly from the fully-supervised one, and is consistently better than PUPPI performance. Therefore, we can conclude that our semi-supervised approach is still effective on the

DELPHES simulation samples with different detector effects on charged and neutral particles.

As discussed in the main contents, the effectiveness of this semi-supervised approach and the performances still need to be carefully evaluated on Geant-based full simulations and the real collision data, which is currently undergoing and will be discussed in the future studies.

References

1. G. Aad et al., Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. Phys. Lett. B **716**, 1–29 (2012). https://doi.org/10.1016/j.physletb.2012.08.020. arXiv:1207.7214 [hep-ex]
2. S. Chatrchyan et al., Observation of a new boson at a mass of 125 GeV with the CMS Experiment at the LHC. Phys. Lett. B **716**, 30–61 (2012). https://doi.org/10.1016/j.physletb.2012.08.021. arXiv:1207.7235 [hep-ex]
3. S. Ritz, et al., Building for discovery: strategic plan for US particle physics in the global context (2014). https://www.osti.gov/biblio/1320565
4. G. Aad et al., Jet energy scale and resolution measured in proton proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. Eur. Phys. J. C **81**(8), 689 (2021). https://doi.org/10.1140/epjc/s10052-021-09402-3. arXiv:2007.02645 [hep-ex]
5. M. Aaboud et al., Identification and rejection of pile-up jets at high pseudorapidity with the ATLAS detector. Eur. Phys. J. C **77**(9), 580 (2017). https://doi.org/10.1140/epjc/s10052-017-5081-5. [Erratum: Eur. Phys. J. C 77, 712 (2017)]. arXiv:1705.02211 [hep-ex]
6. M. Aaboud et al., Performance of missing transverse momentum reconstruction with the ATLAS detector using proton–proton collisions at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **78**(11), 903 (2018). https://doi.org/10.1140/epjc/s10052-018-6288-9. arXiv:1802.08168 [hep-ex]
7. A.M. Sirunyan et al., Pileup mitigation at CMS in 13 TeV data. JINST **15**(09), 09018 (2020). https://doi.org/10.1088/1748-0221/15/09/P09018. arXiv:2003.00503 [hep-ex]
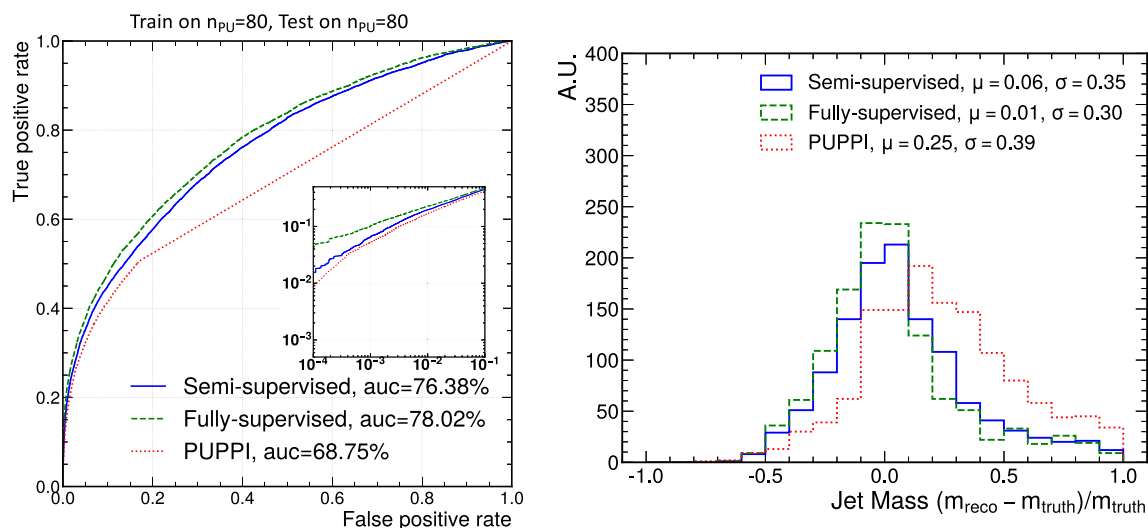

Fig. 11 The ROC curve for the gated GNN on neutral particles under $n_{PU} = 80$ for SSL, fully-supervised learning, and the domain PUPPI algorithm for simulations with resolution effect

8. A.M. Sirunyan et al., Performance of missing transverse momentum reconstruction in proton–proton collisions at $\sqrt{s} = 13$ TeV using the CMS detector. JINST **14**(07), 07004 (2019). https://doi.org/10.1088/1748-0221/14/07/P07004. arXiv:1903.06078 [hep-ex]

9. G. Aad, et al., Luminosity determination in *pp* collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector at the LHC. Report number ATLAS-CONF-2019-021 (2019). https://cds.cern.ch/record/2677054

10. A.M. Sirunyan et al., Precision luminosity measurement in proton–proton collisions at $\sqrt{s} = 13$ TeV in 2015 and 2016 at CMS. Eur. Phys. J. C **81**(9), 800 (2021). https://doi.org/10.1140/epjc/s10052-021-09538-2. arXiv:2104.01927 [hep-ex]

11. O. Brúning, L. Rossi, (eds.), *The High Luminosity Large Hadron Collider: the New Machine for Illuminating the Mysteries of Universe*, vol. 24. (2015). 10.1142/9581

12. M. Aaboud et al., Performance of the ATLAS track reconstruction algorithms in dense environments in LHC Run 2. Eur. Phys. J. C **77**(10), 673 (2017). https://doi.org/10.1140/epjc/s10052-017-5225-7. arXiv:1704.07983 [hep-ex]

13. S. Chatrchyan et al., Description and performance of track and primary-vertex reconstruction with the CMS tracker. JINST **9**(10), 10009 (2014). https://doi.org/10.1088/1748-0221/9/10/P10009. arXiv:1405.6569 [physics.ins-det]

14. A.M. Sirunyan et al., Particle-flow reconstruction and global event description with the CMS detector. JINST **12**(10), 10003 (2017). https://doi.org/10.1088/1748-0221/12/10/P10003. arXiv:1706.04965 [physics.ins-det]

15. M. Cacciari, J. Rojo, G.P. Salam, G. Soyez, Quantifying the performance of jet definitions for kinematic reconstruction at the LHC. JHEP **12**, 032 (2008). https://doi.org/10.1088/1126-6708/2008/12/032. arXiv:0810.1304 [hep-ph]

16. D. Krohn, M.D. Schwartz, M. Low, L.-T. Wang, Jet cleansing: pileup removal at high luminosity. Phys. Rev. D **90**(6), 065020 (2014). https://doi.org/10.1103/PhysRevD.90.065020. arXiv:1309.4777 [hep-ph]

17. M. Cacciari, G.P. Salam, G. Soyez, Use of charged-track information to subtract neutral pileup. Phys. Rev. D **92**(1), 014003 (2015). https://doi.org/10.1103/PhysRevD.92.014003. arXiv:1404.7353 [hep-ph]

18. M. Cacciari, G.P. Salam, G. Soyez, SoftKiller, a particle-level pileup removal method. Eur. Phys. J. C **75**(2), 59 (2015). https://doi.org/10.1140/epjc/s10052-015-3267-2. arXiv:1407.0408 [hep-ph]

19. P. Berta, M. Spousta, D.W. Miller, R. Leitner, Particle-level pileup subtraction for jets and jet shapes. JHEP **06**, 092 (2014). https://doi.org/10.1007/JHEP06(2014)092. arXiv:1403.3108 [hep-ex]

20. D. Bertolini, P. Harris, M. Low, N. Tran, Pileup per particle identification. JHEP **10**, 059 (2014). https://doi.org/10.1007/JHEP10(2014)059. arXiv:1407.6013 [hep-ph]

21. P.T. Komiske, E.M. Metodiev, B. Nachman, M.D. Schwartz, Pileup mitigation with machine learning (PUMML). JHEP **12**, 051 (2017). https://doi.org/10.1007/JHEP12(2017)051. arXiv:1707.08600 [hep-ph]

22. J. Arjona Martínez, O. Cerri, M. Pierini, M. Spiropulu, J.-R. Vlimant, Pileup mitigation at the Large Hadron Collider with graph neural networks. Eur. Phys. J. Plus **134**(7), 333 (2019). https://doi.org/10.1140/epjp/i2019-12710-3. arXiv:1810.07988 [hep-ph]

23. V. Mikuni, F. Canelli, ABCNet: an attention-based method for particle tagging. Eur. Phys. J. Plus **135**(6), 463 (2020). https://doi.org/10.1140/epjp/s13360-020-00497-3. arXiv:2001.05311 [physics.data-an]

24. B. Maier, S.M. Narayanan, G. de Castro, M. Goncharov, C. Paus, M. Schott, Pile-up mitigation using attention. Mach. Learn. Sci. Technol. **3**(2), 025012 (2022). https://doi.org/10.1088/2632-2153/ac7198. arXiv:2107.02779 [physics.ins-det]

25. Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks (2017). arXiv:1511.05493 [cs.LG]

26. S. Agostinelli et al., GEANT4—a simulation toolkit. Nucl. Instrum. Methods A **506**, 250–303 (2003). https://doi.org/10.1016/S0168-9002(03)01368-8

27. A.M. Sirunyan et al., Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques. JINST **15**(06), 06005 (2020). https://doi.org/10.1088/1748-0221/15/06/P06005. arXiv:2004.08262 [hep-ex]

28. M. Aaboud et al., Calibration of light-flavour *b*-jet mistagging rates using ATLAS proton-proton collision data at $\sqrt{s} = 13$ TeV. Report number ATLAS-CONF-2018-006 (2018). http://cds.cern.ch/record/2314418

29. J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, M. Selvaggi, DELPHES 3, a modular framework for fast simulation of a generic collider experiment. JHEP **02**, 057 (2014). https://doi.org/10.1007/JHEP02(2014)057. arXiv:1307.6346 [hep-ex]

30. Shlomi, J., Battaglia, P., Vlimant, J.-R.: Graph neural networks in particle physics (2020). https://doi.org/10.1088/2632-2153/abbf9a. arXiv:2007.13681 [hep-ex]

31. P.T. Komiske, E.M. Metodiev, J. Thaler, Energy flow networks: deep sets for particle jets. JHEP **01**, 121 (2019). https://doi.org/10.1007/JHEP01(2019)121. arXiv:1810.05165 [hep-ph]

32. H. Qu, L. Gouskos, ParticleNet: jet tagging via particle clouds. Phys. Rev. D **101**(5), 056019 (2020). https://doi.org/10.1103/PhysRevD.101.056019. arXiv:1902.08570 [hep-ph]

33. S.R. Qasim, J. Kieseler, Y. Iiyama, M. Pierini, Learning representations of irregular particle-detector geometry with distance-weighted graph networks. Eur. Phys. J. C **79**(7), 608 (2019). https://doi.org/10.1140/epjc/s10052-019-7113-9. arXiv:1902.07987 [physics.data-an]

34. X. Ju, et al., Graph neural networks for particle reconstruction in high energy physics detectors, in *33rd Annual Conference on Neural Information Processing Systems* (2020)

35. J. Pata, J. Duarte, J.-R. Vlimant, M. Pierini, M. Spiropulu, MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. Eur. Phys. J. C **81**(5), 381 (2021). https://doi.org/10.1140/epjc/s10052-021-09158-w. arXiv:2101.08578 [physics.data-an]

36. T. Sjöstrand, S. Ask, J.R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C.O. Rasmussen, P.Z. Skands, An introduction to PYTHIA 8.2. Comput. Phys. Commun. **191**, 159–177 (2015). https://doi.org/10.1016/j.cpc.2015.01.024. arXiv:1410.3012 [hep-ph]

37. R. Corke, T. Sjostrand, Interleaved parton showers and tuning prospects. JHEP **03**, 032 (2011). https://doi.org/10.1007/JHEP03(2011)032. arXiv:1011.1759 [hep-ph]

38. X. Zhu, *Semi-supervised learning with graphs* (Carnegie Mellon University, Pittsburgh, 2005)

39. T.N. Kipf , M. Welling, Semi-supervised classification with graph convolutional networks, in *International Conference on Learning Representations (ICLR)* (2017)

40. S. Fortunato, Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)

41. A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis. Phys. Rev. E **80**(5), 056117 (2009)

42. P. Li, I. Chien, O. Milenkovic, Optimizing generalized pagerank methods for seed-expansion community detection. in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2019)

43. W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in *NIPS* (2018)

44. Y. Li, R. Zemel, M. Brockschmidt, D. Tarlow, Gated graph sequence neural networks, in *Proceedings of ICLR'16*

(2016). https://www.microsoft.com/en-us/research/publication/gated-graph-sequence-neural-networks/

45. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539

46. M. Cacciari, G.P. Salam, G. Soyez, The anti-k_t jet clustering algorithm. JHEP **04**, 063 (2008). https://doi.org/10.1088/1126-6708/2008/04/063. arXiv:0802.1189 [hep-ph]